Thèse n° 8166

Unsupervised Visual Entity Abstraction towards 2D and 3D Compositional Models

Présentée le 25 novembre 2022

Faculté des sciences et techniques de l'ingénieur
Laboratoire de traitement des signaux 4
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Beril BESBINAR

Acceptée sur proposition du jury

Dr D. Gillet, président du jury
Prof. P. Frossard, directeur de thèse
Prof. C. De Vleeschouwer, rapporteur
Prof. A. Smolic, rapporteur
Prof. A. Alahi, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

Acknowledgements

I assume getting (or even targeting) a Ph.D. offers an authentic experience to each and every individual who braves it, but, I think the one that resulted in this thesis was quite particular for one's mental sanity. Changing between quite different topics multiple times, working on things at the wrong time, and consequently facing several mental blocks and breakdowns might sound familiar to whoever has at least attempted research, but, after all going "well" according to some norms, I still feel the impact it has had on me and I hope this scar will heal into some personal growth everyone talks about in their thesis. Some may call me a drama queen, and to be fair, I think I was indeed a bit of a grumpy drama queen in the last years, but I somehow managed to see the so-called finish line and write this piece of appreciation in the end with a degree that I don't believe in much, anymore. So, when I thank people here, with or without their names, I can assure you that it's sincere, it's genuine, and it's not because it's what I'm expected to do. The people I mention here have had a significant impact on my life in some way. So, this piece of text might be long, please bear with me because it's the most real part of this thesis.

Let me start with Pascal, again not because it's the way to do this, but for the fact that he is the person who made this thesis even possible. Thank you Pascal for believing in me more than you should have, and making me feel safe and secure when I was playing in my sand pool: making up problems and trying to solve them in weird ways. I remember thinking that having a Ph.D. advisor you enjoy having meetings with is at least as important as having research interests aligned, and reflecting on it today, it seems like I was right (and this is one of the rare things I was right about!). I felt like I could do things after our meetings, at least till recently (and that change is definitely on me). Thanks a lot for your trust and support throughout this long journey.

I am also very grateful for the great jury I had: Prof. Alexandre Alahi, Prof. Aljosa Smolic, Prof. Christophe De Vleeschouwer, and Dr. Denis Gillet: not only that they converted my most embarrassing presentation into a pleasant experience, but they also provided insightful feedback, and I truly enjoyed the research discussions we had during the exam.

As a long-lasting member of LTS4, I've seen different generations at LTS4, all of whom are nice, bright people; and I consider myself lucky to call some of them friends. It was a pleasure to meet you all guys; Stefano, Mattia, Francesca, Seyed, Renata, Mireille, Roberto, Guille, Apostolos, Clément, Isabela, Ortal, Nikos, Ahmet, Harshitha, Javier, Clémentine, Jelena, Ádám, Yamin, William, Arun, and of course, Dorina and Hermina. Before Covid happened, we had a very fun corridor, and it would not be fair if I don't name some of the nice people from the neighboring labs at that time: Andreas, Vlad, Rodrigo, Nikos, Nicolas, Baran, Vaggelis; the honorary member of the corridor, Kostas, and obviously the corridor girls: Helena, Irene, and Anne-flore (Hermina and Pinar will have their share later). I fondly remember the barbecues, hikes, ski trips, dinners, and game nights we had, as well as those spontaneous long weekends wherever we could find cheap tickets for, be it Berlin, Malaga, Fuerteventura, or some amusement park near Barcelona. Getting a bit out of the comfort of our lovely corridor, but still staying inside the EPFL periphery, I also want to mention a few other names who made Lausanne an even more liveable place for me: Jorge, Denys, Thibault, and Georg; I hope we can have more of those social days and nights somewhere, somehow. To be honest, it became even more clear how all these people made my daily life enjoyable after many of them left Lausanne, even Switzerland. But, in the end, I can certainly say that every cloud has a silver lining, indeed: this Ph.D. at least made me meet a handful of awesome people and I'm extremely grateful for that!

Some special thanks go to Helena, for making me believe that a flat-share can be actually fun; I miss our tea-time conversations. And my latest officemate, Dorina; thanks for trusting in me this much, encouraging me every time I was down, and putting up with my grumpy face on those few days I made it to the office. I've learned a lot from you about being a good researcher, and a very kind person at the same time.

Talking about offices, Hermina, my dearest ex-officemate (as much as we both hate the ex-there and dream about sharing the office again somewhere, somehow): I hope you know how difficult it is for me to describe anything about that awesome, crazy, incredible experience we had in the office. We always talked about how embarrassing and fun it'd be if someone had implemented a hidden camera in the office, but words really cannot describe what happened inside those four walls (and I'm happy that they cannot) and nothing would do justice to those memories. Frankly, things did not stay there (I was a bit afraid about that at some point because of her great remote communication skills) and I'll just say that I'm looking forward to the days to come: both of us stress-free but still as "coo-coo" as ever!

And here it comes, the so-called Turkish gang, the family far from home. We somehow found each other while not fancying a Turkish-only circle of friends, but the particular sense of humor this group embraced is worth whatever it takes! These were the people I asked for when I opened my eyes in the hospital after a near-death experience, and no one else could have made me laugh that hard in such a condition. To honor their names here; thank you Alp for all the hugs, which were always very much needed (and those are horribly missed these days), Ahmet for getting me into (at least trying) powerlifting, Ali for making me feel like a chill person after every time I talk to you, Ceyhun for all the joy and laughter, Okan for demonstrating how to be the brave one, and the last but not the least, Pınar for changing my perspective about friendship. Regarding that last point, I always felt privileged to have met some incredible people in my younger years and to consider them lifelong friends, but I also thought that this was some kind of a curse because it was even more difficult to be good friends with new people as I get older. You've proven me wrong Pınar, despite all our differences, you've been one of the best things about this Ph.D. journey. You are one such nice storyteller and I admire the part of your journey that I could witness and the person you've become; but above all, you are a true friend, kind and caring, strong and smart. And I believe, some decades from now, we will be still talking about these good old days, however apart life drifts us.

Talking about lifelong friends, I, once more, want to thank Caner and Lalu (a.k.a Esra) for always being there with no judgments. Also, Muscan, for being the best travel buddy ever, and making the Netherlands so close to Switzerland. There's yet so much to explore together, places to visit, and things to try, and I know we'll finally get to do some of them very soon! Also, thank you Leyla for the always-comforting smile or voice, whenever we could get a hold of each other (somehow Zurich or Basel was never that close).

I also want to mention some of my ex-colleagues who became friends right after; the existential discussions we have at random times, about life, research, and about friendships have always lightened the load and reminded me that I'm not alone. Yağız, Emin, and Akın, I truly miss our reunions when we all used to live in continental Europe and I'm looking forward to the next one (hoping it to be sooner than I can think of). And Emrecan, thank you for just being the person you are, a bright, kind friend.

There is one person who suffered the most after me because of this thesis, and I cannot thank him enough for putting up with me, as I have been working late almost all the time, not being with him during so many weekends and holidays, and being a bit grumpy recently. Yet, we still got to enjoy some incredible road trips together and explored cities and the mountains whenever we could. Thank you, Jan, for being on my side, whether it meant telling me I needed to sleep or taking the first steps on meter-deep snow so that I could follow in your footsteps. It was your companionship that made me get out of my bed and continue sometimes, and I really appreciate that you stuck with me through all my self-doubt and difficulty although you never understood why I was "doing this to myself".

Finally, the biggest gratitude goes to my amazing big family, for their unconditional love and support. Being a first-gen Ph.D., I am not sure if I could ever explain to them what I was doing all these years abroad; but no matter what, they've always stood by me, believed in me, and accepted me as I am. Mom, as I always say, who is the strongest woman that I've ever known, thank you for everything you taught me and you sacrificed for me, you are the one true reason why I exist today as I am, seni çok seviyorum!

Britannia Hütte, Saas Fee, *22 August 2022* B.B.

Abstract

Object-centric learning has gained significant attention over the last years as it can serve as a powerful tool to analyze complex scenes as a composition of simpler entities. Well-established tasks in computer vision, such as object detection or instance segmentation, are generally posed in supervised settings. The recent surge of fully-unsupervised approaches for entity abstraction, which often tackle the problem with generative modeling or self-supervised learning, indicates the rising interest in structured representations in the form of objects or object parts. Indeed, these can provide benefits to many challenging tasks in visual analysis, reasoning, forecasting, and planning, and provide a path for combinatorial generalization. In this thesis, we exploit different consistency constraints for disambiguating entities in fully-unsupervised settings. We first consider videos and infer entities that can be modeled by consistent motion between frames at different time steps. We unconventionally opt for representing objects with amodal masks and investigate methods to accumulate information about each entity throughout time for an occlusion-aware decomposition. Approximating motion with parametric spatial transformations enables us to impose cyclic long-term consistency that contributes to reasoning about unseen parts of entities. We then develop a video prediction model based on this decomposition scheme. As the proposed decomposition decouples motion from entity appearance, we attribute the inherent stochasticity of the video prediction problem to our parametric motion model and propose a three-stage training scheme for more plausible prediction outcomes. After deterministic decomposition at the first stage, we train our new model for short-term prediction in stochastic settings. Long-term prediction as the last step helps us learn the distribution of motion present in the dataset for each entity. Finally, we focus on multi-view image settings, and assume two different arrangements where the scene is observed from different viewpoints in both cases. We attempt to find correspondences of the volumetric representations of those observations that are guided by differentiable rendering algorithms. By grouping the volume units based on consistent matching of features, we partition the volumetric representation that leads to the individual rendering of each inferred entity. We present promising outcomes for all of the proposed unsupervised object-representation schemes on synthetic datasets and present different ideas for scaling them up for the adaptation to real-world data as future work.

Résumé

L'apprentissage centré sur l'objet a suscité beaucoup d'intérêt ces dernières années, car il peut servir d'outil puissant pour analyser des scènes complexes comme une composition d'entités plus simples. Des tâches bien établies en vision par ordinateur, telles que la détection d'objets ou la segmentation d'instances, sont généralement posées dans un cadre supervisé. L'essor récent des approches entièrement non supervisées pour l'abstraction d'entités, qui s'attaquent principalement au problème par la modélisation générative ou l'apprentissage auto-supervisé, indique l'intérêt croissant pour les représentations structurées sous forme d'objets ou de parties d'objets. En effet, celles-ci peuvent apporter des avantages à de nombreuses tâches difficiles en matière d'analyse visuelle, de raisonnement, de prévision et de planification, et fournir une voie pour la généralisation combinatoire. Dans cette thèse, nous exploitons différentes contraintes de cohérence pour désambiguïser des entités dans des contextes entièrement non supervisés. Nous considérons d'abord les vidéos et déduisons les entités qui peuvent être modélisées par un mouvement cohérent entre les images à différents pas de temps. Nous choisissons de manière non conventionnelle de représenter les objets avec des masques amodaux et nous étudions des méthodes permettant d'accumuler des informations sur chaque entité au fil du temps pour une décomposition tenant compte des occlusions. L'approximation du mouvement par des transformations spatiales paramétriques nous permet d'imposer une cohérence cyclique à long terme qui contribue au raisonnement sur les parties non vues des entités. Nous développons ensuite un modèle de prédiction vidéo basé sur ce schéma de décomposition. Comme la décomposition proposée dissocie le mouvement de l'apparence de l'entité, nous attribuons la stochasticité inhérente au problème de prédiction vidéo à notre modèle de mouvement paramétrique et proposons un schéma d'apprentissage en trois étapes pour des résultats de prédiction plus plausibles. Après une première étape de décomposition déterministe, nous formons notre nouveau modèle pour la prédiction à court terme dans un contexte stochastique. La prédiction à long terme, qui constitue la dernière étape, nous aide à apprendre la distribution du mouvement présent dans l'ensemble de données pour chaque entité. Enfin, nous nous concentrons sur les paramètres d'images multi-vues, et nous supposons deux arrangements différents où la scène est observée depuis différents points de vue dans les deux cas. Nous tentons de trouver des correspondances entre les représentations volumétriques de ces observations qui sont guidées par des algorithmes de rendu différentiables. En regroupant les unités de volume sur la base d'une correspondance cohérente des caractéristiques, nous partitionnons la représentation volumétrique qui conduit au rendu individuel de chaque entité inférée. Nous présentons des résultats prometteurs pour tous les schémas de représentation d'objets non supervisés proposés sur des ensembles de données synthétiques et nous présentons différentes idées pour les mettre à l'échelle afin de les adapter aux données du monde réel dans le cadre de travaux futurs.

Contents

Acknowledgements **i**

Abstract (English/Français) **v**

1 Introduction **1**

1.1 Visual Entity Abstraction 1
1.2 Thesis Outline 4
1.3 Summary of Contributions 7

2 Related Work **9**

2.1 Visual Entity Abstraction and Layered Representations 9
2.2 Video Generation and Prediction 11
  2.2.1 Video Generation 11
  2.2.2 Video Prediction 12
2.3 Object Representations in 3D 16

3 Continuous Motion and Consistency Objective **17**

3.1 Introduction 17
3.2 Abstraction of Moving Objects 18
  3.2.1 Problem Formulation 18
  3.2.2 Amodal Entity Masks 21
  3.2.3 Occlusion Handling and Inpainting 23
  3.2.4 Motion Modelling 25
  3.2.5 Prediction 26
3.3 Learning 27
3.4 Implementation 30
3.5 Experimental Results 33
  3.5.1 Datasets 33
  3.5.2 Results 35
3.6 Conclusion 43

4 Object-centric Video Prediction **45**

4.1 Introduction 45
4.2 Background 47

4.2.1 Variational Autoencoders 48
4.2.2 Wasserstein GANs 49
4.3 Proposed Approach 50
4.3.1 Object-centric Video Prediction 51
4.3.2 VAE Formulation 51
4.3.3 WGAN Formulation 55
4.4 Implementation 56
4.4.1 Model Implementation 56
4.4.2 Multi-stage Training 57
4.5 Experimental Results 58
4.5.1 Dataset 58
4.5.2 Results 59
4.6 Conclusion 62

5 Discrete Motion in Multi-View Settings 65
5.1 Introduction 65
5.2 Related Work 67
5.2.1 3D Representations 68
5.2.2 Neural Rendering 68
5.2.3 Compositionality in Neural Rendering 70
5.3 Proposed Method 72
5.3.1 Background - Neural Radiance Fields 72
5.3.2 Problem Formulation and Notations 74
5.3.3 Lifting and Fusing 2D Features. 75
5.3.4 Radiance Fields with Voxel Features 76
5.3.5 Feature Correspondance 78
5.3.6 Entity Abstraction 80
5.4 Implementation 81
5.5 Experimental Results 85
5.5.1 Dataset 85
5.5.2 Results 86
5.6 Conclusion 93

6 Conclusion 97
6.1 Summary 97
6.2 Future Directions 99

A Evaluation Metrics 101
A.1 Image Reconstruction 101
A.2 Object Abstraction 101

B Derivation of Lower Bound for Stochastic Video Prediction Model 103

Bibliography **122**

Curriculum Vitae **123**

1 Introduction

1.1 Visual Entity Abstraction

Deep learning has become the *de facto* framework for the vast majority of tasks in computer vision and pattern recognition (LeCun et al., 2015; Krizhevsky et al., 2012; Ronneberger et al., 2015; Long et al., 2015; Redmon et al., 2016; Bertinetto et al., 2016; He et al., 2017; Qi et al., 2017; Mildenhall et al., 2020), with proven success in supervised tasks. However, there are known issues with these mostly well-performing artificial neural networks, including but not limited to, adversarial examples (Goodfellow et al., 2014b), the limited ability of out-of-distribution generalization (Nguyen et al., 2015) or the lack of modularity (Sabour et al., 2017) at its most common forms. An alternative approach to remedy some of these bottlenecks is building compositional models of the world that are grounded in physics and psychology (Lake et al., 2017). Scene understanding is one such task in computer vision that can be tackled in a compositional way with self-supervised methods towards this objective rather than fully data-driven models trained with manual annotations.

The structure of natural visual scenes is often profoundly rich. Scene understanding thus consists in discovering its building elements, like objects or planes, capturing the interactions and relationships between those, and modeling dynamics if observations are available for different time instants or different viewpoints. With the non-modular state-of-the-art models proposed to date, these tasks remain a challenging research area. One possible solution for improving the limitations of modern neural networks and developing scene understanding and visual reasoning is to introduce abstraction or structured representations centered around visual perception aiming towards combinatorial generalization. We claim that it can be possibly achieved by combining complementary strengths of "hand-crafted" and "end-to-end" approaches rather than forcing a discrete choice between the two. In other words, rather than feeding ample amounts of data to today's modern deep architectures with enormous capacity, one can design deep neural network models with the inductive biases developed over the years for processing pipelines with hand-crafted features. Such inductive bias can be used to represent objects or object parts in images to facilitate the analysis of a given scene. Prior work (Diuk et al., 2008; Battaglia et al., 2018; Bapst et al., 2019) has indeed showcased how relational reasoning and control problems can benefit from object-oriented representations. However, the definition of objects and methods for structured representations remains a greatly open question (Alexe et al., 2010).

Although it might seem oblivious at first sight, we, humans, are proficient at disambiguating *objects* (or more generally meaningful parts), when we observe a scene, even if this one consists of objects we have not seen before. There is a vast literature in psychology and neuroscience to understand this phenomenon. One recognized hypothesis in psychology is known as Gestalt laws of perception (Koffka, 2013), which aims to explain how principles of grouping in human perception can be enclosed over a set of low-level cues. These rules are usually organized into five categories, namely, *similarity, proximity, continuity, enclosure* and *common fate*. In essence, they indicate that things that are similar and close to each other, or things that move together, are more likely to "belong together". This means that they can be grouped together, and our brain will make them appear as simple as possible in a way that they form some entity (Wagemans et al., 2012). Another approach to studying human cognition defines four core knowledge systems for building mental representations of *objects*, *persons*, *spatial relationships* and *numerosity*, based on studies on infants (Spelke and Kinzler, 2007). Among these core systems, the one for object representation that focuses on the *cohesion*, *continuity* and *contact* of objects seems to have been studied most extensively (Aguiar and Baillargeon, 1999; Leslie and Keeble, 1987; Spelke, 1990). One interesting finding is that the ability to perceive object boundaries and complete shapes of objects that move out of sight is present even without any visual experience in human infants or chicks (Valenza et al., 2006; Lea et al., 1996). On the other hand, a similar phenomenon cannot be observed for inanimate objects even after months of visual experience (Huntley-Fenner et al., 2002). This emphasizes the importance of motion for the abstraction of entities in human perception. Overall, there is no single accepted hypothesis for the perceptual grouping in our visual system, but one can argue that it is mostly holistic, meaning that a scene is perceived as a sum of its components, and it is likely to require a recurrence to attain the ability to process low-level features with attention. Moreover, it is not class-specific, we do not need previous exposure to a particular type of object to be able to disambiguate it; and it is easier to do so, almost granted, if that object is subject to some motion.

In contrast, most of the established tasks in computer vision that are related to objects are initially posed as supervised problems and hardly benefit from this body of work in human perception. For example, we can list object detection, which calls for a bounding box and a class label describing the object inside the predicted bounding box for *all* the objects in an image; semantic segmentation, which requires per-pixel class assignments for a set of predetermined labels; or instance segmentation, which differentiates between individual instances of a class by per-pixel label assignment while possibly discarding a subset of pixels. And all of these tasks are tackled in supervised, and very often, class-dependent settings. Their video counterpart, namely video object segmentation, aims to partition video frames into multiple segments that might correspond to objects or object parts. This problem is addressed both in supervised and unsupervised settings. Unsupervised video object segmentation mostly refers to the testing scenario where there is no human input to the system, yet the training is done with labeled datasets. Labeling is, however, extremely expensive, particularly for video datasets. And supervision in many forms contradicts the ultimate purpose of structured generalization as it means that segmentation methods might become confined to representing objects of classes seen during training. Hence, the lack of adaptivity might limit the success of such methods in the long term.

There is, however, a recent trend in the machine learning community to take inspiration from the human visual system for fully-unsupervised methods toward object-oriented models. This is mostly achieved by generative models or self-supervised learning. These methods can operate on still images, where the main objective consists of grouping pixels based mostly on *proximity* and *similarity*. In videos, they can additionally incorporate the inductive bias of *common fate* to abstract objects or object parts. Independently of the type of dataset, the recent object-oriented methods are often categorized into two groups, those that are based on local attention and mostly represent objects with bounding boxes (Eslami et al., 2016; Crawford and Pineau, 2019), and the others that represent the data as scene mixture models where the decomposed modules define the appearance of entities together with a per pixel assignment mask (Greff et al., 2017; Van Steenkiste et al., 2018; Greff et al., 2019; Locatello et al., 2020). One common aspect of these two categories, though, is the fact that both of them operate as inverse graphics based on representation learning with the purpose of "analysis by synthesis". Indeed, the input frame is first encoded into a set of latent variables, which are then decoded into individual entities for re-composing the input frame, or possibly for predicting a frame in the future in the video case. An important design pattern then emerges regarding the dynamics of decomposition of the latent variables representing *slots*, which often refer to meaningful parts composing the observed scene. This design can follow different models:

- Universal slots, where the input frame is encoded globally into permutation invariant entity representations (Greff et al., 2019; Locatello et al., 2020),
- Sequential slots, which indicates the sequential decomposition of the input frame into slot representations that only use the residual of the input frame after the inference of a previous slot; in other words, these are methods that attend to a single slot at a time (Veerapaneni et al., 2020; Zablotskaia et al., 2021),
- Spatial slots that first partition the input into spatial tiles and focus on those tiles independently (Jiang et al., 2019), and,
- Category slots, such as capsules, where each slot is attributed to a category from a set of predefined objects or object parts (Sabour et al., 2017).

Figure 1.1 illustrates the different design patterns for entity abstraction.

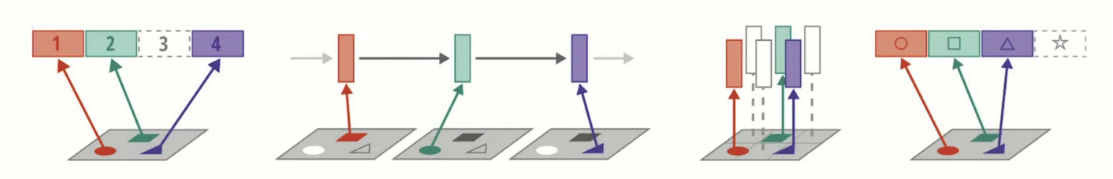


Figure 1.1: Design patterns for entity abstraction representing: (a) universal (b) sequential (c) spatial (d) categorical slots. Image taken from Greff (2020).

In this thesis, we follow a global processing model similar to universal slots. We opt for a fully unsupervised approach to decompose input frames into individual entities, which exhibit some notion of consistency between different perspectives of the same scene. These different perspectives are tightly coupled to motion. For example, they can be different snapshots in time if the entities are moving; or they can be different viewpoints if the entities are *mostly* static.

For the first part of the thesis, presented in Chapters 3-4, our objective is to learn masks to describe *moving* entities, and use the input frame to infer object appearances with masks while taking occlusions into account. As we do not follow the common encoding-decoding approach, the learning objective is modified from reconstruction to prediction in Chapter 3. We later extend this model for object-centric video prediction in Chapter 4, where the prediction is formulated as sampling parametric motion for each inferred entity from a time-dependent conditional distribution. In the second part of the thesis, presented in Chapter 5, we consider two configurations of a scene consisting of the same entities seen from multiple different viewpoints and opt for a feature volume representation. We aim to find correspondences between two feature volumes for voxel grouping to account for the changes in these two different configurations. We detail the structure of the thesis in the next section.

1.2 Thesis Outline

One way to describe the notion of representation is to state it as a formal system that "makes certain entities and types of information explicit" towards achieving a task that manipulates the information (Marr, 2010). Various types of representation hence differ fundamentally in what information they choose to make explicit. Traditional methods in computer vision mostly engage in representations that correspond to interpretable entities, whereas modern deep learning techniques tend to operate in high-dimensional spaces that are harder to understand and explain. Object-oriented methods, in general, appear to be somewhere in between. They represent entities with *latent variables* that are frequently accompanied by a pixel mask, which is somewhat similar to an indicator function. They decode these latent representations back to pixel values to account for the appearance of the corresponding visual entity. These representations are more and more commonly referred to as *slot representations* (Locatello et al., 2020). However, this kind of *autoencoding* initially resulted in blurry outcomes, particularly in the case of textured entities (Finn et al., 2016) and they can be harder to scale up to more sophisticated scenes at higher spatial resolutions. We suggest that one way to mitigate the drawbacks of autoencoding the visual appearance for abstracting entities is to represent them via masks that segment the input image or video frame. The masks can operate as indicator functions to infer the appearance of each visual entity, but, such formulation would require the definition of a task different from the reconstruction. Similarly, video representation poses the same challenge with the addition of time dimension, which also constrains the autoencoding approach with *slowly-varying* representations for each entity. However, the per-frame reconstruction error averaged over time might as the training objective leads to averaged representations resulting in blurry reconstructions (Mathieu et al., 2015). In contrast, the masking strategy is less likely to produce blurry results, yet, inferring masks that indicate entities in a time-consistent manner is not trivial given that the reconstruction objective is no more plausible.

The initial motivation for an entity abstraction approach in this thesis emerged from the idea that disambiguating entities that *move* in a given sequence would lead to more physically plausible video prediction outcomes as we can individually warp entities to future time steps and compose them in an occlusion aware manner, rather than averaging all possible outcomes in pixel space for a given dataset. Although simplistic in motivation, such a pipeline would come with its own challenges. First and foremost, without supervision, decoupling "objects" is an ill-posed problem. We first try to tackle that by narrowing our search space to *moving objects* in videos, which we believe is a better-posed problem. However, by construction, per-frame processing becomes unattainable as we seek cues for motion. By multi-frame processing, we aim for entity masks that would dissect an input frame into different entities. As any such segmentation can easily lead to perfect reconstruction, the per-frame reconstruction objective is replaced with object-centric prediction in videos. For the prediction task, on the other hand, one needs to take occlusions into account, inpaint previously unseen parts of an object when warping it to any time step in the future, and predict motion based on the content and ongoing motion.

In the first part of the thesis, we investigate different ideas for achieving this segregation task with amodal masks, as amodal masks help with occlusion-aware composition and they behave as an indicator map for inpainting. In essence, we show that one can infer such masks based on low-level cues induced by convolutional neural networks (Krizhevsky et al., 2012), or their explicit similarity, and continue tracking them with some notion of memory in the event of occlusions. We consider different ways of composition in those events based on the inferred amodal masks and experiment with different auxiliary objectives to constrain the problem, such as enforcing the masks for entities to be spatially connected or sparse or introducing a cyclic long-term prediction objective to reinforce the amodal mask approach. As the field is relatively young, there are few different datasets used for reporting performance in unsupervised object-oriented learning, and the uniformity is very limited. Thus, we first present our results on our own dataset that is composed of simple sprites, similar to available datasets, but with more events of occlusion to build challenging conditions. We then compare our results to a recent benchmark that investigates the performance of the state-of-the-art methods (Weis et al., 2020). We show that our novel method is capable of inferring entity masks for moving objects as a virtue of the proposed prediction objective in contrast to existing methods trained with reconstruction tasks.

In the following part, we develop an object-centric video prediction model based on our decomposition framework. Video prediction can be tackled by identifying variables of change and only predicting the future of those variables. With an object-oriented approach, we can claim that object appearances are less likely to change during the short windows of time, and change emerges from how they move and interact. Hence, we simplify the video prediction problem by attributing the scene dynamics only to the motion of inferred entities. The parameters of motion for each entity, hence, are treated as random variables; and the video prediction problem boils down to the estimation of their distribution and sampling from the estimated distribution when we need to predict a few next frames given a short video sequence. We suggest that the motion parameters for each entity should vary in a way to ensure continuity. Hence, we opt for a time-variant distribution and condition the motion parameters on their values in the past time steps. In addition, we claim that only a subspace of motion would be suitable for each entity. For example, elongation is usually not suitable for human or animal body parts, or any rigid man-made objects. Hence, we also condition the motion parameters on the appearance of the corresponding entity at the given time step. In stochastic settings, where the parametric motion is defined by random variables, this leads to certain dependencies for learning the distributions of plausible motion. We model our assumptions with a graphical model and experiment with the two most common generative approaches for approximating the posterior distribution of these variables. For the first one, we principle a method based on the dependencies of the latent variables for maximizing the likelihood of the observed data points. Secondly, we try an adversarial training scheme for video prediction by treating it as a conditional generation problem. To observe the advantages and bottlenecks of these two different approaches, we generate sequences where objects may adhere to different motion patterns. We keep the deterministic decomposition steps for entity abstraction and present a comparative study for the prediction outcomes generated by these two approaches. Results indicate that content-bases motion is a plausible assumption, and our pipeline can produce multi-step prediction outcomes while learning the distribution of the motion patterns to some extent. However, stochastic object-centric video prediction remains a challenging problem related to the well-known problem of mode-collapse faced by popular generative models.

In the final part, we change the scenario to a more static setting with the intuition that entities can often be observed *relocated* in a given, static scene at two different times rather than moving continuously without any external manipulations. In other words, it is likely to observe static scenes, which refers to static objects, at discrete times, and the objects might be relocated between those observations. As discrete snapshots in time would result in very limited information for interactions between objects, such as occlusions, we assume multi-view observations for each time step and target the decoupling of entities that can be represented by the same motion in 3D. For this purpose, we first adopt the recently proposed differentiable rendering method, Neural Radiance Fields(NeRF) (Mildenhall et al., 2020) to infer the 3D structure of a given scene from multiple 2D observations. NeRF is proposed to represent a single scene by over-fitting a function to map 3D coordinates to color and density. We first re-formulate it to operate on a discretized, bounded 3D world for multiple arrangements of the same scene, which requires generalization to represent "different" scenes with the same model. We then solve a correspondence problem for the discretized units for which the density value for rendering is high and group them according to inferred correspondences to account for entities. Despite the great interest in 3D representations, particularly NeRF and its variants, fully unsupervised methods for entity abstraction in 3D is very limited, hence, we generate a small dataset to showcase the performance of our proposal. Our results demonstrate that neural rendering approaches can be used for learning features for a hybrid 3D-representation schemes, which enables unsupervised reasoning directly in three dimensions.

1.3 Summary of Contributions

In this thesis, we take a novel approach to represent objects in 2D and 3D without any labels. The main contributions of this thesis are summarized as follows:

- We propose a novel object-centric representation of videos by segregating frames with masks that are derived from amodal representations. Contrary to the methods in the literature, our model explicitly represents both visible and occluded parts of each entity and does not require any iterative processing steps. Training of the proposed model is guided by short- and long-term prediction objectives, and supported by auxiliary losses which impose geometrically plausible entity masks. Results on simulated datasets validate that our approach is not only capable of abstracting entities based on consistent motion, but it also produces promising single-frame predictions.

- We principle on object-centric stochastic video prediction framework by attributing the stochasticity of the problem to the motion of decoupled moving entities. To the best of our knowledge, it is the first approach to bridge the gap between object-centric and stochastic video prediction algorithms. We experiment with two popular generative models for the video prediction problem under conditional-generation settings and propose a structured algorithm to estimate the distribution of motion observed in a given dataset. With a comparative study, we demonstrate the bottlenecks of each generative approach for this challenging problem.

- We formulate a problem of unsupervised entity abstraction in 3D that outputs object representations that can be rendered from any viewpoint. We adopt recent neural rendering techniques for a hybrid 3D representation, *i.e.*, we represent each scene as a combination of a feature volume and a scene-agnostic rendering function, where the latter guides learning the first one. Such volumetric representations enable an unsupervised decomposition algorithm based on 3D correspondences of the feature volumes when the objects are relocated. To the best of our knowledge, our method is the first attempt to abstract entities based on a formulation that is directly in three dimensions.

2 Related Work

In this chapter, we present a brief summary of the literature on visual entity abstraction and video prediction followed by a short coverage of object representations in 3D.

2.1 Visual Entity Abstraction and Layered Representations

The idea of representing videos as moving layers dates back to the 1990s (Wang and Adelson, 1993) and it has been of consistent interest to date for computer vision researchers (Jojic and Frey, 2001; Jepson et al., 2002; Ye et al., 2022). The objective consists in decomposing each video frame into persistent layers that smoothly transform over time. This line of approaches achieves the decomposition for each video independently, i.e., their optimized models do not generalize to other videos. The problem is heavily ill-posed, yet the recent work (Ye et al., 2022) demonstrates how one can successfully obtain a decomposition by inferring a sprite, i.e., a canonical texture image, for describing each group individually. Each group is eventually transformed into each frame with non-rigid geometric transformations for composition, after being masked by their jointly inferred masks. However, the model needs to be trained for each sequence separately, which heavily limits its practical use in its current form.

Recently, we witness a more discretized form of layered representations commonly referred to as (co-)part segmentation, which mostly involves either a group of images of a specific category or two frames of a video. These methods investigate the part structure in order to obtain meaningful intermediate representations of articulated objects. It has been initially formulated in supervised settings (Bourdev and Malik, 2009; Branson et al., 2011; Azizpour and Laptev, 2012) and has been recently addressed in unsupervised settings (Hung et al., 2019; Siarohin et al., 2021; Gao et al., 2021a; Liu et al., 2021). Supervision, when applied, is mostly guided by class-dependent annotations, hence, supervised methods struggle to generalize across different classes. On the other hand, formulating the problem in an unsupervised fashion is extremely challenging by a plain reconstruction objective emerging from the composition of inferred parts. Some additional constraints to decrease the complexity of the problem can be exemplified by semantic consistency (Hung et al., 2019; Siarohin et al., 2021), geometric concentration or compactness (Hung et al., 2019; Gao et al., 2021a; Liu et al., 2021) or motion consistency (Siarohin et al., 2021; Gao et al., 2021a). The variety of constraints, in particular for unsupervised settings, has been surely influential in other applications; however, the class-dependent implementations remain a bottleneck to be addressed.

The fully-unsupervised, class-agnostic approaches for structured representations that can be used for multiple instances of the input data, are relatively recent. These methods take inspiration from human perception and try to mimic the pathways used for understanding complex scenes with previously unseen objects as a sum of simpler and meaningful parts. Among the nominal works, AIR (Eslami et al., 2016) attempts to understand a scene with no supervision by computing object poses, i.e., 2D or 3D bounding boxes, and treating inference as an iterative process, one object at a time, which is a typical example for sequential slots presented in Figure 1.1. The spatially-invariant extension, SPAIR (Crawford and Pineau, 2019), tries to scale AIR-like approaches with fully convolutional representations for the same bounding box target. SQAIR (Kosiorek et al., 2018) extends AIR for sequential data where the non-convolutional latent variables are inferred one at a time for a single frame and used for conditioning the corresponding latent in the next time step. Another method that attends objects sequentially, TBA (He et al., 2019) is a deterministic approach, which positions itself as a reprioritized attentive tracking method that uses spatial transformers (Jaderberg et al., 2015) for motion modeling. More recently, MONet (Burgess et al., 2019) proposes to use the same iterative approach of processing one object at a time towards obtaining attention masks to compose a given image as the masked sum of decoded representations. GENESIS (Engelcke et al., 2019) represents the scene as a Gaussian mixture model where the mixing probabilities are considered as spatial attention masks, and they are implemented with an autoregressive prior, corresponding to the same iterative processing of a given frame. MONet and GENESIS might look very similar at a first glance, but they differ fundamentally in their principles: while GENESIS is a pure generative model that attempts to infer the data distribution with structured representations, MONet is a deterministic inference scheme given an input frame. A concurrent work, IODINE (Greff et al., 2019), also models the scene as a mixture model with similar spatial attention masks; however, it opts for global processing of the given frame, exemplifying global slots illustrated in Figure1.1. On the other hand, the inference scheme consists in iterative refinement, which can be computationally expensive. NEM (Greff et al., 2017) and its sequential and relational extensions (Van Steenkiste et al., 2018) can be considered as more elementary versions of IODINE, where the inference is achieved by expectation maximization via recurrent neural networks. PROVIDE (Zablotskaia et al., 2021) improves upon IODINE by extending the iterative amortized inference onto 2D spatio-temporal space. Slot Attention (Locatello et al., 2020) also proposes an iterative procedure for decomposing an input image into visually similar parts using self-attention (Vaswani et al., 2017), which is similar to the well-known clustering algorithm k-means, in essence. Slot Attention is also very recently extended to videos where prediction of motion is used as the main training objective, hence optical flow is used as an additional input for training (Kipf et al., 2022). On the other hand, PARTS (Zoran et al., 2021) integrates the slot attention mechanism into iterative amortized inference to improve the refinement step. Another example that opts for spatial attentive masks and operates with global slots is OP3 (Veerapaneni et al., 2020), which abstracts entities in an unsupervised manner and additionally uses them for prediction and planning. The main difference between OP3 and IODINE is that OP3 models sequential data with dynamic latent variables whereas IODINE is originally formulated for images. In a nutshell, all these methods attack a challenging and ill-posed problem, yet succeed to demonstrate encouraging results. On the other hand, they mostly rely on the reconstruction objective, which may result in blurry outcomes as discussed earlier. One can say that the iterative process is another bottleneck for this group of work, whether it is used for processing one object at a time or adopted for refinement of the objects that are inferred in parallel.

Another group of recent approaches like SILOT (Crawford and Pineau, 2020), STOVE (Kossen et al., 2019), SCALOR (Jiang et al., 2020), and G-SWM (Lin et al., 2020a) addresses the same problem by explicitly modeling entity representations by interpretable attributes. They mostly differ in the way they factorize the entity latent variables, hence the way they model the dependencies as well as in the method used for inferring the corresponding distributions. SILOT is similar to SQAIR with spatially invariant computations and factorizes the object latent variable into attributes like "where", "what", "depth" and "presence" to account for all the objects that are visible, or invisible, at a given time step. SCALOR, from another perspective, uses a proposal rejection mechanism for the same problem. STOVE also factorizes the latent variable into similar object attributes, and additionally uses a graph neural network for explicitly modeling the relations between multiple objects. G-SWM claims to develop a unified model using the key strengths of all aforementioned approaches in this group. All SILOT, SCALOR, and STOVE opt for bounding box representations as descendants of AIR, which unfortunately limits the representation power for more complex objects.

To unify the evaluation protocol for such (sequential) models, Weis et al. (2020) recently proposed a benchmarking scheme to investigate how different methods compare regarding their ability to detect, segment, and track individual objects. They sample the methods across different approaches and evaluate MONet (which they also extend as ViMON for sequences), TBA, and OP3 as representatives of spatial attention methods, spatial attention methods with factored latent, and spatial mixture models, respectively.

Our representation model presented in Chapter 3, in principle, is similar to spatial mixture models as we prefer masks over bounding boxes, which results in more descriptive representations. However, it is not a generative model and compositional reconstruction is not the ultimate training objective; what we propose is more of a deterministic decomposition based on *moving* objects and supported by cyclic-long-term prediction. There is no iterative process at any point in our method in contrast to the majority of approaches in the literature, which makes training and inference more efficient. Our method can be also considered as a class-agnostic co-part segmentation in sequences with continuity assumption.

2.2 Video Generation and Prediction

2.2.1 Video Generation

Generative models have been considered as a tool for learning representations in an unsupervised manner and disentangling the underlying reasons for variation in a given dataset. The nominal works of Generative Adversarial Networks (GANs) (Goodfellow et al., 2014a) and Variational Autoencoders (VAEs) (Kingma and Welling, 2013) have led to an ample amount of methods for generating images, and we have observed very rapid progress from low-resolution outcomes with visually perceptible artifacts to very high resolution, semantically meaningful computer-generated images in the last years. Video generation is usually considered to be more challenging due to higher dimensionality and difficulty in generating physically plausible continuation of events. Hence, progress is rather limited compared to image generation. One fundamental approach to facilitate video generation is to disentangle content and motion (Hsu et al., 2017; Tulyakov et al., 2018; Yingzhen and Mandt, 2018; Wang et al., 2020; Zhu et al., 2020; Han et al., 2021) by assuming a static content latent vector and a time-varying motion counterpart.

Even under more constrained environments, such as video-to-video synthesis, where the output video is mapped from an input source video rather than a noise vector as in generative modeling, the outcomes are prone to artifacts like flickering due to extreme high dimensionality of the problem (Wang et al., 2018; Mallya et al., 2020). On the other hand, these methods have led video prediction approaches to include an extra input, such as a semantic segmentation map of input frames, so that the per-pixel video prediction task is constrained by the auxiliary task of predicting the future of the extra input (Luc et al., 2017) resulting in more consistent and plausible outcomes.

2.2.2 Video Prediction

Video generation approaches aim to generate a sequence of smoothly and plausibly changing frames from noise vectors, whereas video prediction uses a given sequence of frames as context and tries to infer the subsequent frames. Prediction is often considered as *conditional generation*. Although no labels are required for training models for video prediction, this setup is not considered an unsupervised task either, because the target signal is also present in a given dataset. There is a substantial amount of work in video prediction with the advent of deep neural networks (Oprea et al., 2020; Rasouli, 2020) and we will provide a limited overview to give a general idea about the main approaches and challenges in the field by mostly focusing on methods that aim at compositional prediction. In this section, we review methods that tackle the video prediction problem in deterministic and stochastic frameworks. After visiting a few works that aim for compositionality for better prediction ability, we conclude with a summary of stochastic recurrent models, which are mostly formulated for prediction problems in diverse settings for different data modalities.

Deterministic Video Prediction

Early work for video prediction (Graves, 2013; Ranzato et al., 2014; Srivastava et al., 2015; Mathieu et al., 2015; Finn et al., 2016; Reda et al., 2018; Wichers et al., 2018) mainly relied on recurrent architectures such as vanilla RNNs or LSTMs (Hochreiter and Schmidhuber, 1997), convolutional LSTMs (Xingjian et al., 2015) or their extensions to spatio-temporal domain, such as predRNN (Wang et al., 2022) or CubicLSTM (Fan et al., 2019), for deterministic video prediction. The prevailing approach is to feed the input frame or features to the recurrent architectures to update the states at a given time and use the output of recurrent units for regressing the frame at the next time step. This line of work is considered deterministic in the sense that the same input results in the same prediction outcome and the latent representations are treated as deterministic variables. This single-step prediction framework can be extended for long-term prediction by feeding the regressed output as the input of the time step, in an autoregressive fashion. A sub-class of these works (Finn et al., 2016; Jia et al., 2016) use recurrent units to learn pixel-wise transformations, mostly in the form of convolutional kernels, in order to warp the input frame to the next time step. Although the latter group performs slightly better, these deterministic video prediction algorithms based on latent representations mostly suffer from blurry outputs as a result of pixel-wise reconstruction error (Mathieu et al., 2015).

Incorporating an adversarial loss into the training scheme has been one of the common methods to improve the fidelity of predicted frames (Lotter et al., 2015; Villegas et al., 2017; Vondrick and Torralba, 2017; Luc et al., 2020). The method by Villegas et al. (2017), in particular, constitutes one of the early attempts to implicitly disentangle motion and content for the prediction problem, in an analogous fashion to disentangled video generation. It has been soon followed by similar approaches (Hsieh et al., 2018; Guen and Thome, 2020). Vondrick and Torralba (2017) use Spatial Transformers (Jaderberg et al., 2015) for pixel-wise transformers in adversarial settings. Many other works have opted for optical flow estimation, either with (Liang et al., 2017; Lu et al., 2017) or without (Walker et al., 2015) adversarial loss. Different approaches, such as VoxelFlow (Liu et al., 2017) and frequency-based features (Xu et al., 2018) were also implemented for high-fidelity outcomes. All these models successfully improve the visual quality of predicted frames, yet, they are still deterministic approaches that tackle an inherently stochastic problem. We consider the video prediction problem as inherently stochastic because there are multiple plausible sequences that can resume the given input frames.

Stochastic Video Prediction

There are however further challenges related to the stochastic video prediction problem. Modeling future distributions of frames given one or few initial samples is very challenging due to the high-dimensional nature of the task, in addition to the highly entangled content and motion features. Early works converted deterministic frameworks into stochastic counterparts by VAE or GAN-based formulation of latent variables. For example, the method by Walker et al.

(2016) builds upon the model by Walker et al. (2015), and Babaeizadeh et al. (2017) modifies the work of Finn et al. (2016) using VAEs, whereas Lee et al. (2018) introduce adversarial loss to further improve the version by Babaeizadeh et al. (2017). However, most of these approaches learn the distribution of the proposed latent variables from the whole sequence and sample unconditionally at each time step at inference time. One can easily claim that conditioning latent variables sequentially would help with the continuity of predicted frames, at the expense of a more sophisticated learning strategy. Indeed, Denton and Fergus (2018) demonstrate how a time-varying distribution for the stochastic latent variables can improve the modeling capabilities, which is further extended by Castrejon et al. (2019) with hierarchical latent variables and higher capacity networks to produce visually appealing and coherent samples. Large networks are certainly shown to improve the visual quality of the output frames (Villegas et al., 2019); however, Franceschi et al. (2020) demonstrate that increasing the number of layers or neurons in the network is not the only solution: their method proposes to use residual connections for stochastic variables and for decoupling frame synthesis and dynamics. There are also other adaptations from different domains: VAE counterparts of dynamic convolutions (Xue et al., 2016), quantized autoencoders (Walker et al., 2021), transformers with three-dimensional self-attention and spatiotemporal subscaling (Weissenborn et al., 2019), as well as conditional pixel-wise generation (Kalchbrenner et al., 2017). All of them model the data distribution for video prediction with a range of different modalities.

Compositional Video Prediction

On the other hand, some entity abstraction methods, such as OP3 (Veerapaneni et al., 2020) and STOVE (Kossen et al., 2019), are principled and inherently serve for the video prediction problem as a virtue of the time-varying entity latent distribution in their formulation. Furthermore, Gao et al. (2019) divide the prediction problem into two parts: first, they predict a part of the future frame by extrapolating the input optical flow to account for regions that move fluidly. They train an additional inpainting module for the parts of the future frame which are previously occluded according to the model. Wu et al. (2020) further demonstrates how decoupling the scene and considering moving objects can improve video prediction outcomes in real-world sequences. There is a lot in common between this approach and our proposal regarding the prediction philosophy; however, they perform supervised training for learning to detect moving objects, and working on real-world sequences enables them to use a pre-trained inpainting module. For detecting moving objects, they use instance maps, optical flow, and semantic maps as extra inputs, the availability of which is heavily limited to certain scenarios such as autonomous driving. In contrast, we propose to first learn how to decouple objects and use this decomposition model for object-centric video prediction without any need for annotations.

Stochastic Recurrent Networks

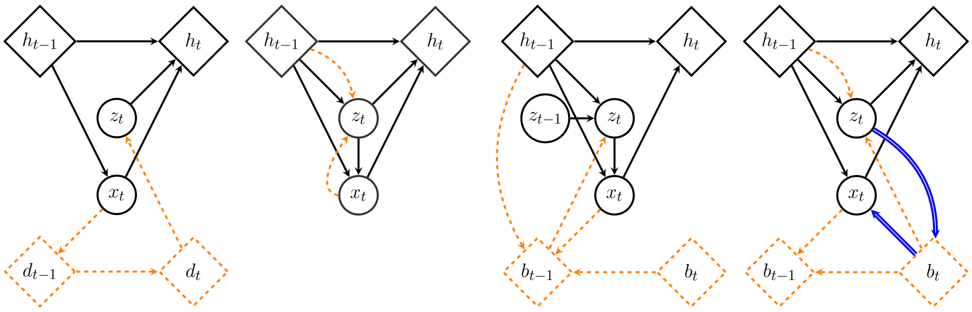
A state-space model (Durbin and Koopman, 2012) characterizes a stochastic, discrete-time process by describing how the latent variables and model outputs (observable variables of the model) change over time. Sequence modeling via state-space models is suitable for many tasks, which include but are not limited to trading, weather forecasting, language, and audio modeling, as well as video prediction.

Recently, the description of the evolution of latent variables in these models are formulated with recurrent neural networks (Bayer and Osendorfer, 2014; Chung et al., 2015; Fraccaro et al., 2016; Krishnan et al., 2016; Goyal et al., 2017; Hafner et al., 2019; Franceschi et al., 2020). Recurrent neural networks (RNNs) already involve a hidden state that is updated at each time step, and the evolution of the hidden state is deterministic. These recent efforts augment RNNs with stochastic latent variables and train the resultant stochastic RNNs with amortized variational inference, or with an auto-encoding framework as in Variational Autoencoders (Kingma and Welling, 2013). These networks mainly differ in modeling the dependencies between deterministic states and stochastic latent variables, which is usually a design choice that depends on the target application.

Figure 2.1 illustrates some well-known stochatic RNNs, namely, STORN (Bayer and Osendorfer, 2014), VRNN (Chung et al., 2015), SRNN (Fraccaro et al., 2016) and Z-Forcing (Goyal et al., 2017). We can clearly see different dependency assumptions between the deterministic states, h_t, and the stochastic units, z_t indicated by arrows. For example, in STORN (Bayer and Osendorfer, 2014), the stochastic units, z_t, are sampled independently at each time step. However, for the computation of the prior distribution $p(z_t|h_t)$, VRNN (Chung et al., 2015) prefers including the past information via the deterministic state of the previous time step. On a different note, SRNN (Fraccaro et al., 2016) separates the deterministic and stochastic parts, and more importantly, it conditions the stochastic variable z_t on z_{t-1} in an analogy to state-space models. It proposes a backward RNN for computing the approximate posterior, which involves an auxiliary deterministic state indicated by dashed lines in Figure 2.1(2.1.3). In contrast, Z-forcing (Goyal et al., 2017) removes the sequential dependence of stochastic variables and proposes auxiliary tasks of encoding information about the future, illustrated by double arrows in Figure 2.1(2.1.4).

Stochastic recurrent networks aim to bring the best of state space models and recurrent neural networks in order to better model sequential data. However, they do not scale up well to higher dimensional spaces, which limits their use for video prediction with auto-encoding approaches. On the other hand, we disentangle factors of variation for the video prediction problem, which allows us to work on sub-tasks in lower dimensional spaces.

We aim to develop a method that can extend our deterministic framework for stochastic prediction. We claim that the majority of the deterministic framework should stay deterministic, and the stochasticity should emerge from the predicted motion for each entity. Given that we predict motion parameters in an autoregressive way (Equation 3.44), SRNN (Fraccaro et al.,



((2.1.1)) STORN ((2.1.2)) VRNN ((2.1.3)) SRNN ((2.1.4)) Z-Forcing

Figure 2.1: Computation graph for generative models of sequences that use latent variables: STORN (Bayer and Osendorfer, 2014), VRNN (Chung et al., 2015), SRNN (Fraccaro et al., 2016) and Z-Forcing (Goyal et al., 2017). Graphs illustrate the generative models which predict the next observation in the sequence x_t, given previous ones $\{x_t\}_{t=1}^{t-1}$. Diamonds represent deterministic states and z_t stands for the stochastic latent variable. Dashed lines represent the computation that is part of the inference model. Double lines indicate auxiliary predictions implied by the auxiliary cost in Z-Forcing. Image taken from Goyal et al. (2017).

2016) model stands out as a suitable framework for modeling the time-varying distribution of motion parameters. Moreover, as these models are mainly designed for low-dimensional spaces (the inference schemes become very expensive otherwise), the adaptation of these approaches for the estimation of low-dimensional motion dynamics remains fairly accessible.

2.3 Object Representations in 3D

Similar to other object-related tasks in computer vision, 3D object detection is mostly posed as a supervised learning problem (Arnold et al., 2019; Liang et al., 2021). However, there is a small number of promising works that tackle the problem of unsupervised entity abstraction in 3D to alleviate the ambiguities that emerge from projection to 2D while forming images, such as occlusion.

Henderson and Lampert (2020) propose a 3D-aware generative model that employs a differentiable renderer to decompose moving objects from monocular videos recorded by a moving camera. For the same set-up of moving objects observed from multiple viewpoints, Li et al. (2021) suggest factorizing the latent variables in a spatio-temporal manner to decouple objects with their generative modeling. Li et al. (2020) challenge the idea of 3D inference from a single 2D observation and propose a multi-view framework for static objects, in which the typical entity-specific latent variables are updated over different views. To develop a 3D understanding of the given static scene, the network is required to predict the appearance from novel viewpoints during training. Stelzner et al. (2021) replace the decoder of SlotAttention (Locatello et al., 2020) model with a differentiable rendering algorithm in order to compose a static scene from unlabeled RGB-D data. Very recently, Kabra et al. (2021) leverage the shared structure across different scenes and decouple the latent representations into time-invariant object representations and time-varying viewpoints. They model the observations from RGB video input and train the proposed network by view synthesis based on inferred latent variables. All these recent methods contribute to the challenging problem of unsupervised 3D object decomposition by enforcing the 3D consistency of learned representations via explicit or implicit rendering. However, the mechanisms proposed for decoupling objects are mostly based on 2D cues. Moreover, posing the problem with generative modeling limit the application of these methods to relatively low-resolution datasets.

Du et al. (2020) also work on a monocular RGB video, but contrary to 2D object representations of a given video, they model object motion by the rigid-body transformation in three dimensions and render object masks based on 3D attributes. Gao et al. (2021b) also target 3D object representations, and they propose to obtain them by auto-encoding 3D transformations. However, the method by Gao et al. (2021b) requires explicit 3D models of objects for training, and the resultant 3D representations are only studied for object classification or image retrieval. In the last chapter of this thesis, we also aim for 3D representations for unsupervised object decomposition with the help of differentiable rendering. Differently from existing work, we propose to obtain volumetric (3D) features by lifting and fusing 2D cues, then address decomposition directly in three dimensions.

3 Continuous Motion and Consistency Objective

3.1 Introduction

As discussed in previous chapters, entity or object abstraction is of crucial importance for many tasks in computer vision and robotics. In order to alleviate the need for labor-intensive manual labeling, recent unsupervised approaches for entity abstraction mainly focus on autoencoding input frames into a set of latent variables, whose inference becomes a critical design choice, as detailed in Chapter 2. The input frame is reconstructed from the set of inferred latent variables, and the parameters of the model are learned by minimizing the reconstruction error. We argue that the decoding element in these methods, which maps latent variables back to input space, plays a crucial role and can be considered as an overhead that limits the application of these models to more sophisticated datasets. Hence, we take a different approach and formulate the unsupervised entity abstraction problem in a way to disambiguate *moving* objects with a masking strategy. We try to group pixels of an input frame into objects that can be represented by consistent motion between multiple time steps. In other words, we design the training objective of our new method to perform prediction-by-parts.

Whether formulated for a single-step or multiple time-steps, object-centric video prediction requires accurate representations of all moving entities in the sense that both visible and occluded parts of an entity should be modeled as completely as possible. In other words, to be able to predict a video frame at a time step, the model needs to "imagine" the full appearance of each entity, and warp "full" entities onto the target time step to successfully compose a frame from its moving and static parts. Consequently, in this chapter, we explore different strategies for generating an amodal mask for each moving entity and deduce the entity appearance for pixel positions indicated by the amodal masks. As amodal masks model both the visible and occluded parts of entities, we also work on the inverse problem of obtaining a visibility mask from amodal masks, which can be also referred to as a segmentation or composition mask. The purpose of such a visibility mask is to compose a frame from the inferred amodal masks and corresponding entity appearances based on their relative depth.

We set the learning objective of the proposed pipeline to be a combination of single-step and multi-step prediction. Single-step prediction ensures slowly-varying motion and multi-step prediction reinforces the inference of amodal masks and assists the inpainting module to "imagine" occluded parts of entities. To constrain the problem a bit further, we pose the multi-step prediction in a cyclic manner, *i.e.*, given two time instances t_1 and t_2, we predict the video frame at time t_2 using the entities and amodal masks inferred at time t_1, and vice versa.

Our proposed pipeline differs fundamentally from the latent-based models (Kosiorek et al., 2018; Greff et al., 2019; Veerapaneni et al., 2020; Locatello et al., 2020) by the principle that it is driven by masking with the objective of multi-step prediction. Although spatial attention is often used for such models (Kosiorek et al., 2018; Veerapaneni et al., 2020; Weis et al., 2020; Kipf et al., 2022), the outcome is often fed to a decoder to generate the entity appearance, rather than inferring the appearance from the input frame via dissecting it into its parts with masks. In addition, many methods (Veerapaneni et al., 2020; Zablotskaia et al., 2021; Kipf et al., 2022) refine the masks iteratively for a given time step, which clearly improves the mask outcomes, but such improvement is achieved at the expense of additional computation. Our method does not involve any iterative computation steps, neither for processing entities of a given frame nor for refining the inferred masks or the entity appearances. Experimental results on different datasets indeed demonstrate the efficiency and representation ability of the proposed model for moving objects.

3.2 Abstraction of Moving Objects

In this section, we first introduce the problem formulation and our modeling assumptions. We then describe our frame representation based on decomposing the scene into *moving* entities and background and explain our sequential modeling with a geometric approximation of motion for each entity in the frame. We outline the details for each sub-block for inferring amodal masks and the visibility mask, inpainting, and modeling motion. We then continue with the characterization of two different implementations that fit our problem formulation. After describing our main training objective of single-step and cyclic multi-step prediction losses, we introduce some auxiliary loss terms that aid our decomposition method.

3.2.1 Problem Formulation

Given a sequence of frames $\{\mathbf{x}_t\}_{t=1}^{T} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, our main objective is to decompose each frame $\mathbf{x}_t$ at time step t into K entities $\mathbf{e}_t^k$, $k = 1, \ldots, K$ which are spatially compact, share similar local features and at the same time, can be described by coherent motion between different time steps. We represent the support of each entity on the image plane by an amodal mask, $\mathbf{m}_t^k$ that encompasses both visible and occluded parts of an entity. The background then can be inferred from the set of masks $\left\{\mathbf{m}_t^k\right\}_{k=1}^{K}$ and it represents all the pixels that do not belong to any of the K entities, denoted by $\mathbf{m}_t^{K+1}$ and referred to as background mask. To handle occlusions, we use a binary composition mask $\mathbf{M}_t$ with $K+1$ channels so that each pixel is

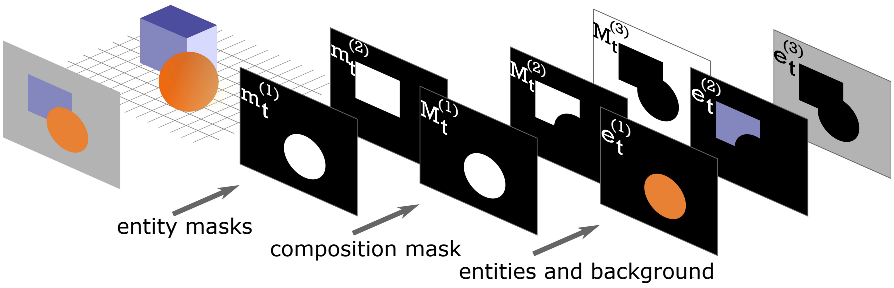


Figure 3.1: Illustration of proposed frame decomposition: the amodal masks $\{\mathbf{m}_t^k\}_{k=1}^2$ indicate the support of each object on the image plane, while the composition mask $\{\mathbf{M}_t^k\}_{k=1}^2$ represent only the visible parts.

occupied by one and only one visual entity or the background.

Figure 3.1 illustrates the notions in our decomposition model for a scene with two objects. For the illustrated case, the sphere occludes the rectangular prism when the scene is projected to the 2D plane of the given image. Note that the proposed object masks aim to cover the full support of each object when they are projected to the image plane individually. However, it is not trivial to infer them when the only source of information is a static image. On the other hand, a sequence of images where the sphere and the prism in Figure 3.1 move in different directions, as in Figure 3.2, or with different speeds, provides richer information for the inference of forenamed amodal masks.

If we continue with the example case in Figure 3.2, we can observe that the new structure of the scene reveals previously unseen parts of the entity $\mathbf{e}_t^2$ as well as the background $\mathbf{e}_t^3$, which are depicted with regions colored in red for the given scenario. Hence, to use a single-frame or multi-step prediction objective with inferred masks, we need to implement an inpainting step before warping any entity to compose the predicted frame at a different time step. Although there are multiple ways to realize inpainting, we opt for a relatively simple approach of accumulating information about an entity's appearance throughout time.

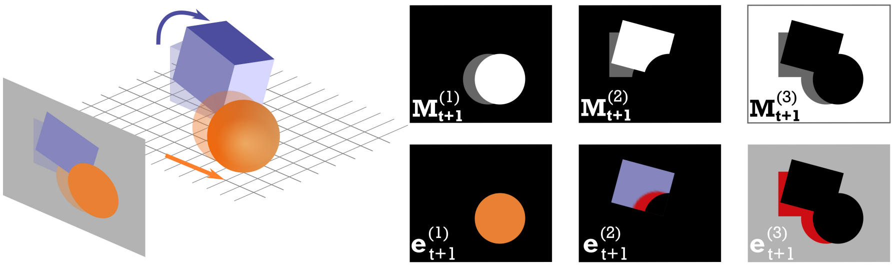


Figure 3.2: (Left)A possible continuation of the scene in Figure 3.1 at a consecutive time step: (right-top) composition mask at time $t+1$, overlaid with the composition mask at time t illustrates the 2D effect of 3D motion. (right-bottom) Entities and the background at the predicted time step reveal previously unseen parts, indicated by regions in red, which represents the pixels to be inpainted

In order to predict a frame at future time steps, we need to forecast the displacement of each entity. In this work, we assume that the motion of entities between consecutive frames can be approximated by planar motion. For example, the motion for each entity in Figure 3.2 can be perfectly expressed by a simple geometric transformation of each entity and/or mask. Hence, upon decomposition, we aim to predict the 3 × 3 affine or projective transformation matrix denoted as $\mathbf{z}^k$ for $\forall\, k$, which will morph each entity for the prediction of the next video frame. We denote the warping operation with predicted parameters by $T_{\mathbf{z}_t^k}(\cdot)$.

For learning such a compositional structure, we use both the next frame and cyclic long-term prediction as pretext tasks. In other words, we assume access to the whole sequence of frames during training time and formulate our problem as grouping pixels that can be consistently warped between different time steps by an approximate motion model. As discussed earlier, we derive the entities from the current frame using a masking strategy. This makes the widely used reconstruction objective impractical in our method as any partitioning strategy can easily lead to perfect reconstruction at a given time step. Prediction rather requires a system to make reasonable guesses based on available information, which, in our model, mainly refers to the motion of each entity and its appearance.

More concretely, at each time step:

- We decompose the input frame $\mathbf{x}_t$ into K entities and background, $\{\mathbf{e}_t^k\}_{k=1}^{K+1}$, with the help of amodal entity masks and resultant background mask $\{\mathbf{m}_t^k\}_{k=1}^{K+1}$,
- To account for any parts of the entities or the background, which can be newly exposed in the predicted frame, we utilize two inpainting modules $g_e(\cdot)$ and $g_b(\cdot)$, such that each inpainted entity and the background can be obtained by $\widetilde{\mathbf{e}}_t^k = g_e\left(\mathbf{e}_t^k, \widehat{\mathbf{e}}_{t-1}^k\right)$ for $k = 1, \dots, K$ and $\widetilde{\mathbf{e}}_t^{K+1} = g_b\left(\mathbf{e}_t^{K+1}, \widetilde{\mathbf{e}}_{t-1}^{K+1}\right)$, respectively,
- We predict parameters of an approximate motion model, denoted by $\mathbf{z}_t^k$, and warp each entity to the next time step, resulting in their predicted appearance at time step $t+1$, represented by $\{\widehat{\mathbf{e}}_{t+1}^k\}_{k=1}^{K}$,
- We also warp the mask of each entity to the next time step to infer the background and composition masks at time $t+1$, represented by $\{\widehat{\mathbf{m}}_{t+1}^k\}_{k=1}^{K+1}$, and $\widehat{\mathbf{M}}_{t+1}$, respectively,
- We finally synthesize the predicted frame $\widehat{\mathbf{x}}_{t+1}$ using the warped entities and the composition mask.

The approximation for motion might seem to restrict the modeling capability of the proposed method at first glance; however, using a parametric model enables the simple and computationally efficient combination of motion for multiple time steps, which leads to easy implementation of the aforementioned cyclic time-consistency constraint. Furthermore, modeling with more powerful motion models, such as optical flow, would require an extra step of clustering based on the predicted motion field. Such an extra step would complicate the main objective of entity abstraction by prediction, which already involves many tasks to be solved jointly. Finally, approximated motion stays consistent with our problem formulation by strictly constraining the grouping of pixels.

To summarize, we list our notation in Equations (3.1)-(3.17), where operator $\widetilde{\cdot}$ indicates variables after inpainting, and the operator $\widehat{\cdot}$ stands for predicted variables.

$$\mathbf{x}_t : \text{input frame at time } t \tag{3.1}$$

$$\mathbf{m}_t^k : \text{amodal mask at time } t \text{ for entity } k, \quad k \in \{1, \ldots, K\} \tag{3.2}$$

$$\mathbf{m}_t^{K+1} : \text{background mask inferred from } \left\{\mathbf{m}_t^k\right\}_{k=1}^{K} \tag{3.3}$$

$$\mathbf{M}_t^k : \textit{binary} \text{ composition mask at time } t, \quad k \in \{1, \ldots, K+1\} \tag{3.4}$$

$$f_m(\cdot) : \text{function mapping amodal masks to composition mask, } \textit{i.e.}$$

$$\left\{\mathbf{M}_t^k\right\}_{k=1}^{K+1} = f_m\left(\left\{\mathbf{m}_t^k\right\}_{k=1}^{K+1}\right) \tag{3.5}$$

$$\mathbf{e}_t^k : \text{entity } k \text{ inferred at time } t \text{ from input frame } \mathbf{x}_t \text{ via } \mathbf{M}_t^k, \quad k \in \{1, \ldots, K\} \tag{3.6}$$

$$\mathbf{e}_t^{K+1} : \text{background inferred at time } t \text{ from input frame } \mathbf{x}_t \text{ via } \mathbf{M}_t^{K+1} \tag{3.7}$$

$$g_e(\cdot) : \text{entity inpainting function} \tag{3.8}$$

$$g_b(\cdot) : \text{background inpainting function} \tag{3.9}$$

$$\widetilde{\mathbf{e}}_t^k : \text{entity } k \text{ after inpainting, such that } \widetilde{\mathbf{e}}_t^k = g_e\left(\mathbf{e}_t^k, \widehat{\mathbf{e}}_t^k\right), \quad k \in \{1, \ldots, K\} \tag{3.10}$$

$$\widetilde{\mathbf{e}}_t^{K+1} : \text{background after inpainting, such that } \widetilde{\mathbf{e}}_t^{K+1} = g_b\left(\mathbf{e}_t^{K+1}, \widetilde{\mathbf{e}}_{t-1}^{K+1}\right) \tag{3.11}$$

$$\mathbf{z}_t^k : \text{parameters of approximate motion model for entity } k, \quad k \in \{1, \ldots, K\} \tag{3.12}$$

$$T_{\mathbf{z}_t^k}(\cdot) : \text{warping operator with parameters } \mathbf{z}_t^k \tag{3.13}$$

$$\widehat{\mathbf{e}}_{t+1}^k : \text{inpainted entities warped onto time step } t+1 \text{ such that}$$

$$\widehat{\mathbf{e}}_{t+1}^k = T_{\mathbf{z}_t^k}\left(\widetilde{\mathbf{e}}_t^k\right), \quad k \in \{1, \ldots, K\} \tag{3.14}$$

$$\widehat{\mathbf{m}}_{t+1}^k : \text{amodal masks warped onto time step } t+1 \text{ such that}$$

$$\widehat{\mathbf{m}}_{t+1}^k = T_{\mathbf{z}_t^k}\left(\mathbf{m}_t^k\right), \quad k \in \{1, \ldots, K\} \tag{3.15}$$

$$\widehat{\mathbf{m}}_{t+1}^{K+1} : \text{predicted background mask for time } t+1 \text{ inferred from } \left\{\widehat{\mathbf{m}}_t^k\right\}_{k=1}^{K} \tag{3.16}$$

$$\widehat{\mathbf{M}}_{t+1}^k : \text{predicted composition mask for time step } t+1 \text{ such that}$$

$$\left\{\widehat{\mathbf{M}}_t^k\right\}_{k=1}^{K+1} = f_m\left(\left\{\widehat{\mathbf{m}}_t^k\right\}_{k=1}^{K+1}\right) \tag{3.17}$$

3.2.2 Amodal Entity Masks

There are different ways to represent entities in a frame, which include but are not limited to, 2D or 3D bounding boxes, occupancy maps, or segmentation masks. In this work, we represent each entity with a mask that embodies both visible and occluded parts. This amodal segmentation approach requires a good understanding of the 3D structure of a scene while manipulating 2D images. However, inference of invisible parts necessitates the accumulation of information throughout time as well as content-based imagination ability, in general. We hence follow an autoregressive approach to obtain amodal entity masks, which can be described by the function $f_1(\cdot)$ in Equation (3.18):

$$\mathbf{m}_t^k = f_1\left(\mathbf{x}_t, \mathbf{m}_{t-1}^k\right). \tag{3.18}$$

As unsupervised amodal segmentation is not a trivial problem, we experimented with different approaches and we present two of those which we found more promising than the others. These two approaches for the function $f_1(\cdot)$ are denoted by $f_1^{(1)}(\cdot)$ and $f_1^{(2)}(\cdot)$. The first approach keeps the spatial information of the input frame $\mathbf{x}_t \in [0,1]^{H\times W\times 3}$, where H, W denote the image height and weight and maps the input frame to feature vectors $\mathbf{f}_t^{(1)}$ of the same spatial size, as described in Equation(3.19). It then converts the feature maps to a set of K entity masks. For this approach, we impose the autoregressive inference on image features, as expressed in Equation(3.20), in order to promote slowly varying representations for continuity. The masks are inferred together as indicated by in Equation(3.21), where the function $f_{1,2}^{(1)}(\cdot)$ takes all K masks from the previous time step as input. We denote the composition these two operations, *i.e.*, functions $f_{1,1}^{(1)}(\cdot)$ and $f_{1,2}^{(1)}(\cdot)$ by $f_1^{(1)}(\cdot)$.

$$f_1^{(1)} : \mathbf{x}_t \in [0,1]^{H\times W\times 3} \rightarrow \mathbf{f}_t^{(1)} \in \mathbb{R}^{H\times W\times C_f^{(1)}} \rightarrow \left\{\mathbf{m}_t^k\right\}_{k=1}^K \in [0,1]^{H\times W\times K} \tag{3.19}$$

$$\mathbf{f}_t^{(1)} = f_{1,1}^{(1)}\left(\mathbf{x}_t, \mathbf{f}_{t-1}^{(1)}\right) \tag{3.20}$$

$$\left\{\mathbf{m}_t^k\right\}_{k=1}^K = f_{1,2}^{(1)}\left(\mathbf{f}_t^{(1)}, \left\{\mathbf{m}_{t-1}^k\right\}_{k=1}^K\right) \tag{3.21}$$

The second approach rather models the similarity of local windows of the input frame explicitly. For this purpose, the function $f_1^{(2)}(\cdot)$ first maps the input frame to feature vectors $\mathbf{f}_t^{(2)}$ of a lower spatial resolution, such that, $\mathbf{f}_t^{(2)} \in \mathbb{R}^{H_c\times W_c\times C_f^{(2)}}$, where $H_c < H$, $W_c < W$. Then, it computes the entries of the vector $\mathbf{a}_t$ by cosine feature similarity of each unique pair of features $\mathbf{f}_t^{(2)}[i,j]$ and $\mathbf{f}_t^{(2)}[u,v]$ as expressed in Equation (3.24), where (i,j) and (u,v) are spatial indexes for the features $\mathbf{f}_t^{(2)}$ and $\|\cdot\|$ denotes the L2 norm. Then, we obtain a state vector $\mathbf{s}_{\mathbf{m},t}^k$ for each entity individually, denoted by $\mathbf{s}_{\mathbf{m},t}^k$ in Equation (3.25), which is updated in an autoregressive manner. The state vector serves as a memory unit, and it is updated for each entity individually. In the last step, entity masks are inferred from the corresponding state vectors and mapped to the range $[0,1]$ with the help of a sigmoid function, as given in Equation (3.26). The functions $f_{1,i}^{(2)}$ for $i \in \{1,2,3,4\}$ collectively compose the function $f_1^{(2)}$.

$$f_1^{(2)} : \mathbf{x}_t \in [-1,1]^{H\times W\times 3} \rightarrow \mathbf{f}_t^{(2)} \in \mathbb{R}^{H_c\times W_c\times C_f^{(2)}} \rightarrow \left\{\mathbf{m}_t^k\right\}_{k=1}^K \in [0,1]^{H\times W\times K} \tag{3.22}$$

$$\mathbf{f}_t^{(2)} = f_{1,1}^{(2)}(\mathbf{x}_t) \tag{3.23}$$

$$\mathbf{a}_t = \frac{\mathbf{f}_t^{(2)}[i,j]^T \mathbf{f}_t^{(2)}[u,v]}{\left\|\mathbf{f}_t^{(2)}[i,j]\right\| \left\|\mathbf{f}_t^{(2)}[u,v]\right\|} \ \forall [i,j],[u,v] \in H_c \times W_c$$

$$\text{such that } i < u, j < v \tag{3.24}$$

$$\mathbf{s}_{\mathbf{m},t}^k = f_{1,3}^{(2)}\left(\mathbf{s}_{\mathbf{m},t-1}^k, f_{1,2}^{(2)}(\mathbf{a}_t)\right), \quad \forall k \in \{1,\dots,K\} \tag{3.25}$$

$$\mathbf{m}_t^k = sigmoid\left(f_{1,4}^{(2)}(\mathbf{s}_{\mathbf{m},t}^k)\right), \quad \forall k \in \{1,\dots,K\} \tag{3.26}$$

After obtaining the amodal entity masks, either by using $f_1^{(1)}$ or $f_1^{(2)}$, we infer the background mask $\mathbf{m}_t^{K+1}$ from the set of entity masks to indicate all pixels that do not belong to any of the entities. Its computation is presented in Equation(3.27), where **0** and **1** represent images with values that are all zeros, and all ones, respectively.

$$\mathbf{m}_t^{K+1} = \max\left(\mathbf{0}, \mathbf{1} - \sum_{k=1}^{K} \mathbf{m}_t^k\right) \tag{3.27}$$

3.2.3 Occlusion Handling and Inpainting

Given amodal masks, our next task is to compute the composition mask which takes occlusions into account. To start with, we want the generation of the composition mask to be invariant to the depth ordering of amodal masks, that is, any ordering of entity indexing with k must lead to the same composition of the image. And at the composition step, the entity closer to the camera needs to be prioritized. This is the main motivation behind our composition mask, which selects one entity for each visible pixel in the image based on the content of the frame. Hence, our aim is to define a function $f_m(\cdot)$ which achieves $\{\mathbf{M}_t^k\}_{k=1}^{K+1} = f_m\left(\{\mathbf{m}_t^k\}_{k=1}^{K+1}\right)$, where the composition mask $\mathbf{M}_t^k$ is activated only once across $K+1$ channels for each pixel location.

Inferring the visibility mask can be considered a combinatorial problem, which tackles the ordering of amodal masks according to the relative depth of objects they are associated with. There are multiple ways to approach this problem, and here, we provide two different solutions. Our first idea is to explicitly order entity masks $\{\mathbf{m}_t^k\}_{k=1}^{K}$ such that the mask with index $k = 1$ indicates the entity that is closest to the camera. Hence, at composition time, we can discard all other entities with indices $k > 1$ for the pixels where $\mathbf{m}_t^1$ is nonzero. Under such a formulation, we want the masks to be almost binary valued so that they act as strong indicators of entity presence. Hence, for the first implementation of $f_m^{(1)}(\cdot)$, we use a soft thresholding mechanism, to binarize the entity masks as $\left\{\overline{\mathbf{m}}_t^k\right\}_{k=1}^{K}$ as given in Equation (3.28), and order them using a permutation matrix $\mathbf{P}_t \in \{0,1\}^{K\times K}$ as described in Equation (3.29). The scalars α, β in Equation (3.28) are hyperparameters, and the operator $\otimes$ stands for the permutation of binarized entity masks $\left\{\overline{\mathbf{m}}_t^k\right\}_{k=1}^{K} \in \{0,1\}^{H\times W\times K}$ with matrix $\mathbf{P}_t$ in the third dimension. The computation of the composition mask is then finalized by Equation (3.30), which achieves the aforementioned prioritization. We denote the group of operations realized in Equations (3.28)-(3.30) with $f_m^{(1)}(\cdot)$.

$$\overline{\mathbf{m}}_t^k = sigmoid\left(\alpha * (\mathbf{m}_t^k - \beta)\right) \tag{3.28}$$

$$\left\{\overline{\overline{\mathbf{m}}}_t^k\right\}_{k=1}^K = \mathbf{P}_t \otimes \left\{\overline{\mathbf{m}}_t^k\right\}_{k=1}^K \tag{3.29}$$

$$\mathbf{M}_t^k = \begin{cases} \overline{\overline{\mathbf{m}}}_t^k & \text{if } k = 1 \\ \max\left(0, \overline{\overline{\mathbf{m}}}_t^k - \sum_{l=1}^{k-1} \overline{\overline{\mathbf{m}}}_t^l\right) & \forall k \in \{2, \dots, K+1\} \end{cases} \tag{3.30}$$

However, learning a permutation in a differentiable pipeline can be challenging. Another option is to learn two scalars α^k, β^k for each entity mask, which can implicitly achieve ordering by changing the scale of masks according to the relative depth of objects. Hence, we use these two scalars to modulate the amodal as given in Equation (3.32) before applying a $softmax$ function across the last dimension. Note that the temperature τ in Equation (3.33) enables control over the similarity between $\mathbf{M}_t$ and a binary-valued mask. We find the choice of its value critical for the convergence of the training, as it balances between the propagation of errors for training and having binary-like masks to represent objects. We recommend choosing an initial value between 0.5 – 1.0 and decreasing its value throughout training iterations to a smaller value, such as 0.1. We will denote the group of operations realized in Equations (3.31)-(3.33) with $f_m^{(2)}(\cdot)$, which is the second alternative for the computation of the visibility mask.

$$\{\alpha_t^k, \beta_t^k\}_{k=1}^{K+1} = f_{m,1}^{(2)}(\mathbf{x}_t, \{\mathbf{m}_t^k\}_{k=1}^{K+1}) \tag{3.31}$$

$$\overline{\mathbf{m}}_t^k = \alpha_t^k \mathbf{m}_t^k + \beta_t^k \tag{3.32}$$

$$\mathbf{M}_t^k = \frac{\exp\left(\overline{\mathbf{m}}_t^k / \tau\right)}{\sum_{k=1}^{K+1} \exp\left(\overline{\mathbf{m}}_t^k / \tau\right)} \tag{3.33}$$

Once we compute the composition mask, either by $f_m^{(1)}(\cdot)$ or $f_m^{(2)}(\cdot)$, the resultant $\mathbf{M}_t$ describes the visible part of each entity at the given time step t. Hence, a simple Hadamard product between the input frame and the corresponding channel of the composition mask will induce the entity appearance $\mathbf{e}_t^k$ as described in Equation (3.34). To complement the amodal approach for masks, we inpaint the entities $\{\mathbf{e}_t^k\}_{k=1}^K$ and the background $\mathbf{e}_t^{K+1}$ based on all the information available up to time t as in Equations (3.35)-(3.36). Note that entity inpainting operation uses the predicted entity from time $t-1$, $\widehat{\mathbf{e}}_{t-1}^k$, which is obtained by warping the inpainted entity at time $t-1$, $\widetilde{\mathbf{e}}_{t-1}^k$, to the time step t such that $\widehat{\mathbf{e}}_t^k = T_{\mathbf{z}_{t-1}}(\widetilde{\mathbf{e}}_{t-1}^k)$. This operation is explained in the next section and it can be considered as registration of entity appearance in time with predicted motion parameters.

$$\mathbf{e}_t^k = \mathbf{M}_t^k \odot \mathbf{x}_t \tag{3.34}$$

$$\widetilde{\mathbf{e}}_t^k = g_e(\mathbf{e}_t^k, \widehat{\mathbf{e}}_{t-1}^k) \quad \text{for } k = 1, \ldots, K \tag{3.35}$$

$$\widetilde{\mathbf{e}}_t^{K+1} = g_b(\mathbf{e}_i^{K+1}, \widetilde{\mathbf{e}}_{t-1}^{K+1}) \tag{3.36}$$

Note that we use two different functions to update the appearance of each entity and the background, $g_e(\cdot)$ and $g_b(\cdot)$ respectively. They both can be considered as time-causal inpainting functions based on previously exposed information. The reason for different inpainting functions for the entities and the background is the fact that visual characteristics as well as the area to be inpainted for the background are likely to be different than for the individual entities.

In more detail, for entity inpainting, we first compose a candidate appearance based on the amodal and the composition mask. It uses the visible information inferred at time t for each entity and complements it for the invisible part from the prediction at time $t-1$. More formally, the candidate appearance $\bar{\mathbf{e}}_t^k$ is computed according to Equation (3.37), which is later refined with a residual function $g_{e,r}(\cdot)$ as in Equation (3.38).

$$\bar{\mathbf{e}}_t^k = \left(\mathbf{M}_t^k \odot \mathbf{x}_t\right) + \left([\mathbf{m}_t^k - \mathbf{M}_t^k] \odot \widehat{\mathbf{e}}_{t-1}^k\right) \tag{3.37}$$

$$\widetilde{\mathbf{e}}_t^k = \bar{\mathbf{e}}_t^k + g_{e,r}(\bar{\mathbf{e}}_t^k) \tag{3.38}$$

Similarly, we compute a residual image for painting the background inferred at time t, $\mathbf{e}_t^{K+1}$ by using the function $g_{b,r}(\cdot)$ For the gating mechanism, we use an image of ones, $\mathbf{1}$ as the whole canvas can be considered as background once the moving entities are removed. Hence, the inpainting function $g_b(\cdot)$ follows Equations (3.39)-(3.40).

$$\bar{\mathbf{e}}_t^{K+1} = \widetilde{\mathbf{e}}_{t-1}^{K+1} + g_{b,r}(\mathbf{e}_t^{K+1}, \widetilde{\mathbf{e}}_{t-1}^{K+1}) \tag{3.39}$$

$$\widetilde{\mathbf{e}}_t^{K+1} = \left(\mathbf{M}_t^{K+1} \odot \mathbf{x}_t\right) + \left([\mathbf{1} - \mathbf{M}_t^{K+1}] \odot \bar{\mathbf{e}}_t^{K+1}\right) \tag{3.40}$$

3.2.4 Motion Modelling

Once we decompose the input video frame, we need to predict how each entity will appear in the next time step. We approximate the motion of each entity between consecutive frames by a parametric transformation described by a 3 × 3 matrix $\mathbf{Z}_t^k$. We, in fact, assume explicit parameters of the transformation In other words, the parametrization $\mathbf{z}_t^k$ is composed of a set of scalars $\mathbf{z}_t^k = \{\theta^k, t_x^k, t_y^k, \log(s_x^k), \log(s_y^k), r^k\}$ denoting the rotation angle, translation in x- and y-directions, scaling in x- and y-directions and shear, respectively. The affine transformation matrix $\mathbf{Z}_k^k$ is then obtained by Equation (3.41).

$$\mathbf{Z}_t^k = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & r & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.41}$$

The affine transformation indicated by the 3 × 3 matrix $\mathbf{Z}_t^k$ can be used for warping any mask or image. Indeed, as used in Spatial Transformers (Jaderberg et al., 2015), warping a pixel grid at frame resolution and computing the values at the warped grid with bilinear sampling can be applied to any visual input as described by Equations (3.42)-(3.43), where (u_t^k, v_t^k) stand for pixel coordinates, *i.e.*, the pixel grid indexes. We denote the overall warping operation by $T_{\mathbf{z}_t}(\cdot)$ or $T_{\mathbf{Z}_t}(\cdot)$, which means that the predicted appearance of the inpainted entity k at time step $t+1$ can be expressed by $\widehat{\mathbf{e}}_{t+1}^k = T_{\mathbf{z}_t}(\widetilde{\mathbf{e}}_t^k)$. Such an approximation works because we apply these transformations locally by layering the input image with entity masks and transforming each mask-entity pair with a dedicated set of parameters.

$$\begin{bmatrix} u_{t+1}^k \\ v_{t+1}^k \\ 1 \end{bmatrix} = \mathbf{Z}_t^{(k)} \begin{bmatrix} u_t^k \\ v_t^k \\ 1 \end{bmatrix} \tag{3.42}$$

$$\widehat{\mathbf{m}}_{t+1}^k[u_{t+1}^k, v_{t+1}^k] = \sum_{x=1}^{W} \sum_{y=1}^{H} \mathbf{m}_t^k[u,v] \max(0, 1 - |u_t^k - u|) \max(0, 1 - |v_t^k - v|) \tag{3.43}$$

The set of affine parameters $\mathbf{z}_t^k$ for each entity is computed with another autoregressive function $f_z(\cdot)$ based on the current frame and corresponding entity mask in the last two time steps, as we believe that the motion is content-dependent. Moreover, in order to better model the continuity of the motion, we make use of a recursive state $\mathbf{s}_{\mathbf{z},t}^k$ for the prediction of affine parameters. Overall, we represent the function to compute affine parameters by $f_z(\cdot)$, which is composed of functions $f_{z,1}(\cdot)$ and $f_{z,2}(\cdot)$, described in Equations (3.44)-(3.45), respectively. The function $\mathbf{z}_t^k$, hence, denotes the set of operations performed by $f_{z,1}(\cdot)$ and $f_{z,2}(\cdot)$.

$$\mathbf{s}_{\mathbf{z},t}^k = f_{z,1}(\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k, \mathbf{z}_{t-1}^k, \mathbf{s}_{\mathbf{z},t-1}^k) \tag{3.44}$$

$$\mathbf{z}_t^k = f_{z,2}(\mathbf{s}_{\mathbf{z},t-1}^k) \tag{3.45}$$

3.2.5 Prediction

Once the current frame, $\mathbf{x}_t$ is decomposed into entities via masks, and inpainted with the mechanisms described above, our final task is to project each entity to the next time step with the predicted transformation parameters in order to compose the predicted frame, $\widehat{\mathbf{x}}_{t+1}$. The warping of each inpainted object to time $t+1$ with corresponding parameters is achieved via Equation (3.46).

$$\widehat{\mathbf{e}}_{t+1}^k = T_{\mathbf{z}_t^k}\left(\widetilde{\mathbf{e}}_t^{(k)}\right) \quad \forall k \in \{1, \ldots, K\} \tag{3.46}$$

In order to infer the composition mask at the next time step, we also warp the amodal entity masks to the next time step using the same parameters and predict the background mask at time $t+1$ as in Equation (3.48). The function $f_m(\cdot)$ can be either $f_m^{(1)}(\cdot)$ described by Equations (3.28)-(3.30) or $f_m^{(2)}(\cdot)$ described by Equations (3.31)-(3.33).

$$\widehat{\mathbf{m}}_{t+1}^{K+1} = \max\left(\mathbf{0}, \mathbf{1} - \sum_{k=1}^{K} T_{\mathbf{z}_t^k}\left(\mathbf{m}_t^k\right)\right) \tag{3.47}$$

$$\widehat{\mathbf{M}}_{t+1} = f_m\left(\left\{T_{\mathbf{z}_t^k}\left(\mathbf{m}_t^k\right)\right\}_{k=1}^{K}, \widehat{\mathbf{m}}_{t+1}^{K+1}\right) \tag{3.48}$$

We finally compose the predicted frame using the inpainted entities warped to time $t+1$, $\{\widehat{\mathbf{e}}_{t+1}^k\}_{k=1}^K$, inpainted background $\widetilde{\mathbf{e}}_{t+1}^{K+1}$ and predicted composition mask $\widehat{\mathbf{M}}_{t+1}$ as given in Equation (3.49).

$$\widehat{\mathbf{x}}_{t+1} = \bigcup_{k=1}^{K}\left(\widehat{\mathbf{M}}_{t+1}^{(k)} \odot \widehat{\mathbf{e}}_{t+1}^{(k)}\right) \cup \left(\widehat{\mathbf{M}}_{t+1}^{(K+1)} \odot \widetilde{\mathbf{e}}_{t}^{(K+1)}\right) \tag{3.49}$$

As the occlusion mask and the resultant background masks do not overlap by construction, the union operations in Equation (3.49) can be easily implemented as pixel-wise summation.

3.3 Learning

The proposed framework is composed of many different sub-blocks, which can be described by functions $f_1(\cdot)$, which infers the amodal mask, $f_m(\cdot)$, which computes the composition mask from amodal masks, inpainting functions $g_b(\cdot)$ and $g_e(\cdot)$, as well the function $f_z(\cdot)$ which predicts the geometric transformation parameters. Some of them are (partially) described by the explicit operations, which are described in the previous section; however, the majority of them cannot be formulated explicitly. Hence, we parameterize these functions by neural networks and we aim to implement the model proposed in Section 3.2.5 for entity abstraction in videos as an end-to-end trainable pipeline. In this section, we describe the loss terms which can be used for training the parameters of such a network.

Prediction Loss

To start with, our main objective is to minimize the average $L1$ reconstruction error between the predicted frame and ground truth next frame for the whole sequence of length T, which can be denoted by $\mathscr{L}_{\text{prediction}}$ as in Equation (3.50).

$$\mathscr{L}_{\text{prediction}} = \frac{1}{T-1}\sum_{t=1}^{T-1} \|\mathbf{x}_{t+1} - \widehat{\mathbf{x}}_{t+1}\|_1 \tag{3.50}$$

Consistency Loss

The difference between consecutive frames is often very small and the network can end up learning identity transformation parameters for motion with almost random grouping pixels as amodal masks. Hence, instead of using a loss function that would only consist of a reconstruction term for single-step prediction as in Equation (3.50), we reinforce the learning with additional loss terms, and our main contribution is the loss term for cyclic long-term prediction. We argue that the motion of an object between multiple time steps is expected to be more articulated, hence, long-term prediction can have stronger cues for our abstraction task. Long-term prediction objective also can help to learn to infer invisible parts of the object as the occlusions can be more articulated in frames that are further apart in time.

For this purpose, at each training step when we have access to the whole video sequence, we sample two time instants $t_1 \sim U[1, T-1)$ and $t_2 \sim U(1, T-1]$, where $t_1 < t2$ and $U[\,]$ resembles a discrete uniform distribution, and T denotes the length of the given video sequence. As a new pair of (t_1, t_2) is sampled thousands of times, we visit each possible pair of video frames for associated multi-step prediction. We then use the set of masks $\{\mathbf{m}_{t_1}^k\}_{k=1}^K$, inpainted entities $\{\widetilde{\mathbf{e}}_{t_1}^k\}_{k=1}^K$ and the background $\widetilde{\mathbf{e}}_{t_1}^{K+1}$, and the set of transformation parameters $\{\mathbf{Z}_t^k\}_{t=t_1}^{t_2-1}$ to predict the frame at time t_2 by following our prediction steps. We combine the parameters from t_1 to $t_2 - 1$, which are all predicted by the model, by simple matrix multiplication as in Equation (3.51). We then use the resultant 3×3 matrix to warp all the masks of time t_1 as in Equation (3.52) for the prediction of the frame at time t_2. After deriving the background mask $\widehat{\mathbf{m}}_{t_2}^{K+1}$ with Equation (3.53), we can obtain the composition mask for t_2 using the Equation (3.54).

$$\mathbf{Z}_{\circ}^k = \mathbf{Z}_{t_2-1}^k \mathbf{Z}_{t_2-2}^k \ldots \mathbf{Z}_{t_1}^k, \quad \forall k \tag{3.51}$$

$$\widehat{\mathbf{m}}_{t_2}^k = T_{\mathbf{Z}_{\circ}^k}\left(\mathbf{m}_{t_1}^k\right), \quad \text{for} k \in \{1, \ldots, K\} \tag{3.52}$$

$$\widehat{\mathbf{m}}_{t_2}^{K+1} = \max\left(\mathbf{0}, \mathbf{1} - \sum_{k=1}^{K} T_{\mathbf{Z}_{\circ}^k}\left(\mathbf{m}_{t_1}^k\right)\right) \tag{3.53}$$

$$\widehat{\mathbf{M}}_{t_2} = f_m\left(\left\{T_{\mathbf{Z}_{\circ}^k}\left(\mathbf{m}_{t_2}^k\right)\right\}_{k=1}^{K}, \widehat{\mathbf{m}}_{t_2}^{K+1}\right) \tag{3.54}$$

We can then form the predicted frame at time t_2, $\mathbf{x}_{t_2}$, as $\widehat{\mathbf{x}}_{t_2}$ by warping all the inpainted entities with combined parameters $\mathbf{Z}_{\circ}$ and use the derived composition mask $\widehat{\mathbf{M}}_{t_2}$ to compose the predicted frame as described in Equation (3.55).

$$\widehat{\mathbf{x}}_{t_2} = \bigcup_{k=1}^{K}\left(\widehat{\mathbf{M}}_{t_2}^{(k)} \odot T_{\mathbf{Z}_{\circ}^k}\left(\widetilde{\mathbf{e}}_{t_1}^{(k)}\right)\right) \cup \left(\widehat{\mathbf{M}}_{t_2}^{(K+1)} \odot \widetilde{\mathbf{e}}_{t_1}^{(K+1)}\right) \tag{3.55}$$

Similarly, we use the masks $\{\mathbf{m}_{t_2}^k\}_{k=1}^K$ and inpainted entities and the background $\{\widetilde{\mathbf{e}}_{t_2}^k\}_{k=1}^{K+1}$ from time step t_2 to predict the frame at time t_1, $\mathbf{x}_{t_1}$ as $\widehat{\mathbf{x}}_{t_1}$ using the inverse of the combined transformation parameters, *i.e.*, $\mathbf{Z}_\circ^{-1}$ following the same long-term prediction steps. We then use the average of these two long-term prediction errors and define our cyclic consistency loss term $\mathscr{L}_{\text{consistency}}$ as in Equation (3.56).

$$\mathscr{L}_{\text{consistency}} = \frac{\left\|\mathbf{x}_{t_2} - \widehat{\mathbf{x}}_{t_2}\right\|_1 + \left\|\mathbf{x}_{t_1} - \widehat{\mathbf{x}}_{t_1}\right\|_1}{2} \tag{3.56}$$

Background Loss

As we assume a static background, we can enforce all dynamic elements of the scene to be represented by entities and penalize the discrepancy between the background information sampled at time t_1 and t_2, as described in Equation (3.57).

$$\mathscr{L}_{\text{background}} = \left\|\widetilde{\mathbf{e}}_{t_2}^{K+1} - \widetilde{\mathbf{e}}_{t_1}^{K+1}\right\|_1 \tag{3.57}$$

Mask Sparsity and Concentration

Additionally, we want the entity masks to be spatially sparse and localized. We accordingly define two loss terms promoting the sparsity of masks, given by Equation (3.58), and their geometric concentration, given by Equation (3.59). The term $\mathscr{L}_{\text{mask concentration}}$ penalizes the spread of each amodal mask around its center of mass, which is denoted by (c_u^k, c_v^k) in Equation (3.59). And the tuple (u, v) stands for pixel coordinates. The idea of mask concentration is borrowed from co-part segmentation models (Hung et al., 2019).

$$\mathscr{L}_{\text{mask sparsity}} = \frac{1}{HWK(T-1)} \sum_{t=1}^{T-1} \sum_{k=1}^{K} \sum_{u,v} \|\mathbf{m}_t^k[u, v]\|_1 \tag{3.58}$$

$$\mathscr{L}_{\text{mask concentration}} = \frac{1}{HWK(T-1)} \sum_{t=1}^{T-1} \sum_{k=1}^{K} \sum_{u,v} \left(\|(u, v) - (c_u^k, c_v^k)\|_2^2 \frac{\mathbf{m}_t^k[u, v]}{\|\mathbf{m}_t^{(k)}[u, v])\|_1} \right) \tag{3.59}$$

Without the additional constraint on entity masks, it is possible to end up with representations that poorly abstract entities by including artifacts, particularly from textureless parts of the frame. We find constraining the entity masks to be compact crucial for inferring more precise entity masks.

Our training objective is the minimization of a total loss term $\mathscr{L}$, which is a weighted sum of (a subset of) aforementioned loss terms. We will provide the exact loss terms for the different implementations in the next section.

3.4 Implementation

In Section 3.2.5, we proposed different approaches for the moving entity abstraction problem in videos. After exhausting experimentation, we empirically found two implementations combining different options presented in Section 3.2.5 achieve better performance than other combinations. Hence, in this section, we provide the details of these two different implementations. Both models share the prediction objective, *i.e.*, they are both trained with a loss function that includes both $\mathscr{L}_{\text{prediction}}$ and $\mathscr{L}_{\text{consistency}}$ terms. However, they differ in the way the masks are computed and the occlusion handled.

Model 1.

For the first model, we opt for mask computation dictated by a group of operations $f_1^{(1)}$ described in Equations (3.19)-(3.19). Briefly, the input frame is processed by an autoregressive feature extractor which keeps the spatial resolution the same. In particular, we use three layers of convolutional LSTM units (Xingjian et al., 2015) after three layers of residual blocks (He et al., 2016) for the inference of object masks $\mathbf{m}_t$. The last convolutional LSTM layer has K hidden units, the output of which is later normalized with instance normalization (Ulyanov et al., 2016) and mapped to the range $[0, 1]$ with a sigmoid function.

Occlusion handling for this model is achieved by ordering the masks based on their inferred relative depth, which is expressed as the permutation matrix $\mathbf{P}_t$. The permutation matrix is approximated as a doubly stochastic matrix (DSM), whose rows and columns sum up to 1. We first process input per-entity, $\{\mathbf{x}_t, \mathbf{m}_t^k\}$ with convolutional layers. The output is then fed to fully connected layers (MLP), where the uppermost layer has K hidden units. The overall output of K set of variables is then reshaped to construct a $K \times K$ matrix, which is validated to be a DSM via the Gumbel-Sinkhorn algorithm (Mena et al., 2018).

For the prediction of transformation parameters, we use another CNN to process visual input of $\{\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k\}$, and feed its output, as well as the parameters from the previous time step, $\mathbf{z}_{t-1}^k$ as inputs to another MLP to obtain $\mathbf{z}_t^k$, which is then converted to an affine matrix by Equation (3.41).

The function parameters for learning the permutation matrix and geometric transformations are shared across all entities. In addition, the per-entity processing is achieved in parallel with no sequential processing or iterative refinements involved, which makes our model scalable when increasing K, both in terms of computation and memory, in contrast to existing iterative algorithms (Zablotskaia et al., 2021; Veerapaneni et al., 2020).

Finally, for inpainting objects, the residual function $f_{e,r}(\cdot)$ in Equation(3.38) is implemented as a shallow CNN with a small perceptive field. On the other hand, a larger perceptive field is preferred for the residual function used for background inpainting, *i.e.*, the function $f_{b,r}(\cdot)$ in Equation(3.39).

We use the loss function $\mathscr{L}^{(1)}_{\text{total}}$ in Equation (3.60) for training the parameters of Model 1 with Adam optimizer (Kingma and Ba, 2014). We set $\lambda_1 = 1$, and use a varying λ_2, which starts from 0.25 and increases to 1 through early training in order to prevent the divergence of predicted transformation parameters. The auxiliary loss terms for constraining entity masks contribute less to the overall loss term with $\lambda_3 = \lambda_4 = 0.1$. The whole end-to-end differentiable pipeline for Model 1 is illustrated in Figure 3.3.

$$\mathscr{L}^{(1)}_{\text{total}} = \lambda_1 \mathscr{L}_{\text{prediction}} + \lambda_2 \mathscr{L}_{\text{consistency}} + \lambda_3 \mathscr{L}_{\text{mask sparsity}} + \lambda_4 \mathscr{L}_{\text{mask concentration}} + \lambda_5 \|\phi\|_2^2 \quad (3.60)$$

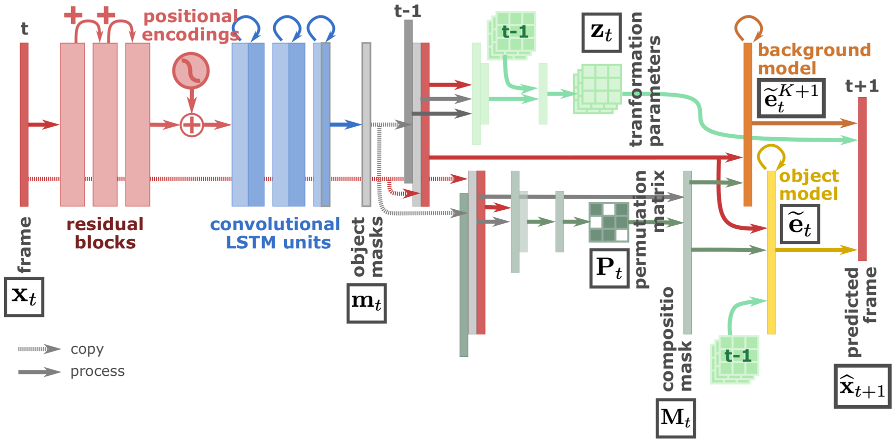


Figure 3.3: Implementation of Model 1 for our unsupervised video object entity abstraction method: the input frame $\mathbf{x}_t$ is first processed by residual blocks, and later fed into convolutional LSTM blocks for autoregressive processing of image features. The resultant object masks $\mathbf{m}_t$ are then ordered by a permutation mask $\mathbf{P}_t$, which is inferred from the masks and input frame. Similarly, the parameters for the predicted motion $\mathbf{z}_t$ are computed autoregressively based on the input frame, masks, and predicted parameters for the previous time step. Finally, the predicted frame is composed from inpainted entities $\tilde{\mathbf{e}}_t$ which are warped with $\mathbf{z}_t$, and the background $\tilde{\mathbf{e}}_t^{K+1}$, with the help of composition mask $\mathbf{M}_t$.

Model 2.

For the second model, we choose to represent each input frame based on the similarity of local, overlapping windows. More precisely, we use sliding windows for processing the input image to extract features $\mathbf{f}_t^{(2)}$ in Equation 3.23. This serves our ultimate task of mask representation well. Hence, we apply the operations of $f_1^{(2)}$ described in Equations (3.22)-(3.25). The initial feature extraction function $f_{1,1}^{(2)}(\cdot)$ in Equation (3.23) is implemented as three residual-CNN blocks (He et al., 2016), where each block is composed of two convolutional layers, each followed by layer normalization (Ba et al., 2016). We use the kernel size of 5 for all convolutional layers and stride size of 2 for the first two blocks, hence downsampling the image resolution 4-times to $H/4 \times W/4$ for the pairwise similarity computation. Upon the computation of the affinity vector using the cosine similarity given in Equation (3.24), we apply a single-layer MLP as $f^{(2)}_{1,2}(\cdot)$ in Equation (3.25) to generate the content vector, also using layer normalization before the activation function. The recursive function Equation (3.24), $f^{(2)}_{1,3}(\cdot)$, which is used for all entity state updates individually, is implemented as an LSTM Unit, where the initial states $\{\mathbf{s}^{\mathbf{k}}_{\mathbf{m},\mathbf{0}}\}_{k=1}^{K}$ are learnable parameters initialized orthogonally across the entities (Saxe et al., 2013). We use a dropout rate of 20% for both recurrent and input connections of the LSTM, as well as the content vector generating MLP during training. The entity states are then reshaped to $H/2 \times W/2$ and the amodal masks are generated by the function $f^{(2)}_{1,4}(\cdot)$ which is implemented as bilinear up-sampling followed by a 3-layer CNN.

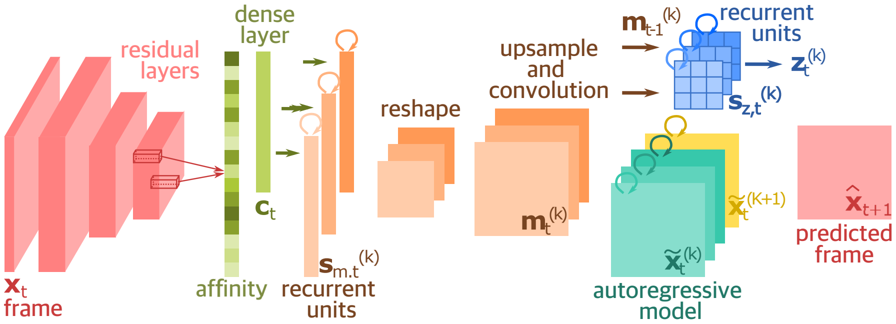


Figure 3.4: Implementation of Model 2 for our unsupervised video object entity abstraction method: the input frame $\mathbf{x}_t$ is first processed by residual blocks with pooling operation, which results in a low-resolution image feature grid. We then compute an "affinity vector" which measures the similarities between all unique pairs of image feature vectors. The frame $\mathbf{x}_t$ is then represented by a single content vector $\mathbf{c}_t$ explicitly describing similarities, and entity-specific mask states $\mathbf{s}^k_{\mathbf{m},t}$ are individually updated based on their previous values and the content vector. Mask states are then mapped to entity masks $\mathbf{m}^k_t$ via upsampling and deconvolution operations. Affine parameters $\mathbf{m}^k_t$ for each entity are computed based on previous values, and entity masks, in order to warp inpainted entities $\tilde{\mathbf{m}}^k_t$ to compose the predicted frame $\hat{\mathbf{m}}^k_{t+1}$

For handling occlusions, we opt for the modulation approach implemented by $f^{(2)}_{m}(\cdot)$ as detailed in Equations (3.31)-(3.33). For learning the modulation parameters $\{\alpha^k_t, \beta^k_t\}_{k=1}^{K+1}$, we follow spatially adaptive normalization (Park et al., 2019b) and use a 3-layer CNN to infer $\{\alpha^k_t, \beta^k_t\}_{k=1}^{K+1} \in \mathbb{R}^{H\times W\times 1}$ based on the input frame and entity masks, $\mathbf{x}_t, \{\mathbf{m}^k_t\}_{k=1}^{K+1}$.

Finally, the inpainting approach and the computation of geometric transformation parameters are the same as their counterparts in Model 1.

We use the loss function $\mathscr{L}^{(2)}_{\text{total}}$ in Equation (3.61) for training the parameters of Model 2 with rmsprop optimizer (Tieleman and Hinton, 2012). Similar to model 1, we set $\lambda_1 = 1$, $\lambda_3 = 0.5$ and use a varying λ_2, which starts from 0.5 and increases to 1 through early training, as we found this model to be more stable against the divergence of predicted motion parameters. The coefficient for the mask sparsity is set to $\lambda_4 = 0.1$. The overall pipeline is briefly illustrated in Figure 3.4.

$$\mathscr{L}^{(2)}_{\text{total}} = \lambda_1 \mathscr{L}_{\text{prediction}} + \lambda_2 \mathscr{L}_{\text{consistency}} + \lambda_3 \mathscr{L}_{\text{background}} + \lambda_4 \mathscr{L}_{\text{mask sparsity}} + \lambda_5 \|\phi\|_2^2 \tag{3.61}$$

3.5 Experimental Results

In this section, we first describe the datasets that are used for studying the performance of our novel representation model. Afterward, we present the experimental results for the two implemented models, separately.

3.5.1 Datasets

In order to understand the strengths and weaknesses of the proposed method, as well as our different implementations, we generate a set of simulated datasets. In addition, we use the modified version of CLEVRER (Yi et al., 2019), which is a dataset used for visual reasoning tasks, to demonstrate the performance of the proposed method in more realistic conditions. In this section, we explain how we generate the datasets and emphasize the key differences between them.

Dataset 1.

To start with, we generate video sequences with $T = 11$ frames of size 64×64 pixels, where K_d objects are initially located in random positions in random quadrants of the image. During the sequence, each object moves from the initial quadrant of the frame towards the diagonally opposite quadrant with a random speed, which induces occlusion later in sequences. The order of objects when they occlude with each other is also chosen randomly. Objects are randomly selected from a shape set of circle, square, triangle with $5^3 - 1$ different color choices. The background color is set to gray for all sequences.

We generate three variants with $K_d \in \{2, 3, 4\}$ objects present in a frame. Each variant contains 10,000 training sequences, 1000 validation sequences, and 1000 test sequences. A sample sequence from each variant can be seen in Figure 3.5.

Dataset 2.

Dataset 2 is a modified version of Dataset 1, where the background color is also chosen randomly. Each variant in Dataset 2 contains 20,000 training sequences, 2000 validation sequences, and 2000 test sequences. We use Dataset 2 for ablation studies to analyze the model performance when the number of entities it assumes, which is represented by K in the problem formulation, matches exactly the number of objects present in a given sequence, or vice versa.

Dataset 3.

Dataset 3 is similar to Dataset 2, however, the number of objects in a frame is randomly chosen from the set $\{2, 3, 4\}$ with equal chance. More precisely, Dataset 3 is composed of a total number

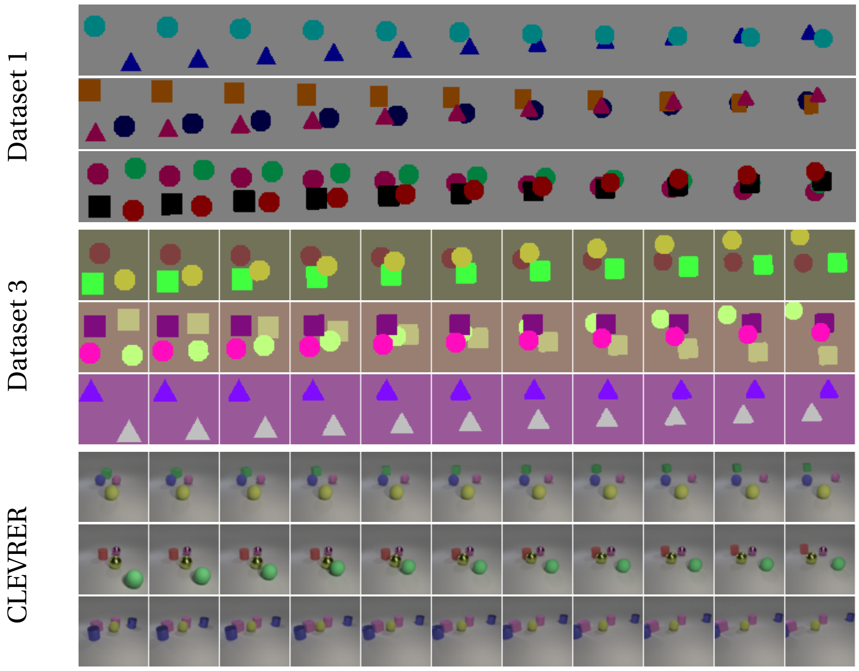


Figure 3.5: Sample sequences from different datasets which are used for training and evaluation of the proposed method.

of 20,000 sequences, where each sequence has an equal chance of containing 2,3 or 4 objects, and each batch is arranged to accommodate sequences with a different number of objects. In addition, motion dynamics for simulating objects are slightly different. The initial position is again chosen randomly in a random quadrant, but in Dataset 3, objects can move towards any other quadrant, which is not restricted to the diagonal opposite as in Dataset 1 and Dataset 2. Three sample sequences from Dataset 3 can be seen in Figure 3.5. Note that we do not visualize samples from Dataset 2 as they are very similar to samples from Dataset 3.

modified-CLEVRER

We use a modified version of the CLEVRER dataset (Yi et al., 2019), which is frequently used for evaluating computational methods for visual reasoning tasks. The original dataset involves videos with 100 frames that contain events of *enter*, *exit*, and *collision* of objects. Objects are simulated with different shapes (cube, sphere, and cylinder) and colors (gray, red, blue, green, brown, cyan, purple, and yellow). We use the videos simulating *collision*, where two objects are involved in the event and others mostly stay static. In order to discard parts of the videos where the objects are all static, and articulate the motion when objects are moving, we create new sequences by taking the middle part of each video, more precisely, between frames 7 and 67. We use every other frame in the given range of the original sequence. We end up having 8760 training sequences of length 11. Three sample sequences from Dataset 3 can be seen in Figure 3.5.

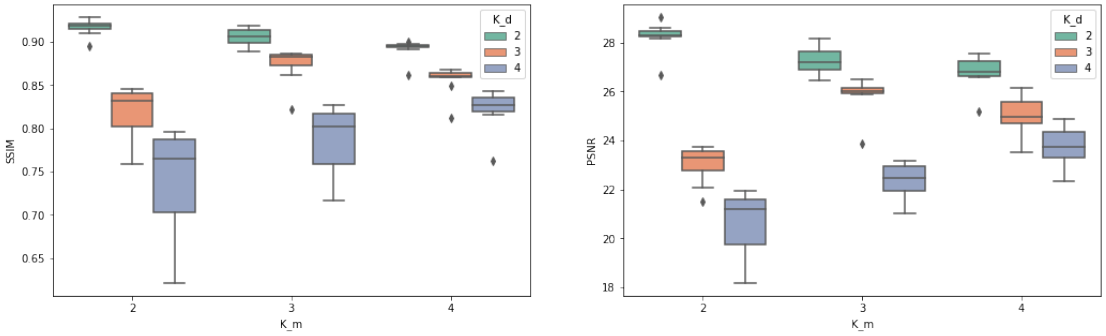


Figure 3.6: The prediction ability of Model I presented as SSIM and PSNR metrics for different numbers of objects in the model (K_m) and in the test dataset (K_d). The variation in each boxplot, *i.e.*, a specific trained model tested on a specific dataset, depicts the variation in prediction quality for each time step, $t \in 1, \dots, 10$ at a given sequence.

3.5.2 Results

In this Section, we present a qualitative and quantitative evaluation of the proposed method, more precisely, Model 1 and Model 2, which are trained and evaluated on the datasets introduced in the previous Section. In fact, we train and evaluate Model 1 only on Dataset 1 because our attempts to train it on other datasets failed despite the considerable amount of effort. We then present the corresponding study of Model 2 on Dataset 2 to analyze the effect of the most important hyperparameter K, which indicates the number of objects. We then conclude with the evaluation of Model 2 on Dataset 3 and modified ClEVRER datasets.

For evaluation, we report the reconstruction quality of the predicted frame in terms of peak-to-noise signal ratio (PSNR) and structural similarity (SSIM) metrics, whose implementations are detailed in Appendix A.

Model 1.

We train three instances of Model 1, whose implementation is detailed in Section 3.4. Each trained model delivers a different number of $K_m = K \in \{2, 3, 4\}$ object representations, and these models are trained with variants of Dataset 1 with $K_d = K_m$ objects. We then evaluate the performance of each instance on all variants of Dataset 1. Consequently, in some cases, the number of objects the model can represent is equal to or greater than the number of objects present in the given video sequences. In this case, we expect our model to decouple each moving entity and provide a representation that can model the support of the corresponding object in the 2D image plane. However, for the other cases when $K_m < K_d$, the model cannot decouple the entities perfectly due to its construction.

We are interested in such a study because increasing the model capacity for representing more objects leads to a more challenging problem due to arising cases with heavy occlusions. On the other hand, a model with very few object representations can perform to a limited extent.

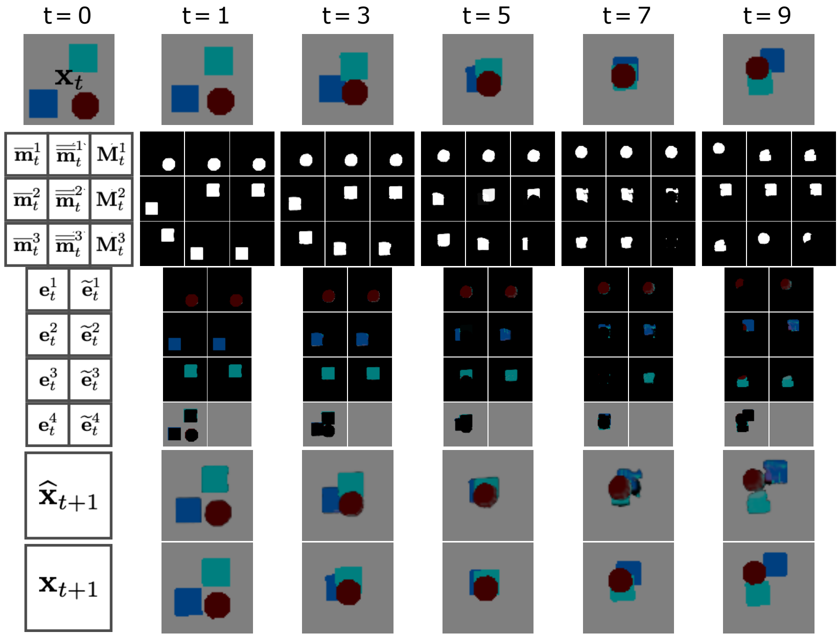
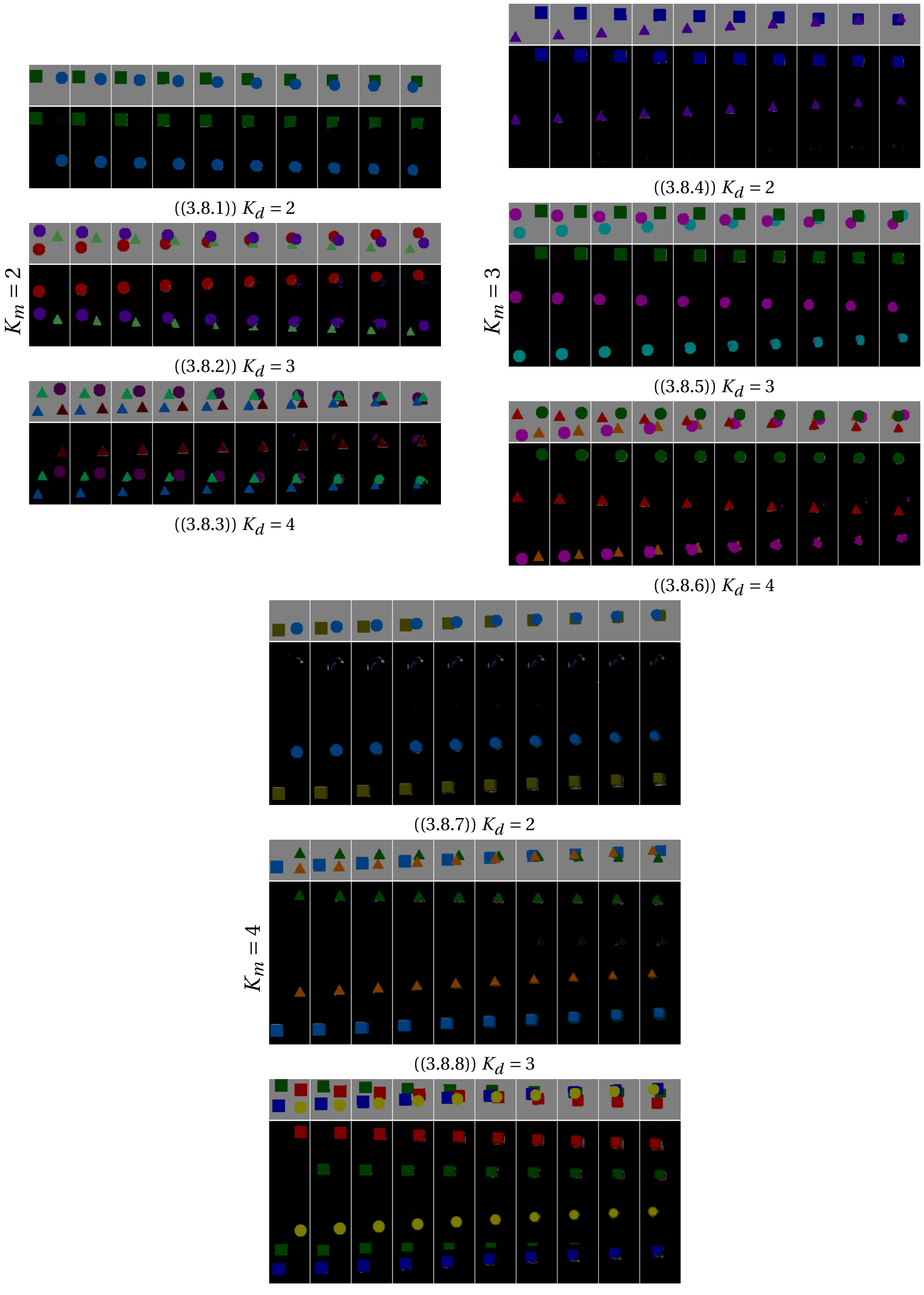

t = 0
t = 1
t = 3
t = 5
t = 7
t = 9
$\mathbf{x}_t$
$\overline{\mathbf{m}}_t^1$ $\overline{\overline{\mathbf{m}}}_t^1$ $\mathbf{M}_t^1$
$\overline{\mathbf{m}}_t^2$ $\overline{\overline{\mathbf{m}}}_t^2$ $\mathbf{M}_t^2$
$\overline{\mathbf{m}}_t^3$ $\overline{\overline{\mathbf{m}}}_t^3$ $\mathbf{M}_t^3$
$\mathbf{e}_t^1$ $\tilde{\mathbf{e}}_t^1$
$\mathbf{e}_t^2$ $\tilde{\mathbf{e}}_t^2$
$\mathbf{e}_t^3$ $\tilde{\mathbf{e}}_t^3$
$\mathbf{e}_t^4$ $\tilde{\mathbf{e}}_t^4$
$\hat{\mathbf{x}}_{t+1}$
$\mathbf{x}_{t+1}$

Figure 3.7: Illustrative output for a test sequence with object masks after binarization, $\overline{\mathbf{m}}_t^k$, and ordering, $\overline{\overline{\mathbf{m}}}_t^k$; occlusion mask, $\mathbf{M}_t^k$, objects and background before and after inpainting, $\{\mathbf{e}_t^k\}_{k=1}^{K+1}$ and $\{\tilde{\mathbf{e}}_t^k\}_{k=1}^{K+1}$ respectively, as well as the ground-truth and predicted frames $\mathbf{x}_{t+1}$ and $\hat{\mathbf{x}}_{t+1}$, for a sequence and trained model with $K_d = K_m = 3$.

Figure 3.6 illustrates our related findings. We present the reconstruction quality of the predicted sequences on a test set of 1000 simulated videos. More precisely, presented PSNR and SSIM values are computed between $\{\mathbf{x}_t\}_{t=2}^T$ and $\{\hat{\mathbf{x}}_t\}_{t=2}^T$, and the variance for each entry in the boxplot demonstrates the variance of the prediction quality across time. As expected, the model is capable of representing the objects well as long as $K_d \leq K_m$, where the deterioration in performance when $K_d > K_m$ is due to limited representation capacity. Moreover, the performance slightly degrades with a larger number of objects when $K_d = K_m$, due to the increasing complexity arising from heavy occlusions.

In Figure 3.7, we visually probe the model outcomes in detail. We illustrate binarized amodal masks $\overline{\mathbf{x}}_t^k$, and the output when they are ordered by the predicted permutation matrix, $\overline{\overline{\mathbf{x}}}_t^k$, which can model the accurate ordering at time $t = 5$ and $t = 7$. Even when occlusions happen, our model can represent the full support of each object, as it can be particularly seen for time instances $t = 5$ and $t = 7$. Furthermore, the inpainted entities, denoted by $\tilde{\mathbf{x}}_t^k$ can retain the appearance information reasonably well. The comparison between the predicted frame and the ground truth, which can be seen at the bottom of the Figure, indicates the successful prediction of approximate motion parameters.

We present more visual samples demonstrating the model outcomes, namely the inpainted entities, $\tilde{\mathbf{x}}_t^k$, in Figure 3.8.



((3.8.1)) $K_d = 2$

((3.8.2)) $K_d = 3$

((3.8.3)) $K_d = 4$

((3.8.4)) $K_d = 2$

((3.8.5)) $K_d = 3$

((3.8.6)) $K_d = 4$

((3.8.7)) $K_d = 2$

((3.8.8)) $K_d = 3$

((3.8.9)) $K_d = 4$

Figure 3.8: Test sequences $\{\mathbf{f}_t\}_{t=1}^{10}$ with inferred objects $\{\widetilde{\mathbf{e}}_t^{(k)}\}_{t=1}^{10}$ for different K_m and K_d. Each row of images correspond to a different K_m, a single model, whereas columns represent the test set with different number of objects, K_d.

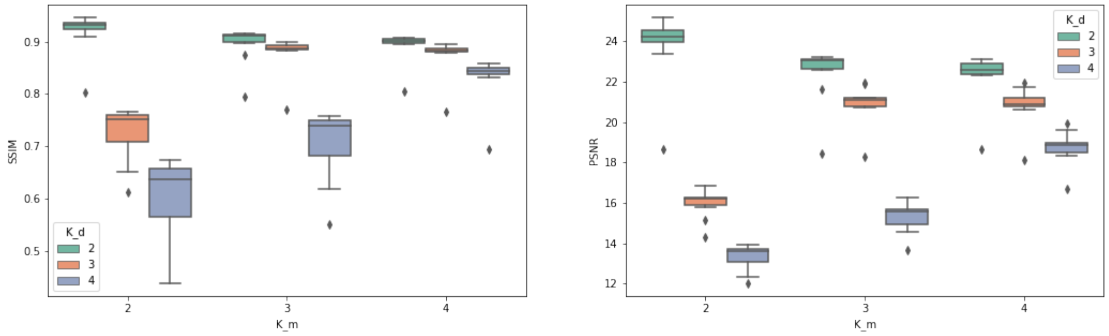


Figure 3.9: The prediction ability of Model 2 presented as SSIM and PSNR metrics for different numbers of objects represented by the model (K_m) and present int the dataset (K_d). The variation of each boxplot, *i.e.*, performance of a specific trained model on a specific dataset, depicts the variation in prediction quality for each time step, $t \in \{1, \ldots, 10\}$ at a given sequence.

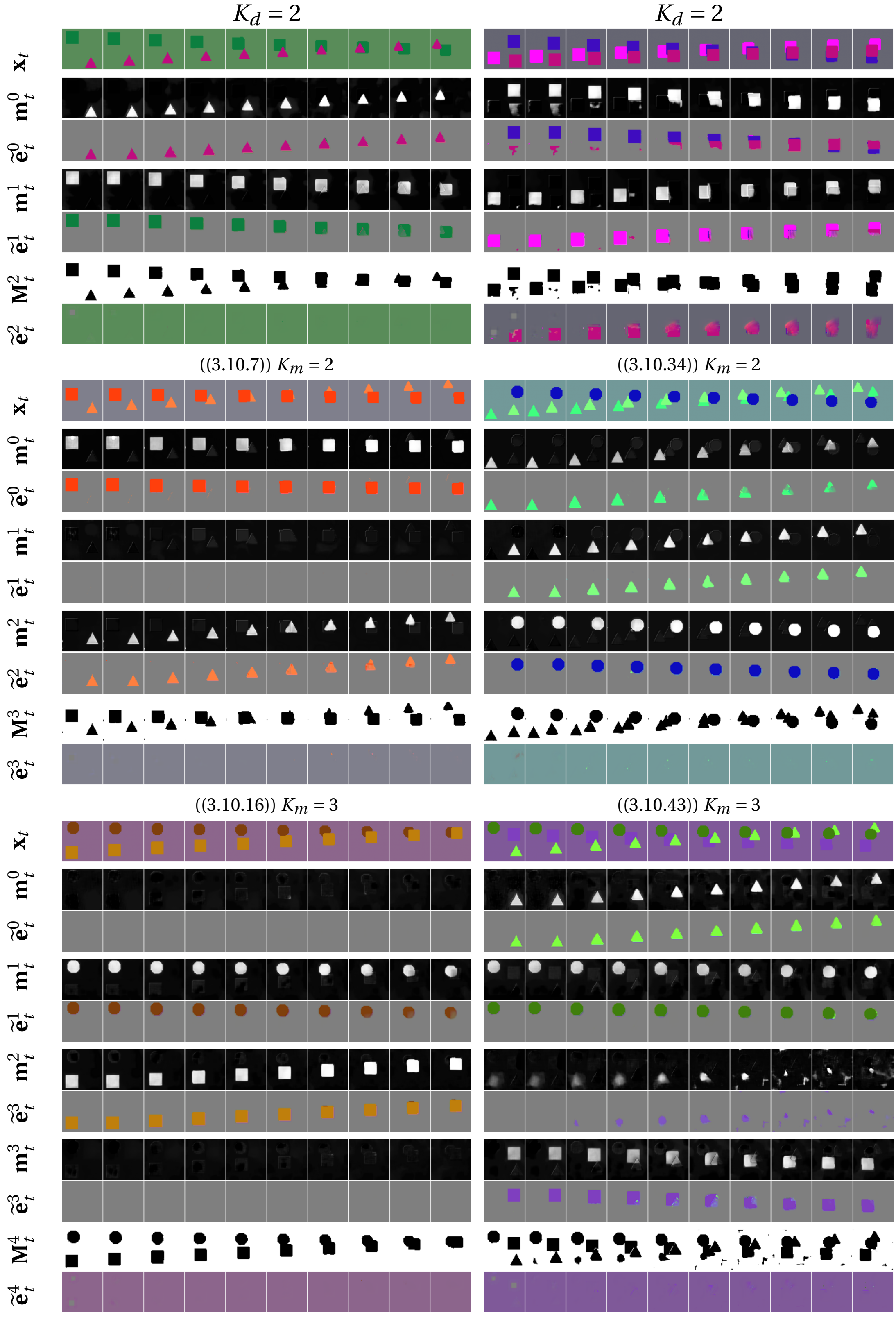
Model 2.

We continue with the evaluation of Model 2. We start with the results of the same experiment for Model 2 by using Dataset 2, which is slightly more challenging than Dataset 1 due to the varied background color. In fact, modeling gray background with Model 2 is trivial as this implementation operates on normalized images such that $\mathbf{x}_t^k \in [-1,1]^{H \times W \times 3}$, and gray background corresponds to an image of all zero values. Hence, we find it fairer to evaluate the inpainting capability of the proposed model and implementation on sequences with a colored background.

Figure 3.9 depicts our related findings, which agree with the results previously presented in Figure 3.6 for Model 1. Implementation Model 2, hence, provides more stable, and also faster, training while not compromising performance.

We further illustrate some visual outcomes of Model 2 in Figure 3.10. This time, we present inferred amodal masks, $\mathbf{m}_t^k$, accompanied with associated entities, $\tilde{\mathbf{x}}_t^k$. The entities are visualized on a gray background due to the normalization operation, as mentioned earlier. The results prove that our method is capable of decoupling entities and retaining relevant information despite the low contrast between entities and the background, as in cases with $K_d = 2; K_m = 2$ and $K_d = 3; K_m = 4$; similar-looking entities, such as the case $K_d = 3; K_m = 3$; or under heavy occlusion as in the case with $K_d = K_m = 4$. Even when the model capacity is limited to represent entities, like the case with $K_d = 4; K_m = 2$, the model performs reasonably well by decoupling some of the moving entities and attributing the others to the background, and blurring the entities represented in the background to satisfy the background constraints dictated by $\mathscr{L}_{\text{background}}$.

Lastly, Figure 3.11 presents the visual results from the test set of the modified CLEVRER dataset. We can observe that our model represents the object entering the scene and moving to perform the *collision* event successfully in all cases. The second object which is involved in the *collision*



((3.10.7)) $K_m = 2$

((3.10.34)) $K_m = 2$

((3.10.16)) $K_m = 3$

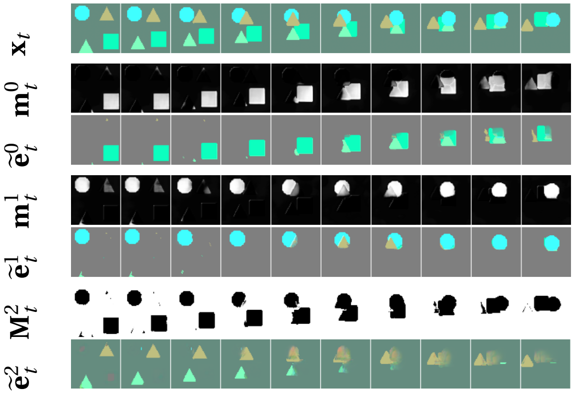
((3.10.43)) $K_m = 3$

((3.10.27)) $K_m = 4$

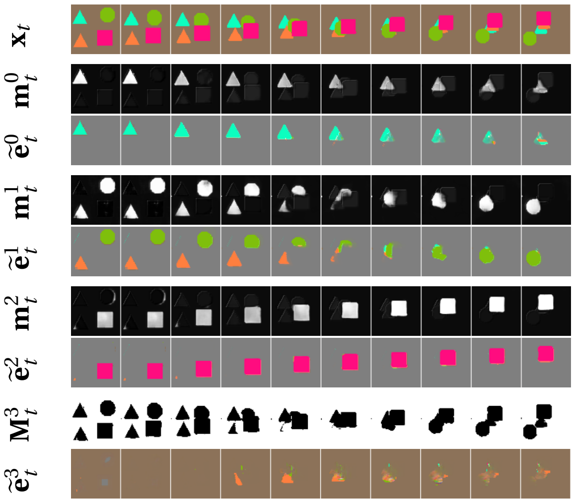
((3.10.54)) $K_m = 4$

Figure 3.10: Test sequences $\{\mathbf{x}_t\}_{t=1}^{10}$ with inferred (inpainted) objects, $\{\tilde{\mathbf{e}}_t^k\}_{t=1}^{10}$, amodal masks, $\{\mathbf{m}_t^k\}_{t=1}^{10}$, and the background mask, $\{\mathbf{M}_t^{K_m+1}\}_{t=1}^{10}$, for $K_m \in \{2,3,4\}$ and $K_d \in \{2,3\}$. Each row of clustered images correspond to a different K_m, *i.e.*, a single trained model, whereas columns represent the test set with different number of objects, K_d.

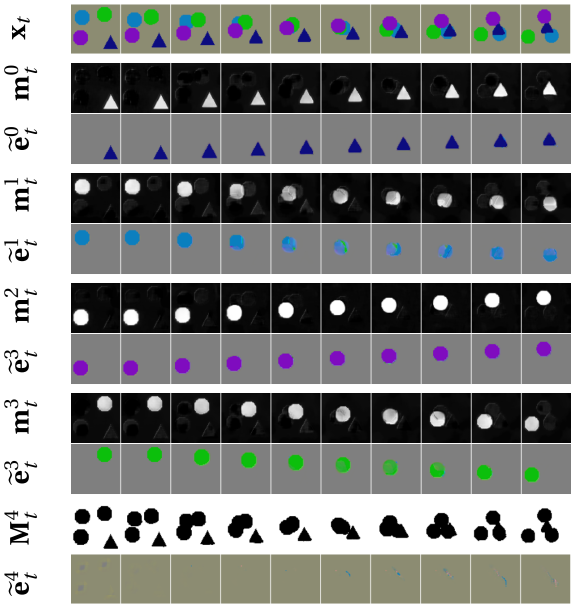
$K_d = 4$



((3.10.61)) $K_m = 2$

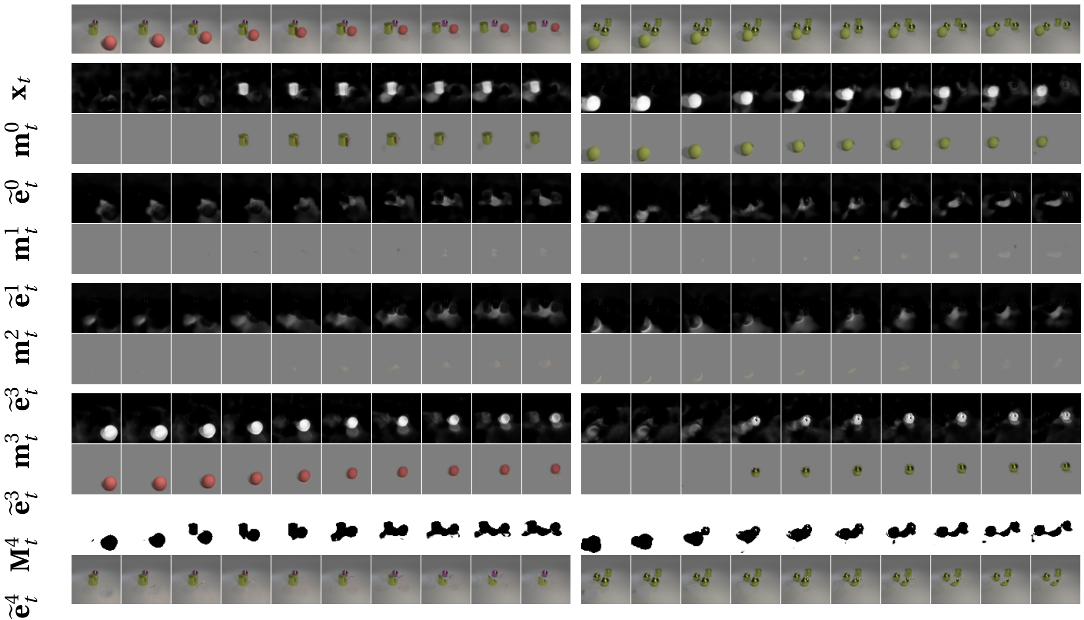


((3.10.70)) $K_m = 3$

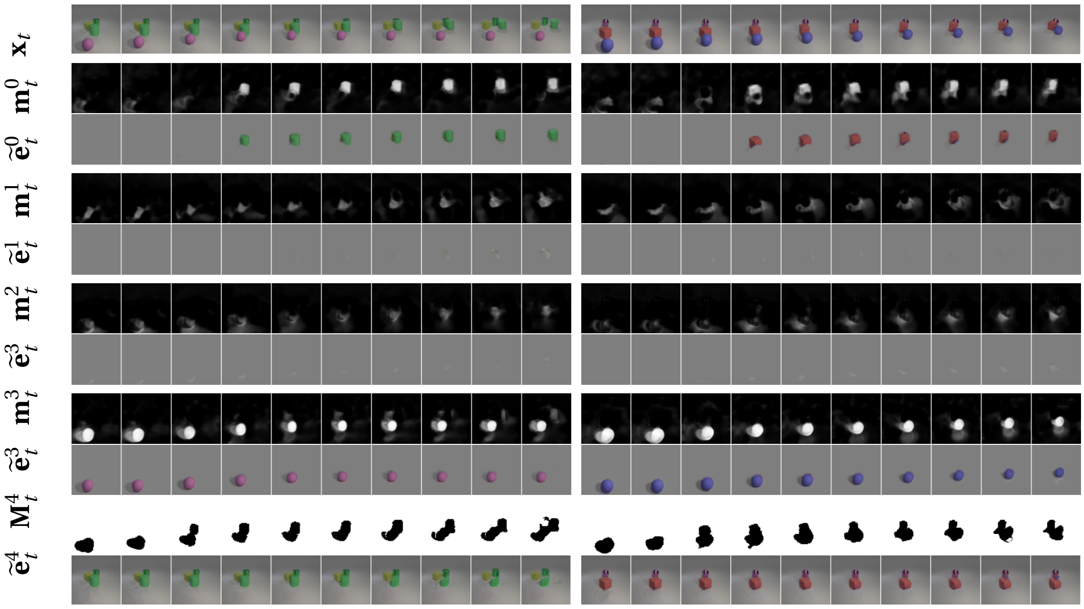


((3.10.81)) $K_m = 4$

Figure 3.10: (continued) $\{\mathbf{x}_t\}_{t=1}^{10}$ with inferred (inpainted) objects, $\{\tilde{\mathbf{e}}_t^k\}_{t=1}^{10}$ amodal masks, $\{\mathbf{m}_t^k\}_{t=1}^{10}$, and the background mask, $\{\mathbf{M}_t^{K_m+1}\}_{t=1}^{10}$, for $K_m \in \{2,3,4\}$ and $K_d = 4$. Each row of clustered images correspond to a different K_m, *i.e.*, a single trained model.


$\mathbf{x}_t$
$\mathbf{m}_t^0$
$\widetilde{\mathbf{e}}_t^0$
$\mathbf{m}_t^1$
$\widetilde{\mathbf{e}}_t^1$
$\mathbf{m}_t^2$
$\widetilde{\mathbf{e}}_t^3$
$\mathbf{m}_t^3$
$\widetilde{\mathbf{e}}_t^3$
$\mathbf{M}_t^4$
$\widetilde{\mathbf{e}}_t^4$

((3.11.11)) Sample 1 ((3.11.33)) Sample 3


$\mathbf{x}_t$
$\mathbf{m}_t^0$
$\widetilde{\mathbf{e}}_t^0$
$\mathbf{m}_t^1$
$\widetilde{\mathbf{e}}_t^1$
$\mathbf{m}_t^2$
$\widetilde{\mathbf{e}}_t^3$
$\mathbf{m}_t^3$
$\widetilde{\mathbf{e}}_t^3$
$\mathbf{M}_t^4$
$\widetilde{\mathbf{e}}_t^4$

((3.11.22)) Sample 2 ((3.11.44)) Sample 4

Figure 3.11: Test sequences $\{\mathbf{x}_t\}_{t=1}^{10}$ with inferred (inpainted) objects, $\{\widetilde{\mathbf{e}}_t^k\}_{t=1}^{10}$, amodal masks, $\{\mathbf{m}_t^k\}_{t=1}^{10}$, and the background mask, $\{\mathbf{M}_t^{K_m+1}\}_{t=1}^{10}$, for the modified CLEVRER dataset.

event is usually detected around the collision time, and otherwise, represented to be the part of a static object. Other slots seem to model shadows that change over time due to the motion of objects. In addition, the background gets inpainted reasonably well. Although the second object, which starts moving after the collision, is still depicted as a part of the background in order to maintain background similarity across different time steps, the background mask discards those parts as a result of zero values at corresponding pixel locations.

3.6 Conclusion

In this Chapter, we proposed a novel method for representing videos in an object-centric manner. Contrary to existing methods, we formulate the problem with a prediction objective. We decompose each video frame into a pre-defined number of entities. Each entity is described for its visible and occluded parts with the help of amodal masks and inpainting operations. In the proposed framework, we approximate the motion of each entity by geometric transformations and predict entity-specific transformation parameters in order to characterize their motion. After warping each amodal mask to a different time instance using the associated motion parameters, we compute a visibility mask for the predicted time step. We also warp each inpainted entity using the same approximate motion parameters. The predicted frame is finally synthesized by masking the warped entities with the predicted visibility mask.

This end-to-end differentiable model is trained with a combination of single-step and cyclic multi-step prediction objectives. Our novel consistency loss term forces the model to predict a frame that is multiple time steps ahead of the input frame. The inverse of the approximation for this multiple-step motion is used to perform the prediction backward in time between the two time instances at hand. Training is further supported by auxiliary loss terms that promote spatial compactness and/or sparsity of amodal masks. Experimental results proved that our formulation can decouple moving entities of a given video sequence without any manual annotations. We empirically found the most critical constraint for meaningful entity abstraction is the cyclic consistency term; however, each loss term as well as their weight in the final training objective is still very important for the final outcome. The model successfully infers the amodal masks that represent the moving entities despite challenging conditions, such as low contrast between background and entities, heavy occlusion, or high visual similarity between entities. We believe that our model can be translated to more realistic datasets by using pre-trained networks for feature extraction and increasing the representation capacity of other functions to handle more sophisticated object shapes and motion patterns.

4 Object-centric Video Prediction

4.1 Introduction

Video prediction can be formulated under both deterministic and stochastic frameworks. Deterministic frameworks refer to models which can output only one prediction outcome when they are given a sequence of images. If the problem is defined to be the next(single)-frame prediction, that outcome is a single frame. Otherwise, these models output a sequence of video frames that are a plausible continuation of the input sequence. On the other hand, stochastic frameworks estimate a distribution for the outcome, which can be again designed for a single frame or a sequence of frames. One can then sample different prediction results from the distribution estimated by the stochastic model.

The framework presented in Chapter 3 for entity abstraction already has the ability for video prediction by its construction. Indeed, given a sequence of frames $\{\mathbf{x}_t\}_{t=1}^{T_1}$, one of the model's outcomes is the predicted frame $\widehat{\mathbf{x}}_{T_1+1}$. Following an autoregressive approach, *i.e.*, feeding $\left\{\{\mathbf{x}_t\}_{t=1}^{T_1}, \widehat{\mathbf{x}}_{T_1+1}\right\}$, we can obtain $\widehat{\mathbf{x}}_{T_1+2}$, hence, any sequence of length T_2 by repeating this autoregressive prediction step T_2 times. However, the predicted sequence $\{\widehat{\mathbf{x}}_t\}_{t=T_1+1}^{T_1+T_2}$ is always the same for the same input sequence of $\{\mathbf{x}_t\}_{t=1}^{T_1}$, which does not fully address the probabilistic nature of the prediction problem. Figure 4.1 depicts such deterministic prediction results with Model 2, which is explained in Section 3.5.2. The predicted frames before the red line, *i.e.*, $t = 5$, exemplifies next-frame prediction outcomes, whereas the predictions after the red line are obtained in an autoregressive manner.

In this Chapter, we extend our entity abstraction pipeline from Chapter 3 towards short-term stochastic object-centric video prediction as an attempt to bridge the gap between object-centric deterministic prediction approaches and stochastic frameworks. We believe that the object appearances are less likely to change in short windows of time, hence, we attribute the stochasticity of the problem only to the motion of entities, and treat motion parameters as random variables. Consequently, instead of directly predicting the approximated motion of each entity in the next frames, we estimate a conditional distribution for the geometric

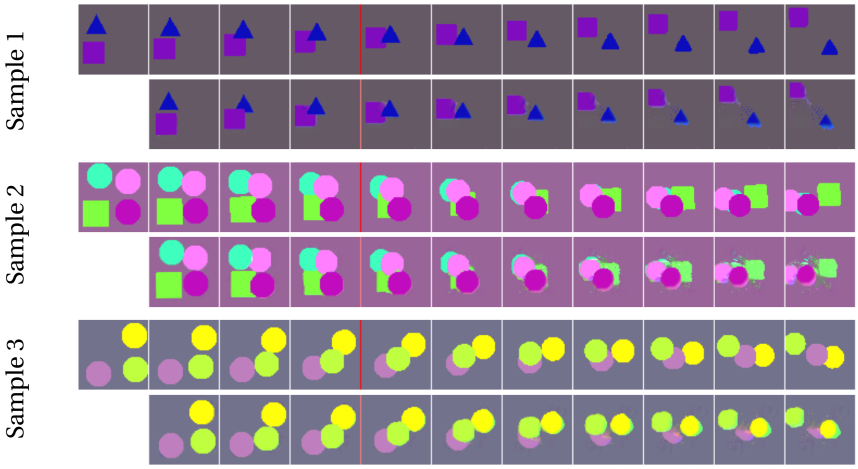


Figure 4.1: Deterministic prediction based on Model 2 trained with Dataset 3. First row in each sample depicts the input sequence, $\{\mathbf{x}_t\}_{t=1}^{11}$, whereas the second row demonstrates the predicted frames $\{\widehat{\mathbf{x}}_t\}_{t=2}^{11}$. The red column after $t = 4$ indicates the start of autoregressive processing, *i.e.*, the predicted frame at time $t-1$, $\widehat{\mathbf{x}}_{t-1}$, is fed into the model as the input frame after $t = 5$.

transformation parameters that model the motion individually for each entity. Then, at each time step, we sample the transformation parameters from the proposed entity-specific distribution. In other words, by estimating the distribution of entity-specific motion parameters for predicted frames, we can sample different predicted sequences, and model the prediction problem under uncertainty.

We then propose two different methods for the estimation of the such conditional distribution, where the parameters depend on their values in the past. We try to cover two main approaches in generative modeling for a comparative study of our approach as both approaches are known to suffer from different issues for modeling distributions. The first one stems from state-space models, and it is formulated within a graphical model based on the dependencies between model variables. We propose a principled approach for approximating the motion distribution present in a given dataset based on the constructed graphical model. The second method follows an adversarial approach based on Wasserstein Generative Adversarial Networks (Arjovsky et al., 2017). In this case, we propose a three-stage training scheme for better representing the motion distribution.

This chapter is organized as follows. We review generative models with an emphasis on Variational Autoencoders (VAEs) (Kingma and Welling, 2013) and Wasserstein Generative Adversarial Networks (WGANs) (Arjovsky et al., 2017), which constitute the bases of our proposed methods. We then present our problem formulation and introduce the methods for the estimation of the time-varying conditional distribution of motion parameters. After demonstrating the results of both methods on illustrative examples, we conclude this chapter with our remarks on the object-centric stochastic video prediction problem.

4.2 Background

In this section, we visit two well-known families of generative models, namely, Variational Encoders (VAEs) and Wasserstein Generative Adversarial Networks (WGANs), which are among the most favored approaches for image generation.

Generative models aim to capture a data distribution, which can be only observed via a limited number of data samples. More formally, generative models assume that the observations $x_1, x_2, \ldots, x_n$ are independent and identically distributed (*i.i.d.*) samples from an underlying true distribution for data, which can be represented by $p(X)$. Note that we use capital letters, e.g., X, for random variables, and lowercase letters, e.g., x, for their values. The goal of generative models is to produce a distribution $q(X)$ based on the data samples which minimizes a divergence[1] between the true and modelled distributions, $D_\gamma(p|q)$, formulated in Equation (4.1), where the function γ depends on the algorithm.

$$D_\gamma(p\|q) = \mathbb{E}_{x \sim p(X)}\left[\gamma\left(\frac{q(x)}{p(x)}\right)\right] \tag{4.1}$$

Some algorithms(MacKay, 1992) parametrize the approximate distribution by θ, *i.e.*, $q(x;\theta)$, such that it belongs to a probability family which has nice properties and makes solving the problem more tractable. However, finding the right distribution family for the problem at hand is not trivial either. There is a trade-off between its expressive power and specificity; it needs to be expressive enough to represent the data samples well while being specific enough so that it does not require too many data samples, or compute power, for model construction.

To facilitate the choice of distributions, generative models, such as Variational Autoencoders (VAEs) (Kingma and Welling, 2013) or Generative Adversarial Networks (GANs) (Goodfellow et al., 2014a), formulate the problem using latent variables that can model the underlying factors of variation in data. The latent variable z is defined by the distribution $p(Z)$, and it is mapped to the input space by a deterministic generator function, $g(\cdot)$. Hence, the representation capacity of the generative model is jointly defined by $p(Z)$ and $g(\cdot)$. Then, generative modeling can be formulated as in Equation (4.2), where the generator function $g(z;\theta)$ is represented by the distribution $p(x|z;\theta)$ which makes the dependence of x on z explicit.

$$p(x) = \int p(x|z;\theta)p(z)dz. \tag{4.2}$$

This formulation translates to sampling z according to the distribution $p(Z)$ and passing the sampled values through the generator function $g(z;\theta)$ to model the data distribution $p(X)$.

---

[1] D is a divergence between two variables p, q if it satisfies the following two properties:

- $D(p\|q) >= 0$
- $p = q \Leftrightarrow D(p,q) = 0$.

These samples can be considered as generated samples and are denoted by $\hat{x}$. The minimization of the divergence in Equation (4.1) is then achieved by using the pairs $\{(x_i, \hat{x}_i)\}_{i=1}^n$, where n represents the number of data samples used for computing the expectation value in Equation (4.1).

In this section, we briefly review Variational Autoencoders and Wasserstein Generative Adversarial Networks (Wasserstein GANs or WGANs), which are the two methods that we use for representing time-varying geometric transformation parameter distributions in our object-centric video prediction formulation. Briefly, VAEs and WGANs mostly differ in the formulation of the divergence to be minimized, as well as in the methods used for the constraining of the assumed latent distribution $p(Z)$.

4.2.1 Variational Autoencoders

Two main problems arise from the generative formulation in Equation (4.2), which are the characterization of the latent variable z and the integral over z. Variational Autoencoders (Kingma and Welling, 2013) propose to characterize the distribution $p(Z)$ by a simple distribution, such as unit-variance Gaussian $\mathcal{N}(0, I)$ where I denotes the identity matrix. However, sampling any z from a simple distribution does not guarantee that the corresponding generated sample $\hat{x}$ establishes high similarity to real data samples x. Hence, modeling the data distribution would require sampling too many latent variables, and very few of them can contribute to the estimation of $p(X)$. VAEs attempt to make sampling of the latent variables more efficient via an additional distribution $q(X|z;\phi)$, which models the z values that generate samples that are more likely to belong to the true data distribution $p(X)$ (Doersch, 2016). They try to achieve this goal by minimizing the Kullback-Leibler divergence[2] between the distributions $p(Z)$ and $q(Z|X)$. If the parametrization of the generative process $p(X|Z;\theta)$ is denoted by $p_\theta(X|Z)$ and the encoding process $q(X|Z;\phi)$ by $q_\phi(X|Z;\phi)$, the learning process of VAEs boils down to the minimization of the Evidence Lower Bound (ELBO) expressed in Equation (4.3) for the observed data samples $\{x_i\}_{i=1}^n$.

$$\mathscr{L}_{\mathrm{VAE}}(\theta, \phi; X) = \sum_{i=1}^{n} -D_{\mathrm{KL}}(q_\phi(z|x) \| p_\theta(z)) + \mathbb{E}_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] \tag{4.3}$$

The first term in Equation (4.3) can be described by a closed form for a set of distributions. The second term, on the other hand, is usually interpreted as the reconstruction quality of the generated samples obtained by decoding the sampled latent variables. The expectation value is computed over the data samples in the training set, where each data point in the training set is considered to be the desired outcome for a generated sample. VAEs implement the parametrizations θ and ϕ as neural networks, which can be learned with gradient-based methods. We need to sample latent variables $z \sim q_\phi(Z|X)$ for the computation of the second

---

[2] $D_{\mathrm{KL}}(p\|q)) = -\sum_{x \sim \mathscr{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right)$ for a discrete random variable X with support $\mathscr{X}$.

term in Equation (4.3). However, backpropagation is not possible through the sampling operation. As a solution, VAEs propose an operation called "reparametrization trick" to alleviate the sampling problem (Kingma and Welling, 2013). As they formerly define $p(Z)$ to follow a simple distribution, they separate the sampling and use neural networks to estimate the parameters of the chosen simple distribution. For example, if the distribution is chosen to be a multivariate Gaussian, the approximate distribution $q_\phi(Z|X)$ can be described by the mean vector $\mu(x)$ and the covariance matrix $\Sigma(x)$, which are inferred from the input data x. Then, sampling $z \sim \mathcal{N}\left(\mu(x), \Sigma(x)\right)$ translates to sampling $\epsilon \sim \mathcal{N}(0, I)$ and computing $z = \mu(x) + \Sigma^{\frac{1}{2}}(x) * \epsilon$. In this way, the model can use a gradient descent algorithm to learn neural network parameters θ, ϕ so that the network parametrized by ϕ maps the input to the latent variables, and the network parametrized by θ maps the sampled latent variables to the input space.

4.2.2 Wasserstein GANs

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014a) represent another popular choice for the unconditional generation of new data samples. Briefly, they use the Jensen-Shannon divergence, $D_{\text{JS}}(\cdot\|\cdot)^3$, for the problem defined in Equation (4.2). To alleviate the need for integration for the computation of chosen divergence, they work with its dual form (Nowozin et al., 2016), which converts the integral into a sampling operation. Hence, the generator function for GANs can be learned via playing a min-max game given in Equation (4.4). The function $\sigma(\cdot)$ in Equation (4.4) indicates the $sigmoid$ operator, and the function $d(\cdot)$ is often referred to as discriminator. The last term is sometimes modified in different implementations to solve numerical problems.

$$g, d = \text{arg}\min_g \text{arg}\max_d \mathbb{E}_{x\sim p(x)}[\log \sigma(d(x))] + \mathbb{E}_{x\sim q(x|z))}[\log(1-\sigma(d(x))] \tag{4.4}$$

However, GANs inherit a well-known problem of training instability (Gulrajani et al., 2017; Tolstikhin et al., 2017). The cause of this instability is commonly attributed to the discontinuity of Jensen-Shannon (JS) divergence with respect to generator parameters θ. Wasserstein GANs (Arjovsky et al., 2017), thus, propose to minimize the Wasserstein distance $W(p,q)^4$, which can circumvent the issues related to the discontinuities of the JS divergence. In particular, WGANs minimize the divergence emerging from Kantorovich-Rubinstein duality (Villani, 2009), which is given in Equation (4.5). The function $c(\cdot)$ is usually referred to as the critic function, and the constraint $\|c\|_L \leq 1$ means that the function $c(\cdot)$ belongs to the family of L-Lipschitz functions.

$$D_W(p,q) = \max_{c, \|c\|_L \leq 1} \mathbb{E}_{x\sim p(x)}[c(x)] - \mathbb{E}_{x\sim q(x|z)}[c(x)] \tag{4.5}$$

---

[3] $D_{\text{JS}}(p\|q) = \frac{1}{2}D_{\text{KL}}(p\|q) + \frac{1}{2}D_{\text{KL}}(q\|p)$

[4] p-Wasserstein distance between two probability measures μ, ν: $W_p(\mu,\nu) = \left(inf_{\gamma\in\Gamma(\mu,\nu)} \int c(x,y)^p c\gamma(x,y)\right)^{\frac{1}{p}}$, where $c(x,y)$ is a distance metric, γ denotes a coupling

Mitigating the problems related to the JS divergence by opting for Wasserstein distance, which is continuous and almost differentiable everywhere, WGAN targets more stable training and better modeling for the data distribution. The constraint $\|c\|_L \leq 1$ has been tackled by different approaches, such as weight clipping (Arjovsky et al., 2017), gradient penalty (Gulrajani et al., 2017) and spectral normalization (Miyato et al., 2018). Among these, the gradient penalty method is found to be the most promising implementation as it achieves better stability than weight clipping while providing more representation flexibility compared to spectral normalization (Su, 2018). The gradient penalty replaces the constraint $\|c\|_L \leq 1$ with the norm of gradients $\|\Delta_x c(\cdot)\|$ and the min-max game for learning the critic function $c(\cdot)$ and the generator function $g(\cdot)$ results in Equations (4.6)-(4.7). The expression $pq(x)$ in Equation (4.6) represents a random linear interpolation of $p(x)$ and $q(x|z)$, and λ is a hyperparameter. This algorithm is also denoted by WGAN-GP.

$$c = \arg\max_c \mathbb{E}_{x\sim p(x)}[c(x)] - \mathbb{E}_{x\sim q(x|z)}[c(x)] - \lambda \mathbb{E}_{x\sim pq(x)}[(\|\Delta_x c(x)\| - 1)^2] \tag{4.6}$$

$$g = \arg\max_g \mathbb{E}_{x\sim q(x|z)}[-c(x)] \tag{4.7}$$

Such formulation indicates a two-step update of overall model parameters. In the first step, parameters of the critic function $c(\cdot)$ are updated according to Equation (4.6). It is then followed by the update of the generator function, $g(\cdot)$, parameters based on Equation (4.7). In practice, the first operation is repeated multiple times before updating the generator function.

In summary, WGANs use Wasserstein distance instead of JS divergence by learning a continuous-valued critic function to mitigate the instability problem of training generative adversarial networks. The gradient penalty further improves the training stability by avoiding the vanishing- or exploding-gradient issues that may arise from other forms of constraints to ensure that the critic function, $c(\cdot)$, is a L-Lipschitz function. On the other hand, similar to other variants of generative adversarial networks, once trained, WGANs generate data from sampled noise. In the conditional version, the noise input is supplemented by a latent variable, which enables more control over the generated output. Hence, by using the history of geometric transformation parameters as conditional input, we aim to learn a WGAN model that lets us sample new parameters for the prediction of video frames.

4.3 Proposed Approach

In this Section, we first remind the dynamics of our deterministic entity abstraction method developed in Chapter 3. We then pose the prediction problem in a stochastic framework, where we treat the motion parameters as random variables. For training the stochastic framework, we propose two different methods; the first one involves the VAE formulation and suggests a graphical model. This model is similar to that of SRNN (Fraccaro et al., 2016) for approximating the posterior distribution of motion parameters, which is needed for stochastic prediction problem. For the second method, we adopt the WGAN framework and propose a three-stage learning scheme to better capture the motion distribution in a given dataset.

4.3.1 Object-centric Video Prediction

As detailed in Chapter 3, our entity abstraction method consists in:

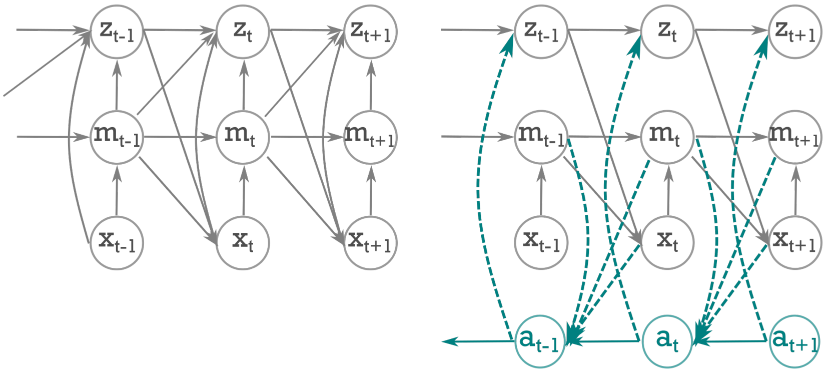
- Decomposition of the input frame $\mathbf{x}_t$ into K entities and background, $\{\mathbf{e}_t^k\}_{k=1}^{K+1}$, by the help of amodal entity masks $\{\mathbf{m}_t^k\}_{k=1}^{K}$, where $\mathbf{m}_t^k = f_1(\mathbf{x}_t, \mathbf{m}_{t-1}^k)$
- Inpainting of each entity such that $\widetilde{\mathbf{e}}_t^k = g_e\left(\mathbf{e}_t^k, \widehat{\mathbf{e}}_t^k\right)$ for $k = 1, \ldots, K$, where $\widehat{\mathbf{e}}_t^k$ denotes the entity predicted at time $t-1$
- Inpainting of the background $\widetilde{\mathbf{e}}_t^{K+1} = g_b\left(\mathbf{e}_t^{K+1}, \widetilde{\mathbf{e}}_{t-1}^{K+1}\right)$,
- Prediction of parameters of an approximate motion model, denoted by $\mathbf{z}_t^k$, such that $\mathbf{z}_t^k = f_z(\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k, \mathbf{z}_{t-1}^k)$,
- Warping each inpainted entity and amodal mask to the time step $t+1$, represented by $\widehat{\mathbf{e}}_{t+1}^k = T_{\mathbf{z}_t}(\widetilde{\mathbf{e}}_t^k)$, $\widehat{\mathbf{m}}_{t+1}^k = T_{\mathbf{z}_t}(\mathbf{m}_t^k)$
- Prediction of composition mask for time step $t+1$ and synthesizing the predicted frame based on the composition mask, warped (inpainted) entities as well as the inpainted background.

In our new stochastic prediction problem, we assume now that $\mathbf{z}_t^k$'s are random variables, and our objective is to approximate the posterior distribution $p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_{t-1})$. In this way, once we infer the amodal masks and inpainted entities from an input frame (in a deterministic manner), we can sample from the approximated distribution to predict the frame at the next time step. Repeating the sampling operation based on input frames or previously generated frames, we can perform single or multi-step prediction.

In the next section, we explain how we factorize the posterior distribution, which leads us to a first solution for approximating the posterior distribution based on VAE formulation.

4.3.2 VAE Formulation

As a matter of design, we choose to model motion as a time-varying distribution, which is shown to improve performance on video prediction tasks (Denton and Fergus, 2018). As discussed in the earlier sections, stochastic formulation requires a sampling operation in the pipeline. In our VAE formulation, we use the reparametrization trick (Kingma and Welling, 2013) and learn statistics of a predefined parametric distribution, to make our pipeline end-to-end differentiable. In order to explain how the proposed distribution of transformation parameters can be learned, let us first form the graphical model corresponding to our framework.



((4.2.1)) Prediction network ((4.2.2)) Inference network

Figure 4.2: Graphical model of the proposed VAE formulation for the stochastic prediction problem. Video frame $\mathbf{x}_t$ and inferred object masks $\mathbf{m}_t$ are used for modeling the posterior distribution for time-varying distribution of motion parameters, $\mathbf{z}_t$, *i.e.*, $p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_t)$. The posterior is then approximated by the distribution $q_{\phi_1}(\mathbf{z}_{t-1}|\mathbf{x}_{t+1:T}, \underline{\mathbf{m}}_{t:T}, \mathbf{z}_{t-1})$, which uses the backward recurrent auxiliary state $\mathbf{a}_t$.

If we consider object masks as deterministic internal representations of raw input, $\mathbf{x}_t$, our framework corresponds to the directed graphical model illustrated in Figure 4.2(4.2.1). Thus, given a sequence of frames $\mathbf{x}_{1:t}$, the initial object masks $\mathbf{m}_0^k$ and initial transformation parameters $\mathbf{z}_0^k$ for training, mutual learning of object masks and the distribution of transformation parameters to predict the subsequent frame at any time step t can be formalized as a maximum-likelihood problem, where we maximize the joint probability $p_\theta(\mathbf{x}_{t+1}, \mathbf{m}_{1:t}, \mathbf{z}_{1:t}|\mathbf{x}_{1:t}, \mathbf{m}_0, \mathbf{z}_0)$, where θ denotes all trainable parameters. If we follow a Markovian assumption for next frame prediction, mask inference, and estimation of motion dynamics, the likelihood factorizes over time as given in Equation (4.8).

$$p_\theta(\mathbf{x}_{t+1}, \mathbf{m}_{1:t}, \mathbf{z}_{1:t}|\mathbf{x}_{1:t}, \mathbf{m}_0, \mathbf{z}_0) = \prod_t \underbrace{p_{\theta_1}(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{m}_t, \mathbf{z}_t)}_{I} \underbrace{p_{\theta_2}(\mathbf{m}_t|\mathbf{x}_t, \mathbf{m}_{t-1})}_{II} \underbrace{p_{\theta_3}(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_{t-1})}_{III} \tag{4.8}$$

$$= \prod_t \delta(\mathbf{x}_{t+1} - \widehat{\mathbf{x}}_{t+1})\, \delta(\mathbf{m}_t - f_1(\mathbf{x}_t, \mathbf{m}_{t-1}))\, p_{\theta_3}(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_{t-1}) \tag{4.9}$$

The three processes in Equation(4.8) with Markovian assumption can be regarded as predicted frame synthesis (I), autoregressive object mask inference (II), and motion estimation (III). As both the frame synthesis and mask inference steps are designed to be deterministic, the corresponding distributions denoted as (I) and (II) in Equation (4.8) can be replaced by delta-dirac functions around the estimated values, as in Equation (4.9). The only remaining process of motion modeling, $p_{\theta_3}(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_{t-1})$ can be modeled as a Gaussian distribution, whose parameters, namely, mean vector μ_t and covariance matrix Σ_t as given in Equation (4.10) are to be learned via a nonlinear function $f_z(\cdot)$ as described in Equation (4.11).

$$\mathbf{z}_t^k \sim \mathcal{N}(\mu_t^k, \Sigma_t^k) \tag{4.10}$$

$$(\mu_t^k, \Sigma_t^k) = f_z(\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k, \mathbf{z}_{t-1}^k) \tag{4.11}$$

For training the proposed framework, we assume N independently observed sequences of length T, denoted as $\mathbf{x}_{1:T}^{(i)}$ for $i = 1, \ldots, N$. The task is to maximize the log-likelihood of the joint distribution in Equation (4.8) for all training sequences, which is described in Equation (4.12). The log-likelihood of the joint distribution can be factorized over individual sequences, i.e., $\mathscr{L}(\theta) = \sum_{i=1}^{N} \mathscr{L}^{(i)}(\theta)$ as in Equation (4.13), and further in time, as in Equation (4.14).

$$\begin{aligned}
\mathscr{L}(\theta) &= \log\Big(\{p_\theta(\mathbf{x}_{t+1}^{(i)}, \mathbf{m}_{1:t}^{(i)}, \mathbf{z}_{1:t}^{(i)} \,|\, \mathbf{x}_{1:t}^{(i)}, \mathbf{m}_0^{(i)}, \mathbf{z}_0^{(i)})\}_{t=0}^{T-1}\Big) \quad \text{for } i = 1, \ldots, N && (4.12)\\
&= \sum_{i=1}^{N} \log\Big(\{p_\theta(\mathbf{x}_{t+1}^{(i)}, \mathbf{m}_{1:t}^{(i)}, \mathbf{z}_{1:t}^{(i)} \,|\, \mathbf{x}_{1:t}^{(i)}, \mathbf{m}_0^{(i)}, \mathbf{z}_0^{(i)})\}_{t=1}^{T}\Big) && (4.13)\\
&= \sum_{i=1}^{N} \sum_{t=1}^{T-1} p_{\theta_1}(\mathbf{x}_{t+1}^{(i)} | \mathbf{x}_t^{(i)}, \mathbf{m}_t^{(i)}, \mathbf{z}_t^{(i)})\, p_{\theta_2}(\mathbf{m}_t^{(i)} | \mathbf{x}_t^{(i)}, \mathbf{m}_{t-1}^{(i)})\, p_{\theta_3}(\mathbf{z}_t^{(i)} | \mathbf{x}_t^{(i)}, \mathbf{m}_t^{(i)}, \mathbf{m}_{t-1}^{(i)}, \mathbf{z}_{t-1}^{(i)}) && (4.14)
\end{aligned}$$

However, direct maximization of the loss function $\mathscr{L}^{(i)}(\theta)$ is not possible due to the intractable nature of the term $p_{\theta_3}(\mathbf{z}_t^{(i)} | \mathbf{x}_t^{(i)}, \mathbf{m}_t^{(i)}, \mathbf{m}_{t-1}^{(i)}, \mathbf{z}_{t-1}^{(i)})$ in Equation (4.13). Instead, we suggest an inference network parametrized by ϕ to approximate the posterior p_θ by q_ϕ. Instead of $\mathscr{L}(\theta)$, we maximize an Evidence Lower Bound (ELBO), $\mathscr{F}(\theta, \phi)$, jointly parametrized by θ and ϕ, which can be similarly factorized over individual training sequences as $\mathscr{F}(\theta, \phi) = \sum_{i=1}^{N} \mathscr{F}^{(i)}(\theta, \phi) < \mathscr{L}^{(i)}(\theta)$. Each lower bound term $\mathscr{F}^{(i)}(\theta, \phi)$ can be expressed as:

$$\mathscr{F}^{(i)}(\theta, \phi) = \iint q_\phi(\mathbf{m}_{1:T}^{(i)}, \mathbf{z}_{1:T}^{(i)} | \mathbf{x}_{1:T}^{(i)}, \mathbf{m}_0^{(i)}, \mathbf{z}_0^{(i)})\, \frac{\{p_\theta(\mathbf{x}_{t+1}^{(i)}, \mathbf{m}_{1:t}^{(i)}, \mathbf{z}_{1:t}^{(i)} \,|\, \mathbf{x}_{1:t}^{(i)}, \mathbf{m}_0^{(i)}, \mathbf{z}_0^{(i)})\}_{t=0}^{T-1}}{q_\phi(\mathbf{m}_{1:T}^{(i)}, \mathbf{z}_{1:T}^{(i)} | \mathbf{x}_{1:T}^{(i)}, \mathbf{m}_0^{(i)}, \mathbf{z}_0^{(i)})}\, d\mathbf{z}_{1:T}\, d\mathbf{m}_{1:T} \tag{4.15}$$

To further proceed with the inference network, we assume access to the whole sequence $\mathbf{x}_{1:T}$ at any time step t for the prediction of transformation parameters. This allows us to obtain the object masks a priori during training; hence, motion prediction at any time step t corresponds to the estimation of $p_{\theta_3}(\mathbf{z}_t | \mathbf{x}_{1:T}, \mathbf{m}_{1:T}, \mathbf{z}_{t-1})$ rather than to $p_{\theta_3}(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_{t-1})$. As a virtue of the conditional independence properties of the graphical model in Figure 4.2(4.2.1) this term can be further factorized as $p_{\theta_3}(\{\mathbf{z}_t\}_{t=1}^{T} | \mathbf{x}_{1:T}, \mathbf{m}_{1:T}, \mathbf{z}_0) = \prod_t p_{\theta_3}(\mathbf{z}_t | \mathbf{x}_{t+1:T}, \mathbf{m}_{t:T}, \mathbf{z}_{t-1})$. It means that, at training time, the distribution of transformation parameters at time t depends neither on the object masks prior to $t-1$, nor on the frames in the past, but only on the ones at the current and future time steps. The dependence on future transformation parameters arises from the fact that we assume access to the whole sequence at training time to capture the motion dynamics in training sequences better by exposing the model to the whole sequence of frames. Similar conclusions and approaches can also be found in methods by Krishnan et al. (2015) and Fraccaro et al. (2016). However, at inference time, the model has access to only the parameters up to time t.

We design the approximate posterior q_ϕ in a similar fashion as the true posterior p_θ factorizes in terms of autoregressive object mask inference, motion estimation, and predicted frame synthesis. We use the same deterministic step for obtaining object masks, i.e., $q(\mathbf{m}_t|\mathbf{x}_t,\mathbf{m}_{t-1}) = p_{\theta_2}(\mathbf{m}_t|\mathbf{x}_t,\mathbf{m}_{t-1})$ as in Equation (4.16). We also keep a similar structure for the inference network for motion prediction and utilize an auxiliary state $\mathbf{a}_t$ to indicate the dependency of $\mathbf{z}_t$ on the future frames and masks, as given in Equation (4.17). The update for the auxiliary state requires backward processing in time (Fraccaro et al., 2016), we achieve that via a backward recurrent function $g_{\phi_a}(\cdot)$. The resultant graphical model corresponding to the inference network is depicted in Figure 4.2(4.2.2).

Hence, during training, we first realize a full forward pass in time for the inference of object masks $\{\mathbf{m}_t\}_{t=1}^{T}$, and proceed with a full backward pass to update the auxiliary states, $\{\mathbf{a}_t\}_{t=1}^{T-1}$. The final forward pass completes the cycle by estimating the parameters of the approximate posterior $q_{\phi_1}(\cdot)$ as described by Equations (4.16)-(4.17), which is also assumed to be a Gaussian with parameters $(\mu_{t,q}^k, \Sigma_{t,q}^k)$. We compose the predicted frames using the object masks from the first forward pass and transformation parameters sampled from $\mathcal{N}(\mu_{t,q}^k, \Sigma_{t,q}^k)$. Our approach can thus be considered as a structured way of approximating the posterior for transformation parameters in the proposed pipeline by traversing the whole sequence.

$$q_\phi(\mathbf{m}_{1:T},\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{m}_0,\mathbf{z}_0) = q_{\phi_1}(\mathbf{z}_{1:T}|\mathbf{x}_{1:T},\mathbf{m}_{1:T},\mathbf{z}_0)\ \{\underbrace{q(\mathbf{m}_t|\mathbf{x}_t,\mathbf{m}_{t-1})}_{\delta(\mathbf{m}_t-f_1(\mathbf{x}_t,\mathbf{m}_{t-1}))}\}_{t=1}^{T-1} \tag{4.16}$$

$$\prod_{t=0}^{T-1} q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T},\mathbf{m}_{t:T}) = \prod_{t=0}^{T-1} q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_t) \tag{4.17}$$

$$\text{where} \quad \mathbf{a}_t = f_{\phi_a}(\mathbf{a}_{t+1},[\mathbf{x}_{t+1},\mathbf{m}_t,\mathbf{m}_{t+1}])$$

Since both the generative model and the inference network factorizes over time, the problem of maximizing the term given in Equation (4.15) evolves to the maximization of Equation (4.18) for each sequence, where $\underline{\mathrm{m}}_t$ represents the output of the deterministic mask generation step, i.e., $\underline{\mathrm{m}}_t = f_1(\mathbf{x}_t,\mathbf{m}_{t-1})$ which arises from the integral of the δ-dirac function in Equation (4.15).

$$\begin{aligned}\mathscr{F}^{(i)}(\theta,\phi) = \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi}\Big[&\mathbb{E}_{q_{\phi_1}}\left[\log\left(p_{\theta_1}(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{m}_t,\mathbf{z}_t)\right)\right] \\ &- D_{\mathrm{KL}}\left(q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T},\underline{\mathbf{m}}_{t:T}) \,||\, p_{\theta_3}(\mathbf{z}_t|\mathbf{x}_t,\underline{\mathbf{m}}_t,\underline{\mathbf{m}}_{t-1},\mathbf{z}_{t-1})\right)\Big]\end{aligned} \tag{4.18}$$

The first term in Equation(4.18) can be regarded as the prediction quality of the framework where the transformation parameters are sampled from the approximate posterior q_ϕ, whereas the second term tries to bring the approximate posterior closer to posterior distribution p_θ. Note that the derivation of our loss term results in a formulation that highly resembles the VAE formulation for image generation. The main difference is that the first term for quantifying the image generation quality is replaced by the term for prediction quality.

4.3.3 WGAN Formulation

The second method follows all the assumptions stated by the previous VAE formulation. Namely, we keep the deterministic mask and entity inference functions from our decomposition model. Stochasticity is attributed to motion modeling, and the parameters of the geometric transformation, $\mathbf{z}_t$, are inferred with an additional noise input as in conditional GAN frameworks (Mirza and Osindero, 2014). More precisely, we compute the transformation parameters by $\mathbf{z}_t^k = f_z(\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k, [\mathbf{z}_{t-1}^k; \epsilon])$ with the noise vector of ϵ being sampled from standard normal distribution, *i.e.*, $\epsilon \sim \mathcal{N}(0, I)$. The operator $[\cdot;\cdot]$ here indicates the concatenation. Hence, the posterior distribution $p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_{t-1})$, this time, does not obey to any particular family of parametric distributions. After sampling the parameters of the geometric transformation from this time-varying distribution, the synthesis of the predicted frame is performed in the same deterministic manner as in Chapter 3.

The key difference between the two formulations under VAE and WGAN frameworks is related to the approximation of the posterior distribution. VAE-based first approach approximates the posterior with another distribution $q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{a}_t) = f_{\phi_a}(\mathbf{a}_{t+1}, [\mathbf{x}_{t+1}, \mathbf{m}_t, \mathbf{m}_{t+1}])$ and penalizes the divergence between these two distributions during training. In the second approach, we learn the parameters of the nonlinear function, $f_z(\cdot)$ which maps the noisy vector to any distribution, via adversarial training. The resultant posterior distribution can be also represented by $q(\mathbf{z}_t | \mathbf{z}_{t-1}, \mathbf{a}_t)$ with $\mathbf{a}_t = [\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k, \epsilon]$. For this purpose, we generate a sequence of predicted frames, $\{\hat{\mathbf{x}}_t\}_{t=2}^T$ with sampled motion parameters at each step, and consider this operation as sampling the sequence $\{\mathbf{x}_t\}_{t=2}^T$ from the distribution $q(\mathbf{x}_T)$. The sequences in the dataset, on the other hand, are considered to be sampled from the true distribution $p(\mathbf{x}_T)$. We then define a critic function that maps a given sequence $\{\mathbf{x}_t\}_{t=2}^T$ to a scalar value, such that $c(\cdot) : \mathbb{R}^{H \times W \times 3 \times (T-1)} \rightarrow \mathbb{R}$. In order to learn the parameters of the prediction framework, we use the iterative updates proposed by WGANs-GP (Gulrajani et al., 2017), which are expressed in Equations (4.19)-(4.20). The sampling operation $\{\mathbf{x}_t\}_{t=2}^T \sim pq(\mathbf{x}_T)$ in Equation (4.19) corresponds to a random linear interpolation of $p(\mathbf{x}_T)$ and $q(\mathbf{x}_T)$. It is achieved by sampling a value ρ from a uniform distribution, *i.e.*, $\rho \sim U[0,1]$ and computing $\left(\rho \{\mathbf{x}_t\}_{t=2}^T + (1-\rho) \{\hat{\mathbf{x}}_t\}_{t=2}^T\right)$. The function $g(\cdot)$ denotes the collective operator of functions $f_1(\cdot), f_m(\cdot), g_e(\cdot), g_b(\cdot), f_z(\cdot)$ and $T_{\mathbf{z}_t}$ that are briefly described in Section 4.3.1 and detailed in Chapter 3. The expectation operators in Equations (4.19)-(4.20) are computed greedily over each batch during training.

$$
\begin{aligned}
c = \arg\max_c \mathbb{E}_{\{\mathbf{x}_t\}_{t=2}^T \sim p(\mathbf{x}_T)} \left[c(\{\mathbf{x}_t\}_{t=2}^T) \right] - \mathbb{E}_{\{\mathbf{x}_t\}_{t=2}^T \sim q(\mathbf{x}_T)} \left[c(\{\mathbf{x}_t\}_{t=2}^T) \right] \\
- \lambda \mathbb{E}_{\{\mathbf{x}_t\}_{t=2}^T \sim pq(\mathbf{x}_T)} \left[(\| \Delta_{\{\mathbf{x}_t\}_{t=2}^T} c(\{\mathbf{x}_t\}_{t=2}^T) \| - 1)^2 \right] \qquad (4.19)
\end{aligned}
$$

$$
g = \arg\max_g \mathbb{E}_{\{\mathbf{x}_t\}_{t=2}^T \sim q(\mathbf{x}_T)} \left[-c\left(\{\mathbf{x}_t\}_{t=2}^T \right) \right] \qquad (4.20)
$$

4.4 Implementation

In this section, we briefly describe the implementation details describing our model for object-centric stochastic video prediction.

4.4.1 Model Implementation

To begin with, we use the implementation of Model 2, which is developed in Chapter 4.3.1, as the backbone of our stochastic prediction pipeline. For both methods detailed in this chapter, all function implementations of Model 2 remain the same, with the important exception of $f_z(\cdot)$. In the deterministic framework, the function $f_z(\cdot)$ is designed to predict the geometric transformation parameters based on the input frame, $\mathbf{x}_t$, amodal masks of the last two time steps, $\mathbf{m}_t^k, \mathbf{m}_{t-1}^k$, as well as the predicted transformation parameters from the previous time step, $\mathbf{z}_{t-1}^k$. The sequential information is, in fact, handled by the help of a recurrent state $\mathbf{s}_{\mathbf{z},t-1}^k$. More precisely, the function $f_z(\cdot)$ performs the operations described in Equations (4.21)-(4.22).

$$\mathbf{s}_{\mathbf{z},t}^k = f_{z,1}(\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k, \mathbf{z}_{t-1}^k, \mathbf{s}_{\mathbf{z},t-1}^k) \tag{4.21}$$

$$\mathbf{z}_t^k = f_{z,2}(\mathbf{s}_{\mathbf{z},t-1}^k) \tag{4.22}$$

VAE-based Implementation

For the stochastic prediction network with our VAE formulation, we keep the structure of the function $f_z(\cdot)$ implementation the same. However, rather than predicting the parameters $\mathbf{z}_{t-1}^k$, it now predicts the mean vector μ_t^k and the covariance matrix Σ_t^k describing the true posterior distribution $p_{\theta_3}(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_{t-1})$. The modified function $f_z^{\text{VAE}}(\cdot)$, thus performs the operations described in Equations (4.23)-(4.25). Consequently, the number of hidden units in the penultimate layer of the MLP that parameterizes the function $f_{z,2}$ is doubled in order to accommodate the diagonal entries of Σ_t^k and is now denoted by $f_{z,2}^s(\cdot)$.

$$\mathbf{s}_{\mathbf{z},t}^k = f_{z,1}^{\text{VAE}}(\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k, \mathbf{z}_{t-1}^k, \mathbf{s}_{\mathbf{z},t-1}^k) \tag{4.23}$$

$$(\mu_t^k, \Sigma_t^k) = f_{z,2}^{\text{VAE}}(\mathbf{s}_{\mathbf{z},t-1}^k) \tag{4.24}$$

$$\mathbf{z}_t^k = \mu_t^k + \Sigma_t^{k\frac{1}{2}} * \epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I) \tag{4.25}$$

With the modification of the function $f_z^{\text{VAE}}(\cdot)$, the implementation of the prediction network that is described in Figure 4.2(4.2.1) is complete. For the implementation of the inference network, which is depicted in Figure 4.2(4.2.2), we use another Gated Recurrent Unit (GRU), denoted by $f_{\phi_a,2}$ which performs the calculation $\mathbf{a}_t = f_{\phi_a}(\mathbf{a}_{t+1}, [\mathbf{x}_{t+1}, \mathbf{m}_t, \mathbf{m}_{t+1}])$ presented in Equation (4.17). The function f_{ϕ_a} can be further described as a composition of three functions, $f_{\phi_a,1}(\cdot), f_{\phi_a,2}(\cdot)$ and $f_{\phi_a,3}(\cdot)$. In fact, first, the function $f_{\phi_a,1}(\cdot)$ processes the visual input of $\{\mathbf{x}_{t+1}, \mathbf{m}_t^k, \mathbf{m}_{t+1}^k\}$. Its parametrization follows the same structure of the CNN that processes the visual input of $\{\mathbf{x}_{t+1}, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k\}$ for the prediction network, which is denoted by $f_{z,1}(\cdot)$. The output of $f_{\phi_a,1}(\cdot)$ is then fed to an MLP before processing the auxiliary state with the new GRU unit which operates backward in time, which corresponds to the function , $f_{\phi_a,2}(\cdot)$. A final MLP maps the GRU state $\mathbf{a}_t$ and the transformation parameters predicted at time step $t-1$, *i.e.*, $\mathbf{z}_{t-1}^k$, to the mean vector $\mu_{q,t}^k$ and the covariance matrix $\Sigma_{q,t}^k$ which describe the approximate posterior distribution, such that $(\mu_{q,t}^k, \Sigma_{q,t}^k) = f_{\phi_a,3}(\mathbf{a}_t^k, \mathbf{z}_{t-1})$.

WGAN-GP-based Implementation

The implementation of the second approach is relatively more trivial compared to the first one. This time, we keep the same structure for the recurrent state $\mathbf{s}_{\mathbf{z},t-1}^k$ as in the deterministic case, which is also described in Equation (4.26). The stochasticity is, however, introduced by injecting Gaussian noise for the computation of the transformation parameters from the recurrent state as given in Equation (4.27).

$$\mathbf{s}_{\mathbf{z},t}^k = f_{z,1}^{\text{WGAN}}(\mathbf{x}_t, \mathbf{m}_t^k, \mathbf{m}_{t-1}^k, \mathbf{z}_{t-1}^k, \mathbf{s}_{\mathbf{z},t-1}^k) \tag{4.26}$$

$$\mathbf{z}_t^k = f_{z,2}^{\text{WGAN}}\left([\mathbf{s}_{\mathbf{z},t-1}^k; \epsilon]\right) \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, I) \tag{4.27}$$

On the other hand, the critic function, $c(\cdot)$ has no resemblance to any implemented function, either in the deterministic framework or in the VAE-based stochastic framework. For the implementation of the function $c(\cdot) : \mathbb{R}^{H\times W\times 3\times(T-1)} \to \mathbb{R}$, we opt for a 3D convolutional neural network with three layers. Such an architecture enables us to retain information regarding both the spatial and the temporal changes in a given sequence. We use a kernel size of 5 for both spatial and temporal processing of the input video frames. A spatial stride size of 2 at each convolutional layer downsamples the input video sequence spatially by a factor of 8. The resultant features are then reshaped and fed into a 3-layer MLP, which has a single hidden unit in the last layer.

4.4.2 Multi-stage Training

Direct optimization of the loss function $\mathcal{F}^{(i)}(\theta, \phi)$ in Equation (4.15) or the min-max game defined by Equations (4.19)-(4.20) is unnecessarily challenging. It is due to the fact that the decoupling method implemented by the deterministic model heavily relies on accurate descriptions of entities, even under occlusions. The representation ability of the deterministic model thus comes from the cyclic consistency loss defined in Chapter 3; however, it constrains the stochastic problem. To alleviate this conflict, we propose a multi-stage training scheme that encapsulates both VAE- and WGAN-GP-based approaches.

We first train the deterministic Model 2 in a deterministic fashion with the loss term defined in Equation (3.61). Under VAE settings, it corresponds to using $\mathbf{z}_t^k = \mu_t^k$ for the prediction network. It also means that the second term in Equation (4.18) is discarded during the first phase of the multi-stage training. Similarly, under WGAN settings, the min-max game defined by Equations (4.19)-(4.20) is discarded, hence the parameters of the critic function are not updated. For the computation of the transformation parameters, the recurrent state is padded with zeros instead of Gaussian noise. This stage shapes the functions for amodal mask inference and inpainting; however, it does not update the parameters of the inference network in the VAE-based pipeline, nor does it change the parameters of the critic function in the WGAN-GP-based pipeline.

After the first stage, we freeze the parameters of all the functions in the pipeline of Model 2, with the only exception of $f_z(\cdot)$. We activate stochastic prediction; the transformation parameters are computed according to Equations (4.23)-(4.25) or Equations (4.26)-(4.27) / Hence, optimize the VAE-loss function in Equation (4.18) or play the min-max game in Equations (4.19)-(4.20). This second stage aims to bring stochasticity to the framework and update the parameters of designated functions.

An optional third stage aims to capture the distribution of the random variable $\mathbf{z}_t^k$ better. The second stage models the time-varying posterior distribution based on a single-step prediction, which provides limited information about the true distribution. Hence, for the optional third stage, we predict a sequence in an autoregressive manner to model the uncertainties in the dataset better. We do it by feeding the predicted frame at time t as the input frame at time $t+1$, *i.e.*, $\mathbf{x}_{t+1} \leftarrow \widehat{\mathbf{x}}_{t+1}$.

4.5 Experimental Results

The evaluation of image prediction methods is not trivial. Frequently used reconstruction quality metrics do not apply to the stochastic prediction methods as their objective is to generate sequences depicting as many different plausible outcomes as the method can model. Hence, in this section, we present some illustrative visual outcomes.

4.5.1 Dataset

Inspired by the evaluation of G-SWM (Lin et al., 2020b), which is an object abstraction method that explicitly models the location of the abstracted entities, we create a simple dataset for evaluating the representation capacity of proposed stochastic frameworks. The dataset contains video sequences of a single simulated object, *i.e.*, a colored sprite selected from a shape set of `circle, square, triangle` with 5^3-1 different color choices. The object is initially placed at the middle top of the input frame. For every sequence in the dataset, the object moves to bottom of the frame with constant speed. After time $t=5$, the object follows different patterns based on its shape. If the object is an instance of the `circle` shape, it continues its motion towards the bottom-middle of the frame. Otherwise, with an equal chance, it continues its motion one of the bottom corners. We generated 20,000 training sequences, 2000 validation sequences, and 2000 test sequences. Some example sequences from the test set can be seen in

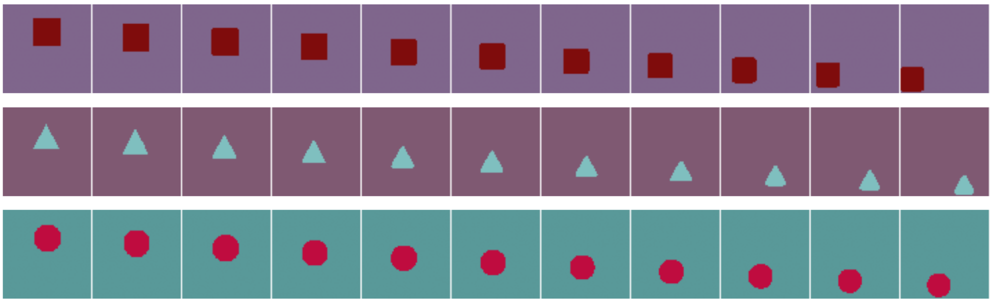


Figure 4.3: Sample sequences from the test set: in each sequence, an object starts moving from the middle-top of the input frame toward to bottom of the frame with constant speed. After time $t = 5$, the object follows different patterns based on its shape: `circles` head toward the bottom-middle whereas `squares` or `triangles` move towards one of the bottom corners.

Figure 4.3.

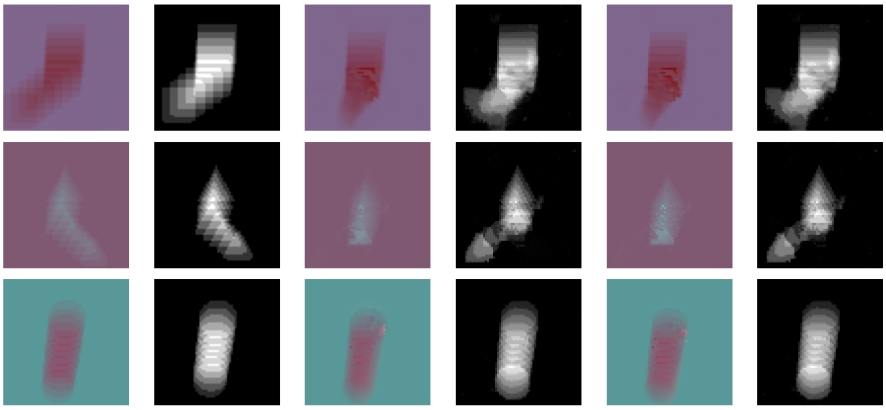
4.5.2 Results

In this section, we present the performance of the two proposed models under different conditions.

We start with the model based on our sequential VAE formulation. We train the model for an initial 50K training steps in deterministic settings for single-frame prediction. Later, we switch to the stochastic settings and train the model for another 50K training steps only to update the parameters of the motion prediction function as explained in Section 4.4.2. Finally, we perform long-term prediction instead of single-step prediction for training and update the parameters of the full model for the last 40K training steps.

We illustrate some representative test cases for each object class quasi-randomly chosen from the test set. In particular, we randomly choose one sequence for each object class, and sample 100 sequences for different numbers of context frames, namely, $n_{\text{context}} = 1$ and $n_{\text{context}} = 4$. The context frames refer to the time-steps where the input is the ground-truth frame. After n_{context} we switch to prediction in an autoregressive manner by using the previously predicted frame as the input to the system, *i.e.*, $\mathbf{x}_{t+1} \leftarrow \hat{\mathbf{x}}_{t+1}$. In Figure 4.4, we visualize the average frame over the whole sequence and a total number of 100 sampled sequences. The ground-truth average sequence illustrates the motion pattern for the given test sample. The average of the sampled sequences would ideally indicate two main patterns of the sampled motion, towards either the bottom corner of the frame for `square` and `triangle` and single pattern towards the bottom-middle of the frame for `circle`. We can first observe that the model can capture shape-based motion patterns reasonably well. However, the capability of the proposed model for covering the shape-based motion distribution seems to be limited, fixating mostly on one direction for `square` and `triangle` object types.

In order to investigate whether the average images are a good way of representing the sampling



(a) Ground Truth (b) $n_{\text{context}} = 1$ (c) $n_{\text{context}} = 4$

Figure 4.4: The average frame and entity mask image for the VAE-based model. (a) The first two columns illustrate the test sequence and ground-truth object masks averaged over time. (b) The middle two columns represent the average images for 100 different samples for $n_{\text{context}} = 1$, while (c) the last two column represent the average images for 100 different samples for $n_{\text{context}} = 4$. We can observe that the model can capture the shape-based motion patterns while failing to represent different modes of motion for a given shape, *i.e.*, motion towards either bottom-corner for `square` and `triangle` object types.

power of the proposed method, we illustrate two sampled sequences for each test case in Figure 4.5. The sequences are chosen in such a way that the PSNR with the averaged ground truth sequence is either the highest or the lowest. The visualization of sample sequences enforces our previous observation that the VAE-based model is capable of understanding shape-specific motion patterns when we inspect the object masks. However, we can also observe that there is a discrepancy between the sampled sequences and corresponding object masks: the model seems to have a weaker ability to retain the information regarding the object's appearance. We can say that introduction of stochasticity is degraded the quality of our abstracted representations, to some extent. Moreover, we can observe that the model can sometimes sample different plausible transformation parameters for the same input: when $n_{\text{content}} = 1$, the sampled sequences for the `triangle` object is a promising example, particularly when we focus on the object masks.

To understand the effect of different training stages on modeling the motion parameter distributions, we also visualize two sequences sampled for the same test cases in Figure 4.6, where we use the model before the last stage of training. In other words, the sequences in Figure 4.6 are sampled when the model is trained for single-frame prediction using stochastic transformation parameters; yet, the sampling is performed in an autoregressive manner as for the previous results for a fairer comparison. One main observation we can make is the poor object abstraction performance, because the background seems to contain parts of object appearance, particularly after time n_{content}. The last training stage seems to improve the problem with object-background decoupling, yet, it does in a way that the model loses the ability to retain the object appearance information. Moreover, the model seems to rec-

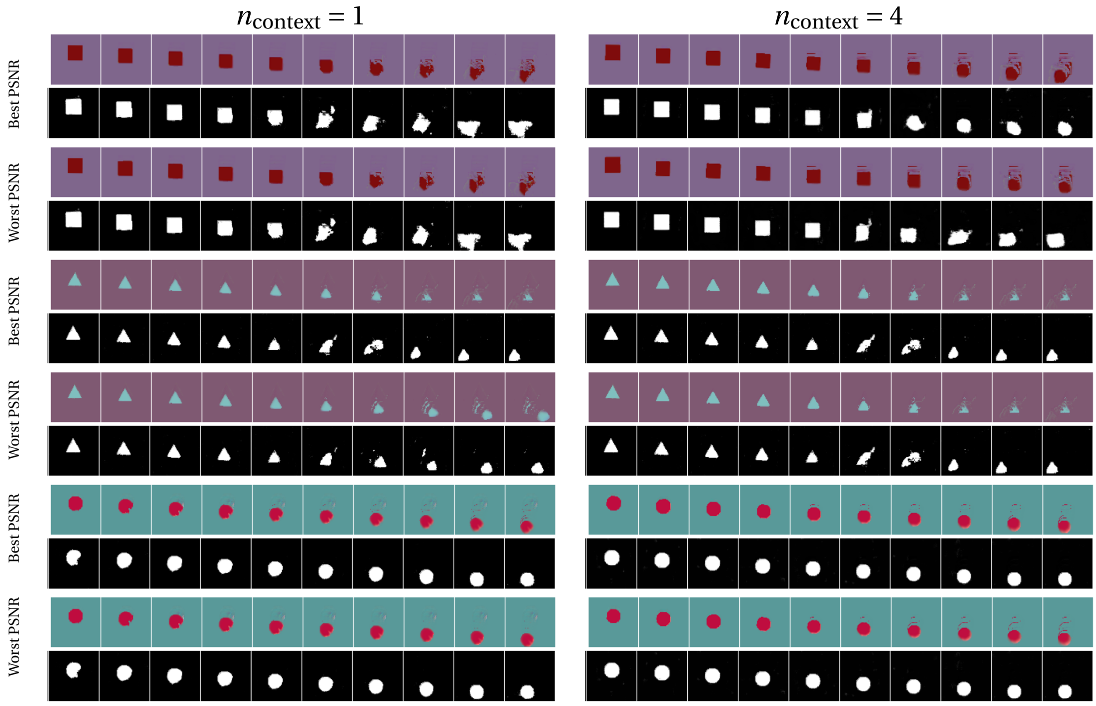


Figure 4.5: Two sampled sequences for each the test cases in Figure 4.5 that corresponds to the best and the worst PSNR of the average frames with the ground-truth. The first column represents the sampled sequences for $n_{\text{context}} = 1$ while the second column represents the sampled sequences for $n_{\text{context}} = 4$.

ognize shape-specific motion patterns better after the last stage of training, which can be better observed on the object masks of the `triangle` sequence in both cases. Overall, this experiment articulates the difficulty of stochastic modeling of video prediction problem while successfully abstracting entities in each frame. Despite all our efforts to isolate the learning of these two phenomena, the results show that simultaneously improving the performance for both remains a challenging problem.

We perform the same experiments for our second, WGAN-based approach. The average frame and the object mask image for 100 sampled sequences are depicted in Figure 4.7. After an initial inspection, we can say that the second approach also seems to learn shape-based motion patterns. Moreover, the average frame seems to be more consistent with the average mask image in contrast to the previous approach. When we inspect the sampled sequences in Figure 4.8, we can observe the same effect where the frames and the object masks are more aligned. The quality of the output frames is also higher compared to their counterparts sampled from the VAE-based model. This can be attributed to the adversarial training, which is known for more visually plausible outputs. Yet, we cannot observe different motion patterns for a given scene/frame/entity, which can be related to the well-known mode collapse problem in Generative Adversarial Networks (Salimans et al., 2016). Similar observations can be made for the sequences sampled with the model before the last training stage, which can be observed

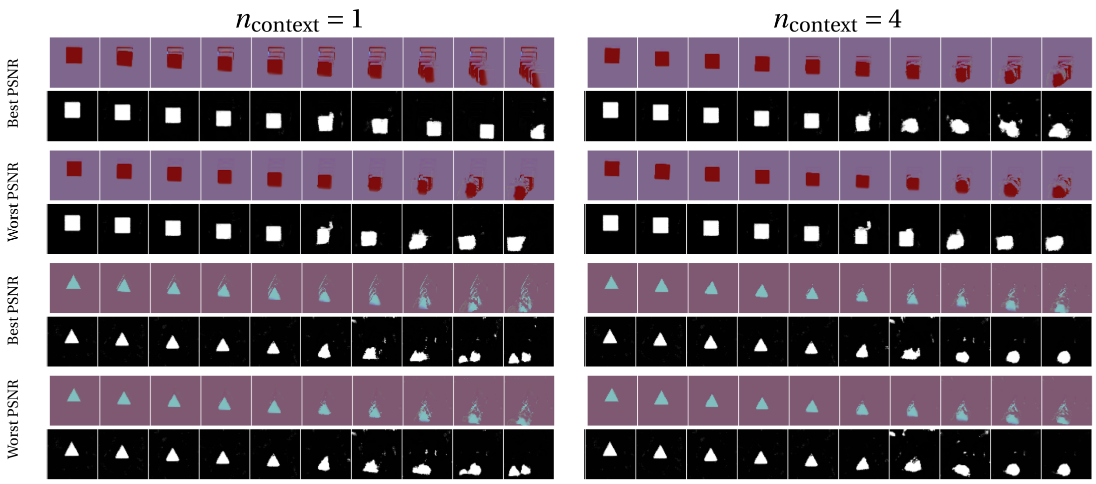


Figure 4.6: Partial replication of the results in Figure 4.5 for the model state before the last stage of training.

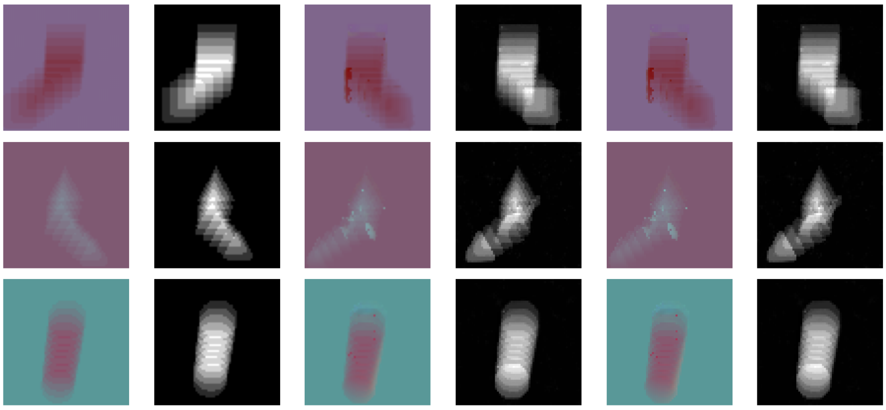
in Figure 4.9.

4.6 Conclusion

In this chapter, we extended our unsupervised entity abstraction method, which already grants deterministic single-step prediction as a result of its construction, to stochastic object-centric video prediction. For this purpose, we treat the geometric transformation parameters as random variables and principle two different methods to approximate its time-varying posterior distribution.

The first method sketches a graphical model based on the dependencies between model variables. The training objective is then derived from the maximum likelihood estimation, which leads to a differentiable method where the posterior distribution is approximated with the help of a backward-recurrent auxiliary state. The algorithm traverses the sequence of frames three times during the training time. This approach resembles a sequential VAE, which is often claimed to suffer less from mode collapse compared to adversarial methods, which we use for our second method. Being consistent with such a claim, the experiments showed that the first approach indeed models the stochasticity better compared to our second approach, yet, it suffers from poor prediction quality and worsened entity abstraction.

On the other hand, the second approach for approximating the posterior distribution is less computationally heavy. This method follows adversarial training based on the sequences generated by predicting the next frame at each time instance, and the "real" video sequences sampled from the training set. By using a spatio-temporal critic function, the parameters of the function that characterizes the posterior distribution are learned via min-max optimization. The results indicate a better prediction quality for the second approach, where the inferred entity masks stay consistent with the predicted frames. In addition, both approaches managed



(a) Ground Truth (b) $n_{\text{context}} = 1$ (c) $n_{\text{context}} = 4$

Figure 4.7: The average frame and entity mask image for the WGAN-based model. (a) The first two columns illustrate the test sequence and ground-truth object masks averaged over time. (b) The middle two columns represent the average images for 100 different samples for $n_{\text{context}} = 1$, while (c) the last two column represent the average images for 100 different samples for $n_{\text{context}} = 4$. We can observe that the model can capture the shape-based motion patterns while failing to represent different modes of motion for a given shape, *i.e.*, motion towards either bottom-corner for `square` and `triangle` object types.

to learn shape-specific motion patterns, and generate plausible sequences given a single context frame. Hence, our results demonstrate that the object-centric approaches are a promising step tackling the highly challenging problem of stochastic video prediction.

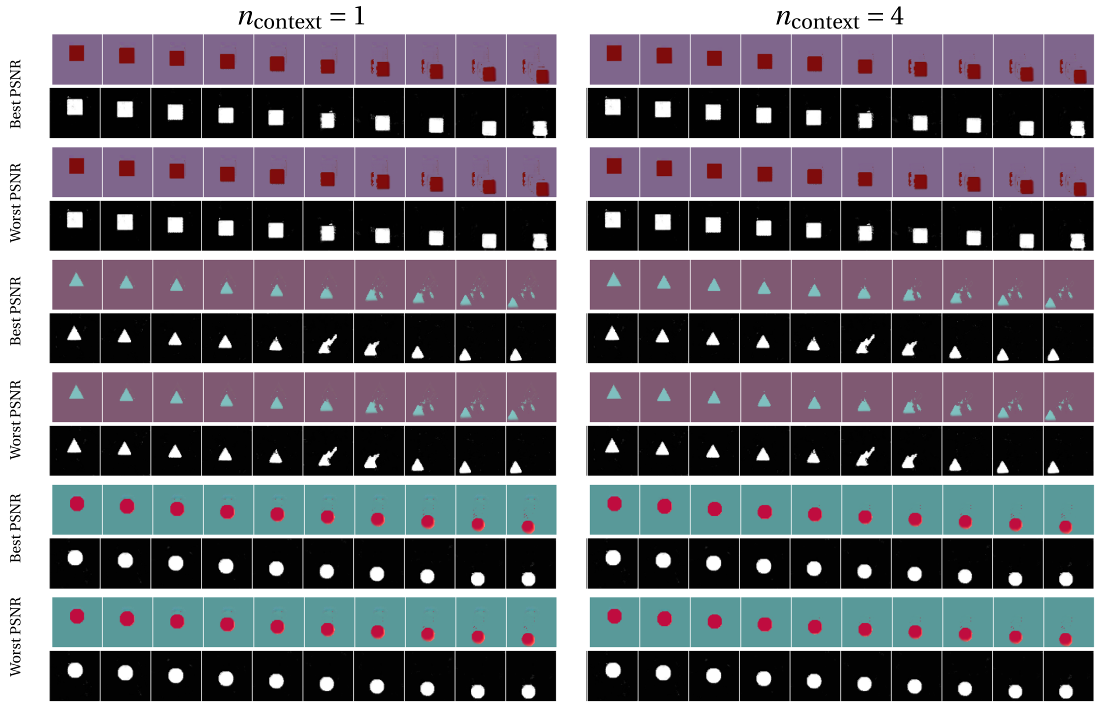


Figure 4.8: Two sampled sequences for each the test cases in Figure 4.8 that corresponds to the best and the worst PSNR of the average frames with the ground-truth. The first column represents the sampled sequences for $n_{\text{context}} = 1$ while the second column represents the sampled sequences for $n_{\text{context}} = 4$.

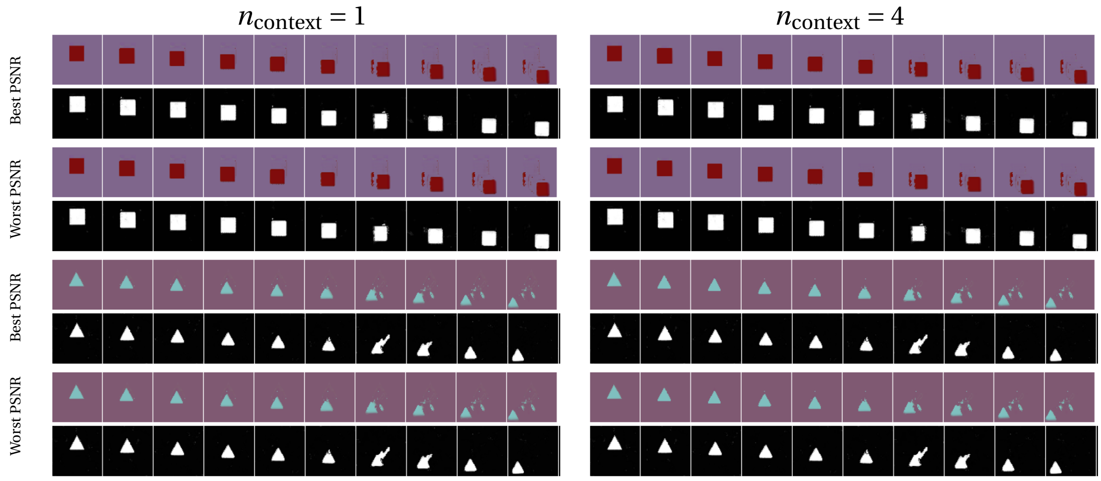


Figure 4.9: Partial replication of the results in Figure 4.8 for the model state before the last stage of training.

5 Discrete Motion in Multi-View Settings

5.1 Introduction

Object abstraction in 2D with motion clues inherits many challenges that emerge from the ambiguity of the inferred scene structure, such as occlusions or in-plane rotations. This is largely due to the information loss caused by projecting the 3D world in two dimensions. On the other hand, research in three dimensions involves different complexities. First and foremost, the data representation in 3D varies broadly depending on the application. A scene can be represented by points in 3D space (point clouds), surfaces (meshes), or solid cubical units (voxels). Regardless of the chosen data representation type, the capacity needed for describing the 3D scene increases with an additional dimension compared to its 2D settings, and it comes at the cost of increased computational and memory requirements. Likewise, annotating data in 3D becomes more cumbersome and labor-intensive. Hence, even with the great efforts of the research community in contributing different types of 3D labeled datasets, which consist of 3D bounding boxes (Ahmadyan et al., 2021), 3D Computer Aided Design (CAD) models (Xiang et al., 2014), category-specific 3D point cloud collections (Wu et al., 2015) or per-point semantic labels for point clouds (Armeni et al., 2017; Behley et al., 2019), the availability of annotated 3D datasets stay much smaller than that of labeled 2D image collections. This limited data availability emphasizes the importance of self-supervised, or more generally, unsupervised methods for data in three dimensions, which comes with numerous associated challenges.

The constraints that one could impose for learning 3D representations without manual annotations depend heavily on the type of data representation, which itself is tightly coupled to 3D data acquisition or generation. One way to obtain a dataset that is described in three dimensions is to model it using computer-aided software, such as CAD models, which requires skilled experts for the generation of complex objects or scenes. Another option is to capture data in specific environments, for example, using camera arrays with controlled illumination or with active depth sensors. In contrast, one can reconstruct the 3D structure solely based on 2D images of the scene at the cost of inferior accuracy, but with the benefit of alleviating the need for specialized equipment or manual labor of creating models with targeted software. In more detail, image-based 3D reconstruction algorithms permit the inference of the most likely 3D shape or scene structure that results in a given collection of images under the assumption of known viewpoints, material, or lighting (Furukawa and Hernández, 2015). The problem is generally ill-posed if there is no assumption of these physical factors of variation as different settings and combinations may result in the same set of images. The most common approach to circumvent this ambiguity is to find stereo-correspondences between images to extract geometry, which is commonly referred to as Multi-View Stereo (MVS) when the camera parameters are available, or jointly inferred while matching visual cues from images (Furukawa and Hernández, 2015). Camera parameters here refer to the orientation and world coordinates of the camera, which are also known as camera extrinsic parameters, along with the focal length and pixel sensor size describing intrinsic parameters.

In these image-based 3D reconstruction algorithms, the global structure has traditionally been inferred by matching visual keypoints (Lowe, 1999; Schonberger and Frahm, 2016) using principles from multi-view geometry (Hartley and Zisserman, 2003). Lately, neural networks have become a popular choice for learning 3D structure in many different forms, such as occupancy maps (Mescheder et al., 2019), signed distance functions (Park et al., 2019a), scene representations (Sitzmann et al., 2019) or radiance fields (Mildenhall et al., 2020). This rapidly growing field has demonstrated how the geometry and appearance of a scene can be successfully inferred from a single image, or more frequently, from a set of registered images. In particular, Neural Radiance Fields (NeRFs) (Mildenhall et al., 2020) has become the dominant scene representation model as a virtue of its promising high-fidelity outcomes. Although the initial NeRF model suffered from a computational bottleneck, large data requirements, and the inability to generalize to new scenes, subsequent work has rapidly addressed many of these concerns with regularization or consistency constraints for rendering with few available views (Jain et al., 2021; Kim et al., 2021), special data structures or input encodings for shorter training and inference times (Liu et al., 2020; Yu et al., 2021a; Müller et al., 2022) as well as multi-view principled approaches for generalization to novel scenes (Trevithick and Yang, 2021; Wang et al., 2021b; Yu et al., 2021b; Chen et al., 2021; Johari et al., 2022).

However, the compositionality of these models in terms of entity abstractions has only been explored in a very limited way despite the several benefits that compositional 3D models can offer, such as robot perception and planning, or scene editing. In this chapter, we propose a principled method that can both achieve free-view rendering and disambiguate entities in a fully unsupervised manner. As discussed in earlier chapters, we consider the object abstraction problem in a static scene very ill-posed, hence, in this chapter, we assume observations of a scene from different viewpoints that are acquired at two different time instances, between which some of the objects are relocated. Such an approach for disambiguation bears a resemblance to the concept of *moving* objects in previous chapters, with the strong difference of motion continuity assumption. On the other hand, this approach mandates some generalization ability for the rendering module, which prevents using NeRF models in their simplest form.

We hence propose using a discretized volumetric representation for bounded scenes for neural rendering so that the volume can later be grouped into smaller groups to represent entities. For rendering, such a volumetric representation must consist of occupancy and color information at the minimum; however, relying only on occupancy and color would require a high volumetric spatial resolution for high-fidelity rendering outcomes, resulting in high memory requirements. Therefore, we design the volumetric representation to embed more information by accommodating a feature vector in each volume unit. The observations from two time instances are represented by individual feature volumes so that we can use the volumetric features for both rendering and finding correspondences between the two volumes. In order to use a single rendering module for both feature volumes, we ensure that the volumetric features depend on observed images. Thus, we construct the volumetric representation by lifting 2D features from a subset of input images for each time instance. Overall, this foundational feature volume construction is jointly learned with the rendering module in a scene-agnostic manner so that a scene captured at different times from multiple viewpoints can be possibly represented by the proposed method without any further optimization. Our method shares the main motivation of a principled multi-view geometry approach for aggregating image features; however, we propose to model a bounded scene with a low-resolution feature volume where the feature for each voxel unit is aggregated from image features at all available viewpoints, which is constructed once and for all. Provided that the learned feature volume can represent the underlying geometry sufficiently well, we propose to represent objects as clusters of volume units based on consistent feature similarity across two time instances. In order to cluster voxel units, we first find correspondences between two feature volumes using algorithms from optimal transport (Peyré and Cuturi, 2018). We work with spatially localized groups of voxels to test feature consistency between two volumes based on correspondences and estimate a rigid body transformation for each group. We then finalize the grouping by including all voxel units that can be represented by the same rigid body transformation and establish feature similarity across volumes after being registered by the transformation. By recursive application of grouping, feature consistency check, 3D registration across volumes by an estimated rigid body transformation, and re-grouping, we divide each feature volume into clusters that resemble re-located entities. Once the clustering is finalized, it is possible to render each arrangement of the scene as well as its composing entities individually. To the best of our knowledge, it is the first unsupervised algorithm to abstract entities directly in 3D without the assumption of continuous motion.

5.2 Related Work

As the name indicates, 3D computer vision works on problems related to understanding the world in three dimensions. It has abundant applications in robotics, 3D scene understanding, autonomous driving and virtual/augmented/mixed reality. With recent advances in deep learning, increase in compute power and large-scale 3D geometry datasets, we witness impressive progress in 3D computer vision. In this section, we briefly visit how data-driven approaches and representations have improved the state-of-the-art in different problems in 3D when combined with principles adopted from 3D geometry. We then overview rapidly developing field of neural rendering, and introduce few recent methods that target compositionality in neural rendering algorithms.

5.2.1 3D Representations

Stereo vision works with principles similar to the binocular vision system of humans. We perceive the world in three dimensions thanks to the slight difference between what each of our eye sees while focusing on the same point in space and maintaining a significant overlap between their field of views. In a similar fashion to our brain, which interprets each view to create the perception of depth, stereo vision algorithms processes 2D images of a scene captured from different view points in order to find the 2D points corresponding to the same 3D point in space and use the disparity to compute a depthmap.

In particular, multi-view stereo (MVS) tackles the long-standing problem of 3D representations of a scene given calibrated overlapping images (Hartley and Zisserman, 2003; Furukawa and Hernández, 2015). Early MVS approaches aimed for volumetric optimization based on photo-consistency constraints (Faugeras and Keriven, 2002; Vogiatzis et al., 2007) or sweeping planes (Gallup et al., 2007), with optional depth map construction (Liu et al., 2009; Furukawa et al., 2009). Voxel (De Bonet and Viola, 1999) or mesh (Kazhdan and Hoppe, 2013) based representations have also been frequently used for novel-view rendering. In principle, these methods aim to reverse the process of image acquisition and create a three dimensional model with images that are captured from different viewpoints. In short, a typical pipeline consists of camera calibration, depth determination and registration. Camera calibration is a step needed to determine the extrinsic and intrinsic properties of the camera. Depth determination is often achieved by finding matches between two overlapping images and triangulating those points in 3D. Registration in this context refers to merging inferred depth maps from available viewpoints into a single 3D model.

In the last decade, learning based methods have gradually improved the performance on MVS problems and related sub-tasks. For example, depth inference (Huang et al., 2018; Im et al., 2019) , surface reconstruction from principled 3D volume with sweeping planes (Yao et al., 2018), multiple cost volumes (Gu et al., 2020), and point-based approaches (Chen et al., 2019) have led to benefits with respect to older MVS methods. Neural networks have also been employed for 3D reconstruction in the last few years as a purely data-driven approach (Tewari et al., 2022). For example, the line of work known as implicit neural representations learns a continuous function that maps 3D world coordinates to a preferred attribute, such as an indicator of occupancy, color and/or density. Using a continuous function to express an arbitrary topology permits a representation with higher capacity and alleviates bottlenecks of discrete canonical representations, such as point clouds, voxels or meshes. However, learning such high-dimensional functions require long training times and many data samples.

In order to aid the learning process, multi-view consistency principles have recently been integrated into neural rendering methods (Wang et al., 2021b; Yu et al., 2021b; Chen et al., 2021; Trevithick and Yang, 2021; Sun et al., 2021b; Ji et al., 2017; Saito et al., 2019; Murez et al., 2020). These methods here greatly improved the state of the art and enabled more efficient training schemes.

5.2.2 Neural Rendering

Despite the recent advances in different 3D representations, rendering a scene from a novel viewpoint given observations from a set of other viewpoints stays as a challenging task that requires an adequate understanding the geometry of the scene. This long-standing problem at the intersection of computer vision and computer graphics is approached as an interpolation problem by early works (Shum and He, 1999; Heigl et al., 1999). The quality of rendered images has substantially improved over the years with the methods from classical computer graphics, which mostly stem from the perspective of physics and heavily rely on the correctness of the geometry, light, camera and surface modelling. On the other hand, data-driven approaches have more statistical foundation, which can greatly help building explicit representation of scenes based on sufficient number of observations. Neural rendering (Tewari et al., 2020) aims to combine the advantages of both approaches and provide image-based rendering based on representations learned with neural networks.

Recently, coordinate based methods, very often referred as implicit functions, have become a popular representation for 3D reconstruction (Park et al., 2019a; Mescheder et al., 2019; Sitzmann et al., 2019; Mildenhall et al., 2020). Among those, Neural Radiance Fields (NeRFs) (Mildenhall et al., 2020) notably attracted a lot of attention due to their representation capacity. In short, a NeRF representation learns a continuous function parameterized by Multi-Layer Perceptrons(MLPs) to map 3D world coordinates to density and R,G,B colors for rendering. By construction, a NeRF model is capable of representing a single, static scene observed from multiple viewpoints. A differentiable ray marching algorithm enables end-to-end training for this high dimensional function using calibrated image collections of a given scene. However, it requires sampling many points along a ray from the camera for recovering the color value of a single image pixel for the main training objective of photo consistency loss. Such a computational complexity leads to long training and inference times, yet, NeRFs achieve remarkable performance for free view rendering.

Some follow-up work improved the original formulation for faster inference by using different data structures, such as octrees (Yu et al., 2021a), or voxels (Liu et al., 2020; Sun et al., 2021a), by using thousands of tiny MLPs to represent smaller parts of the given scene (Reiser et al., 2021) as well as by factorizing the rendering formula (Hedman et al., 2021; Garbin et al., 2021). Another option for faster training is to adopt meta-learning techniques to start the learning process from a better initialization point (Tancik et al., 2021). One of the main contributions of the the original NeRF model is the fact that it uses harmonic embeddings (Mildenhall et al., 2020; Tancik et al., 2020) that encode scalar position and view direction values with a multi-resolution sequence of sine and cosine functions. As shown by Tancik et al. (2020), using harmonic mappings profoundly help learning high frequency functions of 3D scenes from low-dimensional coordinate values. Müller et al. (2022) have recently showcased how replacing harmonic embeddings with a multi-resolution hash table of trainable feature vectors permits the use of a smaller MLPs without sacrificing quality, hence leading to faster computation (which can be further improved by highly-optimized implementation). On the other hand, in order to improve the training efficiency of a NeRF model, Xu et al. (2022) first construct a point cloud based on registered images and then propose to aggregate information from k-nearest points to the sampled ray point while rendering with the ray-marching formulation used for the original NeRF model. This does not only result in a compact representation of the scene but also in faster training times.

Apart from training and inference times, NeRFs suffer from the need for large collections (tens to hundred(s)) of registered images due to greedy scheme of learning. The geometry of a given scene is learned by exhaustive sampling of millions of points in 3D space repeatedly. In order to improve the data efficiency of the NeRF model, different strategies that employ geometrical principles or semantic constraints have been developed. One idea is to enforce semantic consistency across viewpoints via a pre-trained image classification network (Jain et al., 2021) with the aim of training a NeRF model with fewer images. Kim et al. (2021) have achieved similar rendering quality as the original model with dramatically fewer number of images by ensuring compactness of scenes along individual rays and enforcing consistency across rays in a neighborhood. Similarly, regularizing the geometry and appearance of patches rendered from unobserved viewpoints leads to better data efficiency and improves the few-shot rendering problem (Niemeyer et al., 2021), where the number of observed images is very limited. Depth supervision is also demonstrated to lead to faster training of NeRF with fewer input images (Deng et al., 2022).

On the other hand, the bottleneck of generalization is generally targeted by borrowing ideas from multi-view geometry. Concurrent works (Trevithick and Yang, 2021; Chen et al., 2021; Wang et al., 2021b) mitigate the need for per-scene optimization by conditioning per-pixel rendering function on back-projected image features from neighbouring viewpoints. These features can be aggregated across those viewpoints before the rendering volume function as in GRF (Trevithick and Yang, 2021) or they can be aggregated within the NeRF framework and used as a residual feature as in PixelNeRF (Yu et al., 2021b). On the contrary, these features can be used to learn how to aggregate the color values inferred from neighbouring viewpoints as in IBRNet (Wang et al., 2021b). From a different perspective, MVSNeRF (Chen et al., 2021) proposes a low-resolution plane-swept cost volume (Nozick et al., 2008) to generalize to new scenes when very few viewpoints are available, hence, targeting a few-shot generalizable neural rendering model. GeoNeRF (Johari et al., 2022) further improves MVSNeRF with cascaded cost volumes and using an attention based approach for aggregating images features from different viewpoints. Overall, this active line of research shows how neural rendering models can be implemented to generalize over previously unseen scenes when the right 3D inductive biases are incorporated.

However, all the above *conditional* NeRF models require backprojection of each sampled 3D point along an associated ray onto available viewpoints in a greedy manner. If we want to render images for multiple novel viewpoints, this greedy approach requires redundantly many computations, which can be exchanged with memory by storing the conditioned features in a volumetric representation. Despite the successful scene-specific optimization of volumetric representations (Liu et al., 2020; Sun et al., 2021a; Lazova et al., 2022; Fang et al., 2022), to the best of our knowledge, our method is the first one to bridge the gap between conditional NeRF models and voxel based representations for neural rendering.

5.2.3 Compositionality in Neural Rendering

Despite the great efforts devoted to improve the NeRF model for data or computational efficiency, compositionality stayed largely undiscovered till very recently. When achieved, compositional models in 3D can serve many applications in robotics and autonomous navigation as the intelligent agents, which is a term used for anything that can take decisions based on sensory input, take actions in three dimensions, mostly guided by 2D sensory input. Compositional models can also advance efficient and physically plausible scene editing.

The first attempt to retain any compositionality within NeRF models is proposed by Zhang et al. (2020) with a decomposition of the integral operator in ray marching formulation in Equation (5.1). They successfully model foreground and background information; however, it lacks any notion of further compositional granularity, such as objects or object parts. For a finer compositional structure, Yu et al. (2021c) propose to use two NeRF instances for rendering foreground and background information, and condition the foreground instance on object specific latent variables and the background NeRF instance on a single background latent variable, respectively. The latent vectors for objects or for the background are all inferred from 2D images. Similarly, Stelzner et al. (2021) replace the decoder of the SlotAttention (Locatello et al., 2020) model with a single NeRF instance for rendering and composing the scene from RGB-D data. Yang et al. (2021) propose using object and background path-ways for rendering. However, they use two trainable feature volumes as in (Liu et al., 2020) for the scene and the objects, while each object is further characterized by an instance-specific code, and supervised by an instance-segmentation map for 2D images. Xie et al. (2021) also adopt a conditional approach based on object-specific latent vectors. However, they work with scenes with single objects from a single category and infer a segmentation of the category-specific instance at inference time.

Targeting a more object-centric or semantic decomposition for scene editing Kobayashi et al. (2022) propose distilling features with off-the-shelf image feature extractors towards editable neural rendering fields. They map each 3D coordinate to a semantic feature descriptor and disambiguate semantically similar parts of a given scene based on a query. However, the feature descriptors are optimized to be scene-specific, which heavily limits the generalization ability of the method.

Another small group of work pursues compositionality in 3D by the help of motion information. Yuan et al. (2021) assume multi-view data at multiple subsequent time steps and use a two-pathway approach for modelling static and dynamic parts of a scene, where the latter is aligned with the static counterpart via continuous rigid body motion. The continuity assumption of the learned rigid body motion helps the method to decompose a scene into static and dynamic parts, where the dynamic part is presented to be always a single element in their experiments. Moreover, synchronized multi-view videos are not as easily accessible as multi-view images or monocular videos. On the other hand, D^2NeRF (Wu et al., 2022) requires only a monocular video to successfully segment and decouple dynamic objects and the static background. Nevertheless, both of the models learn to represent a single dynamic scene at a time and lack the ability decompose the dynamic content further into object level instance.

To the contrary of above methods that can achieve decompositionality only at the foreground-background level for static scenes or at the static-dynamic level for dynamic scenes, there are few works that attempt to construct object-level compositional neural rendering models. Yet they cannot decompose a given scene. For example, Guo et al. (2020) propose representing each object with a dedicated implicit function for light transport, which is learned individually. They later use this *object corpus* to render a scene given object bounding boxes and illumination conditions. Similarly, Yang et al. (2022) rely on pre-captured objects to understand a complex scene with an unknown arrangement guided by neural rendering. Zhang et al. (2021) further extends editable and composable neural rendering models to free view video rendering.

Overall, there is a clear interest in compositional neural rendering, with few works that can also decompose a given scene. However, these decompositional methods motivate decoupling using 2D clues and do not fully exploit the 3D structure imposed by the registered image inputs. We propose below to use high representation capacity of explicit volumetric grids to abstract entities directly using the aggregated information in three-dimensions. We also target decoupling multiple objects given two sets of images acquired at different times without any need for continuous data acquisition, or synchronization in time. Our method is generic and can be applied to different scenes provided that the volumetric feature representation can be constructed.

5.3 Proposed Method

In this section, we first present the fundamentals of Neural Radiance Fields (NeRFs) and formally introduce our problem formulation. Then, we describe the proposed method for constructing a feature volume from given observations of a scene and rendering based on these volumetric representations. We later explain the proposed unsupervised method for entity abstraction in 3D based on clustering volumetric units through feature correspondences. We finish this section by outlining our implementation.

5.3.1 Background - Neural Radiance Fields

Neural Radiance Fields are a novel representation of 3D scenes by a continuous volumetric scene function that enables synthesizing novel views of the given scene. In the original formulation (Mildenhall et al., 2020), a NeRF model outputs the view-dependent emitted color $\mathbf{c} = (r, g, b)$ and volume density σ for each point $\mathbf{x} = (x, y, z)$ in the 3D space seen from direction (θ, ϕ), which can be expressed by a 3D Cartesian unit vector $\mathbf{d}$. Following the classical volume rendering technique of ray marching (Kajiya and Von Herzen, 1984), the expected color C of a camera ray $\mathbf{r}$ parametrized by t, *i.e.*, $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, can be expressed as in Equation (5.1) where t_n and t_f denote near and far bounds, and the function $T(t)$ stands for the accumulated transmittance.

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma(\mathbf{r}(t))\; \mathbf{c}(\mathbf{r}(t), \mathbf{d})\; dt \tag{5.1}$$

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds\right) \tag{5.2}$$

This continuous integral is approximated by quadrature using a discrete set of points that are sampled from evenly spaced bins, as expressed in Equation (5.3). The approximated color for each ray, $\hat{C}(\mathbf{r})$, hence corresponding pixel color is computed according to Equation (5.4) using the discretized transmittance in Equation (5.5), where δ stands for the distance between sampled points.

$$t_i \sim U\left[t_n + \frac{i-1}{N}\left(t_f - t_n\right),\; t_n + \frac{i}{N}\left(t_f - t_n\right)\right] \tag{5.3}$$

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i\, \left(1 - \exp(-\sigma_i \delta_i)\right) \mathbf{c}_i \tag{5.4}$$

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \tag{5.5}$$

Mildenhall et al. (2020) use a multilayer perceptron (MLP) to map harmonic embeddings of the sampled locations and camera direction to the color and volume density values. Harmonic embeddings map a scalar value to a sequence of L sine and cosine functions such that $\gamma(x) = \left[\sin(2^0 x), \sin(2^1 x), \dots, \sin(2^{L-1} x); \cos(2^0 x), \cos(2^1 x), \dots, \cos(2^{L-1} x)\right]$. The independent encodings of the 3-dimensional location input, $\gamma(\mathbf{x} = (x, y, z))$ are first mapped to a middle representation $\mathbf{e}$, such that, $\mathbf{e} = F^1(\gamma(\mathbf{x}))$. This representation is later mapped to view-dependent emitted color, $\mathbf{c} = F^{\text{color}}(\mathbf{e}, \gamma(\mathbf{d}))$, and density, $\sigma = F^{\text{density}}(\mathbf{e})$. Each F function is parametrized by an MLP and the overall function can be summarized as $F_\Theta : \gamma(\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$.

Mildenhall et al. (2020) further follow a hierarchical volume sampling via two instances of the same network, referred to as coarse and fine networks, where a second set of points are sampled using the output of the first network, which are expected to be more biased towards the relevant parts of the volume. The training objective is set to the squared error for pixel

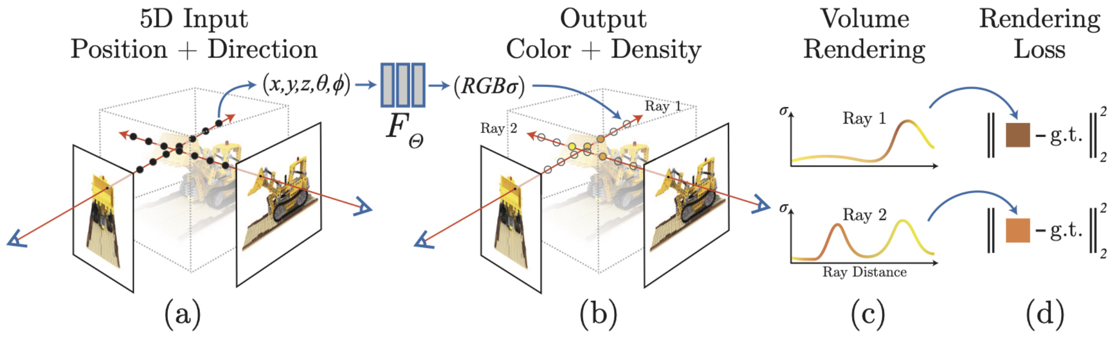


Figure 5.1: Illustration of the NeRF model: using the 5D coordinates (location and viewing direction)(a), color and volume density of sampled points(b) are computed for synthesizing the image for a query viewpoint via differentiable ray marching operation(c). The parameters of the mapping function F_Θ are learned by minimizing the discrepancy between the rendered image and the ground truth(d). Image taken from Mildenhall et al. (2020).

color for each ray in the batch, where the error is computed for outcomes of both coarse and fine networks. Figure 5.1 illustrates the main principles of the differentiable rendering pipeline of the NeRF model.

As a virtue of their compact representation, Neural Radiance Fields have transformed many tasks in both computer graphics and computer vision research: they represent a 3D scene through parameters of a continuous-valued function that maps each point in the 3D world to a density and color value when seen from a particular direction. Unlike point clouds, it is a dense representation and in contrast to voxel representations, it does not suffer from memory bottleneck for high-resolution expression of scenes. Furthermore, it can be conveniently learned from a set of registered images of a scene, without any need for manual modeling. Its differentiable and intuitive formulation makes NeRFs a suitable tool for 3D computer vision tasks provided that the compute need associated with its training can be afforded.

5.3.2 Problem Formulation and Notations

In order to abstract entities in three dimensions, we assume registered images for two different states of a scene, which we will refer to as arrangements, where the composing entities are relocated. In other words, we observe a scene at two different time instances from multiple viewpoints and we assume that the volumetric changes between those two time instances can be attributed to its composing entities. Our aim is to represent the static part of the scene, as well as its relocated composing entities, in 3D in a way that we can synthesize any views for each of them individually. The problem setup is illustrated in Figure 5.2.

Hence, we represent the scene in three dimensions by a combination of an arrangement-specific feature volume and an arrangement-agnostic function that maps any 3D point to color and volume density values using the features. We construct the feature volumes based on observations, *i.e.*, images, of the corresponding state. The mapping function, which is an

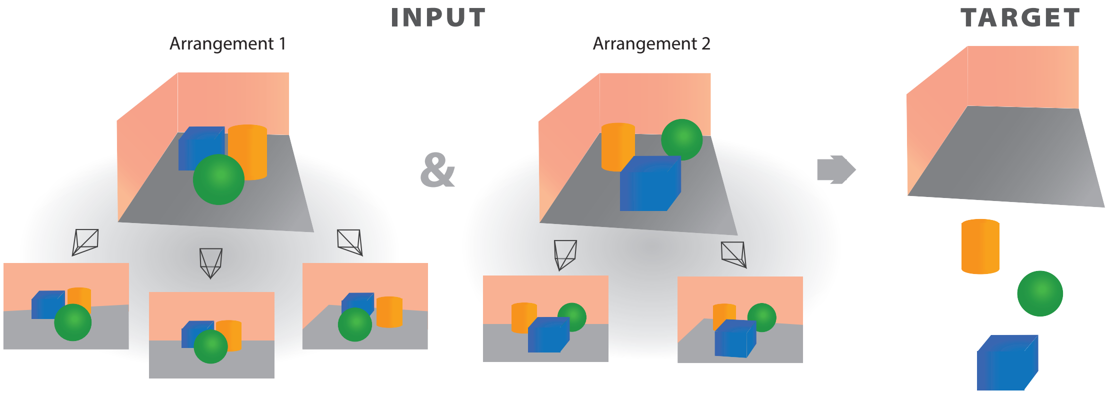


Figure 5.2: Problem set-up: we assume registered images of a scene in two states when the composing entities are relocated: we refer to the different states of the scene as arrangements. Our target is to decompose M radiance fields to represent the relocated entities in 3D.

example of conditional NeRFs, then uses the interpolated features of any sampled point to output color and volume density values, which can be used to render an image from a query viewpoint using the ray marching in NeRF formulation.

More formally, given a set of N images, $I_i, i \in \{1, \ldots, N\}$, with corresponding camera intrinsic parameters $\mathbf{K}_i$ and extrinsic parameters $\mathbf{E}_i = [\mathbf{R}_i | \mathbf{t}_i]$; our aim is to infer M radiance fields as partitions of an underlying bounded scene. For this purpose, we first construct a feature volume $\mathbf{V} \in \mathbb{R}^{H_\mathbf{V} \times W_\mathbf{V} \times D_\mathbf{V} \times C_\mathbf{V}}$, where $H_\mathbf{V}, W_\mathbf{V}, D_\mathbf{V}$ denote the spatial dimensions of a the feature volume and $C_\mathbf{V}$ is the volume feature dimension. We then use the implicit function F_Θ for rendering images based on the volumetric representation $\mathbf{V}$.

In fact, we use two sets of images, $\{I_i^{(1)}\}_{i=1}^{N_1}$ and $\{I_i^{(2)}\}_{i=1}^{N_2}$ to construct the corresponding feature volumes $\mathbf{V}^{(1)}$ and $\mathbf{V}^{(2)}$ guided by the same implicit function F_Θ. The feature volumes are constructed by lifting image features of the corresponding arrangement to world coordinates based on camera parameters, and rendering is achieved by a conditional NeRF representation. We then find directed feature correspondences $\mathbf{P}^{(1)\to(2)}$, or $\mathbf{P}^{(2)\to(1)}$, between feature volumes $\mathbf{V}^{(1)}$ and $\mathbf{V}^{(2)}$. We cluster M subset of voxels $\mathbf{u}_{(i)}^{(1)}, \mathbf{u}_{(i)}^{(2)}$ where $i \in 1, \ldots, M$ based on correspondences, which stand for re-located entities. Each subset of voxels is described by their 3D coordinates and corresponding features, which enables NeRF-based novel view synthesis for each composing entity.

5.3.3 Lifting and Fusing 2D Features.

For the feature volume construction, we use a subset of N_f images such that $N_f < N_1$ and $N_f < N_2$, and they will be occasionally referred to as keyframes from now on. We use an image encoder to enrich the color information at each pixel, and map R,G,B colors to a higher dimensional feature vector which is more likely to contain any semantic or geometric information. The image encoder will be denoted by $\xi : \mathbb{R}^{H_\mathrm{I} \times W_\mathrm{I} \times 3} \to \mathbb{R}^{H_\mathrm{I} \times W_\mathrm{I} \times C_\mathrm{I}}$, where C_I

indicates the dimension of image features and $(H_\mathbf{I}, W_\mathbf{I})$ indicate image dimensions. Then, all points on the 3D grid of shape $H_\mathbf{V} \times W_\mathbf{V} \times D_\mathbf{V}$, *i.e.*, $\mathbf{x} = (x, y, z) \in \left\{\{x_i\}_{i=1}^{H_\mathrm{V}} \times \{y_j\}_{j=1}^{W_\mathrm{V}} \times \{z_k\}_{k=1}^{D_\mathrm{V}}\right\}$ are projected onto N_f keyframe viewpoints using the corresponding camera parameters. More specifically, the point $\mathbf{x}$ is projected to the j^{th} keyframe as $\mathbf{x}^j$, where $j \in 1, \ldots, N_f$, using camera intrinsic parameters $\mathbf{K}_j$ and extrinsic parameters $\{\mathbf{R}_j, \mathbf{t}_j\}$ according to Equation (5.6). The sub-pixel coordinates (u^j, v^j) in the corresponding image $\mathbf{I}_j$ is then computed by assuming a pinhole camera model as expressed in Equation (5.7).

$$[x^j, y^j, z^j]^T = \mathbf{R}_j^T(\mathbf{K}_j^{-1}[x, y, z]^T - \mathbf{t}_j) \tag{5.6}$$

$$u^j = x^j / z^j, \quad v^j = y^j / z^j \tag{5.7}$$

We first check if projected points lie within the camera frustum[1] of the corresponding viewpoint, and place zeros as features if the point is outside the associated observable space. Otherwise, for a 3D point within the camera frustum with projected sub-pixel coordinates (u^j, v^j), we use bilinear interpolation to compute $\xi\left(\mathbf{I}_j(u^j, v^j)\right)$ from the feature vectors of 4-neighboring pixels. The resultant feature vector will also be denoted by $\xi^j(\mathbf{x})$ for clarity (Note that $\xi(\cdot)$ denotes the generic image encoding function whereas $\xi^j(\mathbf{x})$ depicts the value of resultant features at pixel coordinates associated with 3D point $\mathbf{x}$ when it is projected to jth keyframe).

After we project the point $\mathbf{x}$ onto all keyframes and sample corresponding image features, we have a set of N_f feature vectors, some of which might be padded with zeros as a result of not being observed from the corresponding viewpoint. The goal is to represent $\mathbf{x}$ with a single feature vector that aggregates the information from N_f feature vectors. While aggregating features from different viewpoints, we need to take the possibility of occlusion into account: the point $\mathbf{x}$ can be occluded by other entities in the scene when observed from a subset of given viewpoints. Hence, before aggregating, we propose to enhance the set of feature vectors $\{\xi^j(\mathbf{x})\}_{j=1}^{N_f}$ by augmenting information across viewpoints. In order to help with uncovering occlusions, we first concatenate each feature vector with color values at corresponding sub-pixel locations, $\mathbf{I}_j(u^j, v^j)$, again bilinearly interpolated from neighboring pixels. Then, the function $\psi : \mathbb{R}^{N_f \times C_\mathrm{I}+3} \rightarrow \mathbb{R}^{N_f \times C_\mathrm{V}}$ outputs the enhanced features as $\bar{\xi}^j(\mathbf{x})$ by $\{\bar{\xi}^j(\mathbf{x})\}_{j=1}^{N_f} = \psi\left(\left\{[\xi^j(\mathbf{x}), \mathbf{I}_j(u^j, v^j)]\right\}_{j=1}^{N_f}\right)$, where $[\cdot, \cdot]$ denotes concatenation.

In order to obtain a single feature vector at the corresponding 3D point, we need a permutation invariant aggregation function to summarize the information acquired from all available viewpoints. Some common choices for permutation invariant aggregation functions are "mean" or "maximum" operators. It means, for the mean operator, the aggregator function averages the N_f for each of $C_\mathbf{V}$ dimensions. We denote the aggregation function as $\zeta : \mathbb{R}^{N_f \times C_\mathrm{V}} \rightarrow \mathbb{R}^{C_\mathrm{V}}$, and it obtains the feature at each location $\mathbf{x} = (x, y, z)$, such that $\mathbf{V}(\mathbf{x}) = \zeta\left(\{\bar{\xi}^j(\mathbf{x}))\}_{j=1}^{N_f}\right)$. The feature volume construction is briefly illustrated in Figure 5.3, end summarized in Equations (5.8)-(5.13).

---

[1] The truncation with parallel planes describing the visible space in 3D world that may appear on the screen

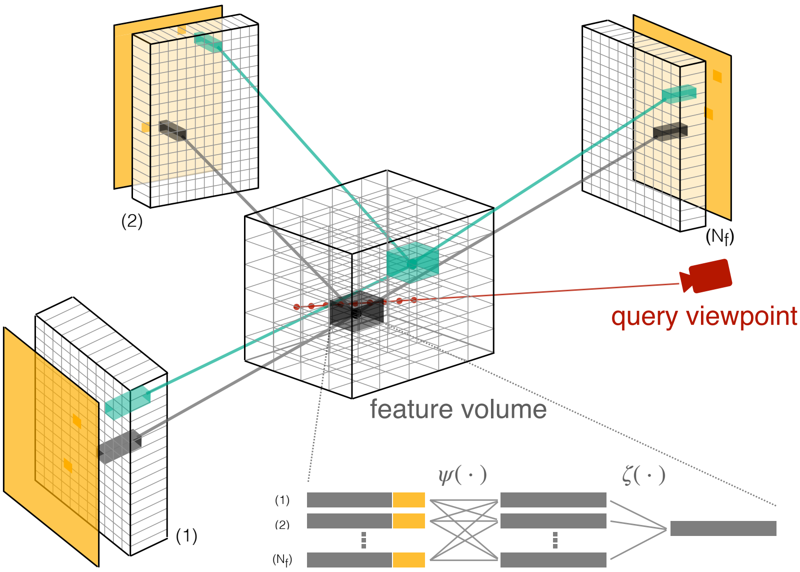


Figure 5.3: Proposed feature volume construction: we first project all volume unit centers to available viewpoints and sample 2D features at the projected pixel locations (depicted as gray vectors), as well as the corresponding color values in the images (depicted as yellow vectors). After updating the set of sampled features by enabling interactions across different viewpoints, denoted by function $\psi(\cdot)$, we aggregate the features with function $\zeta(\cdot)$.

$$\begin{align}
&\text{image encoder:} && \xi : \mathbb{R}^{H_\mathrm{I}\times W_\mathrm{I}\times 3} \rightarrow \mathbb{R}^{H_\mathrm{I}\times W_\mathrm{I}\times C_\mathrm{I}} \tag{5.8}\\
&\text{sampled image feature:} && \xi^j(\mathbf{x}) = \xi\left(\mathbf{I}_j(u^j, v^j)\right)\ (j^{th}\ \text{keyframe}) \tag{5.9}\\
&\text{cross-view feature enhancer:} && \psi : \mathbb{R}^{N_f\times(C_\mathrm{I}+3)} \rightarrow \mathbb{R}^{N_f\times C_\mathbf{V}} \tag{5.10}\\
&\text{enhanced features:} && \left\{\bar{\xi}^j(\mathbf{x})\right\}_{j=1}^{N_f} = \psi\left(\left\{[\xi^j(\mathbf{x}), \mathbf{I}_j(u^j, v^j)]\right\}_{j=1}^{N_f}\right) \tag{5.11}\\
&\text{aggregation function:} && \zeta : \mathbb{R}^{N_f\times C_\mathbf{V}} \rightarrow \mathbb{R}^{C_\mathbf{V}} \tag{5.12}\\
&\text{aggregated features:} && \mathbf{V}(\mathbf{x}) = \zeta\left(\left\{\bar{\xi}^j(\mathbf{x})\right\}_{j=1}^{N_f}\right) \tag{5.13}
\end{align}$$

5.3.4 Radiance Fields with Voxel Features

Once the feature volume is constructed, we adopt conditional NeRF models for synthesizing images. Instead of harmonic positional embeddings, each point in 3D is represented by a $C_\mathbf{V}$-dimensional feature vector that is obtained from the feature volume $\mathbf{V}$ via trilinear interpolation[2]. More specifically, instead of $\gamma(\mathbf{x})$, we use $\mathbf{V}(\mathbf{x})$ for the middle representation in NeRF formulation, *i.e.*, $\mathbf{e} = F^1(V(\mathbf{x}))$, which is later mapped to density, $\sigma = F^{\text{density}}(\mathbf{e})$, and

---

[2] Trilinear interpolation is an adaptation of liner interpolation to three dimensions: it approximates the value of a function at any intermediate point (x, y, z) within the local axial rectangular prism linearly, using function data on the lattice point

color, $\mathbf{c} = F^{\text{color}}(\mathbf{e}, \gamma(\mathbf{d}))$ by using the direction $\mathbf{d}$ of the query viewpoint for obtaining the viewpoint-dependent color information. Note that the scene geometry solely depends on the inferred feature, which forces the image encoder, the feature enhancer, and the aggregation function to result in a feature volume that represents well the scene geometry.

We also adopt the same hierarchical sampling strategy of the original NeRF algorithm for sampling points along each ray. We use a coarse and a fine network, described as $F_{\Theta}^{\text{coarse}} : \big(\mathbf{V}(\mathbf{x}), \gamma(\mathbf{d})\big) \rightarrow (\mathbf{c}^{\text{coarse}}, \sigma^{\text{coarse}})$ and $F_{\Theta}^{\text{fine}} : \big(\mathbf{V}(\mathbf{x}), \gamma(\mathbf{d})\big) \rightarrow (\mathbf{c}^{\text{fine}}, \sigma^{\text{fine}})$, and use the output of the coarse network to sample more informed points along the ray towards denser regions. More specifically, the resultant color of a ray $\mathbf{r}$ in Equation (5.4) can be rewritten as $\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i^{\text{coarse}}$ where N_c stands for the number of points sampled along the ray $\mathbf{r}$ for the evaluation of the coarse network. Upon normalization of weights w_i's by $\hat{w}_i = \frac{w_i}{\sum_{j=1}^{N_c} w_j}$, we obtain a piecewise-constant PDF along the ray. The second set of N_f points are then sampled using inverse transform sampling[3], and the fine network is evaluated with all $N_c + N_f$ sampled points to obtain $\hat{C}_f(\mathbf{r})$.

5.3.5 Feature Correspondance

After we construct the feature volumes $\mathbf{V}^{(1)}, \mathbf{V}^{(2)}$ such that $\mathbf{V}^{(1)}, \mathbf{V}^{(2)} \in \mathbb{R}^{H_\text{V} \times W_\text{V} \times D_\text{V} \times C_\text{V}}$, we continue with the task of associating volume units across $\mathbf{V}^{(1)}$ and $\mathbf{V}^{(2)}$. As the volume is expected to be sparse due to the void in the 3D world, a unit-to-unit association of two feature volumes is not only computationally expensive but also redundant. Hence, we first discard voxel units in both volumes, the density of which falls below a threshold. Note that the density of each volume unit can be computed with the density branch of the fine NeRF module using the features accommodated at the voxel unit. More specifically, we work with subsets of voxel units $\overline{\mathbf{V}}^{(1)}$ and $\overline{\mathbf{V}}^{(2)}$ such that $\overline{\mathbf{V}}^{(1)} = \big\{\mathbf{V}^{(1)}(\mathbf{x}) \mid F^{\text{density,fine}}\big(F^1\big(\mathbf{V}^{(1)}(\mathbf{x})\big)\big) < \tau^{(1)}\big\}$, and, $\overline{\mathbf{V}}^{(2)} = \big\{\mathbf{V}^{(2)}(\mathbf{x}) \mid F^{\text{density,fine}}\big(F^1\big(\mathbf{V}^{(2)}(\mathbf{x})\big)\big) < \tau^{(2)}\big\}$ for all $\mathbf{x} = (x, y, z) \in \Big\{\{x_i\}_{i=1}^{H_\text{V}} \times \{y_j\}_{j=1}^{W_\text{V}} \times \{z_k\}_{k=1}^{D_\text{V}}\Big\}$. $\tau^{(1)}, \tau^{(2)}$ are determined according to the maximum density computed for voxel units at each feature volume. We similarly represent the center coordinates of voxel units of the pruned feature volumes with $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, such that $\mathbf{x}^{(1)} = \big\{\mathbf{x} \mid F^{\text{density,fine}}\big(F^1\big(\mathbf{V}^{(1)}(\mathbf{x})\big)\big) < \tau^{(1)}\big\}$ and $\mathbf{x}^{(2)} = \big\{\mathbf{x} \mid F^{\text{density,fine}}\big(F^1\big(\mathbf{V}^{(2)}(\mathbf{x})\big)\big) < \tau^{(1)}\big\}$ for all $\mathbf{x} = (x, y, z) \in \Big\{\{x_i\}_{i=1}^{H_\text{V}} \times \{y_j\}_{j=1}^{W_\text{V}} \times \{z_k\}_{k=1}^{D_\text{V}}\Big\}$.

One bottleneck that arises from discarding volume units with associated low-density values is the fact that pruned feature volumes might contain a different number of voxel units, i.e., $|\overline{\mathbf{V}}^{(1)}| \neq |\overline{\mathbf{V}}^{(2)}|$ ($|\cdot|$ denotes cardinality). This prevents unit-to-unit, hence one-to-one association based on voxel features, and resultant many-to-one association can substantially harm the estimation of rigid body transformation for clusters. Hence, we opt for a stochastic mapping between two pruned feature volumes, denoted by a doubly stochastic matrix $\mathbf{P}$ such that $\mathbf{P}^{(1)\rightarrow(2)} \in [0,1]^{|\overline{\mathbf{V}}^{(1)}| \times |\overline{\mathbf{V}}^{(2)}|}$, where each row and column of the matrix $\mathbf{P}$ sums up to 1. Stochastic mapping permits many-to-many associations with an associated level of strength, which lets

---

[3] Inverse transform sampling is a method for generating random samples from a probability distribution using its cumulative distribution function.

us adapt their contribution to the estimation of rigid body transformations.

The problem of finding feature correspondences between two pruned feature volumes can be considered as mapping two probability distributions that can be observed through sampled features $\overline{\mathbf{V}}^{(1)}$ and $\overline{\mathbf{V}}^{(2)}$ at locations $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$. If we characterize the distributions by $p(x)$ and $q(x)$, respectively, the matrix $\mathbf{P}^{(1)\to(2)}$ describes how much probability mass from one point in the support of $p(x)$ is assigned to point in the support of $q(x)$. Furthermore, the cost associated with moving a voxel unit from $\mathbf{x}^{(1)}$ to $\mathbf{x}^{(2)}$, in other words, establishing correspondence between two volume units, can be defined by a cost value $\mathbf{C}_{ij}$, where $i \in \{1,\ldots,|\mathbf{x}^{(1)}|\}$, and $j \in \{1,\ldots,|\mathbf{x}^{(2)}|\}$. Hence, the total cost of establishing correspondences between $\overline{\mathbf{V}}^{(1)}$ and $\overline{\mathbf{V}}^{(2)}$ can be calculated by the Frobenius inner product between $\mathbf{P}^{(1)\to(2)}$ and $\mathbf{C}$ such that $< \mathbf{C}, \mathbf{P}^{(1)\to(2)} >= \sum_{ij} \mathbf{C}_{ij} \mathbf{P}_{ij}^{(1)\to(2)}$. Note that if the entries of the cost matrix $\mathbf{C}$ are defined by a distance, hence $\mathbf{C}_{ij} = \mathbf{C}_{ji}$, it means that the coupling matrix is not directional, and can be described by $\mathbf{P}$ such that $\mathbf{P} = \mathbf{P}^{(1)\to(2)} = \mathbf{P}^{(2)\to(1)^T}$ where the operator ^{T} denotes matrix transpose.

Minimization of the total cost associated with finding correspondences is the problem of Optimal Transport (Villani, 2009), which aims to find the lowest cost $L_{\mathbf{C}}$ over all possible coupling matrices. Particularly, Kantorovich formulation (Kantorovich, 1942) solves the problem in Equation (5.14) with constrains expressed in Equations (5.15)-(5.16), where $\mathbf{1}$ stands for a column vector of 1s, and vectors $\mathbf{a}, \mathbf{b}$ represent probability weight vectors for $p(x)$ and $q(x)$, respectively.

$$L_{\mathbf{C}} = \min_{\mathbf{P}} < \mathbf{C}, \mathbf{P} > \tag{5.14}$$

$$\text{subject to} \quad \mathbf{P}\mathbf{1} = \mathbf{a}, \tag{5.15}$$

$$\mathbf{P}^T\mathbf{1} = \mathbf{b} \tag{5.16}$$

Although obtaining the solution to the problem presented above is not computationally trivial, entropic regularization allows the use of a simple alternate minimization scheme as a result of the introduced convexity. Sinkhorn's algorithm (Cuturi, 2013) indeed proposes that the solution to the relaxed problem in Equations (5.17)-(5.20), which has $|\mathbf{x}^{(1)}| \cdot |\mathbf{x}^{(2)}|$ variables and $|\mathbf{x}^{(1)}| + |\mathbf{x}^{(2)}|$ constraints in our formulation, can be obtained with iterative updating of two vectors of sizes $|\mathbf{x}^{(1)}|, |\mathbf{x}^{(2)}|$. Although these iterative updates can suffer from numerical instabilities, log-domain computations are shown to improve the stability of the method (Peyré and Cuturi, 2018).

$$L_{\mathbf{C}} = \min_{\mathbf{P}} < \mathbf{C}, \mathbf{P} > -\epsilon H(\mathbf{P}) \tag{5.17}$$

$$\text{subject to} \quad \mathbf{P}\mathbf{1} = \mathbf{a}, \tag{5.18}$$

$$\mathbf{P}^T\mathbf{1} = \mathbf{b} \tag{5.19}$$

$$\text{where} \quad H(\mathbf{P}) = -\sum_{ij} \mathbf{P}_{ij} \log \mathbf{P}_{ij} \tag{5.20}$$

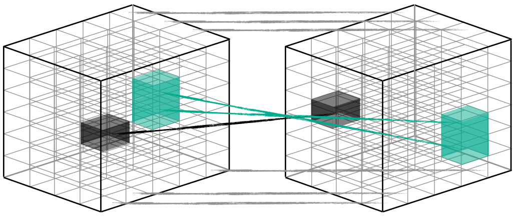


Figure 5.4: Hypothetical feature matching for entity abstraction

We approach the feature correspondence problem as finding a coupling that minimizes the distance between features of associated voxel units. More formally, we construct our cost matrix with the squared Euclidean distance between voxel feature pairs, as in Equation (5.21). We then use Sinkhorn's algorithm in the log domain to solve the minimization problem. We refer the readers to the comprehensive work by Peyré and Cuturi (2018) for further details on Sinkhorn's algorithm.

$$\mathbf{C}_{ij} = \|\mathbf{V}^{(1)}(\mathbf{x}_i) - \mathbf{V}^{(2)}(\mathbf{x}_j)\|_2^2 \tag{5.21}$$

$$\text{where} \quad \mathbf{x}_i \in \mathbf{x}^{(1)}, \mathbf{x}_j \in \mathbf{x}^{(2)} \tag{5.22}$$

It is important to emphasize that the coupling is obtained based on voxel features so that $\mathbf{P}^{(1)\to(2)}\mathbf{V}^{(1)}/\mathbf{a} \approx \mathbf{V}^{(2)}$, yet, it can be also used to obtain the canonical coordinates of mapped features in the second feature volume. We will denote the coordinates of mapped features by $\mathbf{x}^{(1)\to(2)}$ such that $\mathbf{x}^{(1)\to(2)} = \mathbf{P}^{(1)\to(2)}\mathbf{x}^{(1)}/\mathbf{a}$. Note that $|\mathbf{x}^{(1)}| = |\mathbf{x}^{(1)\to(2)}|$, which is not necessarily the same as $|\mathbf{x}^{(2)}|$. The correspondence between the world coordinates of the matched features will allow the estimation of a set of rigid body transformations, which leads to our 3D entity abstraction algorithm, which is described in the next section.

5.3.6 Entity Abstraction

For abstracting relocated entities directly in three dimensions, we propose to cluster voxel units based on the feature correspondence obtained by the method explained in the previous section. The hypothetical matching with voxel features is illustrated in Figure 5.4.

For this purpose, we follow a greedy approach. We begin with a randomly chosen voxel unit $\mathbf{x}_o^{(1)}$ from the pruned features $\overline{\mathbf{V}}^{(1)}$ and find other voxels in $\mathbf{x}^{(1)}$ that are in a spatial proximity of the chosen voxel $\mathbf{x}_o^{(1)}$. We represent the initial cluster of chosen voxels with a subscript $\{i\}$, *i.e.*, $\mathbf{x}_{\{i\}}^{(1)}$, where $\{i\}$ stands for a set of indices, not a single voxel unit. We then determine the mapped coordinates $\mathbf{x}_{\{i\}}^{(1)\to(2)}$. In order to choose the voxel units from the subset $\{i\}$ that have consistent feature similarity, we compute the features at mapped coordinates via trilinear interpolation, *i.e.*, $\mathbf{V}^{(2)}(\mathbf{x}_{\{i\}}^{(1)\to(2)})$ and eliminate the voxels from the subset $\{i\}$ with $\|\overline{\mathbf{V}}_{\{i\}}^{(1)} - \mathbf{V}^{(2)}(\mathbf{x}_{\{i\}}^{(1)\to(2)})\|_2^2 > \tau_F$, where τ_F is a hyperparameter. The updated indices are then represented by $\{j\}$, and $|\{j\}| \leq |\{i\}|$.

Once we determine a set of world coordinates $\mathbf{x}^{(1)}_{\{j\}}$ and $\mathbf{x}^{(1)\to(2)}_{\{j\}}$ based on cross-arrangement feature similarity, we compute the rigid body transformation described by a 3×3 rotation matrix $\mathbf{R}_j$ and 3-dimensional translation vector $\mathbf{t}_j$ with the least-squares estimation proposed by Umeyama (1991). In other words, we look for the optimal transformation $[\mathbf{R}_j|\mathbf{t}_j]$ that minimizes $\|\mathbf{R}_j(\mathbf{x}^{(1)}_{\{j\}} - \mathbf{t}_j) - \mathbf{x}^{(1)\to(2)}_{\{j\}}\|^2_2$. This minimization problem can be solved by computing the 3×3 covariance matrix between sets of 3D points in $\mathbf{x}^{(1)}_{\{j\}}$ and $\mathbf{x}^{(1)\to(2)}_{\{j\}}$, S_j, and computing the singular value decomposition of S_j such that $S_j = U_j \Sigma_j V_j^T$. When we discard the reflections, the rotation matrix we are looking for is given by $\mathbf{R}_j = V_j U_j^T$. The translation vector $\mathbf{t}_j$ can be then computed by $\mathbf{t}_j = \mathbf{R}_j^T \mathbf{x}^{(1)\to(2)}_{\{j\}} - \mathbf{x}^{(1)}_{\{j\}}$.

As the final step, we transform all voxel coordinates of the pruned voxels $\overline{\mathbf{V}}^{(1)}$, $\mathbf{x}^{(1)}$ to the coordinate system of second feature volume, $\mathbf{V}^{(2)}$, with the cluster rigid body transform as $\mathbf{R}_j(\mathbf{x}^{(1)} - \mathbf{t}_j)$ and compute corresponding features via trilinear interpolation: $\mathbf{V}^{(2)}\left(\mathbf{R}_j(\mathbf{x}^{(1)} - \mathbf{t}_j)\right)$. We then expand the indices $\{j\}$ to $\{k\}$ by including all voxel units which establish feature similarity once transformed by $[\mathbf{R}_j|\mathbf{t}_j]$, and still exhibit high volume density values in the second feature volume, such that $\mathbf{x}^{(1)}_{\{k\}} \leftarrow \left\{\mathbf{x} \,\middle|\, \mathbf{x} \in \mathbf{x}^{(1)}, \left\|\overline{\mathbf{V}}^{(1)} - \mathbf{V}^{(2)}\left(\mathbf{R}_j(\mathbf{x} - \mathbf{t}_j)\right)\right\|^2_2 < \tau_F \text{ and } \sigma^{(2)}\left(\mathbf{R}_j(\mathbf{x} - \mathbf{t}_j)\right) > \tau^{(2)}\right\}$, where $\sigma^{(2)}(\cdot)$ estimates the density values in the second volume at a given location via triliniear interpolation from $F^{\text{density,fine}}\left(F^1\left(\mathbf{V}^{(2)}(\mathbf{x})\right)\right)$.

We repeat this procedure for a given number of clusters, or until we cluster all voxel units with consistent feature correspondences. More specifically, we remove the voxels in subset $\{k\}$ from $\overline{\mathbf{V}}^{(1)}$ and $\mathbf{x}^{(1)}$, delete corresponding entries in $\mathbf{P}^{(1)\to(2)}$ and $\mathbf{a}$, and iterate over the steps for (i) finding a spatially localized initial cluster, and corresponding voxels in the second arrangement, (2) discarding the voxels which do not establish feature consistency, (3) computing the rigid body transform between the cluster coordinates in two arrangements, and finally (4) using the transform to map all the voxel coordinates in the pruned feature volume and selecting voxels that present high feature similarity and volume density values.

The procedure for entity abstraction is summarized in Algorithm 1.

5.4 Implementation

This section provides details regarding the implementation of the proposed model. Given two sets of registered images as observations of a scene, which correspond to two different arrangements as illustrated in Figure 5.2, our aim is to represent the scene with arrangement-specific feature volumes and an arrangement-agnostic rendering function. The feature volumes are constructed from a subset of scene images, which are also referred to as keyframes. The image features of keyframes are lifted to canonical coordinates and aggregated across keyframes to obtain a single representation for each unit in the feature volume. The rendering function then samples features from the volume for mapping canonical coordinates to color and density values. Using differentiable ray marching algorithms, our model can render an image from

Algorithm 1: Entity Abstraction in 3D

```
Input: pruned feature volumes: $\overline{\mathbf{V}}^{(1)}, \overline{\mathbf{V}}^{(2)}$, e.g.,
        $\overline{\mathbf{V}}^{(1)} = \left\{\mathbf{V}^{(1)}(\mathbf{x}) \mid F^{\text{density,fine}}\left(F^{1,\text{fine}}\left(\mathbf{V}^{(1)}(\mathbf{x})\right)\right) < \tau^{(1)}\right\}$
        $\overline{\mathbf{V}}^{(2)} = \left\{\mathbf{V}^{(2)}(\mathbf{x}) \mid F^{\text{density,fine}}\left(F^{1,\text{fine}}\left(\mathbf{V}^{(2)}(\mathbf{x})\right)\right) < \tau^{(2)}\right\}$
            for all $\mathbf{x} = (x, y, z) \in \left\{\{x_i\}_{i=1}^{H_\mathrm{V}} \times \{y_j\}_{j=1}^{W_\mathrm{V}} \times \{z_k\}_{k=1}^{D_\mathrm{V}}\right\}$
    coordinates associated with pruned feature volume: $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$, e.g.,
        $\mathbf{x}^{(1)} = \left\{\mathbf{x} \mid F^{\text{density,fine}}\left(F^{1,\text{fine}}\left(\mathbf{V}^{(1)}(\mathbf{x})\right)\right) < \tau^{(1)}\right\}$
        $\mathbf{x}^{(2)} = \left\{\mathbf{x} \mid F^{\text{density,fine}}\left(F^{1,\text{fine}}\left(\mathbf{V}^{(2)}(\mathbf{x})\right)\right) < \tau^{(2)}\right\}$
            for all $\mathbf{x} = (x, y, z) \in \left\{\{x_i\}_{i=1}^{H_\mathrm{V}} \times \{y_j\}_{j=1}^{W_\mathrm{V}} \times \{z_k\}_{k=1}^{D_\mathrm{V}}\right\}$
    coupling from first feature volume to the second: $\mathbf{P}^{(1)\to(2)}$
    weight vector for pruned feature volume $\overline{\mathbf{V}}^{(1)}$ : $\mathbf{a}$
    number of clusters: $K$
    threshold values for feature similarity, spatial vicinity and density: $\tau_F, \tau_d, \tau^{(2)}$
Operators: $V^{(2)}(\mathbf{x})$ : feature interpolation function from $\mathbf{V}^{(2)}(\mathbf{x})$
        $\sigma^{(2)}(\mathbf{x})$ : density interpolation function from $F^{\text{density,fine}}\left(F^{1,\text{fine}}\left(\mathbf{V}^{(2)}(\mathbf{x})\right)\right)$:
        del $(\cdot, \{k\})$ : delete operator for entries correspond to voxels in cluster $\{k\}$
        rand$(\cdot)$: random selection from a given set
Output   : Clusters of features and corresponding 3D coordinates $\left\{\left(\overline{\mathbf{V}}^{(1)}_{\{k\}}, \mathbf{x}^{(1)}_{\{k\}}\right)\right\}_{k=1}^{K}$
Initialize :  $\underline{\mathbf{x}}^{(1)} \leftarrow \mathbf{x}^{(1)}$
              $\underline{\mathbf{V}}^{(1)} \leftarrow \overline{\mathbf{V}}^{(1)}$
              $\underline{\mathbf{P}}^{(1)\to(2)} \leftarrow \mathbf{P}^{(1)\to(2)}$
              $\underline{\mathbf{a}} \leftarrow \mathbf{a}$

for $k \in 1, \dots, K$ do
    $\mathbf{x}_o^{(1)} \leftarrow$ rand$(\underline{\mathbf{x}}^{(1)})$
    $\mathbf{x}_{\{i\}}^{(1)} = \left\{\mathbf{x} \mid \mathbf{x} \in \underline{\mathbf{x}}^{(1)}, \|\mathbf{x} - \mathbf{x}_o^{(1)}\|_2^2 < \tau_d\right\}$
    $\mathbf{x}_{\{i\}}^{(1)\to(2)} = \underline{\mathbf{P}}^{(1)\to(2)} \mathbf{x}_{\{i\}}^{(1)} / \underline{\mathbf{a}}$
    $\left(\mathbf{x}_{\{j\}}^{(1)}, \mathbf{x}_{\{j\}}^{(1)\to(2)}\right) \leftarrow \left\{(\mathbf{x}_1, \mathbf{x}_2) \mid \mathbf{x}_1 \in \mathbf{x}_{\{i\}}^{(1)}, \mathbf{x}_2 \in \mathbf{x}_{\{i\}}^{(1)\to(2)}, \|\underline{\mathbf{V}}_{\{i\}}^{(1)} - V^{(2)}(\mathbf{x}_2)\|_2^2 < \tau_F\right\}$
    $[\mathbf{R}_j | \mathbf{t}_j] \leftarrow \arg\min_{\mathbf{t} \in \mathbb{R}^3, \mathbf{R} \in SO(3)} \|\mathbf{R}_j (\mathbf{x}_{\{j\}}^{(1)} - \mathbf{t}_j) - \mathbf{x}_{\{j\}}^{(1)\to(2)}\|_2^2$
    $\left(\overline{\mathbf{V}}_{\{k\}}^{(1)}, \mathbf{x}_{\{k\}}^{(1)}\right) \leftarrow \left\{\left(\mathbf{V}^{(1)}(\mathbf{x}), \mathbf{x}\right) \mid \mathbf{x} \in \mathbf{x}^{(1)}, \|\overline{\mathbf{V}}^{(1)} - V^{(2)}\left(\mathbf{R}_j (\mathbf{x} - \mathbf{t}_j)\right)\|_2^2 < \tau_F, \sigma^{(2)}\left(\mathbf{R}_j (\mathbf{x} - \mathbf{t}_j)\right) < \tau^{(2)}\right\}$
        $\underline{\mathbf{x}}^{(1)} \leftarrow$ del$(\underline{\mathbf{x}}^{(1)}, \{k\})$
        $\underline{\mathbf{V}}^{(1)} \leftarrow$ del$(\underline{\mathbf{V}}^{(1)}, \{k\})$
      $\underline{\mathbf{P}}^{(1)\to(2)} \leftarrow$ del$(\mathbf{P}^{(1)\to(2)}, \{k\})$
        $\underline{\mathbf{a}} \leftarrow$ del$(\underline{\mathbf{a}}, \{k\})$
end
```

any viewpoint. Hence the combination of the feature volumes and the rendering function represents the scene in truly in three dimensions.

Afterwards, using the two feature volumes, we disambiguate the entities in the scene that are relocated between the two states of the given scene. For this purpose, we learn a coupling matrix that indicates the learned feature correspondences between two volumes. We then decouple entities by clustering the volume units whose feature correspondences can be expressed consistently by a rigid body transformation between coordinate frames of the two feature volumes. In what follows, we detail the implementation of each step for this proposed pipeline.

Feature Volume Construction

To start with, the image encoder is implemented as an hourglass convolutional neural network, which has a strong resemblance to UNET (Ronneberger et al., 2015). It has three blocks in the downstream branch, where each block is composed of two convolutional layers. We employ Sigmoid Linear Unit (SiLU) (Elfwing et al., 2018) as the activation function after each convolutional layer, followed by layer normalization (Ba et al., 2016) adapted to convolutional outcomes. Maximum pooling operation downsamples the convolutional features at the end of each block, which is also connected to the upstream via skip connections. We found training a UNET-like image encoder from scratch performs better than using a pre-trained convolutional neural network as a feature extractor for our particular dataset.

The feature refinement operator $\psi(\cdot)$ is implemented as a Multi-Head Self-Attention(MHSA) (Vaswani et al., 2017) module, which, in a nutshell, applies scaled dot-product attention several times in parallel. We implement the attention module such that each channel in the sampled image features can be attended across all keyframes. We apply two consecutive MHSA layers with five heads, each followed by conventional residual connection and layer normalization.

The aggregation function $\zeta(\cdot)$ is implemented with attention-pooling (Yang et al., 2020). In other words, we compute a weighted sum of N_f feature vectors where weights are obtained from the features themselves with two MLPs with SiLU activation. In the last layer, a $softmax$ function normalizes the weights, to sum up to one. More formally, the feature at point $\mathbf{x} = (x, y, z)$ is computed by $\mathbf{V}(x, y, z) = \zeta\left(\left\{\bar{\xi}^j(\mathbf{x})\right\}_{j=1}^{N_f}\right) = \sum_{j=1}^{N_f} w_j^{\text{attn}}\, \bar{\xi}^j(\mathbf{x})$ where $\mathbf{w}^{\text{attn}} = softmax\left(F^{\text{attn}}\left(\bar{\xi}^j(\mathbf{x})\right)\right)$. We empirically found this strategy to perform better than widely used $mean$ and $variance$ operators as the aggregation function, whether used alone or when their results are concatenated.

We then process the constructed feature volume with a three-dimensional hourglass convolutional neural network, which again has a strong resemblance to 3D-UNET (Çiçek et al., 2016). The network consists of only one downsampling step, and each of the resultant three blocks is composed of only one three-dimensional convolutional layer. The activation function is again chosen as SiLU and is followed by layer normalization.

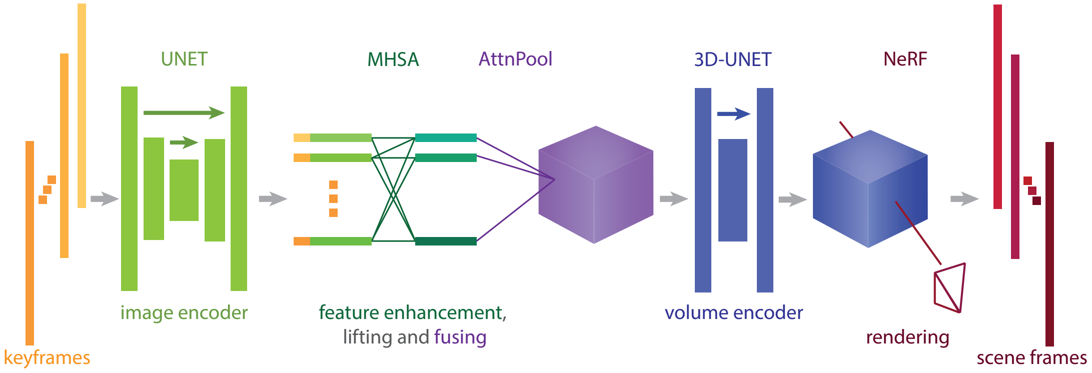


Figure 5.5: Implemented pipeline: we use a UNET-like image encoder for extracting features that are lifted to world coordinates, and refined by Multi-Head Self Attention (MHSA) layers. Feature fusing is achieved by Attention Pooling (AttnPool) resulting in the initial version of the feature volume. A 3D-UNET-like network post processes the feature volume, which is later used by rendering the frames using a small NeRF model.

Novel View Synthesis

For the NeRF networks, F_Θ^{coarse} and F_Θ^{fine} are implemented as MLPs with lower capacities compared to the original implementation. $F^1, F_\Theta^{\text{color}}$ and $F_\Theta^{\text{density}}$ that compose each F_Θ are implemented as 6-layer, 2-layer, and a single layer MLPs with 128, [64,3] and 1 output units, respectively. Similar to the original implementation, there is a skip layer in the network of $F^1(\cdot)$ after the third layer. The nonlinearity function is again chosen to be SiLU except in the last layers for color and density. The color output is produced following a *sigmoid* activation whereas the volume-density is obtained by *softplus* function, which is a smooth approximation to well-known rectified linear units. In contrast to 64 ray point samples for the coarse network and 128 additional samples for the fine network in the original implementation (Mildenhall et al., 2020), we sample 32 points for the coarse, and additional 32 points for the evaluation of the fine network, when not stated otherwise in the experiments section.

The overall pipeline for feature volume construction and conditional neural rendering can be seen in Figure 5.5. The model is trained with mean-squared reconstruction error for both coarse and fine networks, as expressed in Equation (5.23), where R denotes the set of rays sampled for an image. If not stated otherwise, we sample 4096 rays per image at each training step. Similarly, if not stated otherwise, the keyframes are not fixed for each scene. Indeed, we randomly sample keyframes from the training set at each training step after excluding the frames to be rendered. We then construct the feature volume for a single scene and single arrangement and render N_B images using the same feature volume before updating the parameters of the model. We use AdamW (Loshchilov and Hutter, 2017) optimizer with a weight decay of 1×10^{-5} and an initial learning rate of 5×10^{-4}, which is later halved twice after gradient steps 2000 and 5000.

$$\mathcal{L}_r = \frac{1}{|R|} \sum_{\mathbf{r} \in R} \left[\|C(\mathbf{r}) - \hat{C}_c(\mathbf{r})\|_2^2 + \|C(\mathbf{r}) - \hat{C}_f(\mathbf{r})\|_2^2 \right] \tag{5.23}$$

Optional Depth Supervision

In addition to the rendering loss presented in Equation (5.23), we consider optional depth-supervision because NeRF-based rendering is shown to struggle for inferring the accurate scene geometry (Wei et al., 2021; Oechsle et al., 2021; Yariv et al., 2021; Wang et al., 2021a) even when the synthesized images present high fidelity. In fact, in the original formulation, the inferred density is perturbed with Gaussian noise of unit variance before the last activation function in order to prevent floaters. Later, many follow-up work (Hedman et al., 2021; Yu et al., 2021a; Oechsle et al., 2021; Deng et al., 2022; Niemeyer et al., 2021; Kim et al., 2021) have proposed different auxiliary loss terms for density regularization, such as Cauchy (Hedman et al., 2021) or exponential (Yu et al., 2021a) penalty terms for promoting the sparsity of points with positive density values; or patch-based Total Variation (TV) regularization of inferred depth from unobserved viewwpoints (Niemeyer et al., 2021). Few other works proposed additional depth supervision either for a sparse set of points used for estimation of camera parameters (Deng et al., 2022), or dense depth maps (Johari et al., 2022). Hence, here, we present two other auxiliary loss terms which will complement the reconstruction loss in Equation (5.23) for some of our experiments. The first auxiliary loss is the exponential density regularization loss $\mathcal{L}_{\text{density}}$ in Equation (5.24) And the second one penalizes the mean-squared error of the inverse depth, as given in Equation (5.25), where $D(\mathbf{r})$ stands for the ground truth depth value corresponding to ray $\mathbf{r}$ and $\hat{D}_c(\mathbf{r})$ and $\hat{D}_f(\mathbf{r})$ stand for the rendered depth values via the coarse and fine NeRF networks, respectively. Equation (5.26) presents the depth rendering using the NeRF formulation.

$$\mathcal{L}_{\text{density}} = \frac{1}{|R|} \sum_{\mathbf{r} \in R, i \in \mathbf{r}} |1 - \exp(-0.5\sigma_i)| \tag{5.24}$$

$$\mathcal{L}_{\text{depth}} = \frac{1}{|R|} \sum_{\mathbf{r} \in R} \left[\left\| \frac{1}{D(\mathbf{r})} - \frac{1}{\hat{D}_c(\mathbf{r})} \right\|_2^2 + \left\| \frac{1}{D(\mathbf{r})} - \frac{1}{\hat{D}_f(\mathbf{r})} \right\|_2^2 \right] \tag{5.25}$$

$$\hat{D}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma\, (\mathbf{r}(t))\; dt \tag{5.26}$$

Feature Matching

For pruning the feature volumes, we use density thresholds $\tau^{(1)}, \tau^{(2)}$ which indicate the 10% of the maximum density value computed for corresponding feature volume. For matching features, we implement Sinkhorn iterations in the log domain to solve the minimization problem in Equation (5.17), where we use $\epsilon = 0.01$ for entropy regularization. We realize 5000

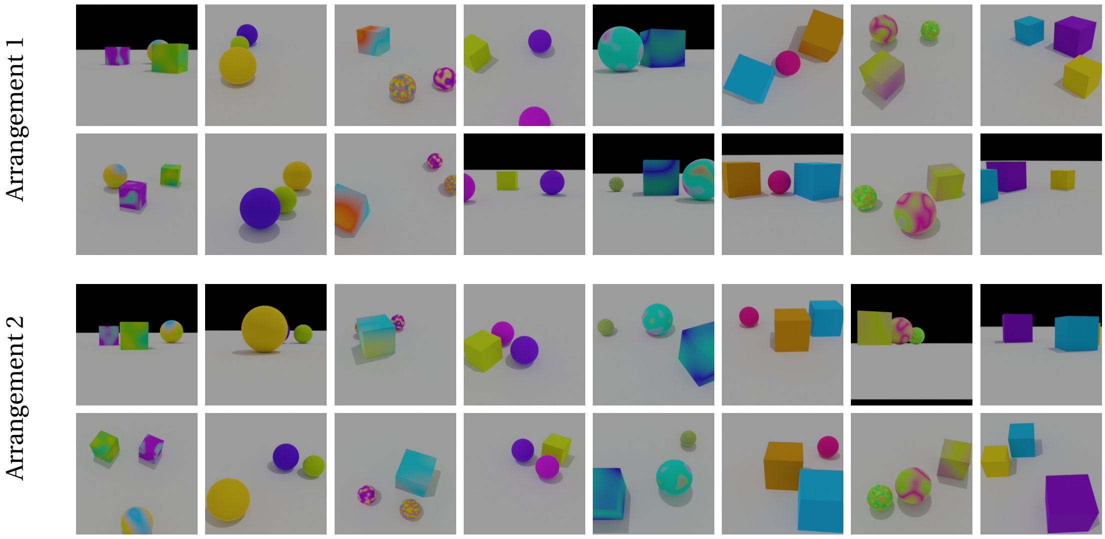


Figure 5.6: Eight scenes composed of simple shapes in two different arrangements. Each column represents a different scene, and each row within a specified arrangement group correspond to images from different viewpoints.

iterations, which often ensures convergence of the algorithm. We decide on the threshold for feature similarity, τ_F, based on the median feature norm of initial cluster $\mathbf{x}_{\{i\}}^{(1)}$. Finally, we demonstrate our results for $M = 5$ clusters for our dataset, which is composed of three relocated objects.

5.5 Experimental Results

5.5.1 Dataset

Since object abstraction via relocalization is not a conventional task yet, it is challenging to find a dataset tailored for such a problem formulation. Hence, we created a small-scale dataset to evaluate the performance of the proposed method. For this purpose, we used Kubric (Greff et al., 2022), a wrapper based on Blender, and we created a total number of twelve simple scenes each composed of three basic objects. The objects are chosen randomly from "cube" or "sphere" classes and located at randomly chosen locations on a plain gray floor. The scale and the color of each object instance are also chosen randomly. We use textured objects for half of the scenes and kept the others with a uniform color. We rendered 64 training, 32 validation, and 64 test images of size 256 x 256 pixels from random viewpoints in the upper hemisphere which are always directed at the scene center. We then repositioned the objects for the second arrangement and rendered a second set of images with randomly sampled viewpoints. Eight of the scenes can be seen in Figure 5.6, where they are grouped concerning object placements, which correspond to arrangements.

For evaluation of the rendering quality, we use widely accepted metrics Peak Signal-to-Noise Ratio (PSNR) and Structured Similarity Index (SSIM). Details of the implementation can be found in Appendix A. We evaluate the entity abstraction qualitatively, by visualizations of colored voxel centers. Note that the clustered voxels can also be used to render each entity individually.

5.5.2 Results

Ablation Studies

We first conduct an ablation study to determine the performance improvements due to the architectural components in the proposed implementation. The baseline model consists of the feature volume construction based on features sampled from image encoder outcomes, which are aggregated by attention pooling and NeRF-based rendering explained in Section 5.4. We then add the MHSA module that refines the sampled image features before pooling, and the final model additionally includes feature volume refinement by the 3D-UNET module. The training objective is set to be the mean squared reconstruction error. We train these three models for a single scene with two arrangements, and we train each model with a batch size of two. In practice, this setup implies that every two images use the same feature volume for rendering before the gradient update of the parameters. And every model needs to successfully represent two different scene arrangements. The keyframes are randomly chosen for each batch, excluding the images to be rendered. We use 16 keyframes for this ablation study and train all the models for 2500 epochs, which corresponds to nearly 80K update steps. We additionally train a NeRF (Mildenhall et al., 2020) model for each arrangement individually where the network is set to be equivalent to the rendering module of the pipeline, where the input is $\big(\gamma(\mathbf{x}),\gamma(\mathbf{d})\big)$ with $L=10$ for $\gamma(\mathbf{x})$ and with $L=4$ for $\gamma(\mathbf{d})$ (L denotes the number of sine and cosine functions for harmonic embeddings). We train both NeRF models, one for each arrangement, with the same training settings for the same number of iterations. Table 5.1 summarize our main findings, where the reconstruction quality is reported for each arrangement in parenthesis, and the overall performance for the scene is indicated under arrangement-wise outcomes. First of all, it can be clearly observed that both feature refinement modules contribute to the overall rendering performance. In particular, the volumetric refinement, which is possible due to our volumetric problem formulation, seems to boost the performance by a substantial amount. It can be also stated that our full pipeline performs better than the "tiny" NeRF models trained individually for each arrangement, under the same training configuration.

Next, we investigate the effect of density regularization or depth supervision with respect to the rendering quality of our proposed method. We opt for a training strategy similar to the previous ablation study, where we fix the number of keyframes for volume construction to 16 and use the full pipeline. By only changing the loss function across experiments for this ablation study, we observe variations in the rendering quality. We perturb the density with

Table 5.1: Ablation Study for Architectural Improvements. The numbers in parenthesis in the first of line of each row denote arrangement-specific reconstruction performance, while other numbers indicate average performance.

	PSNR	SSIM
baseline	(21.68) (22.27) 21.98	(0.72) (0.75) 0.74
baseline + MHA	(22.30) (23.49) 22.90	(0.75) (0.80) 0.78
baseline + MHA + 3D UNet	(**28.34**) (**31.41**) **29.87**	(**0.88**) (**0.90**) **0.90**
NeRF(Mildenhall et al., 2020)	(22.01) (21.79)	(0.84) (0.79)

Gaussian noise of variance 0.5, except when we use depth supervision with the auxiliary loss defined in Equation 5.25. We again train each model for a single scene for 2500 epochs, which correspond to nearly 80K update steps. Table 5.2 presents the outcomes, which imply that both density regularization and depth supervision indeed improve the rendering quality individually. Depth regularization, in particular, decreases the discrepancy between the rendering performance between arrangements, whereas depth supervision improves the performance of the second arrangement more significantly, which we attribute to the sequential processing of arrangements in batches for gradient updates. However, when applied together, depth supervision and density regularization fall short of their individual contributions, possibly due to the fact that uniform floor of the scenes in the dataset allows geometrically inaccurate voxel feature representations, which contradicts the depth loss.

Table 5.2: Ablation study for density regularization and depth supervision with corresponding loss functions. The numbers in parenthesis in the first of line of each row denote arrangement-specific reconstruction performance, while other numbers indicate average performance.

	PSNR	SSIM
$\mathscr{L}_r$	(25.53) (30.65) 28.09	(0.84) (0.92) 0.88
$\mathscr{L}_r + 0.0001\,\mathscr{L}_{\text{density}}$	(27.18) (30.47) 28.83	(0.86) (0.92) 0.89
$\mathscr{L}_r + 0.001\,\mathscr{L}_{\text{density}}$	(**28.34**) (31.41) **29.87**	(**0.88**) (0.90) **0.90**
$\mathscr{L}_r + 0.1\,\mathscr{L}_{\text{depth}}$	(27.62) (**31.52**) 29.57	(0.86) (**0.94**) **0.90**
$\mathscr{L}_r + 0.0001\,\mathscr{L}_{\text{density}} + 0.1\,\mathscr{L}_{\text{depth}}$	(26.99) (28.97) 27.98	(0.85) (0.90) 0.87

We further visualize the inferred scene structure by computing the R,G,B, and density values at discrete unit centers, *i.e.*, at voxel coordinates. The resolution of the volumetric grid, however, is not bounded by the dimension of feature volume used for training thanks to trilinear sampling used for computing the features along a ray shoot for rendering. We can upsample the feature volume in the same way, and compute R,G,B, and density values for the upsampled feature grid. Hence, we construct the feature volume using a subset of test images, 16 frames in this particular case and visualize grid colors if the density for the corresponding 3D point is higher than a threshold τ_σ. In Figure 5.7, we provide such visualizations for the models trained for this ablations study, hence, each row in Figure 5.7 corresponds to the same row in Table 5.2, and columns present two different arrangements of the same scene. We use $\tau_\sigma = 0.1$ for all models trained without depth supervision, and we use $\tau_\sigma = 0.01$ for the others. We observe that higher reconstruction quality does not always imply accurately recovered geometry by the method. For example, increasing the coefficient of the density regularization term, denoted by $\mathscr{L}_{\text{density}}$ increases the reconstruction quality as indicated by the corresponding (second and third) rows in Table 5.2. However, it lowers the volume density values of many voxels defining the scene floor such that we observe floating objects when we discard the voxel units associated with low volume density value, as illustrated in Figure 5.7.

Next, we investigate the effect of the number of keyframes for feature volume construction. We use 4, 8, and 16 keyframes for each experiment in this ablation study, where we use the full pipeline, and apply density regularization with a coefficient of 1×10^{-3}, without depth supervision. As the keyframes are chosen randomly at each training step from the training set after discarding the images that will be rendered, the total number of gradient updates remains the same for training these three separate models. Table 5.3 suggests that our model is capable of recovering the scene structure reasonably well even with a few keyframes. Choosing keyframes randomly at each training step ensures better generalization to novel viewpoints at test time, for both feature volume construction and rendering.

Table 5.3: Ablation Study for Number of Keyframes used in Feature Volume Construction. The numbers in parenthesis in the first of line of each row denote arrangement-specific reconstruction performance, while other numbers indicate average performance.

	PSNR		SSIM	
4-views	(27.44)	(30.00)	(0.85)	(0.91)
	28.72		0.88	
8-views	(**29.78**)	(31.35)	(**0.90**)	(**0.93**)
	30.57		**0.91**	
16-views	(28.34)	(**31.41**)	(0.88)	(0.90)
	29.87		0.90	

Multi-scene, Multi-Arrangement Training

After investigating the contribution of different implemented blocks in the proposed method, auxiliary losses, and the number of keyframes used for feature volume construction, we continue with multi-scene, multi-arrangement training. In other words, we would like to use the same volumetric feature construction and feature-based rendering functions to model multiple scenes of similar characteristics. This can be considered as the extension of the

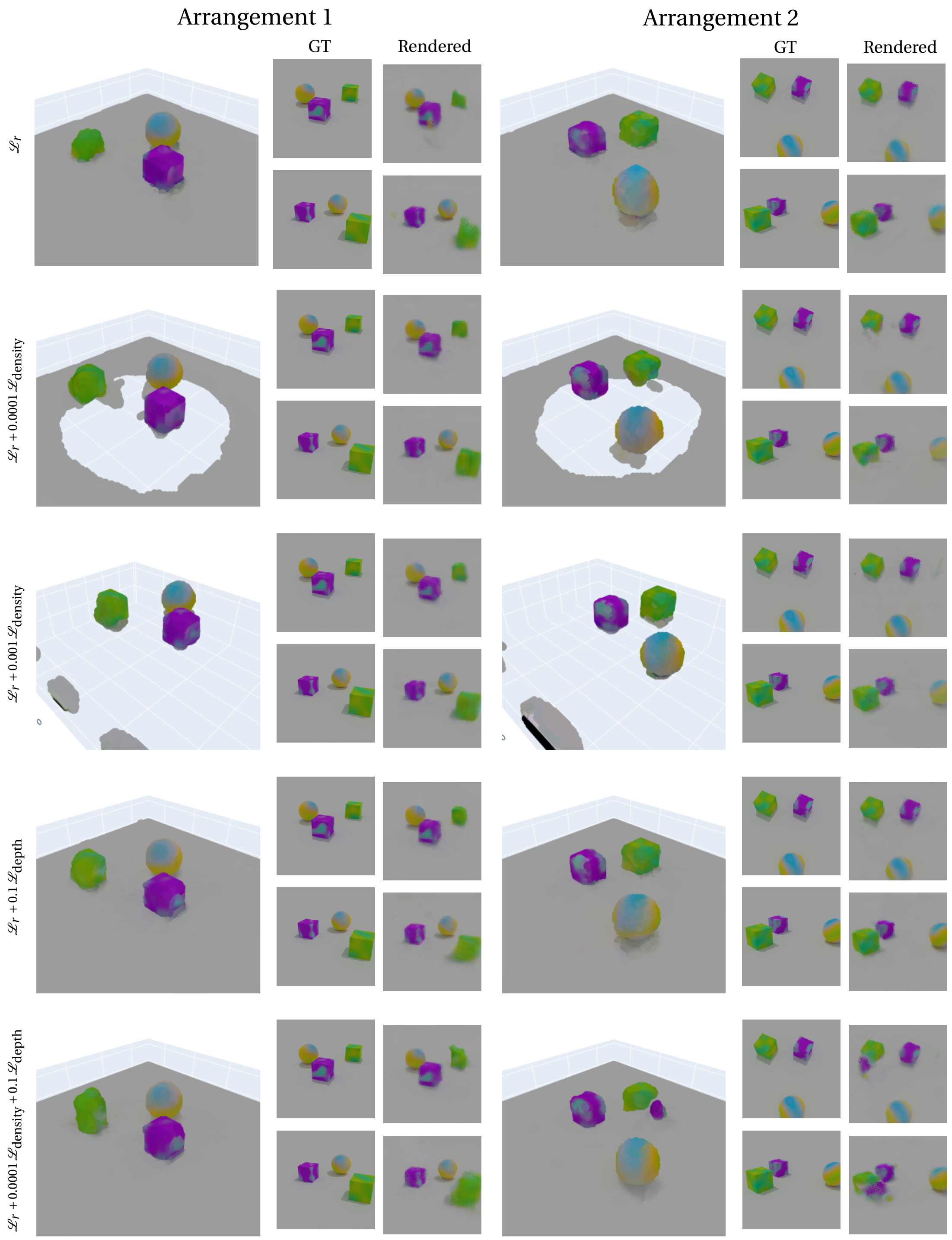

Arrangement 1
GT
Rendered
Arrangement 2
GT
Rendered
$\mathcal{L}_r$
$\mathcal{L}_r + 0.0001\,\mathcal{L}_{\text{density}}$
$\mathcal{L}_r + 0.001\,\mathcal{L}_{\text{density}}$
$\mathcal{L}_r + 0.1\,\mathcal{L}_{\text{depth}}$
$\mathcal{L}_r + 0.0001\,\mathcal{L}_{\text{density}} + 0.1\,\mathcal{L}_{\text{depth}}$

Figure 5.7: Qualitative results for the ablation study that investigates the effect of density regularization and depth supervision. We visualize the colored voxel center points in 3D after upsampling by a factor of two andthresholding the density. Next to each 3D visualization, we compare the rendering outputs (right) to ground truth(left) for two different test views, presented as two rows. Qualitative results can be found in Table 5.2.

Table 5.4: Rendering performance of the proposed model on the test set of scenes used for training. Values in parentheses indicate arrangement specific performance.

		Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Scene 7	Scene 8	Mean
Greedy Training	**PSNR**	(26.82) (27.66) 27.24	(25.06) (25.61) 25.34	(28.29) (26.64) 27.47	(25.57) (28.74) 27.16	(27.66) (27.21) 27.43	(27.82) (26.67) 27.24	(26.91) (29.40) 28.16	(21.22) (28.34) 24.78	**26.85**
	SSIM	(0.87) (0.89) 0.88	(0.84) (0.88) 0.86	(0.92) (0.85) 0.88	(0.87) (0.91) 0.89	(0.90) (0.88) 0.89	(0.91) (0.93) 0.88	(0.88) (0.93) 0.90	(0.84) (0.94) 0.89	**0.88**
Batched Training	**PSNR**	(28.12) (30.09) 29.10	(28.86) (29.20) 29.03	(29.66) (28.63) 29.14	(29.23) (31.53) 30.38	(29.16) (29.90) 29.53	(29.92) (27.35) 28.63	(27.95) (30.62) 29.28	(25.88) (28.84) 27.36	**29.06**
	SSIM	(0.86) (0.92) 0.89	(0.94) (0.94) 0.94	(0.93) (0.91) 0.92	(0.93) (0.95) 0.94	(0.93) (0.94) 0.93	(0.96) (0.89) 0.93	(0.90) (0.94) 0.92	(0.91) (0.95) 0.93	**0.92**

previous model, which is trained in an arrangement-agnostic way, such that a single model can represent both arrangements of a given scene. For this purpose, we use the dataset introduced in Section 5.5.1.

In our first experiment, we use the full pipeline presented in Section 5.4 with 16 keyframes for feature volume construction. For training, in addition to the mean-squared reconstruction error $\mathscr{L}_r$, we employ the auxiliary depth supervision introduced in Equation (5.25). Although the model trained with depth supervision does not achieve the best quantitative performance in the related ablation study, it infers the scene geometry better, as depicted in Figure 5.7, which is a key element for our entity abstraction algorithm. We again follow the greedy training approach used for previous experiments: at each training step, we randomly select two frames from the training set of a specific scene, and a specific arrangement. We then randomly sample 16 keyframes from the same dataset after excluding the two frames to be rendered. In this setting, we consider these two images as the "batch", and average the training objective over this batch for performing the gradient update on model parameters. We continue with the same scene and same arrangement until all 64 training images are rendered. We then switch to the other arrangement of the same scene and later continue with another scene until we finish processing all eight scenes in the training set. We train the model for 2500 epochs, which corresponds to 640K gradient updates with a batch size of two, with distributed training on two GPUs. At inference time, we also randomly sample the keyframes from the test set for every batch of two test frames to be rendered. Results under "Greedy Training" in Table 5.4 summarize the reconstruction quality of the rendered images from the 64 new viewpoints in the test set. We can observe that the method performs well across different scenes and different arrangements.

We later test the model on 4 novel scenes, which are not used during training. Despite the scene-agnostic formulation, we found out that the model cannot generalize well to novel scenes as presented in Table 5.5 The same phenomenon can be also observed in concurrent volumetric feature representations based on neural rendering (Lazova et al., 2022).

To investigate whether a different training strategy can help with generalization to novel scenes, we train another model in slightly different settings instead of the sequential processing of each scene and each arrangement. For this second attempt, we construct two feature volumes

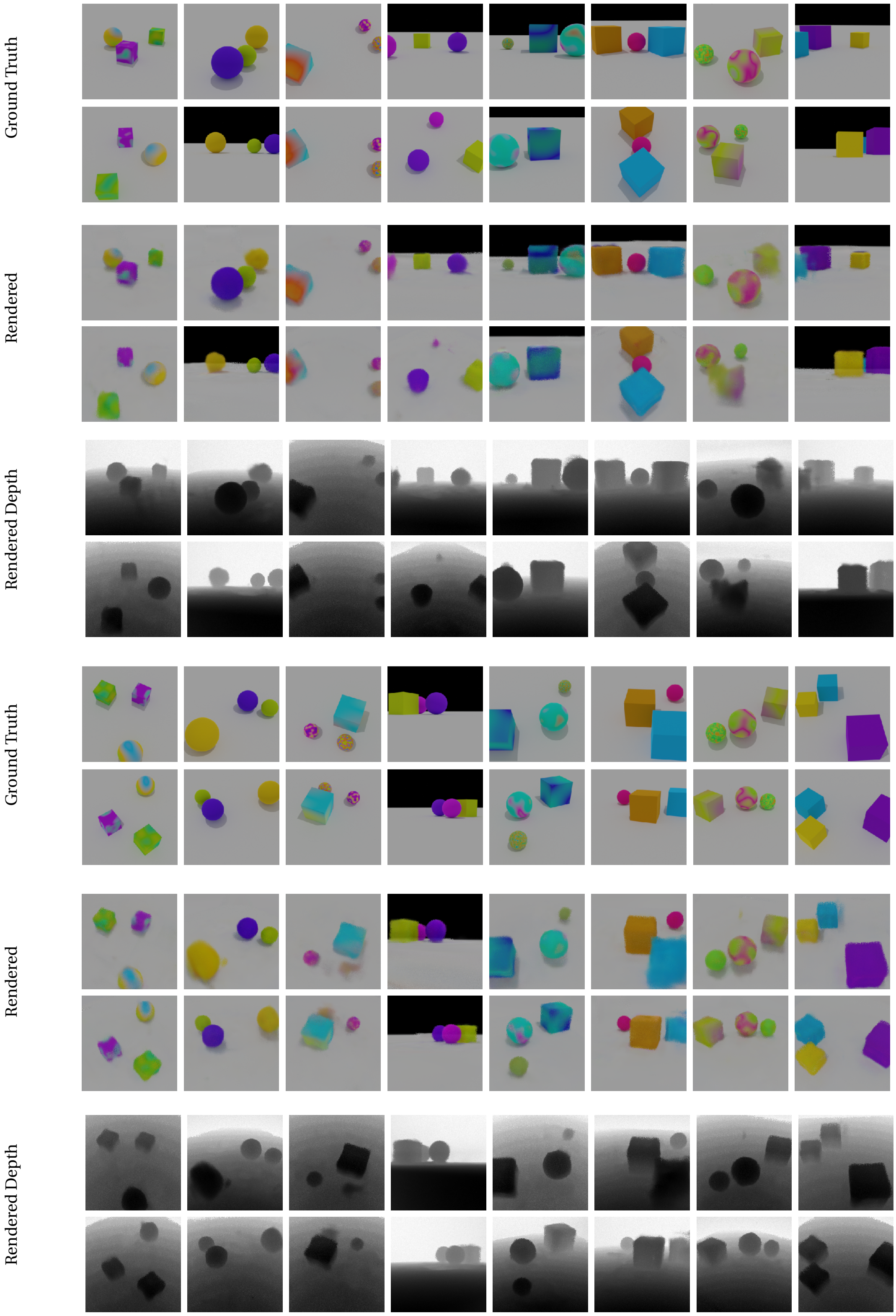


Figure 5.8: Rendering outputs for eight scenes composed of simple shapes in two different arrangements, which are introduced in Figure 5.6. Each column represents a different scene, and each row within a specified arrangement group corresponds to an image from a different viewpoint.

Table 5.5: Rendering performance of the proposed model on the test set of novel scenes, which are used for training. Values in parentheses indicate arrangement specific performance.

		Scene 9	Scene 10	Scene 11	Scene 12	Mean
Greedy Training	**PSNR**	(19.63) (19.81) 19.72	(16.46) (16.35) 16.40	(18.26) (17.83) 18.04	(15.99) (17.65) 16.82	**17.75**
	SSIM	(0.50) (0.52) 0.52	(0.45) (0.44) 0.44	(0.60) (0.47) 0.54	(0.34) (0.54) 0.45	**0.49**
Batched Training	**PSNR**	(20.21) (20.12) 20.17	(16.59) (16.21) 16.40	(18.12) (17.52) 17.85	(16.72) (16.77) 16.75	**17.80**
	SSIM	(0.52) (0.56) 0.54	(0.50) (0.44) 0.46	(0.58) (0.45) 0.51	(0.46) (0.48) 0.47	**0.50**

concurrently, which do not necessarily represent the same scene. For each feature volume, we render 4 different training images from the corresponding training set. We average the training loss over these 8 images. For this setting, we use 8 keyframes for feature volume construction, and similar to the first training, we employ depth supervision. Results under "Batched Training" in Table 5.4 present the improved performance with the second training strategy. Furthermore, the same number of 2500 epochs corresponds to 160K training steps in this setting with the same distributed training on two GPUs. Hence, the batched approach leads to better performance after substantially shorter training time, at the expense of a slightly increased memory footprint. We believe that averaging gradients over different scenes or arrangements prevent overfitting for feature extraction and refinement modules. However, the performance improvement does not reflect the generalization scenario as can be seen in Table 5.5.

To complete, in Figure 5.8, we present two rendered images for randomly selected test viewpoints, and associated rendered depth maps, for both arrangements of the eight training scenes. We can observe that the scene geometry is captured reasonably well for almost all scenes and arrangements.

Entity Abstraction

In this section, we provide a qualitative evaluation of the proposed clustering-based entity abstraction.

We here present the clustering of voxels based on feature consistency. For illustrative purposes, we plot both arrangements in the same canonical coordinates by adding an offset value to the coordinates of one of the arrangements. Consequently, we adjust any relations depicted on the center coordinates of voxels. In the first step of Figure 5.9, we can observe correspondences between two feature volumes for voxels whose associated density value is higher than a threshold. The following steps depict the last stage of the proposed abstraction algorithm for each cluster, where the center coordinates of the (remaining) voxels in the first arrangement are mapped to the second one by the estimated rigid body transformation $[\mathbf{R}_j|\mathbf{t}_j]$.

Our results illustrate the ability of the proposed method for decoupling entities in an unsuper-

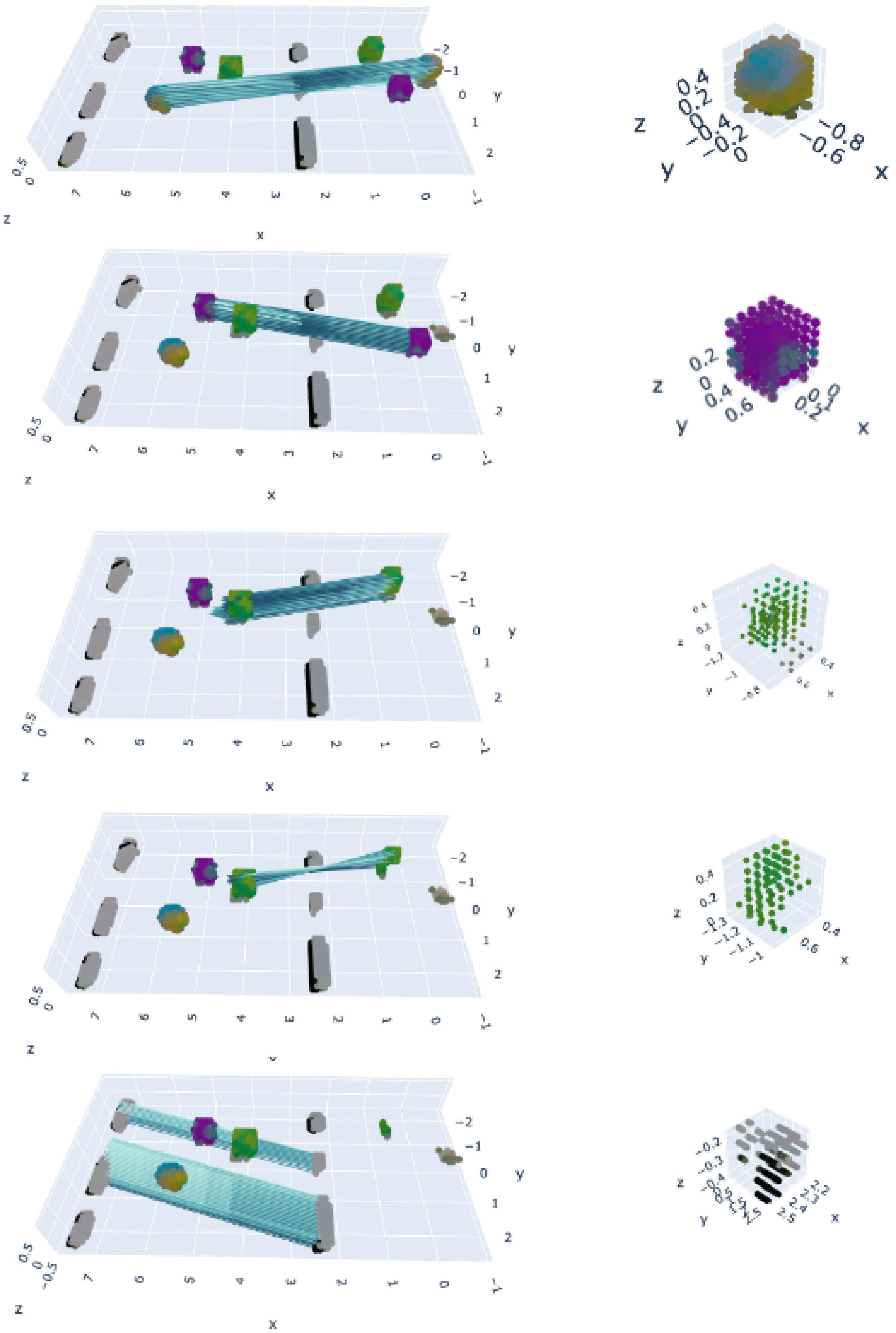


Figure 5.9: Clustered correspondences between two feature volumes using the proposed greedy algorithm. Figures on the left depict the correspondences that can be modeled by the same rigid body transformation. The resultant first five clusters of voxel units are illustrated on the right.

vised way.

5.6 Conclusion

In this chapter, we proposed a novel method for unsupervised entity abstraction in 3D scenes based on two different arrangements observed from multiple viewpoints. For this purpose, we represent the scene with a dense feature volume, which is constructed from 2D image features based on the principles of 3D geometry. The image features are learned jointly with a rendering function that allows projecting the 3D information of the scene to any query viewpoint with the help of recently proposed neural rendering algorithms. We provide a baseline implementation for the fundamental problem of lifting and fusing 2D information and improve upon this baseline by incorporating modern deep learning architectures for enhancing both the 2D and the 3D features. We provide an extensive ablation study to examine the contributions of different factors in the proposed method to the rendering quality. Experimental results show that our model is capable of describing multiple 3D scenes in different states with a hybrid representation of lifted feature volumes and a rendering function, as long as the model is exposed to the scene during training. We achieve superior rendering quality compared to representing 3D scenes only by a rendering function.

The proposed representation allows us to decouple entities in the scene without any supervision provided that they are relocated between the two observed states of the scene (the static scene can be represented by a single rigid body transformation that aligns the canonical coordinates of two feature volumes, hence, disambiguating entities with the proposed approach would become an ill-posed problem). In order to disambiguate entities, we work with feature volumes of the two different states of the scene after pruning them individually based on associated volume density. We then establish correspondences between two feature volumes with a stochastic coupling matrix. By searching for volume clusters that can be represented by the same rigid body transformation between two states of the scene, we obtain entity representations that can be rendered from any query viewpoint. Experimental results show that our novel 3D scene representation and entity abstraction method is a promising step towards understanding scene structure directly in three dimensions and without any supervision.

6 Conclusion

6.1 Summary

In this thesis, we explored different methods for disambiguating objects, or visual entities in video sequences or 3D scenes, in an unsupervised manner. Objects play an important role for holistic understanding of scenes, hence, for prediction and planning. Most of the existing object-related problems are defined within a supervised framework, where the algorithms are guided by different forms of manual annotations. These supervised methods perform impressively well within their training distribution, while mostly failing to parse novel scenes into their composing entities. Compositional methods, on the other hand, offer better generalization properties. Thus, as a part of the recent trend for unsupervised decomposition algorithms, we take a step back and investigate possible ways of representing scenes in both two- and three-dimensions in an object-centric manner.

For this purpose, we use motion clues to disambiguate entities, inspired by the findings in neuroscience which tell us that the infants perceive the world based on things "that move". It indeed is a reasonable assumption for unsupervised decomposition as the problem becomes heavily ill-defined in static settings. Hence, we start with decomposing video frames based on moving objects that are modelled by consistent approximated motion between different time steps. Related methods in the literature mostly opt for latent based representations in an auto-encoding manner. In other words, they decompose an input frame into a set of latent variables, which are tracked throughout the video sequence, and the latent variables are decoded back to image space to compose the given video frame at a given time. We claim that the performance of these methods are bounded heavily by their decoding quality. In addition, the extension of these methods to other tasks like prediction or planning suffer from averaging effects that emerge from the reconstruction objective applied over the sequence. Hence, we rather sketch the problem in a prediction framework, and represent moving objects by amodal masks that model both visible and occluded parts of objects.

Object-centric prediction is a challenging problem, which requires understanding of the compositional structure and proper prediction of each constituent entity in the following time steps. In Chapter 3, we propose a novel method for decoupling moving objects by a combination of single- and multi- step prediction objectives. We obtain amodal object masks for all moving entities, infer visible parts of objects at a given time step, and perform inpainting for the occluded regions for a holistic representation. We then predict parameters for geometric transformations that can describe the motion of each entity between different time instants. By warping each entity to a different time instant based on its predicted motion approximation, and predicting a composition mask for the corresponding images, we synthesize predicted frames. In order to support our holistic representation, we define a cyclic multi-step prediction objective in addition to the frequently used next-frame counterpart. Experimental results on illustrative datasets show that our novel representation is capable of not only decomposing the scene into moving entities and a static background, but it can also predict the frame without any manual annotations or computationally expensive iterative approaches.

Next, we extend the proposed framework for stochastic video prediction. Prediction is inherently a stochastic problem as there are often multiple possible continuations of a given sequence. In Chapter 4, we attribute the stochasticity of the problem to the motion of composing entities in a video frame. For this purpose, we convert the parameters of the geometric transformations used in Chapter 3 into random variables and propose two methods for approximating the related posterior distribution. This distribution defines how the motion parameters should be sampled for the prediction of the following frames. We sketch our first method under the framework of Variational Autoencoders (Kingma and Welling, 2013), which leads to a graphical model similar to state space models. It guides us to a method for approximating the posterior distribution by the help of an auxiliary state ,which is computed backwards in time based on inferred object masks and the input sequence. For the second approach, we follow an adversarial training scheme formulated under a Wasserstein GAN (Gulrajani et al., 2017) framework. In contrary to the first proposal, this method does not require traversal of the input video sequence multiple times, yet, it involves the design of a critic function, which plays a key role for the stability of training, as well as the quality of the approximated distribution. Our illustrative experimental results demonstrate that the proposed object-centric stochastic prediction model can circumvent the well-known issue of blurry outcomes for short term video prediction while representing data distribution reasonably well.

Finally, we propose to use motion clues between two time steps to represent entities in 3D by the help of multiple observations of the scene at each time step, called states or arrangements of the scene. We opt for feature volumes to represent each state of the scene, and decompose entities by clustering voxels based on their feature similarity across volumes. In other words, we represent entities by clusters of feature volumes that can be modeled by a rigid body transformation while retaining feature consistency. In order to construct our volumetric representation, we extract features from observed images and lift them to world coordinates based on the camera parameters associated with observations. We learn the feature construction jointly with a rendering function, which maps any point in world coordinates to color an volume density values based on the associated feature vector. Hence, the scene can be rendered from any viewpoint by using ray-marching algorithms. Our results show that the proposed method can represent multiple scenes successfully and achieve better rendering performance compared to instances of Neural Radiance Fields (Mildenhall et al., 2020) with the same capacity. The resultant feature volume, hence, provides a viable scene representation solution for unsupervised entity abstraction.

6.2 Future Directions

While novel representations proposed in this thesis proved to be promising alternatives to the other approaches in a recently blooming research problem of unsupervised object-centric representations, they can be definitely improved regarding their representation capacity. Namely, video object representations can be improved upon pre-training some of its sub-blocks. The modular implementation indeed provides room for improvements. For example, learning inpainting functions a-priori would be possible, again in a self-supervised manner, by a denoising approach. Moreover, approximating the motion by simple geometric transformations limit the application of the method to a subset of motion observed in real world sequences because it cannot model in-plane rotations or deformable motion. This limitation can be overcome by decomposing the motion into geometrical approximation and sub-pixel deformable motion, where the first aligns the objects in time in the best possible way and the latter corrects the inaccuracies arising from the initial approximation. Finally, the static background assumption is another limitation that prevents applying the proposed method to a variety of datasets. We consider the extension of our method to more realistic motion patterns as an interesting future direction.

Any improvement upon our video decomposition model would inherently benefit to the associated stochastic video prediction framework. The latter can be further improved by modelling the motion distribution in a way that is more representative than isotropic Gaussian. Although it helps with computations, the independence that is assumed between transformation parameters might not be always satisfied. For example, the translation in x- and y-directions are often correlated in real world data. One such improvement can be obtained by incorporating flow-based models (Rezende and Mohamed, 2015), which map simple distributions to arbitrarily complex ones in an invertible manner, for modelling the distribution of motion parameters. In addition, approximation of the posterior distribution can be further improved by better designed implementations.

Finally, our novel method for representation of scenes presented in Chapter 5 is also an interesting avenue for future work. To start with, our probabilistic approach to correspondence problem can be incorporated into feature volume construction, as the Sinkhorn operations that we use for the computation of the coupling matrix is fully differentiable. During training, when two feature volumes of the same scene are constructed, we can transfer features between two volumes to ensure feature consistency for the volume units that represent the same objects in two different arrangements. Moreover, the method suffers from memory bottleneck as the number of feature units grow cubically with increasing spatial resolution. It is possible to circumvent the memory bottlenecks by gradual pruning of the feature volume based on the inferred volume density. It also enables increasing the spatial resolution accordingly, as implemented for scene-specific voxel representations (Liu et al., 2020). Another interesting extension would be applying the Kalman-based morphing of unbounded 3D scene to a constrained volume (Barron et al., 2022), which would extend applicability of our method to any given scene. Lastly, the greedy entity abstraction method based on correspondences can be improved by pre-processing, such as initial plane extraction, to constrain the problem more towards the entities by decoupling them from "background". It can be further refined by iterative estimation of the rigid body transformation parameters to alleviate the harm caused by outliers, in a similar manner to RANSAC algorithm (Fischler and Bolles, 1981) which has been successfully used for many years.

The progress is more rapid than ever in computer vision as a result of the advancements in machine learning, in particular, deep learning algorithms. Resultant scalable solutions recently attracted an unprecedented amount of attention from different communities thanks to the large computational resources allocated for related lines of research. However, it is not very likely, nor is it sustainable, to maintain the current rate of progress by just increasing the scale of these methods. The content is constantly changing, we need better generalization to make reliable predictions under changing distributions. This generalization can partly come from compositional methods. And it can be further improved by multi-modal learning based on different data modalities, such as audio, visual and language. Novel representation methods proposed in this thesis are currently not capable of modelling real world data, yet, they aim to provide a different perspective for compositional approaches towards the ultimate objective of better data representation and generalization.

A Evaluation Metrics

A.1 Image Reconstruction

Given an input image $\mathbf{x}$ of shape $H \times W \times 3$ and rendered image $\hat{\mathbf{x}}$, Peak Signal-to-Noise Ratio (PSNR) as computed according to Equation (A.1), where $\mathbf{x}[i,j]$ denotes the pixel value at pixel coordinates (i,j).

$$PSNR = 20\log_{10}\frac{\max(\mathbf{x})}{\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\left(\mathbf{x}[i,j]-\hat{\mathbf{x}}[i,j]\right)^2} \tag{A.1}$$

On the other hand, Structured Similarity Index (SSIM) is computed as given in Equation (A.2) for two image windows of the same size with average pixel values μ_x, μ_y, variances σ_x^2, σ_y^2 and covariance σ_{xy}. The coefficients c_1 and c_2 in Equation (A.2) are computed according to the dynamic pixel value range L with $c_1 = (0.01L)^2$ and $c_2 = (0.03L)^2$.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{A.2}$$

A.2 Object Abstraction

Adjusted Rand Index(ARI) (Rand, 1971) is a measure for clustering similarity and is commonly used by unsupervised abstraction methods (Greff et al., 2019) such that the segmentation masks are treated as cluster assignments. By discarding the background, some methods adopt it for only for segmented objects, and refer to it as Foreground Adjusted Rand Index (FG-ARI).

Another metric often used for segmentation tasks is Intersection over Union (IoU) for resultants and groundtruth masks. Weis et al. (2020) suggest treating the segmentation as a match if the computed IoU for a pair of masks is greater than 0.5.

B Derivation of Lower Bound for Stochastic Video Prediction Model

The joint problem of object mask inference, parameter estimation for presumed planar motion and next frame synthesis, given initial object masks $\mathbf{m}_0$ and initial transformation parameters $\mathbf{z}_0$, can be formulated as a maximum likelihood problem. We assume N independently observed sequences of length T, $\mathbf{x}^{(i)}_{1:T_i}$ for $i = 1, \ldots, N$ for training to maximize the likelihood given in Equation (B.3).

$$\begin{aligned}
\mathcal{L}(\theta) &= \log p_\theta(\{\widehat{\mathbf{x}}^i_{2:T}, \mathbf{m}^i_{1:T-1}, \mathbf{z}^i_{1:T-1}\} | \{\mathbf{x}^i_{1:T}, \mathbf{m}^{(i)}_0, \mathbf{z}^{(i)}_0\}_{i=1}^N) && \text{(B.1)}\\
&= \sum_{i=1}^N \log p_\theta(\widehat{\mathbf{x}}^i_{2:T}, \mathbf{m}^i_{1:T-1}, \mathbf{z}^i_{1:T-1} | \mathbf{x}^i_{1:T}, \mathbf{m}^{(i)}_0, \mathbf{z}^{(i)}_0) && \text{(B.2)}\\
&= \sum_{i=1}^N \mathcal{L}^{(i)}(\theta) && \text{(B.3)}
\end{aligned}$$

We factorize the individual terms $\mathcal{L}^{(i)}(\theta)$ over time as follows:

$$p_\theta(\widehat{\mathbf{x}}_{2:T}, \mathbf{m}_{1:T-1}, \mathbf{z}_{1:T-1} | \mathbf{x}^i_{1:T}, \mathbf{m}_0, \mathbf{z}_0) = \prod_{t=1}^{T-1} p_{\theta_1}(\widehat{\mathbf{x}}_{t+1} | \mathbf{m}_t, \mathbf{z}_t, \mathbf{x}_t)\ p_{\theta_2}(\mathbf{m}_t | \mathbf{m}_{t-1}, \mathbf{x}_t)\ p_{\theta_3}(\mathbf{z}_t | \mathbf{x}_t, \mathbf{m}_t, \mathbf{m}_{t-1}, \mathbf{z}_{t-1}) \tag{B.4}$$

Furthermore, for the inference of transformation parameters at time t, we assume full access to the object masks, $\mathbf{m}_{1:T-1}$ as well as the training sequence itself, $\mathbf{x}_{1:T}$, which corresponds to both the input and the ground truth for the output. Hence, once *d-separation* is applied to the corresponding graphical model presented in Figure 4.2, the parameter estimation term in Equation (B.4) can be factorized as in Equation (B.5).

$$p_{\theta_3}(\mathbf{z}_{1:T-1} | \mathbf{m}_{1:T-1}, \mathbf{x}_{1:T}) = \prod_{t=1}^{T-1} p_{\theta_3}(\mathbf{z}_t | \mathbf{x}_{t:T}, \mathbf{m}_{t:T-1}, \mathbf{z}_{t-1}) \tag{B.5}$$

As explained in Section 4, the maximization of the posterior distribution is not possible due to its intractable nature, so, we approximate the posterior distribution via an inference network.

The approximate posterior, which uses the same mask inference step and follows a similar factorization over training sequences and over time is given in Equation B.6.

$$q_\phi(\mathbf{m}_{1:T-1},\mathbf{z}_{1:T-1}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T-1} p_{\theta_2}(\mathbf{m}_t|\mathbf{m}_{t-1},\mathbf{x}_t)\ q_\phi(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_t = g_{\phi_a}(\mathbf{a}_{t+1},[\mathbf{x}_t,\mathbf{m}_t,\mathbf{m}_{t+1}])) \tag{B.6}$$

Hence, the lower bound for the ith training sequence $\mathbf{x}^{(i)}_{1:T}$, can be obtained as in Equation (B.7). Note that, $\underline{\mathbf{m}}_t$ that is used as an input to the backward recursive function $g_\phi(\cdot)$ in Equation (B.8) represents the inferred masks as the outcome of the deterministic step corresponding to $p_{\theta_2}(\mathbf{m}_t|\mathbf{m}_{t-1},\mathbf{x}_t) = \delta(\mathbf{m}_t - \underline{\mathbf{m}}_t)$ and it is a result of integration over the masks at later steps.

$$\mathscr{F}^{(i)}(\theta,\phi) = \iint q_\phi(\mathbf{m}_{1:T-1},\mathbf{z}_{1:T-1}|\mathbf{x}_{1:T}) \log \frac{p_\theta(\mathbf{x}_{2:T},\mathbf{m}_{1:T-1},\mathbf{z}_{1:T-1})}{q_\phi(\mathbf{m}_{1:T-1},\mathbf{z}_{1:T-1}|\mathbf{x}_{1:T})}\, d\mathbf{z}_{1:T-1}\, d\mathbf{m}_{1:T-1} \tag{B.7}$$

$$\begin{aligned} &= \iint \sum_{t=1}^{T-1} q_\phi(\mathbf{m}_{1:T-1},\mathbf{z}_{1:T-1}|\mathbf{x}_{1:T}) \\ &\quad * \log \frac{p_{\theta_1}(\widehat{\mathbf{x}}_{t+1}|\mathbf{m}_t,\mathbf{z}_t,\mathbf{x}_t)\ p_{\theta_2}(\mathbf{m}_t|\mathbf{m}_{t-1},\mathbf{x}_t)\ p_{\theta_3}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{m}_t,\mathbf{x}_t)}{q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_t = g_{\phi_a}(\mathbf{a}_{t+1},[\mathbf{x}_t,\underline{\mathbf{m}}_t],\underline{\mathbf{m}}_{t+1}]))\, p_{\theta_2}(\mathbf{m}_t|\mathbf{m}_{t-1},\mathbf{x}_t)}\, d\mathbf{z}_{1:T-1}\, d\mathbf{m}_{1:T-1} \end{aligned} \tag{B.8}$$

$$\begin{aligned} &= \iint \sum_{t=1}^{T-1} q_\phi(\mathbf{m}_{1:T-1},\mathbf{z}_{1:T-1}|\mathbf{x}_{1:T}) \log p_{\theta_1}(\widehat{\mathbf{x}}_{t+1}|\mathbf{m}_t,\mathbf{z}_t,\mathbf{x}_t)\, d\mathbf{z}_{1:T-1}\, d\mathbf{m}_{1:T-1} \\ &\quad + \iint \sum_{t=1}^{T-1} q_\phi(\mathbf{m}_{1:T-1},\mathbf{z}_{1:T-1}|\mathbf{x}_{1:T}) \frac{p_{\theta_3}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{m}_t,\mathbf{x}_t)}{q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_t)}\, d\mathbf{z}_{1:T-1}\, d\mathbf{m}_{1:T-1} \end{aligned} \tag{B.9}$$

$$\begin{aligned} &= \sum_{t=1}^{T-1} \Bigg[\iint q_\phi(\mathbf{m}_{1:T-1},\mathbf{z}_{1:T-1}|\mathbf{x}_{1:T}) \log p_{\theta_1}(\widehat{\mathbf{x}}_{t+1}|\mathbf{m}_t,\mathbf{z}_t,\mathbf{x}_t)\, d\mathbf{z}_{1:T-1}\, d\mathbf{m}_{1:T-1} \\ &\quad + \iint q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_t = g_{\phi_a}(\mathbf{a}_{t+1},[\mathbf{x}_t,\underline{\mathbf{m}}_t,\underline{\mathbf{m}}_{t+1}]) \log \frac{p_{\theta_3}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{m}_t,\mathbf{x}_t)}{q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{a}_t)}\, d\mathbf{z}_{1:T-1}\, d\mathbf{m}_{1:T-1} \Bigg] \end{aligned} \tag{B.10}$$

$$= \sum_{t=1}^{T-1} \mathbb{E}_{q_{\phi_2}} \Big[\mathbb{E}_{q_{\phi_1}} \big[\log\big(p_{\theta_1}(\widehat{\mathbf{x}}_{t+1}|\mathbf{x}_t,\underline{\mathbf{m}}_t,\mathbf{z}_t)\big)\big] - KL\big(q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T},\underline{\mathbf{m}}_{t:T}) || p_{\theta_3}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_t,\underline{\mathbf{m}}_t)\big) \Big] \tag{B.11}$$

In Equation (B.11), the expectation over distribution q_{ϕ_1} corresponds to expectation over $q_{\phi_1}(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x}_{t:T},\underline{\mathbf{m}}_{t:T})$, while q_{ϕ_2} stands for the marginal distribution of $\mathbf{z}_{t-1}$ in the variational approximation to the posterior $q_\phi(\mathbf{z}_{1:t}|\mathbf{x}_{t:T-1},\underline{\mathbf{m}}_{t:T},\mathbf{z}_0)$. Following [14], we approximate the expectation via Monte Carlo estimate, using the sequential formulation in Equation (B.6).

Thus, as the training objective, we maximize $\mathscr{F}^{(i)}(\theta,\phi)$ over all available training sequences with respect to trainable parameters θ and ϕ. The first term in Equation (B.11) is implemented as the reconstruction error when the masks inferred as a result of the deterministic state and that they are processed with the parameters sampled from approximate posterior $q_{\phi_1}(\cdot)$ in a sequential manner satisfying Equation (B.6).

Bibliography

[1] Aguiar, A. and Baillargeon, R. (1999). 2.5-month-old infants' reasoning about when objects should and should not be occluded. *Cognitive psychology*, 39(2):116–157.

[2] Ahmadyan, A., Zhang, L., Ablavatski, A., Wei, J., and Grundmann, M. (2021). Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[3] Alexe, B., Deselaers, T., and Ferrari, V. (2010). What is an object? In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 73–80. IEEE.

[4] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

[5] Armeni, I., Sax, S., Zamir, A. R., and Savarese, S. (2017). Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*.

[6] Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., and Mouzakitis, A. (2019). A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795.

[7] Azizpour, H. and Laptev, I. (2012). Object detection using strongly-supervised deformable part models. In *European Conference on Computer Vision*, pages 836–849. Springer.

[8] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[9] Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R. H., and Levine, S. (2017). Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*.

[10] Bapst, V., Sanchez-Gonzalez, A., Doersch, C., Stachenfeld, K., Kohli, P., Battaglia, P., and Hamrick, J. (2019). Structured agents for physical construction. In *International Conference on Machine Learning*, pages 464–474. PMLR.

[11] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479.

[12] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

[13] Bayer, J. and Osendorfer, C. (2014). Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*.

[14] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. (2019). Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307.

[15] Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. (2016). Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer.

[16] Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1365–1372. IEEE.

[17] Branson, S., Perona, P., and Belongie, S. (2011). Strong supervision from weak annotation: Interactive training of deformable part models. In *2011 International Conference on Computer Vision*, pages 1832–1839. IEEE.

[18] Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*.

[19] Castrejon, L., Ballas, N., and Courville, A. (2019). Improved conditional vrnns for video prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7608–7617.

[20] Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., and Su, H. (2021). Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133.

[21] Chen, R., Han, S., Xu, J., and Su, H. (2019). Point-based multi-view stereo network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1538–1547.

[22] Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

[23] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer.

[24] Crawford, E. and Pineau, J. (2019). Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3412–3420.

[25] Crawford, E. and Pineau, J. (2020). Exploiting spatial invariance for scalable unsupervised object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3684–3692.

[26] Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.

[27] De Bonet, J. S. and Viola, P. (1999). Poxels: Probabilistic voxelized volume reconstruction. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 418–425. Citeseer.

[28] Deng, K., Liu, A., Zhu, J.-Y., and Ramanan, D. (2022). Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891.

[29] Denton, E. and Fergus, R. (2018). Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*.

[30] Diuk, C., Cohen, A., and Littman, M. L. (2008). An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 240–247.

[31] Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

[32] Du, Y., Smith, K., Ulman, T., Tenenbaum, J., and Wu, J. (2020). Unsupervised discovery of 3d physical objects from video. *arXiv preprint arXiv:2007.12348*.

[33] Durbin, J. and Koopman, S. J. (2012). *Time series analysis by state space methods*, volume 38. OUP Oxford.

[34] Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11.

[35] Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2019). Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*.

[36] Eslami, S., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. (2016). Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems*, 29:3225–3233.

[37] Fan, H., Zhu, L., and Yang, Y. (2019). Cubic lstms for video prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8263–8270.

[38] Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., and Tian, Q. (2022). Fast dynamic radiance fields with time-aware neural voxels. *arXiv preprint arXiv:2205.15285*.

[39] Faugeras, O. and Keriven, R. (2002). *Variational principles, surface evolution, PDE's, level set methods and the stereo problem*. IEEE.

[40] Finn, C., Goodfellow, I., and Levine, S. (2016). Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72.

[41] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

[42] Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. (2016). Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pages 2199–2207.

[43] Franceschi, J.-Y., Delasalles, E., Chen, M., Lamprier, S., and Gallinari, P. (2020). Stochastic latent residual video prediction. *arXiv preprint arXiv:2002.09219*.

[44] Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2009). Manhattan-world stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1422–1429. IEEE.

[45] Furukawa, Y. and Hernández, C. (2015). Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 9(1-2):1–148.

[46] Gallup, D., Frahm, J.-M., Mordohai, P., Yang, Q., and Pollefeys, M. (2007). Real-time plane-sweeping stereo with multiple sweeping directions. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[47] Gao, H., Xu, H., Cai, Q.-Z., Wang, R., Yu, F., and Darrell, T. (2019). Disentangling propagation and generation for video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9006–9015.

[48] Gao, Q., Wang, B., Liu, L., and Chen, B. (2021a). Unsupervised co-part segmentation through assembly. In *International Conference on Machine Learning*, pages 3576–3586. PMLR.

[49] Gao, X., Hu, W., and Qi, G.-J. (2021b). Self-supervised multi-view learning via auto-encoding 3d transformations. *arXiv preprint arXiv:2103.00787*.

[50] Garbin, S. J., Kowalski, M., Johnson, M., Shotton, J., and Valentin, J. (2021). Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14346–14355.

[51] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[52] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

[53] Goyal, A. G. A. P., Sordoni, A., Côté, M.-A., Ke, N. R., and Bengio, Y. (2017). Z-forcing: Training stochastic recurrent networks. In *Advances in neural information processing systems*, pages 6713–6723.

[54] Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[55] Greff, K. (2020). What are objects. URL: https://oolworkshop.github.io/program/ool_34.html. Object-Oriented Learning (OOL): Perception, Representation, and Reasoning Workshop.

[56] Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D. J., Gnanapragasam, D., Golemo, F., Herrmann, C., Kipf, T., Kundu, A., Lagun, D., Laradji, I., Liu, H.-T. D., Meyer, H., Miao, Y., Nowrouzezahrai, D., Oztireli, C., Pot, E., Radwan, N., Rebain, D., Sabour, S., Sajjadi, M. S. M., Sela, M., Sitzmann, V., Stone, A., Sun, D., Vora, S., Wang, Z., Wu, T., Yi, K. M., Zhong, F., and Tagliasacchi, A. (2022). Kubric: a scalable dataset generator.

[57] Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR.

[58] Greff, K., Van Steenkiste, S., and Schmidhuber, J. (2017). Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pages 6691–6701.

[59] Gu, X., Fan, Z., Zhu, S., Dai, Z., Tan, F., and Tan, P. (2020). Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2495–2504.

[60] Guen, V. L. and Thome, N. (2020). Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11474–11484.

[61] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, 30.

[62] Guo, M., Fathi, A., Wu, J., and Funkhouser, T. (2020). Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*.

[63] Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565.

[64] Han, J., Min, M. R., Han, L., Li, L. E., and Zhang, X. (2021). Disentangled recurrent wasserstein autoencoder. *arXiv preprint arXiv:2101.07496*.

[65] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[66] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

[67] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[68] He, Z., Li, J., Liu, D., He, H., and Barber, D. (2019). Tracking by animation: Unsupervised learning of multi-object attentive trackers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1318–1327.

[69] Hedman, P., Srinivasan, P. P., Mildenhall, B., Barron, J. T., and Debevec, P. (2021). Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884.

[70] Heigl, B., Koch, R., Pollefeys, M., Denzler, J., and Gool, L. V. (1999). Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Mustererkennung 1999*, pages 94–101. Springer.

[71] Henderson, P. and Lampert, C. H. (2020). Unsupervised object-centric video generation and decomposition in 3d. *Advances in Neural Information Processing Systems*, 33:3106–3117.

[72] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[73] Hsieh, J.-T., Liu, B., Huang, D.-A., Fei-Fei, L. F., and Niebles, J. C. (2018). Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems*, pages 517–526.

[74] Hsu, W.-N., Zhang, Y., and Glass, J. (2017). Unsupervised learning of disentangled and interpretable representations from sequential data. *Advances in neural information processing systems*, 30.

[75] Huang, P.-H., Matzen, K., Kopf, J., Ahuja, N., and Huang, J.-B. (2018). Deepmvs: Learning multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2821–2830.

[76] Hung, W.-C., Jampani, V., Liu, S., Molchanov, P., Yang, M.-H., and Kautz, J. (2019). Scops: Self-supervised co-part segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–878.

[77] Huntley-Fenner, G., Carey, S., and Solimando, A. (2002). Objects are individuals but stuff doesn't count: Perceived rigidity and cohesiveness influence infants' representations of small groups of discrete entities. *Cognition*, 85(3):203–221.

[78] Im, S., Jeon, H.-G., Lin, S., and Kweon, I. S. (2019). Dpsnet: End-to-end deep plane sweep stereo. *arXiv preprint arXiv:1905.00538*.

[79] Jaderberg, M., Simonyan, K., Zisserman, A., et al. (2015). Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025.

[80] Jain, A., Tancik, M., and Abbeel, P. (2021). Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894.

[81] Jepson, A. D., Fleet, D. J., and Black, M. J. (2002). A layered motion representation with occlusion and compact spatial support. In *European Conference on Computer Vision*, pages 692–706. Springer.

[82] Ji, M., Gall, J., Zheng, H., Liu, Y., and Fang, L. (2017). Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315.

[83] Jia, X., De Brabandere, B., Tuytelaars, T., and Gool, L. V. (2016). Dynamic filter networks. In *Advances in neural information processing systems*, pages 667–675.

[84] Jiang, J., Janghorbani, S., De Melo, G., and Ahn, S. (2019). Scalor: Generative world models with scalable object representations. *arXiv preprint arXiv:1910.02384*.

[85] Jiang, J., Janghorbani, S., De Melo, G., and Ahn, S. (2020). Scalor: Generative world models with scalable object representations. In *ICLR*.

[86] Johari, M. M., Lepoittevin, Y., and Fleuret, F. (2022). Geonerf: Generalizing nerf with geometry priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18365–18375.

[87] Jojic, N. and Frey, B. J. (2001). Learning flexible sprites in video layers. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE.

[88] Kabra, R., Zoran, D., Erdogan, G., Matthey, L., Creswell, A., Botvinick, M., Lerchner, A., and Burgess, C. (2021). Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *Advances in Neural Information Processing Systems*, 34.

[89] Kajiya, J. T. and Von Herzen, B. P. (1984). Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174.

[90] Kalchbrenner, N., Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. (2017). Video pixel networks. In *International Conference on Machine Learning*, pages 1771–1779.

[91] Kantorovich, L. (1942). On the translocation of masses. c. r. *Doklady) Acad. Sci. URSS (NS)*, 37:199–201.

[92] Kazhdan, M. and Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13.

[93] Kim, M., Seo, S., and Han, B. (2021). Infonerf: Ray entropy minimization for few-shot neural volume rendering. *arXiv preprint arXiv:2112.15399*.

[94] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[95] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[96] Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. (2022). Conditional object-centric learning from video. In *ICLR*.

[97] Kobayashi, S., Matsumoto, E., and Sitzmann, V. (2022). Decomposing nerf for editing via feature field distillation. *arXiv preprint arXiv:2205.15585*.

[98] Koffka, K. (2013). *Principles of Gestalt psychology*. Routledge.

[99] Kosiorek, A., Kim, H., Teh, Y. W., and Posner, I. (2018). Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, pages 8606–8616.

[100] Kossen, J., Stelzner, K., Hussing, M., Voelcker, C., and Kersting, K. (2019). Structured object-aware physics prediction for video modeling and planning. *arXiv preprint arXiv:1910.02425*.

[101] Krishnan, R. G., Shalit, U., and Sontag, D. (2015). Deep kalman filters. *arXiv preprint arXiv:1511.05121*.

[102] Krishnan, R. G., Shalit, U., and Sontag, D. (2016). Structured inference networks for nonlinear state space models. *arXiv preprint arXiv:1609.09869*.

[103] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[104] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

[105] Lazova, V., Guzov, V., Olszewski, K., Tulyakov, S., and Pons-Moll, G. (2022). Control-nerf: Editable feature volumes for scene rendering and manipulation. *arXiv preprint arXiv:2204.10850*.

[106] Lea, S. E., Slater, A. M., and Ryan, C. M. (1996). Perception of object unity in chicks: A comparison with the human infant. *Infant Behavior and Development*, 19(4):501–504.

[107] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

[108] Lee, A. X., Zhang, R., Ebert, F., Abbeel, P., Finn, C., and Levine, S. (2018). Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*.

[109] Leslie, A. M. and Keeble, S. (1987). Do six-month-old infants perceive causality? *Cognition*, 25(3):265–288.

[110] Li, N., Eastwood, C., and Fisher, R. (2020). Learning object-centric representations of multi-object scenes from multiple views. *Advances in Neural Information Processing Systems*, 33:5656–5666.

[111] Li, N., Raza, M. A., Hu, W., Sun, Z., and Fisher, R. (2021). Object-centric representation learning with generative spatial-temporal factorization. *Advances in Neural Information Processing Systems*, 34:10772–10783.

[112] Liang, W., Xu, P., Guo, L., Bai, H., Zhou, Y., and Chen, F. (2021). A survey of 3d object detection. *Multimedia Tools and Applications*, 80(19):29617–29641.

[113] Liang, X., Lee, L., Dai, W., and Xing, E. P. (2017). Dual motion gan for future-flow embedded video prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1744–1752.

[114] Lin, Z., Wu, Y.-F., Peri, S., Fu, B., Jiang, J., and Ahn, S. (2020a). Improving generative imagination in object-centric world models. *arXiv preprint arXiv:2010.02054*.

[115] Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. (2020b). Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. *arXiv preprint arXiv:2001.02407*.

[116] Liu, L., Gu, J., Zaw Lin, K., Chua, T.-S., and Theobalt, C. (2020). Neural sparse voxel fields. *Advances in Neural Information Processing Systems*, 33:15651–15663.

[117] Liu, S., Zhang, L., Yang, X., Su, H., and Zhu, J. (2021). Unsupervised part segmentation through disentangling appearance and shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8355–8364.

[118] Liu, Y., Cao, X., Dai, Q., and Xu, W. (2009). Continuous depth estimation for multi-view stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2121–2128. IEEE.

[119] Liu, Z., Yeh, R. A., Tang, X., Liu, Y., and Agarwala, A. (2017). Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4463–4471.

[120] Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*.

[121] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[122] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[123] Lotter, W., Kreiman, G., and Cox, D. (2015). Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1511.06380*.

[124] Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE International Conference on Computer Vision (ICCV)*.

[125] Lu, C., Hirsch, M., and Scholkopf, B. (2017). Flexible spatio-temporal networks for video prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6523–6531.

[126] Luc, P., Clark, A., Dieleman, S., Casas, D. d. L., Doron, Y., Cassirer, A., and Simonyan, K. (2020). Transformation-based adversarial video prediction on large-scale data. *arXiv preprint arXiv:2003.04035*.

[127] Luc, P., Neverova, N., Couprie, C., Verbeek, J., and LeCun, Y. (2017). Predicting deeper into the future of semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 648–657.

[128] MacKay, D. J. (1992). A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

[129] Mallya, A., Wang, T.-C., Sapra, K., and Liu, M.-Y. (2020). World-consistent video-to-video synthesis. In *European Conference on Computer Vision*, pages 359–378. Springer.

[130] Marr, D. (2010). *Vision: A computational investigation into the human representation and processing of visual information*. MIT press.

[131] Mathieu, M., Couprie, C., and LeCun, Y. (2015). Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*.

[132] Mena, G., Belanger, D., Linderman, S., and Snoek, J. (2018). Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*.

[133] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470.

[134] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421. Springer.

[135] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[136] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

[137] Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*.

[138] Murez, Z., As, T. v., Bartolozzi, J., Sinha, A., Badrinarayanan, V., and Rabinovich, A. (2020). Atlas: End-to-end 3d scene reconstruction from posed images. In *European Conference on Computer Vision*, pages 414–431. Springer.

[139] Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

[140] Niemeyer, M., Barron, J. T., Mildenhall, B., Sajjadi, M. S., Geiger, A., and Radwan, N. (2021). Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. *arXiv preprint arXiv:2112.00724*.

[141] Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29.

[142] Nozick, V., de Sorbier, F., and Saito, H. (2008). Plane-sweep algorithm: Various tools for computer vision. In *Technical Committee on Pattern Recognition and Media Understanding (PRMU'08)*, volume 107, pages 87–94.

[143] Oechsle, M., Peng, S., and Geiger, A. (2021). Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599.

[144] Oprea, S., Martinez-Gonzalez, P., Garcia-Garcia, A., Castro-Vargas, J. A., Orts-Escolano, S., Garcia-Rodriguez, J., and Argyros, A. (2020). A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[145] Park, J. J., Florence, P., Straub, J., Newcombe, R., and Lovegrove, S. (2019a). Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174.

[146] Park, T., Liu, M.-Y., Wang, T.-C., and Zhu, J.-Y. (2019b). Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346.

[147] Peyré, G. and Cuturi, M. (2018). Computational optimal transport. arxiv e-prints, page. *arXiv preprint arXiv:1803.00567*.

[148] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

[149] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

[150] Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., and Chopra, S. (2014). Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*.

[151] Rasouli, A. (2020). Deep learning for vision-based prediction: A survey. *arXiv preprint arXiv:2007.00095*.

[152] Reda, F. A., Liu, G., Shih, K. J., Kirby, R., Barker, J., Tarjan, D., Tao, A., and Catanzaro, B. (2018). Sdc-net: Video prediction using spatially-displaced convolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 718–733.

[153] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[154] Reiser, C., Peng, S., Liao, Y., and Geiger, A. (2021). Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14335–14345.

[155] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

[156] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[157] Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. *Advances in neural information processing systems*, 30.

[158] Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., and Li, H. (2019). Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314.

[159] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. *Advances in neural information processing systems*, 29.

[160] Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

[161] Schonberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[162] Shum, H.-Y. and He, L.-W. (1999). Rendering with concentric mosaics. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 299–306.

[163] Siarohin, A., Roy, S., Lathuilière, S., Tulyakov, S., Ricci, E., and Sebe, N. (2021). Motion-supervised co-part segmentation. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9650–9657. IEEE.

[164] Sitzmann, V., Zollhöfer, M., and Wetzstein, G. (2019). Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems (NeurIPS)*, 32.

[165] Spelke, E. S. (1990). Principles of object perception. *Cognitive science*, 14(1):29–56.

[166] Spelke, E. S. and Kinzler, K. D. (2007). Core knowledge. *Developmental science*, 10(1):89–96.

[167] Srivastava, N., Mansimov, E., and Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852.

[168] Stelzner, K., Kersting, K., and Kosiorek, A. R. (2021). Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv preprint arXiv:2104.01148*.

[169] Su, J. (2018). Gan-qp: A novel gan framework without gradient vanishing and lipschitz constraint. *arXiv preprint arXiv:1811.07296*.

[170] Sun, C., Sun, M., and Chen, H.-T. (2021a). Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. *arXiv preprint arXiv:2111.11215*.

[171] Sun, J., Xie, Y., Chen, L., Zhou, X., and Bao, H. (2021b). Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15598–15607.

[172] Tancik, M., Mildenhall, B., Wang, T., Schmidt, D., Srinivasan, P. P., Barron, J. T., and Ng, R. (2021). Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2846–2855.

[173] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., and Ng, R. (2020). Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547.

[174] Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., et al. (2020). State of the art on neural rendering. In *Computer Graphics Forum*, volume 39, pages 701–727. Wiley Online Library.

[175] Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Yifan, W., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., et al. (2022). Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library.

[176] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[177] Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2017). Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*.

[178] Trevithick, A. and Yang, B. (2021). Grf: Learning a general radiance field for 3d representation and rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15182–15192.

[179] Tulyakov, S., Liu, M.-Y., Yang, X., and Kautz, J. (2018). Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535.

[180] Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

[181] Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380.

[182] Valenza, E., Leo, I., Gava, L., and Simion, F. (2006). Perceptual completion in newborn human infants. *Child Development*, 77(6):1810–1821.

[183] Van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*.

[184] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[185] Veerapaneni, R., Co-Reyes, J. D., Chang, M., Janner, M., Finn, C., Wu, J., Tenenbaum, J., and Levine, S. (2020). Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, pages 1439–1456. PMLR.

[186] Villani, C. (2009). *Optimal transport: old and new*, volume 338. Springer.

[187] Villegas, R., Pathak, A., Kannan, H., Erhan, D., Le, Q. V., and Lee, H. (2019). High fidelity video prediction with large stochastic recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 81–91.

[188] Villegas, R., Yang, J., Hong, S., Lin, X., and Lee, H. (2017). Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*.

[189] Vogiatzis, G., Esteban, C. H., Torr, P. H., and Cipolla, R. (2007). Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2241–2246.

[190] Vondrick, C. and Torralba, A. (2017). Generating the future with adversarial transformers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1020–1028.

[191] Wagemans, J., Elder, J. H., Kubovy, M., Palmer, S. E., Peterson, M. A., Singh, M., and von der Heydt, R. (2012). A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6):1172.

[192] Walker, J., Doersch, C., Gupta, A., and Hebert, M. (2016). An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer.

[193] Walker, J., Gupta, A., and Hebert, M. (2015). Dense optical flow prediction from a static image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2443–2451.

[194] Walker, J., Razavi, A., and Oord, A. v. d. (2021). Predicting video with vqvae. *arXiv preprint arXiv:2103.01950*.

[195] Wang, J. Y. and Adelson, E. H. (1993). Layered representation for motion analysis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 361–366. IEEE.

[196] Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021a). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*.

[197] Wang, Q., Wang, Z., Genova, K., Srinivasan, P. P., Zhou, H., Barron, J. T., Martin-Brualla, R., Snavely, N., and Funkhouser, T. (2021b). Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699.

[198] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Liu, G., Tao, A., Kautz, J., and Catanzaro, B. (2018). Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*.

[199] Wang, Y., Bilinski, P., Bremond, F., and Dantcheva, A. (2020). G3an: Disentangling appearance and motion for video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5264–5273.

[200] Wang, Y., Wu, H., Zhang, J., Gao, Z., Wang, J., Yu, P., and Long, M. (2022). Predrnn: A recurrent neural network for spatiotemporal predictive learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

[201] Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., and Zhou, J. (2021). Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5610–5619.

[202] Weis, M. A., Chitta, K., Sharma, Y., Brendel, W., Bethge, M., Geiger, A., and Ecker, A. S. (2020). Unmasking the inductive biases of unsupervised object representations for video sequences. *arXiv preprint arXiv:2006.07034*, 2.

[203] Weissenborn, D., Täckström, O., and Uszkoreit, J. (2019). Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*.

[204] Wichers, N., Villegas, R., Erhan, D., and Lee, H. (2018). Hierarchical long-term video prediction without supervision. *arXiv preprint arXiv:1806.04768*.

[205] Wu, T., Zhong, F., Tagliasacchi, A., Cole, F., and Oztireli, C. (2022). D^{2}nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *arXiv preprint arXiv:2205.15838*.

[206] Wu, Y., Gao, R., Park, J., and Chen, Q. (2020). Future video synthesis with object motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5539–5548.

[207] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

[208] Xiang, Y., Mottaghi, R., and Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE winter conference on applications of computer vision*, pages 75–82. IEEE.

[209] Xie, C., Park, K., Martin-Brualla, R., and Brown, M. (2021). Fig-nerf: Figure-ground neural radiance fields for 3d object category modelling. In *2021 International Conference on 3D Vision (3DV)*, pages 962–971. IEEE.

[210] Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810.

[211] Xu, J., Ni, B., Li, Z., Cheng, S., and Yang, X. (2018). Structure preserving video prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1460–1469.

[212] Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., and Neumann, U. (2022). Point-nerf: Point-based neural radiance fields. *arXiv preprint arXiv:2201.08845*.

[213] Xue, T., Wu, J., Bouman, K., and Freeman, B. (2016). Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in neural information processing systems*, pages 91–99.

[214] Yang, B., Wang, S., Markham, A., and Trigoni, N. (2020). Robust attentional aggregation of deep feature sets for multi-view 3d reconstruction. *International Journal of Computer Vision*, 128(1):53–73.

[215] Yang, B., Zhang, Y., Li, Y., Cui, Z., Fanello, S., Bao, H., and Zhang, G. (2022). Neural rendering in a room: Amodal 3d understanding and free-viewpoint rendering for the closed scene composed of pre-captured objects. *arXiv preprint arXiv:2205.02714*.

[216] Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G., and Cui, Z. (2021). Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788.

[217] Yao, Y., Luo, Z., Li, S., Fang, T., and Quan, L. (2018). Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783.

[218] Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. (2021). Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34.

[219] Ye, V., Li, Z., Tucker, R., Kanazawa, A., and Snavely, N. (2022). Deformable sprites for unsupervised video decomposition. *arXiv preprint arXiv:2204.07151*.

[220] Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., and Tenenbaum, J. B. (2019). Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*.

[221] Yingzhen, L. and Mandt, S. (2018). Disentangled sequential autoencoder. In *International Conference on Machine Learning*, pages 5670–5679. PMLR.

[222] Yu, A., Li, R., Tancik, M., Li, H., Ng, R., and Kanazawa, A. (2021a). Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761.

[223] Yu, A., Ye, V., Tancik, M., and Kanazawa, A. (2021b). pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587.

[224] Yu, H.-X., Guibas, L. J., and Wu, J. (2021c). Unsupervised discovery of object radiance fields. *arXiv preprint arXiv:2107.07905*.

[225] Yuan, W., Lv, Z., Schmidt, T., and Lovegrove, S. (2021). Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13144–13152.

[226] Zablotskaia, P., Dominici, E. A., Sigal, L., and Lehrmann, A. M. (2021). Provide: a probabilistic framework for unsupervised video decomposition. In *Uncertainty in Artificial Intelligence*, pages 2019–2028. PMLR.

[227] Zhang, J., Liu, X., Ye, X., Zhao, F., Zhang, Y., Wu, M., Zhang, Y., Xu, L., and Yu, J. (2021). Editable free-viewpoint video using a layered neural representation. *ACM Transactions on Graphics (TOG)*, 40(4):1–18.

[228] Zhang, K., Riegler, G., Snavely, N., and Koltun, V. (2020). Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*.

[229] Zhu, Y., Min, M. R., Kadav, A., and Graf, H. P. (2020). S3vae: Self-supervised sequential vae for representation disentanglement and data generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6538–6547.

[230] Zoran, D., Kabra, R., Lerchner, A., and Rezende, D. J. (2021). Parts: Unsupervised segmentation with slots, attention and independence maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10439–10447.

Beril Besbinar

CV / ML Researcher

SUMMARY

With 10 years of hands-on experience in computer vision and pattern recognition, I acquired knowledge in many topics, such as neural rendering, 3D vision, object-centric representations, tracking, segmentation and object detection on satellite images, as well as graph-based representations for whole-slide biomedical images. I have worked with both "old-school" methods and modern deep learning architectures. I have been involved in industrial and inter-disciplinary projects.

One thing I can say after this kind of experience in diverse settings is that I am always eager to learn and I immensely enjoy discussing over new ideas.

I am looking for a position where I can work on challenging real-world problems in an open-minded, collaborative environment, where I can learn and contribute.

CONTACT

Address:
Rue Saint-Martin 34
CH-1005 Lausanne

Phone:
+41 78 768 78 09

Email
berilbesbinar@gmail.com

REFERENCES

Pascal Frossard EPFL - LTS4
pascal.frossard@epfl.ch

Dorina Thanou EPFL - CIS
dorina.thanou@epfl.ch

Christian Forster Meta
cfo@fb.com

EDUCATION

Ph.D., School of Communication and Computer Science September 2016 - August 2022
École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
Advisor: Prof. Pascal Frossard
Thesis Title: Unsupervised Visual Entity Abstraction Towards 2D and 3D Compositional Models

M.Sc., Department of Electrical and Electronics Engineering September 2012 - August 2015
Middle East Technical Unitversity (METU), Ankara, Turkey
Advisor: Prof. A. Aydın Alatan
Thesis Title: Hierarchical Representations for Visual Object Tracking by Detection

B.Sc., Department of Electrical and Electronics Engineering September 2008 - July 2012
Middle East Technical Unitversity (METU), Ankara, Turkey
Specialization: Signal Processing and Biomedical

WORK EXPERIENCE

Doctoral Assistant, LTS4 - EPFL September 2016 - August 2022
Lausanne, Switzerland

- Object-centric video representations and object-centric stochastic video prediction
- Graph-based representations for mIHC images for immunotherapy response prediction (iLearn project, in collaboration with SDSC and CHUV)
- Tools: recurrent neural networks (LSTM, GRU), generative models (VAE, WGAN-GP) graphical models, graph-based representations and graph neural networks; using Tensorflow

External Research Collaborator (20%), Meta January 2022 - May 2022
(remote) Zurich, Switzerland

- Unsupervised object-abstraction in 3D
- Tools: optimal transport, rigid-body-transformations; using Pytorch

Research Engineer Intern, Facebook, XR Tech July 2021 - October 2021
(remote) Zurich, Switzerland

- Neural rendering from volumetric representations
- Set up of a working pipeline in company infrastructure from scratch
- Tools: neural radiance fields (NeRF), 3D geometry, self-attention; using Pytorch

Research and Teaching Assistant, EE - METU September 2012 - August 2016
Ankara, Turkey

- Object tracking with particle filters based on autoencoder representations
- Tools: denoising autoencoders; using Theano, particle filters; using Matlab

Researcher, Center for Image Analysis - METU March 2015 - July 2016
Ankara, Turkey

- Literature survery and tutorial on deep learning
- Object tracking with recurrent neural networks
- Tools: recurrent neural networks (LSTM), using Theano

Summer Researcher, LTS4 - EPFL June 2014 - August 2014
Lausanne, Switzerland

- Classification of visual patterns by sparse, parametric decompositions
- Tools: geometric dictionaries, orthogonal matching pursuit; using Matlab

Researcher, EE - METU September 2012 - March 2014
Ankara, Turkey

- Segmentation and object detection on satellite images
- Tools: image processing algorithms, BoW desciptor, SVM.; using Matlab

Engineering Intern, ASELSAN June 2011 - July 2011
Ankara, Turkey

- HF Data Modem tranmitter implementation on a DSP board

TEACHING

Teaching Assistant:
- EPFL, EE 350 - Signal Processing *Fall-2017, Fall-2018, Fall-2019*
- EPFL, COM-500 - Statistical Signal and Data Processing through Applications *Spring 2017, Spring 2018*
- METU, EE-214 - Electronic CIrcuit Laboratory II *Spring 2013, Spring 2014, Spring 2015, Spring 2016 (as the coordinator assistant)*
- METU, EE-213 - Electronic CIrcuit Laboratory I *Fall 2013, Spring 2014, Fall 2015 (as the coordinator assistant)*
- METU, EE-201 - CIrcuit Theory I *Fall 2012*
- METU, EE-301 - Signals and Systems *Fall 2012*

Supervised Student Projects: *(* indicates co-supervision)*
- *Understanding Implicit Neural Representations, Gizem Yuce, *Master Semester Project & Summer Internship, Spring-Summer 2021, EPFL*
- *Graph-based models for immunotherapy response prediction, Asmi Souhail, *Master Thesis, EPFL, Spring 2021*
- *Cell-graph technique for feature extraction, Carl Hatoum, *Master Semester Project, EPFL, Fall 2020*
- Learning multi-graph representations for classification, Yinan Zhang, *Master Semester Project & Master Thesis, EPFL, Spring 2020*
- Wasserstein-GANs for image generation and encoding, Pietro Elia Carta, *Master Semester Project, Spring 2020*
- Cell segmentation in biomedical images, Deniz Sayin Mercadier, *Summer Internship, EPFL, Summer 2018*
- Variational Recurrent Neural Networks for Long-Term Future Frame Prediction, Haziq Bin Razali, *Master Semester Project, EPFL, Fall 2018*
- Transformation invariant deep learning systems, Nicolas Masserey, *Master Semester Project, EPFL, Fall 2018*

PUBLICATIONS

"A Structured Dictionary Perspective on Implicit Neural Representations", Yüce, G.; Ortiz-Jiménez, G., Beşbınar, B.; Frossard, P., in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022

"Self-Supervision By Prediction For Object Discovery In Videos", Beşbınar, B.; Frossard, P., in IEEE International Conference on Image Processing (ICIP), 2021.

"Learning to Represent Whole Slide Images by Selecting Cell Graphs of Patches", Zhang, Y.; Beşbınar, B.; Frossard, P., Medical Imaging with Deep Learning (MIDL) 2021 ***(short paper).***

"Automatic Segmentation of Nuclei in Histopathology Images using Encoding-Decoding Convolutional Neural Networks", Mercadier, D.S.; Beşbınar, B.; Frossard, P., in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.

"Unsupervised Change Detection in Satellite Images using Oversegmentation and Mutual Information", Taşkesen, B.; Beşbınar, B.; Koz, A.; Alatan, A.A., in IEEE Signal Processing and Communications Applications Conference (SIU), 2017.

"Visual object tracking with autoencoder representations,", Beşbınar, B.; Alatan, A.A., in IEEE Signal Processing and Communications Applications Conference (SIU), 2016.

"Inshore ship detection in high-resolution satellite images: approximation of harbors using sea-land segmentation", Beşbınar, B.; Alatan, A.A., in SPIE Remote Sensing 2015, 21-24 September 2015

"Hierarchical Representations for Visual Object Tracking by Detection", Beşbınar Beril, MSc Thesis, METU, September 2015

"Inshore ship detection in multispectral satellite images", Beşbınar, B.; Gürbüz, Y.Z.; Alatan, A.A., in IEEE Signal Processing and Communications Applications Conference (SIU), 2015.

"Sea detection on high-resolution panchromatic satellite images using texture and intensity", Beşbınar, B.; Alatan, A.A., in IEEE Signal Processing and Communications Applications Conference (SIU), 2014.

in preperation:
- *Unsupervised Decomposition of Relocalized Objects with Scene Agnostic Radiance Fields*
- *Immunotheraphy Response Prediction with Hierarchical Representations of Whole-Slide Cell-Graphs*

SCHOLARSHIPS AND AWARDS

- EPFL EDIC Fellowship for 2016-2017
- Summer@EPFL Scholarship for 2014 Summer
- METU Graduate Awards – Course Performance Award for 2012-2013 academic year
- The Scientific and Technological Research Council of Turkey (TUBITAK) - Graduate Student Scholarship, 2012-2014
- METU - Dean's High Honor (7 times), Dean's Honor (1 time)
- METU Electrical and Electronics Engineering - Bulent Kerim Altay Award, Fall 2008, Spring 2009 (awarded to students with 4.0/4.0 GPA)
- Turk Egitim Vakfi - Merit Scholarship, 2007 December - 2012 June (awarded to 40 students nationwide based on their leadership skills)
- METU - Merit Scholarship, 2007 September - 2012 June
- Turkiye Is Bankasi - Golden Youth Award (for ranking 33rd/1.3 million in University Entrance Exam in Mathematics & Science area in 2007)

SKILLS

Language Skills

Turkish: Native
English: Fluent (TOEFL IBT Score: 103/120)
French: Intermediate (A2-B1)

Computer Skills

Operating Systems: Linux (Ubuntu), MacOS, Windows
Programming: Python, C++(limited experience), Matlab
Libraries: Pytorch (also w/ Lightning), Tensorflow, Theano, OpenCV, NumPy, Pandas, Scikit-Learn, Scikit-Image,
Software/Applications: Inkscape, Adobe Illustrator; Scribus; Latex