DISS. ETH NO. 28066

LEARNING DEEP MODELS WITH PRIMITIVE-BASED REPRESENTATIONS

A dissertation submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

presented by

DESPOINA PASCHALIDOU
Diploma in Electrical and Computer Engineering
Aristotle University of Thessaloniki

born on 8 June 1991
citizen of Switzerland

accepted on the recommendation of

Prof. Dr. Luc Van Gool, examiner
Prof. Dr. Andreas Geiger, co-examiner
Prof. Dr. Vittorio Ferrari, co-examiner
Prof. Dr. Federico Tombari, co-examiner
Prof. Dr. Manolis Savva, co-examiner

2021

Despoina Paschalidou: *Learning Deep Models with Primitive-Based Representations*, © 2021

To my parents who always wanted me to become a Doctor.

ABSTRACT

Humans develop a common-sense understanding of the physical behaviour of the world, within the first year of their life. We are able to identify 3D objects in a scene, infer their geometric and physical properties, predict physical events in dynamic environments and act based on our interaction with the world. Our understanding of our surroundings relies heavily on our ability to properly reason about the arrangement of elements in a scene. Inspired by early works in cognitive science that stipulate that the human visual system perceives objects as a collection of semantically coherent parts and in turn uses them to easily associate unknown objects with object parts whose functionality is already known, researchers developed compositional representations capable of capturing the functional composition and spatial arrangement of objects and object parts in a scene. In the first two parts of this dissertation, we propose learning-based solutions for recovering the 3D object geometry using semantically consistent part arrangements. Finally, we introduce a network architecture that synthesizes indoor environments as object arrangements, whose functional composition and spatial configuration follows clear patterns that are directly inferred from data.

First, we present an unsupervised learning-based approach for recovering shape abstractions using superquadric surfaces as atomic elements. We demonstrate that superquadrics lead to more expressive part decompositions while being easier to learn than cuboidal primitives. Moreover, we provide an analytical solution to the Chamfer loss which avoids the need for computational expensive reinforcement learning or iterative prediction.

Next, we introduce a novel 3D primitive representation that defines primitives using an Invertible Neural Network (INN) that implements homeomorphic mappings between a sphere and the target object. Since this representation does not impose any constraint on the shape of the predicted primitives, they can capture complex geometries using an order of magnitude fewer parts than existing primitive-based representations. We consider this representation a first step towards bridging the gap between interpretable and high fidelity primitive-based reconstructions.

Subsequently, we introduce a structure-aware representation that jointly recovers the geometry of a 3D object as a set of primitives as well as its latent hierarchical structure without any part-level supervision. Our model recovers the higher level structural decomposition of various objects in the form of a binary tree of primitives, where simple parts are represented with fewer primitives and more complex parts are modeled with more components. We demonstrate that considering the latent hierarchical layout of an object into parts facilitates reasoning about the 3D object geometry.

Finally, we propose a neural network architecture for synthesizing indoor scenes by plausibly arranging objects within the scene boundaries. In particular, given a room type (e.g. bedroom, living room) and its shape, our model generates meaningful object arrangements by sequentially placing objects in a permutation-invariant fashion. In contrast to prior work, which poses scene synthesis as a sequence generation problem, our model generates rooms as unordered sets of objects. This allows us to perform various interactive scenarios such as room completion, failure case correction, object suggestions with user-provided constraints etc.

To summarize, we propose novel primitive-based representations that do not limit the available shape vocabulary on a specific set of shapes such as cuboids, spheres, planes etc. Next, we introduce a structure-aware representation that considers part relationships and represents object parts with multiple levels of granularity, where geometrically complex parts are modeled with more components and simpler parts with fewer components. Finally, we propose a network architecture that generates indoor scenes by properly arranging objects within a room's boundaries. Our model enables new interactive applications for semi-automated scene authoring that were not possible before.

ZUSAMMENFASSUNG

Bereits im ersten Jahr unseres Lebens entwickeln wir ein gutes Verständnis für das physikalische Verhalten der Welt. Wir sind in der Lage 3D-Objekte in einer Szene zu identifizieren, auf deren geometrischen und physikalischen Eigenschaften zu schliessen, physikalische Ereignisse in dynamischen Umgebungen vorherzusagen und basierend auf unserer Interaktion mit der Welt zu handeln. Das Verständnis unserer Umgebung hängt stark davon ab, inwieweit wir das Arrangement von Elementen in einer Szene richtig analysieren können. Inspiriert von früheren Arbeiten der Kognitionswissenschaft , die besagen, dass das menschliche visuelle System Objekte als eine Sammlung von semantisch kohärenten Teilen erkennt und diese wiederum nutzt, um unbekannte Objekte mit Objektteilen deren Funktionalität bereits bekannt ist, zu assoziieren, haben Forscher Kompositionsdarstellungen entwickelt. Diese Darstellungen können die funktionale Einrichtung und räumliche Anordnung von Objekten und Objektteilen in einer Szene erfassen. Im ersten Teil dieser Dissertation schlagen wir lernbasierte Lösungen vor, zur Wiederherstellung der 3D-Objektgeometrie unter Verwendung von semantisch konsistenten Teilanordnungen. Im zweiten Teil stellen wir eine Netzwerkarchitektur vor, die Innenräume mit Objektanordnungen synthetisiert, deren semantische/funktionale EInrichtung und räumliche Anordnung klaren Mustern folgt, die unmittelbar aus Daten abgeleitet werden.

Wir stellen zuerst einen Ansatz, der sich auf unüberwachtes Lernen basiert, zur Wiederherstellung von Formabstraktionen unter Verwendung von superquadratischen Oberflächen (Superquadric Surfaces) als Elemente vor. Wir zeigen, dass Superquadrate (Superquadrics) zu ausdrucksstärkeren Teilzerlegungen führen und gleichzeitig einfacher zu erlernen sind, im Vergleich zu quaderförmige Primitive. Darüber hinaus bieten wir eine analytische Lösung für den Chamfer-Verlust (Chamfer Loss), die rechenintensives Bestärkendes Lernen oder iterative Vorhersage überflüssig macht.

Zunächst, führen wir eine neuartige 3D-Primitivdarstellung ein, die die Primitive mit Hilfe eines invertierbaren neuronalen Netzes (INN) definiert, das homeomorphische Zuordnungen (Homeomorphic Mappings) zwischen einer Sphäre und dem Zielobjekt implementiert. Da diese Darstellung keine Beschränkung auf die Form der Primitiven auferlegt, können sie komplexe Geometrien mit einer Grössenordnung weniger Teile repräsentie-

ren erfassen. Wir betrachten diese Darstellung als einen ersten Schritt die Kluft zwischen interpretierbaren und originalgetreuen primitiv-basierten Rekonstruktionen zu überbrücken.

Darüber hinaus führen wir eine strukturbewusste Darstellung ein, die die Geometrie eines 3D-Objekts als Primitiven und seine latente hierarchische Struktur wiedererlangt. Unser Modell erlangt die übergeordnete strukturelle Dekomposition verschiedener Objekte in Form eines Binärbaums von Primitiven, wobei einfache Teile mit weniger Primitiven und komplexere Teile mit mehr Komponenten modelliert werden. Wir zeigen, dass die Berücksichtigung des latenten hierarchischen Aufbaus eines Objekts in Teilen, die Argumentation über die 3D-Objektgeometrie erleichtert.

Wir bieten schliesslich eine Netzwerkarchitektur zur Synthese von Innenraumszenen durch die plausible Anordnung von Objekten innerhalb der Szenengrenzen an. Anhand eines Raumtypes (z.B. Schlafzimmer, Wohnzimmer) und seiner Form, erzeugt unser Modell sinnvolle Objektanordnungen, indem es die Objekte nacheinander, auf eine permutationsinvariante Weise, aufstellt. Im Gegensatz zu früheren Arbeiten, die Szenensynthese als Sequenzgenerierung (Sequence Generation Problem) darstellen, synthetisiert unser Modell Räume als ungeordnete Reihe von Objekten. Dies ermöglicht verschiedene interaktive Szenarien wie Raumvervollständigung, Fehlerkorrekturen, Objektvorschläge mit benutzerdefinierten Einschränkungen usw.

Zusammenfassend kann man festhalten, dass wir neue primitiv-basierte Darstellungen vorschlagen, die das verfügbare Formenvokabular nicht auf eine Reihe spezifischer Formen beschränkt wie z.B. Quader, Sphären, Ebenen usw. Darauffolgend, führen wir eine strukturbewusste Darstellung ein, die Teilbeziehungen berücksichtigt und Objektteile mit mehreren Abstraktionsebenen modelliert, wobei geometrisch komplexe Teile mit mehrere Komponenten und einfache Teile mit wenigere Komponenten modelliert werden. Anschliessend schlagen wir eine Netzwerkarchitektur vor, die Innenraumszenen erzeugt, indem sie eine sinnvolle Anordnung von Objekten innerhalb der Raumgrenzen bildet und neuartige interaktive Anwendungen für die halbautomatisierte Erstellung von Szenen ermöglicht.

PUBLICATIONS

The following publications are included as a whole or in parts in this dissertation:

- Despoina Paschalidou, Amlan Kar, Maria Shugrina Karsten Kreis, Andreas Geiger, Sanja Fidler. "ATISS: Autoregressive Transformers for Indoor Scene Synthesis". In: Advances in Neural Information Processing Systems (NeurIPS). 2021
- Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, Sanja Fidler. "Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks". In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2021
- Despoina Paschalidou, Luc van Gool, Andreas Geiger. "Learning Unsupervised Hierarchical Part Decomposition of 3D Objects from a Single RGB Image". In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2020
- Despoina Paschalidou, Ali Osman Ulusoy, Andreas Geiger. "Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids". In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019

Furthermore, the following publication for which I was major contributor was part of my PhD research, but is nevertheless not covered in this thesis. The topics of this publication are outside of the scope of the material covered here:

- Despoina Paschalidou, Ali Osman Ulusoy, Carolin Schmitt, Luc van Gool, Andreas Geiger. "RayNet: Learning Volumetric 3D Reconstructions with Ray Potentials". In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2018

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Prof Andreas Geiger for his guidance and for always trying to teach me how to decide what are the right questions one should ask when doing research. Moreover, I want to express my gratitude to Andreas for taking a chance on me, in the first place and offering me an amazing opportunity to do my PhD under his mentorship. This thesis would have never been in its current state without his guidance. Without his continuous support, understanding and kindness I would not be the person that I am today and for this I am deeply indebted to Andreas. Instead of pushing me to work on his ideas, Andreas always gave me the freedom to work on topics that I was excited about and was always eager to help me and support me towards achieving my goals and for this I feel extremely lucky. Throughout my PhD, Andreas tried to teach me how to better organize my time and my chaotic way of thinking as well as help me find my place in research and while I still need to work on my time management I believe that he did an amazing job. I am also grateful to him for creating a wonderful environment at AVG that always fostered curiosity, creativity and inclusivity. Some of my favourite memories during my PhD are during our ski and CVPR retreats, where we had the chance to discuss research, come up with project ideas, play cards, go skiing and do hikes in the Schwäbische Alb. For all these reasons, I can not imagine a better mentor than Andreas.

Furthermore, I would like to thank my second advisor Prof Luc van Gool for hosting me for 1.5 yeas at CVL at ETH Zurich. I am very grateful that he trusted me and allowed me to pursue the research directions that I was excited about. I genuinely enjoyed our discussions during my stay at ETH and I strongly believe that his support, generosity and kind treatment makes CVL a fantastic scientific environment to do research.

I also feel grateful and blessed for the opportunity to work with Prof Sanja Fidler during my PhD. Sanja is a great role model both as a researcher and as a woman, an amazing mentor and it was a privilege to closely interact with her during my one year internship at NVIDIA. I am deeply inspired by her enthusiasm towards research, eagerness to help and her amazing guidance and I hope that one day I will be able to collaborate with her again. During my stay at NVIDIA, I learned a lot, in particular about generative models, and I was able to become more confident about myself, my research interests and my ideas. This would have not been possible without Sanja's kindness, generosity and support. Sanja made sure that I was integrated in her group and offered me the intellectual freedom to work on problems that I was excited about. Working with Sanja has been one of my dreams since I was a Master student in Greece and I could not have been more happy about having this dream come true.

I am also indebted to David Novotny and Prof Andrea Vedaldi for giving me the great opportunity to join FAIR London and spend 4 wonderful months working with them, learning more about the very popular neural radiance fields and doing cool research using real world data. Working with David has been a very exciting experience in particular when we were brainstorming about the next research steps. His passion for research, patience and willingness to discuss with me at any date or time made my time at FAIR a great experience.

During my PhD, I was also blessed to get to know Prof Michael Black. Even though I was not able to collaborate with Michael, I was always impressed by his passion and excitement for research, his involvement in all the ongoing projects of his group, his support, kind personality and humbleness. I still remember feeling in awe, every time I was attending one of his talks during our joint group meetings. Michael's vision for the future of the field had a huge impact on my perception as a researcher and as a person. I will always be grateful for his support, kindness and advice and I really hope that in the future I will be able to collaborate with him. This will be another dream coming true.

Special gratitude goes to Prof Federico Tombari, Prof Vittorio Ferrari and Prof Manoli Savva for promptly agreeing to participate in my jury. Thank you so much for taking the time to review my thesis.

The unique environment at both MPI, ETH and NVIDIA allowed me to interact and collaborate with amazing researchers that had a huge impact on my personal and career development. First, I would like to thank Ali Osman Ulusoy for collaborating with me during my baby steps as a researcher. Osman is both a great researcher and an amazing person and I feel grateful that I was able to work and learn from him. I also feel blessed for my collaboration with Amlan Kar, Masha Shugrina and Karsten Keris during my time at NVIDIA. All of them made me feel that I am part of the group and definitely contributed into making my internship at NVIDIA one of my favourite memories during my PhD. Thank you so much!

My PhD would definitely have been a very boring experience if it wasn't for the AVG and PS members. Everyone at both AVG and PS contributed into making MPI a fantastic place to do research. To my first officemates David Stutz and Yiyi Liao, thank you so much for making the first months of my PhD less lonely. I am very lucky that I can call both of you my friends. I would particularly like to thank people in Andreas's and Michael's groups for the great memories that we shared: Joel, Benjamin, Lars, Michael, Michi, Simon, Aseem, Eshed, Kashyap, Aditya, Songyou, Katja, Axel, Xu, Caro, Qianli, Timo, Partha, Mohamed, Ahmed, Eric, Sergey, Soubhik, Gul. I would like to particularly thank Joel and Caro for taking the time to help me improve my skiing skills. Without Joel and Caro, I would probably still be afraid of the slopes. So thank you very much for making me more fearless.

When I think about my time at MPI, I recall the wonderful moments that we shared in particular with Vasilis Choutas and Georgios Pavlakos. Georgios is one of the most passionate researchers in our field that has managed to remain humble and kind despite his great accomplishments. His advice, feedback and support have definitely made me a better person and I am very lucky that he is my friend. While I knew Vasilis since my time in Thessaloniki, during our PhD we became close friends and I could not be more proud for all the great things he has achieved. I am grateful for his generosity, hospitality and kindness and for the fact that I know that I can always rely on him in case of need. Vasilis with his wife Evangelia have supported me throughout one of the most difficult periods of my PhD. If they weren't there, I am not sure that I would have managed to overcome this challenge. For this and many other reasons, I am very blessed to have you both as my friends. To the rest of the Greek mafia at Tubingen: Giannis, Dimitris, Rea, Atalanti, Alexandros thank you so much for making my time in Tubingen such an enjoyable experience.

Looking back at the time I spent at CVL, I am very happy I had the chance to meet Menelos Kanakis and Evangelos (Evan) Ntavelis. Both of you made my time at CVL a wonderful experience and I feel very lucky for meeting you. Even though I was always overwhelmed with deadlines and things that needed to be done, you always made sure that I was doing ok. Thank you very much for your patience with me, the emotional support and your positivity. You are both great friends and I am very lucky to have you in my life. I would also like to express my gratitude to Kevis-Kokitsi Maninis for making me feel welcome when I first joined the CVL. Whenever I was overwhelmed with stress, Kevis was always there to make me feel better. I would also like to particularly thank the CVL members: Christos, Stam, Danda, Ajad, David, Martin, Dario, Thomas, Janine, Alex, Andreas, Goutam for their hospitality and for making me feel part of the CVL family.

I also owe a big shout out to the amazing Melanie Feldhofer, Kerstin McGaughey, Camelia Fritz, Kris Haberer, Christina Krueger, Christine Braun-Roth for acting as a shield against any logistical and bureaucratic issues. Your kindness and support play a key role in making both MPI and ETH a wonderful place to do research and I am beyond grateful for all the things you provide us every day.

Without a doubt, the PhD is a very lonely endeavour. However, I was very lucky that I was able to share this journey with awesome people. Without Alexandros Lazaridis, Konstantina Goni, Nam Le, Apostolos Avranas, Christos Zalidis, Triantafyllos Afouras, George Tsolaridis, George Pappadopoulos, Christos Tsirigotis, Vagia Tsiminaki, Christina Pinneri, Ada Xatzipanagiotou, Dimitra Loupi, Rosa Pujades, Tamas Borbath my PhD would have been pretty lonely.

This thesis is dedicated to my parents for always being by my side, supporting me, loving me and believing that I am a much better person than I actually am. Their boundless love and support have allowed me to spread my wings and pursue my dreams. I hope that even though I didn't study Medicine and my career choices led me far away from home they are happy and proud for the things that I have accomplished throughout this journey. Next, I would like to thank my brothers George and Panagiotis for always loving and supporting me. Your passion, strength, perseverance and strive for always becoming a better person is a huge inspiration for me and I could not be more proud of you. Most importantly thank you for being a beacon of positive energy and courage in my life. Without you, my life would have been much more challenging. I also owe a big thanks to my favourite aunt Litsa who supported me and guided me when I first moved in Germany. Thank you for always loving me, believing in me and for being my greatest supporter.

The most special thanks goes to Angelos. There are not enough words that could possibly describe my feelings for you. Therefore I will only say that I feel blessed for having you in my life and I look forward to our next adventures. Without your love and support this dissertation would not have been possible.

CONTENTS

List of Figures xix
List of Tables xxii
1 INTRODUCTION 1
2 LEARNING 3D SHAPE PARSING BEYOND CUBOIDS 7
2.1 Related Work 9
2.2 Method Overview 13
2.2.1 Reconstruction Loss 13
2.2.2 Parsimony Loss 16
2.2.3 Superquadric Parametrization 16
2.3 Experimental Evaluation 17
2.3.1 Superquadrics vs. Cuboids 18
2.3.2 Results on ShapeNet dataset 20
2.3.3 Shape Abstraction from a Single RGB Image 24
2.3.4 Results on SURREAL dataset 25
2.3.5 Analytical Loss Formulation 26
2.4 Implementation Details 27
2.4.1 Metrics Computation 28
2.4.2 Parsimony Loss Details 29
2.5 Discussion 29
3 LEARNING EXPRESSIVE 3D SHAPE ABSTRACTIONS WITH INVERTIBLE NETWORKS 31
3.1 Related Work 33
3.2 Method 35
3.2.1 Primitives as Homeomorphic Mappings 35
3.2.2 Network Architecture 37
3.2.3 Training 38
3.3 Experimental Evaluation 42
3.3.1 Datasets 42
3.3.2 Metrics 42
3.3.3 Baselines 43
3.3.4 Representation Power 45
3.3.5 Reconstruction Accuracy 45
3.4 Ablation Study 48
3.4.1 Invertibility 48
3.4.2 Effect of $p_\theta(\cdot)$ 51

3.4.3 Loss Functions 52
3.4.4 Sensitivity to initialization 57
3.4.5 Semantic Consistency 58
3.5 Implementation Details 59
3.5.1 Network Architecture 59
3.5.2 Training Protocol 61
3.5.3 Sampling Strategy 62
3.6 Discussion 63
4 LEARNING UNSUPERVISED HIERARCHICAL PART DECOMPOSITION OF 3D OBJECTS 67
4.1 Related Work 69
4.2 Method Overview 71
4.2.1 Network Architecture 72
4.2.2 Primitive Parametrization 76
4.2.3 Network Losses 76
4.3 Experimental Evaluation 79
4.3.1 3D Reconstruction on ShapeNet 82
4.3.2 Volumetric Reconstruction 85
4.3.3 3D Reconstruction on D-FAUST 85
4.4 Implementation Details 86
4.4.1 Network Architecture 86
4.4.2 Training 88
4.4.3 Sampling Strategy 89
4.4.4 Empirical Analysis of Loss Formulation 90
4.5 Discussion 92
5 ATISS: AUTOREGRESSIVE TRANSFORMERS FOR INDOOR SCENE SYNTHESIS 95
5.1 Related Work 97
5.2 Method 99
5.2.1 Autoregressive Set Generation 99
5.2.2 Network Architecture 101
5.2.3 Training and Inference 103
5.3 Experimental Evaluation 104
5.3.1 Datasets 104
5.3.2 Baselines 104
5.3.3 Evaluation Metrics 106
5.3.4 Scene Synthesis 107
5.3.5 Applications 108
5.3.6 Perceptual Study 111

5.4 Ablation Study . . . 111
  5.4.1 Mixture of Logistic distributions . . . 112
  5.4.2 Layout Encoder . . . 112
  5.4.3 Transformers with Ordering . . . 113
5.5 Implementation Details . . . 114
  5.5.1 Network Architecture . . . 114
  5.5.2 Object Retrieval . . . 116
  5.5.3 Training Protocol . . . 117
5.6 Limitations . . . 117
5.7 Discussion . . . 118

6 DISCUSSION 119
6.1 Summary of Contributions . . . 119
6.2 Directions for Future Research . . . 120

A ADDITIONAL IMPLEMENTATION DETAILS 123
A.1 Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids . . . 123
  A.1.1 Derivation of Pointcloud-to-Primitive Loss . . . 123
  A.1.2 Empirical Analysis of Reconstruction Loss . . . 125
A.2 Learning Unsupervised Hierarchical Part Decomposition of 3D Objects . . . 128
  A.2.1 Occupancy Function . . . 128
A.3 ATISS: Autoregressive Transformers for Indoor Scene Synthesis 133
  A.3.1 3D-FRONT Data Preprocessing . . . 133
  A.3.2 Perceptual Study . . . 135

B ADDITIONAL EXPERIMENTAL RESULTS 139
B.1 Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids . . . 139
  B.1.1 Qualitative Results on SURREAL . . . 139
B.2 Learning Unsupervised Hierarchical Part Decomposition of 3D Objects . . . 139
  B.2.1 Additional Results on D-FAUST . . . 139
  B.2.2 Additional Results on ShapeNet . . . 144
B.3 Neural Parts: Learning Expressive 3D Shape Abstractions . . 148
  B.3.1 Experiment on D-FAUST . . . 148
  B.3.2 Experiment on FreiHAND . . . 149
  B.3.3 Experiment on ShapeNet . . . 152
  B.3.4 Semantically Consistent Abstractions . . . 157
B.4 ATISS: Autoregressive Transformers for Indoor Scene Synthesis 161
  B.4.1 Applications . . . 161

B.4.2 Scene Synthesis 164

LIST OF FIGURES

Figure 2.1 3D Shape Parsing. 8
Figure 2.2 Superquadrics Shape Space 12
Figure 2.3 Explicit Superquadric Equation 17
Figure 2.4 Reconstruction Loss 18
Figure 2.5 Training Evolution 19
Figure 2.6 Qualitative Results on ShapeNet dataset 19
Figure 2.7 Attention to Details 20
Figure 2.8 Qualitative Results on Motorbikes 21
Figure 2.9 Qualitative Results on Animals 22
Figure 2.10 Qualitative Results on Chairs 22
Figure 2.11 Evolution of Primitives 23
Figure 2.12 Single Image 3D Reconstruction on ShapeNet objects 24
Figure 2.13 Qualitative Results on SURREAL dataset 25
Figure 2.14 Analytical Loss Formulation 26
Figure 2.15 Volume-based Network Architecture 28
Figure 3.1 Expressive Primitives 32
Figure 3.2 Neural Parts: Method Overview 36
Figure 3.3 Conditional Coupling Layer 39
Figure 3.4 Trade-off Reconstruction Quality and # Primitives . . 44
Figure 3.5 Human Body Modelling 45
Figure 3.6 Single Image 3D Reconstruction on D-FAUST 46
Figure 3.7 Single Image 3D Reconstruction on FreiHAND 47
Figure 3.8 Single Image 3D Reconstruction on ShapeNet 48
Figure 3.9 Impact of the INN 49
Figure 3.10 Predicted Primitives with AtlasNet-sphere 50
Figure 3.11 Conditional Coupling Layer 51
Figure 3.12 Training Convergence w/o $p_{\boldsymbol{\theta}}(\cdot)$ 52
Figure 3.13 Ablation Study on Loss Terms 54
Figure 3.14 Impact of $\mathcal{L}_{norm}$ 55
Figure 3.15 Predicted Primitives w/o Normal Consistency Loss . 56
Figure 3.16 Predicted Primitives w/o Overlapping Loss 56
Figure 3.17 Impact of $\mathcal{L}_{cover}$ 57
Figure 3.18 Predicted Primitives w/o Coverage Loss 57
Figure 3.19 Sensitivity to Initialization 58
Figure 3.20 Semantic Consistency 59

Figure 3.21 Feature Extractor 60
Figure 3.22 Conditional Homeomorphism 61
Figure 4.1 Hierarchical Part Decomposition 68
Figure 4.2 Level of Detail. 69
Figure 4.3 Network Architecture 73
Figure 4.4 Structure Network 74
Figure 4.5 Predicted Hierarchies on D-FAUST 80
Figure 4.6 Single Image 3D Reconstruction on ShapeNet 81
Figure 4.7 Predicted Hierarchies on ShapeNet 83
Figure 4.8 Volumetric Reconstruction 84
Figure 4.9 Single Image 3D Reconstruction on D-FAUST 86
Figure 4.10 Feature Encoder Architectures 87
Figure 4.11 Network Architecture Overview 88
Figure 4.12 Geometry Network 89
Figure 4.13 Impact of Proximity Loss 91
Figure 5.1 ATISS: Motivation 96
Figure 5.2 ATISS: Method Overview 99
Figure 5.3 Training Overview 103
Figure 5.4 Qualitative Scene Synthesis Results 105
Figure 5.5 Scene Diversity 107
Figure 5.6 Generalization Beyond Training Data 108
Figure 5.7 Scene Completion 109
Figure 5.8 Failure Case Detection and Correction 110
Figure 5.9 Object Suggestion 110
Figure 5.10 Structure Encoder 115
Figure 5.11 Attribute Extractor 116
Figure 5.12 Failure Cases 117
Figure A.2 Empirical Analysis of Reconstruction Loss 127
Figure A.3 Implicit Surface Function of Superquadrics 128
Figure A.4 Implicit Surface Function 130
Figure A.5 Implicit Surface Function 131
Figure A.6 Implicit Surface Function 132
Figure A.7 Number of object occurrences in Bedrooms and Libraries 134
Figure A.8 Number of object occurrences in Living Rooms and Dining Rooms 135
Figure A.9 Perceptual Study UI 136
Figure B.1 Additional Qualitative Results on SURREAL dataset 140
Figure B.2 Additional Qualitative Results on D-FAUST 141

Figure B.3 Semantic Predictions on D-FAUST 142
Figure B.4 Qualitative Results on D-FAUST 143
Figure B.5 Full Hierarchies 144
Figure B.6 Full Hierarchies 145
Figure B.7 Single Image 3D Reconstruction on ShapeNet 146
Figure B.8 Single Image 3D Reconstruction on ShapeNet 147
Figure B.9 Single Image 3D Reconstruction on D-FAUST 149
Figure B.10 Single Image 3D Reconstruction on D-FAUST 150
Figure B.11 Single Image 3D Reconstruction on FreiHAND 151
Figure B.12 Single Image 3D Reconstruction on ShapeNet Cars . 152
Figure B.13 Single Image 3D Reconstruction on ShapeNet Airplanes 154
Figure B.14 Single Image 3D Reconstruction on ShapeNet Lamps 156
Figure B.15 Single Image 3D Reconstruction on ShapeNet Chairs 157
Figure B.16 Representation Consistency 158
Figure B.17 Temporal Consistency of Predicted Primitives 159
Figure B.18 Semantic Vertices by SMPL-X 159
Figure B.19 Failure Case Detection and Correction 161
Figure B.20 Object Suggestion 162
Figure B.21 Scene Completion 163
Figure B.22 Object Placement 164
Figure B.23 Absolute Difference between Object Co-occurrence in Bedrooms 166
Figure B.24 Absolute Difference between Object Co-occurrence in Living Rooms 167
Figure B.25 Absolute Difference between Object Co-occurrence in Dining Rooms 167
Figure B.26 Absolute Difference between Object Co-occurrence in Libraries 168
Figure B.27 Difference of Per-Object Frequencies 169
Figure B.28 Qualitative Scene Synthesis Results on Bedrooms . . 170
Figure B.29 Qualitative Scene Synthesis Results on Living Rooms 171
Figure B.30 Qualitative Scene Synthesis Results on Dining Rooms 172
Figure B.31 Qualitative Scene Synthesis Results on Libraries . . . 173

LIST OF TABLES

Table 2.1 Quantiative Evaluation on Shapenet objects 23
Table 3.1 Ablation Study of INN 50
Table 3.2 Ablation Study on $p_{\boldsymbol{\theta}}(\cdot)$ 52
Table 3.3 Ablation Study on Loss Terms 53
Table 4.1 Single Image Reconstruction on ShapeNet 82
Table 4.2 Single Image Reconstruction on D-FAUST 85
Table 4.3 Sampling strategy 90
Table 4.4 Number of sampled points. 90
Table 4.5 Sampling Strategy 90
Table 4.6 Impact of Proximity Loss 92
Table 5.1 Scene Synthesis Quantitative Comparison on 3D-FRONT 107
Table 5.2 Generation Time Comparison 109
Table 5.3 Network Parameters Comparison 109
Table 5.4 Perceptual Study on Scene Synthesis 111
Table 5.5 Ablation Study on the Number of Logistic Distributions 112
Table 5.6 Ablation Study on the Layout Encoder Architecture . 113
Table 5.7 Ablation Study on Ordering 113
Table B.1 Single Image 3D Reconstruction on D-FAUST 148
Table B.2 Single Image 3D Reconstruction on ShapeNet Cars . 153
Table B.3 Single Image 3D Reconstruction on ShapeNet Airplanes 155
Table B.4 Single Image 3D Reconstruction on ShapeNet Lamps 155
Table B.5 Single Image 3D Reconstruction on ShapeNet Chairs 157
Table B.6 Scene Syntethis Quantitative Comparison on 3D-FRONT 165

1

INTRODUCTION

> *If we knew what it was we were doing, it would not be called research, would it?*
>
> — Albert Einstein

Within the last few years, artificial deep neural networks (DNNs) have fundamentally transformed a large number of research fields in computer science [16, 23, 172, 155] and cognitive science [184]. While DNNs have demonstrated super-human performance on a variety of applications such as image classification [102, 48], object detection [63, 156], semantic segmentation [115, 74, 21] etc., they fail to demonstrate similar capabilities on more complex perceptual tasks such 3D scene understanding that require high-level reasoning. An important challenge towards building intelligent agents that robustly perceive and interact with their environment is developing a powerful cognitive model capable of perceiving the world at a *functional level*. Namely, we need to develop representations that can simultaneously encode object-specific information i.e. geometry, appearance, semantic identity, part decomposition etc. as well as reason about relationships between elements in a scene such as i.e. spatial arrangement of objects, functional composition etc.

Early attempts towards this goal focus primarily on recovering the 3D geometry [120, 146, 127, 213, 83, 29] and appearance [122, 76, 215, 200, 157] of objects in a scene. An alternative line of research [12, 147, 191, 141] focuses on compositional representations that capture relationships between objects and object parts. Encapsulating the concept of composition in scene representations can tremendously benefit various perceptual tasks that require interacting with the environment (i.e. decide how to interact with objects), manipulating and editing scenes (i.e. change the characteristics of specific regions of a scene), recognising unknown objects without incorporating prior knowledge, interactively creating 3D content etc.

Existing compositional representations focus primarily on capturing the inherent structure of a 3D object as *semantically consistent part arrangements*. The key motivation behind decomposing objects using a set of parts that have a consistent semantic identity across different object instances is that the recovered parts can be leveraged as priors for associating unknown or partially visible objects with object parts whose functionality is already known. In the early days of computer vision, researchers explored various shape primitives for capturing the part geometries such as 3D polyhedral shapes [164], generalized cylinders [12] and geons [10] for representing 3D parts. More recent works [191, 141, 72, 37] have demonstrated the ability of neural networks to learn part-level geometries using 3D cuboids [191, 130, 226], superquadric surfaces [141, 143], spheres [72], convex solids [37, 25], star domains [92] or more general neural primitives [140].

The goal of this dissertation is to recover compositional representations of objects and scenes. In Part I and II, we address the expressivity of primitive-based representations and propose unsupervised learning-based solutions that accurately capture the 3D object geometry as a set of semantically meaningful part arrangements. In Part III, we focus our attention beyond objects and address the task of scene generation. In particular, we propose a generative model that populates indoor environments by generating plausible object arrangements that faithfully capture the spatial and functional relationships of objects in a room i.e. nightstand next to a bed, chairs around table etc.

Part I: Expressive Part-based Representations: Abstracting complex 3D shapes with parsimonious part-based representations has been a long standing goal in computer vision. Tulsiani et al. [191] were among the first to revisit shape primitives in the context of deep learning. In particular, they propose inferring shape abstractions by learning to assemble cuboidal primitives. Given a voxelized 3D object shape as input, a 3D convolutional neural network is used to regress the shape and transformation parameters of a small set of 3D cuboids. The assembled shape is compared against the ground truth mesh based on a distance-field loss. Concurrently, Zou et al. [226] propose a generative recurrent neural network (RNN) that synthesizes multiple plausible shapes based on a set of 3D primitives. Similarly, Niu et al. [130] employ an RNN that iteratively predicts 3D cuboids as well as their symmetry and connectivity relationships from RGB images.

In Chapter 2, we present a learning-based solution to the scene abstraction problem which goes beyond the 3D cuboidal primitives by exploiting superquadrics as atomic elements. We demonstrate that superquadrics lead to more expressive 3D scene parses while being easier to learn than 3D cuboids. Moreover, we provide an analytical solution to the Chamfer loss which avoids the need for computational expensive reinforcement learning or iterative prediction. Our model learns to parse 3D objects into consistent superquadric representations without any part-level supervision.

We perform extensive experiments on various ShapeNet [19] categories as well as the SURREAL human body dataset [194] and demonstrate the flexibility of our model in capturing fine details and complex poses that could not have been represented using cuboids.

While superquadric surfaces and the more general convex solids [37, 25] allow for a richer shape vocabulary than 3D cuboids they still have limited representation capacity, thus fail to accurately reconstruct complex 3D geometries using a small number of primitives/parts. To address this limitation, in Chapter 3, we introduce Neural Parts, a 3D primitive representation that defines primitives using an Invertible Neural Network (INN) which implements homeomorphic mappings between a sphere and the target object. The INN allows us to compute the inverse mapping of the homeomorphism, which in turn, enables the efficient computation of both the implicit surface function of a primitive and its mesh, without any additional post-processing. Our model learns to parse 3D objects into semantically consistent part arrangements without any part-level supervision. We conduct evaluations on ShapeNet [19], D-FAUST [13] and FreiHAND [225] and demonstrate that our primitives simultaneously achieve geometrically accurate and semantically meaningful reconstructions using an order of magnitude fewer primitives than state-of-the-art shape abstraction methods.

Part II: Structure-Aware Part-based Representations: Structure-aware representations go beyond part-level geometry and seek to recover the geometry of each part together with the latent structure of the object. Li et al. [106] propose to represent 3D shapes using a symmetry hierarchy [203] and train a recursive neural network to predict its hierarchical structure. Their network learns a hierarchical organization of bounding boxes and then fills them with voxelized parts. More recently, Mo et al. introduce StructureNet [124] a supervised method which leverages a graph neural network to represent shapes as n-ary graphs. Similarly, Hu et al. [79] propose a supervised model that recovers the 3D structure of a cable-stayed bridge as a binary parsing tree. Notably, the majority of existing structure-aware representations require supervision in terms of part relations as well as primitive annotations. However, this kind of annotation is hardly ever available.

We address this challenge, in Chapter 4, where we introduce a structure-aware representation that jointly recovers the 3D object geometry as a set of primitives as well as their latent hierarchical structure without any part-level supervision. In particular, we employ a neural network that learns to recursively partition an object into its constituent parts by building a latent space that encodes both the part-level hierarchy and the part geometries. The predicted hierarchical decomposition is represented as an unbalanced binary tree of primitives, where simple parts are represented with fewer primitives and more complex parts are modelled with more components. Our experiments on the ShapeNet [19] and D-FAUST [13] datasets demonstrate that considering the organization of parts indeed facilitates reasoning about 3D geometry.

Part III: Structure-Aware Object-based Representations for Scene Synthesis: In the last part of this thesis, we shift our attention from objects and focus on scenes and in particular on the task of generating realistic synthetic scenes by plausibly arranging objects within the scene's boundaries. In Chapter 5, we introduce ATISS, a network architecture for controllable indoor scene synthesis [139]. Given a room type (e.g. bedroom, living room) and its shape, our model generates meaningful object arrangements by sequentially placing objects in a permutation-invariant fashion. We train ATISS to maximize the log-likelihood of all possible permutations of object arrangements in a collection of training scenes, labeled only with object classes and 3D bounding boxes. In contrast to prior work that requires supervision either in the form of relation graphs [197, 218, 117] or scene hierarchies [109] for capturing the object relationships, ATISS infers functional and spatial relations between objects directly from data. Evaluations on four room types in the 3D-FRONT dataset [54] demonstrate that our model consistently generates plausible room layouts that are more realistic than existing methods.

Part I

Expressive Part-based Representations

2

LEARNING 3D SHAPE PARSING BEYOND CUBOIDS

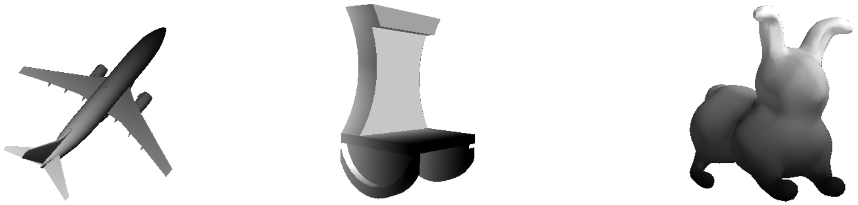
> *Everything in nature is formed upon the sphere, the cone and the cylinder. One must learn to paint these simple figures and then one can do all that he may wish.*
>
> — Paul Cézane

Evolution has developed a remarkable visual system that allows humans to robustly perceive their 3D environment. It has long been hypothesized [10] that the human visual system processes the vast amount of raw visual input into compact parsimonious representations, where complex objects are decomposed into a small number of shape primitives that can each be represented using low-dimensional descriptions. Indeed, experiments show that humans can understand complex scenes from renderings of simple shape primitives such as cuboids or geons [11].

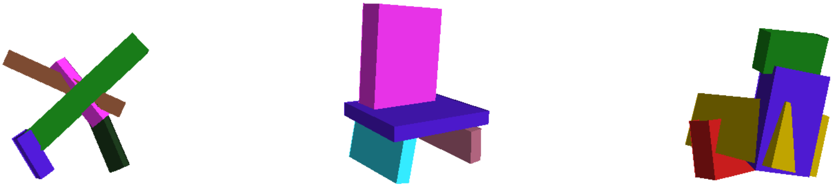
Likewise, machines would tremendously benefit from being able to parse 3D data into compact low-dimensional representations. Such representations would provide useful cues for recognition, detection, shape manipulation and physical reasoning such as path planning and grasping. In the early days of computer vision, researchers explored shape primitives such as 3D polyhedral shapes [164], generalized cylinders [12], geons [10] and superquadrics [147]. However, it proved very difficult to extract such representations from images due to the lack of computation power and data at the time. Thus, the research community shifted their focus away from the shape primitive paradigm.

In the last decade, major breakthroughs in shape extraction were due to deep neural networks coupled with the abundance of visual data. Recent works focus on learning 3D reconstruction using 2.5D [73, 142, 80, 212], volumetric [34, 62, 207, 82, 71, 162], mesh [68, 113] and point cloud [53, 151] representations. However, none of the above are sufficiently parsimonious or interpretable to allow for higher-level 3D scene understanding as required by intelligent systems.

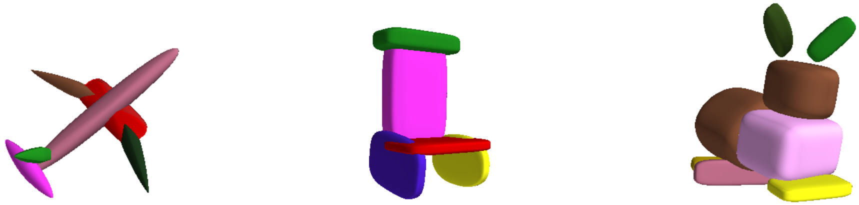
Recently, shape primitives have been revisited in the context of deep learning. In particular, [191, 226, 130] have demonstrated that deep neural networks enable to reliably extract 3D cuboids from meshes and RGB im-



(a) Input Mesh



(b) Cuboids[191]



(c) Superquadric Surfaces(Ours)

Figure 2.1: **3D Shape Parsing.** We consider the problem of learning to parse unstructured 3D data (e.g., meshes or point clouds) into compact part-based representations. Prior work [191, 226, 130] has considered cuboid representations (b) which capture the overall object structure, but lack expressiveness. In this work, we propose an unsupervised model for superquadrics (c), which allows us to capture details such as the body of the airplane and the ears of the rabbit.

ages. Inspired by these works, we propose a novel deep neural network that efficiently extracts parsimonious 3D representations in an unsupervised fashion, conditioned on a 3D shape or 2D image as input. We note that 3D cuboid representations used in prior works [191, 226, 130] are not sufficiently expressive to model many natural and man-made shapes as illustrated in Fig. 2.1, and as a result, cuboid-based representations typically require a large number of primitives to accurately represent common shapes. To address this limitation, we propose to utilize superquadrics, which have been successfully used in computer graphics [9] and classical computer vision [147, 178, 185]. Superquadrics are able to represent a diverse class of shapes such as cylinders, spheres, cuboids, ellipsoids *in a single continuous parameter space* (see Fig. 2.1+2.2). Moreover, their continuous parametrization is particularly amenable to deep learning, as their shape is smooth and varies continuously with their parameters. This allows for faster optimization, and hence faster and more stable training as evidenced by our experiments.

Second, we provide an analytical closed-form solution to the Chamfer distance function which can be evaluated in linear time wrt. the number of primitives. This allows us to compute gradients wrt. the model parameters using standard error backpropagation [167] without resorting to computational expensive reinforcement learning techniques as required by prior work [191]. We consequently mitigate the need for designing an auxiliary reward function. Instead, we formulate a simple parsimony loss to favor configurations with a small number of primitives. We demonstrate the strengths of our model by learning to parse 3D shapes from the ShapeNet [19] and the SURREAL [194]. We observe that our model converges faster than [191] and leads to more accurate reconstructions. The code to reproduce the experiments presented in this chapter can be found at https://github.com/paschalidoud/superquadric_parsing.

2.1 RELATED WORK

In this section, we discuss the most relevant work on deep learning-based 3D shape modeling approaches and review the origins of superquadric representations.

3D Shape Reconstruction: The simplest representation for 3D reconstruction from one or more images are 2.5D depth maps as they can be inferred using standard 2D convolutional neural networks [73, 142, 82, 212]. Since depth maps are view-based, these methods require additional post-processing algorithms to fuse information from multiple viewpoints in order to capture the entire object geometry. As opposed to depth maps, volumetric representations [62, 71, 34, 162, 183] naturally capture the entire 3D shape. While, hierarchical 3D data structures such as octrees accelerate 3D convolutions, the high memory requirements remain a limitation of existing volumetric methods. An alternative line of work [53, 152] focuses on learning to reconstruct 3D point sets. A natural limitation of these approaches is the lack of surface connectivity in the representation. To address this limitation, [113, 68, 199, 159] proposed to directly learn 3D meshes. While mesh-based representations yield smooth reconstructions, they are prone to generating self-intersecting meshes and tend to have high memory requirements. To mitigate this, more recently, researchers revisited the implicit representations and propose to represent the 3D shape as the level-set of a distance function implemented as a neural network [120, 136, 27]. While some of the aforementioned models are able to capture fine surface details, none of them lends itself to parsimonious, semantic interpretations. In this work, we utilize superquadrics which provide a concise and yet accurate representation with significantly less parameters.

Constructive Solid Geometry: Towards the goal of concise representations, researchers exploited constructive solid geometry (CSG) [104] for shape modeling [173, 51]. Sharma et al. [173] leverage an encoder-decoder architecture to generate a sequence of simple boolean operations to act on a set of primitives that can be either squares, circles or triangles. In a similar line of work, Ellis et al. [51] learn a programmatic representation of a hand-written drawing, by first extracting simple primitives, such as lines, circles and rectangles and a set of drawing commands that is used to synthesize a LaTeX program. In contrast to [173, 51], our goal is not to obtain accurate 3D geometry by iteratively applying boolean operations on shapes. Instead, we aim to decompose the depicted object into a parsimonious interpretable representation where each part has a semantic meaning associated with it. Besides, our method does not suffer from ambiguities of an iterative construction process, where different executions lead to the same result.

Learning-based Scene Parsing: Recently, shape primitives have been revisited in the context of deep learning [191, 226, 130]. Niu et al. [130] propose to use a recurrent neural network (RNN) to iteratively predict cuboid primitives as well as symmetry relationships from RGB images. They first train an encoder which encodes the input image and its segmentation into a 80-dimensional latent code. Starting from this root feature, they iteratively decode the structure into cuboids, splitting nodes based on adjacency and symmetry relationships. In related work, Zou et al. [226] utilize LSTMs in combination with mixture density networks to generate cuboid representations from depth maps encoded by a 32-dimensional feature vector. However, both works [130, 226] require supervision in terms of the primitive parameters as well as the sequence of predictions. This supervision must either be provided by manual annotation or using greedy heuristics as in [130, 226].

In contrast, our approach is unsupervised and does not suffer from ambiguities caused by different possible prediction sequences that lead to the same cuboid assembly. Furthermore, [130, 226] exploit simple cuboid representations which do not capture more complex shapes that are common in natural and man-made scenes (e.g., curved objects, spheres). In this work, we propose to use superquadrics [9] which yield a more diverse shape vocabulary and hence lead to more expressive scene abstractions as illustrated in Fig. 2.1.

Recently, Tulsiani et al. [191] proposed a method for 3D shape abstraction using a non-iterative approach which does not require supervision. In particular, they use a convolutional network architecture for predicting the shape and pose parameters of 3D cuboids as well as their probability of existence. They demonstrate that learning shape abstraction from data allows for obtaining consistent parses across different instances in an unsupervised fashion.

In this work, we extend the model of Tulsiani et al. [191] in the following directions. First, we utilize superquadrics, instead of cuboids, which leads to more accurate scene abstractions. Second, we demonstrate that the bi-directional Chamfer distance is tractable and doesn't require reinforcement learning [205] or specification of rewards [191]. In particular, we show that there exists an analytical closed-form solution which can be evaluated in linear time. This allows us to compute gradients wrt. the model parameters using standard error propagation [167] which facilitates learning. In addition, we add a new simple parsimony loss to favor configurations with a small number of primitives.

Superquadric Surfaces: Superquadrics are a parametric family of surfaces that can be used to describe cubes, cylinders, spheres, octahedra, ellipsoids etc. [9]. In contrast to geons [10], superquadric surfaces can be described using a fairly simple parameterization. In contrast to generalized cylinders [10], superquadrics are able to represent a larger variety of shapes. See Fig. 2.2 for an illustration of the shape space.

In 1986, Pentland introduced superquadrics to the computer vision community [147]. Solina et al. [178] formulated the task of fitting superquadrics to a point cloud as a least-squares minimization problem. Chevalier et al. [28] followed a two-stage approach, where the point cloud is first partitioned into regions and then each region is fit with a superquadric. For a thorough survey on superquadrics we refer reader to [81, 177] for details.

In contrast to these classical works on superquadric fitting using non-linear least squares, we present the first approach to train a deep network to predict superquadrics directly from 2D or 3D inputs. This allows our network to distill statistical dependencies wrt. the arrangement and geometry of the primitives from data, leading to semantically meaningful

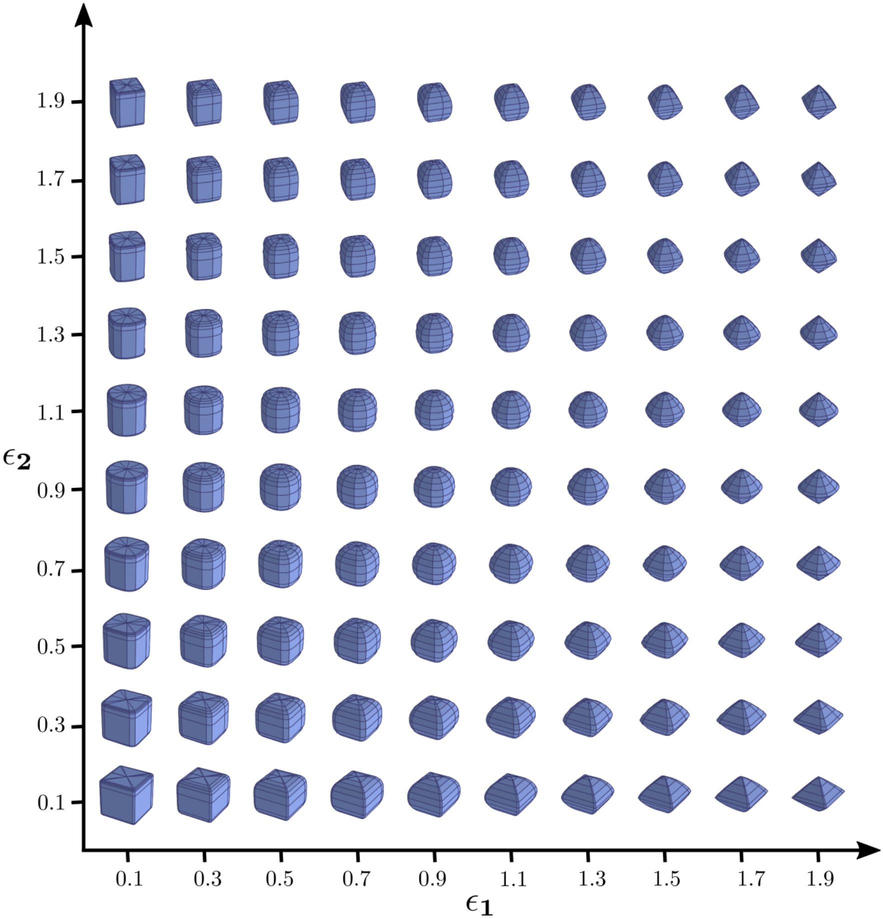


Figure 2.2: **Superquadric Shape Space.** Superquadrics are a parametric family of surfaces that can be used to describe cubes, cylinders, spheres, octahedral ellipsoids, etc. [9]. This figure visualizes superquadrics when varying the shape parameters ϵ_1 and ϵ_2, while keeping the size parameters α_1, α_2 and α_3 constant.

parts at inference time. Towards this goal, we utilize a convolutional network that predicts superquadric poses and attributes, and develop a novel loss function that allow us to train this network efficiently from data. Our model is able to directly learn superquadric surfaces from an unordered 3D point cloud without any supervision on the primitive parameters nor a 3D segmentation as input.

2.2 METHOD OVERVIEW

Given an input **I** (e.g., image, volume, point cloud) and an oriented point cloud $\mathcal{X}$ of the target object, our goal is to estimate the parameters θ of a neural network $\phi_\theta(\mathbf{I})$ that predicts a set of M primitives that best describe the target object. Every primitive is fully described by a set of parameters that define its shape, size and its position and orientation in the 3D space. Additional details for the parameterization of the superquadric representation are provided in Sec. 2.2.3.

Since not all objects and scenes require the same number of primitives, we enable our model to predict a variable number of primitives, hence allowing it to decide whether a primitive should be part of the assembled object or not. To achieve this, we follow [191] and associate every primitive with a binary random variable $z_m \in \{0, 1\}$ which follows a Bernoulli distribution $p(z_m) = \gamma_m^{z_m}(1 - \gamma_m)^{1-z_m}$ with parameter γ_m. The random variable z_m indicates whether the m^{th} primitive is part of the scene $(z_m = 1)$ or not $(z_m = 0)$. We refer to these variables as *existence variables* and denote the set of all existence variables as $\mathbf{z} = \{z_1, \dots, z_M\}$. Our goal is to learn a neural network $\phi_\theta(\mathbf{I})$ which maps an input **I** to a primitive representation $\mathcal{P} = \{(\lambda_m, \gamma_m)\}_{m=1}^M$ that comprises the primitive parameters λ_m and the existence probability γ_m for M primitives. Note that M is only an upper bound on the number of predicted primitives. The final primitive representation is obtained by sampling the existence of each primitive, $z_m \sim \text{Bernoulli}(\gamma_m)$.

One of the key challenges when training such models is related to the lack of direct supervision in the form of primitive annotations. However, despite the absence of supervision, one can still measure the discrepancy between the predicted object and the target object. Towards this goal, we formulate a bi-directional reconstruction objective $\mathcal{L}_D(\mathcal{P}, \mathcal{X})$ and incorporate a Minimum Description Length (MDL) prior $\mathcal{L}_\gamma(\mathcal{P})$, which favors parsimony, i.e. a small number of primitives. Our overall loss function is given as:

$$\mathcal{L}(\mathcal{P}, \mathcal{X}) = \mathcal{L}_D(\mathcal{P}, \mathcal{X}) + \mathcal{L}_\gamma(\mathcal{P}) \tag{2.1}$$

2.2.1 *Reconstruction Loss*

The reconstruction loss measures the discrepancy between the predicted shape and the target shape. While we experimented with the truncated bi-directional loss of Tulsiani et al. [191], we empirically found that the standard Chamfer distance [53] works better in practice and results in less local minima. An empirical analysis on this can be found in Sec. A.1.2. Thus, we use the Chamfer distance in our experiments

$$\mathcal{L}_D(\mathcal{P},\mathcal{X}) = \mathcal{L}_{P\to X}(\mathcal{P},\mathcal{X}) + \mathcal{L}_{X\to P}(\mathcal{X},\mathcal{P}) \tag{2.2}$$

where $\mathcal{L}_{P\to X}$ measures the distance from the predicted primitives $\mathcal{P}$ to the point cloud $\mathcal{X}$ and $\mathcal{L}_{X\to P}$ measures the distance from the point cloud $\mathcal{X}$ to the primitives $\mathcal{P}$. We weight the two distance measures in (2.2) with 1.2 and 0.8, respectively, which empirically led to good results.

Primitive-to-Pointcloud: We represent the target point cloud as a set of 3D points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$. Similarly, we approximate the continuous surface of primitive m by a set of points $\mathcal{Y}_m = \{\mathbf{y}_k^m\}_{k=1}^K$. Details of our sampling strategy are provided in Sec. 2.4. This discretization allows us to express the distance between a superquadric and the target point cloud in a convenient form. In particular, for each point on the primitive $\mathbf{y}_k^m$, we compute its closest point on the target point cloud $\mathbf{x}_i$, and average this distance across all points in $\mathcal{Y}_m$ as follows:

$$\mathcal{L}_{P\to X}^m(\mathcal{P},\mathcal{X}) = \frac{1}{K}\sum_{k=1}^{K} \Delta_k^m \tag{2.3}$$

where

$$\Delta_k^m = \min_{i=1,..,N} \|\mathcal{T}_m(\mathbf{x}_i) - \mathbf{y}_k^m\|_2 \tag{2.4}$$

denotes the minimal distance from the k'th point $\mathbf{y}_k^m$ on the m'th primitive to the target point cloud $\mathcal{X}$. Here, $\mathcal{T}_m(\mathbf{x}) = \mathbf{R}(\lambda_m)\,\mathbf{x} + \mathbf{t}(\lambda_m)$ is a function that transforms a 3D point $\mathbf{x}_i$ in world coordinates into the local coordinate system of the m^{th} primitive. Note that both $\mathbf{R}$ and $\mathbf{t}$ depend on λ_m and are hence estimated by our network.

By taking the expectation wrt. the existence variables $\mathbf{z}$ and assuming independence of the existence variables: $p(\mathbf{z}) = \prod_m p(z_m)$, we obtain the joint loss over all primitives as

$$\begin{aligned}\mathcal{L}_{P\to X}(\mathcal{P},\mathcal{X}) &= \mathbb{E}_{p(\mathbf{z})}\left[\sum_{m=1}^{M} \mathcal{L}_{P\to X}^m(\mathcal{P},\mathcal{X})\right] \\ &= \sum_{m=1}^{M} \gamma_m\, \mathcal{L}_{P\to X}^m(\mathcal{P},\mathcal{X})\end{aligned} \tag{2.5}$$

Note that this loss encourages the predicted primitives to stay close to the target point cloud.

Pointcloud-to-Primitive: While $\mathcal{L}_{P \to X}$ measures the distance from the primitives to the point cloud, $\mathcal{L}_{X \to P}$ measures the distance from the point cloud to the primitives to ensure that each observation is explained by at least one primitive. We start by defining Δ_i^m as the minimal distance from point $\mathbf{x}_i$ to the surface of the m'th primitive:

$$\Delta_i^m = \min_{k=1,..,K} \|\mathcal{T}_m(\mathbf{x}_i) - \mathbf{y}_k^m\|_2 \tag{2.6}$$

Note that in contrast to (2.4), we minimize over the K points from the estimated primitive. Similarly to (2.5), we take the expectation of Δ_i^m over $p(\mathbf{z})$. In contrast to (2.5), we sum over each point in the target point cloud $\mathcal{X}$ and retrieve the distance to the closest primitive m that exists ($z_m = 1$):

$$\mathcal{L}_{X \to P}(\mathcal{X}, \mathcal{P}) = \mathbb{E}_{p(\mathbf{z})} \left[\sum_{\mathbf{x}_i \in \mathcal{X}} \min_{m|z_m=1} \Delta_i^m \right] \tag{2.7}$$

Note that naïve computation of Eq. 2.7 becomes very slow for a large number of primitives M as it requires evaluating the quantity inside the expectation 2^M times. In this work, we propose a novel approach to simplify this computation that results in a linear number of evaluations. Without loss of generality, let us assume that the Δ_i^m's are sorted in ascending order:

$$\Delta_i^1 \leq \Delta_i^2 \leq \cdots \leq \Delta_i^M \tag{2.8}$$

Assuming this ordering, we can state the following: if the first primitive exists, the first primitive will be the one closest to point $\mathbf{x}_i$ of the target point, if the first primitive does not exist and the second does, then the second primitive is closest to point $\mathbf{x}_i$ and so on and so forth. More formally, this property can be stated as follows:

$$\min_{m|z_m=1} \Delta_i^m = \begin{cases} \Delta_i^1, & \text{if } z_1 = 1 \\ \Delta_i^2, & \text{if } z_1 = 0, z_2 = 1 \\ \vdots & \\ \Delta_i^M, & \text{if } z_m = 0, \dots, z_M = 1 \end{cases} \tag{2.9}$$

This allows us to simplify Eq. 2.7 as follows

$$\mathcal{L}_{X \to P}(\mathcal{X}, \mathcal{P}) = \sum_{\mathbf{x}_i \in \mathcal{X}} \sum_{m=1}^{M} \Delta_i^m \, \gamma_m \prod_{\bar{m}=1}^{m-1} (1 - \gamma_{\bar{m}}) \tag{2.10}$$

where $\gamma_{\bar{m}}$ is a shorthand notation which denotes the existence probability of a primitive closer than primitive m. Note that this function requires only M, instead of 2^M, evaluations of the function Δ_i^m. For a detailed derivation of (2.10), we refer the reader to Sec. A.1.1.

2.2.2 *Parsimony Loss*

Despite the bidirectional loss formulation above, our model suffers from the trivial solution $\mathcal{L}_D(\mathcal{P}, \mathcal{X}) = 0$ which is attained for $\gamma_1 = \cdots = \gamma_m = 0$. Moreover, multiple primitives with identical parameters yield the same loss function as a single primitive by dispersing their existence probability. We thus introduce a regularizer loss on the existence probabilities γ which alleviates both problems:

$$\mathcal{L}_\gamma(\mathcal{P}) = \max\left(\alpha - \alpha \sum_{m=1}^{M} \gamma_m, 0\right) + \beta \sqrt{\sum_{m=1}^{M} \gamma_m} \tag{2.11}$$

The first term of (2.11) makes sure that the aggregate existence probability over all primitives is at least one (i.e., we expect at least one primitive to be present) and the second term enforces a parsimonious scene parse by exploiting a loss function sub-linear in $\sum_m \gamma_m$ which encourages sparsity. α and β are weighting factors which are set to 1.0 and 10^{-3} respectively.

2.2.3 *Superquadric Parametrization*

Having specified our network and the loss function, we now provide details about the superquadric representation and its parameterization λ. Note that, in this section, we omit the primitive index m for clarity. Superquadrics define a family of parametric surfaces that can be fully described by a set of 11 parameters [9]. The explicit superquadric equation defines the surface vector $\mathbf{r}$ as

$$\mathbf{r}(\eta, \omega) = \begin{bmatrix} \alpha_1 \cos^{\epsilon_1} \eta \cos^{\epsilon_2} \omega \\ \alpha_2 \cos^{\epsilon_1} \eta \sin^{\epsilon_2} \omega \\ \alpha_3 \sin^{\epsilon_1} \eta \end{bmatrix} \quad \begin{array}{c} -\pi/2 \leq \eta \leq \pi/2 \\ -\pi \leq \omega \leq \pi \end{array} \tag{2.12}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]$ determine the size and $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2]$ determine the global shape of the superquadric.

Following common practice [195], we bound the values ϵ_1 and ϵ_2 to the range $[0.1, 1.9]$ so as to prevent non-convex shapes which are less

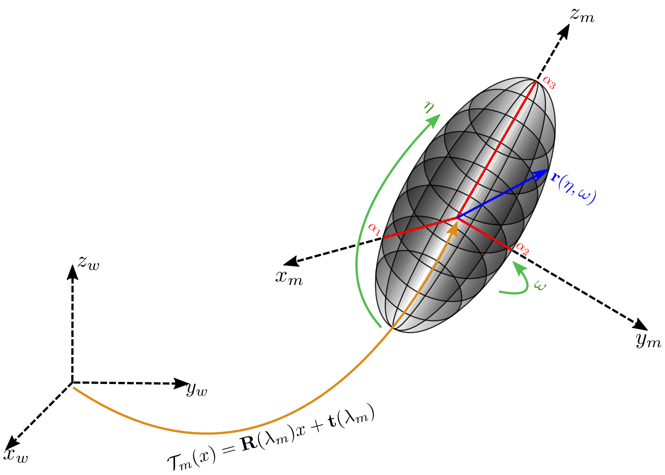

z_m
α_3
η
$\mathbf{r}(\eta, \omega)$
α_1
x_m
α_2
ω
y_m
z_w
y_w
x_w
$\mathcal{T}_m(x) = \mathbf{R}(\lambda_m)x + \mathbf{t}(\lambda_m)$

Figure 2.3: **Explicit Superquadric Equation**. A $3D$ vector $\mathbf{r}(\eta, \omega)$ defines a closed surface in space as η (latitude angle) and ω (longitude angle) change in the given intervals (2.12). The rigid body transformation $\mathcal{T}_m(x)$ maps a point from the world coordinate system to the local coordinate system of the m^{th} primitive.

likely to occur in practice. Eq. 2.12 produces a superquadric in a canonical pose. In order to allow any position and orientation, we augment the primitive parameter λ with an additional rigid body motion represented by a translation vector $\mathbf{t} = [t_x, t_y, t_z]$ and a quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]$ which determine the coordinate system transformation $\mathcal{T}(\mathbf{x}) = \mathbf{R}(\lambda)\,\mathbf{x} + \mathbf{t}(\lambda)$ above. This transformation as well as the angles η, ω and the scale parameters $\alpha_1, \alpha_2, \alpha_3$ are illustrated in Fig. 2.3.

2.3 EXPERIMENTAL EVALUATION

In this section, we present a set of experiments to evaluate the performance of our network in terms of parsing an input 3D shape into a set of superquadric surfaces.

Datasets: We provide results on two 3D datasets. First, we use the *aeroplane*, *chair* and *animals* categories from ShapeNet [19]. Following [191], we train one model per object category using a voxelized binary occupancy grid of size $32 \times 32 \times 32$ as input. Second, we use the SURREAL dataset from Varol et al. [194] which comprises humans in various poses (e.g., standing,

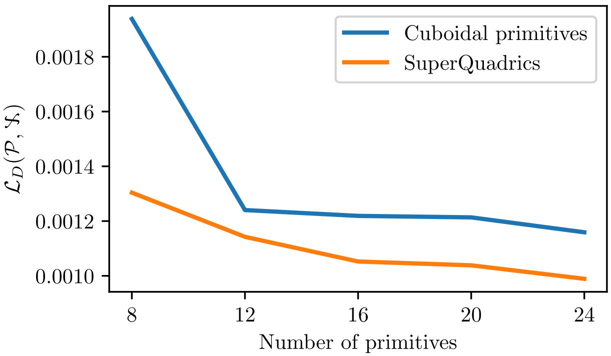


Figure 2.4: **Reconstruction Loss wrt. #Primitives.** We illustrate the reconstruction loss on the test set of the ShapeNet *animal* category for a different number of primitives. Superquadrics (orange) consistently outperform cuboid primitives (blue) due to their diverse shape vocabulary that allows them to better capture fine details of the input shapes.

walking, sitting). Using the SMPL model [116], we rendered 5000 meshes, from which 4500 are used for training and 500 for testing.

Baselines: Most related to ours is the cuboid parsing approach of Tulsiani et al. [191]. Other approaches to cuboid-based scene parsing [130, 226] require ground-truth shape annotations and thus cannot be fairly compared to unsupervised techniques. We thus compare to Tulsiani et al. [191], using their publicly available code[1].

2.3.1 *Superquadrics vs. Cuboids*

We first compare the modeling accuracy of superquadric surfaces wrt. cuboidal shapes which have been extensively used in related work [191, 226, 130]. Towards this goal, we fit *animal* shapes from ShapeNet by optimizing the distance loss function in (2.2) while varying the maximum number of allowed primitives M. To ensure a fair comparison, we use the proposed model for both cases. Note that this is trivially possible as cuboids are a special case of superquadrics. To minimize the effects of network initialization and local minima in the optimization, we repeat the experiment three times with random initializations and visualize the average loss in Fig. 2.4.

---

[1] https://github.com/shubhtuls/volumetricPrimitives

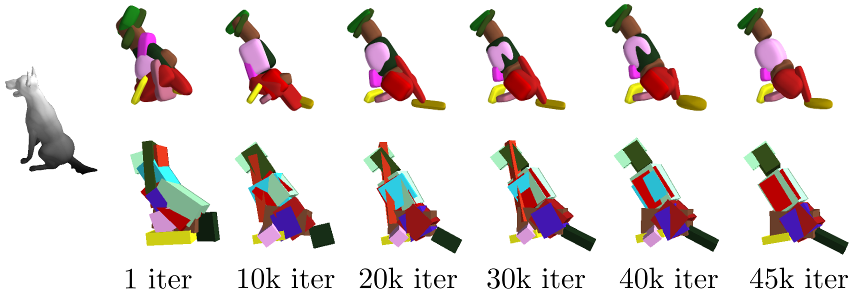


Figure 2.5: **Training Evolution.** We visualize the qualitative evolution of superquadrics (top) and cuboids (bottom) during training. Superquadrics converge faster to more accurate representations, whereas cuboids cannot capture details such as the open mouth of the dog, even after convergence.

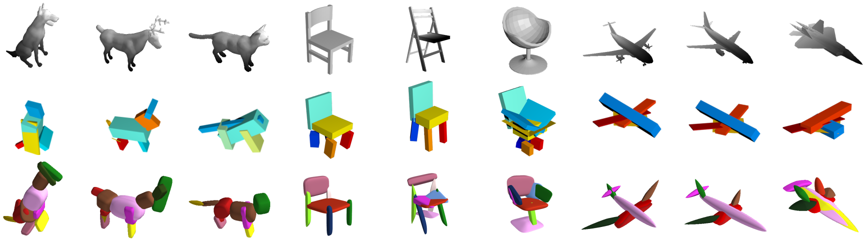


Figure 2.6: **Qualitative Results on ShapeNet dataset.** We visualize predictions for the object categories *animals*, *aeroplane* and *chairs* from the ShapeNet dataset. The top row illustrates the ground-truth meshes from every object. The middle row depicts the corresponding predictions using the cuboidal primitives estimated by [191]. The bottom row shows the corresponding predictions using our learned superquadric surfaces. Similarly to [191], we observe that the predicted primitive representations are consistent across instances. For example, the primitive depicted in green describes the right wing of the aeroplane, while for the animals class, the yellow primitive describes the front legs of the animal.

The results show that for any given number of primitives, superquadrics consistently achieve a lower loss, and hence higher modeling fidelity.

We further visualize the qualitative evolution of the network during training in Fig. 2.5. This figure demonstrates that compared to cuboids, superquadrics better model the object shape, and more importantly that the network is able to converge faster.

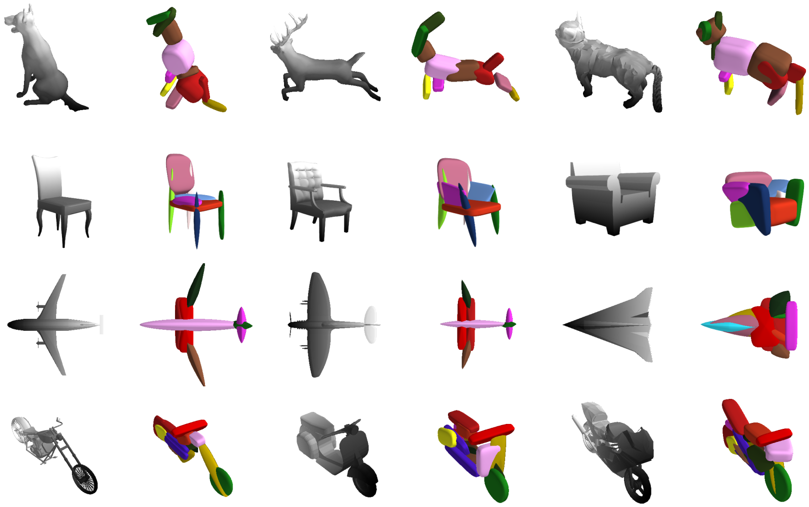


Figure 2.7: **Attention to Details.** Superquadrics allow for modeling fine details such as the tails and ears of animals as well as the wings and the body of the airplanes and wheels of the motorbikes which are hard to capture using cuboids.

2.3.2 *Results on ShapeNet dataset*

We evaluate the quality of the predicted primitives using our reconstruction loss from (2.2) on the ShapeNet dataset and compare to the cuboidal primitives as estimated by Tulsiani et al. [191]. We associate every primitive with a unique color, thus primitives illustrated with the same color correspond to the same object part. For both approaches we set the maximal number of primitives to $M = 20$. From Fig. 2.6, we observe that our predictions consistently capture both the structure as well as fine details (e.g., body, tails, head), whereas the corresponding cuboidal primitives from [191] focus mainly on the structure of the predicted object.

Fig. 2.7 shows additional results in which our model successfully predicts animals, airplanes and also more complicated motorbike parts. For instance, we observe that our model is able to capture the open mouth of the dog using two superquadrics as shown in Fig. 2.7 (left-most animal in third row). In addition, we notice that our model dynamically allocates a variable number of primitives depending on the complexity of the input shape. For example, the left-most airplane in Fig. 2.6, is modelled with 6 primitives whereas the jetfighter (right-most) that has a more complicated shape is modelled with 9 primitives. This can also be observed for the animal

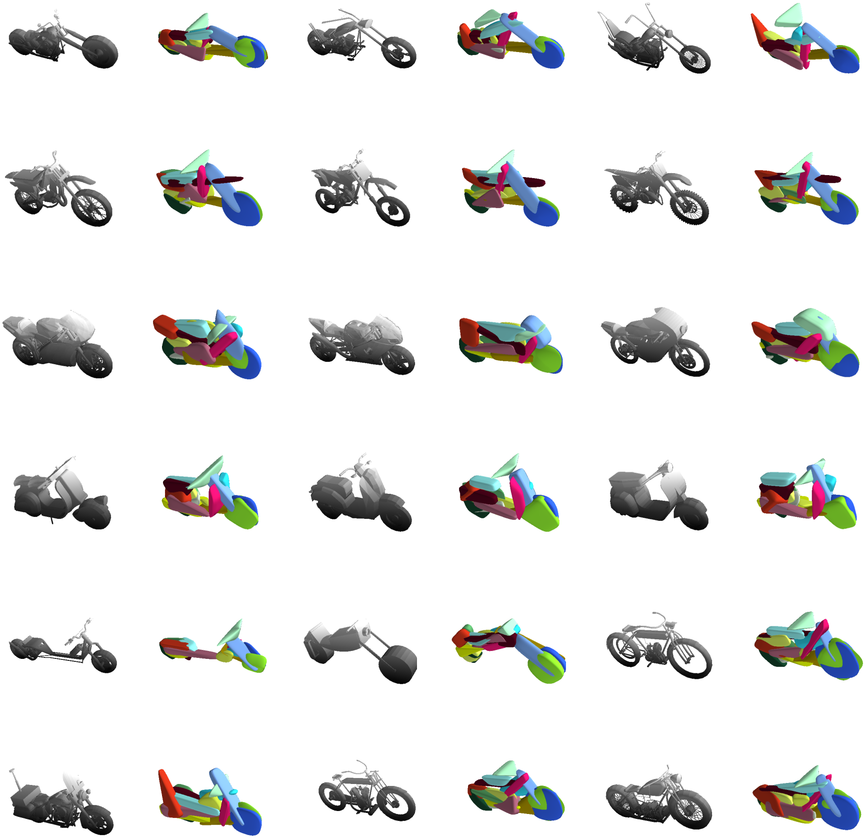


Figure 2.8: **Qualitative Results on Motorbikes.** Our network learns semantic mappings of various object parts of different objects within the same category. Our expressive shape abstractions allow for differentiating between different types of motorbikes (scooter, racing bike, chopper etc.), by sucessfully capturing the shape of various indicative parts such as the wheels or the front fork of the bike.

category, where our model chooses a single primitive for the body of the cat (rightmost animal in Fig. 2.6) while for all the rest it uses two. We remark that our expressive shape abstractions allow for differentiating between different types of objects such as scooter/chopper/racebike (see Fig. 2.8) or airliner/fighter by truthfully capturing the shape of the individual object parts.

Fig. 2.9+2.10 depicts additional predictions on the *animal* and the *chair* object class of the ShapeNet dataset. We observe that for both categories our model consistently captures both the structure and the fine details of the depicted object. Note that chairs that have rounded legs are associated with flattened ellipsoids (Fig. 2.10), this would not have been possible only with cuboids.

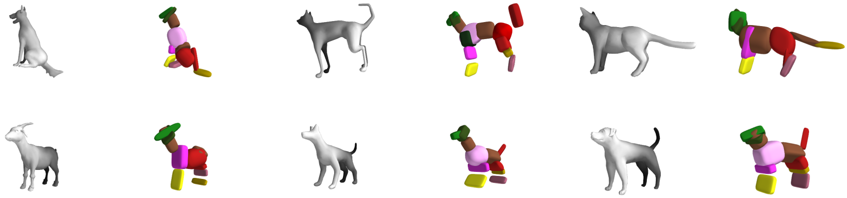


Figure 2.9: **Qualitative Results on Animals.** We visualize the predictions of our network on the *animals* class of the ShapeNet dataset. We remark the consistency across primitives and animal parts as well as the ability of our model to cpature details such as ears and tails of animals that could have not beeen captured using cuboidal primitives

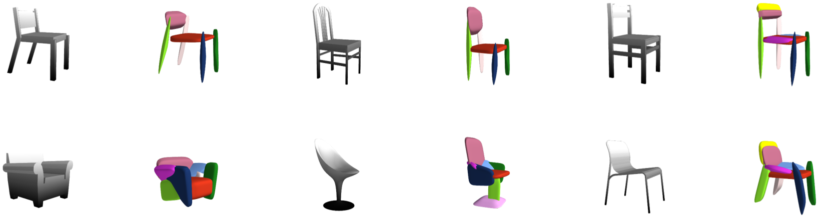


Figure 2.10: **Qualitative Results on Chairs** We visualize the predictions of our network on the *chairs* class of the ShapeNet dataset. We observe the consistency across corespondences between primitives and object parts as well as the ability of our model to capture the shape of rounded parts.

Fig. 2.11 visualizes the training evolution of the predicted superquadrics for three object categories. While initially, the model focuses on the overall structure of the object using mostly blob-shaped superquadrics (ϵ_1 and ϵ_2 close to 1.0), as training progresses it starts attending to details. After convergence, the predicted superquadrics closely match the shape of the corresponding (unknown) object parts.

Finally, we compare the reconstruction quality of our model to [191] wrt. the mean *Chamfer distance* and the mean *Volumetric IoU* of the predicted

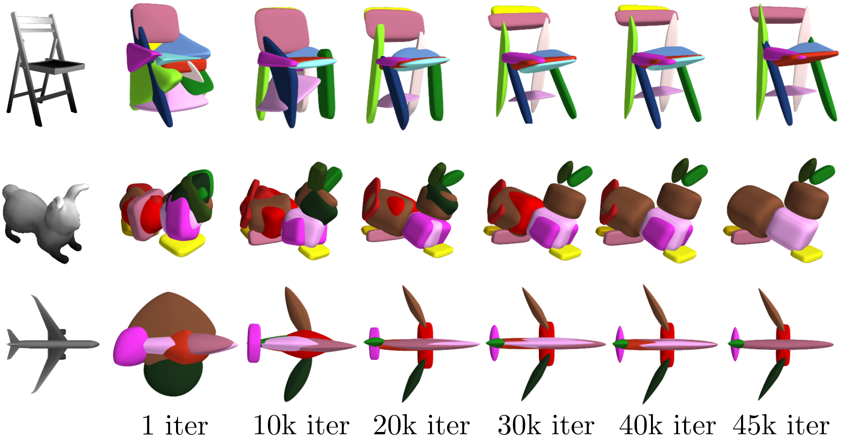


Figure 2.11: **Evolution of Primitives.** We illustrate the evolution of superquadrics during training. Note how our model first focuses on the overall structure of the object and starts to attend to finer details at later iterations.

	Chamfer Distance			Volumetric IoU		
	Chairs	Aeroplanes	Animals	Chairs	Aeroplanes	Animals
Cuboids [191]	0.0121	0.0153	0.0110	0.1288	0.0650	0.3339
Superquadrics	**0.0006**	**0.0003**	**0.0003**	**0.1408**	**0.1808**	**0.7506**

Table 2.1: **Quantitative Evaluation on ShapeNet objects.** We report the mean Chamfer distance (smaller is better) and the mean Volumetric IoU (larger is better) for our model compared to [191].

primitives. Volumetric IoU is defined as the quotient of the volume of the two meshes' intersection and the volume of their union. We obtain unbiased estimates of the volume of the intersection and the union by randomly sampling points from the bounding volume and determining if the points lie inside our outside the ground truth / predicted mesh.

Results are summarized in Tab. 2.1 and we note that our model achieves superior performance in both metrics. Note that in contrast to [191] our primitives are optimized for the Chamfer distance, thus it is expected that our model will outperform [191] wrt. this metric. Moreover, note that cuboids are a special case of superquadrics, thus fitting objects with cuboids is expected to lead to worse results compared to superquadrics.

2.3.3 *Shape Abstraction from a Single RGB Image*

In this section, we use the proposed reconstruction loss to extract shape primitives from RGB images instead of occupancy grids. Towards this goal, we render the ShapeNet models to images, and train an image-based network to minimize the same reconstruction loss also used for our volume-based architecture.

More specifically, we replace the volumetric encoder with a ResNet18 [75], after removing the last fully connected layer. The extracted features are subsequently passed to five independent heads that regress translation $\mathbf{t}$, rotation $\mathbf{q}$, size $\boldsymbol{\alpha}$, shape $\boldsymbol{\epsilon}$ and probability of existence γ for each primitive. During training, we uniformly sample 1000 points, from the surface of the target object, as well as 200 points from the surface of every superquadric. For optimization, we use ADAM [96] with a learning rate of 0.001 and a batch size of 32 for 40k iterations. We observe that our model accurately captures shape primitives even from a single RGB image as input.

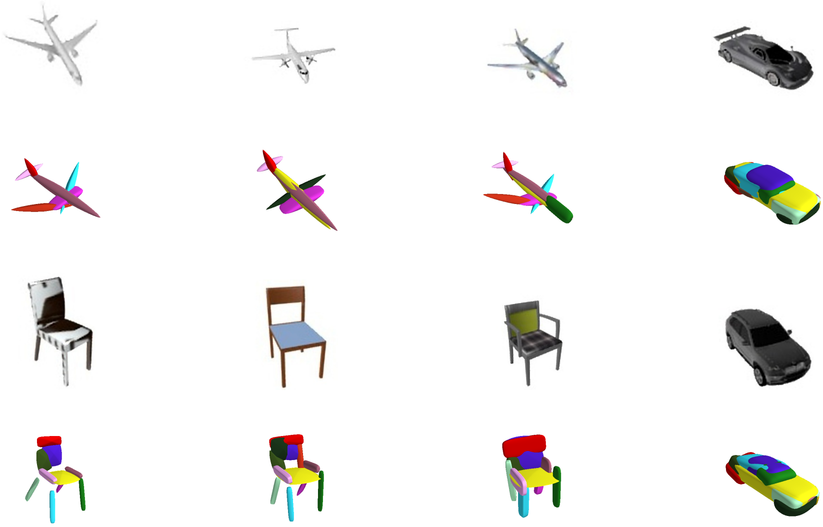


Figure 2.12: **Single Image 3D Reconstruction on ShapeNet objects.** We visualize predictions for various ShapeNet object categories using a single RGB image as input to our model.

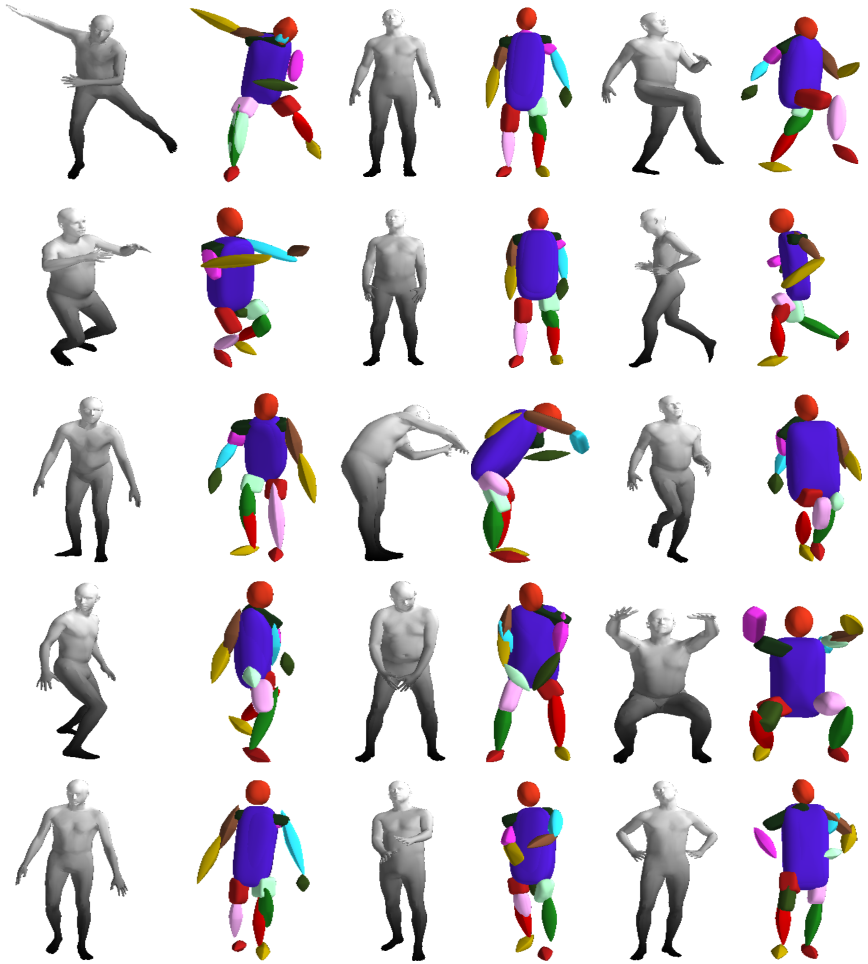
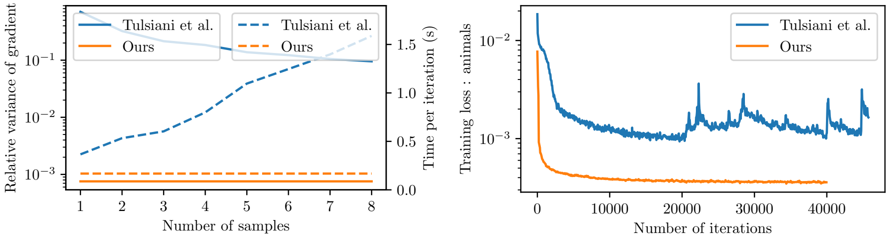


Figure 2.13: **Qualitative Results on SURREAL dataset.** Our network learns semantic mappings of body parts across different body shapes and articulations. For instance, the network uses the same primitive for the left forearm across instances.

2.3.4 *Results on SURREAL dataset*

In addition to ShapeNet, we also demonstrate results on the SURREAL human body dataset in Fig. 2.13. The benefits of superquadrics over simpler shape abstractions are accentuated in this dataset due to the complicated shapes of the human body. Note that our model successfully captures



(a) **Gradient Variance and Iteration Time.**

(b) **Evolution of Training Loss.**

Figure 2.14: **Analytical Loss Formulation.**Fig. 2.14a depicts the variance of the gradient estimates for γ over 300 iterations (solid) as well as the computation time per iteration (dashed) for [191] (blue) and our method (orange). Our analytical loss function provides gradients with orders of magnitude less variance while at the same time decreasing runtime. In Fig. 2.14b, we compare the training loss evolution of [191] (blue) to ours (orange). The sampling based approach of [191] leads to large oscillations while ours converges smoothly.

details that require modeling beyond cuboids: For instance, our model predicts pointy octahedral shapes for the feet, ellipsoid shapes for the head and a flattened elongated superellipsoid for the main body without any supervision on the primitive parameters. Another interesting aspect of our model is the consistency of the predicted primitives, i.e., the same primitives (highlighted with the same color) consistently represent feet, legs, arms etc. across different poses. For more complicated poses, correspondences are sometimes mirrored. We speculate that this behavior is caused by symmetries of the human body.

2.3.5 *Analytical Loss Formulation*

In this section, we compare the evolution of our training loss in Equation (2.2) to the evolution of the training loss proposed by Tulsiani et al. [191] in Fig. 2.14b. While it is important to mention that the absolute values are not comparable due to the slightly different loss formulations, we observe that our loss converges faster with less oscillations. Note that at iteration 20k, [191] starts updating the existence probabilities using reinforcement learning [205] which further increases oscillations. In contrast, our loss function decays smoothly and does not require sampling-based gradient estimation.

To further analyze the advantages of our analytical loss formulation, we calculate the variance of the gradient estimates for the existence probabilities γ over 300 training iterations. Fig. 2.14a compares the variance of the gradients of [191] to the variance of the gradients of the proposed analytical loss (solid lines). Note that the variance in our gradient is orders of magnitude lower compared to [191] as we do not require sampling [205] for approximating the gradients. Simultaneously, we obtain a lower runtime per iteration (dashed lines). While using more samples lowers the variance of gradients approximated using Monte Carlo estimation [191], runtime per iteration increases linearly with the number of samples. In contrast, our method does not require sampling and outperforms [191] in terms of runtime even for the case of gradient estimates based on a single sample. We remark that in both cases the runtime is computed for the entire iteration, considering both, the forward and the backward pass.

2.4 IMPLEMENTATION DETAILS

Our network architecture comprises an encoder and a set of linear layers followed by non-linearities that independently predict the pose, shape and size of the superquadric surfaces. The encoder architecture is chosen based on the input type (e.g. image, voxelized input, etc.). In our experiments, for a binary occupancy grid as input, our encoder consists of five blocks of 3D convolution layers, followed by batch normalization and Leaky ReLU non-linearities. The result is passed to five independent heads that regress translation $\mathbf{t}$, rotation $\mathbf{q}$, size $\boldsymbol{\alpha}$, shape $\boldsymbol{\epsilon}$ and probability of existence γ for each primitive. An overview of our training architecture is provided in Fig. 2.15.

Note that, for the image-based experiment of Sec. 2.3.3 we replace the encoder architecture in Fig. 2.15 with a ResNet18 [75].

For evaluating our loss (2.2), we sample points on the superquadric surface. To achieve a uniform point distribution, we sample η and ω as proposed in [148]. During training, we uniformly sample 1000 points, from the surface of the target object, as well as 200 points from the surface of every superquadric. Note that sampling points on the surface of the objects results in a stochastic approximator of the expected loss. The variance of this approximator is inversely proportional to the number of sampled points. We experimentally observe that our model is not sensitive to the number of sampled points. For optimization, we use ADAM [96] with learning rate 0.001 and a batch size of 32 for 40k iterations. To further increase parsimony,

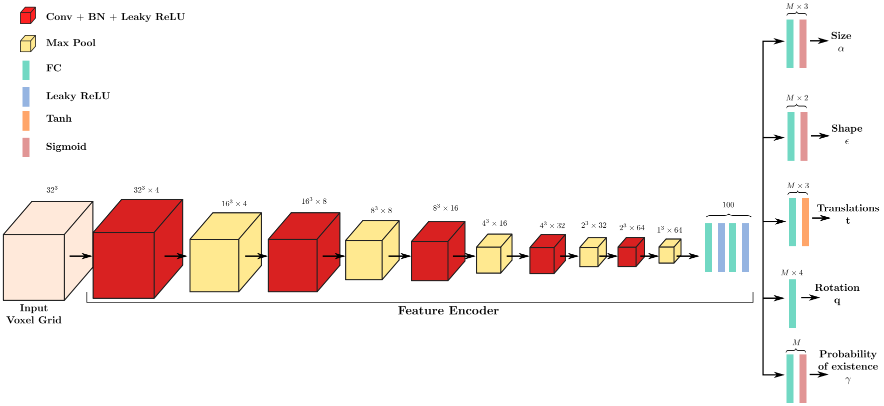


Figure 2.15: **Volume-based Network Architecture.** We visualize the layers that comprise our network architecture. Cubes denote operations that are conducted on 3-dimensional volumes, while rectangles correspond to K-dimensional features. The number above each shape (cube or rectangle) corresponds to the dimensionality of that layer. For instance, $16^3 \times 4$ denotes a feature map of size 16^3 and 4 channels. Following, our notation, M corresponds to the maximum number of primitives predicted.

we then fix all parameters except γ for additional 5k iterations. This step removes remaining overlapping primitives as also observed in [191].

2.4.1 *Metrics Computation*

In this section, we provide additional details regarding the quantitative comparison of Tab. 2.1. For evaluation, we report two metrics the mean *Chamfer distance* and the mean *Volumetric IoU*. Volumetric IoU is defined as the quotient of the volume of the two meshes' intersection and the volume of their union. We obtain unbiased estimates of the volume of the intersection and the union by randomly sampling points from the bounding volume and determining if the points lie inside our outside the ground truth / predicted mesh. The computation of the Chamfer distance is discussed in detail in Sec. 2.2.1.

2.4.2 *Parsimony Loss Details*

We would also like to briefly provide some additional details for our parsimony loss. For completeness, we restate the parsimony loss of (2.11)

$$\mathcal{L}_{\gamma}(\mathcal{P}) = \max\left(\alpha - \alpha \sum_{m=1}^{M} \gamma_m, 0\right) + \beta \sqrt{\sum_{m=1}^{M} \gamma_m} \tag{2.13}$$

Note that the $\sum_{m=1}^{M} \gamma_m$ corresponds to the expected number of primitives in the predicted parsing. Note that our model suffers from the trivial solution $\mathcal{L}_D(\mathcal{P}, \mathcal{X}) = 0$ which is attained for $\gamma_1 = \cdots = \gamma_m = 0$. To avoid this solution, we introduce the first term of Eq. 2.13 that penalizes the prediction when the expected number of primitives is less than 1. The second term penalizes the prediction when the expected number of primitives is large. Note that the maximum value of the second term is $\beta\sqrt{M}$, while the maximum value of the first term is α. Therefore, in order to allow the model to use more than one primitive, we set β to a value smaller than α. Typically $\alpha = 1.0$ and $\beta = 10^{-3}$.

2.5 DISCUSSION

We have presented the first learning-based approach for parsing 3D objects into consistent superquadric representations. Our model successfully captures both the structure as well as the details of the target objects by accurately learning to predict superquadrics in an unsupervised fashion during training. While our model more faithfully captures the geometry of 3D objects in comparison to cuboidal primitives, superquadric surfaces are still relatively simple, namely they cannot capture arbitrarily complex geometries without utilizing a large number of parts. However, increasing the number of parts results in less meaningful shape abstractions. This problem is addressed in Chapter 3.

Moreover, while our analytical solution to the Chamfer loss allows us to efficiently train our model without the need for reinforcement learning or iterative prediction, minimizing the discrepancy between the target and the predicted shape solely based on bidirectional Chamfer loss is problematic as it leads to various local minima, such as primitives that do not occupy meaningful space. To address this, we believe that jointly optimizing for the Chamfer distance and the Intersection over Union (IoU), which seeks to enforce that the volume of the target and the predicted match, would be more beneficial. Moreover, we believe that developing hierarchical strategies as in [226] would allow us to also reason about higher-level part relationships. This problem is addressed in Chapter 4.

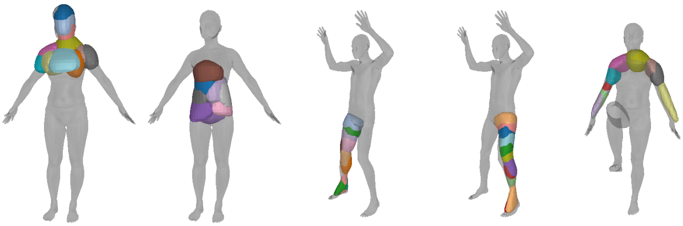
3

LEARNING EXPRESSIVE 3D SHAPE ABSTRACTIONS WITH INVERTIBLE NETWORKS

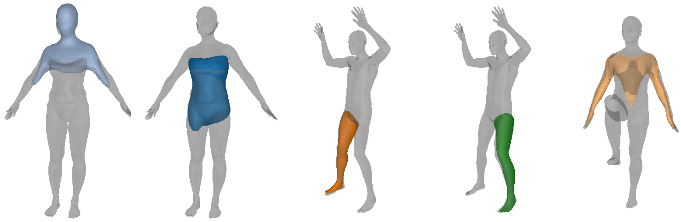
> *There is geometry in the humming of the strings, there is music in the spacing of the spheres.*
>
> — Pythagoras

Recovering the geometry of a 3D shape from a single RGB image is a fundamental task in computer vision and graphics. Existing shape reconstruction models utilize a neural network to learn a parametric function that maps the input image into a mesh [113, 68, 86, 199, 211, 134], a point-cloud [53, 151, 2, 84, 186, 211], a voxel grid [14, 34, 57, 159, 162, 180, 208] or an implicit surface [120, 27, 136, 168, 209, 121]. An alternative line of research focuses on compact low-dimensional representations that reconstruct 3D objects by decomposing them into simpler parts, called primitives [191, 141, 37, 61]. Primitive-based representations seek to infer *semantically consistent part arrangements* across different object instances and provide a more interpretable alternative, compared to representations that only focus on capturing the global object geometry. In Chapter 2, we proposed an unsupervised model for decomposing a 3D object into its constituent parts using superquadric surfaces as geometric primitives. While superquadric surfaces allow for a more diverse shape vocabulary in comparison to previous works, they still have limited representation power and cannot capture complex geometries. Similarly, more recent primitive-based approaches that rely on spheres [72] and convexes solids[37, 25] require a large number of primitives for extracting geometrically accurate reconstructions. However, using more primitives comes at the expense of the interpetability of the reconstruction, namely the final decompositions are less meaningful.

To address this, in this chapter, we introduce Neural Parts, a novel 3D primitive representation that is more *expressive* and *interpretable*, in comparison to alternatives that are limited to convex shapes. We argue that a primitive should be a non trivial genus-zero shape with well defined implicit and explicit representations. These characteristics allow us to efficiently combine primitives and accurately represent arbitrarily complex geometries. To this end, we pose the task of primitive learning as the task of learning *a family of homeomorphic mappings* between the 3D space of a simple genus-



(a) Convexes



(b) Neural Parts

Figure 3.1: **Expressive Primitives.** We address the trade-off between reconstruction quality and sparsity (i.e. number of parts) in primitive-based methods. Prior work [191, 141, 143, 37] has considered convex shapes as primitives, which due to their simplicity, require a large number of parts to accurately represent complex shapes. This results in less interpretable shape abstractions (i.e. primitives are not identifiable parts e.g. legs, arms etc.). In this work, we propose Neural Parts, a novel 3D primitive representation that can represent arbitrarily complex genus-zero shapes and thus yields geometrically more accurate and semantically more meaningful parts compared to simpler primitives.

zero shape (e.g. sphere, cube, ellipsoid) and the 3D space of the target object. We implement this mapping using an Invertible Neural Network (INN) [45]. Being able to map 3D points in both directions allows us to efficiently compute the explicit representation of each primitive, namely its tesselation as well as the implicit representation, i.e. the relative position of a point wrt. the primitive's surface. In contrast to prior work [191, 141, 37, 143] that directly predict the primitive parameters (i.e. centroids and sizes for cuboids and superquadrics and hyperplanes for convexes), we employ the INN to fully define each primitive. Note that while a homeomorphism preserves the genus of the shape it does not constrain it in any other way. As a result, while existing primitives are constrained to a specific family of shapes (e.g. ellipsoids), our primitives can capture arbitrarily complicated genus-zero shapes (see Fig. 3.1). We demonstrate that Neural Parts can be learned in an unsupervised fashion (i.e. without any primitive annotations), directly from unstructured 3D point clouds by ensuring that the assembly of predicted primitives accurately reconstructs the target.

In summary, we make the following **contributions**: We propose the first model that defines primitives as a homeomorphic mapping between two topological spaces through conditioning an INN on an image. Since the homeomorphism does not impose any constraints on the primitive shape, our model effectively decouples geometric accuracy from parsimony and as a result captures complex geometries with an order of magnitude fewer primitives. Experiments on ShapeNet [19], D-FAUST [13] and Frei-HAND [225] demonstrate that our model can parse objects into more expressive and semantically meaningful shape abstractions compared to models that rely on simpler primitives. The code to reproduce the experiments presented in this chapter can be found at https://paschalidoud.github.io/neural_parts.

3.1 RELATED WORK

Learning-based 3D reconstruction approaches can be categorized based on the type of their output representation to: depth-based [87, 73, 142, 47], voxel-based [34, 207, 57, 159], point-based [53, 151, 2], mesh-based [113, 68, 86], implicit-based [120, 27, 136] and primitive-based [191, 130, 141]. Here, we primarily focus on primitive-based methods that are more relevant to the model presented in this chapter. Since our formulation is independent of a specific INN implementation, a thorough discussion of INNs is beyond the scope of this work, thus we refer the reader to [4] for a detailed overview.

3D Representations: Voxels [14, 34, 207, 57, 159, 180, 208] naturally capture the 3D geometry by discretizing the shape into a regular grid. While several efficient space partitioning techniques [119, 162, 183, 71, 161] have been proposed to address their high memory and computation requirements, their application is still limited. A promising new direction explored learning a deformation of the grid itself to better capture geometric details [58]. Pointclouds [53, 151, 2, 84, 186, 211] are more memory efficient but lack surface connectivity, thus post-processing is necessary for generating the final mesh. Most mesh-based methods [113, 68, 86, 199, 211, 64, 134, 24, 37, 219] naturally yield smooth reconstructions but either require a deformable template mesh [199] or represent the geometry as an atlas of multiple mappings [68, 39, 118]. To address these limitations, implicit models [120, 27, 136, 168, 209, 121, 128, 131, 6, 201, 60, 26, 129, 7, 67, 169, 83, 30, 146, 158, 132] have recently gained popularity. These methods represent a 3D shape as the level-set of a distance or occupancy field implemented as a neural network, that takes a context vector and a query point and predicts either a signed distance value [136, 121, 7, 67, 181] or a binary occupancy value [120, 27] for the query point. While these methods result in accurate reconstructions, they lack interpretability as they do not consider the part-based object structure. Instead, in this work, we focus on part-based representations and showcase that our model simultaneously yields both geometrically accurate and interpretable reconstructions. Furthermore, in contrast to implicit models, that require expensive iso-surfacing operations (i.e. marching cubes) to extract a mesh, our model directly predicts a high resolution mesh for each part, without any post-processing.

Structured-based Representations: This line of research seeks to decompose 3D shapes into semantically meaningful simpler parts using either supervision in terms of the primitive parameters [226, 130, 106, 124, 59, 107, 174, 111] or without any part-level annotations [191, 141, 40, 143, 61, 37, 93]. Neural Parts perform primitive-based learning in an unsupervised manner. Traditional primitives include cuboids [191, 130, 226, 106, 124, 50], superquadrics [141, 143], convexes [37, 25, 56], CSG trees [173] or shape programms [51, 187, 114]. Due to the simplicity of the shapes of traditional primitives, the reconstruction quality of existing part-based methods is coupled with the number of primitives, namely a larger number of primitives results in more accurate reconstructions (see Fig. 3.5). However, these reconstructions are less parsimonious and the constituent primitives often lack a semantic interpretation (i.e. are not recognizable parts). Instead, Neural Parts are not restricted to convex shapes and can capture complex geometries with a few primitives. In recent work, [60] propose a 3D representation that decomposes space into a structured set of implicit functions [61]. However, extracting a single part from their prediction is not possible. This is not the case for our model.

Shape Deformations: Deforming a single genus-zero shape into more complicated shapes with graph convolutions [199], MLPs [68, 190] and Neural ODEs [69] has demonstrated impressive results. Groueix et al. [68] were among the first to employ an MLP to implement a homeomorphism between a sphere and a complicated shape. Note that since the deformation is implemented via an MLP, computing the inverse mapping becomes infeasible. Very recently, [69] proposed to learn the deformation of a single ellipsoid using several Neural ODEs. While [69] propose an invertible model, it does not consider any part decomposition or latent object structure.

Instead, we use the inverse mapping of the homeomorphism to define the predicted shape as the union of primitives, by discarding points from a part's surface that are internal to any other part. Thus our model learns to combine multiple genus-zero primitives and is able to reconstruct shapes of arbitrary genus. In addition, we formulate our optimization objective using both the forward and the inverse mapping of the homeomorphism, which in turn allows us to utilize both volumetric and surface information during training. This facilitates imposing additional constraints on the predicted primitives e.g. normal consistency and parsimony and improves performance.

3.2 METHOD

Given an input image we seek to learn a representation with M primitives that best describes the target object. We define our primitives via a *deformation between shapes* that is parametrized as a *learned homeomorphism* implemented with an Invertible Neural Network (INN). Using an INN allows us to efficiently compute the implicit and explicit representation of the predicted shape and impose various constraints on the predicted parts. In particular, for each primitive, we seek to learn a homeomorphism between the 3D space of a simple genus-zero shape and the 3D space of the target object, such that the deformed shape matches a part of the target object. Due to its simple implicit surface definition and tesselation, we employ a sphere as our genus-zero shape. We refer to the 3D space of the sphere as *latent space* and to the 3D space of the target as *primitive space*.

In Sec. 3.2.1, we present the explicit and implicit representation of our primitives as a homeomorphism of a sphere. Subsequently, in Sec. 3.2.2, we present our novel architecture for predicting multiple primitives using homeomorphisms conditioned on the input image. Finally, in Sec. 3.2.3, we formulate our optimization objective.

3.2.1 *Primitives as Homeomorphic Mappings*

A homeomorphism is a continuous map between two topological spaces Y and X that preserves all topological properties. Intuitively a homeomorphism is a continuous stretching and bending of Y into a new space X. In our 3D topology, a homeomorphism $\phi_{\theta} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is

$$\mathbf{x} = \phi_{\theta}(\mathbf{y}) \text{ and } \mathbf{y} = \phi_{\boldsymbol{\theta}}^{-1}(\mathbf{x}) \tag{3.1}$$

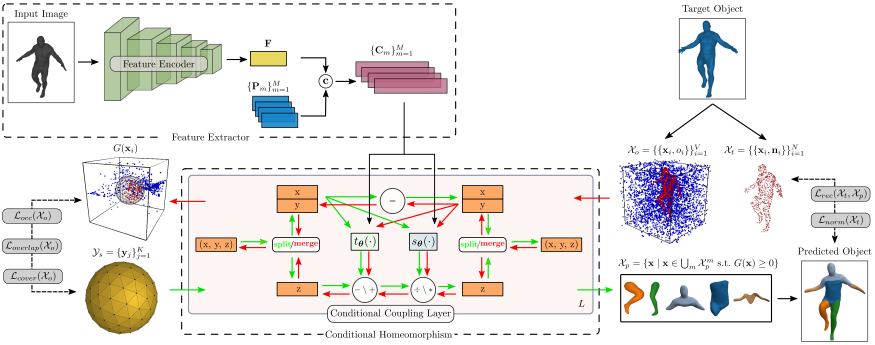

Input Image
Feature Encoder
F
$\{\mathbf{C}_m\}_{m=1}^M$
$\{\mathbf{P}_m\}_{m=1}^M$
c
Feature Extractor
Target Object
$G(\mathbf{x}_i)$
$\mathcal{X}_o = \{\{\mathbf{x}_i, o_i\}\}_{i=1}^V$
$\mathcal{X}_t = \{\{\mathbf{x}_i, \mathbf{n}_i\}\}_{i=1}^N$
$\mathcal{L}_{occ}(\mathcal{X}_o)$
$\mathcal{L}_{overlap}(\mathcal{X}_o)$
$\mathcal{L}_{cover}(\mathcal{X}_o)$
$\mathcal{Y}_s = \{\mathbf{y}_j\}_{j=1}^K$
x
y
z
(x, y, z)
split/merge
$t_{\theta}(\cdot)$
$s_{\theta}(\cdot)$
=
$-\backslash+$
$\div\backslash*$
Conditional Coupling Layer
L
Conditional Homeomorphism
$\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_p)$
$\mathcal{L}_{norm}(\mathcal{X}_t)$
Predicted Object
$\mathcal{X}_p = \{\mathbf{x} \mid \mathbf{x} \in \bigcup_m \mathcal{X}_p^m \text{ s.t. } G(\mathbf{x}) \geq 0\}$

Figure 3.2: **Method Overview.** Our model comprises two main components: A *Feature Extractor* which maps an input image into a per-primitive shape embedding and a *Conditional Homeomorphism* that deforms a sphere into M primitives and vice-versa. First, the *feature encoder* maps the input to a global feature representation $\mathbf{F}$. Then, for every primitive m, $\mathbf{F}$ is concatenated with a learnable primitive embedding $\mathbf{P}_m$ to generate the shape embedding $\mathbf{C}_m$ for this primitive. The Conditional Homeomorphism $\phi_{\boldsymbol{\theta}}(\cdot; \mathbf{C}_m)$ is implemented by a stack of L conditional coupling layers. Applying the *forward mapping* on a set of points $\mathcal{Y}_s$, randomly sampled on the surface of the sphere, generates points on the surface of the m-th primitive $\mathcal{X}_p^m$ (3.9). Using the *inverse mapping* $\phi_{\boldsymbol{\theta}}^{-1}(\cdot; \mathbf{C}_m)$, allows us to compute whether any point in 3D space lies inside or outside a primitive (3.7). We train our model using both *surface* ($\mathcal{L}_{rec}$, $\mathcal{L}_{norm}$) and *occupancy* ($\mathcal{L}_{occ}$) losses to simultaneously capture fine object details and volumetric characteristics of the target object. The use of the *inverse mapping* allows us to impose additional constraints (e.g. discouraging inter-penetration) on the predicted primitives ($\mathcal{L}_{overlap}$, $\mathcal{L}_{cover}$).

where $\mathbf{x}$ and $\mathbf{y}$ are 3D points in X and Y and $\phi_{\boldsymbol{\theta}} : Y \to X$, $\phi_{\boldsymbol{\theta}}^{-1} : X \to Y$ are continuous bijections. In our setting, Y and X correspond to the latent and the primitive space respectively. Using the explicit and implicit representation of a sphere with radius r, positioned at $(0, 0, 0)$ and the homeomorphic mapping from (3.1), we can now define the implicit and explicit representation of a single primitive.

Explicit Representation: The explicit representation of a primitive, parametrized as a mesh with vertices $\mathcal{V}_p$ and faces $\mathcal{F}_p$, can be obtained by applying the homeomorphism on the sphere vertices $\mathcal{V}$ and faces $\mathcal{F}$ as follows:

$$\begin{aligned} \mathcal{V}_p &= \{\phi_{\boldsymbol{\theta}}(\mathbf{v}_j),\ \forall\ \mathbf{v}_j \in \mathcal{V}\} \\ \mathcal{F}_p &= \mathcal{F}. \end{aligned} \tag{3.2}$$

Note that applying $\phi_{\boldsymbol{\theta}}$ on the sphere vertices $\mathcal{V}$ alters their location in the primitive space, while the vertex arrangements (i.e. faces) remain unchanged. Furthermore, since our primitives are defined as a deformation of a sphere mesh of arbitrarily high resolution, we can also obtain primitive meshes of arbitrary resolutions without any post-processing, e.g. marching-cubes.

Implicit Representation: The implicit representation of a primitive can be derived by applying the inverse homeomorphic mapping on a 3D point $\mathbf{x}$ as follows

$$g(\mathbf{x}) = \|\phi_{\boldsymbol{\theta}}^{-1}(\mathbf{x})\|_2 - r. \tag{3.3}$$

To evaluate the relative position of a point $\mathbf{x}$ wrt. the primitive surface it suffices to evaluate whether $\phi^{-1}(\mathbf{x})$ lies inside, outside or on the surface of the sphere. Namely, points that are internal to the sphere are also inside the primitive and points that are outside the sphere are also outside the primitive surface. Note that computing $g(\mathbf{x})$ in (3.3) is only possible because $\phi_{\boldsymbol{\theta}}(\mathbf{x})$ is implemented with an INN.

Multiple Primitives: The homeomorphism in (3.1) implements a single deformation. However, we seek to predict multiple primitives (i.e. deformations) conditioned on the input. Hence, we define a conditional homeomorphism as:

$$\mathbf{x} = \phi_{\boldsymbol{\theta}}(\mathbf{y}; \mathbf{C}_m) \text{ and } \mathbf{y} = \phi_{\boldsymbol{\theta}}^{-1}(\mathbf{x}; \mathbf{C}_m) \tag{3.4}$$

where $\mathbf{C}_m$ is the shape embedding for the m-th primitive and is predicted from the input. Note that for different shape embeddings a different homeomorphism is defined.

3.2.2 *Network Architecture*

Our architecture comprises two main components: (i) the *feature extractor* that maps the input to a vector of per-primitive shape embeddings $\{\mathbf{C}_m\}_{m=1}^{M}$ and (ii) the *conditional homeomorphism* that learns a homeomorphic mapping conditioned on the shape embedding. The overall architecture is illustrated in Fig. 3.2.

Feature Extractor: The first part of the feature extractor module is a ResNet-18 [75] that extracts a feature representation $\mathbf{F} \in \mathbb{R}^D$ from the input image. Subsequently, for every primitive m, $\mathbf{F}$ is concatenated with a learnable primitive embedding $\mathbf{P}_m \in \mathbb{R}^D$ to derive a shape embedding $\mathbf{C}_m \in \mathbb{R}^{2D}$ for this primitive.

Conditional Homeomorphism: We implement the INN using a Real NVP [46] due to its simple formulation. A Real NVP models a bijective mapping by stacking a sequence of simple bijective transformation functions. For each bijection, typically referred to as *affine coupling layer*, given an input 3D point (x_i, y_i, z_i), the output point (x_o, y_o, z_o) is

$$\begin{aligned} x_o &= x_i \\ y_o &= y_i \\ z_o &= z_i \exp\left(s_{\boldsymbol{\theta}}\left(x_i, y_i\right)\right) + t_{\boldsymbol{\theta}}\left(x_i, y_i\right) \end{aligned} \tag{3.5}$$

where $s_{\boldsymbol{\theta}} : \mathbb{R}^2 \rightarrow \mathbb{R}$ and $t_{\boldsymbol{\theta}} : \mathbb{R}^2 \rightarrow \mathbb{R}$ are *scale* and *translation* functions implemented with two arbitrarily complicated networks. Namely, in each bijection, the input is split into two, (x_i, y_i) and z_i. The first part remains unchanged and the second undergoes an affine transformation with $s_{\boldsymbol{\theta}}(\cdot)$ and $t_{\boldsymbol{\theta}}(\cdot)$. We follow [46] and we enforce that consecutive affine coupling layers scale and translate different input dimensions. Namely, we alternate the splitting between the dimensions of the input randomly.

However, the original Real NVP cannot be directly applied in our setting as it does not consider a shape embedding. To address this, we augment the affine coupling layer as follows: we first map (x_i, y_i) into a higher dimensional feature vector using a mapping, $p_{\boldsymbol{\theta}}(\cdot)$, implemented as an MLP. This is done to increase the relative importance of the input point before concatenating it with the high-dimensional shape embedding $\mathbf{C}_m$. The *conditional affine coupling layer* becomes

$$\begin{aligned} x_o &= x_i \\ y_o &= y_i \\ z_o &= z_i \exp\left(s_{\boldsymbol{\theta}}\left(\left[\mathbf{C}_m; p_{\boldsymbol{\theta}}\left(x_i, y_i\right)\right]\right)\right) \\ &\quad + t_{\boldsymbol{\theta}}\left(\left[\mathbf{C}_m; p_{\boldsymbol{\theta}}\left(x_i, y_i\right)\right]\right) \end{aligned} \tag{3.6}$$

where $[\cdot;\cdot]$ denotes concatenation. A graphical representation of our conditional coupling layer is provided in Fig. 3.3.

3.2.3 *Training*

Due to the lack of primitive annotations, we train our model by minimizing the geometric distance between the target and the predicted shape. In the following, we define the implicit and explicit representation of the predicted shape as the union of M primitives.

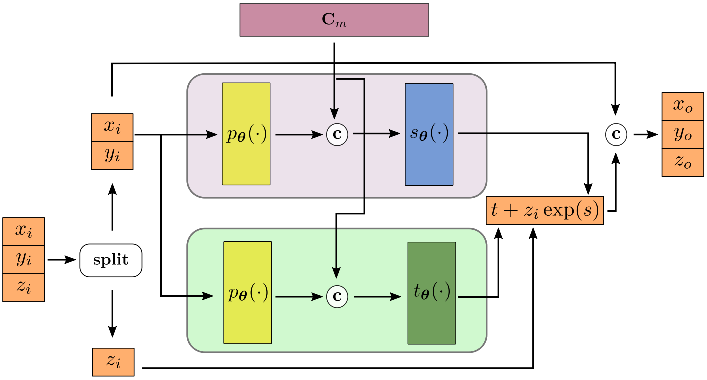


Figure 3.3: **Conditional Coupling Layer.** Pictorial representation of (3.20). The input point (x_i, y_i, z_i) is passed into a *coupling layer* that scales and translates one dimension of the input based on the other two and the per-primitive shape embedding $\mathbf{C}_m$. The scale factor s and the translation amount t are predicted by two MLPs, $s_\theta(\cdot)$ and $t_\theta(\cdot)$. $p_\theta(\cdot)$ is another MLP that increases the dimensionality of the input point before it is concatenated with $\mathbf{C}_m$.

The implicit surface of the m-th primitive can be derived from (3.3), by applying the inverse homeomorphic mapping on points in $3D$ space as follows

$$g^m(\mathbf{x}) = \|\phi_\theta^{-1}(\mathbf{x}; \mathbf{C}_m)\|_2 - r, \ \forall \ \mathbf{x} \in \mathbb{R}^3 \tag{3.7}$$

The implicit surface representation of the predicted object is defined as the union of all per-primitive implicit functions

$$G(\mathbf{x}) = \min_{m \in 0 \ldots M} g^m(\mathbf{x}), \tag{3.8}$$

namely a point is inside the predicted shape if it is inside at least one primitive.

Similarly, the explicit representation of the m-th primitive is a set of points on its surface, let it be $\mathcal{X}_p^m$, that are generated by applying the forward homeomorphic mapping on points on the sphere surface $\mathcal{Y}_s$ in the latent space

$$\mathcal{X}_p^m = \{\phi_\theta(\mathbf{y}_j; \mathbf{C}_m), \ \forall \ \mathbf{y}_j \in \mathcal{Y}_s\}. \tag{3.9}$$

To generate points on the surface of the predicted shape $\mathcal{X}_p$, we first need to generate points on the surface of each primitive and then discard the ones that are inside any other primitive. From (3.8) this can be expressed as follows:

$$\mathcal{X}_p = \{\mathbf{x} \mid \mathbf{x} \in \bigcup_m \mathcal{X}_p^m \text{ s.t. } G(\mathbf{x}) \geq 0\}. \tag{3.10}$$

Loss Functions: Our loss $\mathcal{L}$ seeks to minimize the geometric distance between the target and the predicted shape and is composed of five loss terms:

$$\begin{aligned}\mathcal{L} = &\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_p) + \mathcal{L}_{occ}(\mathcal{X}_o) + \mathcal{L}_{norm}(\mathcal{X}_t)\\ &+ \mathcal{L}_{overlap}(\mathcal{X}_o) + \mathcal{L}_{cover}(\mathcal{X}_o)\end{aligned} \tag{3.11}$$

where $\mathcal{X}_t = \{\{\mathbf{x}_i, \mathbf{n}_i\}\}_{i=1}^N$ comprises surface samples of the target shape and the corresponding normals, and $\mathcal{X}_o = \{\{\mathbf{x}_i, o_i\}\}_{i=1}^V$ denotes a set of occupancy pairs, where $\mathbf{x}_i$ corresponds to the location of the i-th point and o_i denotes whether $\mathbf{x}_i$ lies inside $(o_i = 1)$ or outside $(o_i = 0)$ the target. Note that our optimization objective comprises both occupancy (3.13) and surface losses (3.12)+(3.14), since they model complementary characteristics of the target object e.g. the surface loss attends to fine details that may have small volume, whereas the occupancy loss more efficiently models empty space. We empirically observe that using both significantly improves reconstruction (see Sec. 3.4).

Reconstruction Loss: We measure the surface reconstruction quality using a bidirectional Chamfer loss between the points $\mathcal{X}_p$ on the surface of the predicted shape and the points on the target object $\mathcal{X}_t$ as follows:

$$\begin{aligned}\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_p) = &\frac{1}{|\mathcal{X}_t|} \sum_{\mathbf{x}_i \in \mathcal{X}_t} \min_{\mathbf{x}_j \in \mathcal{X}_p} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 +\\ &\frac{1}{|\mathcal{X}_p|} \sum_{\mathbf{x}_j \in \mathcal{X}_p} \min_{\mathbf{x}_i \in \mathcal{X}_t} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2\end{aligned} \tag{3.12}$$

The first term of (3.12) measures the average distance of all ground truth points to the closest predicted points and the second term measures the average distance of all predicted points to the closest ground-truth points.

Occupancy Loss: The occupancy loss ensures that the volume of the predicted shape matches the volume of the target. Intuitively, we want to ensure that the free and the occupied space of the predicted and the target object coincide. To this end, we convert the implicit surface of the predicted shape from (3.8) to an indicator function and compute the binary cross-entropy loss over all volume samples $\mathcal{X}_o$

$$\mathcal{L}_{occ}(\mathcal{X}_o) = \sum_{(\mathbf{x},o)\in\mathcal{X}_o} \mathcal{L}_{ce}\left(\sigma\left(\frac{-G(\mathbf{x})}{\tau}\right), o\right). \tag{3.13}$$

$\mathcal{L}_{ce}(\cdot,\cdot)$ is the cross-entropy loss, $\sigma(\cdot)$ is the sigmoid function and τ is a temperature hyperparameter that defines the sharpness of the boundary of the indicator function. Note that $\sigma\left(\frac{-G(\mathbf{x})}{\tau}\right)$ is 1 when $\mathbf{x}$ is inside the predicted shape and 0 otherwise.

Normal Consistency Loss: The normal consistency loss ensures that the orientation of the normals of the predicted shape will be aligned with the normals of the target. We penalize misalignments between the predicted and the target normals by minimizing the cosine distance as follows

$$\mathcal{L}_{norm}(\mathcal{X}_t) = \frac{1}{|\mathcal{X}_t|} \sum_{(\mathbf{x},\mathbf{n})\in\mathcal{X}_t} \left(1 - \left\langle \frac{\nabla_{\mathbf{x}} G(\mathbf{x})}{\|\nabla_{\mathbf{x}} G(\mathbf{x})\|_2}, n \right\rangle\right) \tag{3.14}$$

where $\langle\cdot,\cdot\rangle$ is the dot product. Note that the surface normal of the predicted shape for a point $\mathbf{x}$ is simply the gradient of the implicit surface wrt. to point $\mathbf{x}$ and can be efficiently computed with automatic differentiation.

Overlapping Loss: To encourage semantically meaningful shape abstractions, (i.e. primitives represent different object parts), we introduce a non-overlapping loss that penalizes any point in space that is internal to more than λ primitives

$$\mathcal{L}_{overlap}(\mathcal{X}_o) = \frac{1}{|\mathcal{X}_o|} \max\left(0, \sum_{m=1}^{M} \sigma\left(\frac{-g^m(\mathbf{x})}{\tau}\right) - \lambda\right) \tag{3.15}$$

Coverage Loss: The coverage loss makes sure that all primitives cover parts of the predicted shape. In practice, it prevents degenerate primitive arrangements, where some primitives are very small and do not contribute to the reconstruction. We implement this loss by encouraging that each primitive contains at least k points of the target object:

$$\mathcal{L}_{cover}(\mathcal{X}_o) = \sum_{m=1}^{M} \sum_{\mathbf{x}\in\mathcal{N}_k^m} \max\left(0, g^m(\mathbf{x})\right). \tag{3.16}$$

Here $\mathcal{N}_k^m \subset \{(\mathbf{x}, o) \in \mathcal{X}_o | o = 1\}$ contains the k points with the minimum distance from the m-th primitive.

3.3 EXPERIMENTAL EVALUATION

3.3.1 *Datasets*

We evaluate our model on D-FAUST [13], FreiHAND [225] and ShapeNet [19]. D-FAUST contains 38,640 meshes of humans performing various tasks such as “chicken wings" and “running on spot". We follow [143] and use 70%, 20% and 10% for the train, test and validation splits. Furthermore, we filter out the first 20 frames for each sequence that contain the unnatural “neutral pose" necessary for calibration purposes. Note that for D-FAUST, we do not normalize meshes to the unit cube in order to retain the variety of the human body. For ShapeNet [19], we perform category specific training using the same image renderings and train/test splits as [34]. For D-FAUST [13], we follow the experimental evaluation proposed in [143] and for FreiHAND [225], we select the first 5000 hand poses and generate meshes using the provided MANO parameters [165]. We render them from a fixed orientation and use 70%, 20%, 10% for the train, test and validation splits.

3.3.2 *Metrics*

We evaluate our model and our baselines using the volumetric Intersection-over-Union (IoU) and the Chamfer-L_1 distance. We obtain unbiased low-variance estimates for the IoU by sampling 100,000 points from the bounding volume and determining if the points lie inside or outside the target/predicted mesh. Similarly, we obtain an unbiased low-variance estimator of the Chamfer-L_1 distance by sampling 10,000 points on the surface of the target/predicted mesh.

Volumetric IoU is defined as the quotient of the volume of the intersection of the target O_t and the predicted O_p object and the volume of their union.

$$\text{IoU}(O_t, O_p) = \frac{\mid V(O_t \cap O_p) \mid}{\mid V(O_t \cup O_p) \mid} \tag{3.17}$$

where $V(.)$ is a function that computes the volume of a mesh.

The Chamfer-L_1 distance is defined between a set of points sampled on the surface of the target and the predicted mesh. We denote $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$

the set of points sampled on the surface of the target mesh and $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^{M}$ the set of points sampled on the surface of the predicted mesh.

$$\text{Chamfer-}L_1(\mathcal{X}, \mathcal{Y}) = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{y} \in \mathcal{Y}} \|\mathbf{x} - \mathbf{y}\| + \frac{1}{M} \sum_{\mathbf{y} \in \mathcal{Y}} \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{y} - \mathbf{x}\| \tag{3.18}$$

The first term of (3.18) measures the *completeness* of the predicted shape, namely how far is on average the closest predicted point from a ground-truth point. The second term measures the *accuracy* of the predicted shape, namely how far on average is the closest ground-truth point from a predicted point. Note that during the evaluation we do not sample points on the sphere to generate points on the surface of the primitives. Instead we use the predicted meshes to ensure that the generated points are uniformly distributed on the surface of the predicted object.

3.3.3 *Baselines*

In this section, we provide additional details regarding our baselines. For a fair comparison, all baselines use the same feature extractor network, namely ResNet18 [75].

SQs: In SQs [141], the authors employ a convolutional neural network (CNN) to regress the parameters of a set of superquadric surfaces that best describe the target object, in an unsupervised manner, by minimizing the Chamfer-distance between points on the target and the predicted shape. In order to have predictions, with variable number of primitives, each primitive is associated with an existence probability. When the existence probability of a primitive is below a threshold this primitive is not part of the assembled object. We train SQs [141][1] using the provided PyTorch [144] implementation with the default parameters. Similar to our model, we sample 200 points on the surface of each primitive and $2,000$ on the surface of the target object. We train SQs with a batch size of 32 until convergence.

H-SQs: H-SQs [143] consider a hierarchical, geometrically more accurate decomposition of parts using superquadrics. In particular, H-SQs employ a neural network architecture that recursively partitions objects into their constituent parts by building a latent vector that jointly encodes both the part-level hierarchy and the part geometries. H-SQs are trained in an unsupervised fashion, with an occupancy loss function such that the model learns to classify whether points sampled in the bounding box that contains

---

[1] https://superquadrics.com/

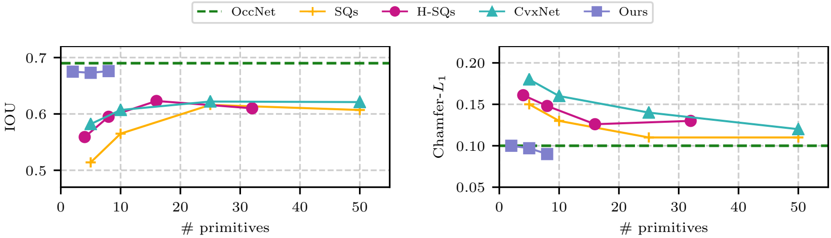


Figure 3.4: **Trade-off Reconstruction Quality and # Primitives**. We evaluate the reconstruction quality of primitive-based methods on the D-FAUST test set for different number of primitives. Neural Parts (**purple**) outperform CvxNet (**turquoise**), SQs (**orange**) and H-SQs (**magenta**) in terms of both IoU (↑) and Chamfer-L1 (↓) for any primitive configuration, even when using as little as 2 primitives. In addition, we show that our reconstructions are competitive to OccNet (dashed) which does not provide a primitive-based representation and requires expensive post-processing for extracting surface meshes.

the target object lie insider or outside the union of the predicted primitives. We train H-SQs [143] using the PyTorch code provided by the authors. Note that the maximum number of primitives for H-SQs needs to be a power of 2, hence we train with a different number of primitives than the other baselines. Again, we train H-SQs with a batch size of 32 until convergence.

CvxNet: CvxNet [37] propose to reconstruct a 3D shape as a collection of convex hulls. We train CvxNet [37][2] using the provided TensorFlow [1] code with the default parameters. To ensure that the evaluation is consistent and fair, we extract meshes using their codebase and evaluate the predicted primitives using our evaluation code. For all datasets, we train CvxNet with a batch size of 32 until convergence using their default parameters.

OccNet: In our evaluation, we also include OccNet [120][3] which we train using the provided code by the authors. In particular, OccNet is trained with a batch size of 32 until convergence. While OccNet does not consider any part-based decomposition of the target object and is not directly comparable with our model, we include it in our analysis as a typical representative of powerful implicit shape extraction techniques

---

[2] https://cvxnet.github.io/

[3] https://github.com/autonomousvision/occupancy_networks

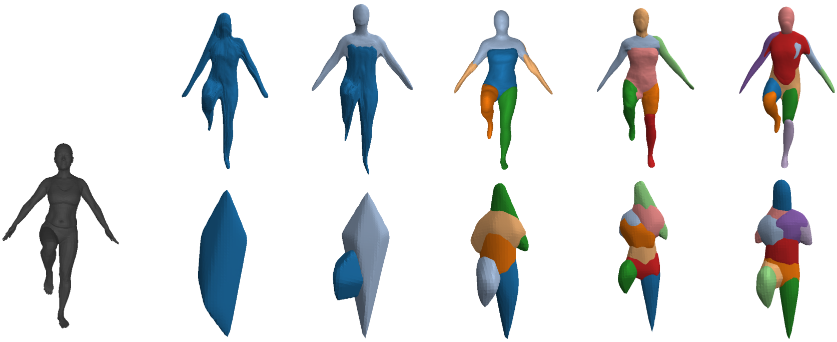


Figure 3.5: **Human Body Modelling**. We visualize the target mesh and the predicted primitives with Neural Parts (first row) and CvxNet (second) using $1, 2, 5, 8$ and 10 primitives.

3.3.4 *Representation Power*

In this experiment, we train our model and our baselines on D-FAUST for different number of primitives and measure their reconstruction quality wrt. IoU and Chamfer-L_1 distance. In particular, we train our model for $2, 5$ and 8 primitives, CvxNet [37] and SQs [141] for $5, 10, 25$ and 50 primitives and H-SQs [143] for $4, 8, 16$ and 32 primitives. We observe that our model achieves more accurate reconstructions for any given number of primitives and is competitive to OccNet that does not reason about parts. In particular, our model achieves 67.3% IOU with only 5 primitives, whereas CvxNet, with 10 times more primitives, achieves 62% (see Fig. 3.4). In Fig. 3.5, we provide a qualitative comparison of reconstructions with different number of parts and we observe that Neural Parts accurately capture the human limbs with as little as one or two primitives, whereas CvxNet cannot capture the arms even with 10 primitives.

3.3.5 *Reconstruction Accuracy*

Dynamic FAUST: In this experiment, we compare CvxNet and SQs with 50 primitives and H-SQs for a maximum number of 32 primitives to Neural Parts with 5 primitives to showcase that our model can capture the human body's geometry using an order of magnitude less primitives. Qualitative results from the predicted primitives using our model and the baselines

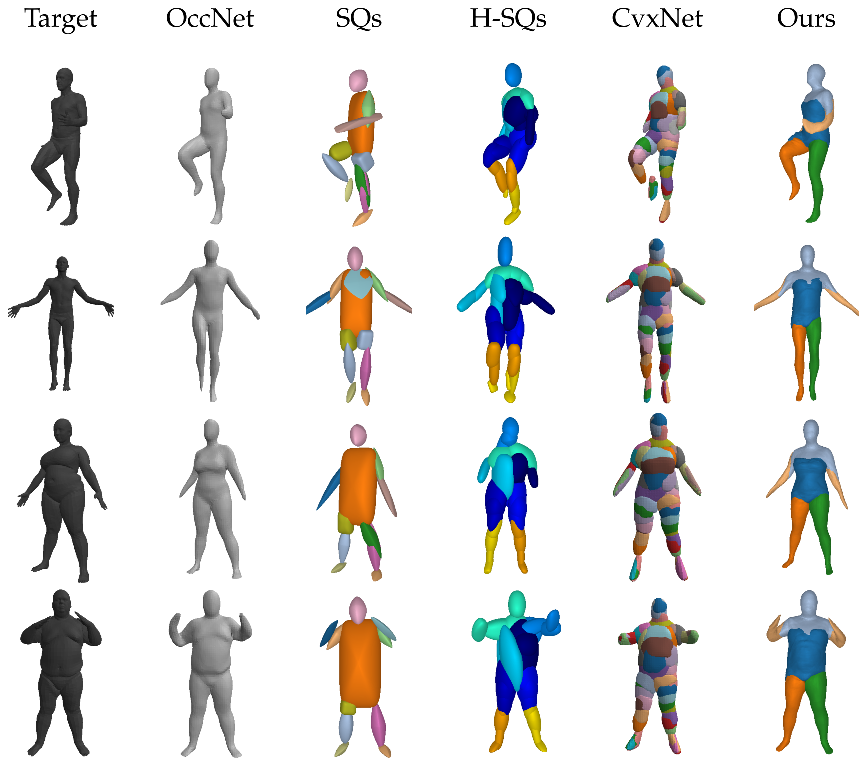
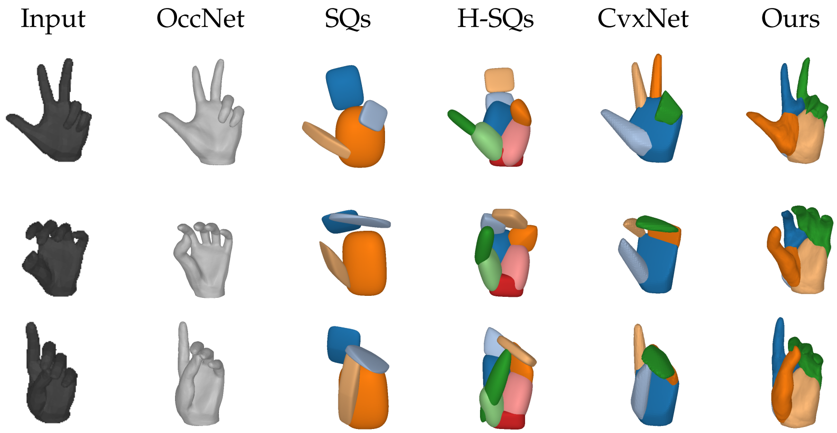


Figure 3.6: **Single Image 3D Reconstruction on D-FAUST**. The input image is shown on the first column and the rest contain predictions of all methods: OccNet (second), primitive-based predictions with superquadrics (third and fourth) and convexes (fifth) and ours with 5 primitives (last).

are summarized in Fig. 3.6. Note that OccNet is not directly comparable with part-based methods, however, we include it in our analysis as a typical representative of powerful implicit shape extraction techniques. We observe that while all methods roughly capture the human pose, Neural Parts result in a part assembly that is very close to the target object. While CvxNet with 50 primitives yield fairly accurate reconstructions, the final representation lacks any part-level semantic interpretation. The quantitative evaluation of this experiment is provided in Fig. 3.4.

FreiHAND: Similarly, we train our model, CvxNet and SQs with 5 and H-SQs with 8 primitives on the FreiHAND dataset and we observe that Neu-

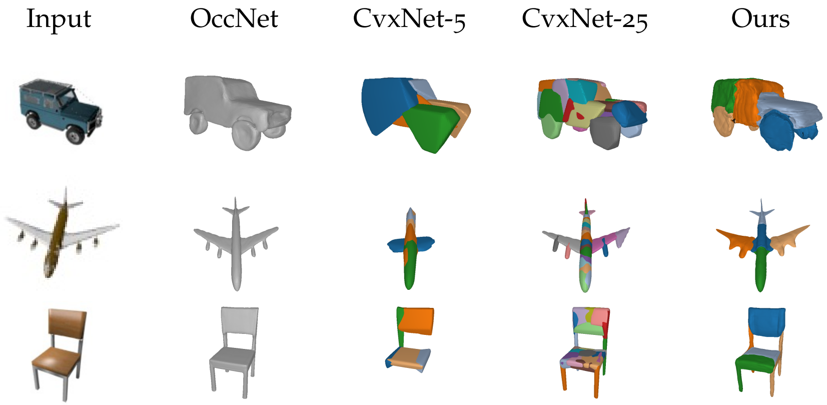


	OccNet	SQs	H-SQs	CvxNet	Ours
IoU	0.891	0.693	0.768	0.832	**0.879**
Chamfer-L_1	0.038	0.093	0.077	0.059	**0.057**

Figure 3.7: **Single Image 3D Reconstruction on FreiHAND**. We compare our model with OccNet, SQs and CvxNet with 5 primitives and H-SQs with 8 primitives. Our model outperforms all primitive based methods in terms of both IoU (↑) and Chamfer-L1 (↓) distance.

ral Parts yield more geometrically accurate reconstructions that faithfully capture fine details, i.e. the position of the thumb, (see Fig. 3.7). In contrast, CvxNet, H-SQs, SQs focus primarily on the structure of the predicted shape and miss out fine details.

ShapeNet: We train our model with 5 primitives and CvxNet with 5 and 25 primitives on cars, planes and chairs and observe that our model results in more accurate reconstructions than CvxNet with both 5 and 25 primitives (see Fig. 3.8). When increasing the number of primitives to 25, CvxNet improves in terms of reconstruction quality but the predicted primitives lack semantic interpretation. For the case of chairs and planes, CvxNet with 25 primitives accurately capture the object's geometry, but when we reduce the primitives to 5 entire object parts are missing.



IoU	OccNet	CvxNet - 5	CvxNet - 25	Ours
cars	0.763	0.650	0.666	**0.697**
planes	0.451	0.425	0.448	**0.454**
chairs	0.432	0.364	0.392	**0.412**

Figure 3.8: **Single Image 3D Reconstruction on ShapeNet**. We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5× less primitives.

3.4 ABLATION STUDY

In this section, we investigate how various components of our system affect the performance of Neural Parts on the single-view 3D reconstruction task on the D-FAUST dataset. In Sec. 3.4.1, we examine the impact of the INN on the performance of our model. Next, in Sec. 3.4.2, we investigate the impact of $p_{\theta}(\cdot)$ and in Sec. 3.4.3, we discuss the effect of each loss term on the overall performance of our model. Unless stated otherwise, we train all models with 5 primitives for 300 epochs.

3.4.1 *Invertibility*

In this section, we examine the impact of the Invertible Neural Network (INN) on the performance of Neural Parts. To this end, we implement a variant of our model that does not consider the inverse mapping of the homeomorphism, namely without the $\phi_{\theta}^{-1}(\mathbf{x})$. As a result, for this variant it is not possible to compute the union of parts, as formulated in (3.10), namely we cannot discard points on a primitive's surface that are internal to another primitive when generating points on the surface of the predicted shape. Moreover, we cannot impose any additional constraints on the predicted primitives. Therefore, we train this variant using the reconstruction loss between the target $\mathcal{X}_t$ and the predicted $\mathcal{X}_{\hat{p}}$ shape, $\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_{\hat{p}})$ from (3.12) as follows:

$$\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_{\hat{p}}) = \frac{1}{|\mathcal{X}_t|} \sum_{\mathbf{x}_i \in \mathcal{X}_t} \min_{\mathbf{x}_j \in \mathcal{X}_{\hat{p}}} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 + \frac{1}{|\mathcal{X}_{\hat{p}}|} \sum_{\mathbf{x}_j \in \mathcal{X}_{\hat{p}}} \min_{\mathbf{x}_i \in \mathcal{X}_t} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \tag{3.19}$$

where $\mathcal{X}_{\hat{p}} = \mathbf{x} \in \bigcup_m \mathcal{X}_p^m$, namely the union of surface points on each primitive.

Furthermore, we introduce an additional baseline that replaces the INN with an MLP. Similar to AtlasNet [68], we use a 4-layer MLP to learn the homeomorphic mappings between the sphere and the target object. Since the homeomorphism is implemented via an MLP it is not possible to define its inverse mapping. Thus, also this baseline is trained using the loss of (3.19). Note that this baseline is slightly different from AtlasNet [68] because instead of having a single MLP for each homeomorphism, we have one MLP for the M homeomorphisms. In particular, for this baseline, we implement the decoder architecture proposed in [68].

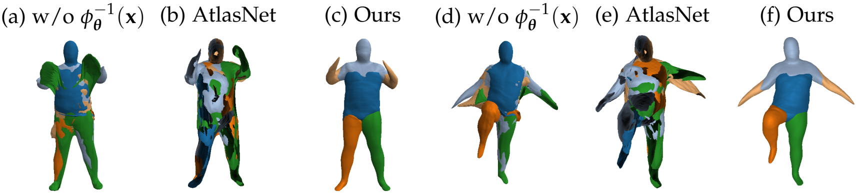

(a) w/o $\phi_\theta^{-1}(\mathbf{x})$ (b) AtlasNet (c) Ours (d) w/o $\phi_\theta^{-1}(\mathbf{x})$ (e) AtlasNet (f) Ours

Figure 3.9: **Impact of the INN.** We visualize the predicted primitives for our model (third and sixth column), a variant of our model that ignores the inverse mapping of the homeomorphism (first and fourth column) and the AtlasNet-sphere baseline that implements the homeomorphism with an MLP.

Tab. 3.1 summarizes the quantitative comparison between our model and the aforementioned baselines. Note that for the AtlasNet-sphere baseline, we cannot compute the IoU since it is not possible to enforce that the predicted primitives do not degenerate to "sheets", which contain no points (zero volume primitives). Indeed, from our evaluation, we observe that almost all predicted primitives (see Fig. 3.10) have zero volume and also inverted normals, which means that the sphere has folded on itself (self-intersections). This is only possible if the MLP is not implementing a homeomorphism. Due to the aforementioned issues, the AtlasNet-sphere cannot be used for learning primitives, however, we validate that it achieves high scores in terms of Chamfer-L_1 distance, which is justified as AtlasNet-sphere is optimized solely on this metric.

	w/o $\phi_\theta^{-1}(\mathbf{x})$	AtlasNet - sphere	Ours
IoU	0.639	$*$	0.673
Chamfer-L_1	0.119	0.087	0.097

Table 3.1: **Ablation Study of INN.** This table shows a quantitative comparison of our model wrt. a variant of our model that does not consider the inverse mapping of the homeomorphism and a second baseline that replaces the INN with an MLP for learning the homeomorphic mapping. We refer to the latter as AtlasNet - sphere, as it is a variant of the original AtlasNet method [68]. We note that our model outperforms both baseline both in terms of IoU and Chamfer-L_1 distance.

On the contrary, our method always learns valid homeomorphisms even when the inverse mapping ($\phi_\theta^{-1}(\cdot)$) is not used during training. This stems from the use of the invertible network, which implements a bijection that makes self-intersections impossible. Formally, for a self-intersection, two different points on the sphere need to be mapped onto the same point on the primitive space. However, this contradicts the bijection; hence our model cannot have self-intersections. This becomes evident from Fig. 3.9a+Fig. 3.9d, where we observe that the predicted primitives do not have inverted normals. Note that without the inverse mapping we cannot enforce that primitives will not interpenetrate, hence they are not semantically meaningful.

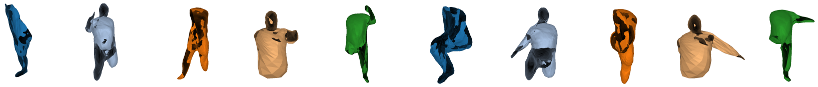


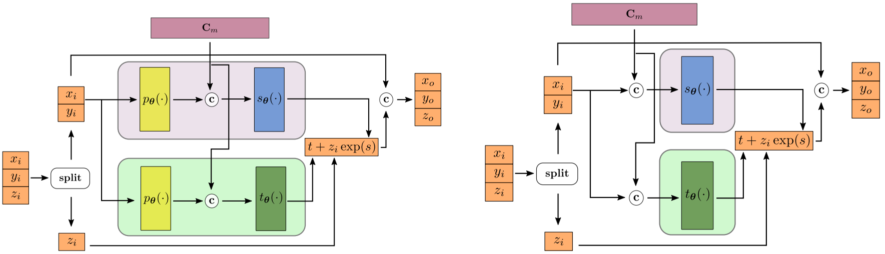
Figure 3.10: **Predicted primitives with AtlasNet-sphere.** We visualize the individual primitives for the predictions of AtlasNet-sphere. Note that some of the predicted primitives are hollow (first, third, sixth and eighth), some have holes (second, seventh) and all of them have inverted normals.

3.4.2 *Effect of* $p_{\boldsymbol{\theta}}(\cdot)$

The original Real NVP [46] cannot be directly applied in our setting as it does not consider a shape embedding. To address this, we augment the affine coupling layer as follows: we first map (x_i, y_i) into a higher dimensional feature vector using a mapping, $p_{\boldsymbol{\theta}}(\cdot)$, implemented as an MLP. This is done to increase the relative importance of the input point before concatenating it with the high-dimensional shape embedding $\mathbf{C}_m$. The *conditional affine coupling layer* becomes

$$\begin{aligned} x_o &= x_i \\ y_o &= y_i \\ z_o &= z_i \exp\left(s_{\boldsymbol{\theta}}\left(\left[\mathbf{C}_m; p_{\boldsymbol{\theta}}\left(x_i, y_i\right)\right]\right)\right) + t_{\boldsymbol{\theta}}\left(\left[\mathbf{C}_m; p_{\boldsymbol{\theta}}\left(x_i, y_i\right)\right]\right) \end{aligned} \tag{3.20}$$

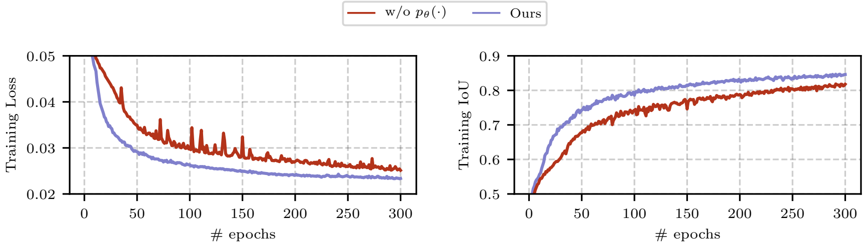
where $[\cdot;\cdot]$ denotes concatenation. A graphical representation of our conditional coupling layer is provided in Fig. 3.11a. In this section, we examine the impact of the mapping $p_{\boldsymbol{\theta}}(\cdot)$ on the performance of our model. In particular, instead of first mapping the input point (x_i, y_i, z_i) into a higher dimensional space and then concatenating it with the per-primitive shape embedding $\mathbf{C}_m$, we concatenate it as is (see Fig. 3.11b).



(a) Ours (b) Conditional Coupling Layer w/o $p_{\boldsymbol{\theta}}(\cdot)$

Figure 3.11: **Conditional Coupling Layer.** Pictorial representation of our conditional affine coupling layer. The input point (x_i, y_i, z_i) is passed into a *coupling layer* that scales and translates one dimension of the input based on the other two and the per-primitive shape embedding $\mathbf{C}_m$. The scale factor s and the translation amount t are predicted by two MLPs, $s_{\boldsymbol{\theta}}(\cdot)$ and $t_{\boldsymbol{\theta}}(\cdot)$. $p_{\boldsymbol{\theta}}(\cdot)$ is another MLP that increases the dimensionality of the input point before it is concatenated with $\mathbf{C}_m$.

We observe that removing $p_{\boldsymbol{\theta}}(\cdot)$ makes it harder for the network to produce a scaling and translation dependent on the input point due to the large difference in the number of dimensions compared to the shape



(a) Evolution of Training Loss (b) Evolution of Training IoU

Figure 3.12: **Training Convergence w/o $p_\theta(\cdot)$.** We illustrate the training evolution over 300 epochs in terms of training loss from (3.11) and IoU for our method with (**purple**) and without $p_\theta(\cdot)$ (**red**). We observe that our method converges smoother and to a lower loss and higher IoU when using $p_\theta(\cdot)$.

embedding $\mathbf{C}_m$. This results in slower convergence in comparison to our full model (see Fig. 3.12). Moreover, we also notice that the reconstruction quality becomes worse both in terms of IoU and Chamfer-L_1 distance (see Tab. 3.2).

	w/o $p_\theta(\cdot)$	Ours
IoU	0.638	0.673
Chamfer-L_1	0.106	0.097

Table 3.2: **Ablation Study on $p_\theta(\cdot)$.** This table shows a numeric comparison of our approach wrt. a variant of our model that does not utilize $p_\theta(.)$, namely instead of first mapping the input point (x_i, y_i, z_i) into a higher dimensional space, we directly concatenate the input point with the per-primitive shape embedding $\mathbf{C}_m$.

3.4.3 *Loss Functions*

Neural Parts are trained in an unsupervised fashion, without any primitive annotations. To address, the lack of part-level supervision, we learn this task by minimizing the geometric distance between the target and the predicted shape. In this section, we discuss how each loss term affects the performance of our model on the single-view 3D reconstruction task. In particular, we train 5 variants of our model and for each one we omit one of the loss terms. We provide both quantitative (see Tab. 3.3) and qualitative comparison (see Fig. 3.13) for the 5 different scenarios. For a fair comparison all models are trained for the same number of epochs.

	w/o $\mathcal{L}_{occ}$	w/o $\mathcal{L}_{rec}$	w/o $\mathcal{L}_{norm}$	w/o $\mathcal{L}_{overlap}$	w/o $\mathcal{L}_{cover}$	Ours
IoU	0.642	0.643	0.669	0.670	0.668	0.673
Chamfer-L_1	0.125	0.150	0.096	0.098	0.096	0.097

Table 3.3: **Ablation Study on Loss Terms.** We investigate the impact of each loss term on the reconstruction accuracy of our model. We report the IoU (↑) and the Chamfer-L_1 distance (↓) on the single-view 3D reconstruction task on D-FAUST dataset.

w/o Reconstruction Loss: The reconstruction loss is a bidirectional Chamfer loss between the surface points on the target and the predicted shape. We note that removing $\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_p)$ results in degenerate primitive arrangements that fail to capture the object geometry.

w/o Occupancy Loss: The occupancy loss enforces that the free and the occupied space of the predicted and the target object will coincide. Therefore, when we remove $\mathcal{L}_{occ}(\mathcal{X}_o)$ from our optimization objective, we observe that the predicted primitives accurately represent geometric parts of the human body but fail to capture the empty space (see hands and legs in first column of Fig. 3.13). Note that the points on the primitive surface that capture empty space have a small distance from points on the target object and as a result, $\mathcal{L}_{rec}(\mathcal{X}_t, \mathcal{X}_p)$ has small values for these points, which makes it difficult to penalize them for covering unoccupied space.

w/o Normal Consistency Loss: The normal consistency loss enforces that the normals of the predicted primitives will be aligned with the normals of the target object. We notice that when we remove $\mathcal{L}_{norm}(\mathcal{X}_t)$, our model yields non-overlapping primitives that accurately capture the geometry of different human body parts (see Fig. 3.15). However, at the locations where one primitive meets its adjacent, the predicted primitives have inconsistent normals, hence the connections seem unnatural. This can be better seen in Fig. 3.14, where we provide close-ups on the locations where two primitives meet and compare with our model. Our model, yields primitives with smooth transitions that look more natural.

w/o Overlapping Loss: The overlapping loss encourages semantically meaningful shape abstractions, where primitives represent distinct geometric parts. To this end, we introduce the $\mathcal{L}_{overlap}(\mathcal{X}_o)$ loss from (3.15) to discour-

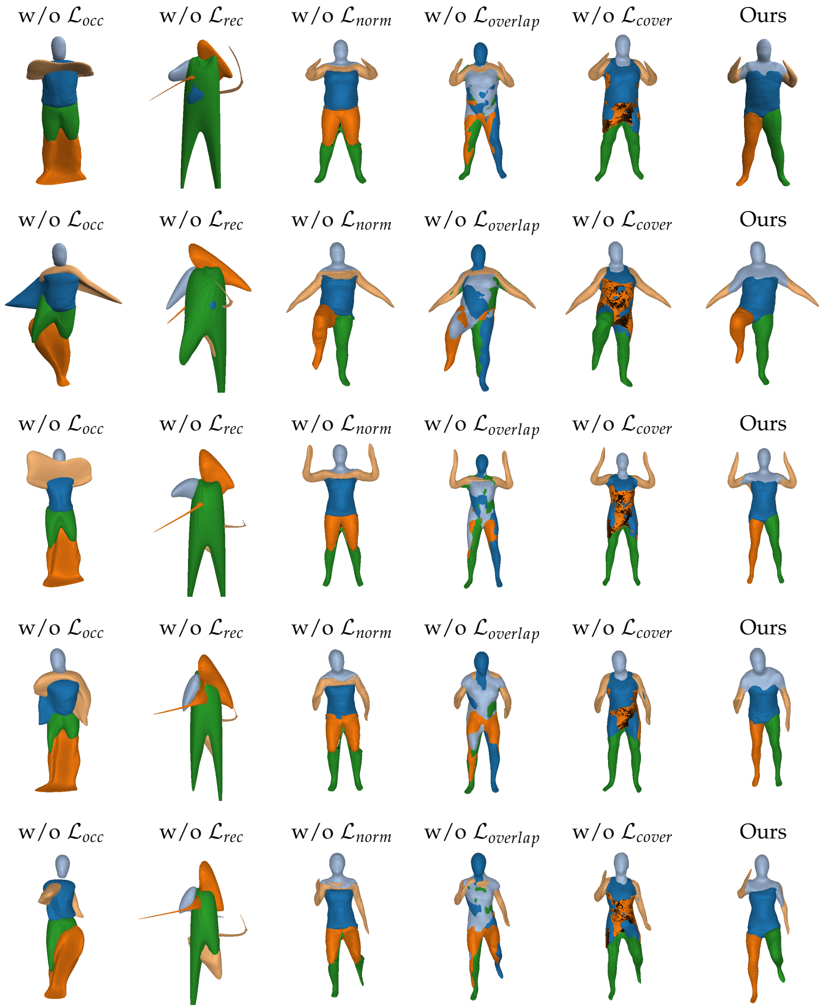
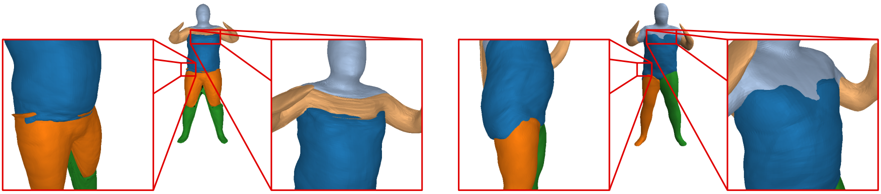
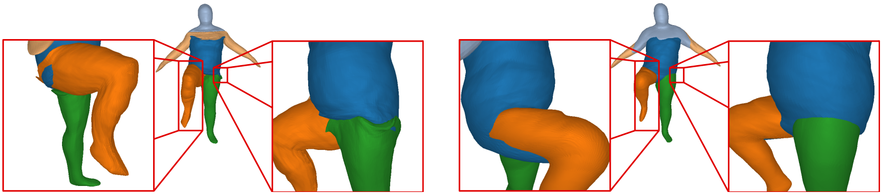


Figure 3.13: **Ablation Study on Loss Terms.** Qualitative evaluation of the impact of the 5 loss terms on the performance of our model. We train Neural Parts, without each loss term and visualize the predicted primitives.



(a) w/o $\mathcal{L}_{normal}$ (b) Ours



(c) w/o $\mathcal{L}_{normal}$ (d) Ours

Figure 3.14: **Impact of $\mathcal{L}_{norm}$.** When removing the $\mathcal{L}_{norm}$ from the optimization objective, the predicted primitives are semantically meaningful but the transitions between one primitive to another are not smooth i.e. do not have consistent normals. This becomes more evident when looking at the locations where primitives meet (marked with a red rectangle). In contrast, our model has smooth transitions and looks more "organic".

age having multiple primitives "containing" the same points from the target object. The reconstructions without the overlapping loss accurately capture the geometry of the human body but fail to yield semantically meaningful shape abstractions, since primitives completely penetrate one another (see Fig. 3.13). In Fig. 3.16, we visualize the predicted primitives for various humans and we notice that while they accurately capture the geometry of the human body, they do not correspond to meaningful geometric parts. In particular, there are 2 primitives that describe the right leg (primitive illustrated in blue and orange), 2 for the left leg (primitives illustrated in orange and green) and 3 for the main body (blue, light blue and green). Since, we want our primitives to correspond to semantic parts, this behaviour is not desirable.

w/o Coverage Loss: The coverage loss makes sure that all primitives will "cover" parts of the target object. In practice, it prevents primitives with minimal volume (see Fig. 3.18, fourth column) that do not contribute to the reconstruction. In Fig. 3.18, we note that when we remove $\mathcal{L}_{cover}(\mathcal{X}_o)$

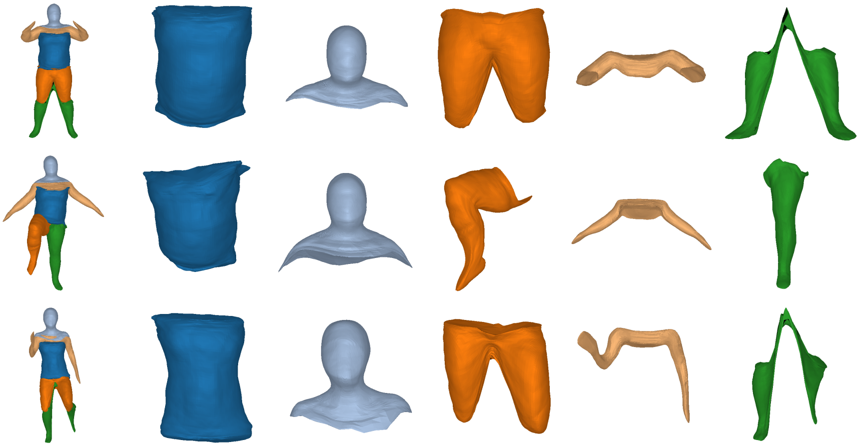


Figure 3.15: **Predicted Primitives w/o Normal Consistency Loss**. We visualize the predicted primitives when training without the $\mathcal{L}_{normal}$ and we observe that even though the predicted primitives are expressive and have a semantic interpretation, the locations where the primitives meet do not have consistent normals.

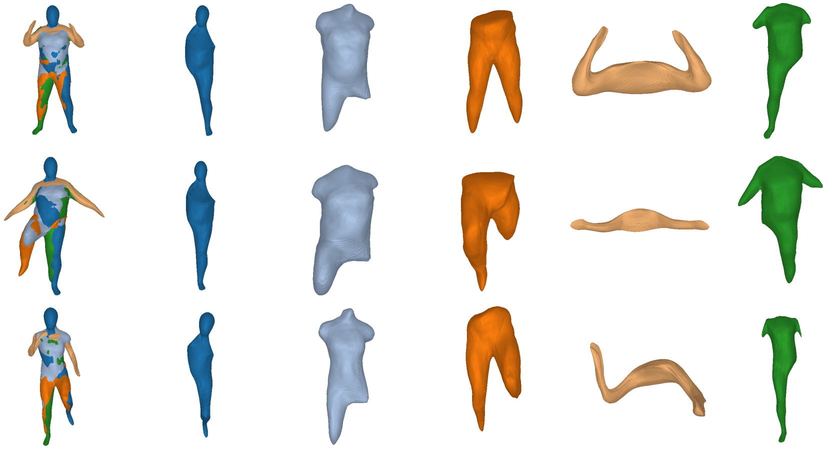


Figure 3.16: **Predicted Primitives w/o Overlapping Loss**. We visualize the predicted primitives when training without the $\mathcal{L}_{overlap}$ and we observe that even though the predicted primitives are expressive and accurately capture the 3D geometry of the human body, they do not have a semantic interpretation, namely multiple primitives describe the same object part.

from our optimization objective, a primitive degenerates to a "sheet" that contains no points but only has surface area (Fig. 3.17).

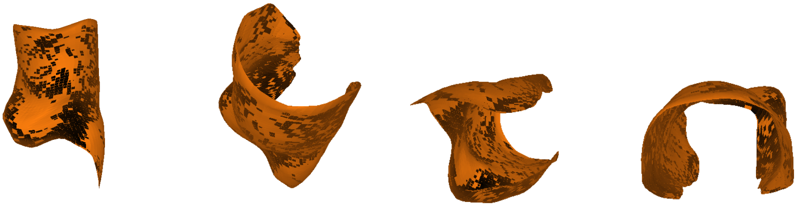


Figure 3.17: **Impact of $\mathcal{L}_{cover}$.** When removing the $\mathcal{L}_{cover}$ some primitives become very thin (with minimum volume) and do not contribute to the reconstruction loss.

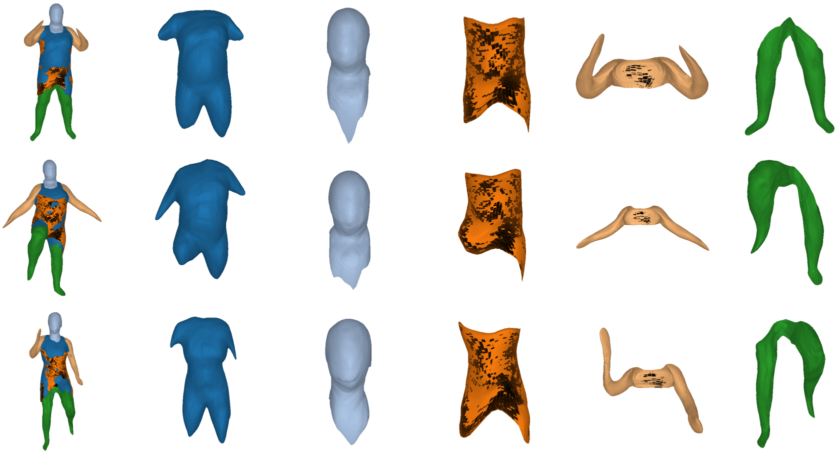


Figure 3.18: **Predicted Primitives w/o Coverage Loss**. We visualize the predicted primitives when training without the $\mathcal{L}_{cover}$ and observe that although some of them are expressive and efficiently capture the 3D geometry, some primitives have minimum volume hence, do not cover parts of the object and in turn, do not contribute to the reconstruction loss.

3.4.4 *Sensitivity to initialization*

In this section, we investigate the sensitivity of our model to initialization, namely we showcase that our model consistently yields almost identical semantic parts for different random initializations. In particular, we train

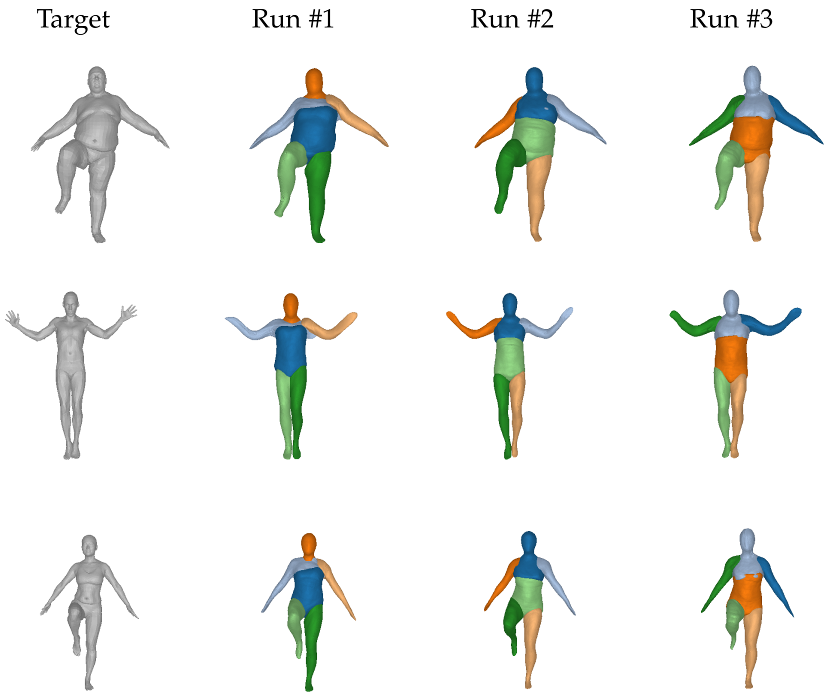
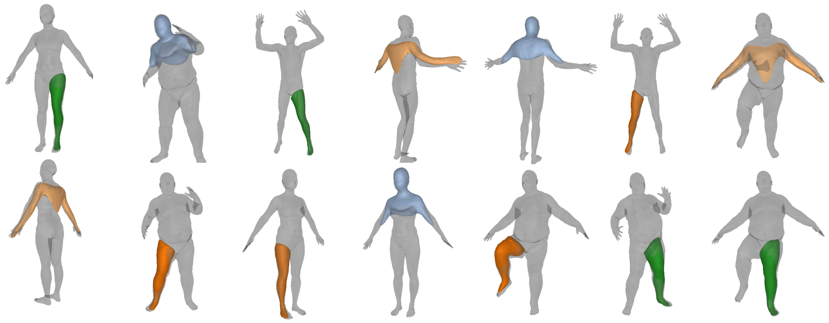


Figure 3.19: **Sensitivity to Initialization**. The input image is shown on the first column and the rest contain predictions of our method when trained three different times with three different random seeds.

our model three times with three different random seeds on D-FAUST with 6 primitives and visualize the predicted primitives on various humans in Fig. 3.19. We observe that while some of the predicted parts for Run#1 are slightly different from the parts in Run#2, 3, they are still semantically meaningful.

3.4.5 *Semantic Consistency*

We now investigate the ability of our model to decompose 3D objects into semantically consistent parts. Similar to [37], we use 5 representative vertex indices provided by SMPL-X [145] (i.e. thumbs, toes and nose) and compute the classification accuracy of those points when using the label of the closest primitive. We compare with CvxNet with 5 and 50 primitives and note that Neural Parts are more semantically consistent (see Fig. 3.20).



	L-thumb	R-thumb	L-toe	R-toe	Nose
CvxNet-5	61.1%	67.1%	98.2%	91.2%	98%
CvxNet-50	29%	37%	56.3%	58.1%	52%
Ours	91.9%	88.2%	99.8%	92.5%	100%

Figure 3.20: **Semantic Consistency.** We report the classification accuracy of semantic vertices on the human body using the label of the closest primitive. Our predicted primitives are consistently used for representing the same human part.

3.5 IMPLEMENTATION DETAILS

In this section, we provide a detailed description of our network architecture. We then describe our training protocol, our sampling strategy and provide details on the computation of the metrics during training and testing. Finally, we also provide additional details regarding our baselines.

3.5.1 *Network Architecture*

Here we describe the architecture of each individual component of our model, illustrated in Fig. 3.2. Our architecture comprises two main components: (i) a *feature extractor* that maps the input into a vector of per-primitive shape embeddings $\{\mathbf{C}_m\}_{m=1}^{M}$ and (ii) the *conditional homeomorphism* that learns a family of homeomorphic mappings between the sphere and the target shape, conditioned on the shape embedding.

Feature Extractor: The first part of the *feature extractor* is the feature encoder. In our experiments, the feature encoder is implemented with a ResNet-18 architecture [75] that is pre-trained on ImageNet [38]. From the original

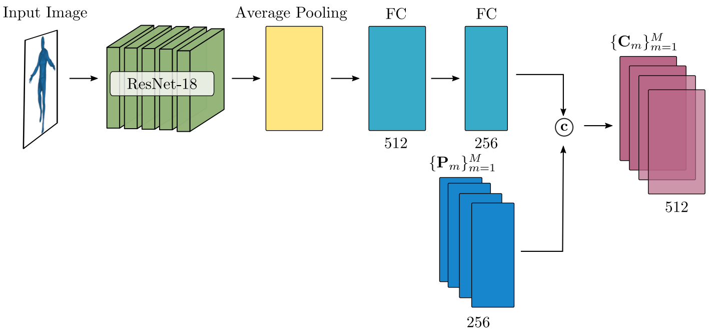
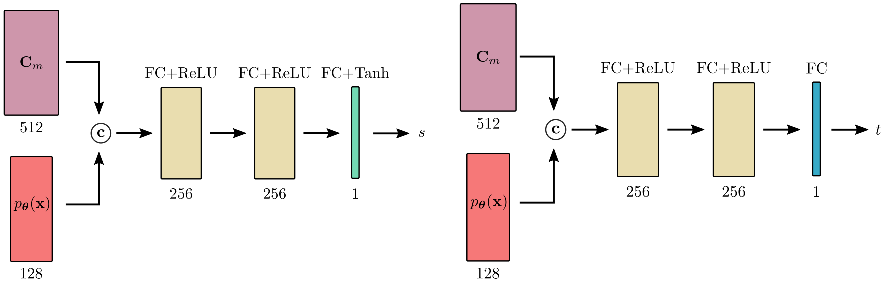


Figure 3.21: **Feature Extractor.**. The feature extractor predicts a vector of per-primitive shape embeddings $\{\mathbf{C}_m\}_{m=1}^M$ conditioned on the input image. Note that depending on the type of the input (e.g. pointcloud, voxel grid), a different feature encoder module needs to be employed.

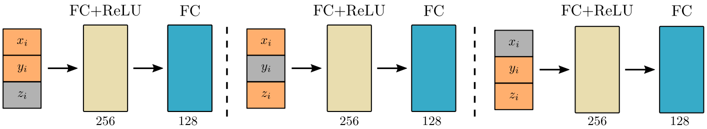
architecture, we remove the final fully connected layer and keep only the feature vector of length 512 after average pooling. Subsequently, we use 2 fully connected layers to map this to the global feature vector $\mathbf{F} \in \mathbb{R}^{256}$. During training, we use the pre-trained batch statistics for normalization. The primitive embedding $\{\mathbf{P}_m\}_{m=1}^M$ is a matrix of size $M \times 256$, that stores a vector per-primitive index that is concatenated with the global image feature $\mathbf{F}$ and outputs a vector of shape embeddings $\{\mathbf{C}_m \in \mathbb{R}^{512}\}_{m=1}^M$ for every primitive. A pictorial representation of the feature extractor is provided in Fig. 3.21.

Conditional Homeomorphism: The conditional homeomorphism is implemented with a Real NVP [46]. It comprises 4 coupling layers and each coupling layer consists of three components: $s_\theta(\cdot)$ that predicts the **scaling factor** s, $t_\theta(\cdot)$ that predicts the **translation amount** t and $p_\theta(\cdot)$ that maps the input 3D point to a higher dimensional space before concatenating it with the per-primitive shape embedding $\mathbf{C}_m$. In particular, $s_\theta(\cdot)$ is a 3-layer MLP with hidden size equal to 256. After each layer we add ReLU non-linearities except for the last layer, where we use a hard tanh with thresholds -10 and 10 (see Fig. 3.22a). Note that we use the tanh to avoid numerical instabilities when computing the exponential of the scaling factor. $t_\theta(\cdot)$ is implemented with the same architecture apart from the tanh activation in the final layer (see Fig. 3.22b). Finally, $p_\theta(\cdot)$ is implemented with a 2-layer MLP with ReLU non-linearities with hidden size 256 and output size 128. An illustration of the components of the conditional homeomorphism is provided in Fig. 3.22.



(a) $s_\theta(\cdot)$ predicts the scaling factor s

(b) $t_\theta(\cdot)$ predicts the translation amount t



(c) $p_\theta(\cdot)$ maps the input point (x_i, y_i, z_i) into a higher dimensional feature space

Figure 3.22: **Conditional Homeomorphism.** The conditional homeomorphism is the backbone of our architecture that given a vector of per-primitive shape embeddings and a set of points on the sphere in the latent space maps them in the primitive space of the target object and vice versa. Here, we visualize $s_\theta(\cdot)$, $t_\theta(\cdot)$ and $p_\theta(\cdot)$ that comprise our *affine coupling layer*.

In particular, we split the dimensions of the input point (x_i, y_i, z_i) in order to scale and translate one of them based on the other two. In practice, this is efficiently implemented via masking one of the dimensions, as illustrated in Fig. 3.22c with gray. For completeness, in equation 3.21 we provide the inverse of the conditional affine coupling layer of (3.5),

$$\begin{aligned} x_i &= x_o \\ y_i &= y_o \\ z_i &= \left(z_o - t_\theta\left(\left[\mathbf{C}_m; p_\theta\left(x_o, y_o\right)\right]\right)\right) \exp\left(-s_\theta\left(\left[\mathbf{C}_m; p_\theta\left(x_o, y_o\right)\right]\right)\right) \end{aligned} \tag{3.21}$$

3.5.2 *Training Protocol*

In all our experiments, we use the Adam optimizer [96] with learning rate $\eta = 10^{-4}$ and no weight decay. For the other hyperparameters of Adam we use the PyTorch defaults. Depending on the number of primitives, we train our model with a different batch size and a different number of iterations.

In particular, for 2 primitives, we use a batch size of 6 for 100k iterations and we aggregate the gradients over 2 batches. For the experiments, with 5 primitives, we use a batch size of 4 for 150k iterations and we again aggregate the gradients over 2 batches. For 8 and 10 primitives, we use a batch size of 4 for 200k iterations and aggregate the gradients over 4 batches. We do not perform any data augmentation.

We weigh the loss terms of (3.11) with 1.0, 0.1, 0.01, 0.1, 0.01. Note that we use smaller weights for $\mathcal{L}_{norm}$ and $\mathcal{L}_{cover}$ since they act as regularizers and we want our model to focus primarily on learning geometrically accurate and semantically meaningful primitives. The impact of each term is discussed in detail in Sec. 3.4.3. The temperature parameter τ in the occupancy $\mathcal{L}_{occ}$ and the overlap loss $\mathcal{L}_{overlap}$ is set to 4×10^{-3}. In addition, the k term for the $\mathcal{L}_{overlap}$ is set to 1.95 since we want to penalize overlapping primitives but ensure that they will be connected. Note that $k = 1$ results in disconnected primitives and $k = 2$ allows primitives to overlap in pairs of two. Hence we select $k = 1.95$ as we empirically observe that it balances connectivity and interpenetration. Finally, we set the k to be equal to 10 for the $\mathcal{L}_{cover}$, as we empirically observe that it leads to good performance.

3.5.3 *Sampling Strategy*

In this section, we discuss our sampling strategy for generating the surface samples $\mathcal{X}_t = \{\mathbf{x}_i, \mathbf{n}_i\}_{i=1}^N$, the occupancy pairs $\mathcal{X}_o = \{\mathbf{x}_i, o_i\}_{i=1}^V$ and the points on the sphere surface $\mathcal{Y}_s = \{\mathbf{y}_j\}_{j=1}^K$.

Surface Samples: We generate samples on the surface of the target mesh by sampling a face with probability proportional to its area and then sampling a point uniformly in that face. We use the corresponding face normals to generate point-normal pairs from the target mesh. During training, we randomly sample $2,000$ points and normals on the target mesh.

Occupancy Pairs: We generate occupancy pairs by sampling $100,000$ points uniformly in the unit cube centered at $(0, 0, 0)$ for each mesh. Subsequently, we compute which of these points lie inside or outside the mesh to generate the occupancy labels. However, because the target objects have a small volume compared to the unit cube, resampling uniformly from the $100,000$ occupancy pairs results in a small amount of points with positive labels. This yields bad reconstructions for parts with small volume such as hands and fingers. To address this, we sample from an unbalanced distribution that, in expectation, results in an equal number of points with positive and negative labels. However, we also compute importance sampling weights in order to reweigh our loss and create an unbiased estimator of the loss with uniform sampling similar to [141]. During training, we sample $5,000$ occupancy pairs.

Sphere Points: In order to generate points on the surface of each primitive, we sample points uniformly on the surface of a sphere with radius r by sampling points from a 3-D isotropic Gaussian and normalizing their length to r. Note that this does not generate uniform samples on the surface of the primitive due to different amounts of stretching and contracting of the sphere by each homeomorphism. However, we empirically observe that this sampling does not impact negatively the training of our model, thus we leave sampling uniformly on the surface of each primitive for future work. During training we sample 200 points for each primitive. In addition, to generate meshes for the predicted primitives, we also need to compute sphere surface points and corresponding faces. We achieve this using the UV parametrization of the sphere [35]. For visualization purposes we use $2,000$ points sampled uniformly in latitude and longitude.

3.6 DISCUSSION

We have presented a novel primitive representation that is more expressive and interpretable in comparison to alternatives that are limited to simpler geometric primitives such as convex shapes. In particular, we posed the task of primitive learning as the task of learning a family of homeomorphic mappings between the 3D shape of a simple genus-zero shape and the 3D shape of the target object. In contrast to prior work that directly predicts the primitive parameters, we employ an INN to fully define each primitive. Utilizing an INN allows us to easily define both the implicit and explicit representation of each part. Our experiments demonstrate that our model yields geometrically accurate and semantically meaningful shape abstractions using only a small number of components. In addition, we show that Neural Parts outperform existing methods, that rely on simpler shapes, both in terms of accuracy and semantic consistency.

While Neural Parts can be learned in an unsupervised fashion, i.e. without any primitive annotations, 3D supervision in the form of a watertight mesh is still required. We believe that it would be desirable to relax this need of supervision and in future work attempt to learn the part-based decomposition using differentiable rendering techniques. Additionally, while Neural Parts do not impose any constraint on the predicted primitives, they still are genus-zero shape. Ideally, we would like to also relax this constraint and be able to also have primitives of arbitrary genus.

Part II

Structure-Aware Part-based Representations

4

LEARNING UNSUPERVISED HIERARCHICAL PART DECOMPOSITION OF 3D OBJECTS

> *The whole is greater than the sum of parts.*
> — Aristotle

Within the first year of their life, humans develop a common-sense understanding of the physical behavior of the world [8]. This understanding relies heavily on the ability to properly reason about the arrangement of objects in a scene. Early works in cognitive science [78, 10, 99] stipulate that the human visual system perceives objects as a hierarchical decomposition of parts. Interestingly, while this seems to be a fairly easy task for the human brain, computer vision algorithms struggle to form such a high-level reasoning, particularly in the absence of supervision.

The structure of a scene is tightly related to the inherent hierarchical organization of its parts. At a coarse level, a scene can be decomposed into objects and at a finer level each object can be represented with parts and these parts with finer parts. Structure-aware representations go beyond part-level geometry and focus on global relationships between objects and object parts. In this work, we propose a structure-aware representation that considers part relationships (Fig. 4.1) and models object parts with multiple levels of abstraction, namely geometrically complex parts are modeled with more components and simple parts are modeled with fewer components. Such a multi-scale representation can be efficiently stored at the required level of detail, namely with less parameters (Fig. 4.2).

Recent breakthroughs in deep learning led to impressive progress in 3D shape extraction by learning a parametric function, implemented as a neural network, that maps an input image to a 3D shape represented as a mesh [113, 68, 86, 199, 211, 134], a pointcloud [53, 151, 2, 84, 186, 211], a voxel grid [14, 34, 57, 159, 162, 180, 208], 2.5D depth maps [87, 73, 142, 47] or an implicit surface [120, 27, 136, 168, 209, 121]. These approaches are mainly focused on reconstructing the geometry of an object, without taking into consideration its constituent parts. This results in non-interpretable reconstructions. To address the lack of interpretability, researchers shifted their attention to representations that employ shape primitives [191, 141, 107, 40, 37, 61, 140]. In Chapter 2 and 3, we introduced two primitive representations

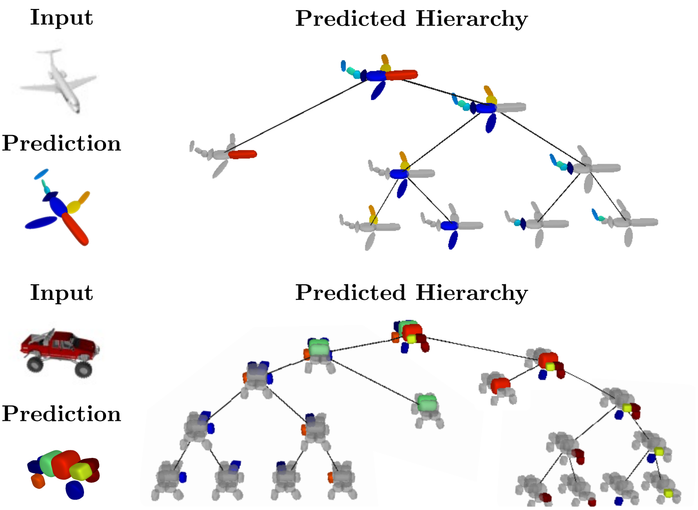


Figure 4.1: **Hierarchical Part Decomposition.** We consider the problem of learning structure-aware representations that go beyond part-level geometry and focus on part-level relationships. Here, we show our reconstruction as an unbalanced binary tree of primitives, given a single RGB image as input. Note that our model does not require any supervision on object parts or the hierarchical structure of the 3D object. We show that our representation is able to model different parts of an object with different levels of abstraction, leading to improved reconstruction quality.

that allow for both geometrically accurate and semantically meaningful shape abstractions. However, while they can recover the part-based object decomposition, they cannot explicitly reason about part relationships.

To address this, in this chapter, we propose a novel neural network architecture that recovers the latent hierarchical layout of an object without structure supervision. In particular, we employ a neural network that learns to recursively partition an object into its constituent parts by building a latent space that encodes both the part-level hierarchy and the part geometries. The predicted hierarchical decomposition is represented as an unbalanced binary tree of primitives. More importantly, this is learned without any supervision neither on the object parts nor their structure. Instead, our model jointly infers these latent variables *during training*.

In summary, we make the following **contributions**: We jointly learn to predict part relationships and per-part geometry without any part-level supervision. The only supervision required for training our model is a watertight mesh of the 3D object. Our structure-aware representa-

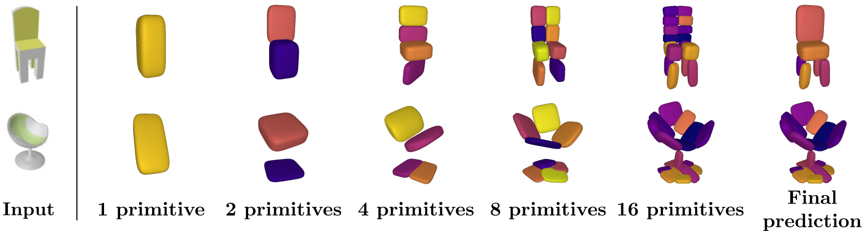


Figure 4.2: **Level of Detail.** Our network represents an object as a tree of primitives. At each depth level d, the target object is reconstructed with 2^d primitives, This results in a representation with various levels of detail. Naturally, reconstructions from deeper depth levels are more detailed. We associate each primitive with a unique color, thus primitives illustrated with the same color correspond to the same object part. Note that the above reconstructions are derived from the *same* model, trained with a maximum number of $2^4 = 16$ primitives. During inference, the network dynamically combines representations from different depth levels to recover the final prediction (last column).

tion yields semantic shape reconstructions that compare favorably to the state-of-the-art 3D reconstruction approach of [120], using significantly less parameters and without any additional post-processing. Moreover, our learned hierarchies have a semantic interpretation, as the same node in the learned tree is consistently used for representing the same object part. Experiments on the ShapeNet [19] and the Dynamic FAUST (D-FAUST) dataset [13] demonstrate the ability of our model to parse objects into structure-aware representations that are more expressive and geometrically accurate compared to approaches that only consider the 3D geometry of the object parts [191, 141, 61, 36]. The code to reproduce the experiments presented in this chapter can be found at https://github.com/paschalidoud/hierarchical_primitives.

4.1 RELATED WORK

We now discuss the most related primitive-based and structure-aware shape representations.

Supervised Structure-Aware Representations: The model presented in this chapter is related to methods that learn structure-aware shape representations that go beyond mere enumeration of object's parts and recover the higher level structural decomposition of objects based on part-level relations [123]. Li et al. [106] represent 3D shapes using a symmetry hierarchy [203] and train a recursive neural network to predict its hierarchical structure. Their network learns a hierarchical organization of bounding boxes and then fills them with voxelized parts. Note that, this model considers supervision in terms of segmentation of objects into their primitive parts. Closely related to [106] is StructureNet [124] which leverages a graph neural network to represent shapes as n-ary graphs. StructureNet considers supervision both in terms of the primitive parameters and the hierarchies. Likewise, Hu et al. [79] propose a supervised model that recovers the 3D structure of a cable-stayed bridge as a binary parsing tree. In contrast our model is unsupervised, i.e., it does not require supervision neither on the primitive parts nor the part relations.

Physics-Based Structure-Aware Representations: The task of inferring higher-level relationships among parts has also been investigated in different settings. Xu et al. [210] recover the object parts, their hierarchical structure and each part's dynamics by observing how objects are expected to move in the future. In particular, each part inherits the motion of its parent and the hierarchy emerges by minimizing the norm of these local displacement vectors. Kipf et al. [100] explore the use of variational autoencoders for learning the underlying interaction among various moving particles. Steenkiste et al. [193] extend the work of [66] on perceptual grouping of pixels and learn an interaction function that models whether objects interact with each other at multiple frames. For both [100, 193], the hierarchical structure emerges from interactions at multiple timestamps. In contrast to [210, 100, 193], our model does not relate hierarchies to motion, thus we do not require multiple frames for discovering the hierarchical structure.

Supervised Primitive-Based Representations: Zou et al. [226] exploit LSTMs in combination with a Mixture Density Network (MDN) to learn a cuboid representation from depth maps. Similarly, Niu et al. [130] employ an RNN that iteratively predicts cuboid primitives as well as their symmetry and connectivity relationships from RGB images. More recently, Li et al. [107] utilize PointNet++ [151] for predicting per-point properties that are subsequently used for estimating the primitive parameters, by solving a series of linear least-squares problems. In contrast to [226, 130, 151], which require supervision in terms of the primitive parameters, our model is learned in an unsupervised fashion. In addition, modelling primitives with superquadrics, allows us to exploit a larger shape vocabulary that is not limited to cubes as in [226, 130] or spheres, cones, cylinders and planes as in [107]. Another line of work, complementary to ours, incorporates the principles of constructive solid geometry (CSG) [104] in a learning framework for shape modeling [173, 51, 187, 114]. These works require rich annotations for the primitive parameters and the sequence of predictions.

Unsupervised Shape Abstraction: Closely related to our model are the works of [191, 141] that employ a convolutional neural network (CNN) to regress the parameters of the primitives that best describe the target object, in an unsupervised manner. Primitives can be cuboids [191] or superquadrics [141] and are learned by minimizing the discrepancy between the target and the predicted shape, by either computing the truncated bidirectional distance [191] or the Chamfer-distance between points on the target and the predicted shape [141]. While these methods learn a flat arrangement of parts, our structure-aware representation decomposes the depicted object into a hierarchical layout of semantic parts. This results in part geometries with different levels of granularity. Our model differs from [191, 141] also wrt. the optimization objective. We empirically observe that for both [191, 141], the proposed loss formulations suffer from various local minima that stem from the nature of their optimization objective. To mitigate this, we use the more robust classification loss proposed in [120, 27, 136] and train our network by learning to classify whether points lie inside or outside the target object. Very recently, [61, 36] explored such a loss for recovering shape elements from 3D objects. Genova et al. [61] leverage a CNN to learn to predict the parameters of a set of axis-aligned 3D Gaussians from a set of depth maps rendered at different viewpoints. Similarly, Deng et al. [36] employ an autoencoder to recover the geometry of an object as a collection of smooth convexes. In contrast to [61, 36], our model goes beyond the local geometry of parts and attempts to recover the underlying hierarchical structure of the object parts.

4.2 METHOD OVERVIEW

In this section, we describe our novel neural network architecture for inferring structure-aware representations. Given an input $\mathbf{I}$ (e.g., RGB image, voxel grid) our goal is to learn a neural network ϕ_θ, which maps the input to a set of primitives that best describe the target object. The target object is represented as a set of pairs $\mathcal{X} = \{(\mathbf{x}_i, o_i)\}_{i=1}^N$, where $\mathbf{x}_i$ corresponds to the location of the i-th point and o_i denotes its label, namely whether $\mathbf{x}_i$ lies inside $(o_i = 1)$ or outside $(o_i = 0)$ the target object. We acquire these N pairs by sampling points inside the bounding box of the target mesh and determine their labels using a watertight mesh of the target object. During training, our network learns to predict shapes that contain all internal points from the target mesh $(o_i = 1)$ and none of the external $(o_i = 0)$. Additional details for our sampling strategy can be found in Sec. 4.4.3 We discuss our sampling strategy in our sup.

Instead of predicting an unstructured set of primitives, we recover a hierarchical decomposition over parts in the form of a *binary tree* of maximum depth D as

$$\mathcal{P} = \{\{p_k^d\}_{k=0}^{2^d-1} \mid d = \{0 \dots D\}\} \tag{4.1}$$

where p_k^d is the k-th primitive at depth d. Note that for the k-th node at depth d, its parent is defined as $p_{\lfloor \frac{k}{2} \rfloor}^{d-1}$ and its two children as p_{2k}^{d+1} and p_{2k+1}^{d+1}.

At every depth level, $\mathcal{P}$ reconstructs the target object with $\{1, 2, \dots, \mathrm{M}\}$ primitives. M is an upper limit to the maximum number of primitives and is equal to 2^D. More specifically, $\mathcal{P}$ is constructed as follows: the root node is associated with the root primitive that represents the entire shape and is recursively split into two nodes (its children) until reaching the maximum depth D. This recursive partition yields reconstructions that recover the geometry of the target shape using 2^d primitives, where d denotes the depth level (see Fig. 4.2). Throughout this chapter, the term *node* is used interchangeably with *primitive* and always refers to the primitive associated with this particular node.

Every primitive is fully described by a set of parameters λ_k^d that define its shape, size and position in 3D space. Since not all objects require the same number of primitives, we enable our model to predict unbalanced trees, i.e. stop recursive partitioning if the reconstruction quality is sufficient. To achieve this our network also regresses a *reconstruction quality* for each primitive denoted as q_k^d. Based on the value of each q_k^d the network dynamically stops the recursive partitioning process resulting in parsimonious representations as illustrated in Fig. 4.1.

4.2.1 *Network Architecture*

Our network comprises three main components: (i) the *partition network* that recursively splits the shape representation into representations of parts, (ii) the *structure network* that focuses on learning the hierarchical arrangement of primitives, namely assigning parts of the object to the primitives at each depth level and (iii) the *geometry network* that recovers the primitive parameters. An overview of the proposed pipeline is illustrated in Fig. 4.3.

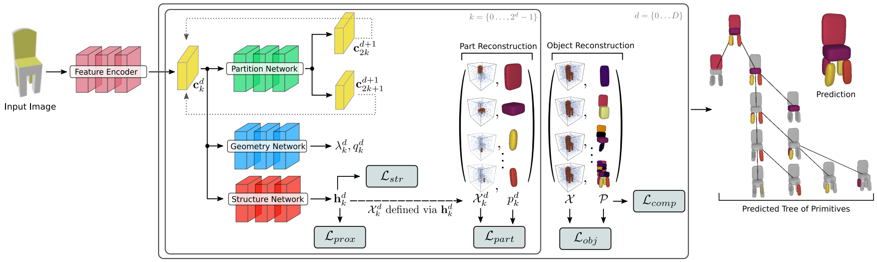


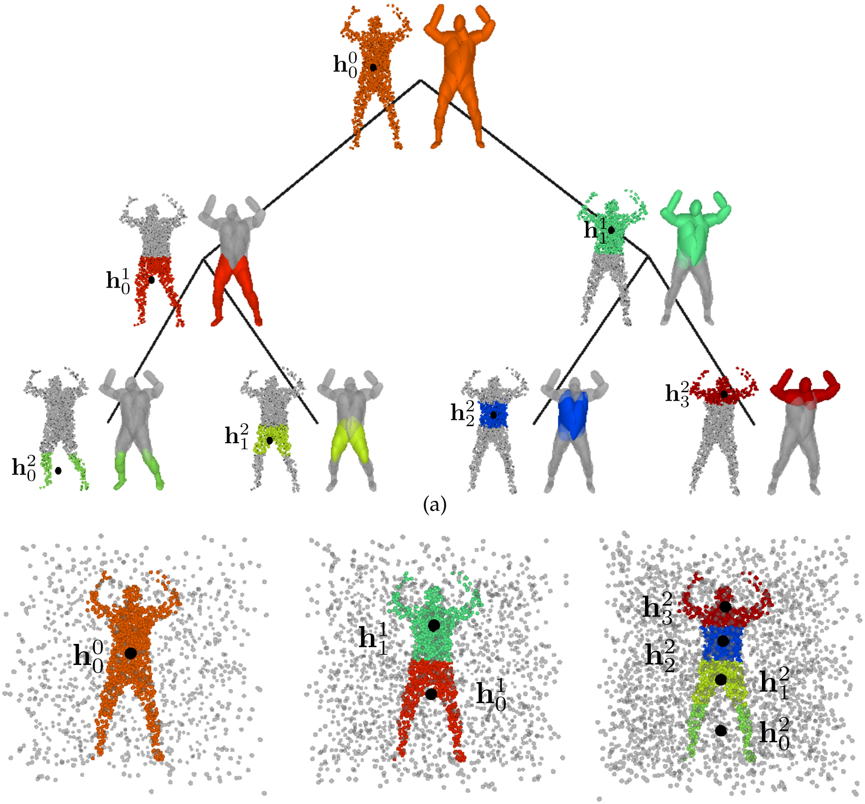
Figure 4.3: **Network Architecture.** Given an input $\mathbf{I}$ (e.g., image, voxel grid), our network predicts a binary tree of primitives $\mathcal{P}$ of maximum depth D. The *feature encoder* maps the input $\mathbf{I}$ into a feature vector $\mathbf{c}_0^0$. Subsequently, the *partition network* splits each feature representation $\mathbf{c}_k^d$ in two $\{\mathbf{c}_{2k}^{d+1}, \mathbf{c}_{2k+1}^{d+1}\}$, resulting in feature representations for $\{1, 2, 4, \ldots, 2^d\}$ primitives where $\mathbf{c}_k^d$ denotes the feature representation for the k-th primitive at depth d. Each $\mathbf{c}_k^d$ is passed to the *structure network* that "assigns" a part of the object to a specific primitive p_k^d. As a result, each p_k^d is responsible for representing a specific part of the target shape, denoted as the set of points $\mathcal{X}_k^d$. Finally, the *geometry network* predicts the primitive parameters λ_k^d and the reconstruction quality q_k^d for each primitive. To compute the reconstruction loss, we measure how well the predicted primitives match the target object (*Object Reconstruction*) and the assigned parts (*Part Reconstruction*). We use plate notation to denote repetition over all nodes k at each depth level d. The final reconstruction is shown on the right.

The first part of our pipeline is a *feature encoder*, implemented with a ResNet-18 [75], ignoring the final fully connected layer. Instead, we only keep the feature vector of length $F = 512$ after average pooling.

Partition Network: The feature encoder maps the input $\mathbf{I}$ to an intermediate feature representation $\mathbf{c}_0^0 \in \mathbb{R}^F$ that describes the root node p_0^0. The partition network implements a function $p_\theta : \mathbb{R}^F \to \mathbb{R}^{2F}$ that recursively partitions the feature representation $\mathbf{c}_k^d$ of node p_k^d into two feature representations, one for each children $\{p_{2k}^{d+1}, p_{2k+1}^{d+1}\}$:

$$p_\theta(\mathbf{c}_k^d) = \{\mathbf{c}_{2k}^{d+1}, \mathbf{c}_{2k+1}^{d+1}\}. \tag{4.2}$$

Each primitive p_k^d is directly predicted from $\mathbf{c}_k^d$ without considering the other intermediate features. This implies that the necessary information for predicting the primitive parameterization is entirely encapsulated in $\mathbf{c}_k^d$ and not in any other intermediate feature representation.



(a)

(b)

Figure 4.4: **Structure Network.** We visualize the centroids $\mathbf{h}_k^d$ and the 3D points $\mathcal{X}_k^d$ that correspond to the estimated part p_k^d for the first three levels of the tree. Fig. 4.4b explains visually Eq. (4.4). We color points based on their closest centroid $\mathbf{h}_k^d$. Points illustrated with the color associated to a part are labeled "internal" ($o = 1$). Points illustrated with gray are labeled "external" ($o = 0$).

Structure Network: Due to the lack of ground-truth supervision in terms of the tree structure, we introduce the structure network that seeks to learn a pseudo-ground truth part-decomposition of the target object. More formally, it learns a function $s_\theta : \mathbb{R}^F \rightarrow \mathbb{R}^3$ that maps each feature representation $\mathbf{c}_k^d$ to $\mathbf{h}_k^d$ a spatial location in $\mathbb{R}^3$.

One can think of each $\mathbf{h}_k^d$ as the (geometric) centroid of a specific part of the target object. We define

$$\mathcal{H} = \{\{\mathbf{h}_k^d\}_{k=0}^{2^d-1} \,|\, d = \{0 \dots D\}\} \tag{4.3}$$

the set of centroids of all parts of the object at all depth levels. From $\mathcal{H}$ and $\mathcal{X}$, we are now able to derive the part decomposition of the target object as the set of points $\mathcal{X}_k^d$ that are internal to a part with centroid $\mathbf{h}_k^d$.

Note that, in order to learn $\mathcal{P}$, we need to be able to partition the target object into 2^d parts at each depth level. At the root level $(d = 0)$, $\mathbf{h}_0^0$ is the centroid of the target object and $\mathcal{X}_0^0$ is equal to $\mathcal{X}$. For $d = 1$, $\mathbf{h}_0^1$ and $\mathbf{h}_1^1$ are the centroids of the two parts representing the target object. $\mathcal{X}_0^1$ and $\mathcal{X}_1^1$ comprise the same points as $\mathcal{X}_0^0$. For the external points, the labels remain the same. For the internal points, however, the labels are distributed between $\mathcal{X}_0^1$ and $\mathcal{X}_1^1$ based on whether $\mathbf{h}_0^1$ or $\mathbf{h}_1^1$ is closer. That is, $\mathcal{X}_0^1$ and $\mathcal{X}_1^1$ each contain more external labels and less internal labels compared to $\mathcal{X}_0^0$. The same process is repeated until we reach the maximum depth.

More formally, we define the set of points $\mathcal{X}_k^d$ corresponding to primitive p_k^d implicitly via its centroid $\mathbf{h}_k^d$:

$$\mathcal{X}_k^d = \left\{ N_k(\mathbf{x}, o) \quad \forall (\mathbf{x}, o) \in \mathcal{X}_{\lfloor \frac{k}{2} \rfloor}^{d-1} \right\} \tag{4.4}$$

Here, $\mathcal{X}_{\lfloor \frac{k}{2} \rfloor}^{d-1}$ denotes the points of the parent. The function $N_k(\mathbf{x}, o)$ assigns each $(\mathbf{x}, o) \in \mathcal{X}_{\lfloor \frac{k}{2} \rfloor}^{d-1}$ to part p_k^d if it is closer to $\mathbf{h}_k^d$ than to $\mathbf{h}_{s(k)}^d$ where $s(k)$ is the sibling of k:

$$N_k(\mathbf{x}, o) = \begin{cases} (\mathbf{x}, 1) & \|\mathbf{h}_k^d - \mathbf{x}\| \leq \|\mathbf{h}_{s(k)}^d - \mathbf{x}\| \land o = 1 \\ (\mathbf{x}, 0) & \text{otherwise} \end{cases} \tag{4.5}$$

Intuitively, this process recursively associates points to the closest sibling at each level of the binary tree where the association is determined by the label o. Fig. 4.4 illustrates the part decomposition of the target shape using $\mathcal{H}$. We visualize each part with a different color.

Geometry Network: The geometry network learns a function $r_\theta : \mathbb{R}^F \rightarrow \mathbb{R}^K \times [0, 1]$ that maps the feature representation $\mathbf{c}_k^d$ to its corresponding primitive parametrization λ_k^d and the reconstruction quality prediction q_k^d:

$$r_\theta(\mathbf{c}_k^d) = \{\lambda_k^d, q_k^d\}. \tag{4.6}$$

4.2.2 *Primitive Parametrization*

For primitives, we use superquadric surfaces. For a detailed analysis of the use of superquadrics as geometric primitives, we refer the reader to [81, 141]. Below, we focus on the properties most relevant to us. For any point $\mathbf{x} \in \mathbb{R}^3$, we can determine whether it lies inside or outside a superquadric using its implicit surface function which is commonly referred to as the *inside-outside function*:

$$f(\mathbf{x};\lambda) = \left(\left(\frac{x}{\alpha_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{\alpha_2}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{\alpha_3}\right)^{\frac{2}{\epsilon_1}} \tag{4.7}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]$ determine the size and $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2]$ the shape of the superquadric. If $f(\mathbf{x};\lambda) = 1.0$, the given point $\mathbf{x}$ lies on the surface of the superquadric, if $f(\mathbf{x};\lambda) < 1.0$ the corresponding point lies inside and if $f(\mathbf{x};\lambda) > 1.0$ the point lies outside the superquadric. To account for numerical instabilities that arise from the exponentiations in (4.7), instead of directly using $f(\mathbf{x};\lambda)$, we follow [81] and use $f(\mathbf{x};\lambda)^{\epsilon_1}$. Finally, we convert the inside-outside function to an *occupancy function*, $g : \mathbb{R}^3 \to [0,1]$:

$$g(\mathbf{x};\lambda) = \sigma\left(s\left(1 - f(\mathbf{x};\lambda)^{\epsilon_1}\right)\right) \tag{4.8}$$

that results in per-point predictions suitable for the classification problem we want to solve. $\sigma(\cdot)$ is the sigmoid function and s controls the sharpness of the transition of the occupancy function. To account for any rigid body motion transformations, we augment the primitive parameters with a translation vector $\mathbf{t} = [t_x, t_y, t_z]$ and a quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]$ [70], which determine the coordinate system transformation $\mathcal{T}(\mathbf{x}) = \mathbf{R}(\lambda)\,\mathbf{x} + \mathbf{t}(\lambda)$. Note that in (4.7), (4.8) we omit the primitive indexes k, d for clarity. Visualizations of (4.8) are provided in Sec. A.2.1.

4.2.3 *Network Losses*

Our optimization objective $\mathcal{L}(\mathcal{P}, \mathcal{H}; \mathcal{X})$ is a weighted sum over four terms:

$$\mathcal{L}(\mathcal{P}, \mathcal{H}; \mathcal{X}) = \mathcal{L}_{str}(\mathcal{H}; \mathcal{X}) + \mathcal{L}_{rec}(\mathcal{P}; \mathcal{X}) + \mathcal{L}_{comp}(\mathcal{P}; \mathcal{X}) + \mathcal{L}_{prox}(\mathcal{P}) \tag{4.9}$$

Structure Loss: Using $\mathcal{H}$ and $\mathcal{X}$, we can decompose the target mesh into a hierarchy of disjoint parts. Namely, each $\mathbf{h}_k^d$ implicitly defines a set of points $\mathcal{X}_k^d$ that describe a specific part of the object as described in (4.4). To quantify how well $\mathcal{H}$ clusters the input shape $\mathcal{X}$ we minimize the sum of squared distances, similar to classical k-means:

$$\mathcal{L}_{str}(\mathcal{H};\mathcal{X}) = \sum_{h_k^d \in \mathcal{H}} \frac{1}{2^d - 1} \sum_{(\mathbf{x},o)\in\mathcal{X}_k^d} o \, \|\mathbf{x} - \mathbf{h}_k^d\|_2 \tag{4.10}$$

Note that for the loss in (4.10), we only consider gradients with respect to $\mathcal{H}$ as $\mathcal{X}_k^d$ is implicitly defined via $\mathcal{H}$. This results in a procedure resembling Expectation-Maximization (EM) for clustering point clouds, where computing $\mathcal{X}_k^d$ is the expectation step and each gradient updated corresponds to the maximization step. In contrast to EM, however, we minimize this loss across all instances of the training set, leading to parsimonious but consistent shape abstractions. An example of this clustering process performed at training-time is shown in Fig. 4.4.

Reconstruction Loss: The reconstruction loss measures how well the predicted primitives match the target shape. Similar to [61, 36], we formulate our reconstruction loss as a binary classification problem, where our network learns to predict the surface boundary of the predicted shape by classifying whether points in $\mathcal{X}$ lie inside or outside the target object. To do this, we first define the occupancy function of the predicted shape at each depth level. Using the occupancy function of each primitive defined in (4.8), the occupancy function of the overall shape at depth d becomes:

$$G^d(\mathbf{x}) = \max_{k\in 0\dots 2^d-1} g_k^d\left(\mathbf{x};\lambda_k^d\right) \tag{4.11}$$

Note that (4.11) is simply the union of the per-primitive occupancy functions. We formulate our reconstruction loss wrt. the object and wrt. each part of the object as follows

$$\mathcal{L}_{rec}(\mathcal{P};\mathcal{X}) = \sum_{(\mathbf{x},o)\in\mathcal{X}} \sum_{d=0}^{D} L\left(G^d(\mathbf{x}), o\right) + \tag{4.12}$$

$$\sum_{d=0}^{D} \sum_{k=0}^{2^d-1} \sum_{(\mathbf{x},o)\in\mathcal{X}_k^d} L\left(g_k^d\left(\mathbf{x};\lambda_k^d\right), o\right) \tag{4.13}$$

where $L(\cdot)$ is the binary cross entropy loss. The first term is an *object reconstruction loss* (4.12) and measures how well the predicted shape at each depth level matches the target shape. The second term (4.13) which we refer to as *part reconstruction loss* measures how accurately each primitive

p_k^d matches the part of the object it represents, defined as the point set $\mathcal{X}_k^d$. Note that the *part reconstruction loss* enforces non-overlapping primitives, as $\mathcal{X}_k^d$ are non-overlapping by construction. We illustrate our reconstruction loss in Fig. 4.3.

Compatibility Loss: This loss measures how well our model is able to predict the expected *reconstruction quality* q_k^d of a primitive p_k^d. A standard metric for measuring the reconstruction quality is the Intersection over Union (IoU). We therefore task our network to predict the *reconstruction quality* of each primitive p_k^d in terms of its IoU wrt. the part of the object $\mathcal{X}_k^d$ it represents:

$$\mathcal{L}_{comp}(\mathcal{P};\mathcal{X}) = \sum_{d=0}^{\mathcal{D}} \sum_{k=0}^{2^d-1} \left(q_k^d - \text{IoU}(p_k^d, \mathcal{X}_k^d)\right)^2 \tag{4.14}$$

During inference, q_k^d allows for further partitioning primitives whose IoU is below a threshold q_{th} and to stop if the reconstruction quality is high (the primitive fits the object part well). As a result, our model predicts an unbalanced tree of primitives where objects can be represented with various number of primitives from 1 to 2^D. This results in parsimonious representations where simple parts are represented with fewer primitives. We empirically observe that the threshold value q_{th} does not significantly affect our results, thus we empirically set it to 0.6. During training, we do not use the predicted reconstruction quality q_k^d to dynamically partition the nodes but instead predict the full tree.

Proximity Loss: This term is added to counteract vanishing gradients due to the sigmoid in (4.8). For example, if the initial prediction of a primitive is far away from the target object, the reconstruction loss will be large while its gradients will be small. As a result, it is impossible to "move" this primitive to the right location. Thus, we introduce a proximity loss which encourages the center of each primitive p_k^d to be close to the centroid of the part it represents:

$$\mathcal{L}_{prox}(\mathcal{P}) = \sum_{d=0}^{D} \sum_{k=0}^{2^d-1} \|\mathbf{t}(\lambda_k^d) - \mathbf{h}_k^d\|_2 \tag{4.15}$$

where $\mathbf{t}(\lambda_k^d)$ is the translation vector of the primitive p_k^d and $\mathbf{h}_k^d$ is the centroid of the part it represents. We demonstrate the vanishing gradient problem in Sec. 4.4.4.2.

4.3 EXPERIMENTAL EVALUATION

In this section, we provide evidence that our structure-aware representation yields semantic shape abstractions while achieving competitive (or even better results) than various state-of-the-art shape reconstruction methods, such as [120]. Moreover, we also investigate the quality of the learned hierarchies and show that the use of our structure-aware representation yields semantic scene parsings.

Datasets: First, we use the ShapeNet [19] subset of Choy et al. [34], training our model using the same image renderings and train/test splits as Choy et al. Furthermore, we also experiment with the Dynamic FAUST (D-FAUST) dataset [13], which contains meshes for 129 sequences of 10 humans performing various tasks, such as "running", "punching" or "shake arms". For each sequence, we filter out the first 20 frames that contain the unnatural "neutral pose" necessary for calibration purposes. We randomly divide these sequences into training (91), test (29) and validation (9).

Baselines: Closely related to our model are the shape parsing methods of [191] and [141] that employ cuboids and superquadric surfaces as primitives. We refer to [141] as SQs and we evaluate using their publicly available code[1]. Moreover, we also compare to the Structured Implicit Function (SIF) [61] that represent the object's geometry as the isolevel of the sum of a set of Gaussians and to the CvxNets [36] that represent the object parts using smooth convex shapes. Finally, we also report results for OccNet [120], which is the state-of-the-art implicit shape reconstruction technique. Note that in contrast to us, [120] does not consider part decomposition or any form of latent structure.

Evaluation Metrics: We evaluate our model and our baselines using the volumetric Intersection over Union (IoU) and the Chamfer-L_1 distance. Note that as our method does not predict a single mesh, we sample points from each primitive proportionally to its area, such that the total number of sampled points from all primitives is equal to 100k. For a fair comparison, we do the same for [191, 141]. Below, we discuss in detail the computation of the volumetric IoU and the Chamfer-L_1.

Volumetric IoU is defined as the quotient of the volume of the intersection of the target $S_{target}$and the predicted S_{pred} mesh and the volume of their union. We obtain unbiased estimates of the volume of the intersection and the union by randomly sampling 100k points from the bounding volume

---

[1] https://superquadrics.com

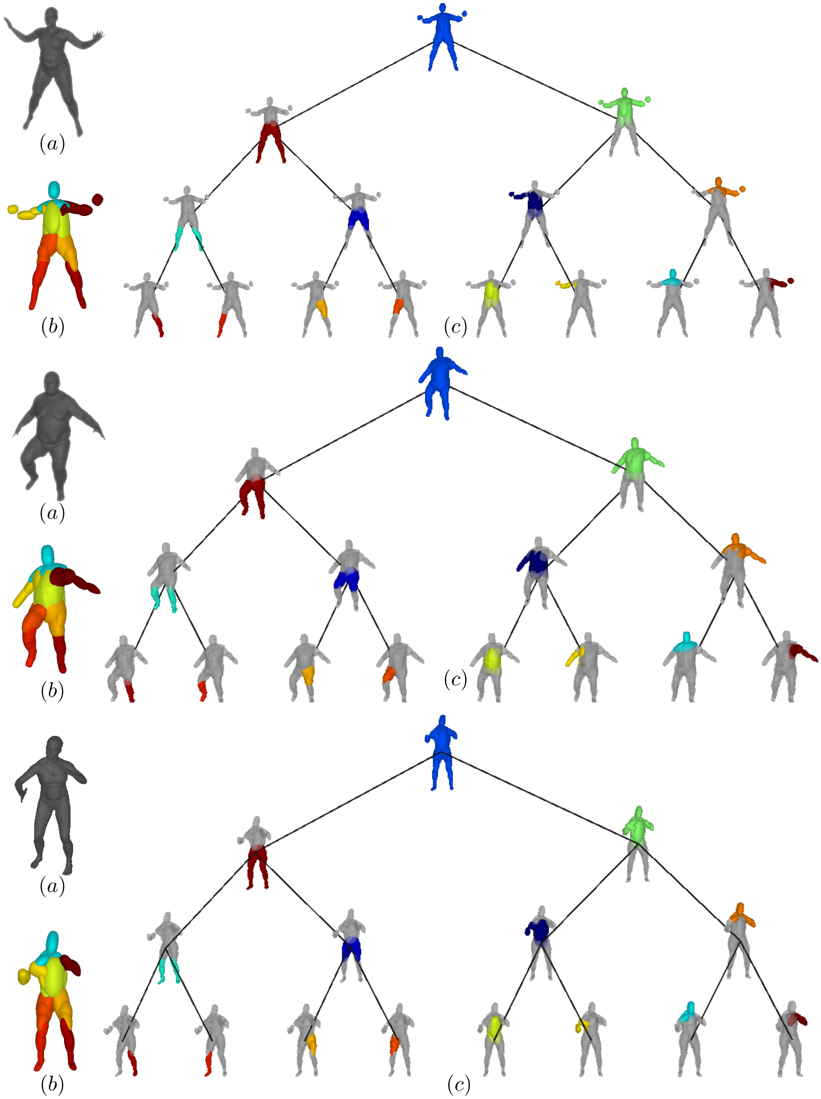


Figure 4.5: **Predicted Hierarchies on D-FAUST.** We visualize the input RGB image (a), the prediction (b) and the predicted hierarchy (c). We associate each primitive with a color and we observe that our network learns semantic mappings of body parts across different articulations, e.g. node $(3, 3)$ is used for representing the upper part of the left leg, whereas node $(1, 1)$ is used for representing the upper body.

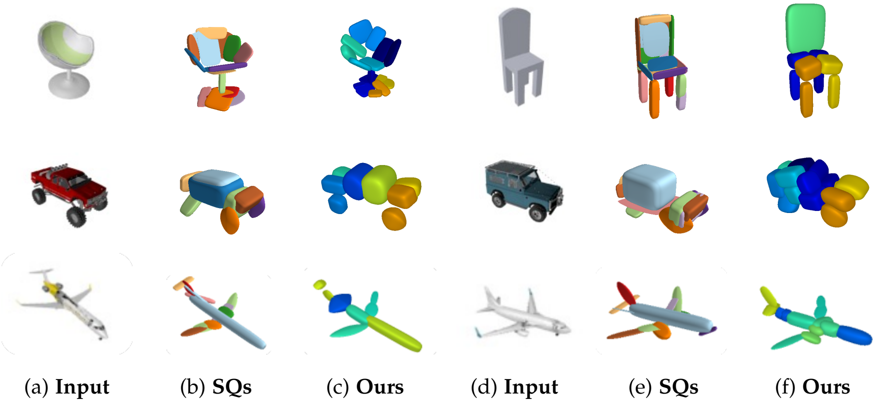


Figure 4.6: **Single Image 3D Reconstruction.** The input image is shown in (a, d), the other columns show the results of our method (c, f) compared to [141] (b, e).

and determining if the points lie inside or outside the target / predicted mesh,

$$\text{IoU}(S_{pred}, S_{target}) = \frac{|\ V(S_{pred} \cap S_{target})\ |}{|\ V(S_{pred} \cup S_{target})\ |} \tag{4.16}$$

where $V(.)$ is a function that computes the volume of a mesh.

We obtain an unbiased estimator of the Chamfer-L_1 distance by sampling 100k points on the surface of the target S_{target} and the predicted S_{pred} mesh. We denote $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ the set of points sampled on the surface of the target mesh and $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^{N}$ the set of points sampled on the surface of the predicted mesh. We compute the Chamfer-L_1 as follows:

$$D_{\text{chamfer}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{N} \sum_{\mathbf{x}_i \in \mathcal{X}} \min_{\mathbf{y}_j \in \cup \mathcal{Y}} \|\mathbf{x}_i - \mathbf{y}_j\| + \frac{1}{N} \sum_{\mathbf{y}_i \in \cup \mathcal{Y}} \min_{\mathbf{x}_j \in \mathcal{X}} \|\mathbf{y}_i - \mathbf{x}_j\| \tag{4.17}$$

The first term of (4.17) measures the *completeness* of the predicted shape, namely how far is on average the closest predicted point from a ground-truth point. The second term measures the *accuracy* of the predicted shape, namely how far on average is the closest ground-truth point from a predicted point. To ensure a fair comparison with our baselines, we use the evaluation code of [120] for the estimation of both the Volumetric IoU and the Chamfer-L_1.

	Chamfer-L_1					IoU				
Category	OccNet [120]	SQs [141]	SIF [61]	CvxNets [36]	Ours	OccNet [120]	SQs [141]	SIF [61]	CvxNets [36]	Ours
airplane	0.147	0.122	**0.065**	0.093	0.175	0.571	0.456	0.530	**0.598**	0.529
bench	0.155	**0.114**	0.131	0.133	0.153	0.485	0.202	0.333	**0.461**	0.437
cabinet	0.167	**0.087**	0.102	0.160	**0.087**	0.733	0.110	0.648	**0.709**	0.658
car	0.159	0.117	**0.056**	0.103	0.141	0.737	0.650	0.657	0.675	**0.702**
chair	0.228	0.138	0.192	0.337	**0.114**	0.501	0.176	0.389	0.491	**0.526**
display	0.278	**0.106**	0.208	0.223	0.137	0.471	0.200	0.491	0.576	**0.633**
lamp	0.479	0.189	0.454	0.795	**0.169**	0.371	0.189	0.260	0.311	**0.441**
speaker	0.300	0.132	0.253	0.462	**0.108**	0.647	0.136	0.577	0.620	**0.660**
rifle	0.141	0.127	**0.069**	0.106	0.203	0.474	**0.519**	0.463	0.515	0.435
sofa	0.194	**0.106**	0.146	0.164	0.128	0.680	0.122	0.606	0.677	**0.693**
table	0.189	**0.110**	0.264	0.358	0.122	0.506	0.180	0.372	0.473	**0.491**
phone	0.140	0.112	**0.095**	0.083	0.149	0.720	0.185	0.658	0.719	**0.770**
vessel	0.218	0.125	**0.108**	0.173	0.178	0.530	0.471	0.502	0.552	**0.570**
mean	0.215	**0.122**	0.165	0.245	0.143	0.571	0.277	0.499	0.567	**0.580**

Table 4.1: **Single Image Reconstruction on ShapeNet.** Quantitative evaluation of our method against OccNet [120] and primitive-based methods with superquadrics [141] (SQs), SIF [61] and CvxNets [36]. We report the volumeteric IoU (higher is better) and the Chamfer-L_1 distance (lower is better) wrt. the ground-truth mesh.

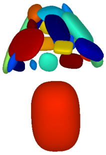
4.3.1 *3D Reconstruction on ShapeNet*

We now evaluate the proposed model on the ShapeNet dataset using three different tasks: (a) RGB single-view 3D reconstruction (b) volumetric 3D reconstruction (i.e., using a voxel representation as input) and (c) interpretability of the representation. For the first experiment, we report qualitative and quantitative results of our model and we compare against various state-of-the-art methods on the task of single view 3D reconstruction. Subsequently, we compare our method wrt. to [191, 141] that were originally introduced for volumetric 3D reconstruction and finally we empirically show that our model leads to semantic and interpretable representations that are consistent across objects of the same class.

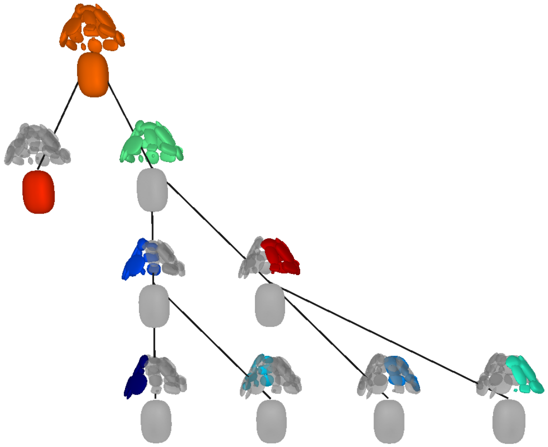
4.3.1.1 *Single Image 3D Reconstruction*

We evaluate our model on the single-view 3D reconstruction task and compare against various state-of-the-art methods. We follow the standard experimental setup and train a single model for the 13 ShapeNet objects. Both our model and [141] are trained for a maximum number of 64 primitives ($D = 6$). For SIF [61] and CvxNets [36] the reported results are

(a) **Input**

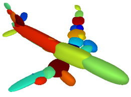


(b) **Prediction**

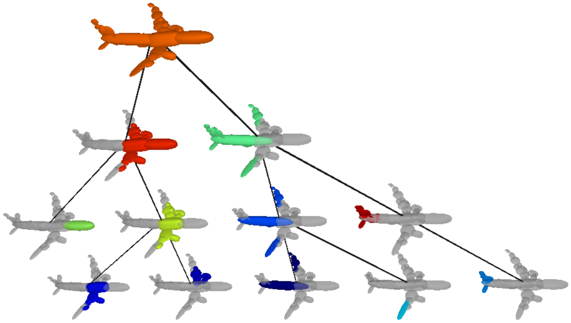


(c) **Predicted Hierarchy**

(d) **Input**



(e) **Prediction**

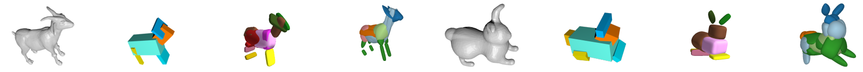


(f) **Predicted Hierarchy**

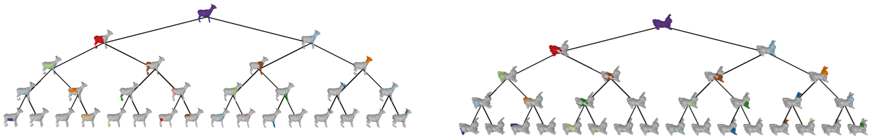
Figure 4.7: **Predicted Hierarchies on ShapeNet**. Our model recovers the geometry of an object as an unbalanced hierarchy over primitives, where simpler parts (e.g. base of the lamp) are represented with few primitives and more complex parts (e.g. wings of the plane) with more.

computed using 50 shape elements. Note, that both [141] and our method use significantly fewer than 50 primitives on average.

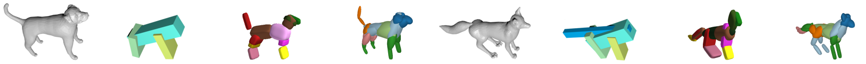
The quantitative results are reported in Tab. 4.1. We observe that our model outperforms the primitive-based baselines in terms of the IoU as well as the OccNet [120] for the majority of objects (7/13). Regarding Chamfer-L_1, our model is the second best amongst primitive representations, as



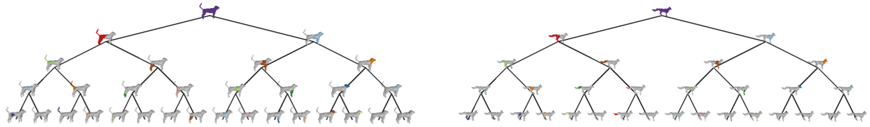
(a) **Input** (b) [191] (c) [141] (d) **Ours** (e) **Input** (f) [191] (g) [141] (h) **Ours**



(i) **Predicted Hierarchy** (j) **Predicted Hierarchy**



(k) **Input** (l) [191] (m) [141] (n) **Ours** (o) **Input** (p) [191] (q) [141] (r) **Ours**



(s) **Predicted Hierarchy** (t) **Predicted Hierarchy**

Figure 4.8: **Volumetric Reconstruction.** We note that our reconstructions are geometrically more accurate. In contrast to [141], our model yields reconstructions where the legs of the animals are not connected. Furthermore, our model accurately captures the ears and tails of the different animals.

[141] is optimized for this metric. This also justifies that [141] performs worse in terms of IoU. While our model performs on par with existing state-of-the-art primitive representations in terms of Chamfer-L_1, it also recovers hierarchies, which none of our baselines do.

A qualitative comparison of our model with SQs [141] is depicted in Fig. 4.6. Fig. 4.7 visualizes the learned hierarchy for this model. We observe that our model recovers unbalanced binary trees that decompose a 3D object into a set of parts.

	IoU	Chamfer-L_1
SQs [141]	0.608	0.189
Ours	**0.699**	**0.098**

Table 4.2: **Single Image Reconstruction on D-FAUST.** We report the volumetric IoU and the Chamfer-L1 wrt. the ground-truth mesh for our model and [141].

4.3.2 *Volumetric Reconstruction*

Our model is closely related to the works of Tulsiani et al. [191] and Paschalidou et al. [141]. Both [191, 141] were originally introduced using a binary occupancy grid as an input to their model, thus we also compare our model with [191, 141] using a voxelized input of size $32 \times 32 \times 32$. We evaluate the modelling accuracy of these three methods on the *animal* class of the ShapeNet dataset. To ensure a fair comparison, we use the feature encoder proposed in [191] for all three. A qualitative evaluation is provided in Fig. 4.8.

Our model yields more detailed reconstructions compared to [191, 141]. For example, in our reconstructions the legs of the animals are not connected and the tails better capture the geometry of the target shape. Again, we observe that our network predicts semantic hierarchies, where the same node is used for representing the same part of the animal.

4.3.3 *3D Reconstruction on D-FAUST*

We also report results on the Dynamic FAUST (D-FAUST) dataset [13], which is very challenging due to the fine structure of the human body. We evaluate our model on the single-view 3D reconstruction task and compare with [141]. Both methods are trained for a maximum number of 32 primitives ($D = 5$). Fig. 4.5 illustrates the predicted hierarchy on different humans from the test set. We note that the predicted hierarchies are indeed semantic, as the same nodes are used for modelling the same part of the human body. Fig. 4.9 compares the predictions of our model with SQs. We observe that while our baseline yields more parsimonious abstractions, their level of detail is limited. On the contrary, our model captures the geometry of the human body with more detail. This is also validated quantitatively, from Tab. 4.2. Note that in contrast to ShapeNet, D-FAUST does not contain long, thin (e.g. legs of tables, chairs) or hollow parts

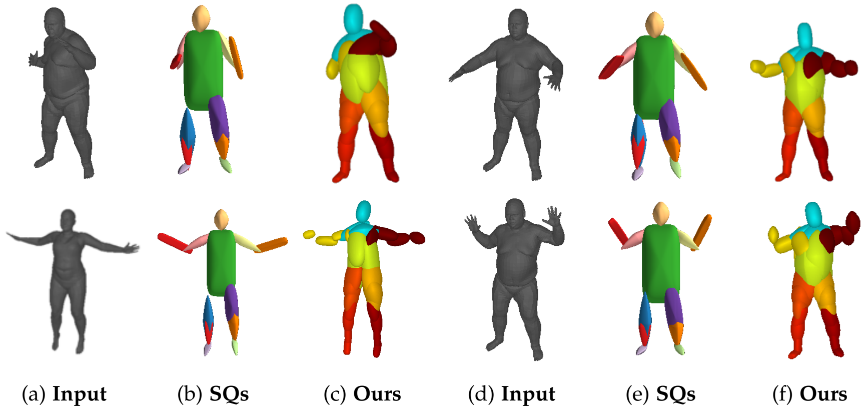


Figure 4.9: **Single Image 3D Reconstruction**. Qualitative comparison of our reconstructions (c, f), to [141] that does not consider any form of structure (b, e). The input RGB image is shown in (a, d). Note how our representation yields geometrically more accurate reconstructions, while being semantic, e.g., the primitive colored in blue consistently represents the head of the human while the primitive colored in orange captures the left thigh.

(e.g. cars), thus optimizing for either Chamfer-L1 or IoU leads to similar results. Hence, our method outperforms [141] also in terms of Chamfer-L1, despite the fact that [141] is optimized for this metric. Due to lack of space, we only illustrate the predicted hierarchies up to the fourth depth level. The full hierarchies are provided in the Fig. B.5+Fig. B.6.

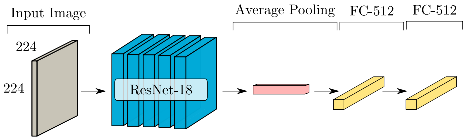
4.4 IMPLEMENTATION DETAILS

In this section, we provide a detailed description of our network architecture. We then describe our sampling strategy and provide details on the metrics we use both for training and testing. Finally, we show how various components influence the performance of our model on the single-view 3D reconstruction task.

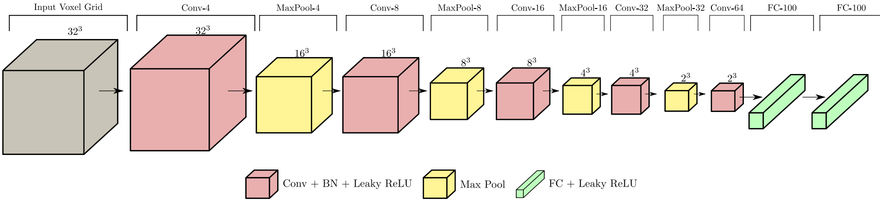
4.4.1 *Network Architecture*

Here we describe the architecture of each individual component of our model, shown in Fig. 4.3.

Feature Encoder: The feature encoder depends on the type of the input, namely whether it is an image or a binary occupancy grid. For the single view 3D reconstruction task, we use a ResNet-18 architecture [75] (Fig. 4.10a), which was pretrained on the ImageNet dataset [38]. From the original design, we ignore the final fully connected layer keeping only the feature vector of length $F = 512$ after average pooling. For the volumetric 3D reconstruction task, where the input is a binary occupancy grid, we use the feature encoder proposed in [141](Fig. 4.10b). Note that the feature encoder is used as a generic feature extractor from the input representation.



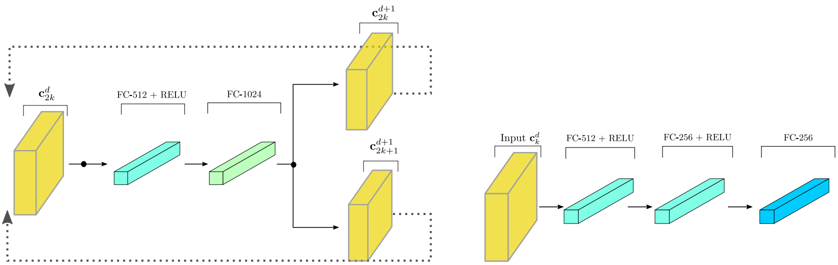
(a) Single-view 3D Reconstruction



(b) Volumetric 3D Reconstruction

Figure 4.10: **Feature Encoder Architectures.** Depending on the type of the input, we employ two different network architectures. (a) For the single view 3D reconstruction task we use a ResNet-18 architecture [75] (b) For a binary occupancy grid as an input, we leverage the network architecture of [141].

Partition Network: The partition network implements a function $p_\theta : \mathbb{R}^F \rightarrow \mathbb{R}^{2F}$ that recursively partitions the feature representation $\mathbf{c}_k^d$ of node p_k^d into two feature representations, one for each child $\{p_{2k}^{d+1}, p_{2k+1}^{d+1}\}$. The partition network (Fig. 4.11a) comprises two fully connected layers followed by RELU non linearity.



(a) Partition Network

(b) Structure Network

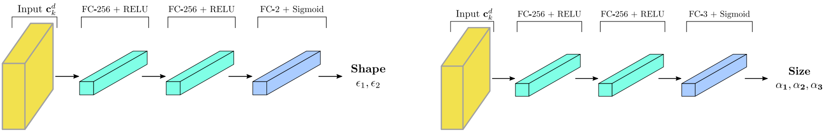
Figure 4.11: **Network Architecture Overview.** The *partition network* (4.11a) is simply one hidden layer fully connected network with RELU non linearity. The gray dotted lines indicate the recursive partition of the feature representation. Similarly, the *structure network* (4.11b) consists of two fully connected layers followed by RELU non linearity.

Structure Network: The structure network maps each feature representation $\mathbf{c}_k^d$ to $\mathbf{h}_k^d$ a spatial location in $\mathbb{R}^3$. The structure network (Fig. 4.11b) consists of two fully connected layers followed by RELU non linearity.

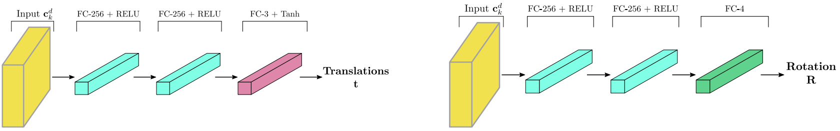
Geometry Network: The geometry network learns a function $r_\theta : \mathbb{R}^F \rightarrow \mathbb{R}^K \times [0, 1]$ that maps the feature representation $\mathbf{c}_k^d$ to its corresponding primitive parametrization λ_k^d and the reconstruction quality prediction q_k^d. In particular, the geometry network consists of five regressors that predict the parameters of the superquadrics (size **ff**, shape **ffl** and pose as translation **t** and rotation **R**) in addition to the reconstruction quality q_k^d. Fig. 4.12 presents the details of the implementation of each regressor.

4.4.2 *Training*

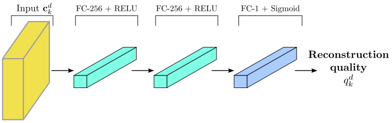
In all our experiments, we use the Adam optimizer [96] with learning rate 0.0001 and no weight decay. For other hyperparameters of Adam we use the PyTorch defaults. We train all models with a batch size of 32 for 40k iterations. We do not perform any additional data augmentation. We weigh the loss terms of (4.9) with 0.1, 0.01, 0.01 and 0.1 respectively, in order to enforce that during the first stages of training the network will focus primarily on learning the hierarchical decomposition of the 3D shape ($\mathcal{L}_s + \mathcal{L}_p$). In this way, after the part decomposition is learned, the network also focuses on the part geometries ($\mathcal{L}_r$). We also experimented with a two-stage optimization scheme, where we first learn the hierarchical



(a) Shape (b) Size



(c) Translation (d) Rotation



(e) Reconstruction quality

Figure 4.12: **Geometry Network.** We detail the specifics of each regressor for predicting the primitive parameters λ_k^d and the reconstruction quality q_k^d.

part decomposition and then learn the hierarchical representation, but we observed that this made learning harder.

4.4.3 *Sampling Strategy*

Sampling a point inside the target mesh has a probability proportional to the volume of the mesh. This yields bad reconstructions for thin parts of the object, such as legs of chairs and wings of aeroplanes. In addition, biasing the sampling towards the points inside the target mesh, results in worse reconstructions as also noted in [120]. To address the first issue (properly reconstructing thin parts), we use an unbalanced sampling distribution that, in expectation, results in sampling an equal number of points inside and outside the target mesh. To counter the second (biased sampling), we construct an unbiased estimator of the loss by weighing the per-point loss inversely proportionally to its sampling probability. We refer to our sampling strategy as *unbiased importance sampling*. Note that throughout all our experiments, we sample 10k points in the bounding box of the target mesh using our sampling strategy.

	IoU	Chamfer-L_1
Uniform	0.383	0.073
Biased	0.351	0.041
Importance	**0.491**	0.073

Table 4.3: Sampling strategy

	IoU	Chamfer-L_1
Importance 2k	0.370	0.074
Importance 5k	0.380	0.076
Importance 10k	**0.491**	**0.073**

Table 4.4: Number of sampled points.

Table 4.5: **Sampling Strategy.** We evaluate the performance of our model while varying the sampling scheme and the number of the sampled points inside the bounding box of the target mesh. We report the volumetric IoU (higher is better) and the Chamfer distance (lower is better) on the test set of the "chair category".

4.4.4 *Empirical Analysis of Loss Formulation*

In this section, we investigate the impact of how various components of our model affect the performance on the single-image 3D reconstruction task.

4.4.4.1 *Impact of Sampling Strategy*

We first discuss how the sampling strategy affects the performance of our model. Towards this goal, we evaluate our model on the single-view 3D reconstruction task using three different sampling strategies: (a) uniform sampling in the bounding box that contains the target object (b) biased sampling (namely sampling an equal number of points inside and outside the target mesh without reweighing) and (c) unbiased importance sampling as described in Sec. 4.4.3. All models are trained on the "chair" object category of ShapeNet using the same network architecture, the same number of sampled points ($N =$10k) and the same maximum number of primitives ($D = 16$). The quantitative results on the test set of the "chair" category are shown in Table 4.5. We observe that the proposed importance sampling strategy achieves the best results in terms of IoU.

Furthermore, we also examine the impact of the number of sampled points on the performance of our model. In particular, we train our model on the "chair" category while varying the number of sampled points inside the bounding box that contains the target mesh. As expected, increasing the number of sampled points results in an improvement in reconstruction quality. We empirically found that sampling 10k points results in the best compromise between training time and reconstruction performance.

4.4.4.2 *Impact of Proximity loss*

In this section, we explain empirically the vanishing gradient problem that emerges from the use of the sigmoid in the occupancy function of (4.8). To this end, we train two variants of our model, one with and without the proximity loss of (4.15). For this experiment, we train both variants on D-FAUST for the single image 3D reconstruction task. Both models are trained for a maximum number of 32 primitives and $s = 10$ and for the same number of iterations.

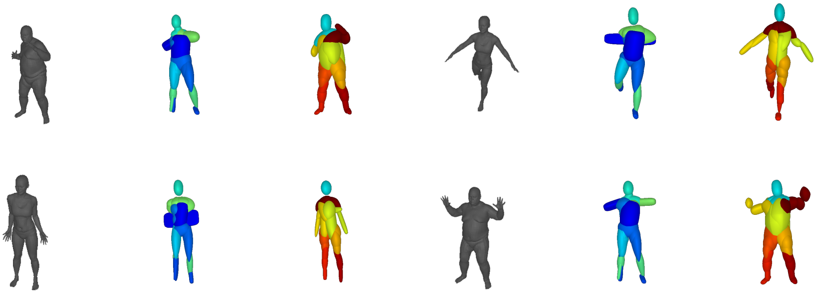

(a) **Input** (b) **without** (c) **Ours** (d) **Input** (e) **without** (f) **Ours**

Figure 4.13: **Impact of Proximity Loss.** We visualize the predictions of two two models, one trained with (**Ours**) and one **without** the proximity loss term. On the left, we visualize the input RGB image (a, d), in the middle the predictions without the proximity loss (b,c) and on the right the predictions of our model with this additional loss term.

Fig. 4.13 illustrates the predictions of both variants. We remark that the predictions of the model that was trained without the proximity loss are less accurate. Note that due to the vanishing gradient problem, the network is not able to properly "move" primitives and as a result, instead of reconstructing the hands of the humans using two or four primitives, the network uses only one. Interestingly, the reconstructions in some cases e.g. (e) do not even capture the human shape properly. However, even though the reconstruction quality is bad, the network is not able to fix it because the gradients of the reconstruction loss are small (even though the reconstruction loss itself is high). This is also validated quantitatively, as can be observed from Table 4.6.

	IoU	Chamfer-L_1
Ours w/o proximity loss	0.605	0.171
Ours	**0.699**	**0.098**

Table 4.6: **Impact of Proximity Loss.** We investigate the impact of the proximity loss. We report the volumetric IoU and the Chamfer distance for two variants of our model, one with and without the proximity loss term.

4.5 DISCUSSION

We have proposed a learning-based approach that jointly predicts part relationships together with per-part geometries in the form of a binary tree of primitives without requiring any part-level annotations during training. Our model yields geometrically accurate shape abstractions that outperform existing part-based techniques while performing competitively with more flexible implicit shape representations. Our formulation allows us to recover higher-level relationships between parts such as spatial proximity, however we believe that a more sophisticated mechanism that would automatically yield semantic components would be more desirable.

Part III

Structure-Aware Object-based Representations for Scene Synthesis

5

ATISS: AUTOREGRESSIVE TRANSFORMERS FOR INDOOR SCENE SYNTHESIS

What I cannot create, I do not understand.
— Richard Feyman

We demonstrated in Part I, that we can learn to recover the 3D object geometry as a set of semantically meaningful parts without any part level supervision. Subsequently, in Part II, we presented a structure-aware representation for 3D objects that jointly reasons about the part geometries and their hierarchical layout in the space and we demonstrated that considering the arrangement of parts facilitates learning geometrically accurate reconstruction. In this chapter, we focus our attention on 3D scenes and instead of considering the arrangement of primitives that best describe a 3D object, we seek to develop a generative model that synthesizes scenes by plausibly arranging objects in an indoor environment.

Generating synthetic 3D content that is both realistic and diverse is a long-standing problem in computer vision and graphics. In the last decade, there has been increased demand for tools that automate the creation of 3D artificial environments for applications like video games and AR/VR, as well as general 3D content creation [216, 20, 217]. These tools can also synthesize data to train computer vision models, avoiding expensive and laborious annotations. Generative models [97, 65, 46, 98, 196] have demonstrated impressive results on synthesizing photorealistic images [32, 15, 89, 33, 90] and intelligible text [154, 16], and are beginning to be adopted for the generation of 3D environments.

Recent works proposed to solve the scene synthesis task by incorporating procedural modeling techniques [153, 149, 88, 41] or by generating scene graphs with generative models [109, 197, 223, 117, 150, 221, 218, 94, 44], whose nodes are associated with objects in the scene and edges model relationships between them. Procedural modeling requires specifying a set of rules for the scene formation process, but acquiring these rules is a time-consuming task, requiring skills of experienced artists. Similarly, graph-based approaches require scene graph annotations, which may be laborious to obtain.

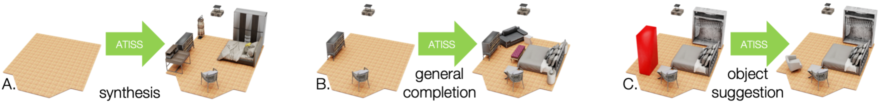


Figure 5.1: **Motivation** In addition to fully automatic layout synthesis (A), our formulation in terms of unordered sets of objects allows our model to be used for novel interactive applications with versatile user control: scene completion given any number of existing furniture pieces of any class pinned to a specific location by the user (B), and object suggestions with user-provided constraints (object centroid constraint shown in red) (C).

Another line of research utilizes CNN-based [198, 163] and transformer-based [202] architectures to generate rooms by autoregressively selecting and placing objects in a scene, i.e. one after the other. These approaches represent scenes as ordered sequences of objects. Typically, the ordering is defined using the spatial arrangement of objects in a room (e.g. left-to-right) [85] or the object class frequency (e.g. most to least probable) [163, 202]. Such orderings impose unnatural constraints on the scene generation process, inhibiting practical applications. For example, in [163, 202], which order objects by class frequency, the probability of a bed (more common) appearing after an ottoman (less common) in the training set is zero. As a result, these methods cannot generate more common objects after less common objects, which makes them impractical for interactive tasks like general room completion and partial room re-arrangement, where input is unconstrained (e.g. Fig.5.1B).

To address these limitations, we pose scene synthesis as an unordered set generation problem and introduce ATISS, a novel autoregressive transformer architecture to model this process. Given a room type (e.g. bedroom, living room) and its shape, our model generates meaningful furniture arrangements by sequentially placing objects in a permutation-invariant fashion. We train ATISS to maximize the log-likelihood of all possible permutations of object arrangements in a collection of training scenes, labeled only with object classes and 3*D* bounding boxes, which are easier to obtain, than costly support relationship [197] or scene graph annotations [109]. Unlike existing works [198, 163, 202], we propose the first model to perform scene synthesis as an autoregressive *set generation* task. ATISS is significantly simpler to implement and train, requires fewer parameters and is up to 8× faster at run-time than the fastest available baseline [202]. Furthermore, we demonstrate that our model generates more plausible object arrangments without any post-processing on the predicted layout. Our formulation allows applying a single trained model to automatic layout synthesis and to a number of interactive scenarios with versatile user input (Fig.5.1), such as automatic placement of user-provided objects, object suggestion with user-provided constraints, and room completion.

5.1 RELATED WORK

Manually building diverse and visually plausible 3D content for indoor and outdoor environments is a time consuming task that requires selecting and placing a large number of diverse assets in a scene. To address this, the research community has recently shifted their attention to the development of generative models that can perform the laborious task of synthetic content creation. Synthetic data has been extensively used in many domains such as autonomous driving [166, 160, 49, 3, 206, 149, 88, 95], indoor scene understanding [220, 179, 110, 171, 5, 222, 54, 112], robotics [204], optical flow estimation [17, 176] etc. In this section, we discuss the most relevant literature on interior scene synthesis, as well as transformer architectures [196] in the context of generative modeling.

Procedural Modeling with Grammars: Procedural modeling describes methods that recursively apply a set of functions for content synthesis. Grammars are a formal instantiation of this idea and have been used for modeling 3D structures such as plants [182], buildings and cities [125, 135], indoor [153] and outdoor [149] scenes. [182] employed reversible-jump MCMC to control the output of stochastic context-free grammars. Meta-Sim [88] learned a model that modifies attributes of scene graphs sampled from a known probabilistic context-free grammar to match visual statistics between generated and real data. [41] extended this model to also learn to sample from the grammar, allowing context dependent relationships to be learnt. Concurrently, [150] employed Grammar-VAE [103] to generate scenes using a scene grammar generated from annotated data. In contrast, our model implicitly encapsulates inter-object relationships, without having to impose hand-crafted constraints.

Graph-based Scene Synthesis: Representing scenes as graphs has been extensively studied in literature [109, 197, 223, 117, 150, 221, 218, 94, 44]. Zhou et al. [223] introduced a neural message passing algorithm for scene graphs that predicts the category of the next object to be placed at a specific location. Similarly, [109, 221, 150, 117] utilized a VAE [97] to synthesize 3D scenes as parse trees [150], adjacency matrices [221], scene graphs [117] and scene hierarchies [109]. Concurrently, [197, 218] adopted a two-stage generation process that disentangles planning the scene layout from instantiating the scene based on this plan. Note that graph-based models require supervision either in the form of relation graphs [197, 218, 117] or scene hierarchies [109]. In contrast, ATISS infers functional and spatial relations between objects directly from data labeled only with object classes and 3D bounding boxes.

Autoregressive Scene Synthesis: Closely related to ATISS are autoregressive indoor scene generation models [198, 163, 202]. Ritchie et al. [163] introduced a CNN-based architecture that operates on a top-down image-based representation of a scene and inserts objects in it sequentially by predicting their category, location, orientation and size with separate network modules. [163] requires supervision in the form of 2D bounding boxes as well as auxiliary supervision such as depth maps and object segmentation masks. In concurrent work, Wang et al. [202] introduced SceneFormer, a series of transformers that autoregressively add objects in a scene similar to [163]. Both [163, 202] use separate models to generate object attributes (e.g. category, location) that are trained independently and represent scenes as ordered sequences of objects, ordered by the category frequency. In contrast, we propose a simpler architecture that consists of a single model trained end-to-end to predict all attributes. We provide experimental evidence that our model generates more realistic object arrangements while being significantly faster. While [163, 202] assume a fixed ordering of the objects in each scene, our model does not impose any constraint on the ordering of objects. Instead, during training, we enforce that our model generates objects with all orderings, in a permutation invariant fashion. This allows us to represent scenes as unordered sets of objects and perform various interactive tasks such as rearranging any object in a room or suggesting new objects given any room.

Transformers for Set Generation: Transformer models [196] demonstrated impressive results on various tasks such as machine translation [175, 133], language-modeling [16, 42], object detection [108, 18, 224], image recognition [48, 188], semantic segmentation [214] as well as on image [138, 91, 22, 52, 189] and music [43] generation tasks. While there are works [105, 101] that utilize the permutation equivariance property of transformers for unordered set processing and prediction, existing generative models with transformers assume ordered sequences [16, 22, 31] even when there exists no natural order such as for pointclouds [126] and objects in a scene [202]. Instead, we introduce an autoregressive transformer for unordered set gen-

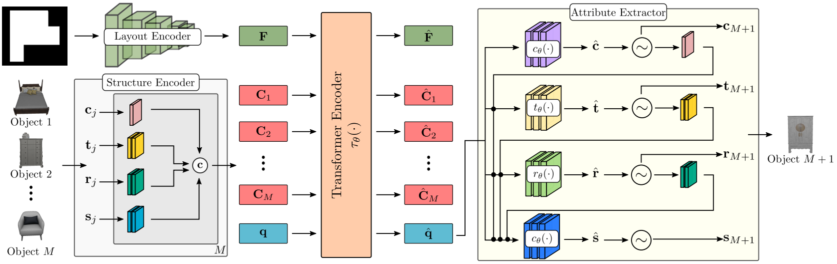


Figure 5.2: **Method Overview.** Starting from a scene with M objects and a floor layout, the **layout encoder** maps the floor into a feature representation $\mathbf{F}$ and the **structure encoder** maps the objects into a context embedding $\mathbf{C} = \{\mathbf{C}_j\}_{j=1}^{M}$. The floor layout feature $\mathbf{F}$, the context embedding $\mathbf{C}$ and a learnable query vector $\mathbf{q}$ are then passed to the **transformer encoder** that predicts $\hat{\mathbf{q}}$. Using $\hat{\mathbf{q}}$ the **attribute extractor** autoregressively predicts the attribute distributions that are used to sample the attributes for the next object to be generated.

eration that enforces that the probability of adding a new element in the set is invariant to the order of the elements already in the set. We show that for the scene synthesis task, our model outperforms transformers that consider ordered sets of elements in every metric.

5.2 METHOD

Given an empty or a partially complete room of a specific type (e.g. bedroom) together with its shape, as a top-down orthographic projection of its floor, we want to learn a generative model that populates the room with objects, whose functional composition and spatial arrangement is plausible. To this end, we propose an autoregressive model that represents scenes as *unordered sets of objects* (Sec. 5.2.1) and describe our implementation using a transformer network (Sec. 5.2.2). Finally, we analyse the training and inference details of our method (Sec. 5.2.3).

5.2.1 *Autoregressive Set Generation*

Let $\mathcal{X} = \{\mathcal{X}_1, \dots, \mathcal{X}_N\}$ denote a collection of scenes where each $\mathcal{X}_i = (\mathcal{O}_i, \mathbf{F}^i)$ comprises the unordered set of objects in the scene $\mathcal{O}_i = \{o_j^i\}_{j=1}^{M}$ and its floor layout $\mathbf{F}^i$. To compute the likelihood of generating $\mathcal{O}_i$ we need to accumulate the likelihood of generating $\{o^i_j\}_{j=1}^M$ autoregressively in *any order*. This is formally written as

$$p_\theta(\mathcal{O}_i|\mathbf{F}^i) = \sum_{\hat{\mathcal{O}}\in\pi(\mathcal{O}_i)} \prod_{j\in\hat{\mathcal{O}}} p_\theta(o^i_j \mid o^i_{<j}, \mathbf{F}^i), \tag{5.1}$$

where $p_\theta(o^i_j \mid o^i_{<j}, \mathbf{F}^i)$ is the probability of the j-th object, conditioned on the previously generated objects and the floor layout, and $\pi(\cdot)$ is a permutation function that computes the set of permutations of all objects in the scene. As a result, the log-likelihood of the whole collection $\mathcal{X}$ is

$$\log p_\theta(\mathcal{X}) = \sum_{i=1}^N \log\left(\sum_{\hat{\mathcal{O}}\in\pi(\mathcal{O}_i)} \prod_{j\in\hat{\mathcal{O}}} p_\theta(o^i_j \mid o^i_{<j}, \mathbf{F}^i)\right). \tag{5.2}$$

However, training our generative model to maximize the log-likelihood of (5.2) poses two problems: (a) the summation over all permutations is intractable and (b) (5.2) does not ensure that all orderings will have high probability. The second problem is crucial because we want our generative model to be able to complete *any partial set* in a plausible way, namely we want any generation order to have high probability. To this end, instead of maximizing (5.2), we maximize the likelihood of generating a scene in all possible orderings, $\hat{p}_\theta(\cdot)$, which is defined as

$$\begin{aligned}\log \hat{p}_\theta(\mathcal{X}) &= \sum_{i=1}^N \log\left(\prod_{\hat{\mathcal{O}}\in\pi(\mathcal{O}_i)} \prod_{j\in\hat{\mathcal{O}}} p_\theta(o^i_j \mid o^i_{<j}, \mathbf{F}^i)\right) \\ &= \sum_{i=1}^N \sum_{\hat{\mathcal{O}}\in\pi(\mathcal{O}_i)} \sum_{j\in\hat{\mathcal{O}}} \log p_\theta(o^i_j \mid o^i_{<j}, \mathbf{F}^i).\end{aligned} \tag{5.3}$$

Note that training our generative model with (5.3) allows us to approximate the summation over all permutations using Monte Carlo sampling thus solving both problems of (5.2).

Modelling Object Attributes: We represent objects in a scene as labeled 3D bounding boxes and model them with four random variables that describe their category, size, orientation and location, $o_j = \{\mathbf{c}_j, \mathbf{s}_j, \mathbf{t}_j, \mathbf{r}_j\}$. The category $\mathbf{c}_j$ is modeled using a categorical variable over the total number of object categories C in the dataset. For the size $\mathbf{s}_j \in \mathbb{R}^3$, the location $\mathbf{t}_j \in \mathbb{R}^3$ and the orientation $\mathbf{r}_j \in \mathbb{R}^1$, we follow [170, 192] and model them with mixture of logistics distributions

$$\begin{aligned}\mathbf{s}_j &\sim \sum_{k=1}^{K} \pi_k^s \text{logistic}(\mu_k^s, \sigma_k^s)\\ \mathbf{t}_j &\sim \sum_{k=1}^{K} \pi_k^t \text{logistic}(\mu_k^t, \sigma_k^t)\\ \mathbf{r}_j &\sim \sum_{k=1}^{K} \pi_k^r \text{logistic}(\mu_k^r, \sigma_k^r)\end{aligned} \tag{5.4}$$

where π_k^s, μ_k^s and σ_k^s denote the weight, mean and variance of the k-th logistic distribution used for modeling the size. Similarly, π_k^t, μ_k^t and σ_k^t and π_k^r, μ_k^r ans σ_k^r refer to the weight, mean and variance of the k-th logistic of the location and orientation, respectively. In our setup, the orientation is the angle of rotation around the up vector and the location is the 3D centroid of the bounding box.

Similar to prior work [163, 202], we predict the object attributes in an autoregressive manner: object category first, followed by position, orientation and size as follows:

$$\begin{aligned}&p_\theta(o_j \mid o_{<j}, \mathbf{F}) =\\ &p_\theta(\mathbf{c}_j|o_{<j}, \mathbf{F})p_\theta(\mathbf{t}_j|\mathbf{c}_j, o_{<j}, \mathbf{F})p_\theta(\mathbf{r}_j|\mathbf{c}_j, \mathbf{t}_j, o_{<j}, \mathbf{F})p_\theta(\mathbf{s}_j|\mathbf{c}_j, \mathbf{t}_j, \mathbf{r}_j, o_{<j}, \mathbf{F}).\end{aligned} \tag{5.5}$$

This is a natural choice, since we want our model to consider the object class before reasoning about the size and the position of an object. To avoid notation clutter, we omit the scene index i from (5.5).

5.2.2 *Network Architecture*

The input to our model is a collection of scenes in the form of 3D labeled bounding boxes with their corresponding room shape. Our network consists of four main components: (i) the *layout encoder* that maps the room shape to a global feature representation $\mathbf{F}$, (ii) the *structure encoder* h_θ that maps the M objects in the scene into per-object context embeddings $\mathbf{C} = \{\mathbf{C}_j\}_{j=1}^{M}$, (iii) the *transformer encoder* τ_θ that takes $\mathbf{F}$, $\mathbf{C}$ and a query embedding $\mathbf{q}$ and predicts the features $\hat{\mathbf{q}}$ for the next object to be generated and (iv) the *attribute extractor* that predicts the attributes of the next object. Our model is illustrated in Fig. 5.2. The layout encoder is simply a ResNet-18 [75] that extracts a feature representation $\mathbf{F} \in \mathbb{R}^{64}$ for the top-down orthographic projection of the floor.

Structure Encoder: The structure encoder h_θ maps the attributes of the j-th object into a per-object context embedding $\mathbf{C}_j$ as follows:

$$\begin{aligned} h_\theta : \mathbb{R}^M \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^1 &\to \mathbb{R}^{L_c} \times \mathbb{R}^{L_s} \times \mathbb{R}^{L_t} \times \mathbb{R}^{L_r} \\ (\mathbf{c}, \mathbf{s}, \mathbf{t}, \mathbf{r}) &\mapsto [\lambda(\mathbf{c}); \gamma(\mathbf{s}); \gamma(\mathbf{t}); \gamma(\mathbf{r})] \end{aligned} \tag{5.6}$$

where L_c, L_s, L_t, L_r are the output dimensionalities of the embeddings used to map the category, the size, the location and the orientation into a higher dimensional space respectively and $[\cdot\,;\cdot]$ denotes concatenation. For the object category $\mathbf{c}_j$ we use a learnable embedding $\lambda(\cdot)$, whereas for the size $\mathbf{s}_j$, the position $\mathbf{t}_j$ and the orientation $\mathbf{r}_j$, we use the positional encoding of [196] as follows

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)) \tag{5.7}$$

where p can be any of the size, position or orientation attributes and $\gamma(\cdot)$ is applied separately in each attribute's dimension. The set of per-object context vectors synthesizes the context embedding $\mathbf{C}$ that encapsulates information for the existing objects in the scene and is used to condition the next object to be generated. Before passing the output of (5.6) to the transformer encoder, we map each $\mathbf{C}_j$ to 64 dimensions using a linear projection.

Transformer Encoder: We follow [196, 42] and implement our encoder τ_θ as a multi-head attention transformer without any positional encoding. This allows us to learn a parametric function that computes features that are invariant to the order of $\mathbf{C}_j$ in $\mathbf{C}$. We use these features to predict the next object to be added in the scene, creating an autoregressive model. The input set of the transformer is $\mathbf{I} = \{\mathbf{F}\} \cup \{\mathbf{C}_j\}_{j=1}^M \cup \mathbf{q}$, with M the number of objects in the scene. $\mathbf{q} \in \mathbb{R}^{64}$ is a learnable object query vector that allows the transformer to predict output features $\hat{\mathbf{q}} \in \mathbb{R}^{64}$ used for generating the next object to be added in the scene. The use of a query token is akin to the use of a mask embedding in Masked Language Modelling [42] or the class embedding for the Vision Transformer [172].

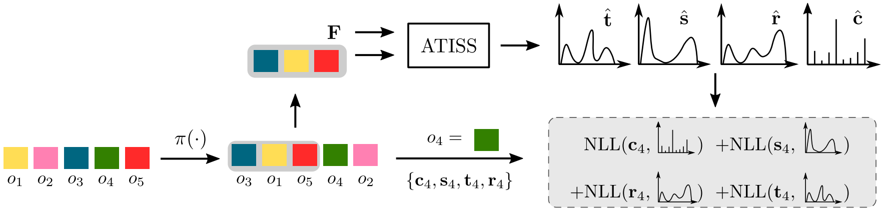

F
ATISS
$\hat{\mathbf{t}}$
$\hat{\mathbf{s}}$
$\hat{\mathbf{r}}$
$\hat{\mathbf{c}}$
$\pi(\cdot)$
o_1 o_2 o_3 o_4 o_5
o_3 o_1 o_5 o_4 o_2
$o_4 =$
$\{\mathbf{c}_4, \mathbf{s}_4, \mathbf{t}_4, \mathbf{r}_4\}$
NLL($\mathbf{c}_4$,) +NLL($\mathbf{s}_4$,)
+NLL($\mathbf{r}_4$,) +NLL($\mathbf{t}_4$,)

Figure 5.3: **Training Overview:** Given a scene with M objects (coloured squares), we first randomly permute them and then keep the first T objects (here $T = 3$). We task our network to predict the next object to be added in the scene given the subset of kept objects (highlighted with grey) and its floor layout feature **F**. Our loss function is the negative log-likelihood (NLL) of the next object in the permuted sequence (green square).

Attribute Extractor: We autoregressively predict the attributes of the next object to be added in the scene using one MLP for each attribute. More formally, the attribute extractor is defined as follows:

$$c_\theta : \mathbb{R}^{64} \to \mathbb{R}^C \qquad \hat{\mathbf{q}} \mapsto \hat{\mathbf{c}} \tag{5.8}$$

$$t_\theta : \mathbb{R}^{64} \times \mathbb{R}^{L_c} \to \mathbb{R}^{3\times3\times K} \qquad (\hat{\mathbf{q}}, \lambda(\mathbf{c})) \mapsto \hat{\mathbf{t}} \tag{5.9}$$

$$r_\theta : \mathbb{R}^{64} \times \mathbb{R}^{L_c} \times \mathbb{R}^{L_t} \to \mathbb{R}^{1\times3\times K} \qquad (\hat{\mathbf{q}}, \lambda(\mathbf{c}), \gamma(\mathbf{t})) \mapsto \hat{\mathbf{r}} \tag{5.10}$$

$$s_\theta : \mathbb{R}^{64} \times \mathbb{R}^{L_c} \times \mathbb{R}^{L_t} \times \mathbb{R}^{L_r} \to \mathbb{R}^{3\times3\times K} \qquad (\hat{\mathbf{q}}, \lambda(\mathbf{c}), \gamma(\mathbf{t}), \gamma(\mathbf{r})) \mapsto \hat{\mathbf{s}} \tag{5.11}$$

where $\hat{\mathbf{c}}, \hat{\mathbf{s}}, \hat{\mathbf{t}}, \hat{\mathbf{r}}$ are the predicted attribute distributions and c_θ, t_θ, r_θ and s_θ are mappings between the latent space and the low-dimensional space of attributes. For the object category, c_θ predicts C class probabilities, whereas, t_θ, r_θ and s_θ predict the mean, variance and mixing coefficient for the K logistic distributions for each attribute. To predict the object properties in an autoregressive manner, we need to condition the prediction of a property on the previously predicted properties. Thus, instead of only passing $\hat{\mathbf{q}}$ to each MLP, we concatenate it with the previously predicted attributes, mapped in a higher-dimensional space using the embeddings $\lambda(\cdot)$ and $\gamma(\cdot)$ from (5.6).

5.2.3 *Training and Inference*

During training, we choose a scene from the dataset and apply a random permutation $\pi(\cdot)$ on its M objects. Then, we randomly select the first T objects to compute the context embedding **C**. Conditioned on **C** and **F**, our network predicts the attribute distributions of the next object to be added in the scene and is trained to maximize the log-likelihood of the $T+1$ object from the permuted scene. A pictorial representation of the training process is provided in Fig. 5.3. To indicate the end of sequence, we augment the C object classes with an additional class, which we refer to as *end symbol*.

During inference, we start with an empty context embedding $\mathbf{C} = \emptyset$ and the floor representation $\mathbf{F}$ of the room to be populated and autoregressively sample attribute values from the predicted distributions of (5.8)-(5.11) for the next object. Once a new object is generated, it is appended to the context $\mathbf{C}$ to be used in the next step of the generation process until the *end symbol* is generated. A pictorial representation of the generation process can be found in Fig. 5.2. In order to transform the predicted labeled bounding boxes to 3D models we use object retrieval. In particular, we retrieve the closest object from the dataset in terms of the euclidean distance of the bounding box dimensions.

5.3 EXPERIMENTAL EVALUATION

In this section, we provide an extensive evaluation of our method, comparing it to existing baselines. We further showcase several interactive use cases enabled by our method, not previously possible.

5.3.1 *Datasets*

We train our model on the 3D-FRONT dataset [54] which contains a collection of $6,813$ houses with roughly $14,629$ professionally designed rooms, populated with 3D furniture objects from the 3D-FUTURE dataset [55]. In our evaluation, we focus on four room types: (i) bedrooms, (ii) living rooms, (iii) dining rooms and (iv) libraries. After pre-processing to filter out uncommon object arrangements and rooms with unnatural sizes, we obtained 5996 bedrooms, 2962 living rooms, 2625 dining rooms and 622 libraries. We use 21 object categories for the bedrooms, 24 for the living and dining rooms and 25 for the libraries. The preprocessing steps are discussed in detail in Sec. A.3.1.

5.3.2 *Baselines*

We compare our model with FastSynth [163] and SceneFormer [202]. Note that both methods were originally evaluated on the SUNCG dataset [179],

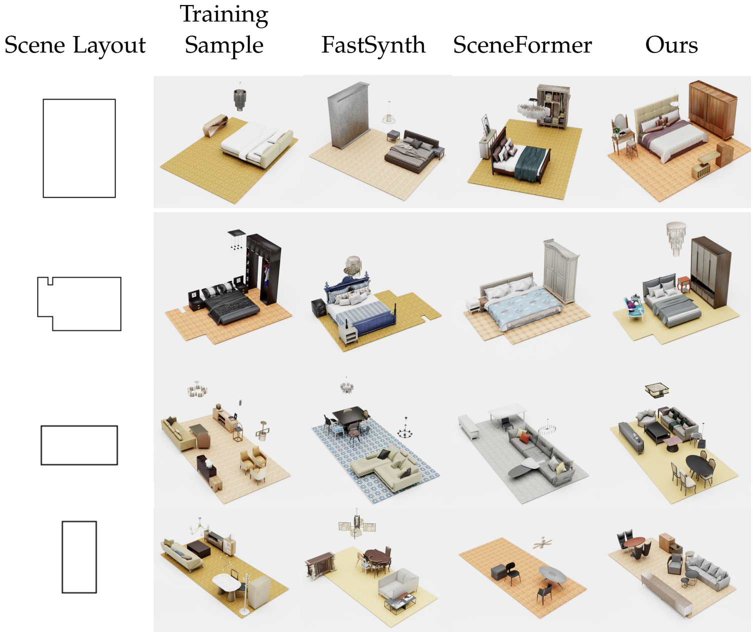


Figure 5.4: **Qualitative Scene Synthesis Results**. Synthesized scenes for three room types: bedrooms (1st+2nd row), living room (3rd row), dining room (4th row) using FastSynth, SceneFormer and our method. To showcase the generalization abilities of our model we also show the closest scene from the training set (2nd column).

which is currently unavailable, thus we retrained both on 3D-FRONT using the augmentation techniques described in the original papers. We also compare with a variant of our model that generates scenes as ordered sequences of objects (Ours+Order). To incorporate the order information to the input, we utilize a positional embedding [196] and a fixed ordering based on the object frequency as described in [202]. To ensure fair comparison, we use the same object retrieval for all methods and no rule-based post-processing on the generated layouts.

FastSynth: In FastSynth [163], the authors employ a series of image-based CNNs to sequentially predict the attributes of the next object to be added in the scene. In addition to 2D labeled bounding boxes they have auxiliary supervision in the form of object segmentation masks, depth maps, wall masks etc. For more details, we refer the reader to [198]. During training, they assume that there exists an ordering of objects in each scene, based on the average size of each category multiplied by its frequency of occurrences in the dataset. Each CNN module is trained separately and the object properties are predicted in an autoregressive manner: object category first, followed by location, orientation and size. We train [163][1] using the provided PyTorch [144] implementation with the default parameters until convergence.

SceneFormer: In SceneFormer [202], the authors utilize a series of transformers to autoregressively add objects in the scene, similar to [163]. In particular, they train a separate transformer for each attribute and they predict the object properties in an autoregressive manner: object category first, followed by orientation, location and size. Similar to [163], they also treat scenes as ordered sequences of objects ordered by the frequency of their categories. We train [202][2] using the provided PyTorch [144] implementation with the default parameters until convergence.

5.3.3 *Evaluation Metrics*

We evaluate our generative model wrt. its ability to generate *realistic* furniture arrangements. To measure the realism of the generated scenes, we follow prior work [163] and report the KL divergence between the object category distributions of synthesized and real scenes from the test set and the classification accuracy of a classifier trained to discriminate real from synthetic scenes. In particular, for the KL divergence, we measure the frequency of object category occurrences in the generated scenes and compare it with the frequency of object occurrences in real scenes. Regarding the scene classification accuracy, we train a classifier to distinguish real from generated scenes. Our classifier is an Alexnet [102] pre-trained on ImageNet, that takes as input a 256^2 top-down image-based representation of a room and predicts whether this scene is real or synthetic. We also report the FID [77] between top-down orthographic projections of synthesized and real scenes from the test set, which we compute using [137] on 256^2 images. We repeat the metric computation for FID and classification accuracy 10 times and report the average.

---

[1] https://github.com/brownvc/fast-synth

[2] https://github.com/cy94/sceneformer

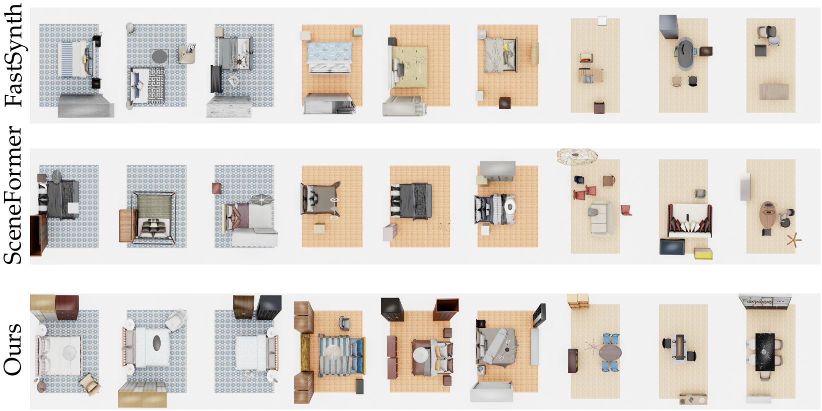


Figure 5.5: **Scene Diversity**. We show three generated scenes conditioned on three different floor plans for bedrooms and dining rooms. Every triplet of columns corresponds to a different floor plan.

	FID Score (↓)				Scene Classification Accuracy				Category KL Divergence (↓)			
	FastSynth	SceneFormer	Ours+Order	Ours	FastSynth	SceneFormer	Ours+Order	Ours	FastSynth	SceneFormer	Ours+Order	Ours
Bedrooms	40.89	43.17	38.67	**38.39**	0.883	0.945	0.760	**0.562**	0.0064	**0.0052**	0.0533	0.0085
Living	61.67	69.54	35.37	**33.14**	0.945	0.972	0.694	**0.516**	0.0176	0.0313	0.0372	**0.0034**
Dining	55.83	67.04	35.79	**29.23**	0.935	0.941	0.623	**0.477**	0.0518	0.0368	0.0278	**0.0061**
Library	37.72	55.34	35.60	**35.24**	0.815	0.880	0.572	**0.521**	0.0431	0.0232	0.0183	**0.0098**

Table 5.1: **Scene Synthesis Quantitative Comparison on 3D-FRONT.** We report the FID score (↓) at 256^2 pixels, the KL divergence (↓) between the distribution of object categories of synthesized and real scenes and the real vs. synthetic classification accuracy for all methods. Classification accuracy closer to 0.5 is better.

5.3.4 *Scene Synthesis*

We start by evaluating the performance of our model on generating plausible object configurations for various room types, conditioned on different floor plans. Fig. 5.4 provides a qualitative comparison of four scenes generated with our model and baselines. In some cases, both [163, 202] generate invalid room layouts with objects positioned outside room boundaries or overlapping. Instead, our model consistently synthesizes realistic object arrangements. We validate this quantitatively in Tab. 5.1, where we compare the generated scenes wrt. their similarity to the original data from 3D-FRONT. Synthesized scenes sampled from our model are almost indistinguishable from scenes from the test set, as indicated by the classification accuracy in Tab. 5.1, which is consistently around 50%. Our model also

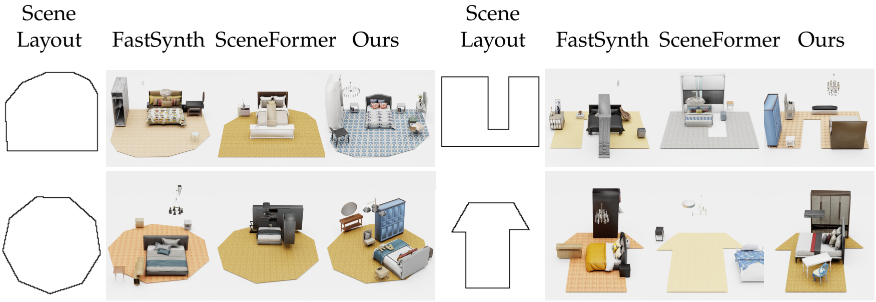


Figure 5.6: **Generalization Beyond Training Data**. We show four synthesized bedrooms conditioned on four room layouts that we manually designed.

achieves lower FID scores for all room types and generates category distributions that are more faithful to the category distributions of the test set, expressed as lower KL divergence.

To showcase that our model generates diverse object arrangements we visualize 3 generated scenes conditioned on the same floor plan for all methods (Fig. 5.5). We observe that our generated scenes are consistently valid and contain diverse object arrangements. In comparison [163, 202] struggle to generate plausible layouts particularly for the case of living rooms and dining rooms. We hypothesize that these rooms are more challenging than bedrooms, for the baselines, due to their significantly smaller volume of training data, and the large number of constituent objects per scene (20 on average, as opposed to 8). To investigate whether our model also generates plausible layouts conditioned on floor plans with uncommon shapes that are not in the training set, we manually design unconventional floor plans (Fig. 5.6) and generate bedroom layouts. While both [163, 202] fail to generate valid scenes, our model synthesizes diverse object layouts that are consistent with the floor plan. Finally, we compare the computational requirements of our architecture to [163, 202]. Our model is significantly faster (Tab. 5.2), while having fewer parameters (Tab. 5.3) than both [163, 202]. Note that [163] is orders of magnitude slower because it requires rendering every individual object added in the scene.

5.3.5 *Applications*

In this section, we present three applications that greatly benefit by our unordered set formulation and are crucial for creating an interactive scene synthesis tool.

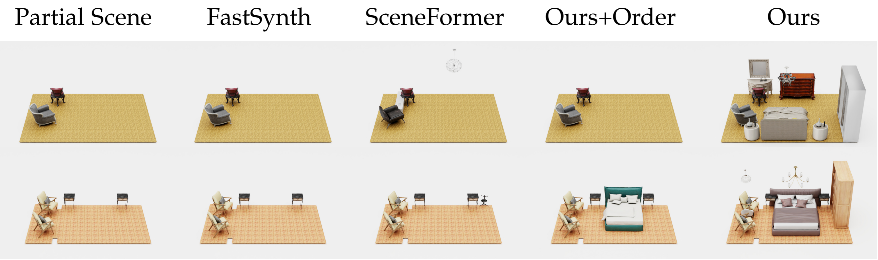


Figure 5.7: **Scene Completion**. Given a partial scene (left column), we visualize scene completions using our model and our baselines. Our model consistently generates plausible layouts.

	Bedroom	Living	Dining	Library
FastSynth [163]	13193.77	30578.54	26596.08	10813.87
SceneFormer [202]	849.37	731.84	901.17	369.74
Ours [163]	**102.38**	**201.59**	**201.84**	**88.24**

Table 5.2: **Generation Time Comparison.** We measure time (ms) to generate a scene, conditioned on a floor plan.

FastSynth [163]	SceneFormer [202]	Ours
38.180	129.298	36.053

Table 5.3: **Network Parameters Comparison.** We report the number of network parameters in millions.

Scene Completion: Starting from a partial scene, the task is to populate the empty space in a meaningful way. Since both [163, 202] are trained on sorted sequences of objects, they first generate frequent objects (e.g. beds, wardrobes) followed by less common objects. As a result, incomplete scenes that contain less common objects cannot be correctly populated. This is illustrated in Fig. 5.7, where all baselines fail to add objects in a scene that contains two armchairs, resulting in bedrooms without beds. For the case where a scene contains two armchairs and two nightstands [163, 202] again fail to generate any object. Our model successfullly generates plausible completions with multiple objects such as lamps, wardrobes and dressing tables.

Failure Case Detection and Correction: We showcase that our model is able to identify and correct unnatural object arrangements. Given a scene, we compute the likelihood of each object, according to our model, conditioned on the other objects in the scene. We identify problematic objects as those with low likelihood and sample a new location from our generative model to rearrange it. We test our model in various scenarios such as overlapping objects, objects outside the room boundaries and objects in unnatural positions and show that it successfully identifies problematic objects (highlighted in green in Fig. 5.8) and rearranges them into a more

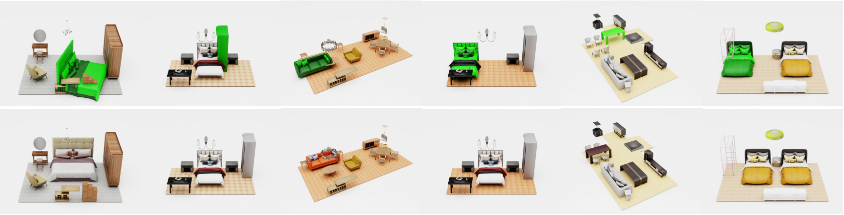


Figure 5.8: **Failure Case Detection and Correction**. We use a partial room with unnatural object arrangements. Our model identifies the problematic objects (first row, in green) and relocates them into meaningful positions.

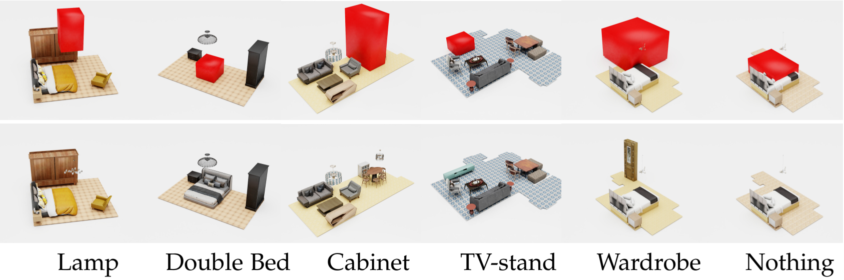


Figure 5.9: **Object Suggestion**. A user specifies a region of acceptable positions to place an object (marked as red boxes, 1st row) and our model suggests suitable objects (2nd row) to be placed in this location.

plausible position. Note that this task cannot be performed by methods that consider ordering because they assign very low likelihood to common objects appearing after rare objects e.g. beds after cabinets.

Object Suggestion: We now test the ability of our model to provide object suggestions given a scene and user specified location constraints. To perform this task we sample objects from our generative model and accept the ones that fullfill the constraints provided by the user. Fig. 5.9 shows examples of location constraints (red box in top row) and the corresponding objects suggested (bottom row). Note that even when the user provided region is partially outside the room boundaries (4th, 5th column), suggested objects always reside in the room. Moreover, if the acceptable region overlaps with another object, our model suggests adding nothing (6th column). This task requires computing the likelihood of an object conditioned on an arbitrary scene, which [202, 163] cannot perform due to ordering.

5.3.6 *Perceptual Study*

We conducted two paired Amazon Mechanical Turk perceptual studies to evaluate the quality of our generated layouts against [163] and [202]. We sample 6 bedroom layouts for each method from the same 211 test set floor plans. Users saw 2 different rotating 3D scenes per method randomly selected from 6 pre-rendered layouts. Random layouts for each floor plan were assessed by 5 different workers to evaluate agreement and diversity across samples for a total of 1055 question sets per paired study. Generated scenes of [163] were judged to contain errors like interpenetrating furniture 41.4% of the time, nearly twice as frequently as our method, while [202] performs significantly worse (Tab. 5.4). Regarding realism, the scenes of [163] were more realistic than ours in only 26.9% of the cases. We conclude that our method outperforms the baselines in the key metric, generation of realistic indoor scenes, by a large margin. Additional details are provided in Sec. A.3.2.

Method	Condition	Mean Error Frequency ↓	More ↑ Realistic	Realism CI 99%
FastSynth [163]	vs. Ours	0.414	0.269	[0.235, 0.306]
SceneFormer [202]	vs. Ours	0.713	0.165	[0.138, 0.196]
Ours	vs. Both	**0.232**	**0.783**	[0.759, 0.805]

Table 5.4: **Perceptual Study Results**. Aggregated results for two A/B paired tests. Our method was judged more realistic with high confidence (binomial confidence interval with $\alpha = 0.01$ reported) and contained fewer errors.

5.4 ABLATION STUDY

We now investigate how various components of our model affect its performance on the scene synthesis task. In Sec. 5.4.1, we investigate the impact of the number of logistic distributions in the performance of our model. Next, in Sec. 5.4.2, we examine the impact of the architecture of the layout encoder. In Sec. 5.4.3, we compare ATISS with two variants of our model that consider ordered sets of objects. Unless stated otherwise, all ablations are conducted on the bedroom scenes of the 3D-FRONT [54] dataset.

5.4.1 *Mixture of Logistic distributions*

We represent objects in a scene as labeled 3D bounding boxes and model them with four random variables that describe their category, size, orientation and location, $o_j = \{\mathbf{c}_j, \mathbf{s}_j, \mathbf{t}_j, \mathbf{r}_j\}$. The category $\mathbf{c}_j$ is modeled using a categorical variable over the total number of object categories C in the dataset. For the size $\mathbf{s}_j \in \mathbb{R}^3$, the location $\mathbf{t}_j \in \mathbb{R}^3$ and the orientation $\mathbf{r}_j \in \mathbb{R}^1$, we follow [170, 192] and model them with a mixture of logistic distributions respectively. In this experiment, we test our model with different numbers for logistic distributions for modelling the object attributes. Results are summarized in Tab. 5.5.

	FID (↓)	Classification Accuracy (↓)	Category Distribution (↓)
$K = 1$	41.71 ± 0.4008	0.7826 ± 0.0080	0.0491
$K = 5$	40.41 ± 0.2491	0.5667 ± 0.0405	0.0105
$K = 10$	**38.39** ± 0.3392	**0.5620** ± 0.0228	0.0085
$K = 15$	40.41 ± 0.4504	0.5980 ± 0.0074	0.0095
$K = 20$	40.39 ± 0.3964	0.6680 ± 0.0035	**0.0076**

Table 5.5: **Ablation Study on the Number of Logistic Distributions.** This table shows a quantitative comparison of our model with different numbers of K logistic distributions for modelling the size, location and orientation of each object.

As it is expected, using a single logistic distribution (first row in Tab. 5.5) results in worse performance, since it does not have enough representation capacity for modelling the object attributes. We also note that increasing the number of logistic distributions beyond 10 hurts performance wrt. FID and classification accuracy. We hypothesize that this is due to overfitting. In our experiments we set $K = 10$.

5.4.2 *Layout Encoder*

We further examine the impact of the layout encoder on the performance of our model. To this end, we replace the ResNet-18 architecture [75], with an AlexNet [102]. From the original architecture, we remove the final classifier layers and keep only the feature vector of length 9216 after max pooling. We project this feature vector to 64 dimensions with a linear projection layer. Similar to our vanilla model, we do not use an AlexNet pre-trained on ImageNet because we empirically observed that it resulted in worse performance.

	FID (↓)	Classification Accuracy (↓)	Category Distribution (↓)
AlexNet	40.40 ± 0.2637	0.6083 ± 0.0034	**0.0064**
ResNet-18	**38.39** ± 0.3392	**0.5620** ± 0.0228	0.0085

Table 5.6: **Ablation Study on the Layout Encoder Architecture.** This table shows a quantitative comparison of ATISS with two different layout encoders.

Tab. 5.6 compares the two variants of our model wrt. to the FID score, the classification accuracy and the KL-divergence. We remark that our method is not particularly sensitive to the choice of the layout encoder. However, using an AlexNet results in slightly worse performance, hence we utilize a ResNet-18 in all our experiments.

5.4.3 *Transformers with Ordering*

In this section, we analyse the benefits of synthesizing rooms as unordered sets of objects in contrast to ordered sequences. To this end, we train two variants of our model that utilize a positional embedding [196] to incorporate order information to the input. The first variant is trained with random permutations of the input (Ours+Perm+Order), similar to our model, whereas the second with a fixed ordering based on the object frequency (Ours+Order) as described in [163, 202]. We compare these variants to our model on the scene synthesis task and observe that the variant with the fixed ordering (second row Tab. 5.7) performs significantly worse as the classifier can identify synthesized scenes with 76% accuracy. Moreover, we remark that besides enabling all the previously presented applications, training with random permutations also improves the quality

	FID (↓)	Classification Accuracy (↓)	Category Distribution (↓)
Ours+Perm+Order	40.18 ± 0.2831	0.6019 ± 0.0060	0.0089
Ours+Order	38.67 ± 0.5552	0.7603 ± 0.0010	0.0533
Ours	**38.39** ± 0.3392	**0.5620** ± 0.0228	0.0085

Table 5.7: **Ablation Study on Ordering.** This table shows a quantitative comparison of our approach wrt. two variants of our model that represent rooms as ordered sequence of objects.

of the synthesized scenes (first row Tab. 5.7). However, our model that is permutation invariant, namely the prediction is the same regardless of the order of the partial scene, performs even better (third row Tab. 5.7). We conjecture that the invariance of our model will be more even more crucial for training with either larger datasets or larger scenes i.e. scenes with more objects, because observing a single order allows reasoning about all permutations of the partial scene.

5.5 IMPLEMENTATION DETAILS

In this section, we provide a detailed description of our network architecture. We then describe our training protocol and provide details on the metrics computation during training and testing. Finally, we also provide additional details regarding our baselines.

5.5.1 *Network Architecture*

Here we describe the architecture of each individual component of our model illustrated in Fig. 5.2. Our architecture comprises four components: (i) the *layout encoder* that maps the room shape to a global feature representation $\mathbf{F}$, (ii) the *structure encoder* that maps the M objects in a scene into per-object context embeddings $\mathbf{C} = \{\mathbf{C}_j\}_{j=1}^{M}$, (iii) the *transformer encoder* that takes $\mathbf{F}$, $\mathbf{C}$ and a query embedding $\mathbf{q}$ and predicts the features $\hat{\mathbf{q}}$ for the next object to be generated and (iv) the *attribute extractor* that autoregressively predicts the attributes of the next object.

Layout Encoder: The first part of our architecture is the *layout encoder* that is used to map the room's floor into a global feature representation $\mathbf{F}$. We follow [198] and we model the floor plan with its top-down orthographic projection. This projection maps the floor plan into an image, where pixel values of 1 indicate regions inside the room and pixel values of 0 otherwise. The layout encoder is implemented with a ResNet-18 architecture [75] that is not pre-trained on ImageNet [38]. We empirically observed that using a pre-trained ResNet resulted in worse performance. From the original architecture, we remove the final fully connected layer and replace it with a linear projection to 64 dimensions, after average pooling.

Structure Encoder: The structure encoder maps the attributes of each object into a per-object context embedding $\mathbf{C}_j$. For the object category $\mathbf{c}_j$, we use a learnable embedding, which is simply a matrix of size $C \times 64$, that stores a

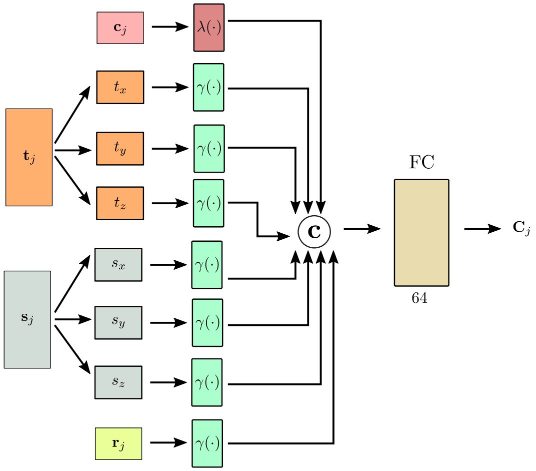


Figure 5.10: **Structure Encoder.** The structure encoder predicts the per-object context embeddings $\mathbf{C}_j$ conditioned on the object attributes. For the object category $\mathbf{c}_j$, we use a learnable embedding $\lambda(\cdot)$, whereas for the location $\mathbf{t}_j$, the size $\mathbf{s}_j$ and orientation $\mathbf{r}_j$ we employ the positional encoding from (5.12). Note that the positional encoding $\gamma(\cdot)$ is applied separately in each dimension of $\mathbf{t}_j$ and $\mathbf{s}_j$.

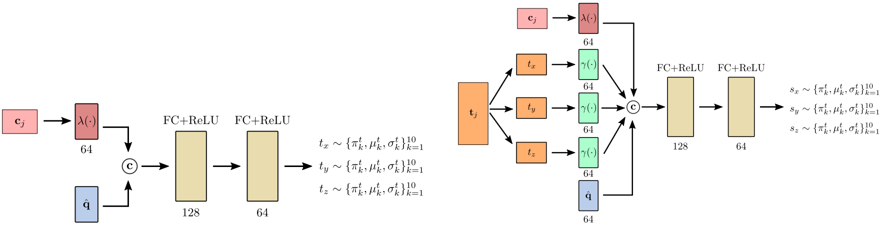
per-object category vector, for all C object categories in the dataset. For the size $\mathbf{s}_j$, the position $\mathbf{t}_j$ and the orientation $\mathbf{r}_j$, we use the positional encoding of [196] as follows

$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p)) \qquad (5.12)$$

where p can be any of the size, position or orientation attributes and $\gamma(\cdot)$ is applied separately in each attribute's dimension. In our experiments, L is set to 32. The output of each embedding layer, used to map the category, size, location and orientation in a higher dimensional space, are concatenated into an 512-dimensional feature vector, which is then mapped to the per-object context embedding $\mathbf{C}_j \in \mathbb{R}^{64}$, using a linear projection layer. A pictorial representation of the structure encoder is provided in Fig. 5.10.

Transformer Encoder: We follow [196, 42] and implement our transformer encoder as a multi-head attention transformer without any positional encoding. Our transformer consists of 4 layers with 8 attention heads. The queries, keys and values have 64 dimensions and the intermediate representations for the MLPs have 1024 dimensions. To implement the transformer architecture we use the transformer library provided by Katharopoulos et al.

[91][3]. The input set of the transformer is $\mathbf{I} = \{\mathbf{F}\} \cup \{\mathbf{C}_j\}_{j=1}^{M} \cup \mathbf{q}$, where M denotes the number of objects in the scene and $\mathbf{q} \in \mathbb{R}^{64}$ is a learnable object query vector that allows the transformer to predict output features $\hat{\mathbf{q}} \in \mathbb{R}^{64}$ used for generating the next object to be added in the scene.


$\mathbf{c}_j$
$\lambda(\cdot)$
64
$\hat{\mathbf{q}}$
ⓒ
FC+ReLU
FC+ReLU
128
64
$t_x \sim \{\pi_k^t, \mu_k^t, \sigma_k^t\}_{k=1}^{10}$
$t_y \sim \{\pi_k^t, \mu_k^t, \sigma_k^t\}_{k=1}^{10}$
$t_z \sim \{\pi_k^t, \mu_k^t, \sigma_k^t\}_{k=1}^{10}$
$\mathbf{c}_j$
$\lambda(\cdot)$
64
$\mathbf{t}_j$
t_x
t_y
t_z
$\gamma(\cdot)$
64
$\gamma(\cdot)$
64
$\gamma(\cdot)$
64
$\hat{\mathbf{q}}$
64
ⓒ
FC+ReLU
FC+ReLU
128
64
$s_x \sim \{\pi_k^t, \mu_k^t, \sigma_k^t\}_{k=1}^{10}$
$s_y \sim \{\pi_k^t, \mu_k^t, \sigma_k^t\}_{k=1}^{10}$
$s_z \sim \{\pi_k^t, \mu_k^t, \sigma_k^t\}_{k=1}^{10}$

(a) $t_\theta(\cdot)$ predicts the parameters of the mixture of logistics distribution for the location $\mathbf{t}$.

(b) $s_\theta(\cdot)$ predicts the parameters of the mixture of logistics distribution for the size $\mathbf{s}$.

Figure 5.11: **Attribute Extractor.** The attribute extractor consists of four MLPs that autoregressively predict the object attributes. Here we visualize the MLP $t_\theta(\cdot)$ for the location attribute (left side) and the MLP $s_\theta(\cdot)$ for the size attribute (right side).

Attribute Extractor: The attribute extractor autoregressively predicts the attributes of the next object to be added in the scene. The MLP for the object category is a linear layer with 64 hidden dimensions that predicts C class probabilities per object. The MLPs for the location, orientation and size predict the mean, variance and mixing coefficient for the K logistic distributions for each attribute. In our experiments we set $K = 10$. The size, location and orientation attributes are predicted using a 2-layer MLP with RELU non-linearities with hidden size 128 and output size 64. A pictorial representation for the MLPs $t_\theta(\cdot)$ and $\sigma_\theta(\cdot)$ used to predict the parameters of the mixture of logistics distribution for the location and the size is provided in Fig. 5.11. Note that r_θ is defined in a similar manner.

5.5.2 *Object Retrieval*

During inference, we select 3D models from the 3D-FUTURE dataset [55] to be placed in the scene based on the predicted category, location, orientation and size. In particular, we perform nearest neighbor search through the 3D-FUTURE dataset[55] to find the closest model in terms of object dimensions. While prior work [163, 202] explored more complex object retrieval schemes

---

[3] https://github.com/idiap/fast-transformers

based on object dimensions and object cooccurrences (i.e. favor 3D model of objects that frequently co-occur in the dataset), we note that our simple object retrieval strategy consistently resulted in visually plausible rooms. We leave more advanced object retrieval schemes for future research.

5.5.3 *Training Protocol*

In all our experiments, we use the Adam optimizer [96] with learning rate $\eta = 10^{-4}$ and no weight decay. For the other hyperparameters of Adam we use the PyTorch defaults: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. We train all models with a batch size of 128 for 100k iterations. During training, we perform rotation augmentation with random rotations between $[0, 360]$ degrees. To determine when to stop training, we follow common practice and evaluate the validation metric every 1000 iterations and use the model that performed best as our final model.

5.6 LIMITATIONS

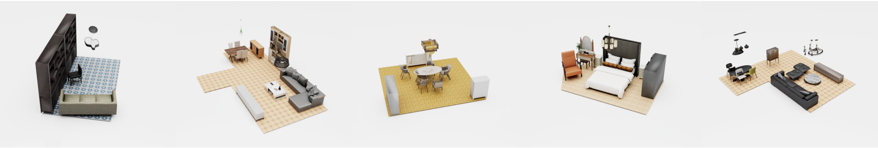


Figure 5.12: **Failure Cases**. We visualize various failure cases of our model for different toom types.

Lastly, we discuss the limitations of our model and show some examples of failure cases in Fig. 5.12. One type of failure case that is illustrated in Fig. 5.12 is overlapping objects, in particular chairs for the case of living rooms and dining rooms (see second and third column in Fig. 5.12). As we already mentioned before, to be able to use the 3D-FRONT dataset, we performed intense filtering to remove objects that intersect with each other. However, we found out that not all problematic arrangements were removed from the dataset, which we hypothesize is the reason for such failure cases. Another type of failure case that we observed, which is also related to the existence of problematic rooms in our training data, is the unnatural orientation of objects (e.g. chair facing the bookshelf in first column of Fig. 5.12 or chair facing opposite of the table in last column of

Fig. 5.12.) Note that these failure cases are quite rare, as also indicated by our quantitative analysis (see Tab. 5.1) as well as the perceptual study in 5.3.6 but our method does not guarantee error-free layouts and there is room for improvement.

Our approach is currently limited to generating object properties using a specific ordering (category first, followed by location, then orientation and lastly size). To further expand the interactive possibilities of our model, we believe that, in the future, object attributes should be generated in an order invariant fashion, similar to the objects in the scene. Furthermore, in our current formulation, the object retrieval is disconnected from the attribute generation. As a result we cannot guarantee that the retrieved objects would match with existing objects in the scene. To address this, it might be beneficial to also incorporate style as an additional object attribute to allow for improved object retrieval. Incorporating style information, would also allows us to generate rooms conditioned on a specific style.

5.7 DISCUSSION

We have presented ATISS, a novel autoregressive transformer architecture for synthesizing 3D rooms as unordered sets of objects. Our method generates realistic scenes that advance the state-of-the-art for scene synthesis. In addition, our novel formulation enables new interactive applications for semi-automated scene authoring, such as general scene completion, object suggestions, anomaly detection and more. We believe that our model is an important step not only toward automating the generation of 3D environments, with impact on simulation and virtual testing, but also toward a new generation of tools for user-driven content generation. By accepting a wide range of user inputs, our model mitigates societal risks of task automation, and promises to usher in tools that enhance the workflow of skilled laborers, rather than replacing them.

We believe that the interactive possibilities of ATISS can be further expanded by considering order invariance to object attributes and incorporating style information. As any machine learning model, our model can introduce learned biases for indoor scenes, and we believe it is important to investigate learning from less structured and more widely available data sources to make this model applicable to a wider range of cultures and environments.

6

DISCUSSION

Research is formalized curiosity. It is poking and prying with a purpose.

— Zora Neale Hurston

6.1 SUMMARY OF CONTRIBUTIONS

In this dissertation, we addressed the task of learning compositional representations both for objects and scenes. In particular, in Part I, we developed primitive-based representations that allow for semantically meaningful and geometrically accurate shape abstractions. In Part II, we proposed a structure-aware representation that jointly recovers the 3D object geometry as a binary tree of primitives that reasons both about the part geometry as well as the part-level relations. Finally, in Part III, we propose a transformer-based network architecture that synthesizes indoor environments by autoregressively placing objects within the scene's boundaries.

In Chapter 2, we proposed the first learning-based solution for parsing 3D objects into consistent superquadric representations. Our model successfully captures both the structure as well as the details of the target objects by accurately learning to predict superquadrics in an unsupervised fashion, directly from data. Employing superquadric surfaces as geometric primitives allows for capturing details that cannot be captured by cuboidal primitives. Due to their diverse shape vocabulary, superquadrics surfaces are easier to learn and yield higher quality reconstructions in comparison to cuboids. More importantly, the predicted primitives are semantically coherent, namely the same primitive is consistently used for representing the same object part across different object instances. This is a valuable trait of our representation that is not enforced by any loss term, instead our model automatically learns to associate specific object parts with specific primitives. We illustrated that our model yields more interpretable reconstructions both in D-FAUST humans and ShapeNet objects while being more geometrically accurate.

In Chapter 3, we presented Neural Parts, a novel 3D primitive representation that defines primitives using an Invertible Neural Network (INN) which implements homeomorphic mappings between a sphere and the target shape. Using an INN to parametrize the homeomorphism allows us to compute both the inverse and the forward mapping of the homeomorphism, which in turn, enables efficient computation of both the implicit surface function of a primitive and its mesh without any additional post-processing. We consider Neural Parts a step towards bridging the gap between semantically meaningful and geometrically accurate primitive-based representations. Our evaluations on various ShapeNet objects, FreiHAND hands and D-FAUST humans showcase that Neural Parts compare favorably to the state-of-the-art primitive-based representations that rely on simpler shapes both in terms of reconstruction quality and semantic consistency.

In Chapter 4, we presented a learning-based approach that jointly predicts part relationships together with per-part geometries in the form of a binary tree of primitives without requiring any part-level annotations during training. In particular, we introduce a neural network architecture that recursively partitions an object into a set of semantically meaningful parts. We illustrated that considering the part-level structure facilitates learning and yields geometrically accurate reconstructions that outperform existing part-based representations, while performing on part with more flexible implicit shape representations.

In Chapter 5, we introduced ATISS, a novel autoregressive transformer architecture for synthesizing 3D rooms as unordered sets of objects. Our formulation results in realistic scenes and enables various applications such as scene completion with any object, detection of unnaturally placed objects as well as object suggestions. These tasks were not possible with existing approaches that represent scenes as ordered sequences of objects. Our extensive experiments on various rooms from the 3D-FRONT dataset demonstrate the ability of ATISS to generate more realistic scenes in comparison to prior work. Moreover, due to the simplicity of our framework, ATISS has fewer learnable parameters, is simpler to implement and train and generates scenes up to 8× faster than existing scene synthesis pipelines.

6.2 DIRECTIONS FOR FUTURE RESEARCH

We believe that our work opens up many exciting research directions towards developing compositional representations for both objects and scenes. Below, we discuss some challenges that we believe should be considered in future work.

Due to the lack of supervision in terms of part-level annotations, existing unsupervised shape abstraction models seek to infer parts by minimizing the discrepancy between the target and the predicted shape [191, 141, 61, 143, 37, 140]. However, while optimizing for the geometry enforces that the reconstructed primitives are geometrically accurate, it does not guarantee that the inferred parts will be either semantically meaningful (i.e. a primitive corresponds to an identifiable part e.g. leg, arms etc.) or semantically consistent (i.e. the same primitive is consistently used for representing the same semantic part). Therefore, we believe that an exciting future research direction is to exploit additional cues for learning meaningful shape abstractions. For example, motion naturally encapsulates the concept of object parts as well as how they are structured in space, thus we anticipate that semantically meaningful shape abstractions can naturally emerge by considering how contiguous regions of an object move/deform across time. While motion is a useful cue for part-based decomposition of non-rigid shapes such as animals, humans etc., it cannot be easily applied for the majority of rigid man-made objects such as e.g. cups, nightstand drawers, etc. To this end, another exciting direction for future research is learning shape abstractions based on the functionality of parts. Typically, functionality-related knowledge about object parts can be derived by modelling how they interact with each other.

Moreover, another exciting research direction is the development of methods that utilize the part-based representations for generating novel 3D shapes such as objects, humans, animals, artificially generated creatures etc. Generating 3D content that is both realistic and diverse is a long-standing problem in computer vision and graphics, since manually creating 3D content is an extremely expensive and laborious endeavour that typically needs to be conducted by experienced artists. During the last decade, there has been an increased demand for tools that automate the 3D content creation process for applications like video games, AR/VR as well as general 3D content creation. Due to their special characteristics, part-based representations are the ideal intermediate representation for such tasks as they allow to automatically interact and edit specific parts of the generated object. While such tools have been proposed before[1], they are exclusively focused on 2D image generation. However, we believe that it is crucial to also develop similar tools that can generate content in 3D.

Finally, another promising direction for future research is the development of compositional representation that go beyond a single object and reason about multiple objects in a scene. We believe that such representations are crucial for various applications, such as autonomous driving and

---

[1] https://ai.googleblog.com/2020/11/using-gans-to-create-fantastical.html

AR/VR, that require reasoning and interacting with different elements of the environment. We hypothesize that existing primitive-based methods cannot be directly employed for reconstructing scenes into semantically consistent parts since they cannot capture neither simple concepts such as empty space nor higher level semantics between elements of the scene. In addition, while for the case of a single object a few primitives might yield meaningful abstractions, for the case of scenes the number of inferred primitives needs to be increased in order ensure that the inferred primitives robustly capture the scene structure. We believe that combining such a holistic scene representations with a generative model would allow us to build interactive tools with control over the object arrangements, object parts and part relationships.

A

ADDITIONAL IMPLEMENTATION DETAILS

A.1 SUPERQUADRICS REVISITED: LEARNING 3D SHAPE PARSING BEYOND CUBOIDS

A.1.1 *Derivation of Pointcloud-to-Primitive Loss*

This section provides the derivation of the pointcloud-to-primitive distance $\mathcal{L}_{X \to P}(\mathcal{X}, \mathcal{P})$ in (2.10). For completeness, we restate our notation briefly. We represent the target point cloud as a set of 3D points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ and we approximate the continuous surface of the m^{th} primitive by a set of 3D points $\mathcal{Y}_m = \{\mathbf{y}_k^m\}_{k=1}^K$. We further denote $\mathcal{T}_m(\mathbf{x}) = \mathbf{R}(\lambda_m)\,\mathbf{x} + \mathbf{t}(\lambda_m)$ as the mapping from world coordinates to the local coordinate system of the m^{th} primitive.

The pointcloud-to-primitive distance, $\mathcal{L}_{X \to P}$, measures the distance from the point cloud to the primitives to ensure that each observation is explained by at least one primitive. It can be expressed as:

$$\mathcal{L}_{X \to P}(\mathcal{X}, \mathcal{P}) = \mathbb{E}_{p(\mathbf{z})} \left[\sum_{\mathbf{x}_i \in \mathcal{X}} \min_{m|z_m=1} \Delta_i^m \right] \tag{A.1}$$

where Δ_i^m denotes the minimal distance from point $\mathbf{x}_i$ to the surface of the m'th primitive:

$$\Delta_i^m = \min_{k=1,..,K} \| \mathcal{T}_m(\mathbf{x}_i) - \mathbf{y}_k^m \|_2 \tag{A.2}$$

Assuming independence of the existence variables $p(\mathbf{z}) = \prod_m p(z_m)$, we can replace the expectations in (A.1) with summations as follows:

$$\mathcal{L}_{X \to P}(\mathcal{X}, \mathcal{P}) = \sum_{z_1} \cdots \sum_{z_M} \left[\sum_{\mathbf{x}_i \in \mathcal{X}} \min_{m|z_m=1} \Delta_i^m \right] p(\mathbf{z}) \tag{A.3}$$

Naïve computation of (A.3) has exponential complexity, i.e. for M primitives it requires evaluating the quantity inside the expectation 2^M times. Our key insight is that (A.3) can be evaluated in linear time if the distances Δ_i^m are sorted. Without loss of generality, we assume that the distances are sorted in ascending order. This allows us to state the following: if the first primitive exists, the first primitive will be the one closest to point $\mathbf{x}_i$ of the target point, if the first primitive does not exist and the second does, then the second primitive is closest to point $\mathbf{x}_i$ and so forth. More formally, this property can be stated as follows:

$$\min_{m|z_m=1} \Delta_i^m = \begin{cases} \Delta_i^1, & \text{if } z_1 = 1 \\ \Delta_i^2, & \text{if } z_1 = 0, z_2 = 1 \\ \vdots & \\ \Delta_i^M, & \text{if } z_m = 0, \dots, z_M = 1 \end{cases} \tag{A.4}$$

Using (A.4) we can simplify (A.3) as follows. We start to carry out the summations over the existence variables one by one. Starting with the summations over z_1, (A.3) becomes:

$$\begin{aligned} \mathcal{L}_{X \to P}(\mathcal{X}, \mathcal{P}) = \sum_{\mathbf{x}_i \in \mathcal{X}} \underbrace{\gamma_1 \sum_{z_2} \cdots \sum_{z_M} \Delta_i^1 \prod_{\bar{m}=2}^{M} p(z_{\bar{m}})}_{(\dagger)} + \\ (1-\gamma_1) \sum_{z_2} \cdots \sum_{z_M} \left[\min_{m \geq 2 | z_m = 1} \Delta_i^m \right] \prod_{\bar{m}=2}^{M} p(z_{\bar{m}}) \end{aligned} \tag{A.5}$$

The expression, marked with $(\dagger)$ corresponds to the case for $z_1 = 1$, namely the 1^{st} primitive is part of the scene. From Eq. A.4, we know that $\min_{m|z_m=1} \Delta_i^m = \Delta_i^1$ for $z_1 = 1$, thus the expression marked with $(\dagger)$, can be simplified as follows,

$$(\dagger) = \gamma_1 \Delta_i^1 \underbrace{\sum_{z_2} \cdots \sum_{z_M} \prod_{\bar{m}=2}^{M} p(z_{\bar{m}})}_{\text{this term evaluates to 1}} = \gamma_1 \Delta_i^1 \tag{A.6}$$

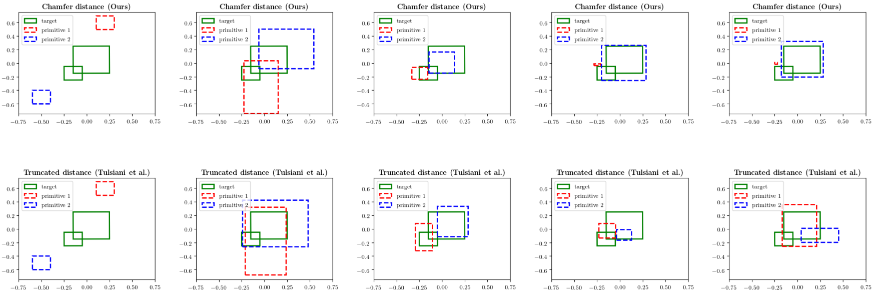
Following this strategy, we can iteratively simplify the remaining terms in (A.5) and arrive at the analytical form of the pointcloud-to-primitive distance stated in (2.10):

$$\begin{aligned} \mathcal{L}_{X \to P}(\mathcal{X}, \mathcal{P}) &= \sum_{\mathbf{x}_i \in \mathcal{X}} \left[\gamma_1 \Delta_i^1 + (1-\gamma_1)\gamma_2 \Delta_i^2 + \cdots + (1-\gamma_1)(1-\gamma_2)\dots\gamma_M \Delta_i^M \right] \\ &= \sum_{\mathbf{x}_i \in \mathcal{X}} \sum_{m=1}^{M} \Delta_i^m \gamma_m \prod_{\bar{m}=1}^{m-1} (1-\gamma_{\bar{m}}) \end{aligned} \tag{A.7}$$

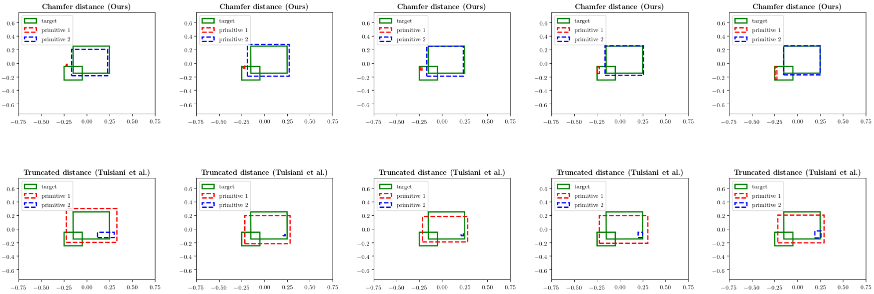
Note that our current formulation assumes that at least one primitive exists in the scene. However, this assumption can be easily relaxed by introducing a “virtual primitive" with a fixed distance to every 3D point on the target point cloud.

A.1.2 *Empirical Analysis of Reconstruction Loss*

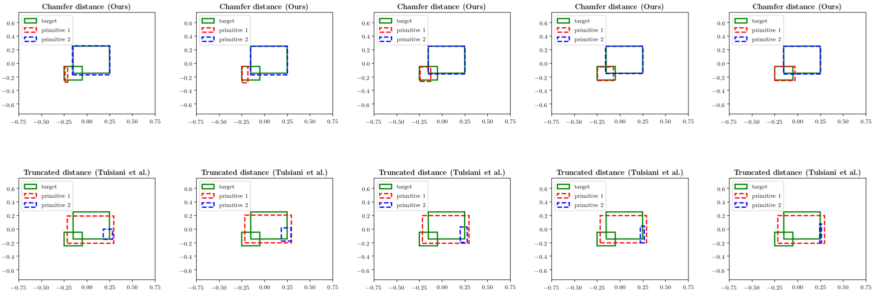
In this section, we provide empirical evidence regarding our claim that our Chamfer-based reconstruction loss leads to more stable training compared to the truncated bi-directional loss of Tulsiani et al. [191]. Towards this goal, we directly optimize/train for the primitive parameters, i.e., not optimizing the weights of a neural network but directly fitting the primitives. We perform this experiment on a 2D toy example and compare the results when using the proposed loss to the results using the truncated distance formulation in [191]. We visualize the evolution of parameters for both optimization objectives as training progresses. We observe that the truncated loss proposed in [191] is more likely to converge to local minima (e.g. figures A.2k-A.2o), while our loss consistently avoids them.



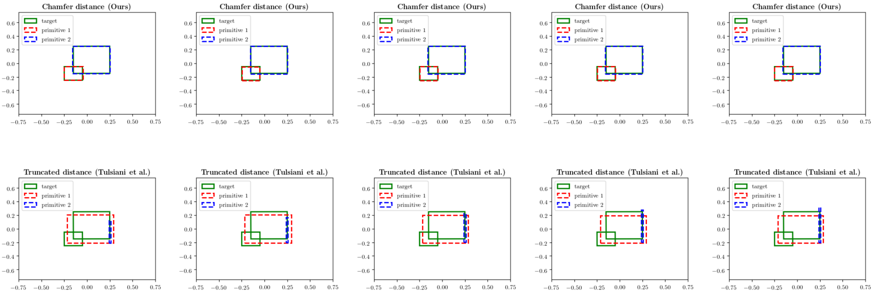
(a) **Iteration** 0 (b) **Iteration** 10 (c) **Iteration** 20 (d) **Iteration** 30 (e) **Iteration** 40



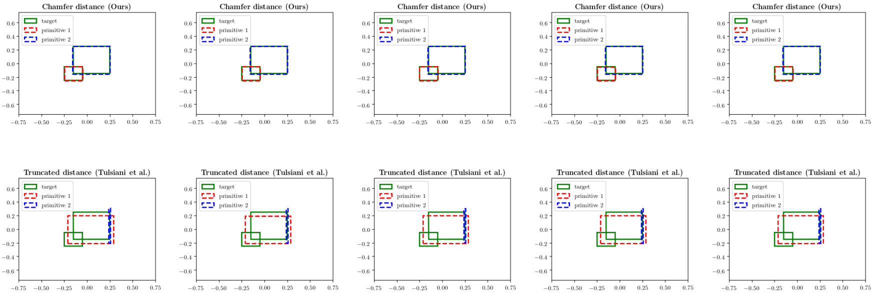
(f) **Iteration** 50 (g) **Iteration** 60 (h) **Iteration** 70 (i) **Iteration** 80 (j) **Iteration** 90



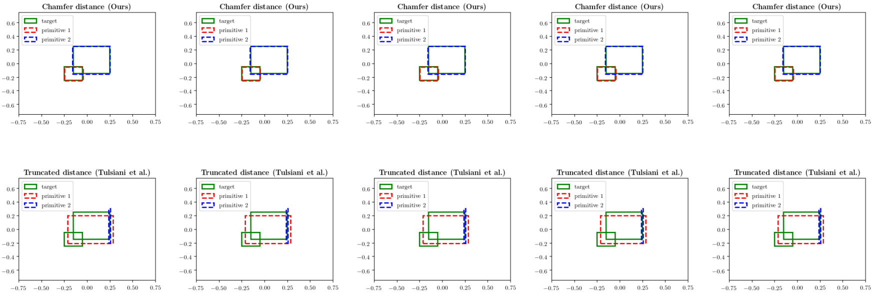
(k) **Iteration** 100 (l) **Iteration** 110 (m) **Iteration** 120 (n) **Iteration** 130 (o) **Iteration** 140



(a) **Iteration** 150 (b) **Iteration** 160 (c) **Iteration** 170 (d) **Iteration** 180 (e) **Iteration** 190



(f) **Iteration** 200 (g) **Iteration** 210 (h) **Iteration** 220 (i) **Iteration** 230 (j) **Iteration** 240



(k) **Iteration** 250 (l) **Iteration** 260 (m) **Iteration** 270 (n) **Iteration** 280 (o) **Iteration** 290

Figure A.2: **Empirical Analysis of Reconstruction Loss**. We illustrate the evolution of two cuboid abstractions using our reconstruction loss with Chamfer distance and the truncated bi-directional loss of Tulsiani et al. [191].

A.2 LEARNING UNSUPERVISED HIERARCHICAL PART DECOMPOSITION OF 3D OBJECTS

A.2.1 *Occupancy Function*

In this section, we provide illustrations of the occupancy function g for different primitive parameters and for different sharpness values. For any point $\mathbf{x} \in \mathbb{R}^3$, we can determine whether it lies inside or outside a superquadric using its implicit surface function which is commonly referred to as the *inside-outside function*:

$$f(\mathbf{x};\lambda) = \left(\left(\frac{x}{\alpha_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{\alpha_2}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{\alpha_3}\right)^{\frac{2}{\epsilon_1}} \tag{A.8}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]$ determine the size and $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2]$ determine the shape of the superquadric. If $f(\mathbf{x};\lambda) = 1.0$, the given point $\mathbf{x}$ lies on the surface of the superquadric, if $f(\mathbf{x};\lambda) < 1.0$ the corresponding point lies inside and if $f(\mathbf{x};\lambda) > 1.0$ the point lies outside the superquadric. To account for numerical instabilities that arise from the exponentiations in (4.7), instead of directly using $f(\mathbf{x};\lambda)$, we follow [81] and use $f(\mathbf{x};\lambda)^{\epsilon_1}$. In addition, we also convert the inside-outside function to an *occupancy function*, $g : \mathbb{R}^3 \to [0,1]$:

$$g(\mathbf{x};\lambda) = \sigma\left(s\left(1 - f(\mathbf{x};\lambda)^{\epsilon_1}\right)\right) \tag{A.9}$$

that results in per-point predictions suitable for the classification problem we want to solve. $\sigma(\cdot)$ is the sigmoid function and s controls the sharpness of the transition of the occupancy function. As a result, if $g(\mathbf{x};\lambda) < 0.5$ the corresponding point lies outside and if $g(\mathbf{x};\lambda) > 0.5$ the point lies inside the superquadric. Fig. A.3 visualizes the range of the implicit surface

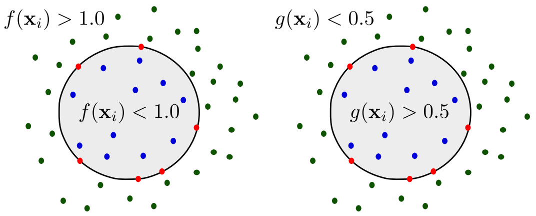
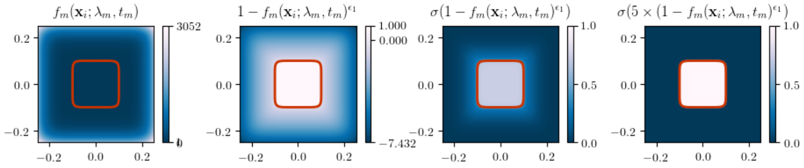
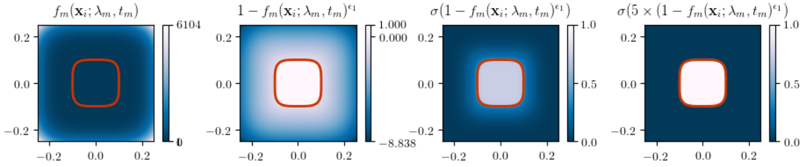


Figure A.3: **Implicit Surface Function of Superquadrics.** We visualize the 2D slice of $f(\mathbf{x}_i)$ and $g(\mathbf{x}_i)$ for a superquadric with $\alpha_1 = \alpha_2 = \alpha_3 = \epsilon_1 = \epsilon_2 = 1$.

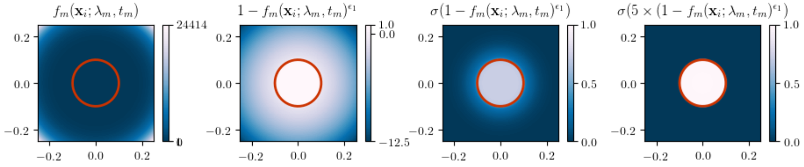
function of superquadrics of (A.8) and (A.9). Fig. A.4+A.5+A.6 visualize the implicit surface function for different values of ϵ_1 and ϵ_2 and different values of sharpness s. We observe that without applying the sigmoid to (4.7) the range of values of (4.7) varies significantly for different primitive parameters.



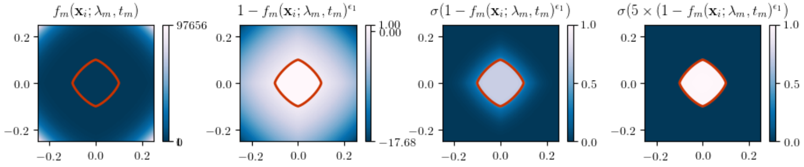
(a) $\epsilon_1 = 0.25$, and $\epsilon_2 = 0.25$



(b) $\epsilon_1 = 0.25$, and $\epsilon_2 = 0.5$

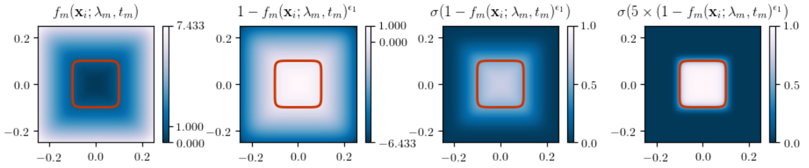


(c) $\epsilon_1 = 0.25$, and $\epsilon_2 = 1.0$

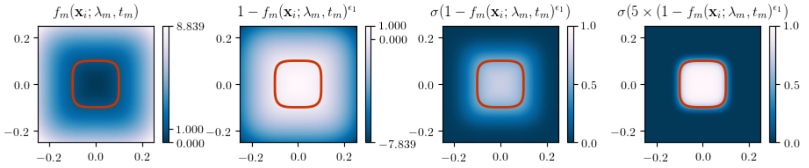


(d) $\epsilon_1 = 0.25$, and $\epsilon_2 = 1.5$

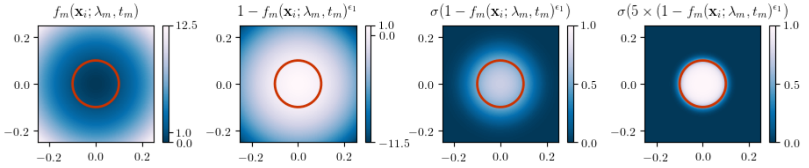
Figure A.4: **Implicit Surface Function.** We visualize the implicit surface function for different primitive parameters and for different sharpness values. The surface boundary is drawn with red.



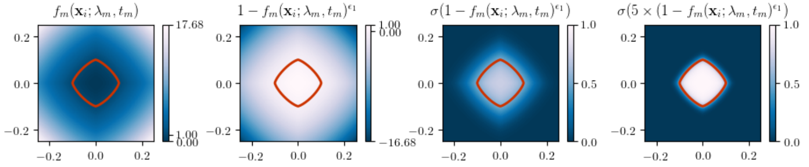
(a) $\epsilon_1 = 1.0$, and $\epsilon_2 = 0.25$



(b) $\epsilon_1 = 1.0$, and $\epsilon_2 = 0.5$

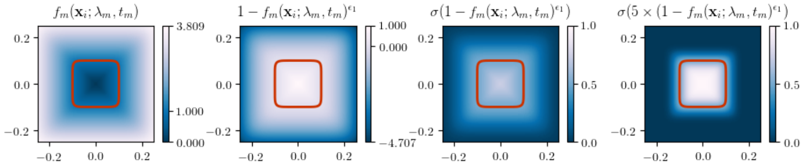


(c) $\epsilon_1 = 1.0$, and $\epsilon_2 = 1.0$

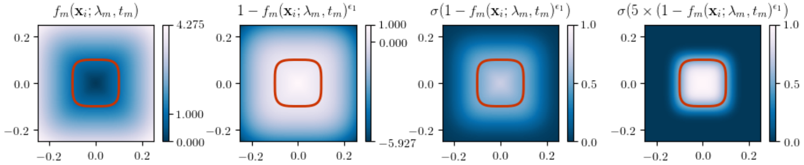


(d) $\epsilon_1 = 1.0$, and $\epsilon_2 = 1.5$

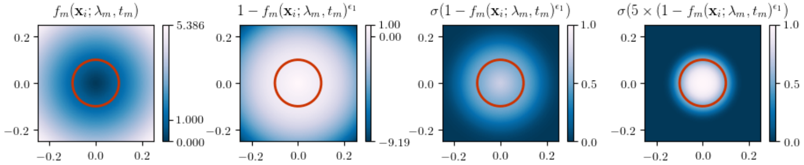
Figure A.5: **Implicit Surface Function.** We visualize the implicit surface function for different primitive parameters and for different sharpness values. The surface boundary is drawn with red.



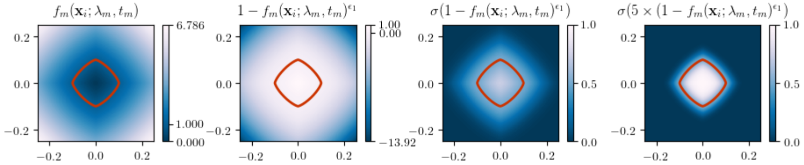
(a) $\epsilon_1 = 1.5$, and $\epsilon_2 = 0.25$



(b) $\epsilon_1 = 1.5$, and $\epsilon_2 = 0.5$



(c) $\epsilon_1 = 1.5$, and $\epsilon_2 = 1.0$



(d) $\epsilon_1 = 1.5$, and $\epsilon_2 = 1.5$

Figure A.6: **Implicit Surface Function.** We visualize the implicit surface function for different primitive parameters and for different sharpness values. The surface boundary is drawn with red.

A.3 ATISS: AUTOREGRESSIVE TRANSFORMERS FOR INDOOR SCENE SYNTHESIS

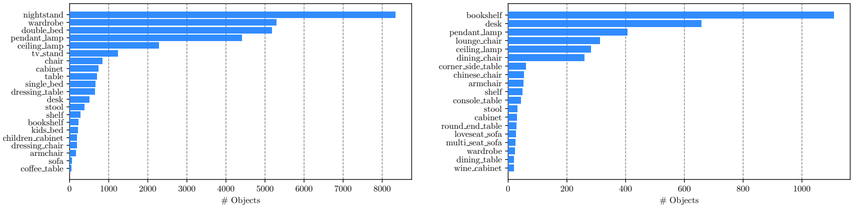
A.3.1 *3D-FRONT Data Preprocessing*

We evaluate our model on the 3D-FRONT dataset [54], which is one of the few available datasets that contain indoor environments. 3D-FRONT contains a collection of 6813 houses with roughly 14629 professionally designed rooms, populated with 3D furniture objects from the 3D-FUTURE dataset [55]. In our experiments, we focused on four room types: (i) bedrooms, (ii) living rooms, (iii) dining rooms and (iv) libraries. Unfortunately, 3D-FRONT contains multiple problematic rooms with unnatural sizes, misclassified objects as well as objects in unnatural positions e.g. outside the room boundaries, lamps on the floor, overlapping objects etc. Therefore, in order to be able to use it, we had to perform thorough filtering to remove problematic scenes. In this section, we present in detail the pre-processing steps for each room type.

The 3D-FRONT dataset provides scenes for the following room types: *bedroom*, *diningroom*, *elderlyroom*, *kidsroom*, *library*, *livingdiningroom*, *livingroom*, *masterbedroom*, *nannyroom*, *secondbedroom* that contain 2287, 3233, 233, 951, 967, 2672, 1095, 3313, 16 and 2534 rooms respectively. Since some room types have very few rooms we do not consider them in our evaluation.

Bedroom: To create training and test data for bedroom scenes, we consider rooms of type *bedroom, secondbedroom* and *masterbedroom*, which amounts to 8134 rooms in total. We start by removing rooms of unnatural sizes, namely rooms that are larger than 6m × 6m in floor size and taller than 4m. Next, we remove infrequent objects that appear in less than 15 rooms, such as chaise lounge sofa, l-shaped sofa, barstool, wine cabinet etc. Subsequently, we filter out rooms that contain fewer than 3 and more than 13 objects, since they amount to a small portion of the dataset. Since the original dataset contained various rooms with problematic object arrangements such overlapping objects, we also remove rooms that have objects that are overlapping as well as misclassified objects e.g. beds being classified as wardrobes. This results in 5996 bedrooms with 21 object categories in total. Fig. A.7a illustrates the number of appearances of each object category in the 5996 bedroom scenes and we remark that the most common category is the nightstand with 8337 occurrences and the least common is the coffee table with 45.

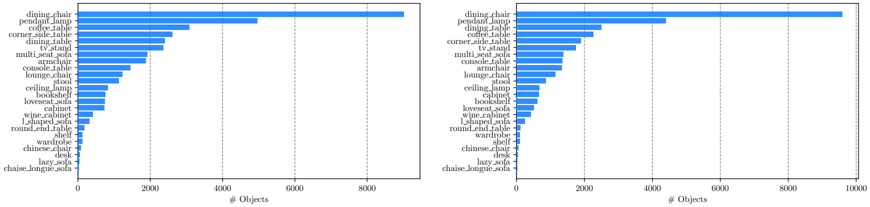
Library: We consider rooms of type *library* that amounts to 967 scenes in total. For the case of libraries, we start by filtering out rooms with unnatural sizes that are larger than 6m × 6m in floor size and taller than 4m. Again we remove rooms that contain overlapping objects, objects positioned outside the room boundaries as well as rooms with unnatural layouts e.g. single chair positioned in the center of the room. We also filter out rooms that contain less than 3 objects and more than 12 objects since they appear less frequently. Our pre-processing resulted in 622 rooms with 19 object categories in total. Fig. A.7b shows the number of appearances of each object category in the 622 libraries. The most common category is the bookshelf with 1109 occurrences and the least common is the wine cabinet with 19.



(a) **Bedrooms** (b) **Libraries**

Figure A.7: **Number of object occurrences in Bedrooms and Libraries.**

Living Room: For the living rooms, we consider rooms of type *livingroom* and *livingdiningroom*, which amounts to 3767 rooms. We follow a similar process as before and we start by filtering out rooms with unnatural sizes. In particular, we discard rooms that are larger than 12m × 12m in floor size and taller than 4m. We also remove uncommon objects that appear in less than 15 rooms such as bed and bed frame. Next, we filter out rooms that contain less than 3 objects and more than 13 objects, since they are significantly less frequent. For the case of living rooms, we observed that some of the original scenes contained multiple lamps without having any other furniture. Since this is unnatural, we also removed these scenes together with some rooms that had either overlapping objects or objects positioned outside the room boundaries. Finally, we also remove any scenes that contain any kind of bed e.g. double bed, single bed, kid bed etc. After our pre-processing, we ended up with 2962 living rooms with 24 object categories in total. Fig. A.8a visualizes the number of occurrences of each object category in the living rooms. We observe that the most common



(a) **Living Rooms** (b) **Dining Rooms**

Figure A.8: **Number of object occurrences in Living Rooms and Dining Rooms.**

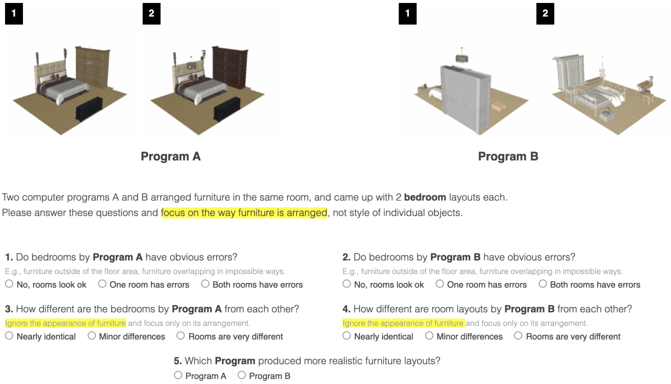
category is the dining chair with 9009 occurrences and the least common is the chaise lounge sofa with 30.

Dining Room: For the dining rooms, we consider rooms of type *diningroom* and *livingdiningroom*, since the *diningroom* scenes amount to only 233 scenes. This results in 3233 rooms in total. For the dining rooms, we follow the same filtering process as for the living rooms and we keep 2625 rooms with 24 objects in total. Fig. A.8b shows the number of occurrences of each object category in the dining rooms. The most common category is the dining chair with 9589 occurrences and the least common is the chaise lounge sofa with 19.

To generate the train, test and validation splits, we split the preprocessed rooms such that 70% is used for training, 20% for testing and 10% for validation. Note that the 3D-FRONT dataset comprises multiple houses that may contain the same room, e.g the exact same object arrangement might appear in multiple houses. Thus splitting train and test scenes solely based on whether they belong to different houses could result in the same room appearing both in train and test scenes. Therefore, instea of randomly selecting rooms from houses but we select from the set of rooms with distinct object arrangements.

A.3.2 *Perceptual Study*

We conducted two paired Amazon Mechanical Turk perceptual studies to evaluate the quality of our generated layouts against FastSynth [163] and SceneFormer [202]. To this end, we first sampled 211 floor plans from the test set and generated 6 scenes per floor plan for each method; no filtering or post-processing was used, and samples were randomly and independently drawn for all methods. Originally, we considered rendering the rooms with the same furniture objects for each floor plan to allow participants to only focus on the layout itself, which is the main focus of this work. However, since the object retrieval is done based on the object dimensions, rescaling the same furniture piece to fit all predicted dimensions would result in unrealistically deformed pieces that could skew perceptual judgements even more heavily. To avoid having participants focusing on the individual furniture pieces, we added prominent instructions to focus on the layout and **not** the properties of selected objects (see Fig. A.9). Each 3D room was rendered as an animated gif using the same camera rotating around the room.



Two computer programs A and B arranged furniture in the same room, and came up with 2 **bedroom** layouts each.
Please answer these questions and focus on the way furniture is arranged, not style of individual objects.

1. Do bedrooms by **Program A** have obvious errors?
E.g., furniture outside of the floor area, furniture overlapping in impossible ways.
○ No, rooms look ok ○ One room has errors ○ Both rooms have errors

2. Do bedrooms by **Program B** have obvious errors?
E.g., furniture outside of the floor area, furniture overlapping in impossible ways.
○ No, rooms look ok ○ One room has errors ○ Both rooms have errors

3. How different are the bedrooms by **Program A** from each other?
Ignore the appearance of furniture and focus only on its arrangement.
○ Nearly identical ○ Minor differences ○ Rooms are very different

4. How different are room layouts by **Program B** from each other?
Ignore the appearance of furniture and focus only on its arrangement.
○ Nearly identical ○ Minor differences ○ Rooms are very different

5. Which **Program** produced more realistic furniture layouts?
○ Program A ○ Program B

Figure A.9: **Perceptual Study UI.** A/B paired questions with rotating 3D scenes (zoom in).

In each user study, users were shown paired question sets: two rooms generated using our method and two generated with the baseline conditioned on the same floor plan. We randomly selected two out of the 6 pre-rendered scenes for the given floor plan, and 5 different workers answered the question set about every floor plan. Namely, the majority of the 6 layouts were shown more than once on average. A / B order was randomized to avoid bias. The question sets posed the same two questions about scenes generated with program A and B, in order to let users focus on the details of the results and to assess errors of the generated layouts. The last question forced participants to choose between A or B, based on which scene looks more realistic.

Specifically, users were instructed to pay attention to errors like interpenetrating furniture and furniture outside of the floor area and answer if none, one or both layouts for each method had errors. We aggregated these statistics to obtain average error rate per layout, with our method performing nearly twice better than the best baseline [163]. The results on realism in Tab. 5.4 (first and second row) specify the fraction of the times users chose the baseline over ours. For example, [163] was judged more realistic than ours only 26.9% of the time. Because there was no intermediate option, this means that 73.1% of the time our method was preferred. The last line in Tab. 5.4, aggregated preference for our method across both studies.

Workers were compensated $0.05 per question set for a total of USD $106. The participation risks involved only the regular risks associated with the use of a computer.

B

ADDITIONAL EXPERIMENTAL RESULTS

B.1 SUPERQUADRICS REVISITED: LEARNING 3D SHAPE PARSING BEYOND CUBOIDS

B.1.1 *Qualitative Results on SURREAL*

In this section, we provide additional qualitative results on the SURREAL human body dataset. In Fig. B.1, we illustrate the predicted primitives of humans in various poses and articulations.

We remark that our model is able to accurately capture the various human body parts using superquadric surfaces. Another interesting aspect of our model, which is also observed in [191], is related to the fact that our model uses the same primitive (highlighted with the same color) to represent the same actual human body part. For example, the head is typically captured using the primitive illustrated with red. For some poses these correspondences are lost. We speculate that this is because the network does not know whether the human is facing in front or behind.

B.2 LEARNING UNSUPERVISED HIERARCHICAL PART DECOMPOSITION OF 3D OBJECTS

B.2.1 *Additional Results on D-FAUST*

In this section, we provide additional qualitative results on the D-FAUST dataset [13]. Furthermore, we also demonstrate that the learned hierarchies are indeed semantic as the same node is used to represent the same part of the human body. Similar to the experiment of Sec. 4.3.3, we evaluate our model on the single-view 3D reconstruction task, namely given a single *RGB image as an input*, our network predicts its geometry as a *tree of primitives as an output*. We compare our model with [141]. Both methods were trained for a maximum number of 32 primitives until convergence. For our method, we set the sharpness value $s = 10$.

In Fig. B.2+B.4, we qualitatively compare our predictions with [141]. We remark that even though [141] is more parsimonious, our predictions are

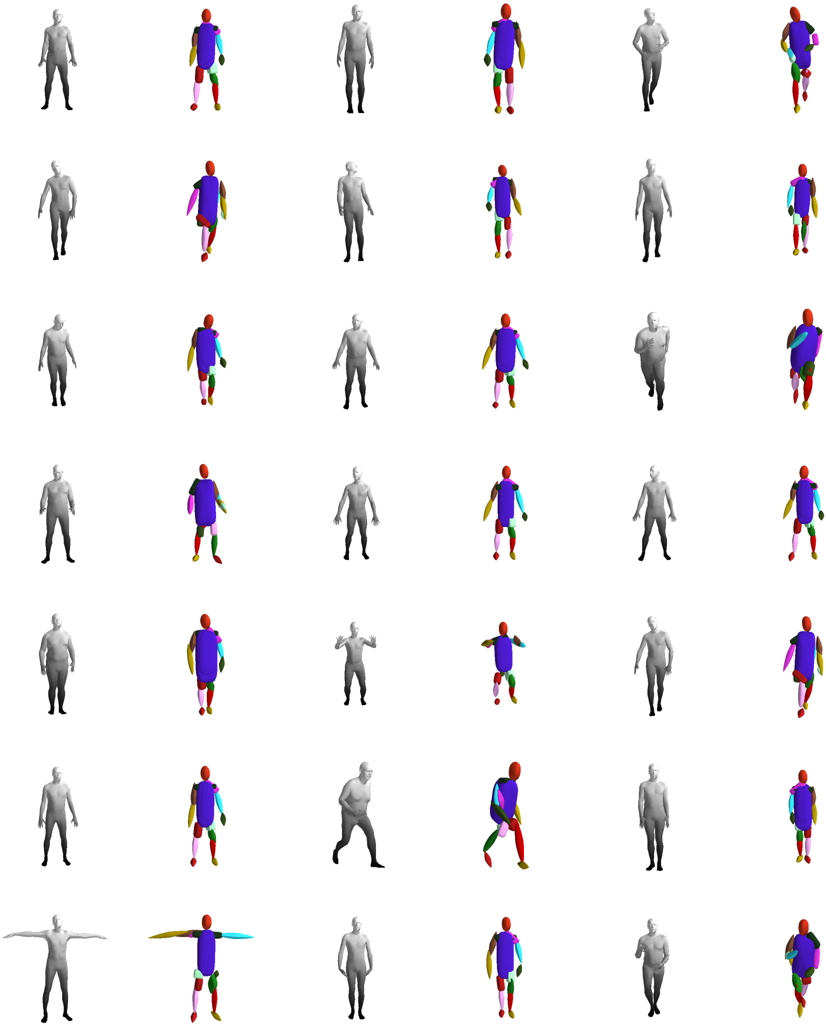
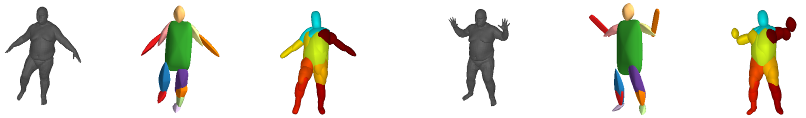
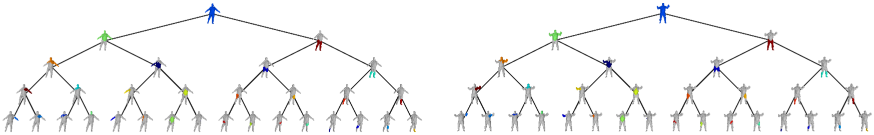


Figure B.1: **Qualitative Results on SURREAL.** Our network learns semantic mappings of body parts across different body shapes and articulations. For instance, the network uses the same primitive for the left forearm across instances.

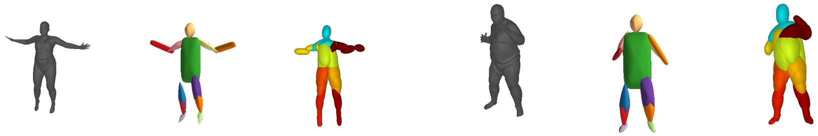
more accurate. For example, we note that our shape reconstructions capture the details of the muscles of the legs that are not captured in [141]. For completeness, we also visualize the predicted hierarchy up to the fourth depth level. Another interesting aspect of our model, which is also observed



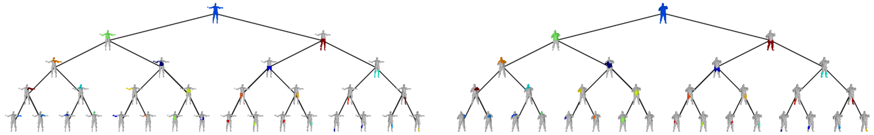
(a) **Input** (b) **SQs[141]** (c) **Ours** (d) **Input** (e) **SQs[141]** (f) **Ours**



(g) **Predicted Hierarchy** (h) **Predicted Hierarchy**



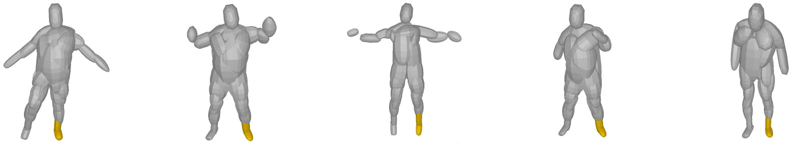
(i) **Input** (j) **SQs[141]** (k) **Ours** (l) **Input** (m) **SQs[141]** (n) **Ours**



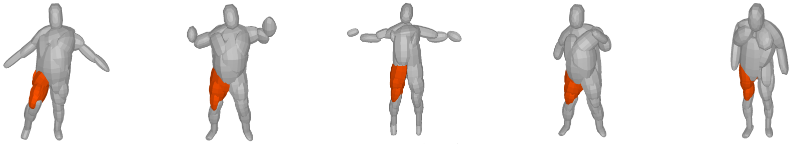
(o) **Predicted Hierarchy** (p) **Predicted Hierarchy**

Figure B.2: **Qualitative Results on D-FAUST.** Our network learns semantic mappings of body parts across different body shapes and articulations while being geometrical more accurate compared to [141].

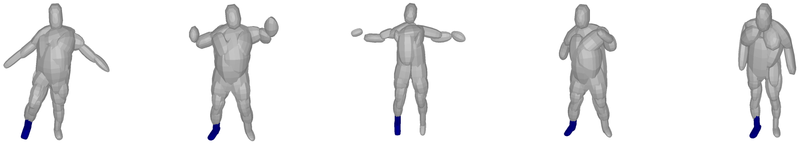
in [141, 191] is related to the semanticness of the learned hierarchies. We note that our model consistently uses the same node for representing the same part of the human body. For instance, node $(4, 15)$, namely the 15-th node at the 4-th depth level, consistently represents the right foot, whereas, node $(4, 12)$ represents the left foot. This is better illustrated in Fig. B.3. In this figure, we only color the primitive associated with a particular node, for various humans, and we remark that the same primitive is used for representing the same body part. Finally, another interesting characteristic



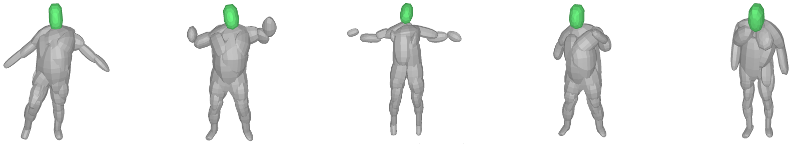
(a) Node $(4,0)$



(b) Node $(3,3)$



(c) Node $(4,3)$

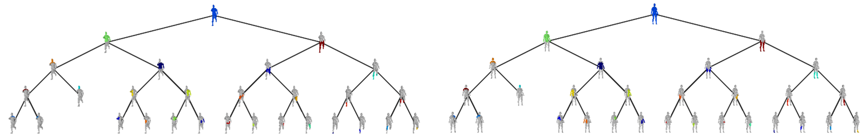


(d) Node $(4,12)$

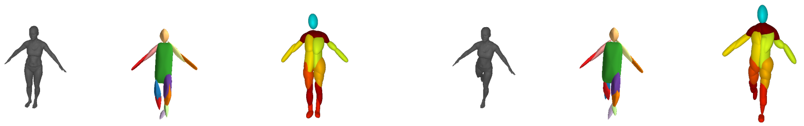
Figure B.3: **Semantic Predictions on D-FAUST.** To illustrate that our model indeed learns semantic hierarchical layouts of parts, here we color a specific node of the tree for various humans and we observe that it consistently corresponds to the same body part.

of our model is related to its ability to use less primitives for reconstructing humans, with smaller bodies. In particular, while the lower part of the human body is consistently represented with the same set of primitives, the upper part can be represented with less depending on the size and the articulation of the human body. This is illustrated in Fig. B.4, where we visualize the predictions of our model for such scenarios.

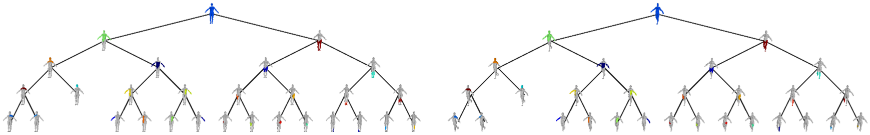
(a) **Input** (b) **SQs[141]** (c) **Ours** (d) **Input** (e) **SQs[141]** (f) **Ours**



(g) **Predicted Hierarchy** (h) **Predicted Hierarchy**



(i) **Input** (j) **SQs[141]** (k) **Ours** (l) **Input** (m) **SQs[141]** (n) **Ours**



(o) **Predicted Hierarchy** (p) **Predicted Hierarchy**

Figure B.4: **Qualitative Results on D-FAUST.** Our network learns semantic mappings of body parts across different body shapes and articulations. Note that the network predicts less primitives for modelling the upper part of the human body.

Below, we provide the full hierarchies of the results on D-FAUST from Fig. 4.5+Fig. 4.9.

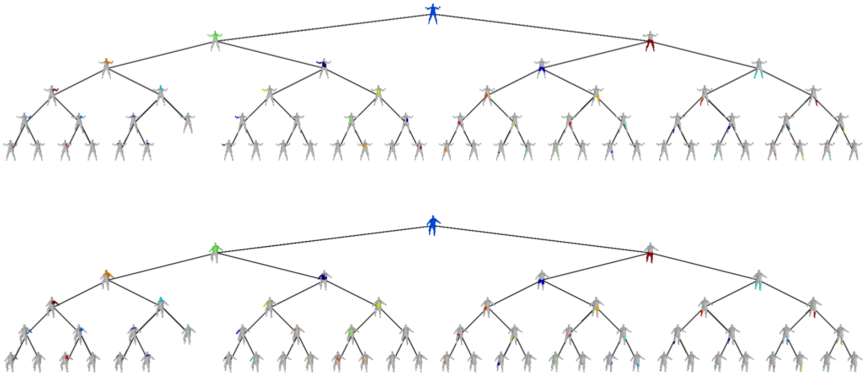


Figure B.5: **Full Hierarchies of Fig. 4.5.** Please zoom-in for details

B.2.2 *Additional Results on ShapeNet*

In this section, we provide additional qualitative results on various object types from the ShapeNet dataset [19]. Furthermore, we also demonstrate the ability of our model to predict semantic hierarchies, where the same node is used for representing the same part of the object. We compare our model qualitatively with [141]. In particular, we train both models on the single-view 3D reconstruction task, using the same image renderings and train/test splits as [34]. Both methods are trained for a maximum number of 64 primitives. For our method, we empirically observed that a sharpness value $s = 10$ led to good reconstructions. Note that we do not compare qualitatively with [61, 36] as they do not provide code. Finally, we also compare our model with [191, 141] on the volumetric reconstruction task, where the input to all networks is a binary voxel grid. For a fair comparison, all models leverage the same feature encoder architecture proposed in [141].

In Fig. B.8+B.7, we qualitatively compare our predictions with [141] for various ShapeNet objects. We observe that our model yields more accurate reconstructions compared to our baseline. Due to the use of the reconstruction quality q_k^d, our model dynamically decides whether a node should be split or not. For example, our model represents the phone in Fig. B.8 (a) using one primitive (root node) and the phone in Fig. B.8 (b), that consists of two parts, with two primitives. This can be also noted for the case of the displays Fig. B.8 (g+j). For more complicated objects, such as aeroplanes, tables and chairs, our network uses more primitives

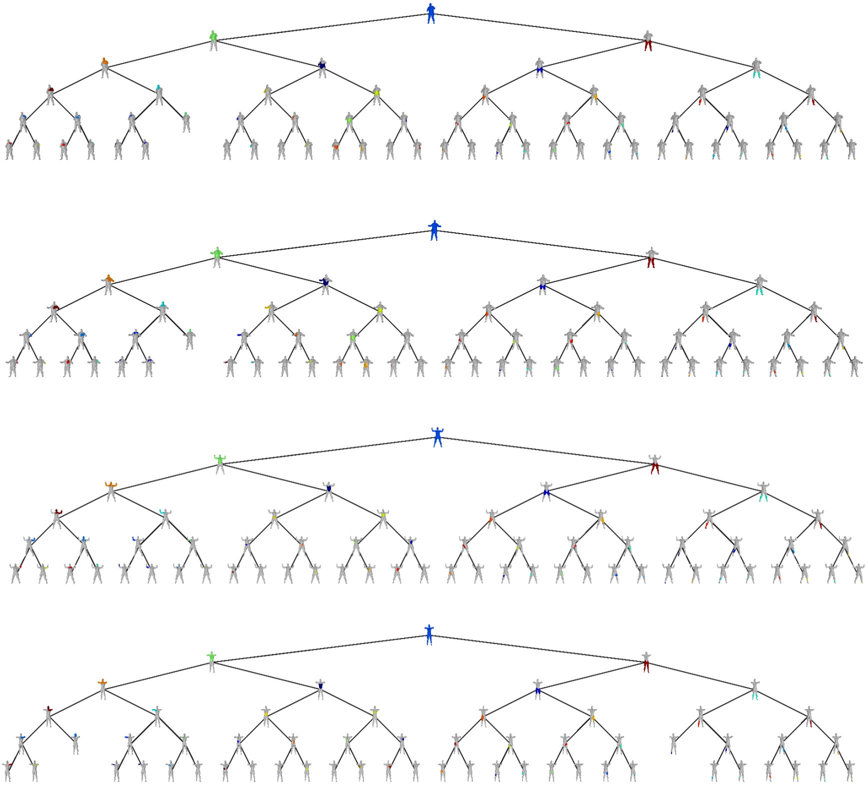
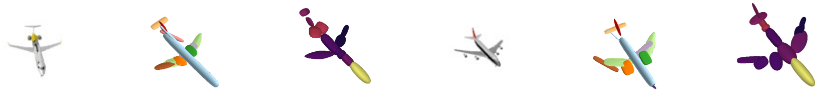
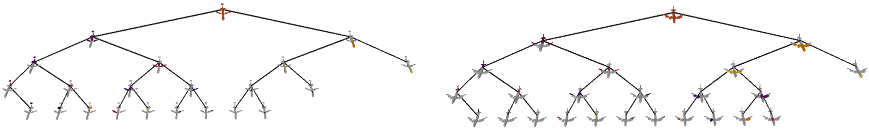


Figure B.6: **Full Hierarchies of Fig. 4.9.** Please zoom-in for details.

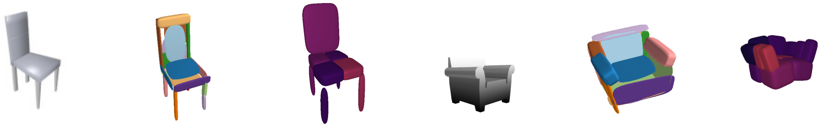
to accurately capture the geometry of the target object. Note that for this experiment we set the threshold for q_k^d to 0.8.



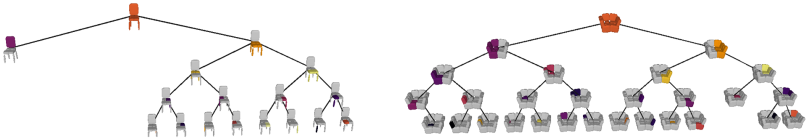
(a) **Input** (b) **SQs[141]** (c) **Ours** (d) **Input** (e) **SQs[141]** (f) **Ours**



(g) **Predicted Hierarchy** (h) **Predicted Hierarchy**



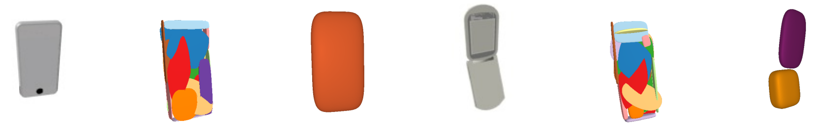
(i) **Input** (j) **SQs[141]** (k) **Ours** (l) **Input** (m) **SQs[141]** (n) **Ours**



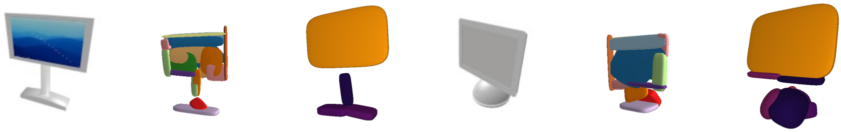
(o) **Predicted Hierarchy** (p) **Predicted Hierarchy**

Figure B.7: **Single Image 3D Reconstruction on ShapeNet.** We visualize the predictions of our model on various ShapeNet objects and compare to [141]. For objects that are represented with more than two primitives, we also visualize the predicted hierarchy.

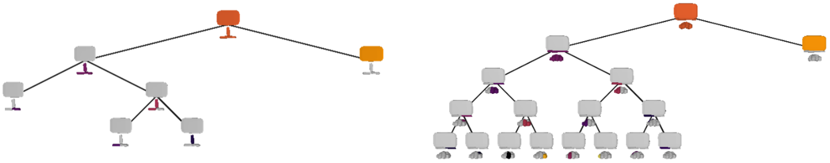
Our network associates the same node with the same part of the object, as it can be seen from the predicted hierarchies in Fig. B.8+B.7. For example, for the displays the second primitive at the first depth level is used for representing the monitor of the display, for the aeroplanes the 4-th primitive in the second depth level is used for representing the front part of the aeroplanes.



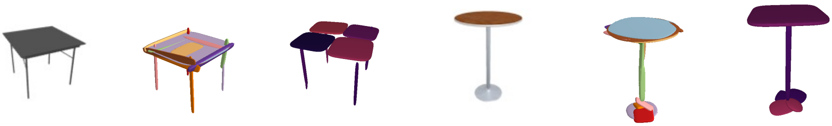
(a) **Input** (b) **SQs[141]** (c) **Ours** (d) **Input** (e) **SQs[141]** (f) **Ours**



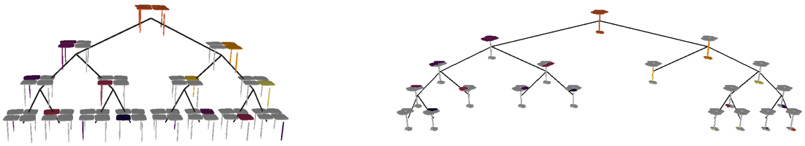
(g) **Input** (h) **SQs[141]** (i) **Ours** (j) **Input** (k) **SQs[141]** (l) **Ours**



(m) **Predicted Hierarchy** (n) **Predicted Hierarchy**



(o) **Input** (p) **SQs[141]** (q) **Ours** (r) **Input** (s) **SQs[141]** (t) **Ours**



(u) **Predicted Hierarchy** (v) **Predicted Hierarchy**

Figure B.8: **Single Image 3D Reconstruction on ShapeNet.** We visualize the predictions of our model on various ShapeNet objects and compare to [141]. For objects that are represented with more than two primitives, we also visualize the predicted hierarchy.

B.3 NEURAL PARTS: LEARNING EXPRESSIVE 3D SHAPE ABSTRACTIONS

B.3.1 *Experiment on D-FAUST*

In this section, we provide additional information regarding our experiments on D-FAUST dataset [13]. D-FAUST contains $38,640$ meshes of humans performing various tasks such as “chicken wings", “running on spot", “shake arms" etc. We follow [143] and use 70%, 20% and 10% for the train, test and validation splits. Furthermore, we filter out the first 20 frames for each sequence that contain the unnatural “neutral pose" necessary for calibration purposes. Note that in contrast to [143], we do not normalize the meshes to the unit cube in order to retain the variety of the human body i.e. tall and short humans. This makes the single view 3D reconstruction task harder, as all models need to efficiently capture both the pose and the size of the human.

Tab. B.1 summarizes the quantitative results from Fig. Fig. 3.4. We notice that for all primitive-based baselines, increasing the number of parts results in improved reconstruction quality in terms of volumetric IoU and Chamfer-L_1 distance. Instead, Neural Parts, decouple the number of primitives from the reconstruction accuracy as the performance of our model is independent of the number of the predicted primitives. This is expected, since our primitives are highly expressive and our model can capture the 3D shape geometry using even a single primitive (see Fig. B.9). We argue that this is a desirable property, as Neural Parts enable us to select the number of parts based on the expected number of semantic parts and not the desired reconstruction quality.

		SQs				H-SQs				CvxNet				Ours			
	OccNet	5	10	25	50	4	8	16	32	5	10	25	50	2	5	8	10
IoU	0.691	0.514	0.565	0.616	0.607	0.559	0.595	0.623	0.610	0.582	0.607	0.622	0.621	0.675	0.673	0.676	0.678
Chamfer-L_1	0.102	0.147	0.129	0.114	0.116	0.161	0.148	0.126	0.129	0.183	0.161	0.139	0.121	0.101	0.097	0.090	0.095

Table B.1: **Single Image 3D Reconstruction on D-FAUST.** We report the volumetric IoU (↑) and the Chamfer-L_1 (↓) wrt. the ground-truth mesh for our model compared to primitive-based methods SQs [141] H-SQs [143], CvxNet [36] for various number of primitives and the non primitive-based OccNet [120]. We observe that our model outperforms all primitive-based baselines with only 2 primitives.

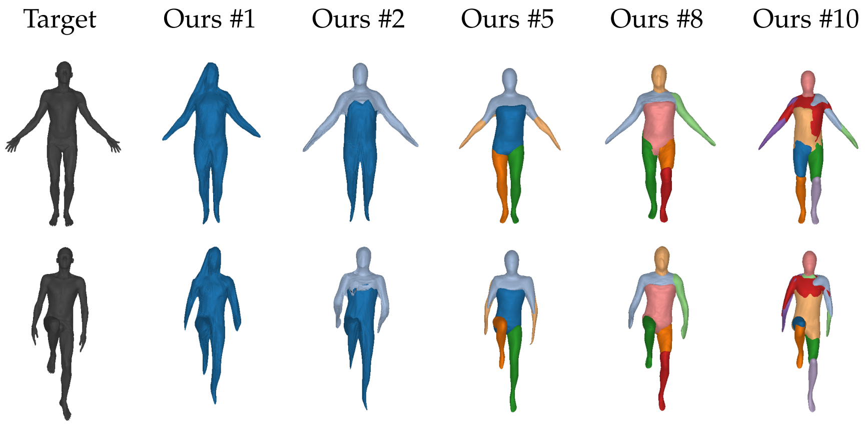


Figure B.9: **Single Image 3D Reconstruction on D-FAUST**. The input image is shown on the first column and the rest contain predictions of our method with different number of primitives.

In Fig. B.10, we provide additional experimental results on the single-view 3D reconstruction task on D-FAUST. In particular, we compare our model with 5 primitives, with CvxNet and SQs with 50 primitives and H-SQs with 32 primitives. We observe that while CvxNet and H-SQs accurately capture the geometry of the human body, their predicted primitives lack any semantic interpretation, as primitives do not correspond to actual geometric parts of the human body. Instead, SQs lead to semantic and parsimonious shape abstractions that are not accurate. On the other hand, Neural Parts achieve both semantic interpretability and high reconstruction quality.

B.3.2 *Experiment on FreiHAND*

In this section, we provide additional information regarding our experiments on FreiHAND dataset [225]. For FreiHAND, we select the first 5000 hand poses and generate meshes using the provided MANO parameters [165]. We render them from a fixed orientation and use 70%, 20%, 10% for the train, test and validation splits. Note that MANO does not generate watertight meshes, thus we need to post-process the meshes in order to create occupancy pairs. This is achieved by adding triangular faces using the following vertex indices 38, 92, 234, 239, 279, 215, 214, 121, 78, 79, 108, 120, 119, 117, 118, 122.

In Fig. B.11, we provide additional qualitative results on the single-view 3D reconstruction task on FreiHAND. We compare our model with

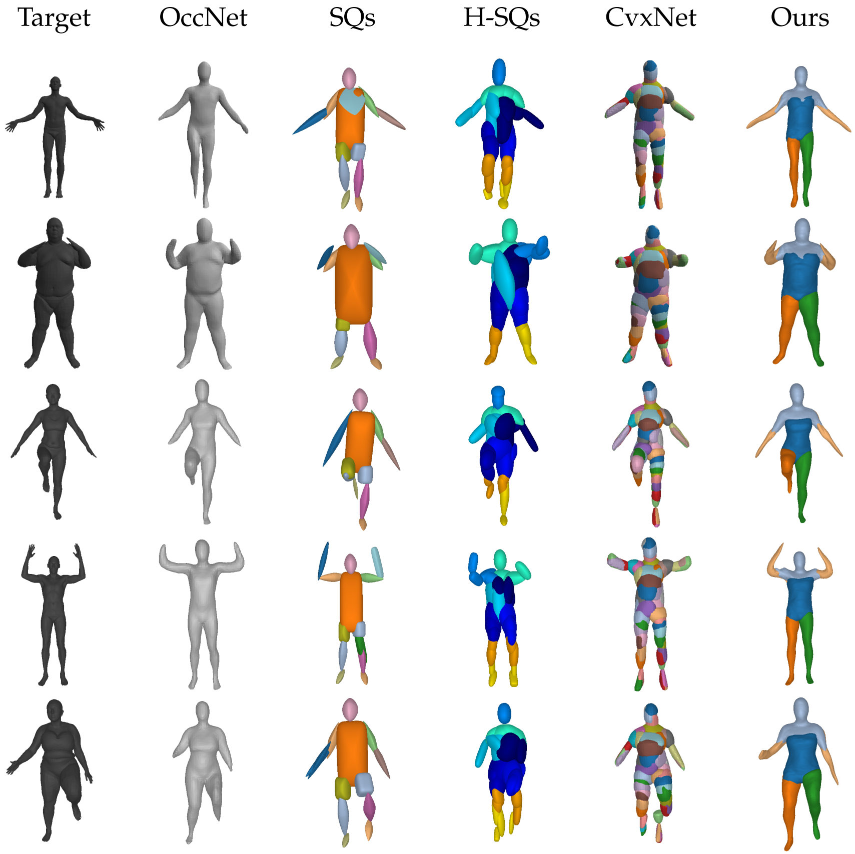


Figure B.10: **Single Image 3D Reconstruction on D-FAUST**. The input image is shown on the first column and the rest contain predictions of all methods: OccNet (second), primitive-based predictions with superquadrics (third and fourth) and convexes (fifth) and ours with 5 primitives (last).

5 primitives to SQs and CvxNet with 5 primitives and H-SQs with 8. Note that for H-SQs we can only use a maximum number of primitives that is a power of 2. H-SQs with 4 primitives performed poorly; thus we increased the number of parts. We observe that Neural Parts yield semantically meaningful parts that accurately capture the hand geometry. Note that accurately modeling objects with non-rigid parts, such as the flexible human fingers, requires primitives that can bend arbitrarily (see thumb in second row Fig. B.11). This is only possible with Neural Parts, as we do not impose any kind of constraint on the primitive shape.

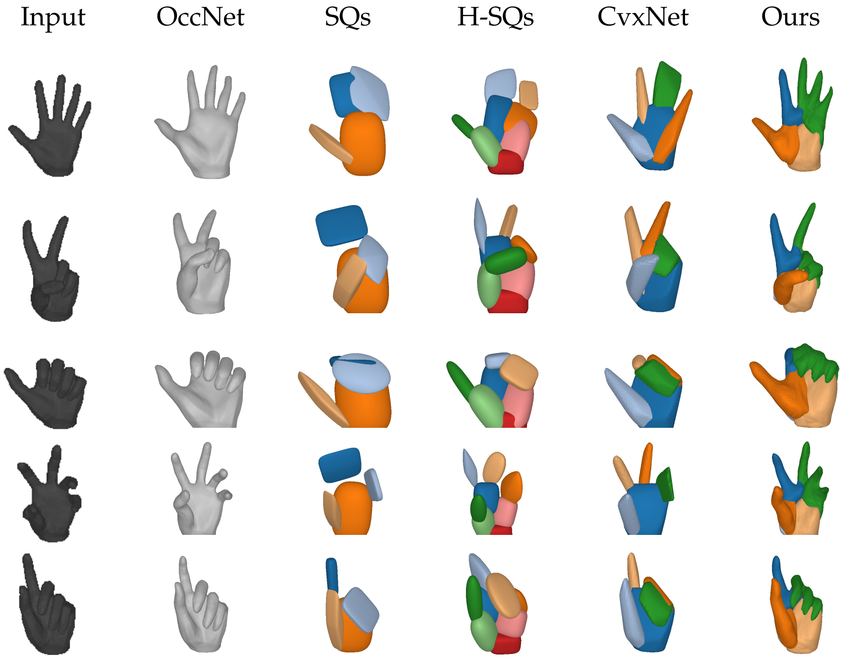


Figure B.11: **Single Image 3D Reconstruction on FreiHAND**. We compare our model with OccNet, SQs and CvxNet with 5 primitives and H-SQs with 8 primitives.

B.3.3 *Experiment on ShapeNet*

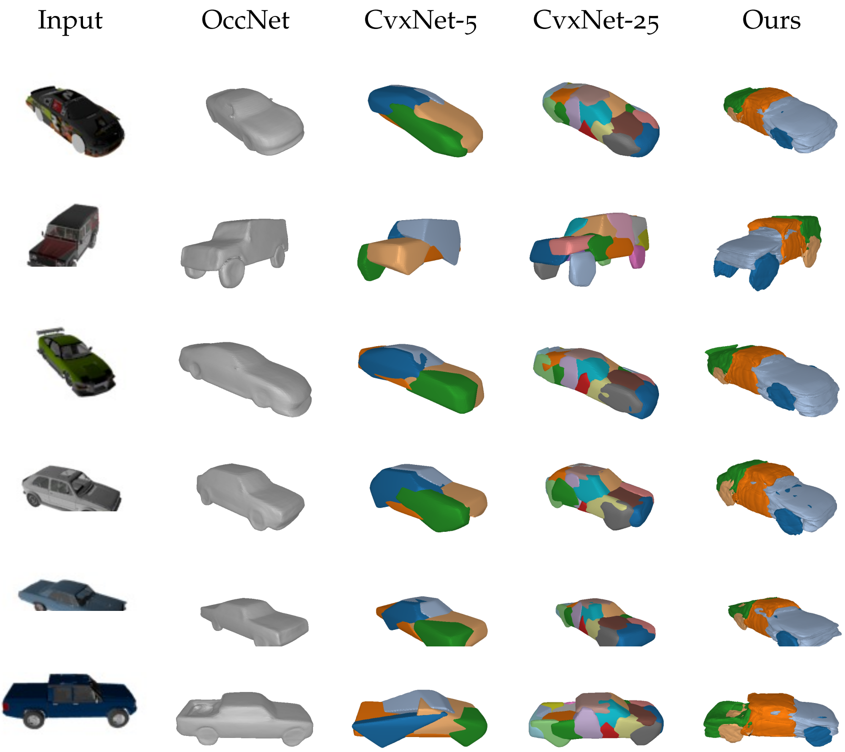


Figure B.12: **Single Image 3D Reconstruction on ShapeNet Cars.** We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5 times less primitives.

For our experiments on the ShapeNet dataset [19] we use the same image renderings and train/test splits of Choy et al. [34]. In order to determine whether points lie inside or outside the target mesh (i.e. for generating the occupancy pairs $\mathcal{X}_o$) we need the meshes to be watertight. For this, we follow [120] and use the code provided by Stutz et al. [180][1] which performs TSDF-fusion on random depth renderings of the object, to create watertight meshes of the object. In Fig. B.12, we provide additional qualitative results

---

[1] https://github.com/davidstutz/mesh-fusion

	OccNet	CvxNet - 5	CvxNet - 25	Ours
IoU	0.763	0.650	0.666	**0.697**
Chamfer-L_1	0.186	0.218	0.210	**0.185**

Table B.2: **Single Image 3D Reconstruction on ShapeNet Cars.** Quantitative evaluation of our method against OccNet [120] and CvxNet [37] with 5 and 25 primitives.

on various ShapeNet cars and compare our model with 5 primitives to CvxNet with 5 and 25 and with the non primitive-based OccNet.

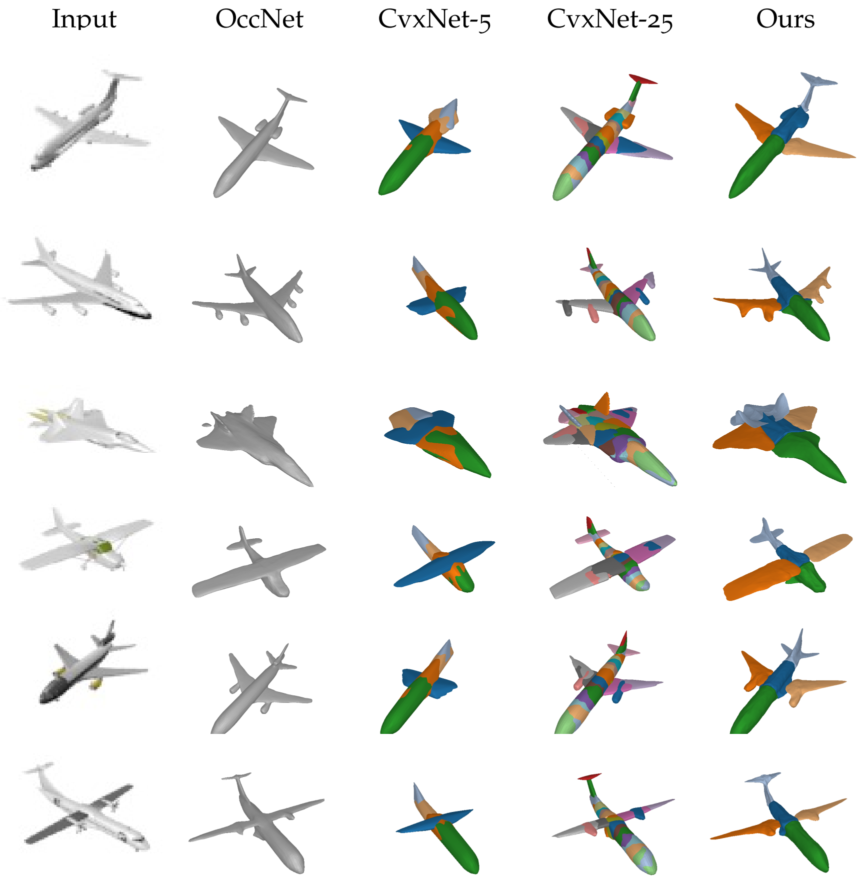


Figure B.13: **Single Image 3D Reconstruction on ShapeNet Airplanes.** We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5 times less primitives.

Subsequently, we train Neural Parts with 5 primitives and compare with CvxNet with 5 and 25 primitives on ShapeNet planes (Fig. B.13 + Tab. B.3), ShapeNet lamps (Fig. B.14 + Tab. B.4) and ShapeNet chairs (Fig. B.15 + Tab. B.5). In Fig. B.13, we provide a qualitative evaluation of our model with CvxNet and OccNet on various planes. We observe that Neural Parts yield more geometrically accurate and semantically meaningful shape abstractions than CvxNet with multiple primitives, i.e. the same primitive is consistently used for representing the tail and the wings of the airplanes. This is also validated quantitatively in Tab. B.3, where we see that our model outperforms CvxNet in terms of IoU and Chamfer-L_1 both with 5 and 25

	OccNet	CvxNet - 5	CvxNet - 25	Ours
IoU	0.451	0.425	0.448	**0.454**
Chamfer-L_1	0.218	0.267	0.245	**0.220**

Table B.3: **Single Image Reconstruction on ShapeNet Airplanes.** Quantitative evaluation of our method against OccNet [120] and CvxNet [37] with 5 and 25 primitives.

primitives. We observe that CvxNet with 5 primitives cannot represent fine details and as a result parts of the target object are not captured, e.g. the turbines of airplanes.

This becomes more evident for the case of lamps, where 5 primitives do not have enough representation power for modelling complex shapes such as the lamp shade or the lamp body (see Fig. B.14 third column), thus entire object parts are missing. Similarly, also for the case of chairs, CvxNet with 5 primitives fail to accurately capture the object's geometry (see Fig. B.15 third column). For example, 5 primitives are not enough for representing the legs of the chair. Instead our model with the same number of parts consistently captures the 3D geometry. On the contrary, CvxNet with 25 primitives yield more geometrically accurate reconstructions, however, the reconstructed primitives are not as semantically meaningful.

	OccNet	CvxNet - 5	CvxNet - 25	Ours
IoU	0.339	0.246	0.278	**0.318**
Chamfer-L_1	0.514	0.613	0.537	**0.347**

Table B.4: **Single Image 3D Reconstruction on ShapeNet Lamps.** Quantitative evaluation of our method against OccNet [120] and CvxNet [37] with 5 and 25 primitives.

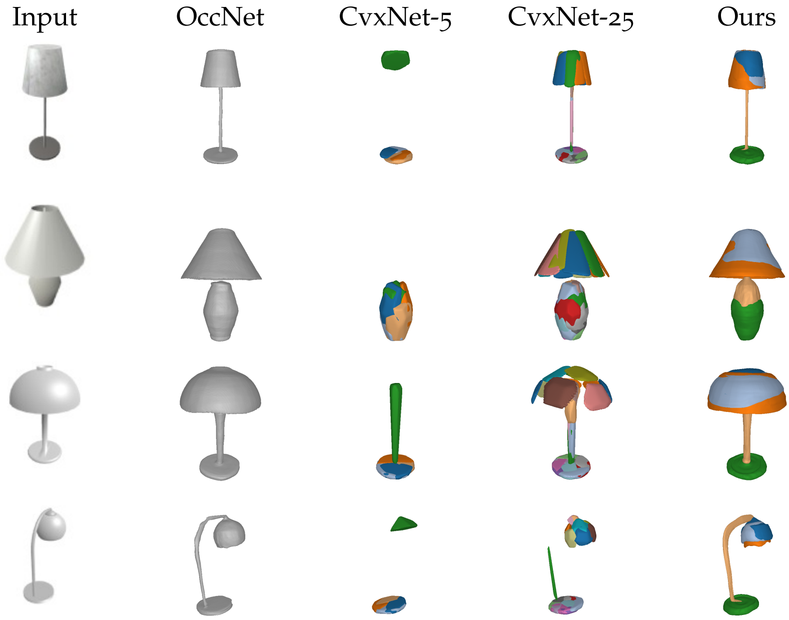


Figure B.14: **Single Image 3D Reconstruction on ShapeNet Lamps.** We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5 times less primitives.

	OccNet	CvxNet - 5	CvxNet - 25	Ours
IoU	0.432	0.364	0.392	**0.412**
Chamfer-L_1	0.312	0.587	0.557	**0.337**

Table B.5: **Single Image Reconstruction on ShapeNet Chairs.** Quantitative evaluation of our method against OccNet [120] and CvxNet [37] with 5 and 25 primitives.

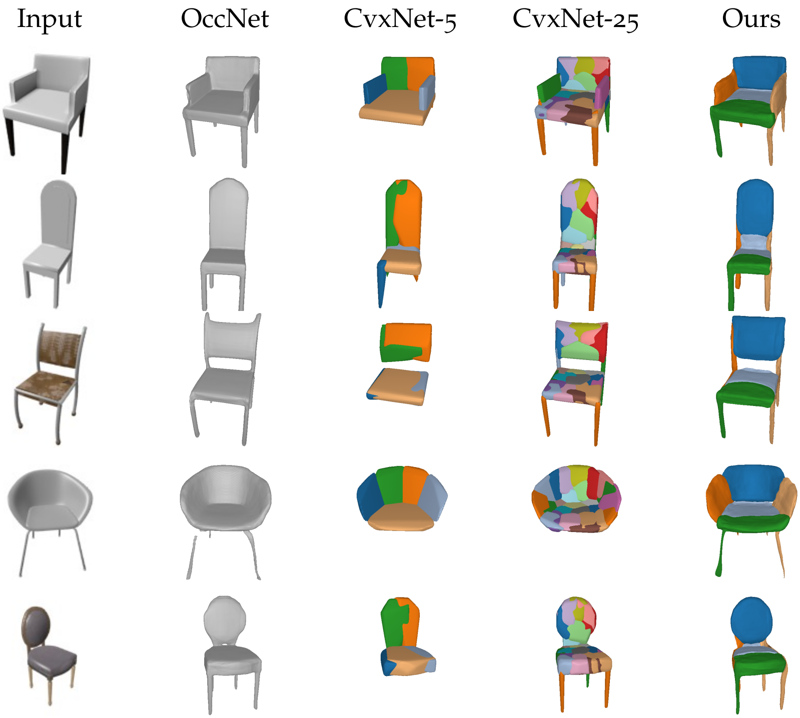


Figure B.15: **Single Image 3D Reconstruction on ShapeNet Chairs.** We compare Neural Parts to OccNet and CvxNet with 5 and 25 primitives. Our model yields semantic and more accurate reconstructions with 5 times less primitives.

B.3.4 *Semantically Consistent Abstractions*

In this section, we provide additional information regarding our Semantic Consistency experiment from Sec. 3.4.5. Furthermore, since D-FAUST contains 4D scans of humans in motion, we also evaluate the temporal consistency of the predicted primitives on various actions.

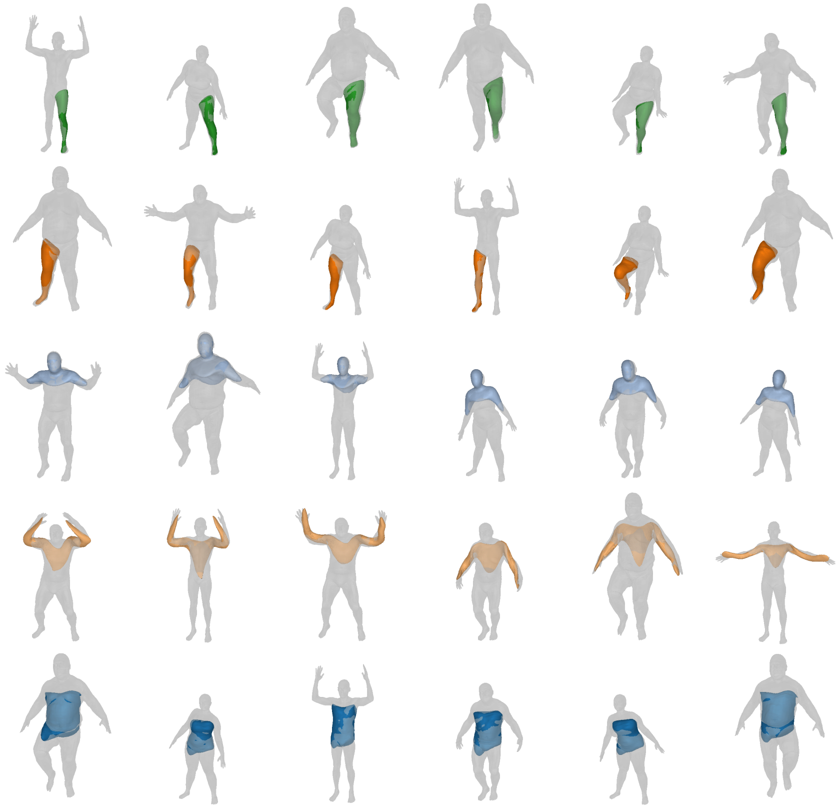


Figure B.16: **Representation Consistency.** Our predicted primitives are consistently used for representing the same human part for different humans.

B.3.4.1 *Temporal Consistency*

In this section, we provide a qualitative analysis on the temporal consistency of the predicted primitives. Results are summarized in Fig. B.17. We observe that Neural Parts consistently use the same primitive for representing the same object part regardless of the breadth of the part's motion. Notably, this temporal consistency is an emergent property of our method and not one that is enforced with any kind of loss.

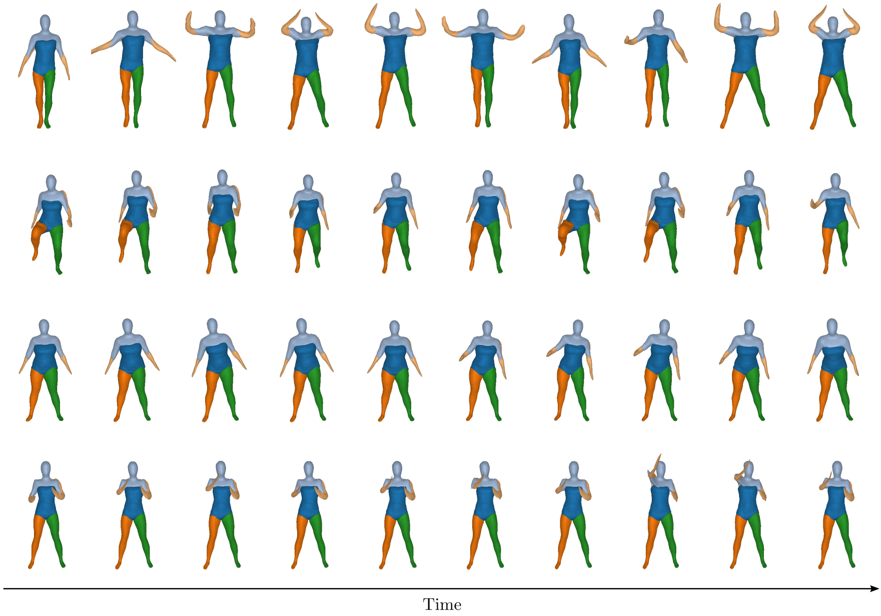


Figure B.17: **Temporal Consistency of Predicted Primitives.** We note that Neural Parts yield primitives that preserve their semantic identity while different humans perform various actions.

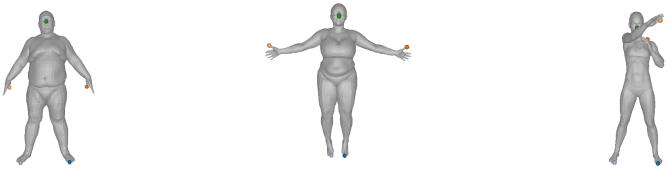


Figure B.18: **Semantic Vertices by SMPL-X.** We highlight the 5 vertex indices that we use to identify left thumb (L-thumb), right thumb (R-thumb), left toe (L-toe), right toe (R-toe) and nose.

B.3.4.2 *Semantic Consistency*

In this experiment, we provide additional information regarding the Semantic Consistency experiment from Sec. 3.4.5. In particular, we evaluate whether a specific human part is represented consistently by the same primitive (see Fig. B.16). To measure this quantitatively, we select 5 representative vertices on each target mesh (the vertex indices are provided by SMPL-X [145]) and measure the classification accuracy of those points when using the label of the closest primitive. A visualization of the 5 vertices on the human body is given in Fig. B.18. Note that the vertex indices are consistent across all subjects because D-FAUST meshes are generated using SMPL [116].

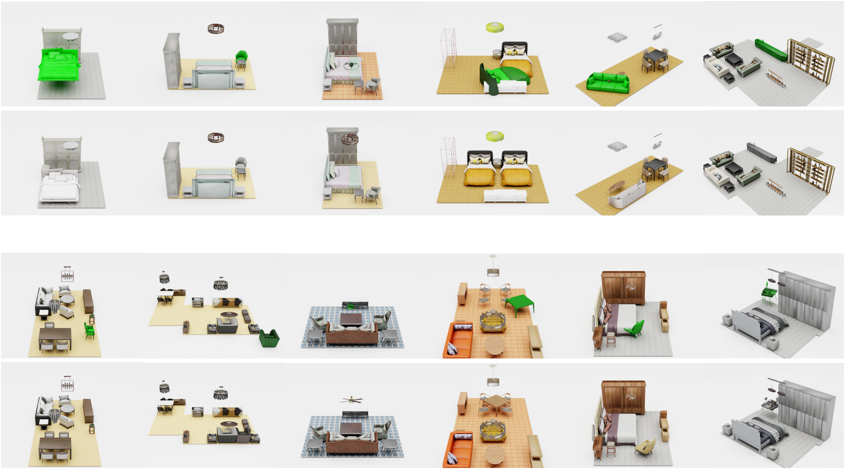


Figure B.19: **Failure Case Detection and Correction**. Starting from a room with an unnatural object arrangement, our model identifies the problematic objects (first row and third row, in green) and relocates them into meaningful positions (second and fourth row).

B.4 ATISS: AUTOREGRESSIVE TRANSFORMERS FOR INDOOR SCENE SYNTHESIS

B.4.1 *Applications*

In this section, we provide additional qualitative results for various interactive applications that benefit greatly by our unordered set formulation.

B.4.1.1 *Failure Case Detection And Correction*

In this experiment, we investigate whether our model is able to identify unnatural furniture layouts and reposition the problematic objects such that they preserve their functional properties. In particular, we identify problematic objects as those with low likelihood and as soon as a problematic object is identified, we sample a new location from our generative model to reposition it. Fig. B.19 shows additional qualitative results on this task. The first and third row show examples of unnatural object arrangements, together with the problematic object, highlighted in green, for each scenario. We note that our model successfully identifies objects in unnatural positions e.g. flying bed (first row, first column Fig. B.19), light inside the

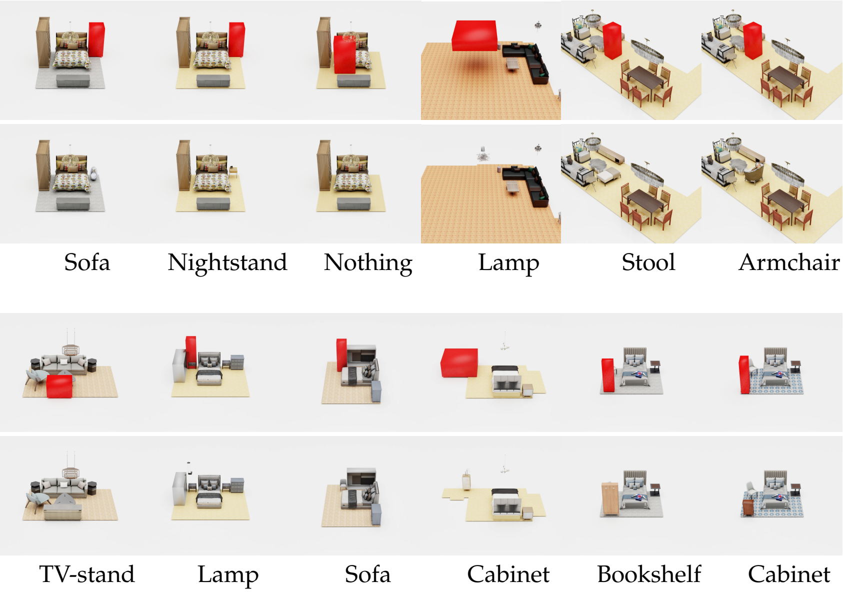


Figure B.20: **Object Suggestion.** A user specifies a region of acceptable positions to place an object (marked as red boxes, first and third row) and our model suggests suitable objects (second and fourth row) to be placed in this location.

bed (first row, third column Fig. B.19) or table outside the room boundaries (third row, fourth column Fig. B.19) as well as problematic objects that do not necessarily look unnatural, such as a cabinet blocking the corridor (first row, sixth column Fig. B.19), a chair facing the wall (third row, first column Fig. B.19) or a lamp being too close to the table (third row, third column Fig. B.19). After having identified the problematic object, our model consistently repositions it at plausible position.

B.4.1.2 *Object Suggestion*

For this task, we examine the ability of our model to provide object suggestions given a scene and user specified location constraints. For this experiment, the user only provides location constraints, namely valid positions for the centroid of the object to be generated. Fig. B.20 shows examples of the location constraints, marked with red boxes, (first and third row) and the corresponding objects suggested by our model (second and fourth row). We observe that our model consistently makes plausible suggestions, and

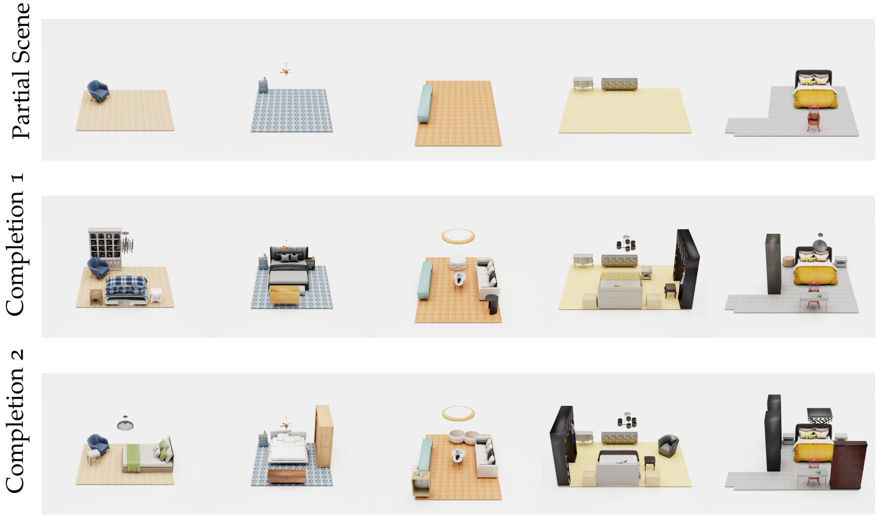


Figure B.21: **Scene Completion.** Starting from a partially complete scene (first row), we visualize two examples of scene completions using our model (second and third row).

for the cases that a user specifies a region that overlaps with other objects in the scene, our model suggests adding nothing (first row, third column Fig. B.20). In Fig. B.20, we also provide two examples, where our model makes different suggestions based on the same location constraints, such as sofa and nightstand for the scenario illustrated in the first and second column and stool and armchair for the scenario illustrated in the fifth and sixth column in the first row.

B.4.1.3 *Scene Completion*

Starting from a partial scene, we want to evaluate the ability of our model to generate plausible object arrangements. To generate the partial scenes, we randomly sample scenes from the test set and remove the majority of the objects in them. Fig. B.21 shows examples for various partial rooms (first row Fig. B.21), as well as two alternative scene completions using our model (second and third row Fig. B.21). We observe that our model generates diverse arrangements of objects that are consistently meaningful. For example, for the case where the partial scene consists of a chair and

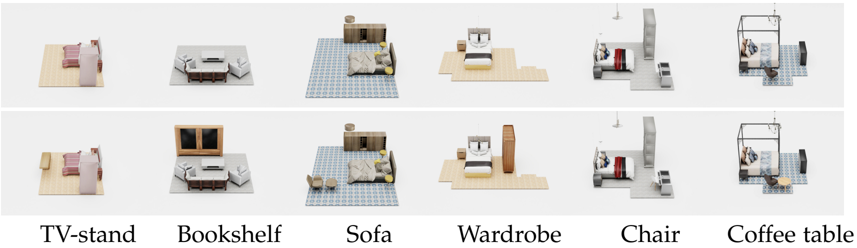


Figure B.22: **Object Placement**. Starting from a partially complete scene, the user specifies an object to be added in the scene and our model places it at a reasonable position. The first rows illustrates the starting scene and the second row the generated scened using the user specified object (third row).

a bed (last column Fig. B.21), our model generates completions that have nightstands surrounding the bed as well as a desk in front of the chair.

B.4.1.4 *Object Placement*

Finally, we showcase the ability of our model to add a specific object in a scene on demand. Fig. B.22 illustrates the original scene (first row) and the complete scene (second row) using the user specified object (third row). To perform this task, we condition on the given scene and instead of sampling from the predicted object category distribution, we use the user provided object category and sample the rest of the object attributes i.e. translation, size and orientation. Also in this task, we note that the generated objects are realistic and match the room layout.

B.4.2 *Scene Synthesis*

In this section, we provide additional qualitative results for our scene synthesis experiment on the four 3D-FRONT rooms. Moreover, since, we repeat the FID score and classification accuracy computation 10 times, in Tab. B.6, we also report the standard deviation for completeness.

Conditioned on a floor plan, we evaluate the performance of our model on generating plausible furniture arrangements and compare with Fast-Synth [163] and SceneFormer [202]. Fig. B.28 provides a qualitative comparison of generated bedroom scenes conditioned on the same floor layout using our model and our baselines. We observe that in contrast to [163, 202], our model consistently generates layouts with more diverse objects. In

	FID Score (↓)			Scene Classification Accuracy			Category KL Divergence (↓)		
	FastSynth	SceneFormer	Ours	FastSynth	SceneFormer	Ours	FastSynth	SceneFormer	Ours
Bedrooms	40.89 ± 0.5098	43.17 ± 0.6921	**38.39** ± 0.3392	0.883 ± 0.0010	0.945 ± 0.0009	**0.562** ± 0.0228	0.0064	**0.0052**	0.0085
Living	61.67 ± 1.2136	69.54 ± 0.9542	**33.14** ± 0.4204	0.945 ± 0.0010	0.972 ± 0.0010	**0.516** ± 0.0075	0.0176	0.0313	**0.0034**
Dining	55.83 ± 1.0078	67.04 ± 1.3043	**29.23** ± 0.3533	0.935 ± 0.0019	0.941 ± 0.0008	**0.477** ± 0.0027	0.0518	0.0368	**0.0061**
Library	37.72 ± 0.4501	55.34 ±0.1056	**35.24** ± 0.2683	0.815 ± 0.0032	0.880 ± 0.0009	**0.521** ± 0.0048	0.0431	0.0232	**0.0098**

Table B.6: **Scene Syntheis Quantitative Comparison on 3D-FRONT.** We report the FID score (↓) at 256^2 pixels, the KL divergence (↓) between the distribution of object categories of synthesized and real scenes and the real vs. synthetic classification accuracy for all methods. Classification accuracy closer to 0.5 is better.

particular, [202] typically generates bedrooms that consist only of a bed, a wardrobe and less frequently also a nightstand, whereas both our model and FastSynth synthesize rooms with more diverse objects. Similarly generated scenes for living rooms and dining rooms are provided in Fig. B.29 and Fig. B.30 respectively. We observe that for the case of living rooms and dining rooms both baselines struggle to generate plausible object arrangements, namely generated objects are positioned outside the room boundaries, have unnatural sizes or populate a small part of the scene. We hypothesize that this might be related to the significantly smaller amount of training data compared to bedrooms. Instead our model, generates realistic living rooms and dining rooms. For the case of libraries (see Fig. B.31), again both [163, 202] struggle to generate functional rooms.

B.4.2.1 *Object Co-occurrence*

To further validate the ability of our model to reproduce the probabilities of object co-occurrence in the real scenes, we compare the probabilities of object co-occurrence of synthesized scenes using our model, FastSynth [163] and SceneFormer [202] for all room types. In particular, in this experiment, we generate 5000 scenes using each method and report the difference between the probabilities of object co-occurrences between real and synthesized scenes. Fig. B.23 summarizes the absolute differences for the bedroom scenes. We observe that our model better captures the object co-occurrence than baselines since the absolute differences for most object pairs are consistently smaller.

This is also validated for the case of living rooms (Fig. B.24), dining rooms (Fig. B.25) and libraries (Fig. B.26), where our model better captures the object co-occurrences than both FastSynth [163] and SceneFormer [202]. Note that from our analysis it becomes evident that while our method better reproduces the probabilities of object co-occurrence from the real scenes, all

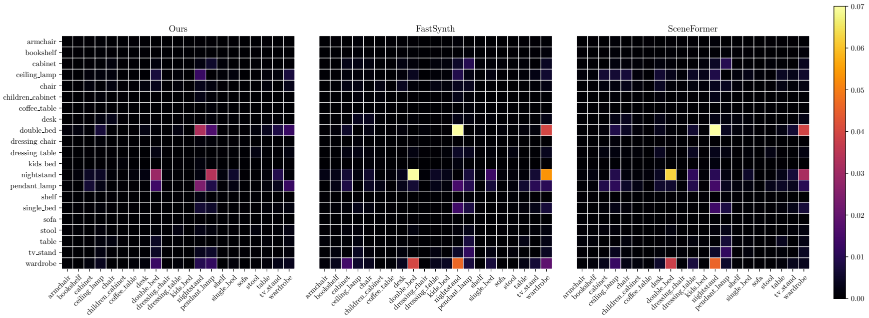


Figure B.23: **Absolute Difference between Object Co-occurrence in Bedrooms.** We visualize the absolute difference of the probabilities of object co-occurrence computed between real and synthesized scenes using ATISS (left), FastSynth (middle) and SceneFormer (right). Larger differences correspond to warmer colors and are worse.

methods are able to generate scenes with plausible object co-occurrences. This is expected, since learning the categories of objects to be added in a scene is a significantly easier task in comparison to learning their sizes and positions in 3D space.

Finally, in Fig. B.27, we visualize the per-object difference in frequency of occurrence between synthesized and real scenes from the test set for all room types. We observe that our model generates object arrangements with comparable per-object frequencies to real rooms. In particular, for the case of living rooms (B.27b), dining rooms (B.27c) and libraries (B.27d) that are more challenging rooms types due to their smaller size, our model has an even smaller discrepancy wrt. the per-object frequencies.

B.4.2.2 *Computational Requirements*

In this section, we provide additional details regarding the computational requirements of our method, presented in Tab. 5.2 and Tab. 5.3. We observe that ATISS requires significantly less time to generate a scene compared to [202, 163]. Note that the computational cost varies depending on the room type, due to the different average number of objects for each room type. Living rooms and dining rooms are typically larger in size, thus more objects need to be generated to cover the empty space. All reported timings are measured on a machine with an NVIDIA GeForce GTX 1080 Ti GPU.

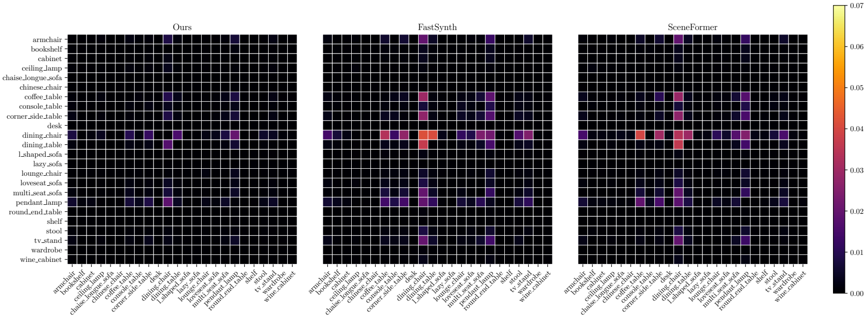


Figure B.24: **Absolute Difference between Object Co-occurrence in Living Rooms.** We visualize the absolute difference of the probabilities of object co-occurrence computed between real and synthesized scenes using ATISS (left-most column), FastSynth (middle column), SceneFormer (right-most column). Lower is better.

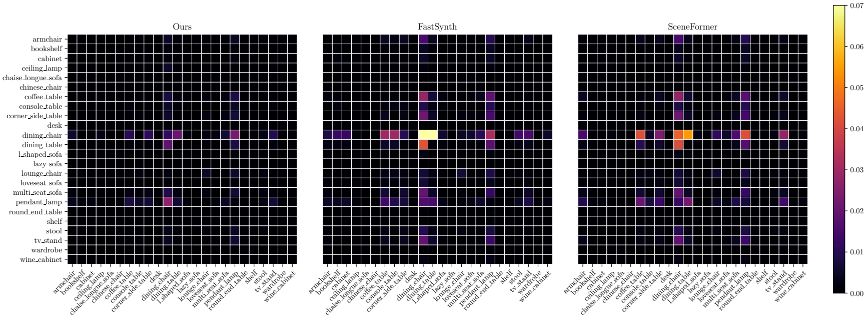


Figure B.25: **Absolute Difference between Object Co-occurrence in Dining Rooms.** We visualize the absolute difference of the probabilities of object co-occurrence computed between real and synthesized scenes using ATISS (left-most column), FastSynth (middle column), SceneFormer (right-most column). Lower is better.

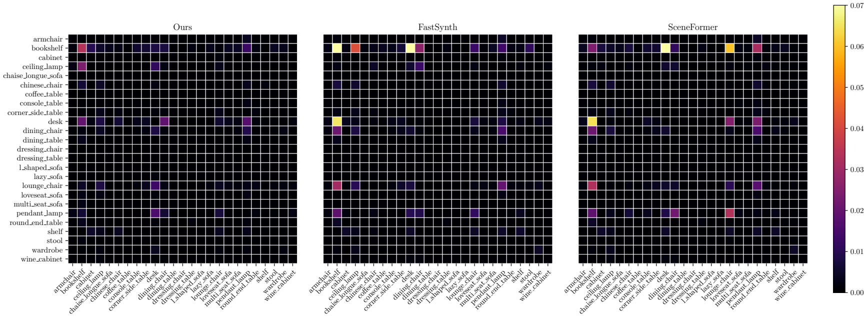
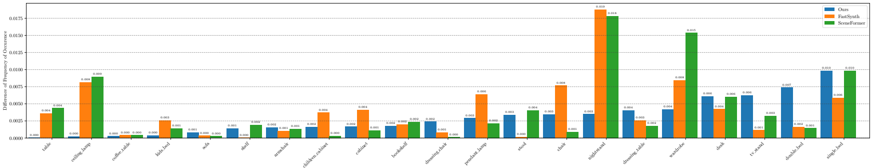
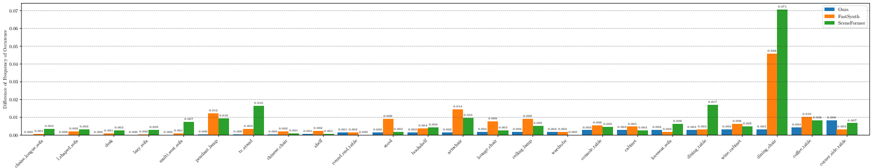


Figure B.26: **Absolute Difference between Object Co-occurrence in Libraries.** We visualize the absolute difference of the probabilities of object co-occurrence computed between real and synthesized scenes using ATISS (left-most column), FastSynth (middle column), SceneFormer (right-most column). Lower is better.

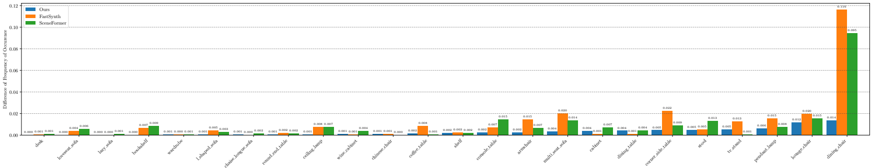
Even though the implementations are not directly comparable, since we cannot guarantee that all have been equally optimized, our findings meet our expectations. Namely, FastSynth [163] requires rendering the scene each time a new object is added, thus it is expected to be significantly slower than both SceneFormer and our model. On the other hand, SceneFormer [202] utilizes four different transformer models for generating the attributes of each object, hence it is expected to be at least four times slower than our model, when generating the same number of objects.



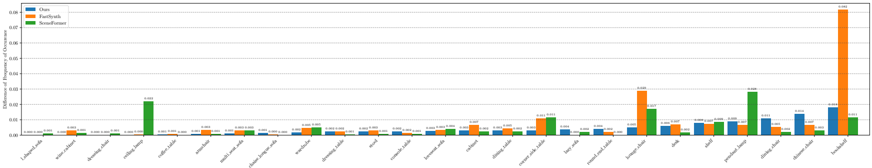
(a) Bedrooms



(b) Living Rooms



(c) Dining Rooms



(d) Libraries

Figure B.27: **Difference of Per-Object Frequencies.** We visualize the absolute difference between the per-object frequency of generated and real scenes using our method, FastSynth [163] and SceneFormer [202] for all room types. Lower is better.

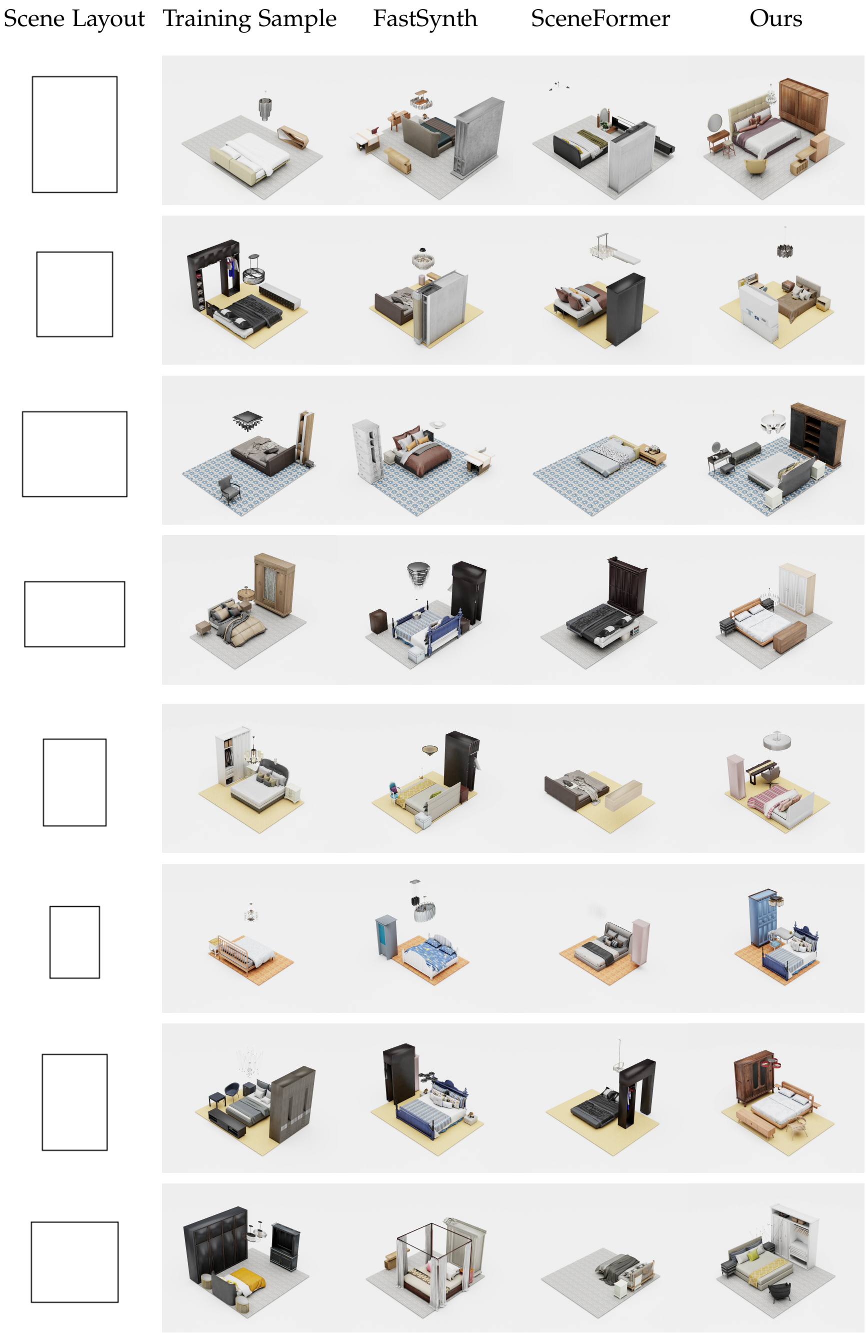


Figure B.28: **Qualitative Scene Synthesis Results on Bedrooms**.

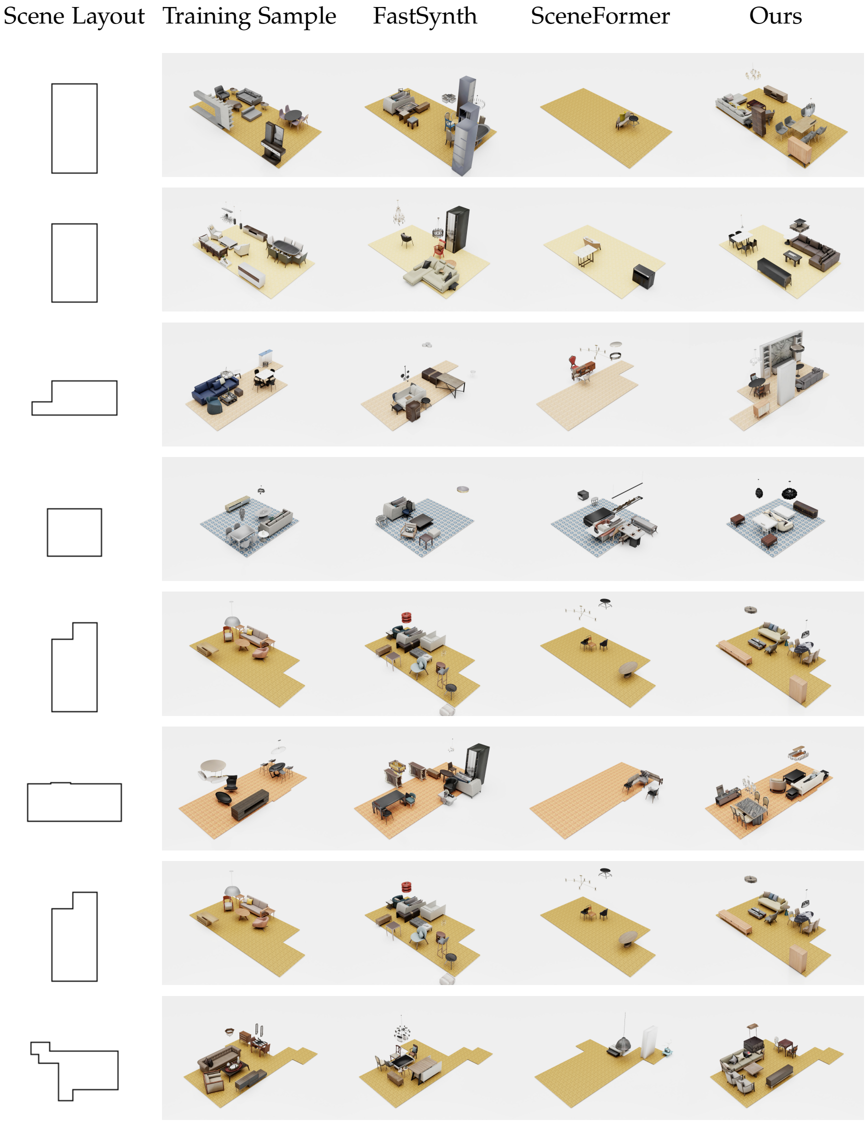


Figure B.29: **Qualitative Scene Synthesis Results on Living Rooms**.

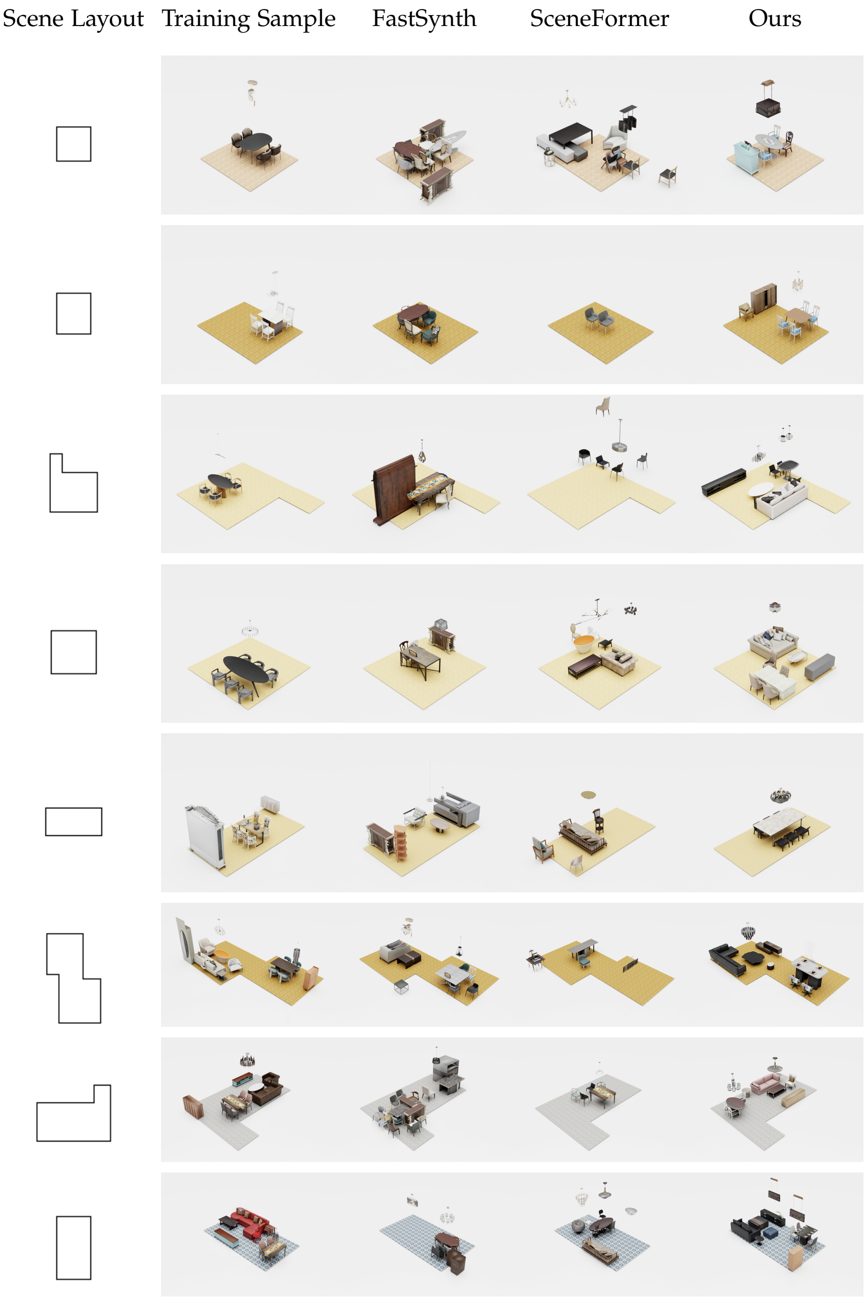


Figure B.30: **Qualitative Scene Synthesis Results on Dining Rooms**.

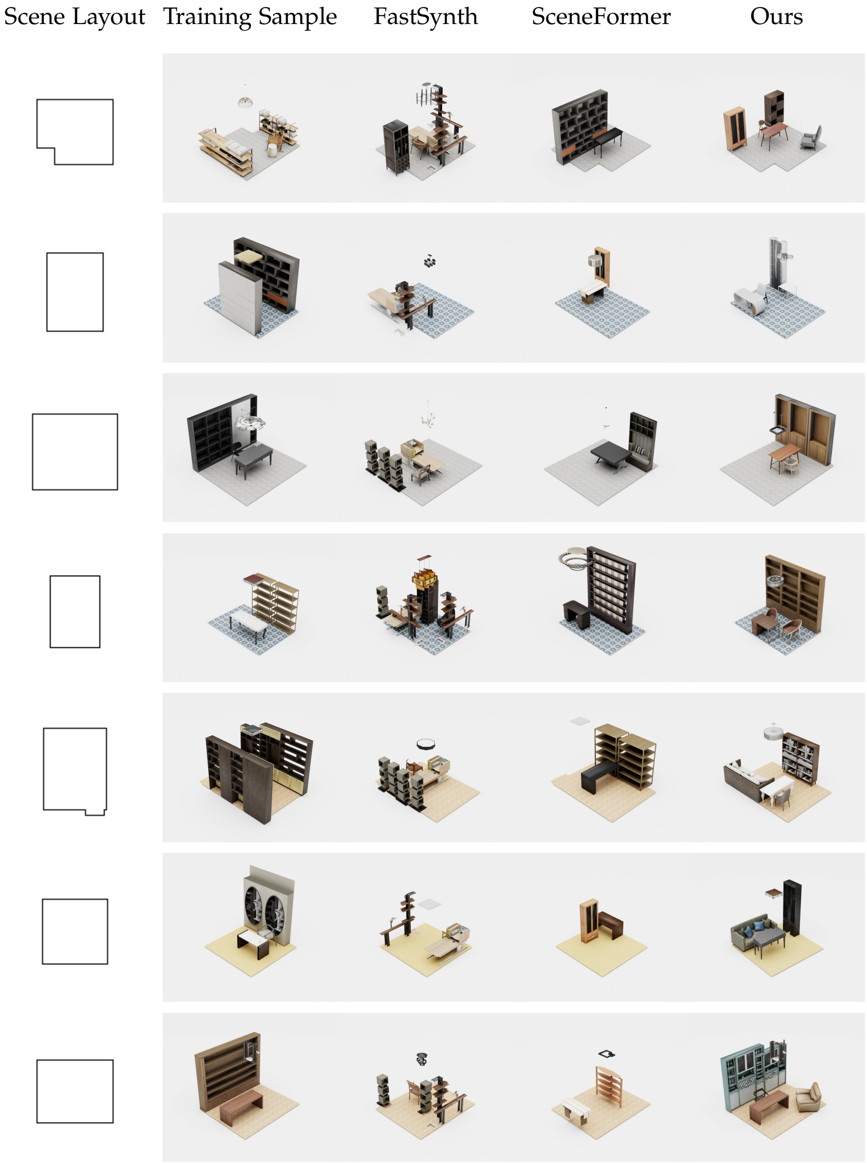


Figure B.31: **Qualitative Scene Synthesis Results on Libraries.**

BIBLIOGRAPHY

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang. Tensorflow: A system for large-scale machine learning. *arXiv.org*, 1605.08695, 2016.

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *Proc. of the International Conf. on Machine learning (ICML)*, 2018.

[3] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets deep learning for car instance segmentation in urban scenes. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2017.

[4] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[5] Iro Armeni, Zhi-Yang He, Amir Roshan Zamir, JunYoung Gwak, Jitendra Malik, Martin Fischer, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[6] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[7] Matan Atzmon and Yaron Lipman. SAL: sign agnostic learning of shapes from raw data. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2562–2571, 2020.

[8] Renée Baillargeon. Infants' physical world. *Current directions in psychological science*, 13(3):89–94, 2004.

[9] Alan H Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications (CGA)*, 1981.

[10] Irving Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 1986.

[11] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94(2):115, 1987.

[12] I Binford. Visual perception by computer. In *IEEE Conference of Systems and Control*, 1971.

[13] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Dynamic FAUST: registering human bodies in motion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] André Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv.org*, 1608.04236, 2016.

[15] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[16] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[17] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2012.

[18] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[19] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv.org*, 1512.03012, 2015.

[20] Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas A. Funkhouser. Attribit: content creation with semantic attributes. In *The 26th Annual ACM Symposium on User Interface Software and Technology, UIST'13, St. Andrews, United Kingdom, October 8-11, 2013*, 2013.

[21] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 40(4):834–848, 2018.

[22] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020.

[23] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv.org*, 2021.

[24] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaako Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[25] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 42–51, 2020.

[26] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. BAE-NET: branched autoencoder for shape co-segmentation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[27] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[28] Laurent Chevalier, Fabrice Jaillet, and Atilla Baskurt. Segmentation and superquadric modeling of 3d objects. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2003.

[29] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *arXiv.org*, volume 2003.01456, 2020.

[30] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[31] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv.org*, 2019.

[32] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[33] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[34] Christopher Bongsoo Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[35] A. H. J. Christensen. A note on geodesic polyhedra: Triangulation and contouring of spheres. *Comput. Graph.*, 3(4):163–165, 1978.

[36] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnets: Learnable convex decomposition. *arXiv.org*, 2019.

[37] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnets: Learnable convex decomposition. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[38] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[39] Zhantao Deng, Jan Bednařík, Mathieu Salzmann, and Pascal Fua. Better patch stitching for parametric surface reconstruction. 2020.

[40] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[41] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[42] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[43] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv.org*, 2020.

[44] Xinhan Di, Pengqian Yu, Hong Zhu, Lei Cai, Qiuyan Sheng, and Changyu Sun. Structural plan of indoor scenes with personalized preferences. *arXiv.org*, 2020.

[45] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

[46] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017.

[47] Simon Donne and Andreas Geiger. Learning non-volumetric depth fusion using successive reprojections. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021.

[49] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017.

[50] Anastasia Dubrovina, Fei Xia, Panos Achlioptas, Mira Shalah, Raphaël Groscot, and Leonidas J. Guibas. Composite shape modeling via latent space factorization. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 8139–8148. IEEE, 2019.

[51] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Joshua B. Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[52] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. 2021.

[53] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[54] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Cao Li, Zengqi Xun, Chengyue Sun, Yiyun Fei, Yu Zheng, Ying Li, Yi Liu, Peng Liu, Lin Ma, Le Weng, Xiaohang Hu, Xin Ma, Qian Qian, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3d-front: 3d furnished rooms with layouts and semantics. *arXiv.org*, abs/2011.09127, 2020.

[55] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve J. Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *arXiv.org*, abs/2009.09633, 2020.

[56] Matheus Gadelha, Giorgio Gori, Duygu Ceylan, Radomír Mech, Nathan Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. Learning generative models of shape handles. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[57] Matheus Gadelha, Subhransu Maji, and Rui Wang. 3d shape induction from 2d views of multiple objects. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017.

[58] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. In *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[59] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao Zhang. SDM-NET: deep generative network for structured deformable mesh. In *ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia (SIGGRAPH Asia)*, 2019.

[60] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas A. Funkhouser. Local deep implicit functions for 3d shape. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[61] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[62] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[63] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[64] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[65] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[66] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[67] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020.

[68] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. AtlasNet: A papier-mâché approach to learning 3d surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[69] Kunal Gupta and Manmohan Chandraker. Neural mesh flow: 3d manifold mesh generationvia diffeomorphic flows. *arXiv.org*, 2020.

[70] William Rowan Hamilton. Xi. on quaternions; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 33(219):58–60, 1848.

[71] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. *arXiv.org*, 1704.00710, 2017.

[72] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge J. Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[73] Wilfried Hartmann, Silvano Galliani, Michal Havlena, Luc Van Gool, and Konrad Schindler. Learned multi-patch similarity. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[74] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[76] Philipp Henzler, Jeremy Reizenstein, Patrick Labatut, Roman Shapovalov, Tobias Ritschel, Andrea Vedaldi, and David Novotný. Unsupervised learning of 3d object categories from videos in the wild. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[77] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[78] Donald D Hoffman and Whitman A Richards. Parts of recognition. *Cognition*, 18(1-3):65–96, 1984.

[79] Fangqiao Hu, Jin Zhao, Yong Hunag, and Hui Li. Learning structural graph layouts and 3d shapes for long span bridges 3d reconstruction. *arXiv.org*, abs/1907.03387, 2019.

[80] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[81] Ales Jaklic, Ales Leonardis, and Franc Solina. *Segmentation and Recovery of Superquadrics*, volume 20 of *Computational Imaging and Vision*. Springer, 2000.

[82] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: an end-to-end 3d neural network for multiview stereopsis. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[83] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[84] Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. GAL: geometric adversarial loss for single-view 3d-object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[85] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. Layoutvae: Stochastic scene layout generation from a label set. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[86] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[87] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[88] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[89] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[90] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. 2020.

[91] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020.

[92] Yuki Kawana, Yusuke Mukuta, and Tatsuya Harada. Neural star domain as primitive representation. *arXiv.org*, 2020.

[93] Yuki Kawana, Yusuke Mukuta, and Tatsuya Harada. Neural star domain as primitive representation. In *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[94] Mohammad Keshavarzi, Aakash Parikh, Xiyu Zhai, Melody Mao, Luisa Caldas, and Allen Y. Yang. Scenegen: Generative contextual scene augmentation using scene graph priors. *arXiv.org*, 2020.

[95] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. Drivegan: Towards a controllable high-quality neural simulation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[96] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015.

[97] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *Proc. of the International Conf. on Learning Representations (ICLR)*, 2014.

[98] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[99] Katherine D Kinzler and Elizabeth S Spelke. Core systems in human cognition. *Progress in brain research*, 164:257–264, 2007.

[100] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *Proc. of the International Conf. on Machine learning (ICML)*, 2018.

[101] Adam R. Kosiorek, Hyunjik Kim, and Danilo J. Rezende. Conditional set generation with transformers. *arXiv.org*, 2020.

[102] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[103] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proc. of the International Conf. on Machine learning (ICML)*, pages 1945–1954. PMLR, 2017.

[104] David H Laidlaw, W Benjamin Trumbore, and John F Hughes. Constructive solid geometry for polyhedral objects. In *ACM Trans. on Graphics*, 1986.

[105] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2019.

[106] Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao (Richard) Zhang, and Leonidas J. Guibas. GRASS: generative recursive autoencoders for shape structures. *ACM Trans. on Graphics*, 36(4), 2017.

[107] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[108] Lingyun Luke Li, Bin Yang, Ming Liang, Wenyuan Zeng, Mengye Ren, Sean Segal, and Raquel Urtasun. End-to-end contextual perception and prediction with interaction transformer. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 5784–5791, 2020.

[109] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao (Richard) Zhang. GRAINS: generative recursive autoencoders for indoor scenes. *ACM Trans. on Graphics*, 2019.

[110] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2018.

[111] Yichen Li, Kaichun Mo, Lin Shao, Minhyuk Sung, and Leonidas J. Guibas. Learning 3d part assembly from a single image. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[112] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Sai Bi, Zexiang Xu, Hong-Xing Yu, Kalyan Sunkavalli, Milos Hasan, Ravi Ramamoorthi, and Manmohan Chandraker. Openrooms: An end-to-end open framework for photorealistic indoor scene datasets. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[113] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[114] Yunchao Liu, Zheng Wu, Daniel Ritchie, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Learning to describe scenes with programs. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[115] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[116] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. on Graphics*, 2015.

[117] Andrew Luo, Zhoutong Zhang, Jiajun Wu, and Joshua B. Tenenbaum. End-to-end optimization of scene layout. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[118] Qianli Ma, Shunsuke Saito, Jinlong Yang, Siyu Tang, and Michael J. Black. SCALE: Modeling clothed humans with a surface codec of articulated local elements. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[119] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[120] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function

space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[121] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[122] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[123] Niloy J. Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, Vladimir G. Kim, and Qi-Xing Huang. Structure-aware shape processing. In *ACM Trans. on Graphics*, pages 13:1–13:21, 2014.

[124] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. In *ACM Trans. on Graphics*, 2019.

[125] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. *ACM Trans. on Graphics*, 2006.

[126] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *Proc. of the International Conf. on Machine learning (ICML)*, 2020.

[127] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[128] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[129] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[130] Chengjie Niu, Jun Li, and Kai Xu. Im2struct: Recovering 3d shape structure from a single RGB image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[131] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[132] Michael Oechsle, Michael Niemeyer, Christian Reiser, Lars Mescheder, Thilo Strauss, and Andreas Geiger. Learning implicit surface light fields. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2020.

[133] Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, 2018.

[134] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single RGB images via topology modification networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[135] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *ACM Trans. on Graphics*, 2001.

[136] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[137] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On buggy resizing libraries and surprising subtleties in FID calculation. *arXiv.org*, 2021.

[138] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *Proc. of the International Conf. on Machine learning (ICML)*, 2018.

[139] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[140] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[141] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[142] Despoina Paschalidou, Ali Osman Ulusoy, Carolin Schmitt, Luc van Gool, and Andreas Geiger. Raynet: Learning volumetric 3d reconstruction with ray potentials. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[143] Despoina Paschalidou, Luc van Gool, and Andreas Geiger. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[144] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv.org*, 1606.02147, 2016.

[145] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[146] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[147] Alex Pentland. Parts: Structured descriptions of shape. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 1986.

[148] Maurizio Pilu and Robert B. Fisher. Equal-distance sampling of supercllipse models. In *Proc. of the British Machine Vision Conf. (BMVC)*, 1995.

[149] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2019.

[150] Pulak Purkait, Christopher Zach, and Ian Reid. SG-VAE: scene grammar variational autoencoder to generate new indoor scenes. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[151] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[152] G.J. Qi, X.S. Hua, Y. Rui, T. Mei, J. Tang, and H.J. Zhang. Concurrent multiple instance learning for image categorization. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[153] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. Human-centric indoor scene synthesis using stochastic grammar. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[154] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *arXiv.org*, 2019.

[155] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *Proc. of the International Conf. on Machine learning (ICML)*, 2021.

[156] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[157] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotný. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021.

[158] Edoardo Remelli, Artem Lukoianov, Stephan R. Richter, Benoît Guillard, Timur M. Bagautdinov, Pierre Baqué, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. 2020.

[159] Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[160] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[161] Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. OctNetFusion: Learning depth fusion from data. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017.

[162] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[163] Daniel Ritchie, Kai Wang, and Yu-An Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[164] Lawrence G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[165] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: modeling and capturing hands and bodies together. *ACM Transactions on Graphics*, 36(6):245:1–245:17, 2017.

[166] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[167] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[168] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[169] Shunsuke Saito, Tomas Simon, Jason M. Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[170] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017.

[171] Manolis Savva, Jitendra Malik, Devi Parikh, Dhruv Batra, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, and Vladlen Koltun. Habitat: A platform for embodied AI research. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[172] Gilad Sharir, Asaf Noy, and Lihi Zelnik-Manor. An image is worth 16x16 words, what is a video worth? In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021.

[173] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[174] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Mech. Parsenet: A parametric surface fitting network for 3d point clouds. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[175] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, 2018.

[176] Maria Shugrina, Ziheng Liang, Amlan Kar, Jiaman Li, Angad Singh, Karan Singh, and Sanja Fidler. Creative flow+ dataset. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[177] Franc Solina. Volumetric models in computer vision-an overview. *Journal of Computing and Information technology*, 1994.

[178] Franc Solina and Ruzena Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 1990.

[179] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[180] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[181] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. *arXiv preprint arXiv:2101.10994*, 2021.

[182] Jerry O. Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Mech, and Vladlen Koltun. Metropolis procedural modeling. *ACM Trans. on Graphics*, 2011.

[183] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

[184] Josh Tenenbaum. Building machines that learn and think like people. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018.

[185] Demetri Terzopoulos and Dimitris N. Metaxas. Dynamic 3d models with local and global deformations: deformable superquadrics. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 1990.

[186] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[187] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Learning to infer and execute 3d shape programs. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[188] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv.org*, 2020.

[189] Shubham Tulsiani and Abhinav Gupta. Pixeltransformer: Sample conditioned signal generation. In *Proc. of the International Conf. on Machine learning (ICML)*, 2021.

[190] Shubham Tulsiani, Nilesh Kulkarni, and Abhinav Gupta. Implicit mesh reconstruction from unannotated image collections. *arXiv.org*, 2020.

[191] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[192] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, 2016.

[193] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018.

[194] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[195] Narunas Vaskevicius and Andreas Birk. Revisiting superquadric fitting: A numerically stable formulation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

[196] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[197] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Planit: planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Trans. on Graphics*, 2019.

[198] Kai Wang, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Trans. on Graphics*, 37(4):70:1–70:14, 2018.

[199] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[200] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[201] WeiyueXu Wang, Qiangeng Xu, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[202] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. *arXiv.org*, 2020.

[203] Yanzhen Wang, Kai Xu, Jun Li, Hao Zhang, Ariel Shamir, Ligang Liu, Zhi-Quan Cheng, and Yueshan Xiong. Symmetry hierarchy of man-made objects. In *EUROGRAPHICS*, 2011.

[204] Philippe Weinzaepfel, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. Visual localization by learning objects-of-interest dense match regression. In *CVPR*, 2019.

[205] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[206] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv.org*, abs/1810.08705, 2018.

[207] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[208] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[209] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomír Mech, and Ulrich Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[210] Zhenjia Xu, Zhijian Liu, Chen Sun, Kevin Murphy, William T Freeman, Joshua B Tenenbaum, and Jiajun Wu. Unsupervised discovery of parts, structure, and dynamics. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019.

[211] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[212] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018.

[213] Lior Yariv, Matan Atzmon, and Yaron Lipman. Universal differentiable renderer for implicit neural representations. *arXiv.org*, 2003.09852, 2020.

[214] Linwei Ye, Mrigank Rochan, Zhi Liu, and Yang Wang. Cross-modal self-attention network for referring image segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[215] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[216] Lap-Fai Yu, Sai Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: automatic optimization of furniture arrangement. *ACM Trans. on Graphics*, 2011.

[217] Lap-Fai Yu, Sai Kit Yeung, and Demetri Terzopoulos. The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE Trans. Vis. Comput. Graph.*, 2016.

[218] Song-Hai Zhang, Shaokui Zhang, Wei-Yu Xie, Cheng-Yang Luo, and Hong-Bo Fu. Fast 3d indoor scene synthesis with discrete and exact layout pattern extraction. *arXiv.org*, 2020.

[219] Xiuming Zhang, Sean Ryan Fanello, Yun-Ta Tsai, Tiancheng Sun, Tianfan Xue, Rohit Pandey, Sergio Orts-Escolano, Philip L. Davidson, Christoph Rhemann, Paul E. Debevec, Jonathan T. Barron, Ravi Ramamoorthi, and William T. Freeman. Neural light transport for relighting and view synthesis. *arXiv.org*, 2008.03806, 2020.

[220] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas A. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[221] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *ACM Trans. on Graphics*, 2020.

[222] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[223] Yang Zhou, Zachary While, and Evangelos Kalogerakis. Scenegraphnet: Neural message passing for 3d indoor scene augmentation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[224] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021.

[225] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[226] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017.

CURRICULUM VITAE

PERSONAL DATA

Name	Despoina Paschalidou
Date of Birth	June 8, 1991
Place of Birth	Athens, Greece
Citizen of	Greece

EDUCATION

2017 – 2021	Autonomous Vision Group and Computer Vision Lab ETH Zürich and Max Planck Institute for Intelligent Systems Doctoral Studies
2009 – 2015	Aristotle University of Thessaloniki Diploma (M.Eng) in Electrical and Computer Engineering
2006– 2009	1st Lyceum of Mikra High-School

PROFESSIONAL EXPERIENCE

2021	Research Intern Facebook AI Research, London, United Kingdom
2020–2021	Research Intern NVIDIA AI Research, Toronto Canada
2019–2020	Research Assistant ETH Zürich
2017–2019	Research Assistant Max Planck Institute for Intelligent Systems, Tübingen Germany