3D Part Scanning, Inspection, and Representation

Tianshuang Qiu

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-96
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-96.html

May 16, 2025

Acknowledgement

I am deeply grateful to Professor Ken Goldberg for welcoming me into AUTOLAB and guiding me through projects in grasping, planning, and vision.

I thank Zehan Ma and Karim El-Refai for their work on OmniScan. I'm grateful to all my collaborators and mentors across my other work. Thank you for your brilliance and friendship. Your support helped me grow through challenging and unfamiliar projects.

I'm thankful to my family: my mother, father, grandma, grandpa, aunt, and uncle, for their endless love and encouragement. A special thanks to Mira Bhatt for her company and support through my various projects.

3D Part Scanning, Inspection, and Representation

by

Tianshuang (Ethan) Qiu

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Angjoo Kanazawa

Spring 2025

3D Part Scanning, Inspection, and Representation

by Tianshuang (Ethan) Qiu

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Ken Goldberg
Research Advisor

12 MAY 2025

(Date)

* * * * * * *

Professor Angjoo Kanazawa
Second Reader

12 May 2025

(Date)

3D Part Scanning, Inspection, and Representation

Abstract

3D Part Scanning, Inspection, and Representation

by

Tianshuang (Ethan) Qiu

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Ken Goldberg, Chair

High-quality 3D part inspection, scanning, and representation are critical for applications across manufacturing, robotics, and virtual environments. This work presents two complementary approaches to advance these capabilities. First, we introduce a novel bimanual robotic system that generates high-quality 3D Gaussian Splat models of physical objects using a single stationary camera, achieving full 360° scanning through handover regrasping to reveal occluded surfaces. This system demonstrates the potential for effective defect detection across diverse objects. Second, we systematically evaluate deep learning pipelines for predicting additive manufacturing part quality using various 3D shape representations including voxels, depth images, distance fields, and point clouds. Our research investigates the impact of dataset size, input resolution, and hyperparameter choices on model performance. Together, these approaches advance the state of automated, cost-effective 3D scanning and quality assessment critical for modern manufacturing and robotics applications.

In honor of the experiments that didn't quite work

Contents

List of Figures

List of Tables

Acknowledgments

Research has been a journey for me. I would not have come this far without support from advisors, friends, and family.

I express my sincere gratitude to Professor Ken Goldberg for welcoming me into his lab and into research in general. I joined AUTOLAB in late 2021, and it has been an incredible learning experience. Since then, I have worked on grasping, manipulation, planning, and vision projects. I have learned so much. I am excited by the many breakthroughs in these fields. I thank Professor Goldberg for offering me this amazing opportunity.

I also thank Professor Angjoo Kanazawa for being on my committee and being my second reader. I am grateful to Professor Sara McMains for her advice on the Additive Manufacturing project. I thank Professor Jeffrey Ichnowski for his guidance on motion planning.

I would also like to thank Chung Min Kim, Justin Kerr, Sara Shonkwiler, Letian (Max) Fu, and Kaiyuan (Eric) Chen for their mentorship. It was scary and challenging to take on projects in unfamiliar fields, and their guidance helped me find direction and confidence for my work.

There are many who have made a great impact on my work. I'm grateful to Zehan Ma and Karim El-Refai for their work on OmniScan, and Dr. Catie Cuan for her dedication to the Breathless Project. I also thank Sandeep Mukherjee and Abby O'Neil for exploring computer vision and large models with me, and Zach Tam and Karthik Dharmarajan for exploring simulations together. I'm thankful to Shreya Ganti, Kush Hari, Justin Yu, and Alonso Martinez for being brilliant collaborators. It was a great honor and joy to work together, and it was awesome to be friends with you all.

Lastly, I'm thankful to my family: my mother, father, grandma, grandpa, aunt, and uncle, for their endless love and encouragement. A special thanks to Mira Bhatt for her company and support through my various projects.

Chapter 1

Introduction

The ability to inspect, scan, and represent the three-dimensional (3D) geometry of physical objects plays a vital role in a wide range of domains, including manufacturing and robotics. With the recent demand for automation and production, robust and scalable methods for capturing and analyzing 3D geometry has grown. Accurate 3D models are essential to many downstream tasks such as digital twin creation, virtual simulation, part inspection, and robotic manipulation. However, current methods cannot achieve true omni-directional scan because the object is never lifted so the bottom could not be scanned. In parallel, additive manufacturing (AM) is transforming how parts are designed and produced, enabling rapid prototyping and on-demand production of complex geometries. AM is also seeing growing adoption for safety-critical end-use components in industrial setups. Yet, quality control remains a major challenge. The ability to automatically predict print quality or identify defects in manufactured parts is essential for ensuring the reliability of AM processes. Recent advances in deep learning, offer the potential for fast, scalable, and non-destructive quality prediction by learning from large datasets of previously manufactured parts.

This thesis explores two complementary approaches for advancing 3D part inspection and quality analysis. In the first part, we develop a robotic system: Omni-Scan, using a bimanual robot and a single fixed camera to generate high-fidelity 3D models of physical objects. The robot rotates the object to and performs in-hand object re-grasping to expose all surfaces. We apply segmentation and depth models to remove the grippers and background, and construct 3D Gaussian Splat models from the resulting multi-view image sequences. We demonstrate that Omni-Scan can be used to inspect a wide variety of industrial and household objects, achieving high accuracy in identifying geometric or visual defects.

In the second part, we evaluate deep learning-based pipelines for predicting additive manufacturing part quality using different 3D shape representations. We systematically compare voxel grids, depth images, distance fields, and point clouds as input modalities, paired with convolutional and transformer-based architectures. We train these models on real-world datasets and analyze how their performance is affected by dataset size and input resolution. Our results highlight that increasing dataset size has a stronger effect on performance than increasing resolution, and that distance field representations paired with 3D CNNs yield the most accurate predictions.

Together, these two contributions address key challenges in automated 3D part analysis from

different angles: one through novel physical data acquisition, and the other through effective data-driven learning methods. This thesis demonstrates how combining accessible hardware, modern vision models, and deep learning techniques can advance the scalability and accessibility of 3D part inspection and quality assessment systems.

Chapter 2

Related Work

2.1 3D Scanning and Part Inspection

3D Reconstruction with Radiance Fields

Neural Radiance Fields [58] are an attractive representation for high quality 3D scene reconstruction from posed RGB images, with a flurry of recent work enhancing quality [2, 8, 7, 55], large-scale scenes [79, 86, 9], optimization speed [60, 13, 33, 34], dynamic scenes [64, 51, 66], and more. Because of its high-quality reconstruction and differentiable properties, NeRF has been explored in robotics for navigation and mapping [2, 98, 75, 69], manipulation [50, 27, 43, 39, 67, 44, 74], and for synthetic data generation [12]. 3D Gaussian Splatting [42] made a major breakthrough in speed and quality of radiance fields, and the field has quickly adopted it for similar applications.

Color-NeuS [97] presents a method for reconstructing object meshes with color from multi-view images or monocular video, using a relighting network to separate view-dependent color from neural volume rendering and extracting surfaces from an SDF network, achieving strong results on in-hand object scanning and public benchmarks like DTU, BlendedMVS, and OmniObject3D. [26] Google Scanned Objects is an open-source dataset of 3D-scanned household items, preprocessed for use in simulation platforms like Ignition Gazebo and Bullet, designed to enhance interactive simulation, synthetic perception, and robotic learning. CHORD, a method for category-level hand-held object reconstruction that deforms a categorical shape prior to recover object shapes, incorporating appearance, shape, and interaction pose awareness, and evaluates performance on the newly introduced COMIC dataset.

In this work we use 3DGS to reconstruct high-quality object models, and in contrast to prior work reconstruct entire objects in high detail with a static camera, via a method of merging multiple scans and accurately masking the object of interest.

Creating 3D Object Scan Data

Conventionally, datasets of 3D objects are constructed with expensive equipment like multiview camera arrays or high precision depth sensors, such as in the Google Scanned Objects [26] or DTU [1] datasets. Other large datasets like Objaverse [21] exist, but are comprised of synthetic objects. In this work, we leverage recent work on multi-view reconstruction from RGB images to alleviate the need for expensive sensors and autonomously digitize real objects with a robot.

Several works explore reconstructing objects in human hands, including Color-NeuS [97], which reconstructs object SDFs by separating view-dependent effects with a relighting network. BundleSDF [88] achieves near real-time tracking and reconstruction from monocular RGB-D video through pose graph optimization. HandTrackNet [14], tracks hand joints and reconstructs poses from depth sequences, and HandNeRF [17], which leverages hand shape constraints to reconstruct scenes from single RGB images. In this work we strive for very high visual quality by utilizing 3DGS to reconstruct fine details like text, and in addition present an automated pipeline using a robot for wholistic scanning without a human.

In pose estimation, FoundationPose [89] employs neural implicit representations with transformer architectures for generalizable 6D object pose estimation. Supporting these advances, Google Scanned Objects [26] provides 3D-scanned household items optimized for simulation platforms.

Automated Part Inspection

Automated part inspection using robotics has advanced significantly with vision systems, machine learning, and sensor integration. Prior work has studied photogrammetry-based 3D reconstruction for inspection where the robot moves a camera around the object on a tabletop [46]. Davtalab et al. (2022) developed a deep learning approach for real-time defect detection in additive manufacturing, improving quality control [19]. Agarwal et al. (2023) proposed a two-stage pipeline combining vision and tactile sensing for aerospace component inspection, achieving high defect identification rates [4]. In this work, we focus on small parts like plug adaptors or cameras, and leverage a dual handover grasp to automatically inspect small parts holistically without occlusions from a tabletop or gripper.

2.2 Deep Learning on 3D Representations

Deep learning (DL) models are powerful tools for a wide range of applications. For example, DL models enable precise image segmentation in medical diagnostics [87], optimize processes in manufacturing [49], and enhance decision-making in autonomous driving [16]. The success of DL models in these areas can be largely attributed to training DL models on enough high quality data, as well as the choice of appropriate input shape representations and corresponding DL architectures [61].

Learning Based Classification Tasks

For deep learning based image classification, input images are represented as pixels using two-dimensional arrays. Convolutional neural networks (CNNs) and DL architectures that build on CNNs (e.g. ResNet50 [37]) are well-suited to finding patterns in images. Image classification models are trained on large, high-quality, labeled image datasets, such as MNIST [23], CIFAR-10 and CIFAR-100 [48], and ImageNet [22]. Deep learning has revolutionized image classification, but only because of careful decision making about training data and DL pipeline architectures.

The application of DL models in predicting the quality and performance under load of three-dimensional mechanical parts is an emerging and promising area of research [41, 92, 25, 90, 45, 6, 52, 85, 62, 29, 57]. Deep learning models have the potential to efficiently learn patterns in part quality and performance data by exploiting similarities between analogous shapes where two additively manufactured parts have similar finite element analysis (FEA) manufacturing process distortion predictions. If successful, DL models will drastically reduce the computational and time requirements to run FEA for part quality and performance data. Speeding up FEA will enable engineers to iterate designs more quickly and discover better engineering designs. Reducing the computational requirements of FEA will make it possible for engineers and researchers without access to extensive GPU compute resources to generate FEA results. Early DL part quality and performance research has generated good results; however, current research is limited to training models on generally small to medium datasets of relatively simple, artificial or geometrically limited parts.

Learning-Based Part Prediction

Existing DL part quality and performance research can be characterized by choices made for various aspects such as: (a) input dataset geometries, (b) dataset size, (c) input shape representation, (d) DL architecture, and (e) prediction type.

Input dataset geometries

Nie et al. [62] and Wang et al. [85] simplify the input to two-dimensional geometries. Jin et al. [41], Nie et al., Wang et al., Dong et al. [25], Khadilkar et al. [45], Eranpurwala et al. [29], and Williams et al. [90] synthetically generate simple CAD parts (see example parts in Fig. 2.1) for the DL models to train on. Liang et al. [52] and Balu et al. [6] generate synthetic biological parts by parameterizing from an original real biological part thus keeping the input geometric variability quite limited.

Dataset size:

The amount of data generated for these research projects varies widely. Wang et al. [85] trains their model on 128 parts. Jin et al. uses 200 parts [41]. Wong et al. [92] uses 477 parts. Dong et al. [25] uses 420 parts rotated into a second orientation for 840 total parts. Liang et al. [52] uses 729 parts. Nie et al. [62], which makes 2D FEA predictions, has 120,960 common cantilevered

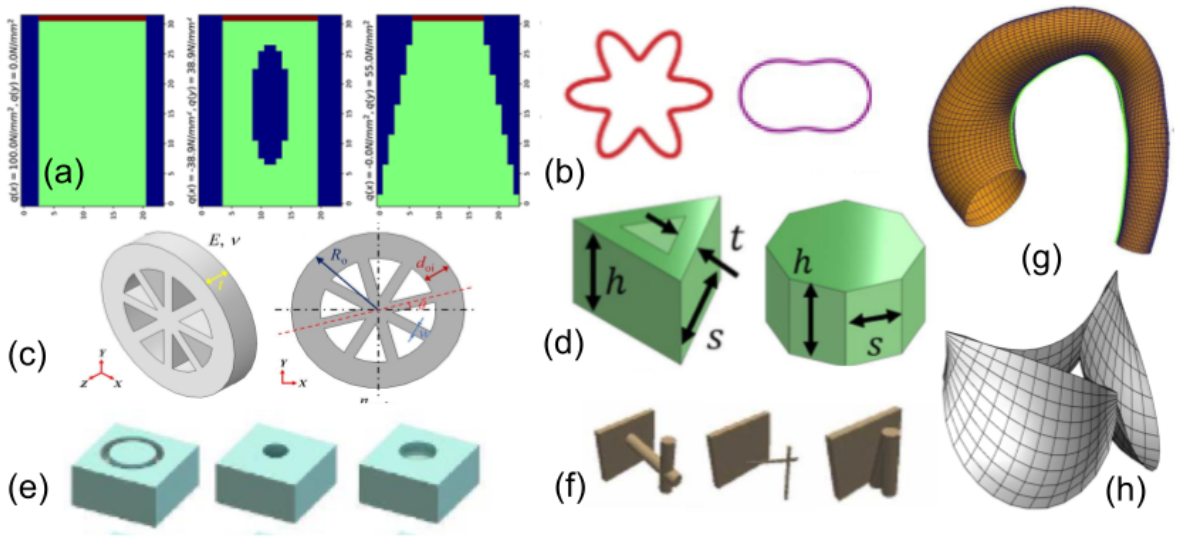


Figure 2.1: Input part geometries from related data-driven part quality and performance prediction research [25, 62, 29, 6, 41, 52, 85, 90]. All parts are synthetically generated for the specific purpose of doing research except in the case of the two biomedical part datasets where parts are created by artificially parameterizing biological parts. Our research uses parts from CAD repositories that more accurately reflect the true variety of mechancial parts than synthetically generated geometries. (a) & (b): two-dimensional geometry inputs [85, 62]. (c) - (f): "simple" synthetic geometries [41, 90, 25, 29]. (g) & (h): geometries based on parameterizing biological part [6, 52].

geometries. Williams et al. [90], which predicts part mass, support material mass, and build time, uses 72,000 parts. Khadilkar et al. [45] uses 16,700 parts. Balu et al. [6] uses 90,941 heart valve parts.

Balu et al. uses a dataset of over 90,000 parts with a constricted range of geometries, all heart valves, while Liang et al. with a seemingly similar level of geometric diversity, all aortas, uses a dataset of 729 parts. Realistically, deep learning models (and even machine learning models) need a great deal of data to learn, but using more data than necessary wastes memory and computation time. In this research, we experimentally evaluate how much data is needed for the AM part quality and functional performance DL domain.

Input shape representation & DL architecture:

For the research projects that investigate three-dimensional geometries (research that simplifies the input to 2D geometries is omitted because it may not generalize well to 3D) shape representation/DL architectures used include voxel representation and 3D CNN models [90, 25, 29], point cloud and CNN-PointNet hybrid model [45], NURBS surface reformatted and convolutional autoencoder algorithm [6], and mesh representation paired encoder-decoder algorithm and/or graph neural network (GNN) [41, 92, 52]. There is a lack of clear guidance as to which shape representa-

Table 2.1: Deep learning part quality and performance research

Paper	Input geometries	# parts	Shape representation	Prediction
Nie et al. [62]	2D cantilevered geometries	120,960	pixel geometry, loading + boundary conditions	stress field
Wang et al. [85]	simple 2D geom., chess pieces	128	2D shape context descriptor	SLA separation stress
Dong et al. [25]	circular/square struts, walls	840	voxels	residual stress
Khadilkar et al. [45]	simple "artificial" geometries	16,700	point cloud + bottom layer image	2D FEA
Balu et al. [6]	"artificial" heart valves	90,941	reformatted NURBS surface	deformation
Williams et al. [90]	simple "artificial" geometries	72,000	voxels	part/support mass, time
Liang et al. [52]	"synthetic" aortas	729	quadrilateral mesh	stress
Jin et al. [41]	parameterized wheels	200	mesh	3D stress
Eranpurwala et al. [29]	simple "artificial" geometries	54,000	voxels	best orientation
Wong et al. [92]	circular/square struts, walls	1,505	mesh	pressure, coeffs (residual stress)
Our Work	mechanical parts	482,214	voxels, point clouds, distance fields, depth images	AM printability scores

tion/DL architecture combinations are most effective for DL part quality and performance models, including FEA surrogate models.

Prediction type:

Current DL models predict part mass and build time [90], stress [41, 25, 52, 62], deformation [6], bottom-up stereolithography separation stress [85], and best additive manufacturing (AM) orientation [29, 57]. Previous related research is summarized in Table 2.1.

Systematic Comparison on Large Dataset

The primary objective of this research is to systematically compare promising shape representation/DL architecture combinations and data requirements for predicting the AM print quality of engineering parts. We investigate the impact of dataset size and resolution on prediction results for AM print quality, which will help inform researchers wanting to make their own datasets or create benchmark datasets especially for additive manufacturing part quality predictions. To see the impact of dataset size on performance, we have collected a large, high quality dataset. Similar to Williams et al. [90] and Eranpurwala et al. [29], the AM print quality labels in this research

are analytically generated and are not the main contribution of this research. We hypothesize that DL pipelines that work well on our AM print quality problem will transfer well to more computationally expensive FEA print quality and performance labels where the analytical approach has significant drawbacks, namely the time and computational resources required.

We evaluate the shape representation/DL architecture pairings based on the following criteria: (1) DL pipeline predictive performance over a range of dataset sizes, (2) time and computational resources required to train DL pipeline, and (3) DL pipeline performance's sensitivity to hyperparameter tuning.

Three-dimensional CAD data is memory intensive, and thus while it is important to have large datasets for training, it is also important not to store and train on more data than needed for good results. Another objective of this research is to advance understanding of the relationship between dataset size and DL performance in this domain. In this research, four shape representation/DL architecture pairings and a baseline model are compared as models to predict AM print quality. Each DL model is trained on data from our real world engineering dataset. The models are systematically tuned and compared. Our main contributions include:

- Comparison of four shape representation/DL architecture pairings to a baseline model for AM print quality prediction:
  - Distance field 3D CNN model,
  - Depth image (from top and bottom) 3D CNN model,
  - Voxel 3D CNN model,
  - Point cloud Transformer model,
  - Mid-point baseline model;
- Analysis of impact of dataset size on DL pipeline predictive performance; runtime, and hyperparameter sensitivity to give a full picture of each model's performance;
- A dataset containing more than 400k varied, real world, labeled CAD parts for model training and comparison.

Chapter 3

Omni-Scan

Creating Visually-Accurate Digital Twin Object Models Using a Bimanual Robot with Handover and Gaussian Splat Merging

3.1 Introduction

"Digital Twins"—visually-accurate 3D reconstructions of physical objects—are useful for many applications, such as automated inspection in manufacturing and Sim2Real learning. However, most 3D scanning methods rely on multiple fixed cameras. Recent advances in 3D reconstruction, such as Neural Radiance Fields (NeRF) [58] and 3D Gaussian Splatting [42], have enabled high-quality novel view synthesis and 3D reconstruction from 2D images. However, in robotic contexts, prior work has used moving wrist-mounted cameras, which significantly limits coverage of the object due to kinematic arm constraints and inability to scan sections of the object surface near the support surface.

We present Omni-Scan, a fully autonomous system for 3D object reconstruction through in-hand scanning with a bi-manual robot which only requires one stationary RGB camera and a stereo depth sensor. The system grasps objects and rotates them in front of the camera, capturing comprehensive views from multiple angles. We incorporate a bi-manual handover process that re-grasps the object, revealing surfaces previously occluded by the gripper to produce an omni-directional (360°) visual 3DGS model of the object. In-gripper scanning presents unique challenges due to object occlusions from the end effector, the need to merge multiple independent scans from different grasps, and the inversion of the typical assumption in neural reconstruction methods where a static scene is captured by a moving camera. To overcome this, we design a masking pipeline that segments the robot arm from the object and background using optical flow, DepthAnything V2 [94], Segment Anything [47] and Segment Anything 2 [68] for each view. We then alter the traditional 3D Gaussian Splatting training pipeline to accommodate this new scanning paradigm. We apply Omni-Scan to industrial part inspection, where it identifies both visual and geometric defects in household and industrial objects from a reference object.

This work makes the following contributions:

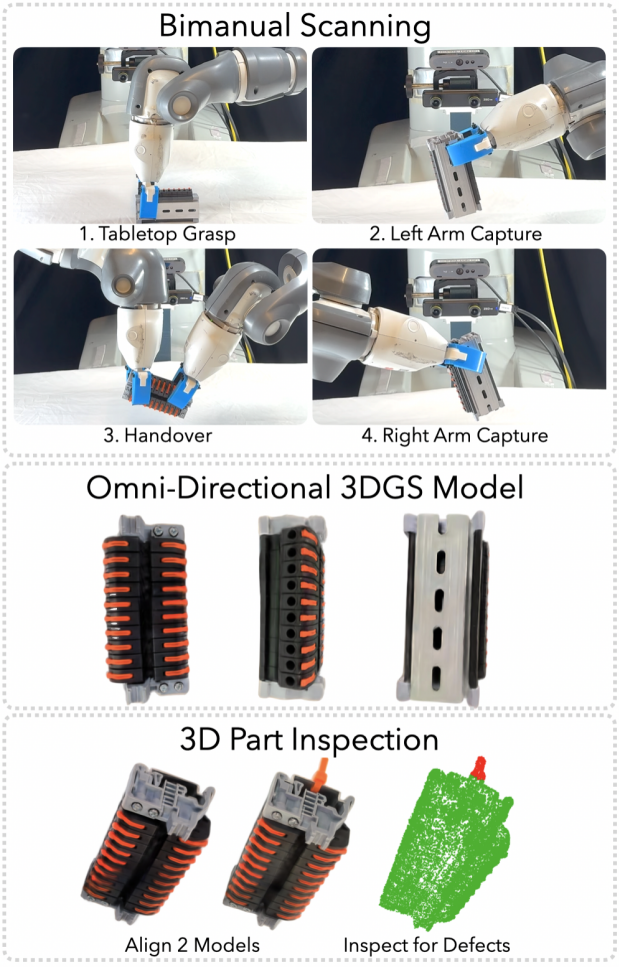


Figure 3.1: The robot grasps an object (wire connector) in any position and orientation for inspection. Omni-Scan then transfers the object between grippers to create a complete scan. The resulting full surface 3DGS model can be compared with a reference model for object inspection.

1. Omni-Scan: A pipeline for bi-manual robot object scanning that includes object grasping, multi-view scanning, handover for complete coverage, and 3D model generation

2. A robust masking and processing approach that accurately distinguishes the object, gripper, and background in captured images

3. A pose optimization and model merging technique that aligns and combines multiple 3DGSs into a cohesive 3D Gaussian Splat

4. A novel method for aligning 3DGS models for inspection

5. Experimental results evaluating the effectiveness of Omni-Scan for visualization and part inspection applications, achieving an accuracy of 83% for defect detection

Figure 3.2: **Reconstructed 3D Gaussian Splats of the 3DGS-Merged Model** We show **rendered views** from reconstructed splat models of objects collected by Omni-Scan. Each object is fully reconstructed without occlusion, even though the data was collected while grasped. In addition, the models capture fine geometric and visual details such as text or notches. See our website for interactive videos of full 3D surfaces.

3.2 Problem Statement

The goal is to create a visually-accurate omni-directional 3D model of a provided object, and then use this reconstruction to inspect for defects. We assume objects are rigid and cannot fit inside a 3cm diameter sphere but can fit inside a 10cm one, as well as the availability of a bi-manual robot with parallel jaw grippers, one fixed high-resolution monocular camera, and one stereo camera. During reconstruction, a target object is placed within the reachable workspace of the robot on a tabletop. We assume the robot is able to grasp and lift the object (i.e it is not too heavy). During defect inspection, a robot is provided with 3DGS models of two reference objects and one new 3DGS model to evaluate. The system analyzes these 3 models to determine if the new model contains a defect and if so where. Defects can be geometric defects, meaning a structural deformation or flaw greater than 4.5mm in size, or visual defects, such as a scratch or a blemish greater than 2mm in size.

3.3 Omni-Scan

Omni-Scan first grasps the object from the tabletop, then while holding it mid-air, scans it by turning the object in front of a fixed camera to capture multiple viewpoints. We then perform a handover, passing the object from one gripper to another to scan it again from a new pose. After collecting the images, we process them with a combination of robot kinematics, Depth Anything, optical flow, and SAM to generate training poses and masks for 3DGS reconstruction. We then train 2 individual Gaussian Splat models (left and right) and merge them into a single, high-quality 3DGS model. We use the resulting model for part inspection by detecting defects compared to

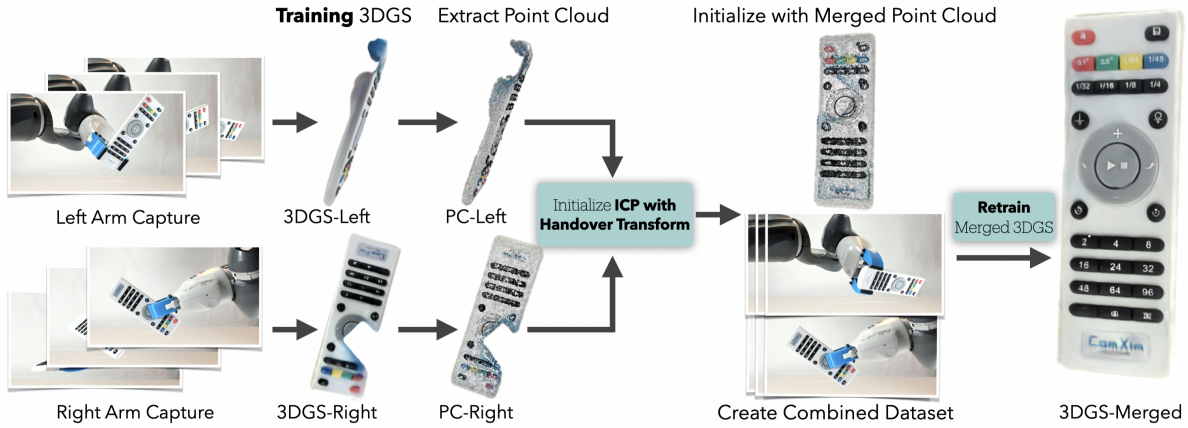


Figure 3.3: **Overview of Training Pipeline** We first train separate 3DGS models for left and right arm captures and extract their Gaussian centers as point clouds. Using the estimated handover transform T_{lr}, we initialize Iterative Closest Point (ICP) algorithm, which iteratively refines the alignment between two point clouds by minimizing the distance between corresponding points, for alignment. The refined transformation from ICP is then used to merge the datasets, enabling training of a unified 3DGS model on the combined dataset.

other examples of the same object.

Scanning Procedure

Tabletop Grasping

We use an ABB YuMi bi-manual robot with soft 3D-printed grippers [28] for compliant caging grasps. Objects are placed randomly within the workspace and captured by a ZED Mini RGB-D camera mounted at the robot base.

We generate a depth image of the object from the stereo image pairs using RAFT-Stereo [72] and one RGB image to generate object masks with SAM [47], filtering the masks by the known location of the table to isolate the object. The depth image is deprojected to create point clouds of both the scene and the isolated object, using DBSCAN [30] to remove noise. Contact-GraspNet [76] then generates candidate grasps on only the object point cloud, and the highest-scoring grasp is planned and executed with the left side gripper using the Jacobi motion planning software [40]. If the grasp is kinematically infeasible or would lead to a collision, the next highest scored grasp is chosen.

Scan Trajectory

After the object is grasped and lifted, the robot performs scanning by rotating the wrist of the gripper 360° in 20 evenly spaced longitudinal positions about its local z-axis. We evenly sample 5 latitudes from the z-axis between -10 and 70 degrees (equaling 100 images). Beyond these limits, occlusions from the gripper prevent the camera from clearly viewing the object. At each latitude,

we collect the pose of the arm that is holding the object T and capture the corresponding 4K image I. The scanning process for one arm takes 6 minutes for 100 images.

Bi-Manual Re-Grasping

Since a portion of the object has been occluded throughout the entire first scan by the robot gripper, the robot then regrasps the object at a different position and scan one more time to capture these regions. To do this the robot moves the object to a predefined end-effector position easily reachable by the other arm. Following a very similar approach as 3.3, Omni-Scan generates grasps on the object point cloud after segmenting the robot arm by deleting depth points overlapping with the URDF model. We then choose the highest scored grasp, accounting for kinematic constraints and collisions. To regrasp the object, the right gripper encloses the object, then the left gripper is released. This right arm then repeats the same scanning process as detailed in 3.3.

Dataset Processing

Pose Processing

We first compute the camera-to-object transform for the left and right scans (100 images per scan). From our calibrated camera, we can get the transform from camera to world T_c. Since we do not directly have the pose of an object center relative to the robot, we approximate it with the transform from the robot to the gripper. The reconstruction of the object is performed in the frame of the gripper.

For each image i, the pose from the camera to the object T_{ic} can be computed by its corresponding $T_i^{-1}T_c$, where T_i is the transform from the robot gripper (that is holding the object) to world, creating capture$_L$ and capture$_R$, consisting of image-transform pairs.

Mask Processing

Our masking pipeline (Figure 3.4) robustly segments the object by systematically filtering out background elements, robot gripper, and robot arm. The pipeline consists of the following key components:

Robot Gripper Segmentation We first segment the robotic gripper to distinguish it from the object being grasped. To achieve this, we train a U-Net segmentation model using 3,000 manually labeled images for 3 objects. The ground truth labels for training U-Net were generated using SAM2 video propagation [68], where manually annotated gripper masks were propagated across frames on a training set of 3 objects. We then run inference on the scans using our trained U-Net models to obtain gripper masks. To refine U-Net masks, we select a pre-defined list of frames where the gripper is unoccluded to prompt SAM2 to generate gripper masks using video propagation.

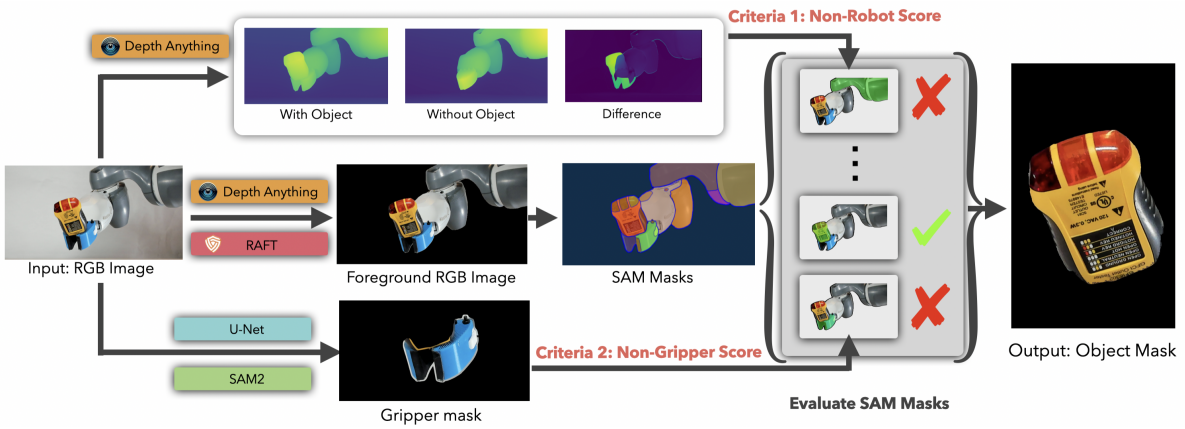


Figure 3.4: **Masking Pipeline** (1) starts with an RGB image of the robot gripper holding an object, (2) extracts the foreground to isolate potential objects, (3) uses SAM to generate candidate object masks, (4) evaluates masks using two criteria: Non-Robot Score (comparing depth with/without object) and Non-Gripper Score (using U-Net and SAM2-generated gripper masks), and (5) outputs a clean object mask containing only the target object, rejecting gripper and robot parts.

Depth and Optical Flow Preprocessing To filter out background elements and distinguish the object from the robot arm, we apply:

1. **Ground Truth Dataset Depth Estimation:** We collected a ground truth dataset where no object is held inside the gripper. We then use DepthAnything V2 [94], a deep learning model for monocular depth estimation, to generate per-pixel depth predictions for each frame. The resulting depth maps are thresholded to segment foreground objects and obtain depth masks. Additionally, we save the predicted per-pixel depth values for all frames as **ground truth depth output**.

2. **Current Dataset Depth Estimation:** We use DepthAnything V2 again to get the predicted per-pixel depth values for all frames as **current depth output** for the current dataset, which will be used to compare with ground truth depth output later in our pipeline.

3. **Current Dataset Optical Flow Refinement:** We notice that DepthAnything may mistakenly classify the floor of the workspace as being close to the camera. To address this, we estimate inter-frame motion using RAFT optical flow [80]. The RAFT model computes dense optical flow by iteratively refining motion estimates at multiple scales using a correlation-based cost volume. We take the intersection of flow masks and depth masks to ensure accurate segmentation, called **foreground masks**. The combination of these masks produce **foreground-filtered images**, which will be fed into SAM in the next step.

Object Mask Generation To isolate the object from the robot and background, first we obtain a foreground mask by thresholding depth values to be small, and in addition keeping only pixels

with non-zero optical flow between neighboring frames. This foreground mask is passed to SAM2 to generate a set of candidate masks. In addition, we process each frame through DepthAnything V2 [94], which provides estimated per-pixel depth. We also perform a one-time calibration where we estimate mono-depth for images from the capture trajectory without an object grasped (empty gripper). The usage of these depth maps is described next.

For each candidate mask M from SAM, we then label it as part of the robot or object based on two scoring functions:

Non-Robot Score:

$$S_{\text{NR}} = \frac{1}{|M|} \sum_{p \in M} |D_{curr}(p) - D_{empty}(p)| \tag{3.1}$$

where D_{curr} is the current depth output, D_{empty} is the empty-gripper depth output, and p represents pixels in the candidate mask. Since the depth of the robot arm in current is typically similar to the empty-gripper depth, the score helps filter out regions corresponding to the arm. In contrast, the object and gripper configuration will differ significantly from the empty gripper depth (where the gripper is fully closed and no objects are in it), resulting in a higher S_{NR} score, indicating a higher likelihood of belonging to the object.

Non-Gripper Score

$$S_{\text{NG}} = 1 - \frac{|M \cap G|}{|M|} \tag{3.2}$$

where G is the gripper mask. A higher S_{NG} score indicates less overlap with the gripper, meaning it's more likely part of the object.

We keep candidate masks with $S_{NR} \geq 150$ and $S_{NG} \geq 0.9$ as our final object mask (threshold empirically determined).

3DGS Training

After obtaining the object masks, Omni-Scan seeks to create one omni-directional Gaussian Splat model of the entire object without occlusions. We do this in the following steps:

1. Create capture_L and capture_R from the left and right arm scans
2. Train Gaussian Splat models, 3DGS_L and 3DGS_R, **individually** on capture_L and capture_R
3. Compute $\text{capture}_{\text{merge}}$ by computing equivalent transforms between capture_L and capture_R
4. Train $\text{3DGS}_{\text{merged}}$ on $\text{capture}_{\text{merge}}$ as the **merged** 3D model

Compute capture_L and capture_R

Using the method outlined in section 3.3, we compute image-transform pairs for al individual scans. This transform is still in the respective grasp frame, so it is only suitable for training the individual models, 3DGS_L and 3DGS_R.

Training Individual Models

We first produce a 3DGS of each of the datasets individually for 16000 steps to get an estimate of the object's geometry. From these splat models, we retrieve colored point clouds P_1, P_2.

Aligning the scans to create capture$_{\text{merge}}$

To create a frame for both captures, we make use of Iterative Closest Point (ICP), an algorithm that iteratively refines the alignment between two point clouds by minimizing the distance between corresponding points. We initialize the relative point cloud transform using the transform between the two robot grippers. Let the left and right gripper positions at handover be T_{lh}, T_{rh}.

Specifically, we make the following definitions. An image taken with the left gripper is i^l, and the right gripper i^r. Its corresponding pose in its *own* frame is T^l_{ic} or T^r_{ic}.

We assign our left capture to be the canonical frame and seek to transform the right capture to the left's frame. It is necessary to compute this transformation in camera frame because 3DGS$_L$ and 3DGS$_R$ are trained with camera to gripper poses. This handover transformation in camera frame is given by $T_c^{-1}T_{lh}$ and $T_c^{-1}T_{lh}$. For each image i taken while the object is held by the right gripper, we can compute its left equivalent transform as

$$T^l_{ic} = (T_c^{-1}T_{lh})^{-1}T_c^{-1}T_{rh}T^r_{ic} \tag{3.3}$$

$$= T_{lh}^{-1}T_cT_c^{-1}T_{rh}T^r_{ic} = T_{lh}^{-1}T_{rh}T^r_{ic} \tag{3.4}$$

We use $T_{lh}^{-1}T_{rh}$ as an initialization for ICP algorithm to align the two colored point clouds P_1, P_2 extracted in 3.3. Let the optimized transform be T^*_{lr}, then the transform for images from the right scan becomes $T^l_{ic} = T^*_{lr}T^r_{ic}$. Transforms for images from the left scan remains unchanged since it is the canonical frame.

Training Omni-Directional Model on Merged Captures

Using the merged colored point clouds $P_1 + T^*_{lr}P_2$ as initialization for the 3DGS model, we train 3DGS$_{\text{merge}}$ on capture$_{\text{merge}}$ for 50000 steps.

Supporting In-Gripper Datasets

For 3DGS training we extend Nerfstudio's Splatfacto model [78, 95] to support multi-dataset training. Naively training a 3DGS on the raw image datasets is infeasible as 3DGS assumes a static scene, while our data seen from the perspective of the camera is inherently inconsistent except for the object. Thus, we must alter the losses to account for this. In addition, we must support training on datasets where the object is occluded by the gripper.

Object Opacity Loss During the training process, Gaussian Splat models produce the **accumulation** metric as well as RGB renders. The accumulation metric measures how much each pixel is covered or influenced by overlapping Gaussians during the rendering process. Accumulation

quantifies the total accumulated alpha (opacity) at each pixel due to the contribution of multiple Gaussians. Lower accumulation values suggest sparse coverage, where fewer Gaussians contribute to the final pixel color. We introduce an L1 loss between the model's **accumulation** and the image's object mask, which attempts to match the rendered opacity to the calculated mask. Intuitively this penalizes any Gaussians outside of the object mask to ensure the resulting model is floater-free and has clean boundaries.

Gripper-Agnostic Losses When combining the datasets, we formulate the loss such that the model is ambivalent towards the area that the gripper occupies. Specifically, any per-pixel loss value that intersects with a gripper mask is set to 0. Importantly, this includes the previously described opacity loss, which ensures the model is able to add Gaussians that are occluded by the gripper in one dataset by analyzing the object from the other dataset's perspective.

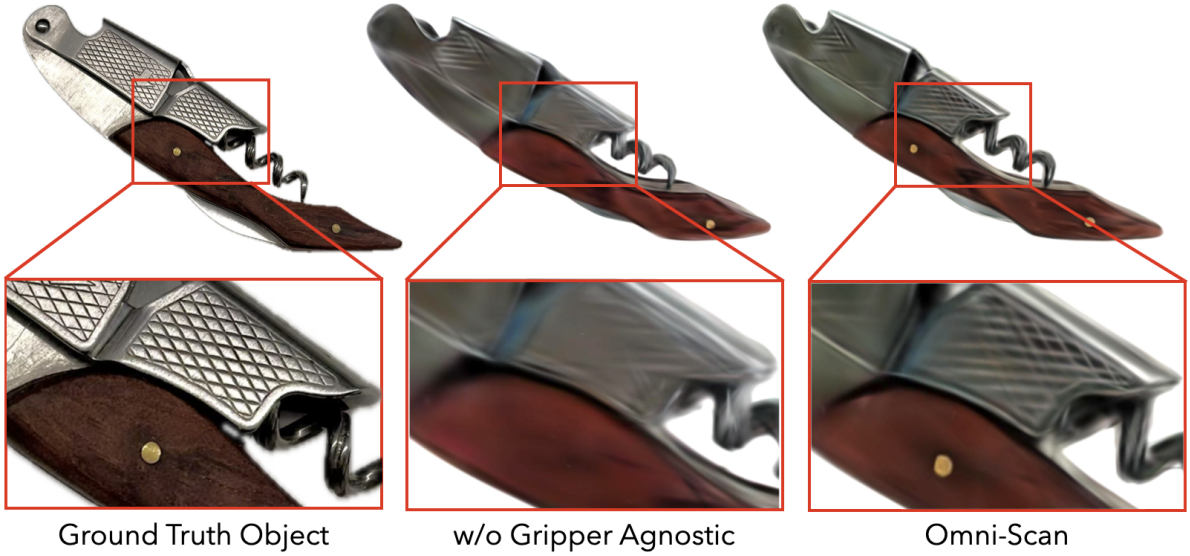


Figure 3.5: **Gripper Agnostic Loss Ablation** We perform an ablation on the Gripper Agnostic Loss, and we observe that reconstruction quality decreases without it. Specifically, the bronze stud and the cross-hatch pattern appear only when we have the Gripper Agnostic Loss.

3.4 Experiments

Physical experiments aim to evaluate 1) the quality of the 3D reconstruction, and 2) the effectiveness of the inspection system for finding defects.

Figure 3.6: **Alignment Ablation** We perform an ablation on the alignment as illustrated in Fig. 3.3 on 5 objects. We present renders for models with no alignment, with only handover transform, and Omni-Scan which uses the handover transform as an initialization for ICP alignment.

	Realsense Camera			Remote Control			Outlet Tester		
	No Alignment	**Handover Only**	**Omni-Scan**	**No Alignment**	**Handover Only**	**Omni-Scan**	**No Alignment**	**Handover Only**	**Omni-Scan**
PSNR ↑	26.52	27.36	**31.12**	23.66	24.52	**26.08**	25.94	24.45	**29.26**
SSIM ↑	0.991	0.991	**0.994**	0.982	0.983	**0.984**	0.986	0.986	**0.989**
LPIPS ↓	0.015	0.015	**0.010**	0.037	0.032	**0.025**	0.019	0.018	**0.011**

Table 3.1: **Omnidirectional Object Reconstruction Quality (Part 1)**. Comparison of reconstruction quality for three objects (Realsense Camera, Remote Control, Outlet Tester). Metrics are averaged over **200** images from left and right gripper scans. Higher PSNR/SSIM and lower LPIPS indicate better reconstruction quality.

	Wine Opener			Wire Connector		
	No Alignment	**Handover Only**	**Omni-Scan**	**No Alignment**	**Handover Only**	**Omni-Scan**
PSNR ↑	23.02	23.95	**30.52**	22.10	22.20	**28.51**
SSIM ↑	0.985	0.984	**0.989**	0.969	0.970	**0.981**
LPIPS ↓	0.020	0.020	**0.011**	0.046	0.038	**0.020**

Table 3.2: **Omnidirectional Object Reconstruction Quality (Part 2)**. Comparison of reconstruction quality for two additional objects (Wine Opener, Wire Connector). Metrics are averaged over **200** images from left and right gripper scans.

Reconstruction

We collect 17 objects for reconstruction, which comprise a range of industrial, office, and household objects. We evaluate the reconstruction quality by comparing object renderings to the 200 ground truth camera images, reporting image similarity metrics (PSNR, SSIM, and LPIPS) on image regions masked by the intersection of the object mask and the accumulation (excluding the gripper) in Table 3.1 and Table 3.2. This penalizes accumulation and shape disparities. $3DGS_{merge}$

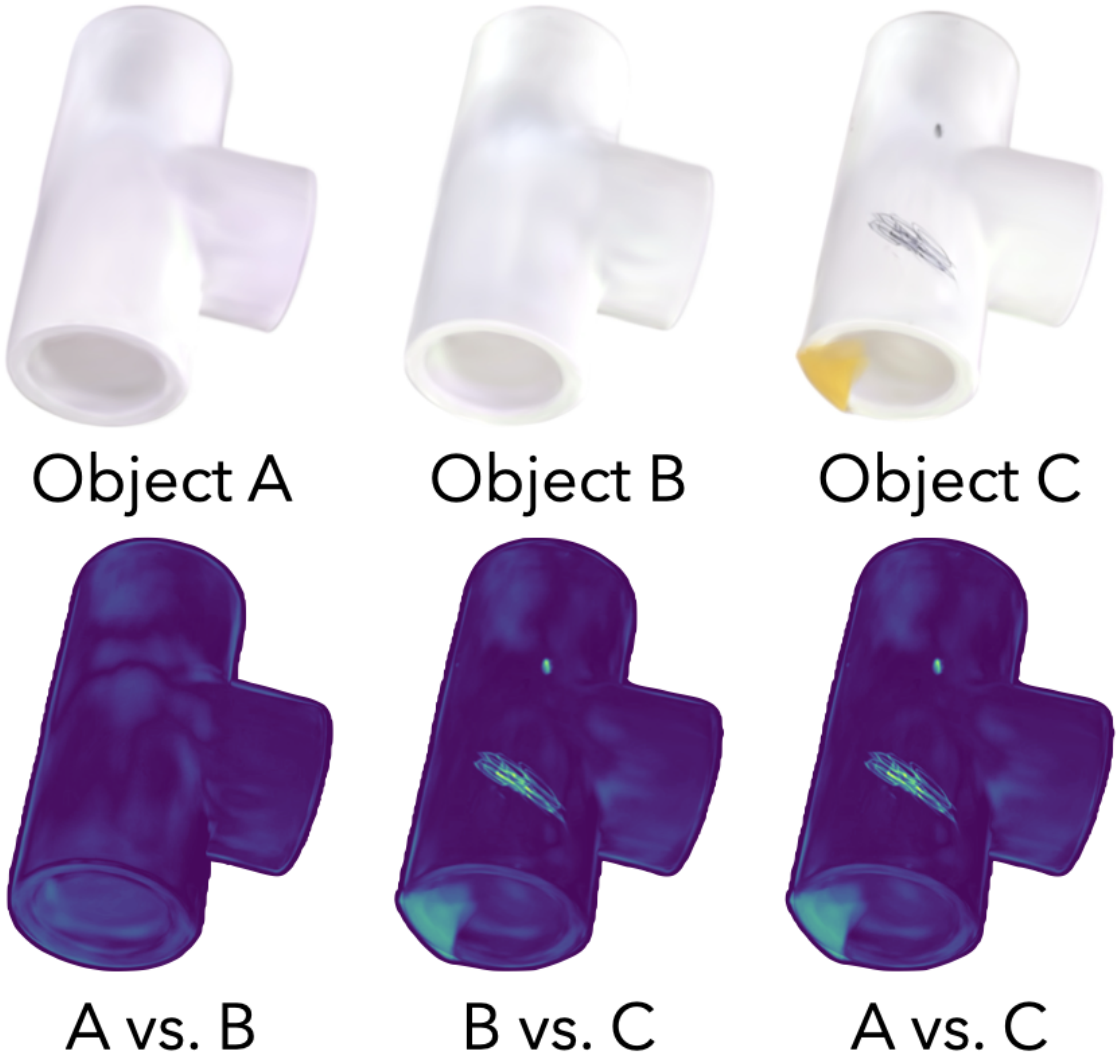


Figure 3.7: **Visual Defect Detection** *Top Row* The rendered RGB of three Omni-Scan models. *Bottom Row* The colorized per-pixel difference after alignment. The highest difference appears in the exact position of the scratch and tape.
is compared to images from the left *and* right hand scans, ensuring that it holistically represents the object.

Results See Figure 3.2 for qualitative multi-view renders of objects reconstructed autonomously by Omni-Scan. Table 3.1 and 3.2 reports image quality metrics across both left and right datasets. Omni-Scan achieves high reconstruction quality, indicating it is able to reconstruct even occluded regions of the object by incorporating information from the un-occluded dataset.

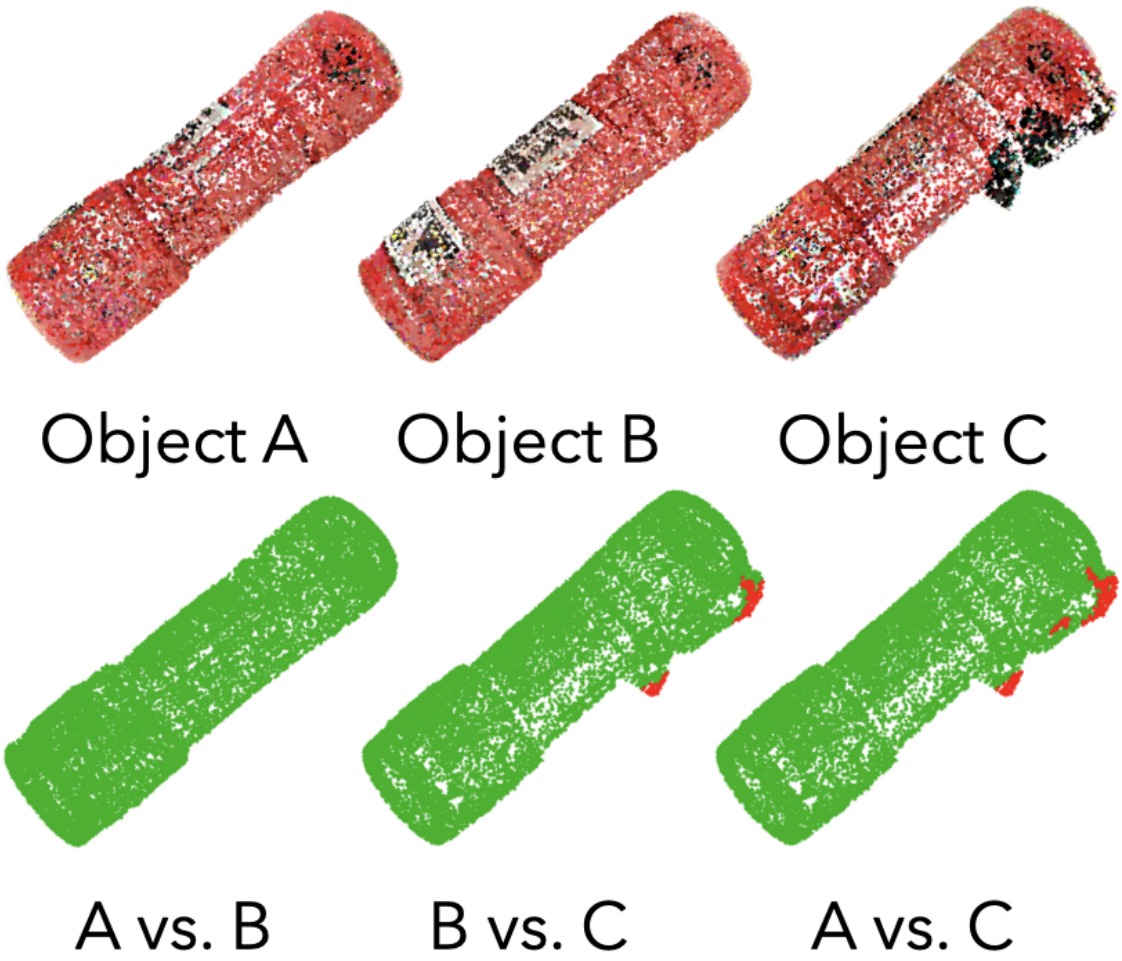


Figure 3.8: **Geometric Defect Detection** *Top Row* The aligned point clouds of three scanned objects. *Bottom Row* The point cloud difference between any two point clouds. Green points are points that are within the minimum distance to any other point on the other point cloud while red points are points which exceed this threshold and are classified as defect points.

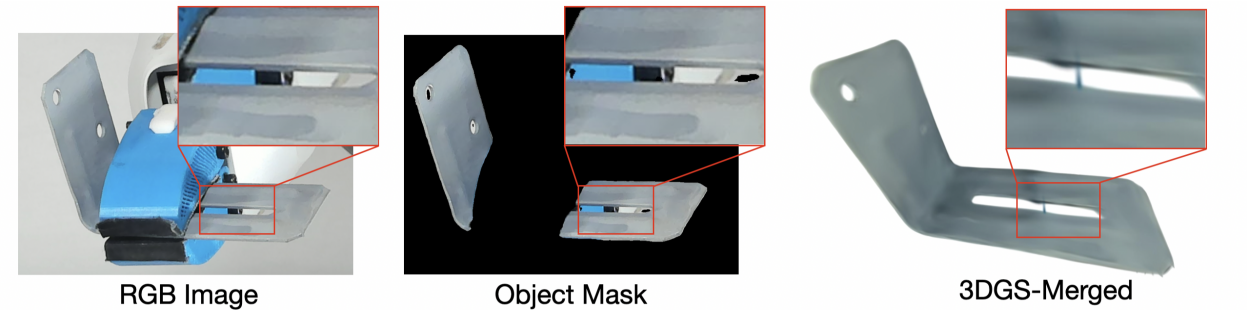


Figure 3.9: **Masking Failure Case** When objects contain cutouts, Omni-Scan may incorrectly include the pixels inside the cutout as part of the mask. This can lead to artifacts in the reconstruction, as shown in the figure, where the middle of the groove appears closed.

Defect Inspection

We apply Omni-Scan for defect inspection on 12 distinct objects, with 3 scans for each object where 2 are of pristine reference objects and 1 contains a visual or geometric defect.

Visual defects are changes made to the visual appearance of the object without significantly affecting its geometry. For the PVC pipe connector in Figure 3.7 we add yellow tape and mark one end of the pipe. Geometric defects are introduced by damaging or otherwise changing the surface geometry of the object. For example, in Figure 3.8 we attach a strap to the end of the flashlight but changes such as bending, breaking, or cutting the object also qualify.

We evaluate the system's ability to identify the defective part of these 3 scans. Omni-Scan highlights the point clouds of physical defects and highlights renders of a difference visual defects. We identify the defective part using a combination of pixel-space analysis and point cloud analysis. We use TEASER++ [93], [70], a fast and robust global registration method, to obtain an initial alignment transformation between the extracted point clouds. This transformation serves as an initialization for ICP, which further refines the alignment between the Gaussian models.

Pixel Differencing We render 100 images from poses that align with the training dataset for the first dataset. Then using the alignment transform of the following 2 datasets, we compute renders of the same location and orientation. We can then directly compute the per-pixel difference of these two renders to evaluate the difference of the models. Since the two non-defective objects should be indistinguishable, we can compare pair-wise distances, and the smallest distance pair are the non-defective parts with the remainder being the defective one as demonstrated in Fig. 3.7.

Pixel Differencing Results Omni-Scan successfully detects visual defects in 4 out of 5 trials. We successfully identified defects such as scratches and tape on the pipe connector as illustrated in Fig. 3.7. Results suggest that our alignment pipeline can achieve pixel-level accuracy. The source of the failure cases is in the masking pipeline, where a portion of the gripper remains inside the object mask. This leads to artifacts in the merged 3DGS, resulting in one non-defective object being significantly different than the other non-defective one. We illustrate this in Fig. 3.9.

Point Cloud Differencing Given the aligned point clouds for any two objects, we compute the difference between them. This is done by computing the minimum distance from a point in one point cloud to any point in the other point cloud. If a point's minimum distance from any other point exceeds our distance threshold of 4.5mm (empirically determined based on our set of objects to cause no false positives), then we classify it as a defective point.

Geometric Defect Detection Results Omni-Scan is able to correctly identify the geometric defect in 6 out of 7 trials. These results indicate that the point clouds generated by training a Omni-Scan are quite consistent among different undamaged objects as they have next to no defect points which exceeded our distance threshold of 4.5mm. This also further reinforces the ability of the alignment pipeline to properly align these models. Point cloud differencing fails on the pressure sensor with a geometric defect of slight sanding on one end of the object and a cut made on another end. These defects are marginal and the resulting point cloud does not noticeably differ from the two reference object point clouds.

3.5 Limitations

One limitation of Omni-Scan is with specularities. When scanning metallic objects, the color as well as the brightness can change depending on the pose of the camera to the object. This leads to issues with alignment and pixel differencing, since the same point on the object may look very different to the model depending on how it was grasped/ scanned. The system also relies on the handover pose as a good initialization for the Iterative Closest Point to estimate the transform between the left and right datasets. If the object slips significantly during handover, the resulting pose estimation ceases to be accurate, and the overall model quality suffers as a result. Since 3DGS models can contain gaussians in their interior, geometric differencing sometimes presents spurious false positives. Future work will explore mesh-based approaches for geometric differencing which better localize geometric defects.

3.6 Conclusion

We present Omni-Scan, a system for autonomous high-quality robotic creation of omni-directional digital twins and defect inspection. Experiments suggest that Omni-Scan constructs models with sufficient visual fidelity to detect visual and geometric defects on household, office, and industrial objects with up to 83% accuracy.

Chapter 4

Shape Representation and Deep Learning Architectures

An Analysis for Additive Manufacturing Part Quality Predictions

4.1 Introduction

AM (often colloquially referred to as 3D printing) processes are a group of relatively novel manufacturing processes that were originally used for prototyping, but are increasingly used to fabricate end-use parts [91], which requires higher quality manufacturing. AM processes give engineers and designers greater geometric design flexibility and shorter product lead times, but are generally more defect-prone than traditional manufacturing processes. AM allows companies to cut outsourcing costs, iterate designs more rapidly, speed up the product development cycle, generate parts with complex geometries, and customize parts. The flexibility in geometric design and potential for faster iterative design of AM processes give them the potential to revolutionize industries such as aerospace and medicine.

With the improvement of AM capabilities, these manufacturing processes will shape the way we design and manufacture a variety of goods. However, AM processes often have substandard mechanical properties and are prone to manufacturing, or build, failures [84]. Improving AM manufacturing quality and reliability is important if these processes are to be used for safety-critical end-use parts.

Most AM processes fabricate parts from the bottom up, layer-by-layer. A 3D CAD model is sliced into thin layers using a slicing software (e.g. Cura [24]). Then the AM process creates one layer of the part at a time, for example by depositing or sintering material on the 2D plane for that layer. Then the machine proceeds to the next layer and repeats until the part is complete. In the case of fused deposition modeling (FDM), one of the most common AM processes, the equipment heats up a plastic filament, which is fed through a nozzle and deposited onto the build platform layer by layer to form the object.

Evaluating if a CAD part in a particular orientation will be manufactured successfully is an on-

going challenge. Researchers have used approaches that include exhaustive search, genetic algorithms, simulated annealing, particle swarm optimization, nonlinear programming solver, heuristic algorithms, and others [11]. These approaches tend to optimize a single objective (e.g. the amount of support volume required) in a particular use case and very infrequently include comparisons to other algorithms, making it challenging to assess algorithm success.

Eranpurwala et al. [29] converts "synthetically" generated mesh files (STL files) into 64^3 voxelized files and uses a 3D convolutional neural network (CNN) to predict optimal print orientation. Parts are labeled by the amount of support volume required to print. The support volume required is calculated using an analytical formula.

Our research builds on this research to predict AM print quality in a given orientation using DL pipelines. However, our main contributions are providing insights into the impact of shape representation/DL architecture and dataset size on AM print quality predictions. While the focus of this research is on one build quality metric for FDM, we hope the findings of this work can be used to help inform other AM print quality predictions with similar data distributions.

The next section (Methods) describes our research methodology including data collection, labeling, and pre-processing, DL pipeline details and selection rationale, and the DL pipeline comparison framework. The following section (Results) describes the experiments and results from the input shape representation/DL model pipelines including hyperparameter tuning, model complexity and runtime requirements, and dataset size sensitivity analysis. The final section discusses potential directions for future work and conclusions.

4.2 Methods

A major gap in research this study addresses is evaluating what shape representation and DL model pairings have enough expressivity to accurately predict AM print quality and performance metrics. This section explains why we selected particular shape representation/DL pipelines, how we collected and processed the part data and created the AM print quality labels, the details of our deep learning (DL) approaches, and finally the process used to systematically compare DL pipelines and evaluate dataset requirements.

We build a 64^3 voxel 3D CNN pipeline because it is a natural analog to CNN image recognition pipelines and as a comparison to other research groups that have used the voxel 3D CNN pipeline for similar research [29, 90, 25]. Related research uses 64^3 voxels with small to medium size datasets (840 – 72,000 parts).

Data collection, labeling, and shape representations

The part data we use are real world parts from the unlabeled FabWave data repository [10], which were in turn scraped from GrabCAD and Autodesk Gallery as mesh files (STL files). We collect over 80,300 parts. We rotate all parts into five additional orientations to place each other axis facing up, i.e., rotating the original part with the $+z$-axis oriented up to instead have the $\pm x$-axis, $\pm y$-axis, and the $-z$-axis oriented up for the five additional orientations. This gives us over 482,000 part-

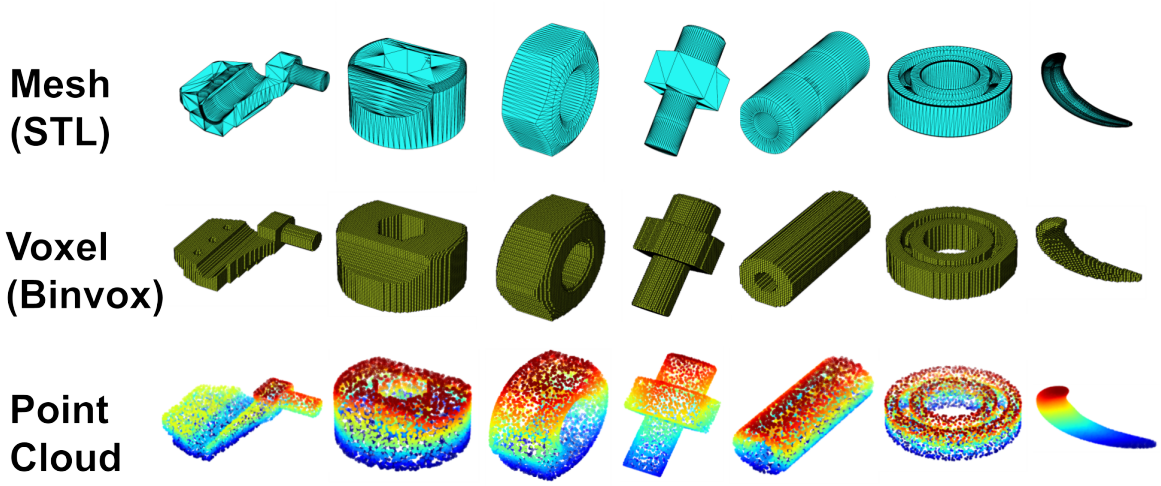


Figure 4.1: Seven example input parts in three shape representations we include in our dataset (original part representation: mesh/STL, voxel/Binvox, point cloud). One of the contributions of this research is training on a large dataset of real mechanical parts, instead of parts synthesized just for research, in multiple shape representations. We show our final two shape representations in the next Figure because they are more involved.

orientation combinations. Rotating parts into new orientations creates a more random distribution of part-orientation combinations and increases the dataset size.

The dataset is randomly shuffled and we first separate out a 10,000 part validation dataset (which we use to validate all models in this research). Then we randomly sample a 200,000 part training dataset from the parts left. We take the first 1,000, 10,000, and 50,000 parts as subsets to explore the effect of dataset size. Finally, we remove any part in these training datasets that exists in the validation dataset in any orientation, even though different orientations usually have different labels, to ensure complete isolation between the training and validation datasets. This gives us four final training datasets of sizes 911, 9,041, 45,091, and 180,294.

We explore different input shape representations to evaluate which input shape representation does the best job of predicting print quality and to try and find the most efficient input (i.e. an input shape representation that will get comparable results in less time or with less computational resources). In robotics, 3D understanding is important for tasks such as navigation and manipulation. In navigation, point clouds are [98] used in SLAM for rapidly exploration. In grasping, depth images have been shown to be good representations for DL models as seen it the Dex-Net series of work [56]. In motion planning, distance fields are useful to approximate a scene in RRT and the GOMP series. We test the following shape representations, in addition to voxels, also with the 3D CNN DL architecture: (1) distance fields and (2) depth images. Finally, we tried a new shape representation/DL architecture pairing, a point cloud Transformer model.

Voxel 3D CNN represents perhaps the most intuitive and easy to understand pairing of shape representation and DL architecture. Different voxel 3D CNN architectures have been used in sev-

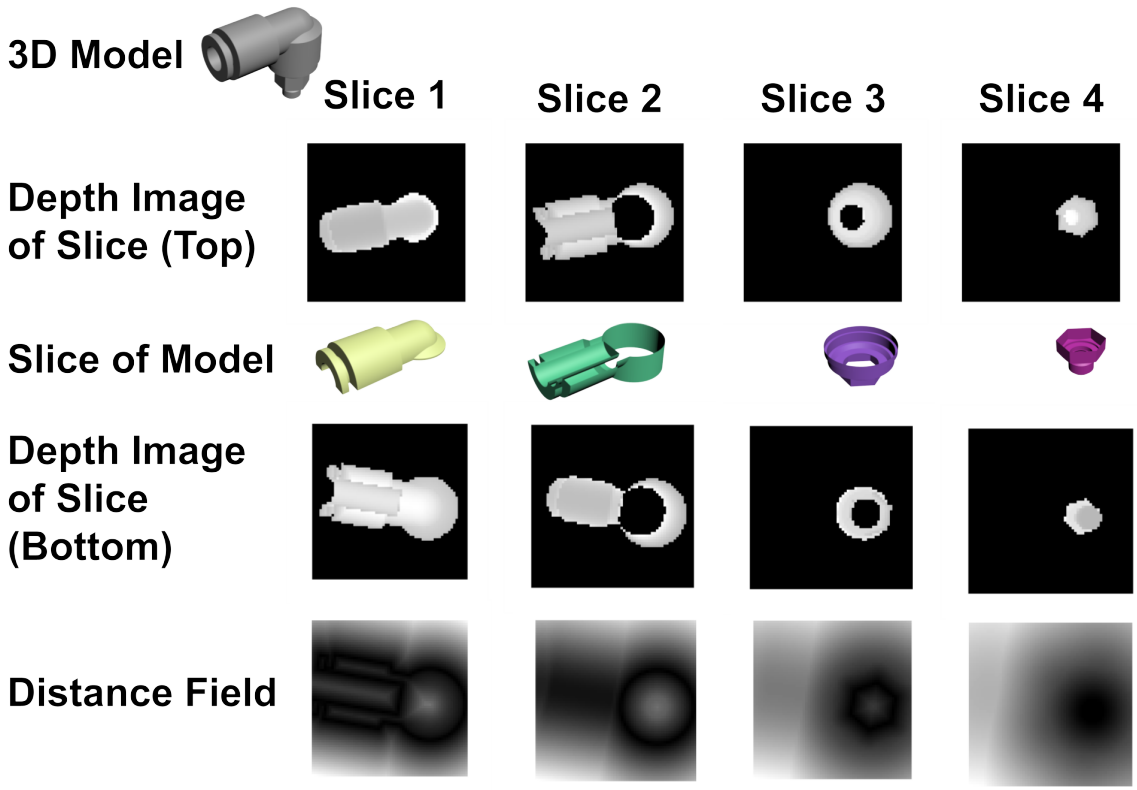


Figure 4.2: One example part, shown in the top left corner, in depth image (from top and bottom) and distance field shape representations. For the actual data representations, parts are split into 64 slices (or layers). For visualization purposes this part is only split into four slices.

eral related papers using 64^3 voxelized input. The depth image and distance field representations mimic the AM process. Point clouds have been used in a variety of DL applications, such as the Point Transformer architecture, which is designed to capture complex local and global relationships from point cloud inputs.

To evaluate input dataset requirements, we isolate input resolution and increase the input resolution from 64^3 to 256^3 to see the impact of the input resolution on performance.

We convert the mesh files (STL files) into 64^3 and 256^3 resolution voxelized files (Binvox files [59, 63]) and point clouds ($2,048 \times 3$ coordinates using `trimesh` [20] , where $2,048$ is the number of points in the point cloud). Example parts in each of these three shape representations are shown in Fig. 4.1.

We transform the parts into the depth image shape representation using the pinhole camera model: starting from a point in space (either the top or bottom of the part), we create one ray per pixel and return the distance traveled until the ray intersects with a surface of the part. If no intersections happen, we return zero. We evenly slice each mesh into 64 horizontal slices to match the resolution of the 64^3 voxel shape representation. Each slice is lowered so that its centroid lies

on the X-Y plane. We compute two depth images for each slice, one taken from above and one taken from below. Each depth image has a resolution 64×64, for a total resolution of 64^3 for the depth image representation of the part.

Each part's distance field shape representation is created by computing the unsigned distance from each query point to the nearest face. Query points are sampled in a square grid and resized to fit each mesh. Each square grid plane is perpendicular to the z-plane and axis-aligned. These planes are positioned to evenly slice the mesh into 64 pieces, identical to the height of the cuts to generate distance fields. Each square grid contains 64×64 points, and thus the total resolution of the field is 64^3. The depth image and distance field shape representations (example in Fig. 4.2) input the part to the DL model as a series of layers, which is similar to how the part is fabricated and thus we predict an expressive input for the DL pipeline.

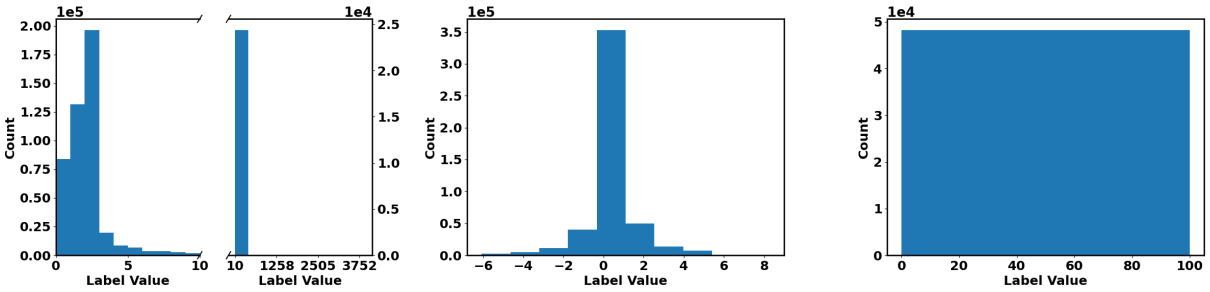
To label the full dataset with AM print quality, we calculate an unprintability score (UPS) for every part in our dataset using the analytical research software Tweaker3 [71], an open source slicer plug-in. Tweaker3 uses three factors to calculate the unprintability score (UPS): (1) area touching the print bed, (2) overhang area (meshes where the angle is greater than 45 degrees), and (3) contour length of the area touching the print bed. (These factors are related to stability, volume of support structure, and warping, respectively.)

Lower UPS scores represent a lower risk of AM print failure. The UPS Tweaker3 output is a description of the AM printability of the part. The labels range from $[0, \infty)$, with zero representing a highly printable orientation-part combination, and larger numbers representing more problematic part-orientation combinations. Using UPS as the labels allows us to label hundreds of thousands of parts and perform DL analysis on these parts.

The UPS label data is heavily right-skewed, as illustrated in Fig. 4.3a: most of the parts have relatively small unprintability scores, but a small percentage of the parts have much larger scores. It is important to be able to predict these larger values because these scores represent parts that likely will have problems in the manufacturing process. It is more challenging to predict right-skewed data, but being able to predict the right-skewed UPS scores is a good simulator for being able to predict more computationally expensive and complicated manufacturing metrics, because other manufacturing quality data may also be skewed with a small percentage of problematic areas that are important to identify (i.e., FEA predictions of areas of high process-induced distortion).

It is reasonable to expect other types of AM manufacturing labels to be non-uniform because more parts will successfully print (or have low residual stress, distortion, etc.) than not. If that were not the case, AM manufacturing processes would not be as successful as they are. On the other hand, there are still a significant number of print failures and it is highly important that any DL model be able to successfully predict when prints will fail (or have issues) even though this may be a small proportion of the total data.

Label data can be processed in several ways to handle skewed data, including taking the log of the data (Fig. 4.3b) and converting scores to their percentiles (Fig. 4.3c). We chose to pre-process the label data into percentiles (i.e., calculate what percentage of the data a value is larger than) in order to make the data evenly distributed. The processed UPS scores range from zero to one.



(a) Raw data distribution. (b) Log data distribution. (c) Percentile data distribution.

Figure 4.3: Histograms of the unprintability score (deep learning label) data: (a) raw data distribution, (b) log data distribution, and (c) percentile data distribution. The raw data, shown in (a), is heavily right skewed. For the raw data graph, the x-axis scale is split to make it possible to visualize the number of higher unprintability scores. We show the data processed by taking the log in (b) and using percentiles in (c). Ultimately we pre-process the data using percentiles so that the unprintability score (labels) are uniformly distributed. This makes the error results more meaningful. If we left the data skewed, we could get results that appeared excellent merely by guessing a low value unprintability score at all times, but actually did not indicate significant learning.

The importance of baseline models

Generally, the goal of DL pipelines in this field is to learn some information about AM build quality. There are a variety of metrics to measure how much the DL model has learned (e.g. L1 loss), but it can be difficult to fairly evaluate how much a DL model has actually learned from performance metrics alone. For comparison, an appropriate baseline model should always be constructed, which is used to give the researcher an idea of how much the DL model(s) learned compared to a simple alternative. We construct a midpoint model that guesses that guesses the average of 0.50, which is the midpoint of the uniformly distributed, processed UPS scores.

This comparison addresses the risk with research projects that use a limited variety of geometries (especially when the geometries are relatively simple) that a DL model may look like it's getting good results, when in reality it is guessing near the average or mode result for all parts. Note that had we used the log of the data (as visualized in Fig. 4.3b) instead of percentiles to pre-process our data, an appropriate baseline model would be approximately at the mean of a normal distribution fitted to the log data. Because of the narrow normal distribution of the logs, a DL model that learned to predict the mean log value, regardless of the input, would have appeared to have low error if error alone was reported without comparison to this appropriate baseline that could have obtained a similarly low error.

Alternatively if we had not pre-processed the highly skewed data at all, we could always guess low UPS values and get lower error than if we attempted to learn when the UPS value is high. However, it is important to be able to predict high UPS scores, which represent likely problematic 3D prints. This is why it is crucial to include an appropriate baseline model in DL applications. A

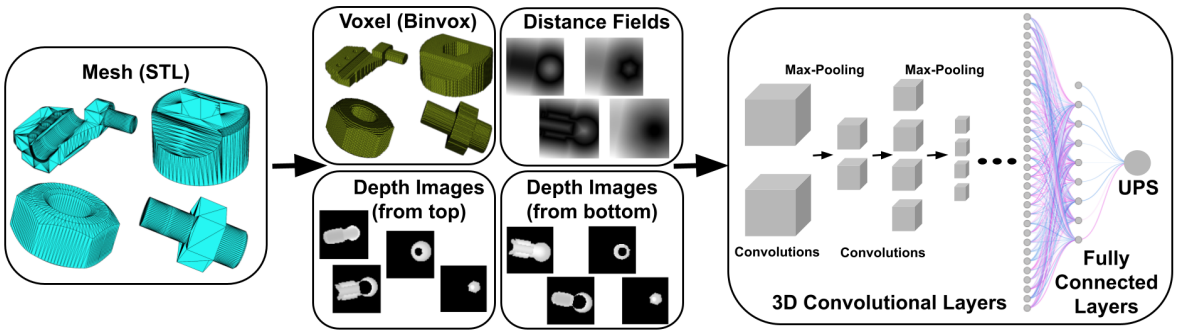


Figure 4.4: 3D CNN model pipelines. Compiled dataset of parts are in the STL file format. Parts are converted to 64^3 and 256^3 voxelized (Binvox) format, 64^3 distance field format, and two 64^3 depth image formats using an automated pipeline and then fed into the 3D CNN model which consists of five or six convolutional layers (depending on input voxel resolution) and two fully connected layers that reduce the output to a single score, the unprintability score (UPS).

baseline model makes it clear how much learning is taking place.

Deep learning pipelines

Voxel, distance field, and depth image 3D CNN models

Three-dimensional convolutional neural networks (3D CNNs) are a class of deep learning models designed to process, analyze, and learn from 3D input data. These networks extend the traditional 2D image-based CNN architecture by adding a third dimension to the input and performing convolutions in 3D space.

We have two closely related 3D CNN architectures: one for the 64^3 resolution voxel, distance field, and depth image inputs and one for the 256^3 resolution voxel input. The overall voxel 3D CNN pipeline is in Fig. 4.4. Each convolutional set contains the convolutional layer itself, followed by batch normalization, and then a non-linear activation function. The convolutional layer kernel size is a hyperparameter that we tune. The outputs from the convolutional layers are padded with zeros to retain their original dimensions. Each set is followed by a max pooling layer with kernel size 2x2x2, which compresses the data to half of the original resolution in all three dimensions with the goal of extracting the latent features from the convolution. The channel count of the convolutional layers also varies, with each layer outputting twice or four times the number of channels it takes in, as described next.

For the 64^3 voxel resolution inputs, our DL architecture has five convolutional sets. We choose five convolutional sets to reduce the resolution from 64^3 to 4^3 by halving the resolution at each layer. The number of channels in each hidden layer increases from two output channels (or nodes) for the first hidden layer to 32 for the final 3D CNN hidden layer. After the convolution sets, a final max pooling layer with kernel size 4x4x4 reduces the 32 channels into 32 numbers (a 1D array of numbers). Two fully connected linear layers each with weights for every neuron and a bias at the

end to shift the output follow, the first mapping those 32 numbers to 8 numbers, then the second from 8 to 1. This final number is the predicted unprintability score.

For the 256^3 voxel resolution, our DL architecture has six convolutional sets. We choose six convolutional sets to reduce the resolution from 256^3 to 8^3 by halving the resolution at each layer. The number of channels in each hidden layer increases from two output channels (or nodes) for the first hidden layer to 256 for the final 3D CNN hidden layer. After the convolution sets, a final max pooling layer with kernel size 8x8x8 reduces the 256 channels into 256 numbers (a 1D array of numbers). Two fully connected linear layers each with weights for every neuron and a bias at the end to shift the output follow, the first mapping those 256 numbers to 16 numbers, then the second from 16 to the UPS.

For both architectures, the networks gradually compress the input data, and extract the latent features from within the convolutional layers. The channel count keeps increasing through the convolution sets in order for the 3D CNN to capture more features from the inputs while compressing the data. Finally, linear layers transform the spacial data we get from the convolutions into the prediction of the UPS.

Point Cloud Transformer Model

In the Transformer architecture [83], multi-headed attention allows the network to learn relational information in different parts of the input simultaneously, which is useful in our context because AM print quality is determined not by the existence of each individual point but by how interactions between points impacts the manufacturing process. An advantage of multi-headed attention is that it is inherently permutation invariant, which is a desired feature in this research since the ordering of individual points does not affect part shape.

Point Transformers [96] build on the PointNet work by using self-attention, which relates different parts of the input geometry to each other, to capture relationships between the input points. Point Transformers outperformed the next best architecture by almost 3% on the segmentation task measured by mean intersection over union (mIoU) as of 2021 when the paper was published [96]. Point Transfomers use vector attention instead of the more common scalar attention because scalar attention loses spatial information (vector attention can capture geometric relationships in 3D and scalar attention cannot) and is not permutation invariant.

Point Transformers use vector attention in a transformer layer. The prediction network has n repeating blocks, each block containing a *Transition Down* layer and a *Transformer Layer*. For the *Transition Down layer*, k-nearest neighbors (kNN) is used to pool local features that are processed with a multi-layer perceptron (MLP). The output is then locally max-pooled to reduce the number of features. We perform a hyperparameter search on this k as well as the n in our results section.

Methodology for Systematic Comparison of DL Models

To evaluate our DL pipelines, we compare performance (using validation L1 loss defined as $L(UPS, \hat{UPS}) = \frac{1}{n}\sum_{i=1}^{n} |UPS_i - \hat{UPS}_i|$), model efficiency (measured by runtime), and hyperparameter tuning sensitivity (by evaluating the average and standard deviation of our results). The

DL pipelines are also directly compared to the baseline model. We contribute to the existing gap in knowledge by comparing and evaluating the DL pipelines and creating a framework for future researchers to evaluate additional DL pipelines, which will help determine what DL pipelines work best for predicting AM print quality. This framework can also be used to compare performance for new areas of DL application.

4.3 Results

We use our DL pipelines and baseline model to predict 3D printing part quality, defined by the UPS. Note that although we predict UPS, this is just one example of a manufacturing part quality or functional performance metric that our models could predict. Predicting UPS is a challenging research problem because the regression data is heavily right-skewed. We believe UPS may have similar data characteristics to other AM manufacturing process labels and hopefully some of our results can be used to inform other data-driven AM prediction research.

The DL pipelines are trained on a range of dataset sizes, from 911 to 180,294. All models are validated on a separate 10,000 part validation dataset.

Mid-point baseline model performance

We run the mid-point baseline model, which predicts that every part will have a processed UPS score of 0.5, on the validation dataset. The mid-point baseline has an L1 loss on the validation set of 0.2496, which is in line with the expected value of 0.25 when guessing 0.5 for every input.

Voxel 3D CNN model performance

We train and tune our 64^3 voxel 3D CNN model on four datasets of sizes 911, 9,041, 45,091 and 180,294, and our 256^3 voxel 3D CNN model on three datasets of sizes 911, 9,041, and 45,091.

We tune the following hyperparameters: kernel size (KS), learning rate (LR), batch size (BS), and non-linear activation function. Kernel size is tuned as either three or five. Learning rate is tuned as 0.001, 0.0001, or 0.00001. Batch size is tuned as 4, 8, or 32 for the 64^3 voxel models, but always 4 for the 256^3 voxel models because that is the largest batch size that can be loaded. Non-linear activation function was initially trained as ReLU or Sigmoid, but we report final results for just ReLU because, as we show below, ReLU performs better across the board.

We do not train the 256^3 voxel 3D CNN model on the 180,294 part dataset because that would be highly resource intensive and we find that we already have enough data to show that increasing dataset size has more impact on model predictive performance than increasing input resolution. Similarly, we train the models for 20 epochs in all cases except for the 256^3 voxel 3D CNN model when trained on the 45,091 dataset. In this single case, we use 10 epochs because of how long it takes to run just one hyperparameter combination (i.e. over a day).

Both voxel 3D CNN models perform better when trained on more data, and the 256^3 voxel 3D CNN model performs better than the 64^3 voxel 3D CNN model when trained on the same size

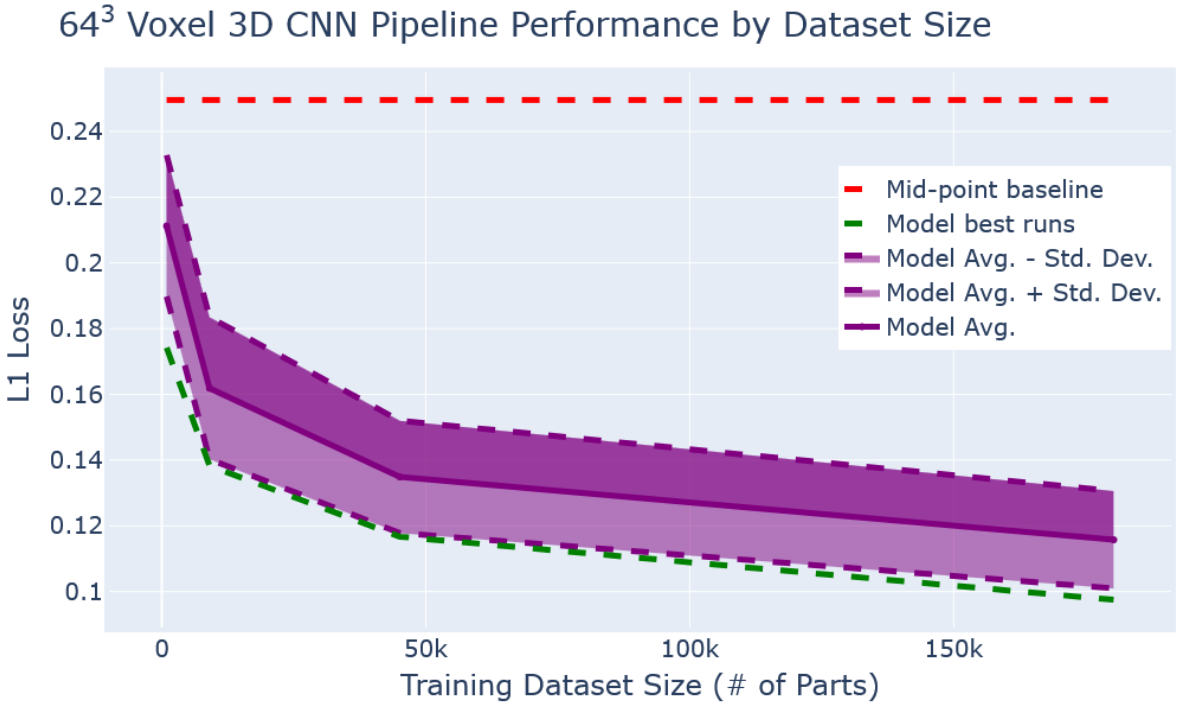


Figure 4.5: 64^3 voxel 3D CNN model performance results (measured using L1 loss) compared to the mid-point baseline model for 911, 9,041, 45,091, and 180,294 training datasets. Model performance improves significantly with increasing training dataset size; however, the effects of additional data decrease as dataset size increases.

dataset. As a trade-off, larger dataset size seems to have a greater impact on performance than higher input resolution (see Tables 4.2 and 4.3). An example: increasing dataset size by a factor of approximately 20 (from 9,041 to 180,294) while decreasing input resolution by a factor of 64 (from 256^3 to 64^3) has a better best performing DL model and better average DL model performance.

Both voxel 3D CNN models' ability to generalize from the training data to the validation data differs based on how much data the model was trained on. For example, for the 64^3 3D CNN model trained on the 180,294 part dataset, the validation L1 loss can be minimized to 0.0975 with the best hyperparameters, compared to 0.1167 for the 45,091 dataset, 0.1383 for the 9,041 part dataset, and 0.1742 for the 911 part dataset with their best hyperparameters. The higher validation losses when training on the smaller datasets occurs mostly due to more overfitting to the training data for the smaller datasets. This can be inferred because the 180,294 part dataset has only a somewhat lower training L1 loss than the 911, 9,041, and 45,091 part datasets, but it has a much lower validation loss. This effect can be seen in Fig. 4.6, which compares training and validation losses for the models trained on the 911 and 45,091 part datasets. The model trained on the larger (45,091 parts) dataset does not overfit, which we can see from the similar training and validation losses, whereas for the smaller (911 parts) dataset, the model does seem to overfit, because the training loss is much lower than the validation loss.

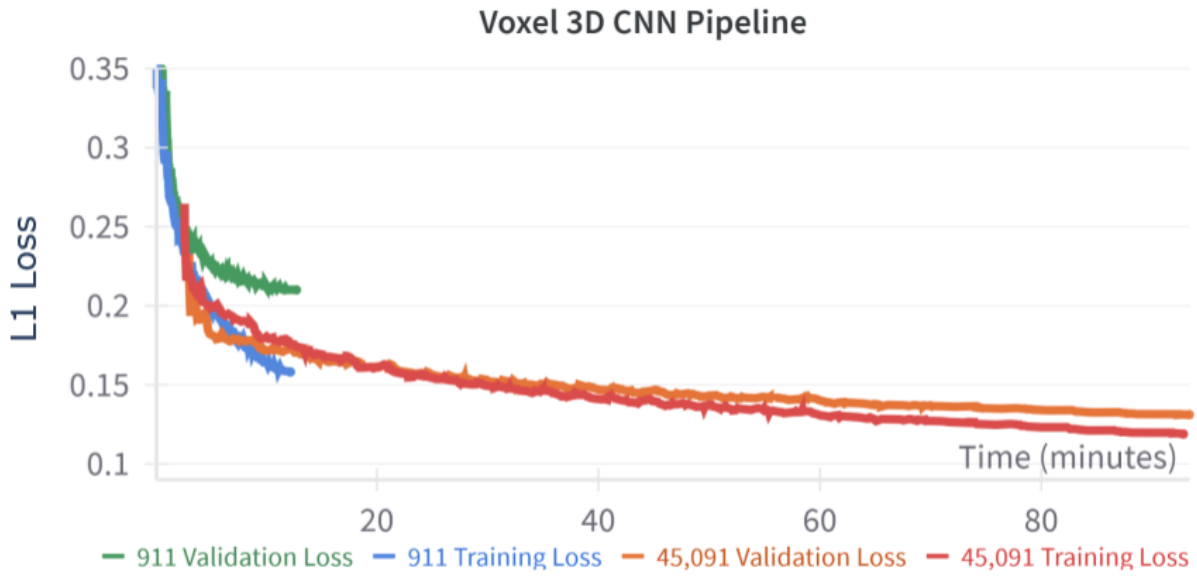


Figure 4.6: Comparison of training and validation losses for the 64^3 3D CNN models trained on 911 and 45,091 part datasets. Graph shows mean and one standard deviation in each direction for training and validation on each dataset. It can be seen that the model overfits more on the 911 part dataset than the 45,091 part dataset due to the larger gap between training and validation performance for the model trained on the 911 part dataset.

The average performance (measured by average validation L1 loss) is significantly better when the 64^3 voxel 3D CNN model is trained on the 180,294 part training dataset or the 256^3 voxel 3D CNN on the 45,091 part training dataset compared to the smaller datasets for each resolution. For the 64^3 voxel 3D CNN model, the average L1 loss is 0.1158 for the 180,294 part dataset compared to 0.1349 for the 45,091 part dataset, 0.1619 for the 9,041 part dataset, and 0.2113 for the 911 part dataset (see Fig. 4.5 for full visualization).

Another advantage of training on the larger datasets is that it makes the model less sensitive to hyperparameter tuning. For example, on the 64^3 voxel 3D CNN model, the standard deviation of the loss is lower for the 180,294 part dataset, 0.0148 compared to 0.0171 for the 45,091 part dataset, and 0.0216 for the 9,041 part dataset.

For the 256^3 voxel 3D CNN model, the average L1 loss is 0.1185 for the 45,091 part dataset, compared to 0.1469 and 0.1936 for the smaller training datasets. For the 45,091 part dataset, the standard deviation in the L1 loss results is also smaller, 0.0099, compared to 0.0177 and 0.0237 for the smaller datasets.

For 64^3 and 256^3 voxel 3D CNN pipelines, increasing dataset size by approximately a factor of 10 (from 911 to 9,041) decreases L1 loss (thus improving model predictive performance) by 23.4% and 24.1%. Increasing dataset size again, this time by a factor of approximately five, decreases L1 loss by 16.7% and 19.3%. The final roughly four-fold increase in dataset size for the 64^3 input

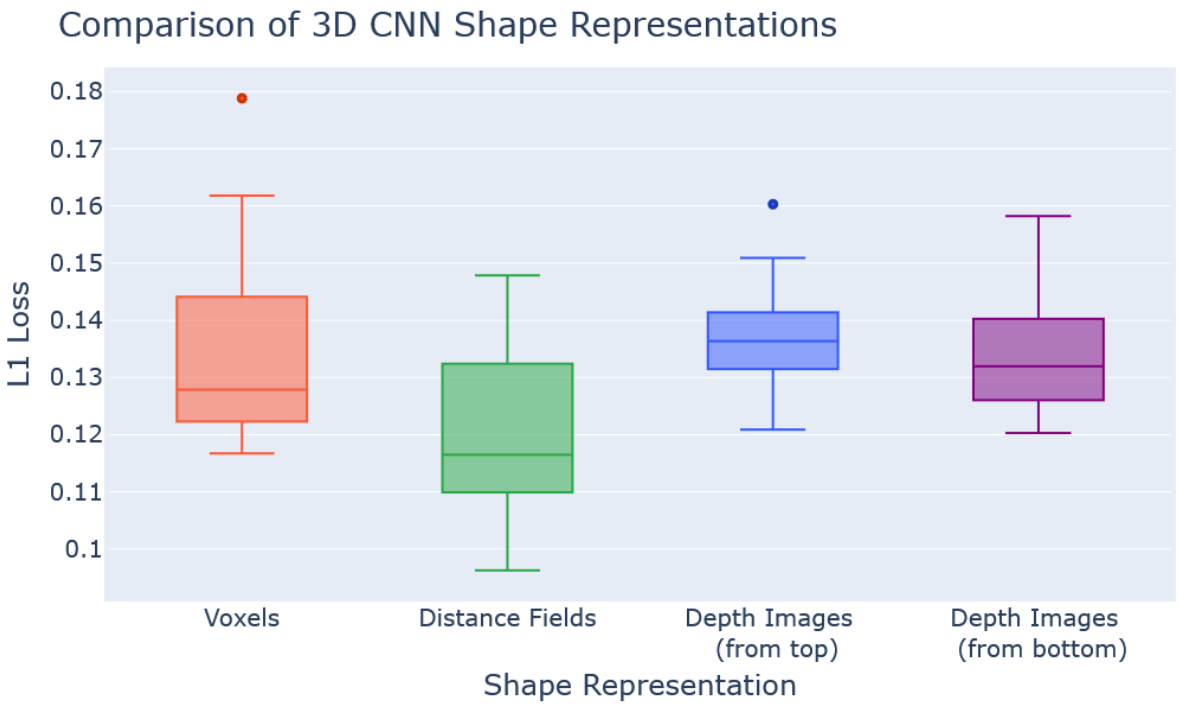


Figure 4.7: Comparison of 64^3 voxel, distance field, and depth image 3D CNN models' performance (measured using L1 loss) trained on the 45,091 part dataset.

resolution decreases L1 loss another 14.2%. In contrast, increasing input resolution by a factor of 64 only decreases L1 loss by 8.4%, 9.3%, and 12.2% (for the 911, 9,041, and 45,091 dataset sizes).

Distance field and depth image 3D CNN model performance

We tune the 64^3 distance field and depth image 3D CNN models on the exact same hyperparameter combinations as the 64^3 voxel 3D CNN model. For the same dataset size (45,091) and input resolution (64^3), the average and best distance field 3D CNN model performs better than both depth image and voxel 3D CNN models. Furthermore, the average 64^3 distance field 3D CNN model trained on a dataset of 45,091 performs almost as well as the average 64^3 voxel 3D CNN model trained on a dataset of 180,294. A comparison between voxel, depth image, and distance field 3D CNN models, all trained on the 45,091 part dataset, is shown in Fig. 4.7. The performance results for the distance field 3D CNN, which performs better than either depth image 3D CNN model, trained on a range of dataset sizes, is illustrated in Fig. 4.8.

Point cloud Transformer model performance

The point cloud Transformer model is trained and tuned on three dataset sizes: 911, 9,041, and 45,091.

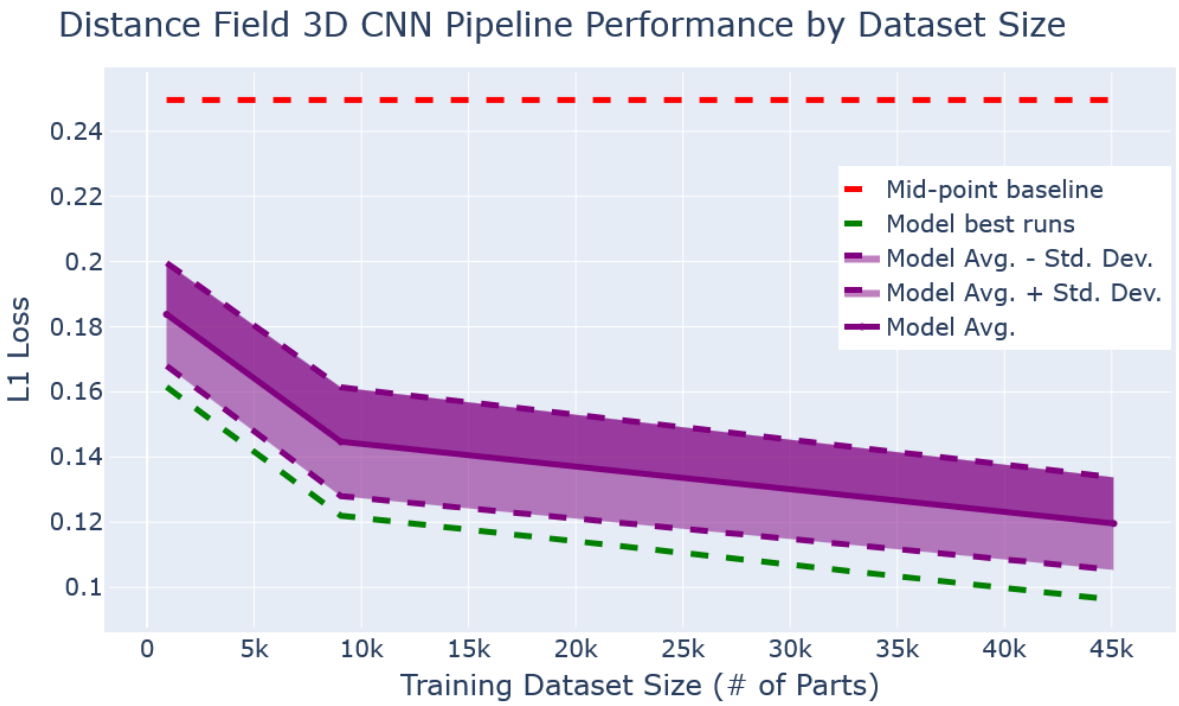


Figure 4.8: Graph showing the Distance Field 3D CNN pipeline's ability to predict additive manufacturing print quality (measured using L1 loss) compared to the mid-point baseline model for 911, 9,041 and 45,091 training datasets.

We tune the following hyperparameters: learning rate (LR), batch size (BS), number of nearest neighbors (k), number of transformer blocks (nb), and transformer embedding dimension (TD). Learning rate is tuned as 0.001 or 0.0001. Batch size is tuned as 4, 16, or 32. K-Nearest Neighbors is 8, 32 or 128. The number of transformer blocks is tuned as 2 or 4. The transformer embedding dimension is 32 or 64. Number of epochs is 20. Non-linear activation function is trained as ReLU, the same implementation used in the Point Transformer architecture [96].

Similar to the 3D CNN pipelines, the Transformer pipeline performs better when it trains on the larger datasets. We ran the point cloud Transformer model on the 180,294 part dataset and it was plateauing. Since computational resources are finite, we do not fully train and tune it on the 180,294 part dataset because we expect the results will be only marginally better, and not affect our conclusions.

Our results suggest that although point cloud Transformers perform well with tasks such as semantic classification and segmentation of point clouds, at the current dataset size and input resolution, they do not perform as well as 3D CNNs for AM part quality problems.

To test if point-cloud resolution could be limiting performance, we increase point cloud resolution, testing point cloud size ranges from 2,048 to 16,384 points. Point cloud Transformer performance actually worsens with increasing point cloud size. This may be because the larger input increases the overall model size and there may not be enough parts to properly train the larger

model. Interestingly, while Transformers that are used in models such as LLaMA and SAM [81] [47] have huge numbers of parameters (13B for the former and 636M for the latter), our hyperparameter search shows that smaller models (around 300K parameters when using lower resolution point cloud inputs) perform better than larger ones (around 1-10M parameters when using higher resolution point cloud inputs) for our task; however, LLaMA is trained on 1.4 trillion tokens and SAM is trained on 11 million images. It is possible that the higher resolution input point cloud Transformer models would perform better if trained on orders of magnitude more data, but our results suggest it is equally possible this architecture is not as well suited for AM applications as 3D CNNs.

Model comparison

We compare four DL pipelines across different dataset sizes and hyperparameters in the two tables below. The DL models are also compared to the baseline model. In both tables, Dataset size refers to the training dataset size. All models are validated on a separate validation dataset of size 10,000. Average and Standard Deviation L1 loss in Table 4.2 and L1 loss in Table 4.3 all represent the validation L1 loss. In Table 3, KS refers to kernel size, LR is learning rate, BS is batch size, and E is epoch. ReLU, or rectified linear unit, is the non-linear activation function used. We test both ReLU and sigmoid for the 64^3 and 256^3 voxel 3D CNN models, but ReLU almost exclusively performs better for both voxel resolutions (better on average, as well as lower standard deviations) and so we exclude sigmoid from the distance field and depth image hyperparameter tuning (Table 2 compares the difference for the 64^3 resolution). For point cloud Transformer tuning, nb is the number of Transformer blocks, TD is the transformer dimension, and k is the number of nearest neighbors checked.

Dataset size	**Input resolution**	**Avg., Std. Dev. L1 Loss**	
		ReLU	**Sigmoid**
45,091	64^3	0.1349, 0.0171	0.1609, 0.0397
9,041	64^3	0.1619, 0.0216	0.2086, 0.0478
911	64^3	0.2113, 0.0215	0.2446, 0.0206

Table 4.1: For Voxel 3D CNN DL pipelines, a comparison of average and standard deviation L1 Losses with using ReLU as the non-linear activation function or using Sigmoid as the activation function for different dataset sizes. ReLU performs almost exclusively better so we only use ReLU when training the depth image and distance field DL pipelines.

We used GTX 2080 Ti graphics cards for training the DL models. Each run uses 1 GPU and 4 Intel Xeon Skylake cores. For the mid-point baseline model, GPU acceleration is insignificant, so we perform this computation on an Intel Xenon CPU.

We summarize our DL models' results over all hyperparameters in Table 4.2. We show the best (measured by lowest validation L1 loss) models in Table 4.3. The training time reported in Table

Table 4.2: Deep learning pipeline & baseline model performance summary for each combination of pipeline, dataset size, and input resolution size.

DL pipeline	Dataset size	Input resolution	Avg. L1 loss	Std. Dev. L1 loss
Voxel - 3D CNN	180,294	64^3	0.1158	0.0148
	45,091	64^3	0.1349	0.0171
	9,041	64^3	0.1619	0.0216
	911	64^3	0.2113	0.0215
	45,091	256^3	0.1185	0.0099
	9,041	256^3	0.1469	0.0177
	911	256^3	0.1936	0.0237
Distance Field - 3D CNN	45,091	64^3	0.1196	0.0143
	9,041	64^3	0.1447	0.0167
	911	64^3	0.1838	0.0158
Depth Images (from top) - 3D CNN	45,091	64^3	0.1372	0.0096
	9,041	64^3	0.1626	0.0076
	911	64^3	0.2011	0.0155
Depth Images (from bottom) - 3D CNN	45,091	64^3	0.1341	0.0104
	9,041	64^3	0.1605	0.0104
	911	64^3	0.2043	0.0146
Point Cloud - Transformer	45,091	(2048, 3)	0.2284	0.0201
	9,041	(2048, 3)	0.2494	0.0323
	911	(2048, 3)	0.2548	0.0044
Mid-point Baseline	N/A	0	0.2496	N/A

4.3 is the time it takes to train the model on that particular combination of hyperparameters and validate the model on the validation dataset. The 64^3 voxel 3D CNN model trained on the 180,294 part dataset and distance field 3D CNN model trained on the 45,091 part dataset have the lowest L1 losses. The distance field 3D CNN model trained on the 9,041 part dataset and the 64^3 voxel 3D CNN model trained on 45,091 part dataset have relatively low L1 losses and take much less time to train.

The first takeaway of the model comparison is that the 3D CNN models are significantly better able to predict UPS than either the point cloud Transformer model or the mid-point baseline model. The best 3D CNN model has 9.62% error compared to 20.11% for the best point cloud Transformer model and 24.96% for the baseline model. The distance field 3D CNN model is the most promising model, although the voxel 3D CNN and depth image 3D CNN are not significantly worse. The distance field shape representation is the most promising because it has the best per-

Table 4.3: Best performing deep learning models (measured by lowest L1 loss) for each pipeline and dataset size

DL pipeline	Dataset size	Input res.	Hyperparameters	L1 loss	Train time
Voxel - 3D CNN	180,294	64^3	ReLU, KS=5, LR=0.001, BS=32	0.0975, E=20	4h 9m
	45,091	64^3	ReLU, KS=5, LR=0.001, BS=32	0.1167, E=20	1h 9m
	9,041	64^3	ReLU, KS=3, LR=0.001, BS=8	0.1383, E=20	19m
	911	64^3	ReLU, KS=5, LR=0.001, BS=4	0.1742, E=20	12m
	45,091	256^3	ReLU, KS=5, LR=0.0001, BS=4	0.1049, E=10	1d 19h 40m
	9,041	256^3	ReLU, KS=5, LR=0.001, BS=4	0.1239, E=20	22h 54m
	911	256^3	ReLU, KS=3, LR=0.0001, BS=4	0.1693, E=20	7h 43m
Distance Fields - 3D CNN	45,091	64^3	ReLU, KS=5, LR=0.001, BS=32	0.0962, E=20	14h 23m
	9,041	64^3	ReLU, KS=5, LR=0.001, BS=32	0.1220, E=20	25m
	911	64^3	ReLU, KS=5, LR=0.001, BS=4	0.1615, E=20	13m
Depth Image (top) - 3D CNN	45,091	64^3	ReLU, KS=5, LR=0.001, BS=32	0.1209, E=20	12h 3m
	9,041	64^3	ReLU, KS=5, LR=0.001, BS=32	0.1527, E=20	27m
	911	64^3	ReLU, KS=3, LR=0.0001, BS=4	0.1830, E=20	16m
Depth Image (bottom) - 3D CNN	45,091	64^3	ReLU, KS=5, LR=0.001, BS=4	0.1203, E=20	14h 52m
	9,041	64^3	ReLU, KS=5, LR=0.001, BS=8	0.1488, E=20	24m
	911	64^3	ReLU, KS=5, LR=0.0001, BS=4	0.1892, E=20	14m
Point Cloud - Transformer	45,091	(2048, 3)	nb=2, TD=64, k=8, LR=1e-4	0.2012, E=20	13h 44m
	9,041	(2048, 3)	nb=2, TD=32, k=8, LR=1e-4	0.2265, E=20	4h 6m
	911	(2048, 3)	nb=4, TD=32, k=8, LR=1e-3	0.2498, E=20	2h 14m
Mid-point Baseline	10,000	0	N/A	0.2496	$<$1m

forming model with 9.62% error and a more than one standard deviation lower L1 loss on average than the corresponding voxel and depth image inputs (better at predicting AM print quality on average). One reason to consider another shape representation is timing. The 64^3 voxel 3D CNN model trained on 45,091 parts takes much less time to train than the 64^3 distance field 3D CNN model trained on 45,091 parts. If training time and resources are severely limited, then the 64^3 voxel shape representation is more promising.

Increasing dataset size significantly improves 3D CNN performance results and decreases hyperparameter sensitivity, mainly because the models overfit less when trained on larger datasets. Increasing data resolution also improves 3D CNN performance, but not at the same rate. For example, an approximately four times increase in dataset size improves model performance about as much as a 64 times increase in data resolution. However, there may be cases were researchers or industry engineers cannot collect more data, in which case increasing data resolution would be a valid second choice.

Chapter 5

Reflection

This thesis presents two complementary approaches for advancing the state of 3D part inspection and representation: Omni-Scan, a system for creating visually-accurate digital twin object models using a bimanual robot with handover and Gaussian Splat merging, and a survey into input part shape representation and deep learning architecture dataset analysis for additive manufacturing part quality predictions. We hope that these methods can make tangible progress toward scalable inspection and quality prediction. During our work, we also noticed some challenges that remain in data acquisition, integration, and generalization.

The Omni-Scan system demonstrates the value of modern vision models by using a general-purpose bimanual robot to autonomously generate visually accurate 3D models. However, issues such as specular reflections, precise alignment between pristine and scanned models, and masking failures can make the system fragile in real-world settings. On the learning side, our investigation into AM print quality prediction revealed promising trends: the model's performance scales with dataset size, and certain representations such as distance fields can rapidly improve even on a smaller dataset. Still, translating these insights to broader settings or applying it to more expensive labels such as complex FEM analysis is challenging.

A central theme across both efforts is the bottleneck of scalable, high-quality physical data. In robotics and manufacturing, acquiring such data continues to require intricate tuning, robust reset strategies, and careful design. One recent work is similar to Omni-Scan, and the system outputs a mesh instead of a Gaussian Splat Model [65]. This work is impressive and seems to address some of the specularity concerns due to the output being a mesh and texture. Additionally, physical parameters such as the center of mass and weight are also learned.

There is exciting future work in the exploration of 3D geometry and understanding. Vision Language Models such as LLaVA [54] [82] project embeddings computed by a vision encoder into a space where this information is fused with the text embeddings produced by previous layers. In the field of foundational models for robotics, recent work has also explored this idea of fusion. In particular, Vision-Language-Action models can take in previous observations and trajectories embedded by Vision Transformers to predict the next motions [35]. Furthermore, early fusion of language and text has been shown to be effective and using CLIP as a joint vision-text encoder could improve model performance on unseen tasks [38]. Interestingly, none of these works has

explicit 3D representation and operates purely on pixels in 2D image space. This is mainly due the impressive performance of current ViT models that mainly work on 2D images and a lack of 3D data. We hope to provide a dataset for this in our work in Chapter 4.

It is possible that with sufficient scaling, 3D understanding for meshes and mechanical design will reach a similarly advanced level as the one for images. VLMs can achieve impressive performance on OCR, a highly specialized task that was thought to be very challenging. However, with much of the compute focused on generality of AI, it is sensible to design systems that are compatible with our current foundational models. Gaussian Splats can output photorealistic reconstruction from novel viewpoints, making it an exciting representation since it would be in distribution where VLMs can perform optimally.

Since Dex-Net, there have been many projects that perform large-scale data-driven grasp networks [56] [31] [32]. From grasping to defect inspection to print-ability prediction, 3D understanding is essential to all these tasks. Demonstrated by the flurry of progress in AI for mathematics such as DeepSeek and Lean-STaR [73] [53], the contemporary paradigm is no longer purely focused on pre-training. In this new paradigm, models learn domain-specific knowledge, potentially also learning to use specialized tools. Engineers can analyze parts with customized software for inspecting the shapes and forces. Perhaps the future of these designs should not be a simple predictive process but instead be an interactive one where the model can also interface with various tools that we have designed in optimal representations.

Beyond the work presented in this thesis, I have explored many other projects that contributed to this broader journey through robotics, vision, and intelligent systems:

- We design a system to plan and assemble structures using a single word as prompt [36]
- We build an algorithm to rapidly plan trajectories to transport boxes [77]
- We make a robot dance like a human dancer [18]
- We build a cloud framework for latency reliability [15]
- We study the task of inserting a deformable gasket into a channel [3]
- We build a model to grasp multiple objects at the same time [5]

I like to think that each project I complete leaves me with a new tool, something I can call upon in the future. And over time, I seem to have built quite a collection.

Bibliography

[1] Henrik Aanæs et al. “Large-Scale Data for Multiple-View Stereopsis”. In: *International Journal of Computer Vision* (2016), pp. 1–16.

[2] Michal Adamkiewicz et al. “Vision-only robot navigation in a neural radiance world”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4606–4613.

[3] Simeon Adebola et al. “Automating Deformable Gasket Assembly”. In: *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. 2024, pp. 4146–4153. DOI: 10.1109/CASE59546.2024.10711836.

[4] Arpit Agarwal et al. “Robotic defect inspection with visual and tactile perception for large-scale components”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 10110–10116.

[5] Wisdom C. Agboh et al. “Learning to Efficiently Plan Robust Frictional Multi-Object Grasps”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 10660–10667. DOI: 10.1109/IROS55552.2023.10341895.

[6] Aditya Balu et al. “A deep learning framework for design and analysis of surgical bioprosthetic heart valves”. In: *Scientific Reports* 9.1 (2019), p. 18560.

[7] Jonathan T Barron et al. “Mip-nerf 360: Unbounded anti-aliased neural radiance fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5470–5479.

[8] Jonathan T Barron et al. “Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5855–5864.

[9] Jonathan T Barron et al. “Zip-nerf: Anti-aliased grid-based neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19697–19705.

[10] Akshay Bharadwaj et al. “Development of a pilot manufacturing cyberinfrastructure with an information rich mechanical CAD 3D model repository”. In: *International Manufacturing Science and Engineering Conference*. Vol. 58745. American Society of Mechanical Engineers. 2019, V001T02A035.

[11] Jannatul Bushra and Hannah D Budinoff. "Orientation Optimization in Additive Manufacturing: Evaluation of Recent Trends". In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 85413. American Society of Mechanical Engineers. 2021, V005T05A003.

[12] Arunkumar Byravan et al. "Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 9362–9369.

[13] Anpei Chen et al. "Tensorf: Tensorial radiance fields". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*. Springer. 2022, pp. 333–350.

[14] Jiayi Chen et al. "Tracking and Reconstructing Hand Object Interactions from Point Cloud Sequences in the Wild". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI. 2023.

[15] Kaiyuan Chen et al. *FogROS2-PLR: Probabilistic Latency-Reliability For Cloud Robotics*. 2024. arXiv: 2410.05562 [cs.RO]. URL: https://arxiv.org/abs/2410.05562.

[16] Pranav Singh Chib and Pravendra Singh. "Recent advancements in end-to-end autonomous driving using deep learning: A survey". In: *IEEE Transactions on Intelligent Vehicles* 9.1 (2023), pp. 103–118.

[17] Hongsuk Choi et al. "HandNeRF: Learning to Reconstruct Hand-Object Interaction Scene from a Single RGB Image". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, to appear.

[18] Catie Cuan et al. *Breathless: An 8-hour Performance Contrasting Human and Robot Expressiveness*. 2024. arXiv: 2411.12361 [cs.RO]. URL: https://arxiv.org/abs/2411.12361.

[19] Omid Davtalab et al. "Automated inspection in robotic additive manufacturing using deep learning for layer deformation detection". In: *Journal of Intelligent Manufacturing* 33.3 (2022), pp. 771–784.

[20] Dawson-Haggerty et al. *trimesh*. 2017-2025. URL: https://trimesh.org/.

[21] Matt Deitke et al. "Objaverse: A universe of annotated 3d objects". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 13142–13153.

[22] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 248–255.

[23] Li Deng. "The MNIST database of handwritten digit images for machine learning research [best of the web]". In: *IEEE signal processing magazine* 29.6 (2012), pp. 141–142.

[24] Gourav Dhore et al. "Exploring 3D printing using Cura: A slicing software". In: *International Journal of Advance Scientific Research & Engineering Trends* 5 (2021), pp. 212–222.

[25] Guoying Dong et al. “A part-scale, feature-based surrogate model for residual stresses in the laser powder bed fusion process”. In: *Journal of Materials Processing Technology* 304 (2022), p. 117541.

[26] Laura Downs et al. “Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2022.

[27] Danny Driess et al. “Reinforcement Learning with Neural Radiance Fields”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[28] Khaled Elgeneidy et al. “Characterising 3D-printed Soft Fin Ray Robotic Fingers with Layer Jamming Capability for Delicate Grasping”. In: *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. 2019, pp. 143–148. DOI: 10.1109/ROBOSOFT.2019.8722715.

[29] Aliakbar Eranpurwala, Seyedeh Elaheh Ghiasian, and Kemper Lewis. “Predicting build orientation of additively manufactured parts with mechanical machining features using deep learning”. In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 84003. American Society of Mechanical Engineers. 2020, V11AT11A023.

[30] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[31] Hao-Shu Fang et al. “GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition(CVPR)*. 2020, pp. 11444–11453.

[32] Hao-Shu Fang et al. “Robust grasping across diverse sensor qualities: The GraspNet-1Billion dataset”. In: *The International Journal of Robotics Research* (2023).

[33] Sara Fridovich-Keil et al. “K-planes: Explicit radiance fields in space, time, and appearance”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12479–12488.

[34] Sara Fridovich-Keil et al. “Plenoxels: Radiance fields without neural networks”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 5501–5510.

[35] Letian Fu et al. *In-Context Imitation Learning via Next-Token Prediction*. 2024. arXiv: 2408.15980 [cs.RO]. URL: https://arxiv.org/abs/2408.15980.

[36] Andrew Goldberg et al. *Blox-Net: Generative Design-for-Robot-Assembly Using VLM Supervision, Physics Simulation, and a Robot with Reset*. 2024. arXiv: 2409.17126 [cs.RO]. URL: https://arxiv.org/abs/2409.17126.

[37] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[38] Huang Huang et al. *OTTER: A Vision-Language-Action Model with Text-Aware Visual Feature Extraction*. 2025. arXiv: 2503.03734 [cs.RO]. URL: https://arxiv.org/abs/2503.03734.

[39] Jeffrey Ichnowski* et al. "Dex-NeRF: Using a Neural Radiance field to Grasp Transparent Objects". In: *Conference on Robot Learning (CoRL)*. 2020.

[40] Inc. Jacobi Robotics. *Jacobi Motion Library – Next Generation Motion Planning*. 2024. URL: https://docs.jacobirobotics.com.

[41] Zeqing Jin et al. "Leveraging graph neural networks and neural operator techniques for high-fidelity mesh-based physics simulations". In: *APL Machine Learning* 1.4 (2023).

[42] Bernhard Kerbl et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". In: *ACM Transactions on Graphics* 42.4 (2023), pp. 1–16. DOI: 10.1145/3592129.

[43] Justin Kerr et al. "Evo-NeRF: Evolving NeRF for Sequential Robot Grasping of Transparent Objects". In: *6th Annual Conference on Robot Learning*. 2022. URL: https://openreview.net/forum?id=Bxr45keYrf.

[44] Justin Kerr et al. "Robot See Robot Do: Imitating Articulated Object Manipulation with Monocular 4D Reconstruction". In: *8th Annual Conference on Robot Learning*. 2024. URL: https://openreview.net/forum?id=2LLu3gavF1.

[45] Aditya Khadilkar, Jun Wang, and Rahul Rai. "Deep learning–based stress prediction for bottom-up SLA 3D printing process". In: *The International Journal of Advanced Manufacturing Technology* 102 (2019), pp. 2555–2569.

[46] Aamir Khan et al. "Vision guided robotic inspection for parts in manufacturing and remanufacturing industry". In: *Journal of Remanufacturing* 11.1 (2021), pp. 49–70.

[47] Alexander Kirillov et al. "Segment anything". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 4015–4026.

[48] Alex Krizhevsky and Geoffrey Hinton. "Learning multiple layers of features from tiny images". In: (2009). URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[49] Sachin Kumar et al. "Machine learning techniques in additive manufacturing: a state of the art review on design, processes and production control". In: *Journal of Intelligent Manufacturing* 34.1 (2023), pp. 21–55.

[50] Yunzhu Li et al. "3d neural scene representations for visuomotor control". In: *Conference on Robot Learning*. PMLR. 2022, pp. 112–123.

[51] Zhengqi Li et al. "Dynibar: Neural dynamic image-based rendering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4273–4284.

[52] Liang Liang et al. "A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis". In: *Journal of The Royal Society Interface* 15.138 (2018), p. 20170844.

[53] Haohan Lin et al. *Lean-STaR: Learning to Interleave Thinking and Proving*. 2025. arXiv: 2407.10040 [cs.AI]. URL: https://arxiv.org/abs/2407.10040.

[54] Haotian Liu et al. *Visual Instruction Tuning*. 2023. arXiv: 2304.08485 [cs.CV]. URL: https://arxiv.org/abs/2304.08485.

[55] Li Ma et al. "Deblur-nerf: Neural radiance fields from blurry images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12861–12870.

[56] Jeffrey Mahler et al. "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics". In: *CoRR* abs/1703.09312 (2017). arXiv: 1703.09312. URL: http://arxiv.org/abs/1703.09312.

[57] Manoj Malviya and K Desai. "Build orientation optimization for strength enhancement of FDM parts using machine learning based algorithm". In: *Computer-Aided Design and Applications* 17 (2019), pp. 783–796.

[58] Ben Mildenhall et al. "NeRF: Representing scenes as neural radiance fields for view synthesis". In: *European Conference on Computer Vision (ECCV)*. 2020.

[59] Patrick Min. *binvox*. Accessed: 2022-09-14. 2004 - 2025. URL: http://www.patrickmin.com/binvox.

[60] Thomas Müller et al. "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM Transactions on Graphics (ToG)* 41.4 (2022), pp. 1–15.

[61] Maryam M Najafabadi et al. "Deep learning applications and challenges in big data analytics". In: *Journal of Big Data* 2.1 (2015), pp. 1–21.

[62] Zhenguo Nie, Haoliang Jiang, and Levent Burak Kara. "Stress field prediction in cantilevered structures using convolutional neural networks". In: *Journal of Computing and Information Science in Engineering* 20.1 (2020), p. 011002.

[63] Fakir S. Nooruddin and Greg Turk. "Simplification and Repair of Polygonal Models Using Volumetric Techniques". In: *IEEE Transactions on Visualization and Computer Graphics* 9.2 (2003), pp. 191–205.

[64] Keunhong Park et al. "HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields". In: *ACM Trans. Graph.* 40.6 (Dec. 2021).

[65] Nicholas Pfaff et al. *Scalable Real2Sim: Physics-Aware Asset Generation Via Robotic Pick-and-Place Setups*. 2025. arXiv: 2503.00370 [cs.RO]. URL: https://arxiv.org/abs/2503.00370.

[66] Albert Pumarola et al. "D-NeRF: Neural Radiance Fields for Dynamic Scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

[67] Adam Rashid et al. "Language Embedded Radiance Fields for Zero-Shot Task-Oriented Grasping". In: *Conference on Robot Learning*. 2023. URL: https://api.semanticscholar.org/CorpusID:261882332.

[68] Nikhila Ravi et al. *SAM 2: Segment Anything in Images and Videos*. 2024. arXiv: 2408.00714 [cs.CV]. URL: https://arxiv.org/abs/2408.00714.

[69] Antoni Rosinol, John J Leonard, and Luca Carlone. "Nerf-slam: Real-time dense monocular slam with neural radiance fields". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 3437–3444.

[70] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. "Fast Point Feature Histograms (FPFH) for 3D registration". In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[71] Christoph Schranz. "Tweaker-auto rotation module for FDM 3D printing". In: *Salzburg Research Salzburg, Austria* (2016).

[72] Krishna Shankar et al. "A learned stereo depth system for robotic manipulation in homes". In: *IEEE Robotics and Automation Letters* 7.2 (2022).

[73] Zhihong Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. 2024. arXiv: 2402.03300 [cs.CL]. URL: https://arxiv.org/abs/2402.03300.

[74] William Shen et al. "Distilled Feature Fields Enable Few-Shot Language-Guided Manipulation". In: *7th Annual Conference on Robot Learning*. 2023.

[75] Edgar Sucar et al. "iMAP: Implicit Mapping and Positioning in Real-Time". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 6229–6238.

[76] Martin Sundermeyer et al. "Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021.

[77] Zachary Tam et al. "BOMP: Bin-Optimized Motion Planning". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 11056–11063. DOI: 10.1109/IROS58592.2024.10801312.

[78] Matthew Tancik et al. "Nerfstudio: A Modular Framework for Neural Radiance Field Development". In: *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH '23. 2023.

[79] Matthew Tancik et al. "Nerfstudio: A modular framework for neural radiance field development". In: *ACM SIGGRAPH 2023 conference proceedings*. 2023, pp. 1–12.

[80] Zachary Teed and Jia Deng. "Raft: Recurrent all-pairs field transforms for optical flow". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer. 2020, pp. 402–419.

[81] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[82] Pavan Kumar Anasosalu Vasu et al. *FastVLM: Efficient Vision Encoding for Vision Language Models*. 2024. arXiv: 2412.13303 [cs.CV]. URL: https://arxiv.org/abs/2412.13303.

[83] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[84] Chengcheng Wang et al. “Machine learning in additive manufacturing: State-of-the-art and perspectives”. In: *Additive Manufacturing* 36 (2020), p. 101538.

[85] Jun Wang et al. “Data-driven simulation for fast prediction of pull-up process in bottom-up stereo-lithography”. In: *Computer-Aided Design* 99 (2018), pp. 29–42.

[86] Peng Wang et al. “F2-NeRF: Fast Neural Radiance Field Training with Free Camera Trajectories”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4150–4159.

[87] Risheng Wang et al. “Medical image segmentation using deep learning: A survey”. In: *IET Image Processing* 16.5 (2022), pp. 1243–1267.

[88] Bowen Wen et al. “BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2023.

[89] Bowen Wen et al. “Foundationpose: Unified 6d pose estimation and tracking of novel objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 17868–17879.

[90] Glen Williams et al. “Design repository effectiveness for 3D convolutional neural networks: Application to additive manufacturing”. In: *Journal of Mechanical Design* 141.11 (2019), p. 111701.

[91] T. Wohlers and ASTM International. *Wohlers Report 2024: 3D Printing and Additive Manufacturing : Global State of the Industry*. Wohlers Associates, 2024. URL: https://books.google.com/books?id=0j7i0AEACAAJ.

[92] Jian Cheng Wong et al. “Graph Neural Network Based Surrogate Model of Physics Simulations for Geometry Design”. In: *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2022, pp. 1469–1475.

[93] H. Yang, J. Shi, and L. Carlone. “TEASER: Fast and Certifiable Point Cloud Registration”. In: *IEEE Trans. Robotics* (2020).

[94] Lihe Yang et al. “Depth Anything V2”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 21875–21911. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/26cfdcd8fe6fd75cc53e92963a656c58-Paper-Conference.pdf.

[95] Vickie Ye et al. “gsplat: An Open-Source Library for Gaussian Splatting”. In: *arXiv preprint arXiv:2409.06765* (2024). arXiv: 2409.06765 [cs.CV]. URL: https://arxiv.org/abs/2409.06765.

[96] Hengshuang Zhao et al. “Point Transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 16259–16268.

[97] Licheng Zhong et al. “Color-NeuS: Reconstructing neural implicit surfaces with color”. In: *2024 International Conference on 3D Vision (3DV)*. IEEE. 2024, pp. 631–640.

[98] Zihan Zhu et al. “NICE-SLAM: Neural Implicit Scalable Encoding for SLAM”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 12786–12796.