3D Object Representations for Robot Perception

by

Benjamin C. M. Burchfiel

Department of Computer Science
Duke University

Date: _____

Approved:

_____
George Konidaris, Supervisor

_____
Carlo Tomasi, Chair

_____
Katherine Heller

_____
Stefanie Tellex

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2019

Abstract

3D Object Representations for Robot Perception

by

Benjamin C. M. Burchfiel

Department of Computer Science
Duke University

Date: _____

Approved:

_____
George Konidaris, Supervisor

_____
Carlo Tomasi, Chair

_____
Katherine Heller

_____
Stefanie Tellex

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University
2019

Abstract

Reasoning about 3D objects is one of the most critical perception problems robots face; outside of navigation, most interactions between a robot and its environment are object-centric. Object-centric robot perception has long relied on maintaining an explicit database of 3D object models with the assumption that encountered objects will be exact copies of entries in the database; however, as robots move into unstructured environments such as human homes, the variation of encountered objects increases and maintaining an explicit object database becomes infeasible. This thesis introduces a general-purpose 3D object representation that allows the joint estimation of a previously unencountered object's class, pose, and 3D shape—crucial foundational tasks for general robot perception.

We present the first method capable of performing all three of these tasks simultaneously, Bayesian Eigenobjects (BEOs), and show that it outperforms competing approaches which estimate only object shape and class given a known object pose. BEOs use an approximate Bayesian version of Principal Component Analysis to learn an explicit low-dimensional subspace containing the 3D shapes of objects of interest, which allows for efficient shape inference at high object resolutions. We then extend BEOs to produce Hybrid Bayesian Eigenobjects (HBEOs), a fusion of linear subspace methods with modern convolutional network approaches, enabling realtime inference from a single depth image. Because HBEOs use a convolutional network to project partially observed objects onto the learned subspace, they allow the object to be larger and more expressive without impacting the inductive power of the model. Experimentally, we show that HBEOs offer signifi-

cantly improved performance on all tasks compared to their BEO predecessors. Finally, we leverage the explicit 3D shape estimate produced by BEOs to further extend the state-of-the-art in category level pose estimation by fusing probabilistic pose predictions with a silhouette-based reconstruction prior. We also illustrate the advantages of combining both probabilistic pose estimation and shape verification, via an ablation study, and show that both portions of the system contribute to its performance. Taken together, these methods comprise a significant step towards creating a general-purpose 3D perceptual foundation for robotics systems, upon which problem-specific systems may be built.

To my wife Eileen,

without you nothing would be possible, because of you everything is within reach.

Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

2D Two Dimensional

3D Three Dimensional

AI Artificial Intelligence

AMT Amazon Mechanical Turk

BEOs Bayesian Eigenobjects

BPCA Bayesian Principal Component Analysis

CNN Convolutional Neural Network

CPU Central Processing Unit

DF Distance Field

DNN Deep Neural Network

DOF Degrees of Freedom

EDT Euclidean Distance Transform

ELU Exponential Linear Unit

EM Expectation-Maximization

GAN Generative Adversarial Network

GMM Gaussian Mixture Model

GPS Global Positioning System

GPU Graphics Processing Unit

HBEO Hybrid Bayesian Eigenobjects

ICP Iterative Closest Points

LSTM Long Short-Term Memory

MAP Maximum a Posteriori Estimation

MDN Mixture Density Network

MLE Maximum Likelihood Estimation

MV Multivariate

PCA Principal Component Analysis

PDF Probability Density Function

PPCA Probabilistic Principal Component Analysis

RAM Random Access Memory

RCNN Region-Based Convolutional Neural Network

RGB Red, Green, and Blue Image

RGBD Red, Green, Blue, and Depth Image

ROS Robot Operating System

SDF Signed Distance Field

SO Special Orthogonal Group

TSDF Truncated Signed Distance Field

USB Universal Serial Bus

VAE Variational Autoencoder

VBPCA Variational Bayesian Principal Component Analysis

Acknowledgements

Thank you to my fantastic advisor George; the most supportive mentor a young researcher could ask for. George's emphasis on curiosity, scientific rigour, and chasing big problems has helped shape me into the scientist I now am. His ingenuity, insight, and thoughtfulness has been an inspiration. George is quick to celebrate the success of others, and patient when ideas fail; he is a truly wonderful scientist to work beside and continuously inspires the best in his students.

To Carlo and Ron, thank you both for being my mentors. You taught me what it means to be a researcher and you ignited my interest in robotics. If it wasn't for the patience, kindness, and attention you both afforded me, I would not be where I am today.

I owe a debt of gratitude to my wonderful committee, Carlo, Stefanie, and Katherine. Thank you for your comments, invaluable feedback, and support.

Thank you to the wonderful members of the IRL lab, both at Duke University and at Brown: Barrett, Cam, Lauren, Ben Matt, Yuu, Andrew, Branka, and Garrett. Thank you as well to my other wonderful collages and collaborators, Ergys, Cassi, Jason, Nakul, Vanya, Eddie, and Thao. I benefited hugely from their insights, discussions, and keen intellects.

Thank you to the wonderful computer science department at Duke University; I count myself extremely fortunate to be surrounded by such amazing people.

Thank you to my wonderful and supportive parents, including Anne, you have been pillars of advice, love, and support. Thank you as well to my little brother, Royden, for being wonderful. Thank you to Eileen, you are my one.

This research was supported in part by DARPA under agreement number D15AP00104. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The content is solely the responsibility of the authors and does not necessarily represent the official views of DARPA.

1

Introduction

The recent proliferation of generally capable and relatively inexpensive mobile manipulation platforms represents a golden opportunity to finally realize the promise of robotics: reliable and versatile robots operating autonomously in our homes, offices, hospitals, and public spaces. These general purpose robots—long a dream of robotics researchers—must be able to operate in unstructured environments that were not explicitly designed with robots in mind. Common examples of desired functionality include robots that can make tea, robots that can fetch a snack from the corner store, robots that can prepare a meal, and robots that can organize a home. Many hurdles exist which must be overcome to realize this vision: hardware (and energy sources) must develop further in capability and come down in price, planning algorithms must progress to allow nuanced and provably-safe operation in complex environments, learning algorithms must advance to allow knowledge to better generalize from previously completed tasks to new ones, and human-robot communication must improve to allow robots to better take instructions from, and provide transparent feedback to, non-domain-expert humans.

One of the most critical of these hurdles is perception, the conversion of sensor input into useful representations of the robot's environment. Perception generally consists of

amalgamating data from RGB cameras, infrared or laser based depth sensors, and potentially more exotic modalities such as GPS signals, haptic sensors, temperature sensors, microphones, and ultrasonic sensors. Because modern robots operate using a *software stack* model, with multiple encapsulated modules deriving their input from the output of a preceding module and providing input to modules further up in the stack, reliable and robust perception—which is one of the lowest modules in the stack—is critical for the success of the entire system; a robot with perfect planning algorithms will still fail catastrophically if given a poor state-representation based on faulty perception. Without significant advances in perception, robots will never be able to operate in unstructured environments, regardless of the advances made in other areas of research.

1.1 Object-Centric Perception

A key differentiator of robotics is the ability for systems to affect the physical world; unlike other intelligent agents, robots taking actions physically interact with their surroundings. In indoor environments, most of this interaction will occur with *objects*: robots must determine when and how to grasp, push, avoid, and manipulate nearby objects to accomplish their goals. A useful general purpose object-centric perception system must thus support these interactions in real-world settings. Specifically, several key attributes are necessary for such a perception system.

Applicability to Novel Objects

If robots are to operate in human-centric environments, their perceptual systems must accommodate the huge object variation that exists in those settings. As of April 2019, Amazon.com sells over 40 million distinct objects, almost all of which could reasonably be encountered in a household setting. It is therefore unreasonable to assume that objects in a robot's environment have been previously encountered, regardless of whether the particularities require a known 3D model, 2D silhouette, or that the object is a member of

a previously trained-upon dataset. Instead, robot perception must assume that encountered objects are all previously unseen and rely upon generalizing from previously observed objects that are similar, but not identical.

Applicability to Partially Observed Objects

Robots, by virtue of being mobile and able to effect change in their surroundings, have the unique advantage of *active sensing*—the ability to influence their sensor inputs by taking actions that change their environment or their position in that environment. Nevertheless, in realistic scenarios, it is infeasible to expect that a robotic agent fully observe all objects it may wish to reason about or interact with. It is also unrealistic to expect a robot to traverse its entire environment, manipulating each object of interest to observe it from all angles. Instead, object-centric perceptual systems must operate on partially observed input; robots obtain sensor information from a single snapshot in time (such as a single RGB or depth image) and are required to reason based on that limited input. Ideally, additional information could be amalgamated over time, fusing multiple sensor-snapshots obtained from multiple spatial and temporal locations.

Applicability to Cluttered Environments

Clutter occurs when multiple objects are present in close vicinity to each other, often physically making contact; in many household areas, such as inside cabinets and drawers, object clutter is the norm. Clutter is a particularly challenging scenario because objects may have significant occlusion with arbitrary portions being observable; in adversarial instances, reasoning about objects in clutter may *require* active sensing—imagine attempting to reason about an object with only a tiny fraction visible—to displace occluding objects. Clutter is also closely related to the task of segmentation: given an environment containing multiple objects, determine which portions of the environment belong to individual object instances.

Classification of Objects by Semantic Type

Object classification, long a mainstay of computer vision in the 2D RGB setting, is critical for many robotic tasks. It is necessary for language-based human robot interaction as humans tend to rely heavily on semantic labels when communicating requests, for example: "Bring me the television remote". Without the ability to ground observed objects to class-labels, these sorts of interactions are infeasible. Semantic class also provides a natural distillation for many useful affordances such as "cups hold liquid", "screwdrivers turn screws", and "light-switches control lights". Furthermore, some notions such as pose only make sense when conditioned on a particular class. It makes little sense to say that a generic object is oriented upwards, but it is unambiguous to say that a cup is upright.

Determination of 3D Pose

Category-level pose estimation—the task of determining the orientation of a previously unseen object, relative to a canonical pose—is useful for many common tasks. For instance, a robot that sets a table may wish to place items upright, a robot using a remote control may require it be oriented at the device it controls, and a robot taking instructions from a human may need to decipher explicit pose commands. As alluded to previously, pose estimation of novel objects relies on an explicit notion of category, to ensure that "canonical pose" is well defined, in contrast to the more common task of instance-based pose estimation, which seeks to find an alignment between a known object model and an observation of that object in the world. While full six degree of pose estimation, predicting an object's translation and orientation relative to the robot, is necessary for robust perception, in the robot-specific setting—where robots are virtually always equipped with a depth-sensor—estimating the translation of an arbitrary object is straightforward given an estimate of its 3D shape and three degree of freedom orientation.

Determination of 3D Shape

3D object shape is directly applicable to many robotic planning and manipulation tasks including grasping objects, avoiding objects, estimating physically-grounded object affordances, understanding and generating natural-language descriptions of objects, and storing objects (packing). Critically, the ability to estimate the 3D shape of novel and partially observed objects allows a robot to perform shape-dependent tasks without exhaustively observing all of the objects it may interact with. For example, given the task of "fetch the pot with a long curved handle", a robot without the ability to estimate 3D shape would be required to observe all possible pots in its environment from many viewpoints before being confident it had not missed a pot satisfying the query. Additionally, if the 3D shape estimate is viewpoint invariant (conditioned on object class), the problem space for many vision tasks becomes dramatically smaller—for example a 3D pot-handle detector need not be trained across all possible viewpoints as in the 2D case.

Confidence Measures and Probabilistic Output

Robots, with their multi-stage reasoning pipelines and ability to take actions that affect the physical world, have unique advantages and challenges. Because gross failures in the perception portion of the reasoning pipeline tend to be catastrophic to later stages (such as a planning module) when naively trusted, prediction confidence measures are especially important in the robot setting. Given such measures, filtering approaches—such as Bayesian filters (Chen, 2003)—can dramatically improve belief estimates and overall robot reliability by allowing a robot to act conservatively in the face of high uncertainty. Additionally, if perceptual confidence is sufficiently poor, robots possess the unique ability to take action to actively influencing their future sensor returns (for example, obtaining a new view of a particular object).

Indeed, imperfect or biased predictions are still useful in robotics; noisy solutions are

useful because they provide valuable hypotheses about that world which may be verified and refined through additional interaction. Humans, for instance, commonly search for objects in their environment and examine multiple items, rejecting several, before finding one that is suitable. While a human's ability to infer object properties is not perfect, it is good enough that people must generally consider only a very small number of objects before finding one with the characteristics they seek. To enable these sorts of behaviors in robots, it is thus critically important that robot perception be probabilistic, providing not just point-estimates but full probability distributions.

Extensibility

Current robot hardware is far more versatile than the software powering it. While a modern mobile manipulation platform (e.g. the Kinova Movo) is physically capable of diverse household tasks such as setting the table, washing dishes, and cooking dinner, our ability to perform these tasks in arbitrary environments is hampered by software. The state-of-the-art in robotics may be able to perform one of these tasks in a specific environment, but would not generalize to a novel setting with new layouts and objects, to say nothing of performing a new task that had not been explicitly considered by the designers. Currently, if a robot's operating environment—or task—changes, its perception pipeline must generally be retrained from scratch. A first step in overcoming this state of affairs is *extensible* perception, with foundational modules designed to be general and problem-specific perception built on top of the foundational portions. This approach allows much of the perception pipeline to be retained when a domain or task changes, resulting in less data and fewer computational resources being required to adapt a robot to a new setting. In the context of object-centric perception, object type, pose, and 3D shape are intrinsically foundational, allowing task-specific perception modules to be constructed from these building blocks. For instance, a grasping predictor might use 3D shape and pose estimates, an affordance estimator might use 3D shape and object category, and a

system that disambiguates between objects based on natural language descriptions might use all three. In this way, a large number of specific perceptual capabilities can be created from a low-level pretrained object representation—with limited data and without requiring retraining of the entire perception system.

Efficiency

Robots, operating in and amongst humans, must be reactive: they ideally need to respond to commands and changes in their environment at least as fast as their human compatriots do. Practically speaking, routine decision making should be realtime, meaning that in nominal circumstances the time required for a robot to accomplish a task should be dominated by the physical constraints of the system, not by processing time. A robot able to fetch a drink only after several minutes of computation is far less desirable than one that begins retrieving the beverage immediately. This has several implications for robot perception, the first being general efficiency. If a method requires many seconds of computation when running on on-board hardware, it is likely not useful in most circumstances. The second implication is that it may be permissible to take a long time to solve "hard" problems if the nominal case is fast. In an object-representation setting, highly unusual objects might require additional computation to process, relative to more typical items. As a result, any-time (or variable-time) algorithms become highly valuable in robot settings; inexact estimates of object properties often suffice for a given task and it is a waste, computationally, to refine estimates more than required. If a robot is instructed to hand a human an upright cup, it may not necessary to estimate pose to within a single degree of resolution, a relatively coarse pose estimate would suffice. Conversely, if a robot is packing a suitcase, it may indeed need to carefully reason about precise object positions and orientations. Variable time algorithms are a perfect fit for these scenarios, allowing initially rough estimates to be refined when needed and when the available computational budget allows.

1.2 The Bayesian Eigenobject Framework for 3D Object Representation

This thesis presents a novel unified framework for representing 3D objects called Bayesian Eigenobjects (BEOs). BEOs learn, from a collection of 3D training meshes, a low-dimensional linear subspace containing the 3D shapes of objects of interest. Our work uses Variational Bayesian Principal Component Analysis (Bishop, 1999b) as the basis for a multi-class object representation; by learning a compact basis for each class, we are able to store previously encountered objects efficiently by retaining only their projection coefficients. Furthermore, novel objects can be localized, classified, and completed by projecting them onto class basis and then projecting back into object space. Because we do not query a database of individual objects, our method scales gracefully with the number of objects in each class, requiring a constant amount of computation for projection and reconstruction even as the number of previously encountered data-points increases, and can accommodate objects of higher resolution than competing methods.

The BEO framework produces a pose-invariant shape representation, a disentangled three degree of freedom pose estimate, and a category prediction for partially observed and novel objects. The current version of our method, HBEO-MDNs, produces a single 3D shape estimate and a distribution over possible classes and object poses—from input consisting of a single segmented depth-image—in realtime. HBEO-MDNs retain the linear subspace-based object representation used by BEOs but replace explicit projection onto that space with a learned convolutional prediction. This decoupling of subspace generation and projection allows for nuanced nonlinear reasoning leveraging complex learned features, provides shape, pose, and class prediction directly from depth images, and, as the subspace no longer directly constrains the projection estimate, allows the subspace to be loose, increasing the expressive power of the method. HBEO-MDNs also provide variable-time category-level pose estimation, producing increasingly accurate pose estimates with larger time budgets.

1.2.1 Organization and Contributions

This document is organized as follows:

- (Chapter 2) We discuss general background in the area of object-centric perception including common 3D object representations and existing approaches to classification, category-level pose estimation, and 3D completion of partially observed objects.

- (Chapter 3) We construct a novel compact representation for 3D objects, BEOs, that disentangles 3D shape, pose, and class. BEOs learn an explicit low-dimensional space representing 3D objects of interest and perform inference in this space. Specifically, we propose employing a variational approximation to Principal Component Analysis to provide regularization to the learned object-subspace and allowing it to better generalize to novel objects, even in the case of few training examples.

- (Chapter 3) We provide a method for applying BEOs to partially observed 3D voxel objects, allowing shape, pose, and class to be estimated jointly given a novel and partially observed query object. This method explicitly minimizes L_2 voxel-space error between a partially observed object and the corresponding portion of its 3D shape estimate.

- (Chapter 4) We extend BEOs to allow prediction given only a single depth-image. This extension also produces a system capable of running in realtime and dramatically improves the quality of the class, pose, and shape predictions. Our BEO extension, Hybrid Bayesian Eigenobjects (HBEOs) replaces the linear subspace-projection step of BEOs with a convolutional network and achieved state-of-the-art pose estimation and 3D completion from depth performance at the time of its publication. HBEOs are extremely fast, performing inference at over $60hz$ on a single consumer GPU.

- (Chapter 5) We propose an improved category-level pose estimation process that leverages the unique capabilities of joint pose and 3D shape estimation. This system, HBEO-MDN, predicts a distribution over poses, instead of a single pose, and uses agreement between observed input and predicted 3D shape and pose to construct an efficient pose-prior. Because this prior is computed in 2D input-image space it is lightweight to compute and the resulting variable time system is still able to run in realtime and further improves pose-estimation performance relative to pure HBEOs. HBEO-MDNs are currently the state-of-the-art method for category-level 3DOF pose estimation. We show via an ablation analysis that both of these contributions, multi-modal distribution prediction and shape-pose consistency priors, contribute to the performance of HBEO-MDNs.

- (Chapter 6) We provide a discussion of the progress made in recent years on robot object-centric perception as well as the challenges that must still be overcome to realize truly general-purpose robot perception, including the applicability of the HBEO framework to other areas of robot perception. As an initial example, we demonstrate the suitability of the learned pose-invariant shape representation to serve as an extensible building block for robotic tasks by incorporating it in a language grounding system which disambiguates between multiple objects based on natural language descriptions. We were able to train the language portion of this system from a small amount of language data from only a single viewpoint and generalize to novel views without additional training and with almost no performance loss compared to a system trained more conventionally from a variety of object views.

2

Background

Object-centric perception is a broad field, spanning 3D object representation, classification, pose-estimation, and segmentation. This area has advanced rapidly over the past decade, largely due to advances in deep learning, inexpensive depth sensors such as the Microsoft Kinect, and the rapid advance of affordable computational hardware, particularly GPUs.

This chapter discusses the individual components of 3D object-centric perception in more detail as well as relevant related work. We first introduce common 3D object representations before discussing relevant object-centeric perception tasks and integrated approaches capable of jointly solving these tasks.

2.1 3D Object Representations

Multiple representations have been proposed to model 3D objects, each with inherent advantages and disadvantages. We provide a brief overview of the most common and use each to visualize the Utah Teapot (Crow, 1987).[1]

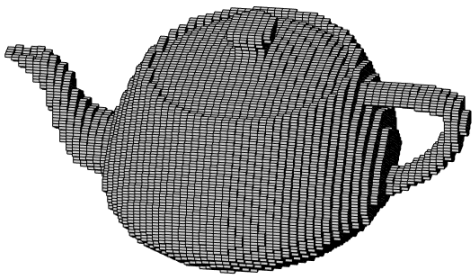


FIGURE 2.1: The Utah Teapot represented using voxels.

2.1.1 Voxels

Voxel representations extend the notion of a 2D image, containing pixels, to three dimensions by modeling an object as a volumetric image defined on a rectangular 3D grid of voxels. In the same way that an image pixel can be identified by its row and column, a voxel can be identified by its 3D index (Kaufman, 1994). Similar to an image pixel, each voxel may store a real number or, if the voxel model is multi-channel, a vector, perhaps describing the voxel's color. One of the most common varieties of voxel representation is the *occupancy grid*, which restricts voxel elements to binary values. One difficulty endemic to voxel representations is rotation; because voxel coordinate systems and cells are rectangular, rotation (in non-90° increments) requires interpolation. As a result, repeated rotations of a voxel model will degrade the object and it is generally better to convert a voxel model into a more rotation-friendly representation (such as a dense point cloud) prior to rotation.

Voxel representations are dense, explicit, and have a fixed size. Because they are

[1] Except in the case of the parametric representation where we illustrate a parametric 3D sphere.

dense, voxels unambiguously represent interiors of objects, allowing hollow items to be distinguished from solid ones. However, this density is a double-edged sword; voxel representations are often large, using significant amounts of storage to represent homogeneous regions of occupied or unoccupied space. Some hierarchical voxel representations, such as octrees (Laine and Karras, 2010), exist however the resulting representation is no longer a fixed size. Voxels are explicit in the sense that they straightforwardly allow intersection and occupancy queries with little additional computation required. Determining if a location in space contains a portion of the object is trivial—simply index into the correct voxel location and evaluate that element. In robot settings, this is quite valuable since collision checking—a critical component of trajectory planning and object manipulation—is one of the most commonly performed planning operations (Laumond et al., 1998). Voxel-based occupancy grids are thus one of the most commonly used representations in robotics. Finally, while modern machine learning has developed techniques capable of performing inference on variable-sized data, particularly in the Natural Language Processing domain with the advent of Recurrent Networks (Mikolov et al., 2010), it is generally more straightforward to operate on fixed-size representations. The majority of inference techniques assume fixed-size input, making voxel representations an appealing prediction target for approaches generating 3D shapes.

2.1.2 Distance Fields

Distance fields (DFs) represent an object using the function $f(x) = d(x, \partial o)$ where o is the set of all points in—or on the surface of—the object and $d(x, \partial o)$ denotes the distance from 3D point x to the nearest point on the surface of o. This representation is surface-based, defining an object by proximity to the closest object-boundary point. Signed distance fields (SDFs) are a slight modification of distance fields which preserve the sign of the distance,

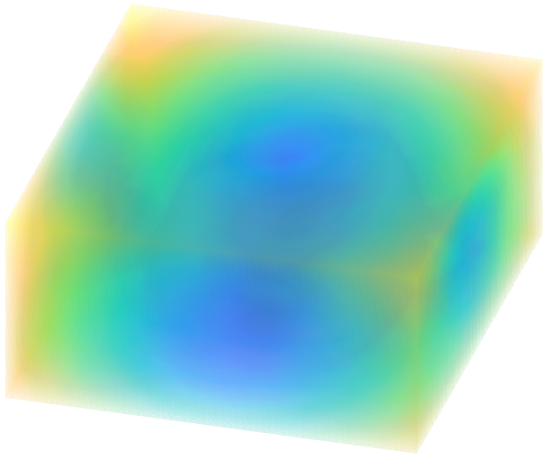


FIGURE 2.2: The Utah Teapot as a signed distance function.

explicitly denoting interior and exterior points:

$$f(x) = \begin{cases} max(-d(x, \partial o), \alpha) & \text{if } x \in o \\ min(d(x, \partial o), \beta) & \text{otherwise.} \end{cases}$$

The signed formulation is useful because it explicitly represents volume; any point such that $f(x) \leq 0$ is a part of the object. If $\alpha > -\infty$ or $\beta < \infty$ the representation becomes a Truncated Signed Distance Field (TSDF), where values larger than β and smaller than α are clipped. Generally, a closed form representation of $f(x)$ is difficult to obtain and a sampling approach is used instead: given a rectangular 3D grid, similar to a voxel representation, each cell's distance to the object's surface is evaluated and the resulting (signed) distance is stored in that cell. The result is a special case of a generic voxel approach, instead of storing a Boolean value indicating occupancy, distance to the object's surface is stored. Such a distance field may be trivially converted into a binary voxel representation by setting all positive values in the field to $False$ and all 0 or negative values $True$. Distance fields are especially appealing when comparing noisy object models, or object models with noisy alignments, due to their spatial smoothness. In a binary voxel representation,

slightly misaligned identical objects may seem significantly different, given an element-wise comparison, if they contain thin structures. A DF, or (T)SDF approach reduces this effect because small translations result in a small perturbations of distance.

2.1.3 Surface Meshes

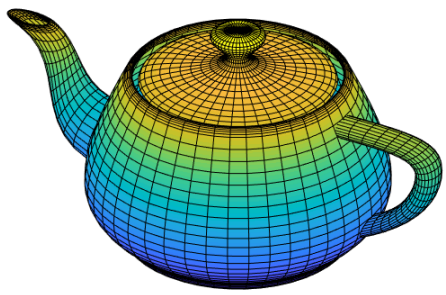


FIGURE 2.3: The Utah Teapot in mesh form.

A surface mesh, or polygon mesh, represents an object by modeling its surface as a collection of polygons organized via a graph structure which specifies a topological relationship between points. The most typical mesh formulation consists of triangles induced via vertices and edges in the graph, where edges correspond to triangle sides and vertices in the graph are triangle vertices. Mesh variants employing other types of polygons also exist and some mesh representations utilize multiple types of polygons to represent a single shape. It should be noted that mesh representations are generally not fixed size as it is common for objects to use varying numbers of polygons depending on their geometric complexity. Surface meshes are also irregular: large numbers of small polygons may be used in areas of high-frequency detail and more homogeneous surface portions may be represented with a relatively small number of larger polygons. Because

they represent object surfaces, and not volumes, meshes tend to be compact and lightweight relative to volumetric representations at the cost of being unable to distinguish between filled and hollow object interiors. Surface mesh models are well-suited to transformations because simply multiplying vertex coordinates by a rotation and translation matrix results in a transformed model with no quality degradation. Resizing a mesh model is also straightforward; the model is centered by subtracting the mean vertex coordinate, $\boldsymbol{\mu}$, from all vertices, each vertex coordinate is then multiplied by the desired scale factor, and the mesh is translated back to the proper location via the addition of $\boldsymbol{\mu}$ to every vertex.

2.1.4 3D Pointclouds

3D pointclouds represent an object as a collection of unordered and unstructured points, $\mathbf{p} = \{x, y, z\}$, in 3D space. Point cloud representations are commonly constructed by projecting each depth-pixel in a range image into 3D space. One of the primary advantages of the pointcloud representation is its flexibility; it is straightforward to project points from multiple sensors at various spatiotemporal locations as long as the transform between each of the sensing locations is known. Pointclouds also have variable density and do not constrain the size of the world-space or require it be known *a priori*. Voxel representations can be converted to dense pointclouds by sampling a point at the center of every voxel while mesh representations can be converted to a surface pointcloud representation by including vertices as points and sampling additional points from the faces. A drawback of pointclouds is their lack of inter-point structure: there is no way to express that two points in a pointcloud are physically connected. As a result, occupied regions of an object must be densely populated with points and ambiguities can still arise when an object has small openings in its structure relative to the density of sampled points. Like meshes, pointcloud representations are straightforward to rotate and translate; however, resizing pointclouds can require point interpolation to preserve density.

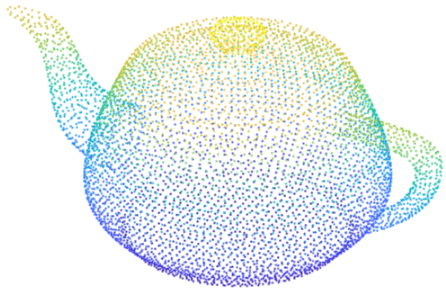


FIGURE 2.4: The Utah Teapot represented as a 3D pointcloud.

2.1.5 Parametric Models

Parameterized surface models directly represent an object via its surface—defined by a paramaterized function $f(\theta)$. Parametric models have the advantage of being extremely compact and highly precise. Because they are not discrete, parametric models are capable of infinite resolution and similarity transformations do not corrupt fine details of the model. Parametric models are not always straightforward to construct; for complex objects such as human bodies (Cheng et al., 2018), they are typically created manually, making them somewhat cumbersome to use as a general-purpose object model. For instance, a 3D sphere with radius r centered at $\{x_0, y_0, z_0\}$ may be defined as the set of all points satisfying

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2$$

where $\theta = \{x_0, y_0, z_0, r\}$.

2.1.6 Parts-Based Representations

Parts-based representations model an object as a graph of simpler object models (parts). This graph specifies how parts are connected to form the entire object. One advantage of

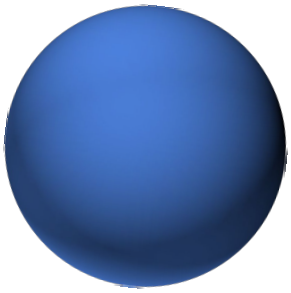


FIGURE 2.5: A parametrically generated 3D sphere.

parts-based approaches is their suitability for modeling articulated objects; by representing each rigid portion of an object as a distinct part, articulation is then defined as the movement (typically—but not always—rotation) of collections of parts. A common example of this type of representation is the Unified Robot Descriptor File used in the Robot Operating System (ROS) (Quigley et al., 2009). While in theory each object part could be any object representation, in practice surface mesh and parametric models are the most commonly used representations for parts-based approaches.

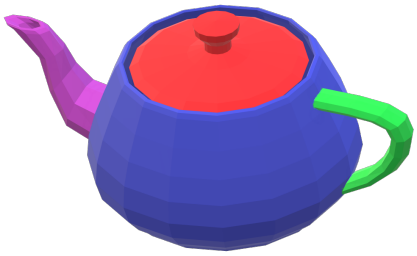


FIGURE 2.6: A parts-based mesh model of the Utah teapot.

2.1.7 Multiview Representations

Multiview object representations (Korn and Dyer, 1987) implicitly model a 3D object's shape via a collection of 2D images from various viewpoints. The most typical multiview

formulation uses a fixed number of discrete viewpoints, creating a compact fixed-size representation. Furthermore, because typical robot sensors such as RGB cameras and depth-sensors produce 2D images, multiview object representations directly correspond to sensor measurements obtained from the viewpoint locations. As a result, multiview representations are extremely fast to construct and a convenient choice for recognition tasks because features extracted from the multiview representation are directly comparable to features obtained from a sensor observation. Recently, mutiview representations have gained popularity for 3D classification tasks; see the following section for more details.

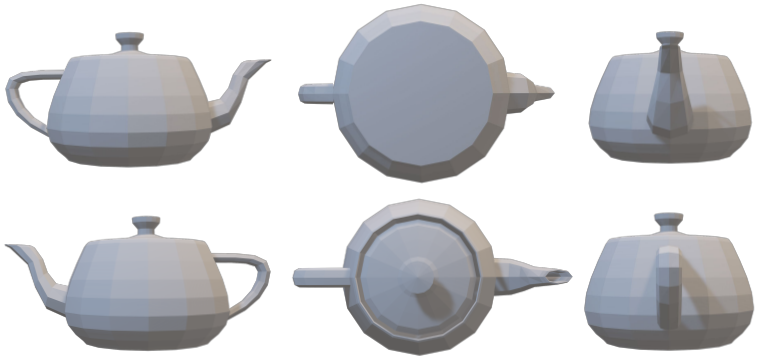


FIGURE 2.7: A multiview depiction of the Utah teapot.

2.2 Segmentation

Segmentation is the task of assigning group labels to sensor data; the most common example being assigning per-pixel membership in an RGB or depth image, although segmentation methods operating on pointclouds have also been proposed (Nguyen and Le, 2013; Wang et al., 2018). While early image-based segmentation was generally unsupervised, often treating the image as a graph and constructing a segmentation based on pixel similarity (Felzenszwalb and Huttenlocher, 2004; Zhang et al., 2008), modern segmentation methods are usually trained in a supervised fashion with individual segments corresponding to semantically meaningful entities. Segmentation can be either instance-level or category-level (also referred to as semantic segmentation). Category-level segmentation maps

every sensor input to a semantic category label (Chen et al., 2017; Lin et al., 2017; Yu et al., 2018), for example all pixels in an image corresponding to cars are mapped to the same label. Category-level segmentation does not disambiguate between multiple instances of the same category. Conversely, instance-level segmentation (Zhang et al., 2016; Liang et al., 2017; He et al., 2017) assigns sensor data a distinct label for each category instance; an instance-level segmentation algorithm would segment an image consisting of 10 cars and 20 pedestrians into 30 distinct segments (if we ignore background). For robot object-centric perception, instance segmentation is generally preferable to category-level segmentation, particularly when observing cluttered scenes. One of the most widely employed instance-level segmentation algorithms is MASK-RCNN (He et al., 2017), a region-based convolutional network designed to segment RGB images. Recently, a unified segmentation paradigm, panoptic segmentation (Kirillov et al., 2019), has been proposed which combines elements of semantic and instance segmentation by predicting individual masks for object instances but single semantic masks for non-object regions of the scene (for example grass). Thus, under a panoptic segmentation regime, every pixel in a scene is assigned both a semantic label and an instance label.

2.3 Classification

Object classification is the task of determining an object's semantic type given a sensor observation of that object. 2D classification, which performs classification based upon 2D input, is extremely well studied (Lowe et al., 1999; Gehler and Nowozin, 2009; Bergamo and Torresani, 2010; B. Browatzkiand and Wallraven, 2011; Krizhevsky et al., 2012; He et al., 2016; Hu et al., 2018; Tan and Le, 2019) and consists of predicting an object label given a single color or depth image. These systems are trained in a supervised fashion on a set of training data and evaluated on a distinct testing dataset; they are designed to determine the class of novel objects. Classification methods historically used handcrafted

image-level features (Viola et al., 2001; Dalal and Triggs, 2005) which were extracted and fed to supervised machine learning models such as support vector machines (Joachims, 1998) or random forests (Breiman, 2001). These methods are fairly effective for small datasets where the structure imposed by human-generated features helps ensure learning generalizes from little training data, but their relative efficacy tends to diminish in data-rich settings. A drawback of fixed features is lack of flexibility; it is difficult for even the most qualified humans to construct ideal features and using a fixed set of suboptimal input features can destroy useful information. It is thus desirable, when there is sufficient training data available, to learn features directly from the data. Modern classification approaches (Krizhevsky et al., 2012; He et al., 2016; Hu et al., 2018) do just this: making use of convolutional architectures to learn a hierarchically applied series of image filters directly from a —typically large—set of training data. These approaches have been hugely successful, achieving top-1 accuracy of over 84 percent across 1000 possible classes in the recent 2018 ImageNet challenge (Russakovsky et al., 2015; Huang et al., 2018).

In the last few years 3D classification has also gained considerable research attention. Modern approaches to 3D object classification rely on convolutional deep networks trained from full 3D models of objects. Broadly speaking, most current 3D classification approaches fall into two categories: multiview and volumetric. Volumetric approaches explicitly represent 3D objects as volumes (generally via voxels) (Maturana and Scherer, 2015; Qi et al., 2016; Shi et al., 2015; Wu et al., 2015; Su et al., 2018) while multiview approaches represent a 3D object as a series of 2D or 2.5D views (Su et al., 2015a; Bai et al., 2016; Su et al., 2018; Ma et al., 2019). Both methods have shown promising results and research has been conducted to combine the two representations (Hegde and Zadeh, 2016). Both 2D and 3D classification algorithms have progressed to the point of being highly-reliable—when sufficient training data is available—often outperforming humans in this setting. These methods work less well when there is a paucity of training examples and few-shot and zero-shot classification remains an open area of research (Chen et al.,

2019).

2.4 Pose Estimation

Pose estimation consists of determining an object's position and orientation in the world. With rigid objects, pose estimation generally consists of either a three degree of freedom orientation estimate or a six degree of freedom orientation and position estimate. Most pose estimation methods are single-instance, estimating the pose of a known object. In these settings, when a full 3D model of the object is available, techniques such as Iterative Closest Points (Besl and McKay, 1992), Bingham Distribution Pose Estimation (J. Glover and Bradski, 2011), and DNN approaches (Tulsiani and Malik, 2015; Xiang et al., 2017) have shown to be effective. Instance-based pose estimation approaches generally fall into one of three categories: inference-based, template-based, and matching-based. Inference-based approaches attempt to directly predict the pose of a known object, typically using a supervised learning algorithm. Because there is no training-testing distribution shift, these approaches (Tulsiani and Malik, 2015; Xiang et al., 2017; Li et al., 2018a) are largely immune to the common supervised learning problems of overfitting and lack of generalization. Template-based approaches perturb objects of interest, offline, and generate templates from these perturbations. To perform pose-estimation, these methods (Liu et al., 2012; Bakry and Elgammal, 2014; Rios-Cabrera and Tuytelaars, 2013) then extract features from the observation and match it to the most similar template, producing a pose estimate. Finally, matching-based approaches (Besl and McKay, 1992; Choi et al., 2012) attempt to directly match an object with its 3D model, iterativly refining pose-estimates until a (typically local) minimum error—between model and observation—is achieved. Some approaches, such as DeepIM (Li et al., 2018b), blend inference-based methods with other techniques to create hybrid systems.

These methods are not, however, designed to predict the pose of a novel object—they

work only when the observed object has been previously encountered. There has been less work on category-level pose estimation—predicting the pose of novel objects given a canonical notion of pose for each object category (Sun et al., 2018; Su et al., 2015b; Fidler et al., 2012). Category-level pose estimation is significantly more challenging than its instance-level counterpart because an exact model of the observed object unavailable. This difficulty is exacerbated if an object has unusual shape or a category of objects possesses significant shape variation. Most existing category-level pose estimation algorithms (Sun et al., 2018; Su et al., 2015b) treat pose estimation as a classification problem, discretizing orientations into discrete bins and predicting a particular pose bin. The current state-of-the-art method for category-level pose estimation (Sun et al., 2018) is capable of discrete pose-estimation into 15 degree bins with roughly 50 percent accuracy, insufficient performance for reliable robot operation.

2.5 3D Shape Completion

3D object completion consists of inferring the full 3D geometry of a partially observed 3D object. Initial work in this area focused on model repair; objects were mostly observed but contained holes and other small patches of missing information. By exploiting surrounding geometry and symmetry properties, these approaches could repair small gaps but were not designed to patch large missing portions of objects (Attene, 2010). More recently, symmetry exploitation methods have been proposed that rely on global-scale object symmetry to estimate the shape of unobserved object regions (Schiebener et al., 2016; Song and Xiao, 2016). Unlike hole filling approaches, symmetry methods can reconstruct large missing portions of objects but fail when when their symmetry heuristic is not applicable.

In robotics, database-based methods are highly popular. These approaches construct a large database of complete, object scans; when a novel object is encountered, it is used as a query into the database—the system attempts to retrieve an object in the database with

similar structure or features to the query. Commonly used features for matching include local point features (B. Drost and Ilic, 2010), shape features (L. Nan and Sharf, 2012), global features (Kim et al., 2013b), or a mixture of local and global features (Kim et al., 2012; Li et al., 2015; Bore et al., 2017). Because the database explicitly contains high quality models of object instances, extremely accurate information on the query object is available if an exact match exists in the database. This is very effective for tasks such as partially specified object completion (Li et al., 2015). A significant drawback exists, however, if an exact match is not found; while some approaches still attempt to find a nearest match in such a case (Li et al., 2015; B. Drost and Ilic, 2010), the results will be poor if the query object is sufficiently different from any in the database. Because instance-based database models are necessarily discrete—containing only a finite number of exemplars—these models will yield poor results if coverage of the object space is insufficiently dense. Furthermore, because the database is explicitly composed of training examples, it necessarily scales with the size of the training input. On moderately sized datasets this may not be an insurmountable issue, but it can become a problem as both the size of the class model and query latency increase with the training size. Hybrid approaches have also been proposed that attempt to perform shape completion locally and use the completed object as a database query to obtain a final estimate (Dai et al., 2017), but these methods require a large and dense object model database.

Parts-based 3D object representations—which represent objects via a dictionary of parts and use single geometric structures to represent objects as a combination of pieces (D. Huber and Hebert, 2004; R.B. Rusu and Hsu, 2010; Marini et al., 2006; Kim et al., 2013a)—have some applicability to 3D shape completion as well. One advantage of parts-based approaches is compactness—a shared dictionary of common parts means that maintaining a database of all previously seen objects is unnecessary. Work exists that proposes using symmetry and prior information to reconstruct hidden portions of objects using arranged parts (Shen et al., 2012; Sung et al., 2015), but requires structured training

data that includes a parts list. Parts-based approaches tend work well when objects contain similar parts arranged in different ways and when part information is available in the training data, and poorly when there is more variation in a class and symmetry assumptions fail to hold. Complex models consisting of numerous parts may also become computationally intractable to reason about as the number of possible configurations is exponential in the number of parts, so such models may need to be represented with low fidelity.

The current state-of-the-art in 3D object completion consists of deep regression methods. These approaches take as input a partially observed object, either from an RGB image (Tulsiani and Malik, 2015; Kar et al., 2015; Choy et al., 2016; Tatarchenko et al., 2016; Soltani et al., 2017; Tulsiani et al., 2017; Wu et al., 2017; Tatarchenko et al., 2017; Sun et al., 2018), or partially-observed shape data (Wu et al., 2015; Dai et al., 2017). These methods learn to map partially observed input to 3D shape via a supervised training regime and produce either an explicit shape prediction—generally via voxels—in canonical pose, or a multiview representation from a collection of fixed viewpoints (Tatarchenko et al., 2016). A variety of objective functions have been suggested for use during network training, with the most common being the l_2 loss, which for voxel-models is the Frobenius norm in voxel space. Although adversarial losses have also been used (Wu et al., 2016), using GAN-based architectures (Goodfellow et al., 2014), they have not shown significant benefits over explicit reconstruction loss (Wu et al., 2017; Sun et al., 2018). While CNN approaches have advanced the state-of-the-art in 3D shape completion, they tend to be data hungry, requiring a large number of training examples, and can struggle with producing high-resolution output, although Dai et al. (2017) attempt to address this issue by maintaining an explicit database of known high-resolution objects which are used to refine direct produced estimates, and Tatarchenko et al. (2017) propose a network architecture which produces an octree-based voxel output, making direct high-resolution predictions more efficient.

2.6 Integrated Approaches

Integrated approaches that perform multiple object-centric perceptual tasks simultaneously have historically been quite rare but are slowly gaining attention. By combining multiple perceptual tasks in a single module, redundant sub-tasks, such as feature extraction, can be performed once, improving computation efficiency. It has also been demonstrated that performing tasks such as category-level pose estimation and 3D shape completion simultaneously can lead to better performance by encouraging consistent predictions and forcing the learned features to better generalize (Sun et al., 2018). Examples of integrated approaches include Render for CNN (Su et al., 2015b) which performs pose-estimation and classification simultaneously, 3D ShapeNets (Wu et al., 2015) which performs classification and 3D completion simultaneously, and Pix3D which performs pose-estimation and 3D shape completion simultaneously. Our work, BEOs and their extensions, are the first object representation to simultaneously perform all three tasks with a single architecture. The current iteration of BEOs—HBEO-MDNs—exhibit state-of-the-art category-level pose estimation, 3D completion, and classification performance relative to other depth-image based methods, are competitive with current RGB-based 3D completion methods, and outperform existing work—of any input modality—on the task of category-level pose estimation.

3

Bayesian Eigenobjects

We set out to design a 3D object representation with the goal of enabling the classification, category-level pose-estimation, and 3D completion of novel objects from partial observations. Additional design goals include compact object representations, the capability to produce high-resolution output, and the ability to learn from a small number ($n < 20$) of object exemplars. To this end, we propose a novel 3D object representation, Bayesian Eigenobjects (BEOs) , designed with these characteristics in mind. BEOs employ Variational Bayesian Principal Component Analysis (VBPCA) as the basis for a low-dimensional multi-class object representation. By learning a compact basis for each class, we are able to represent objects using their projection coefficients—a much more compact representation than an uncompressed volume. With this representation, novel objects can be localized, classified, and completed by projecting them onto class bases and then projecting back into object-space. Our method scales gracefully with the number of objects in each class, requiring constant per-class computation for projection and reconstruction even as the number of previously encountered objects increases.

A key feature of our single, unified, object representation is the ability to perform

partial-object completion in 3D. Because objects in real environments are rarely completely visible from a single vantage point, the ability to produce even a rough estimate of the hidden regions of a novel object can be extremely useful. We applied our method to a dataset of common 3D objects and were able to successfully estimate the rotational pose of novel objects, reconstruct partially unobserved objects, and categorize the class of novel objects. BEOs significantly outperform prior joint classification and completion work in apples-to-apples comparisons, are significantly faster, and can also jointly estimate object pose.

3.1 Background: Principal Component Analysis

Principal Component Analysis (PCA) is a widely used statistical technique for finding an orthogonal basis for data with the first component laying across the dimension of maximum variance (Wold et al., 1987).

Let $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}\}$ be a set of n data points of dimension d. PCA finds an orthogonal basis, $\mathbf{W}$, such that

$$\mathbf{x_i} = \mathbf{W}\mathbf{c_i} + \boldsymbol{\mu} \quad \forall \mathbf{x_i} \in \mathbf{X}, \tag{3.1}$$

where $\mathbf{c_i}$ are the datapoint-specific coefficients for point i and $\boldsymbol{\mu}$ is the empirical mean of the data. For datasets too large to fit in memory, online approximations of PCA have also been proposed (Boutsidis et al., 2015).

PCA has many uses, including constructing a low-dimensional representation of data by retaining only the k most principal components—the components capturing the greatest variation—and projecting points onto the resulting space. Each $\mathbf{c_i}$ vector then has only k elements, and $\mathbf{W}$ is $d \times k$. A PCA-based method conceptually similar to our work is face detection via Eigenfaces (Turk and Pentland, 1991). This method uses PCA to learn a space in which aligned, consistently illuminated, and frontally observed faces reside. This space can then be used to learn a classifier which determines if a sliding window contains

a face. PCA has also been used in the analysis of 3D brain imaging (Zuendorf et al., 2003; Andersen et al., 1999). These methods vectorized a voxelized 3D image of brain activity and performed PCA to determine independent activation components. As a result, they are able to recover regions of the brain that co-activate.

3.1.1 Variational Bayesian Principal Component Analysis

BEOs employ Variational Bayesian Principal Component Analysis (VBPCA) to learn compact bases for classes of objects. VBPCA is an extension of probabilistic PCA (PPCA) (Tipping and Bishop, 1999) which models each datapoint, $\mathbf{x_i}$, as

$$\mathbf{x_i} = \mathbf{W}\mathbf{c_i} + \boldsymbol{\mu} + \epsilon_i \quad \forall \mathbf{x_i} \in \mathbf{X}, \tag{3.2}$$

where $\mathbf{X}$ is a matrix containing all datapoints such that column i of $\mathbf{X}$ is $\mathbf{x_i}$, $\mathbf{W}$ is a basis matrix, $\mathbf{c_i}$ is the projection of $\mathbf{x_i}$ onto that matrix, $\boldsymbol{\mu}$ is the mean of all datapoints, and $\epsilon_\mathbf{i}$ is zero mean Gaussian noise associated with datapoint i. The model parameters for PPCA may be efficiently estimated using expectation-maximization (EM), which alternates between updating the estimate of each datapoint's coefficient, $\mathbf{c_i}$, and updating $\mathbf{W}$, $\boldsymbol{\mu}$, and ϵ. While PPCA is well suited to density estimation and data compression, Bayesian PCA (BPCA) (Bishop, 1999a) further extends this model by introducing (Gaussian) priors (parametrized by $\mathcal{H}$) over the elements of $\boldsymbol{\mu}$ and W allowing BPCA to model the entire posterior probability of model parameters:[1]

$$p(\mathbf{W}, \boldsymbol{\mu}, \mathbf{C}|\mathbf{X}, \mathcal{H}). \tag{3.3}$$

Unfortunately, specifying this posterior probability is problematic; VBPCA overcomes this by approximating the posterior via factorization. As a result, each factor can be iteratively updated separately, during optimization, while the others are held constant. VBPCA approximates the posterior probability as:

$$q(\mathbf{W}, \boldsymbol{\mu}, \mathbf{C}) \approx p(\mathbf{W}, \boldsymbol{\mu}, \mathbf{C}|\mathbf{X}, \mathcal{H}), \tag{3.4}$$

[1] Note that column i of $\mathbf{C}$ is $\mathbf{c_i}$.

where $q(\mathbf{W}, \boldsymbol{\mu}, \mathbf{C})$ is a factored posterior approximation:

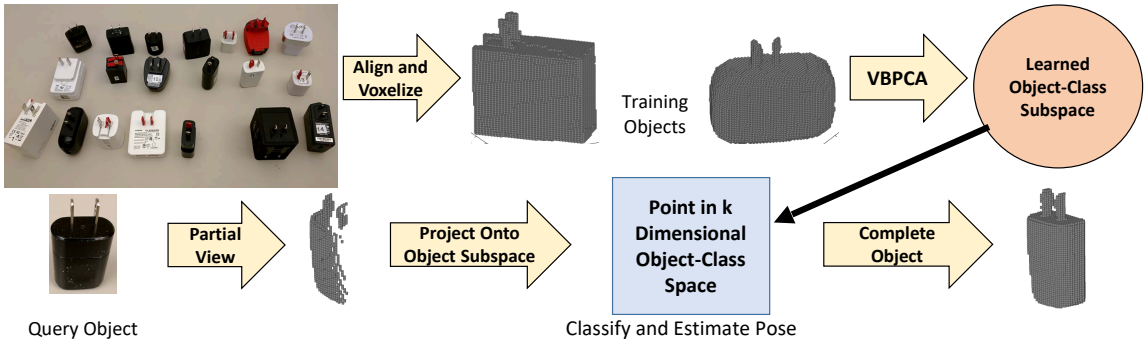
$$q(\mathbf{W}, \boldsymbol{\mu}, \mathbf{C}) = \prod_{i=1}^{d} q(\mu_i) \prod_{i=1}^{d} q(\mathbf{w}_i) \prod_{i=1}^{n} q(\mathbf{c_i}). \tag{3.5}$$

This approximation allows us to use EM to perform VBPCA in the same way it is employed for PPCA. For a more detailed explanation, please refer to Bishop (1999b).

VBPCA can be conceptualized as a probabilistically regularized version of PPCA, providing the advantages of PPCA (including intrinsic density estimation) with increased resilience to over-fitting due to the prior. This property makes it especially well suited for situations where the dimensionality of the problem is high compared to the number of datapoints, i.e. $n \ll d$ (Bishop, 1999b), as is true in our case.

3.2 Overview

The BEO framework constructs a generative model for each class of objects and then uses that model to enable inference about novel partial-object queries. BEOs are learned via a collection of aligned and fully observable 3D object meshes. At inference-time, BEOs receive a partially observed voxel object as input and predict its 3D shape, class, and orientation. Figure 3.1 illustrates an overview of this process.



Top: The training process. Bottom: An example query.

FIGURE 3.1: A high-level overview of the BEO process.

3.2.1 Class Models: Eigenobject Construction via VBPCA

BEOs are learned from a library of known objects of several classes, with each object consisting of a complete 3D scan; see Figure 3.2 for an example collection of aligned and voxelized car models. Each voxelized object is flattened into a 1-dimensional length d vector and treated as a single point in d dimensional voxel space. These points form the dataset upon which we perform VBPCA to find the subspace, $\mathbf{W_s}$ and mean $\boldsymbol{\mu_s}$ specific to class s.

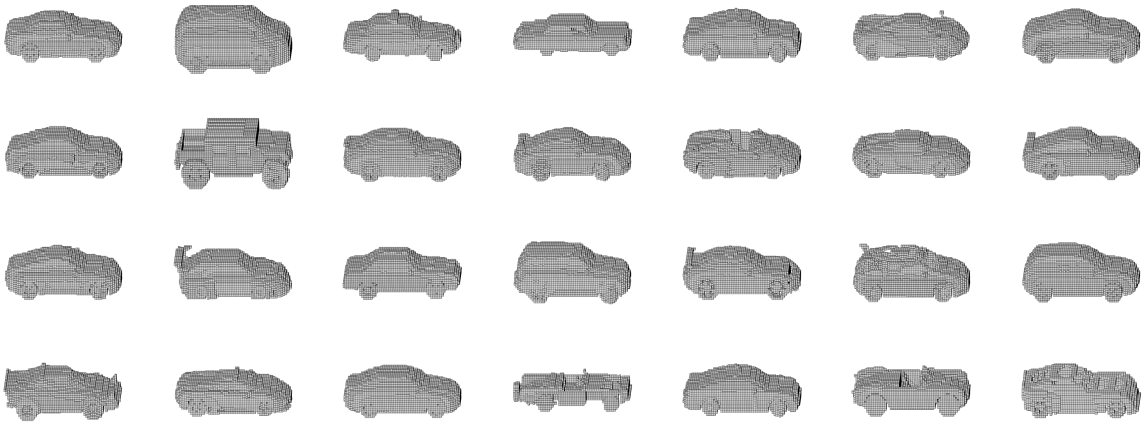


FIGURE 3.2: A collection of aligned and voxelized cars.

Given $\mathbf{W_s}$ and $\boldsymbol{\mu_s}$, a novel object, $\mathbf{o}$, of class s, can be projected onto learned space:

$$\mathbf{o_s}' = \mathbf{W_s}^T(\mathbf{o} - \boldsymbol{\mu_s}). \tag{3.6}$$

Conversely, any point in the space of $\mathbf{W_s}$ can be converted back into flattened voxel form via

$$\hat{\mathbf{o}}_\mathbf{s} = \mathbf{W_s}\mathbf{o_s}' + \boldsymbol{\mu_s}. \tag{3.7}$$

We refer to $\hat{\mathbf{o}}_\mathbf{s}$ as the "completed" version of $\mathbf{o}$ and $\mathbf{o_s}'$ as the "projected" version of $\mathbf{o}$ (with respect to class s).

We need not store or query an entire object database; instead, we need only store $\mathbf{W_s}$ and $\boldsymbol{\mu_s}$ for each class of objects ($\forall s \in S$). We can also represent any object in a given class using a single coefficient vector of dimension k_s. In practice, the number of bases for each

class is far less than the dimensionality of each datapoint ($k_s \ll d$), providing a compact representation.

3.2.2 Object Classification

An essential part of many robot tasks is novel-object classification. Let the learned BEO models for multiple classes be denoted $\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, ..., \boldsymbol{\theta_m}$, where m is the number of classes and $\boldsymbol{\theta_s} = \{\mathbf{W_s}, \boldsymbol{\mu_s}\}$, and let the novel query object be denoted $\mathbf{o_q}$. We wish to estimate the class label, $\hat{l}_q$, of $\mathbf{o_q}$ by selecting $\hat{l}_q$ from set S.

Our classification method leverages the trained low-dimensional space to learn a compact density model for each class; while such a density model is infeasible in 3D space as even size 30^3 objects contain tens of thousands of variables, it is possible in this much smaller projected space where each class is modeled as an anisotropic Gaussian.

From the learned subspaces for each class, we construct a single larger subspace upon which objects across all classes lie:

$$\mathbf{W} = [\mathbf{W_1}, ..., \mathbf{W_m}, \boldsymbol{\mu_1}, ..., \boldsymbol{\mu_m}].$$

$\mathbf{W}$ is a $d \times (k_1 + \ldots + k_m + m)$ matrix with rows corresponding to dimensions in voxel space and columns to basis vectors. Because $\mathbf{W}$ may be low-rank, which will introduce additional variables to estimate without increasing the expressiveness of our model, we find a matrix, $\mathbf{W'}$, with orthonormal columns that span the range of $\mathbf{W}$. This can be straightforwardly accomplished by letting $\mathbf{W'} = \mathbf{U'}$ where $\mathbf{USV^T} = \mathbf{W}$ is the singular value decomposition of $\mathbf{W}$ and $\mathbf{U'}$ is formed by retaining only the first $rank(\mathbf{W})$ rows of $\mathbf{U}$.

After learning the shared subspace, density models for each class can be found by estimating the mean and covariance of m multivariate Gaussian distributions, each of dimensionality $rank(\mathbf{W})$. Estimation of the mean is straightforward:

$$\bar{\mathbf{o}}'_\mathbf{s} = \sum_{\mathbf{o}'_\mathbf{s} \in \mathbf{O}'_\mathbf{s}} \frac{\mathbf{o}'_\mathbf{s}}{n_s}, \tag{3.8}$$

where n_s is the number of training objects in class s and $\mathbf{O'_s}$ is the $rank(\mathbf{W})$ by n_s matrix of projected training objects in class s.

Unfortunately, estimating the population covariance is more difficult. The simplest approach uses the sample covariance matrix:

$$\mathbf{\Sigma_s} = \frac{1}{n_s - 1}\mathbf{\bar{O}'_s}\mathbf{\bar{O}'_s}^T,$$

where $\mathbf{\bar{O}'_s}$ is created by subtracting $\mathbf{\bar{o}'_s}$ from each column in $\mathbf{O'_s}$. Unfortunately, this method works poorly in practice. Although $\mathbf{W'}$ is far smaller than full 3D object-space, it is still fairly large, easily several hundred dimensions. As a result, accurately estimating the covariance for each class with only several hundred datapoints points per class is problematic. We utilize a combination of two forms of covariance shrinkage to regularize the estimated covariance matrix. Starting with $\mathbf{\Sigma_s}$, we first shrink it by adjusting its eigenvalues towards their mean, finding $\mathbf{\Sigma'_s}$. Next, we further regularize $\mathbf{\Sigma'_s}$ by regressing it towards $\mathbf{\Sigma'_{s-2}}$, the so-called "two parameter covariance matrix". Diagonal elements of $\mathbf{\Sigma'_{s-2}}$ are all equal to the mean variance in $\mathbf{\Sigma'_s}$ while non-diagonal elements of $\mathbf{\Sigma'_{s-2}}$ are all equal to the mean off-diagonal covariance in $\mathbf{\Sigma'_s}$. The final estimate for the covariance of a given class is thus

$$\mathbf{\Sigma''_s} = \lambda\mathbf{\Sigma'_s} + (1 - \lambda)\mathbf{\Sigma'_{s-2}}. \tag{3.9}$$

The precise calculation of the optimal shrinkage amount, λ, and the eigenvalue shrinkage amounts used in $\mathbf{\Sigma'}$ are described in detail in Ledoit and Wolf (2015), Daniels and Kass (2001), and Schäfer and Strimmer (2005).

Once probability density functions for each of the classes have been learned, classification is performed in a maximum a posteriori fashion. Given query object $\mathbf{o_q}$, its projection, $\mathbf{o'_q}$, onto $\mathbf{W'}$, and $P(s)$, the prior probability of class s, the most probable class label, $\hat{l}_q$, is

$$\hat{l}_q = \arg\max_{s \in S} \frac{P(s)D(\mathbf{o'_q}|s)}{\sum_{s_j \in C} P(s_j)D(\mathbf{o'_q}|s_j)}, \tag{3.10}$$

where $D(\mathbf{o'_q}|c_i)$ denotes the density of the learned PDF, found using equations 3.8 and 3.9, for class s at location $\mathbf{o'_q}$,

$$D(\mathbf{o'_q}|s) = \frac{1}{\sqrt{(2\pi)^a|\boldsymbol{\Sigma_s}|}} exp\left(-\frac{1}{2}(\mathbf{o'_q} - \bar{\mathbf{o'_s}})^T \boldsymbol{\Sigma}_s^{-1}(\mathbf{o'_q} - \bar{\mathbf{o'_s}})\right), \tag{3.11}$$

where a denotes $rank(\mathbf{W'})$ for readability.

Unlike methods that operate directly on objects, there is no need to tune 3D features for individual classes. Furthermore, our approach can accommodate significantly higher resolution objects than competing DNN methods; in our experiments BEOs were able to model objects of size 273^3, which contain over twenty million voxels, while constructing a DNN with such high-dimensional input is infeasible.

3.2.3 Pose Estimation

Like classification, pose estimation is necessary for many robotic manipulation and planning tasks, especially object manipulation. While it is relatively straightforward to acquire a rough estimate of an object's position given an object detection and segmentation, determining orientation is more difficult. Here, we do not assume we have a model of the query object, making our pose estimation task far more difficult; we cannot simply match the query object to its exact model. We must instead determine its pose relative to a canonical class-specific baseline. We accomplish using a try-verify approach, also known as pose estimation by search (Narayanan and Likhachev, 2016). These approaches use a generative model of the object and sample multiple poses to find one that is maximally likely or minimizes scene error.

Let $\mathbf{R}(\mathbf{o_q}) = \{\mathbf{o_q^1}, \mathbf{o_q^2}, ...\mathbf{o_q^p}\}$ be query object $\mathbf{o_q}$ in p orientations. To estimate the true orientation of $\mathbf{o_q}$, $\hat{r}_q$, one possible solution is

$$\hat{r}_q = \underset{\mathbf{o_q^r} \in \mathbf{R}(\mathbf{o_q})}{\arg\min} \; ||\mathbf{o_q} - \mathbf{o_q^r}||_2. \tag{3.12}$$

This can result in ambiguity however, particularly with only partially specified query objects. This estimator ignores learned class properties; a cabinet, for instance, might project well onto the space of toilets because toilets have a large rectangular back but such a toilet would be highly unusual. Using only Equation 3.12 does nothing to address this, motivating an alternate approach, conceptually based on the density estimator, Equation 3.11, used in classification:

$$\hat{r}_q = \underset{\mathbf{o_q^r} \in \mathbf{R}(\mathbf{o_q})}{\arg\max} \frac{P(r) D(\mathbf{o_q'^r} | s)}{\sum_{r_j \in R} P(r_j) D(\mathbf{o_q'^{r_j}} | s)}, \tag{3.13}$$

where $\mathbf{R}$ is the set of possible poses being searched over and $P(r)$ is the prior probability of pose r.

Equation 3.13 selects the MAP pose estimate; an advantage of this estimator is its sensitivity to the misalignment of important geometry. Consider the example of a bathtub: while mostly symmetrical around their z-axis, bathtubs have a spout at one end which provides key information for pose estimation. An approach based on Equation 3.12 weights error equally in all places while the density method can respect that some regions contain more information than others.

If a high resolution orientation is required, there may be a large number of candidate poses. Fortunately, each candidate rotation can be calculated independently and thus the process is straightforwardly parallel; it is possible to distribute the workload to multiple processors or accelerate it via GPU. Our experiments investigate both full 3DOF pose estimation as well as 1DOF rotation about the z axis.

3.2.4 Partial Object Completion

In real-world environments, robots almost never have entire views of the objects they encounter. Even with the prevalence of multiple-sensor multiple-modality perception on modern robots, obtaining a complete 3D view of an encountered object requires sensing

from multiple sides. If robots are to be mobile and operate outside of laboratory environments, it is unreasonable to expect that they will always perceive objects from numerous vantage points before reasoning about them.

The alternative is to infer, from a partial model of an object and prior knowledge, what the remaining portions of a query object may be. BEOs provide this ability because they offer a generative representation of objects in a class. Each learned basis provides an object-class manifold; if we can find the point on this manifold that best corresponds to a given partially-observed object, projecting back into voxel object-space yields a prediction of the unobserved portions of that object.

Similarly to Li et al. (2015), we assume that the partial query objects consist of filled, empty, and unknown pixels. Let d' denote the number of known (filled and empty) elements of $\mathbf{o_q}$. It is useful to define a d' by d binary selection matrix $\mathbf{V_q}$ such that $\mathbf{V_q o_q} = \mathbf{w_q}$ where $\mathbf{w_q}$ is a length $d' < d$ vector consisting of only the known elements of $\mathbf{o_q}$. $\mathbf{V_q}$ can be created by constructing a size d identity matrix and then removing all rows corresponding to unknown elements of $\mathbf{o_q}$ (e.g. if the ith element of $\mathbf{o_q}$ is unknown then the ith row of the identity is removed). Let $\mathbf{o'_q}$ denote the smallest error projection of $\mathbf{o_q}$ onto basis $\mathbf{W'}$. The error induced by an arbitrary projection $\mathbf{o'^i_q}$ with respect to $\mathbf{o_q}$ is

$$E(\mathbf{o'^i_q}) = ||\mathbf{V_q}(\mathbf{W'o'^i_q}) - \mathbf{w_q}||_2^2. \tag{3.14}$$

The gradient of this error with respect to $\mathbf{o'^i_q}$ is thus

$$E^{'}(\mathbf{o'^i_q})d\mathbf{o'^i_q} = 2\mathbf{W'}^T\mathbf{V_q}^T[\mathbf{V_q}(\mathbf{W'o'^i_q}) - \mathbf{w_q}]. \tag{3.15}$$

This error function is convex; to find the projection that minimizes $E(\mathbf{o'^i_q})$, we set the gradient to 0 and solve for $\mathbf{o'_q}$.

$$\mathbf{A_q o'_q} = \mathbf{b_q} \tag{3.16}$$

where

$$\mathbf{A_q} = \mathbf{W'}^T\mathbf{V_q}^T\mathbf{V_q W'} \tag{3.17}$$

and

$$\mathbf{b_q} = \mathbf{W'}^T \mathbf{V_q}^T \mathbf{w_q}. \tag{3.18}$$

When projecting for classification and pose estimation, we found it helpful in practice to employ gentle lasso regularization (Tibshirani, 1996) when solving for $\mathbf{o'_q}$,

$$\mathbf{o'_q} = \underset{\mathbf{o'^i_q}}{\arg\min} \; ||\mathbf{A_q o'^i_q} - \mathbf{b_q}||_2 + \lambda ||\mathbf{o'^i_q}||_1 \; , \tag{3.19}$$

using $\lambda = 10^{-5}$.

Once we have obtained our projection estimate, $\mathbf{o'_q}$, we can complete the object using Equation 3.7. The completed object, $\hat{\mathbf{o}}_\mathbf{q}$, minimizes the error between itself and the known portion $\mathbf{o_q}$ while predicting the unknown portions of $\mathbf{o_q}$.

3.2.5 BEOs: Joint Pose, Class, and Geometry Estimation

BEOs can be employed to perform all three operations (pose estimation, classification, and geometric completion) simultaneously. This full process consists of simultaneous pose estimation and classification followed by completion and requires maximizing the joint probability over both pose and class. Because both classification and completion are quite sensitive to misalignment, we use a two step approach to pose estimation. In practice, Equation 3.13 is unreliable when used to estimate global orientation for an object with unknown class. To address this, we employ a hierarchical coarse to fine approach; while we employ Equation 3.13 during the fine-tuning phase, we employ a different method for the initial coarse estimate.

We define a coarse error based on the L_2 distance between an object and its back-projected version as,

$$e(\mathbf{o_q}, \hat{\mathbf{o}}^\mathbf{r}_\mathbf{q}) = 1 - \frac{||\mathbf{o_q} - \hat{\mathbf{o}}^\mathbf{r}_\mathbf{q}||_2}{|\mathbf{o_q}|}, \tag{3.20}$$

where a score of 1 denotes a perfect match and a score of 0 indicates that all voxels differ between the object and its back-projection. Intuitively, projecting an object, given its true

orientation, onto its class subspace and then re-projecting it back into voxel-space, should result in a re-projection that closely matches the initial object. An initial estimator might simply find the class and orientation minimizing $e(\mathbf{o_q}, \hat{\mathbf{o}}^{\mathbf{r}}_{\mathbf{q}})$.

In practice, Equation 3.20 works well for objects of the same class, but often fails when applied to objects of very different shape, making it unsuitable for comparing disparate objects across multiple classes. To combat this, we leverage a more nuanced representation of error based on the 3D Euclidean Distance Transform (EDT) (Maurer et al., 2003). From $\mathbf{o_q}$ and $\hat{\mathbf{o}}_{\mathbf{q}}$ we compute the EDT for each object, $\mathbf{D}$ and $\hat{\mathbf{D}}$ respectively. Each distance transform forms a 3D matrix, with the same dimensions as the object from which it was created, and entry in the distance transform is the Euclidean distance between its corresponding voxel in the original object and the closest filled voxel. As a result, entries that correspond to filled voxels have a value of 0 while entries that correspond to voxels far from filled portions of the object have high value. By computing the 2-norm of the difference between distance fields, we can create a more robust measure of inter-object distance:

$$e'(\mathbf{o_q}, \hat{\mathbf{o}}^{\mathbf{r}}_{\mathbf{q}}) = ||\mathbf{D} - \hat{\mathbf{D}}^{\mathbf{r}}_{\mathbf{q}}||_2. \tag{3.21}$$

Because it implicitly captures shape differences between two objects, this distance provides a more robust means of object comparison than Equation 3.20. Refer to Figure 3.3 for an illustration of an example EDT.

During the first, coarse, step, we use EDT error from Equation 3.21 to coarsely estimate the object's pose. We then finely discretize the rotation space in a region around the initial coarse estimate and rely on a modified version of Equation 3.13,

$$\{\hat{r}_q, \hat{\mathbf{l}}_{\mathbf{q}}\} = \underset{\mathbf{o}^{\mathbf{r}}_{\mathbf{q}} \in \mathbf{R}(\mathbf{o_q}),\ s \in S}{\arg\max} \quad \frac{P(r)D(\mathbf{o}^{'\mathbf{r}}_{\mathbf{q}}|s)P(s)}{\sum_{r_j \in R}\sum_{s \in \mathbf{S}} P(r_j)D(\mathbf{o}^{'\mathbf{r_j}}_{\mathbf{q}}|s)P(s)}, \tag{3.22}$$

to obtain our final rotation estimate and classification label by marginalizing over both possible poses and possible classes. Note that $\mathbf{o}^{'\mathbf{r}}_{\mathbf{q}}$ denotes query object $\mathbf{o_q}$, in pose r,

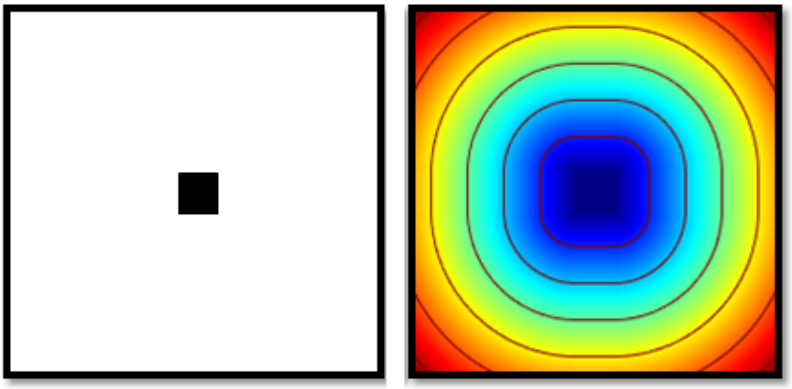


FIGURE 3.3: An example EDT (right) of a 2D square (left).

projected onto W'. Because Equation 3.22 is unreliable when used to estimate global orientation but performs well in a region close to the correct pose, employing it only during the fine-tuning step proves effective.

3.3 Experimental Results

We characterize the performance of our approach using the ModelNet10 dataset (Wu et al., 2015), which consists of 4889 (3991 training and 908 test) aligned 3D objects, each of size 30^3, spread across 10 classes: Bathtubs, Beds, Chairs, Desks, Dressers, Monitors, Night Stands, Sofas, Tables, Toilets. During VBPCA, we automatically selected a basis size capturing 60 percent of variance, between 30 and 70 components per class. and used zero-mean unit-variance Gaussian hyperparameters for regularization. We also illustrate some example high-resolution completions, obtained from 20 USB charging plugs which were manually scanned in our lab using a MakerBot 3D scanner and from two additional synthetic classes sourced from ShapeNet (Chang et al., 2015). We employed coordinate descent congealing (Learned-Miller, 2006) to roughly align the objects in each class, manually inspecting and refining the alignment as required—objects sourced from ModelNet and ShapeNet and were pre-aligned, while our manually scanned objects required alignment.

3.3.1 *Classification and Object Completion*

To provide a direct comparison with competing approaches, we assume the pose of each single-viewpoint query object is known and that the task consists of estimating the object's class label and full 3D geometry. We evaluate against 3DShapeNets (Wu et al., 2015), as well as a baseline which measures similarity to the mean training object in each class and selects the class with the most similar mean element. Completion performance was measured by calculating the Euclidean error between the true geometry of the query object and the estimated completion. For each of the 908 test objects, test queries were created using a single top-down depth view. Table 3.1 summarizes the classification performance of each of these methods while Figure 3.4 contains their completion errors and query times.[2] We observed that the comparatively simple baseline method conflates several geometrically similar classes quite badly and both 3DShapeNets and the baseline had significant trouble with tables, likely because the top down view makes these challenging to distinguish from dressers and nightstands. It should also be noted that while ModelNet10 is a popular benchmark for full-object 3D classification, our experiments explore single-view classification performance, a different task.

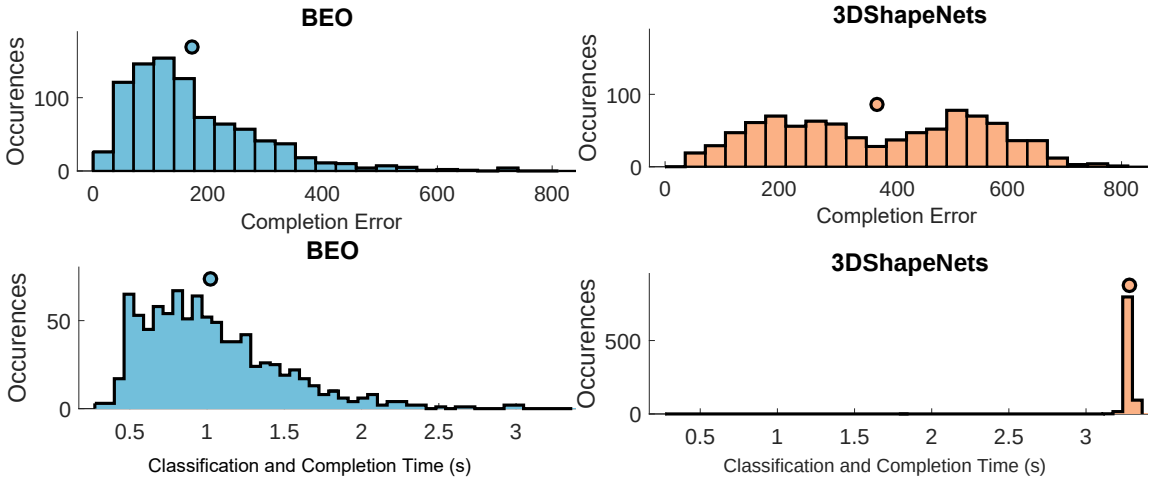


FIGURE 3.4: Completion errors and query time.

[2] Mean error and time are indicated by circular dots.

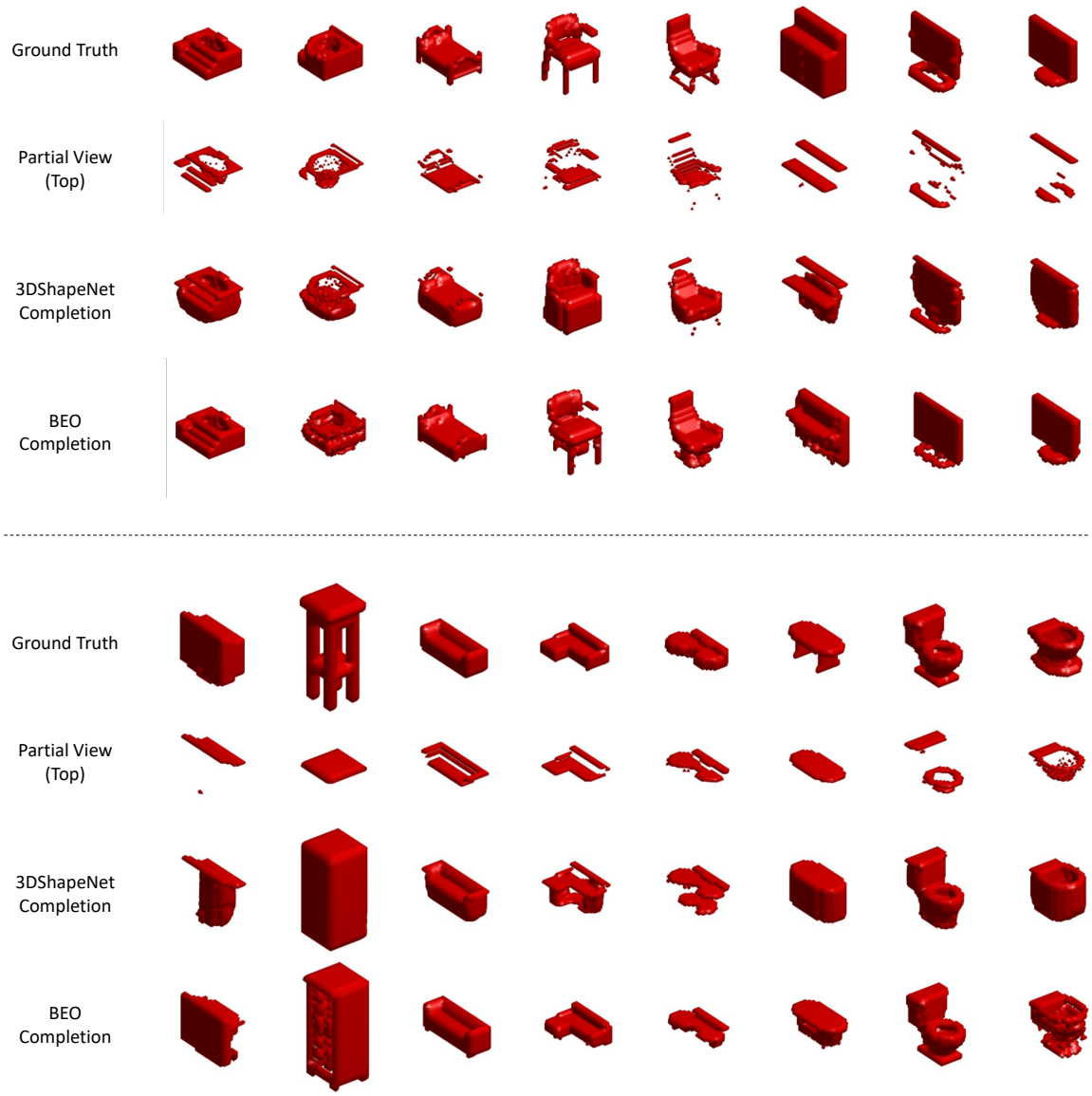


FIGURE 3.5: A sampling of BEO object completions.

In both classification accuracy and reconstruction error, BEO significantly outperforms 3DShapeNets, achieving nearly 20 percent greater classification accuracy. 3DShapeNets particularly struggled with tables, misclassifying them as night stands or dressers in nearly all instances due to their flat horizontal tops. While our BEO approach also exhibited this behavior to a lesser degree, in many instances it was able to leverage the small differences in the size and aspect ratios of these objects to successfully classify them. Furthermore,

Table 3.1: ModelNet10 classification accuracy.

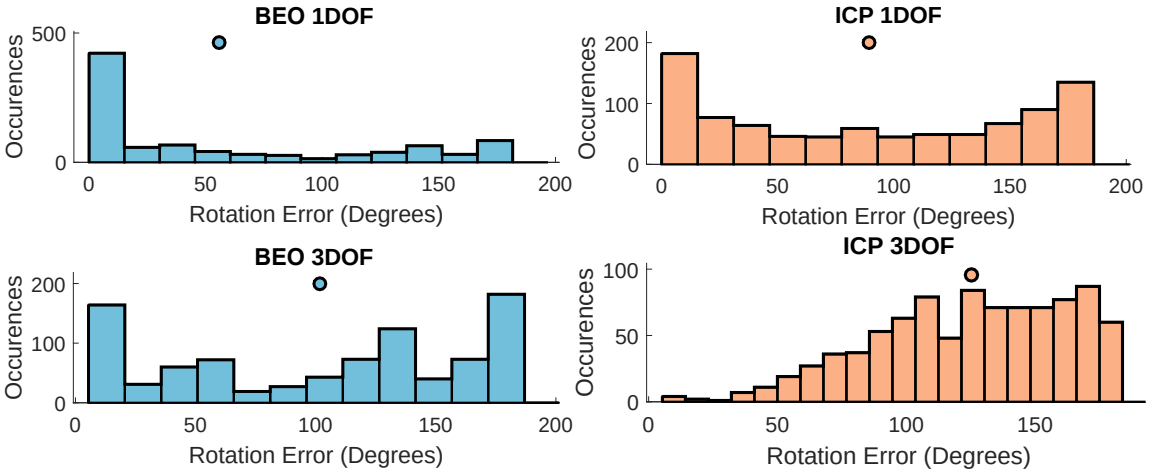
Unknown pose corresponds to 1DOF pose-estimation about the z-axis.

Known Pose	Bathtub	Bed	Chair	Desk	Dresser	Monitor	Night Stand	Sofa	Table	Toilet	**Total**
BEO	48.0	**95.0**	**93.0**	**46.5**	64.0	**91.0**	55.8	**92.0**	**75.0**	80.0	**76.3**
Baseline	70.0	94.0	0.0	17.4	67.4	78.0	**75.6**	88.0	0.0	**82.0**	56.7
3DShapeNets	**76.0**	77.0	38.0	22.1	**90.7**	74.0	38.4	57.0	1.0	79.0	54.4
Unknown Pose	Bathtub	Bed	Chair	Desk	Dresser	Monitor	Night Stand	Sofa	Table	Toilet	**Total**
BEO	4.0	64.0	83.0	16.3	51.2	86.0	36.0	49.0	76.0	46.0	54.5

although our query times exhibit some fluctuation because each query requires solving a variable-size system of equations, our method is approximately three times faster than 3DShapeNets.

BEO-based 3D completion differs significantly from the method used by 3DShapeNets; while 3DShapeNets first classifies an object and then performs completion, BEOs project objects onto the BEO subspace and then classifies them. As a result, 3DShapeNets exhibits bimodal completion performance; when it misclassifies an object, its completion results degrade significantly. BEOs do not exhibit this behavior, sometimes completing an unusual object (with respect to the training set) in a reasonable way, even while misclassifying it. Figure 3.5 illustrates some sample completions from BEOs and 3DShapeNets and is best viewed digitally with zoom.

3.3.2 Pose Estimation



Mean error is indicated by circular dots.

FIGURE 3.6: Pose estimation error for BEOs and an ICP baseline.

To evaluate pose estimation performance, we performed experiments in both 1DOF with 1 degree of precision and $3DOF$ with 20 degrees of precision using the density estimator from Equation 3.13. As 3DShapeNets cannot estimate pose, we compared against an ICP approach that warps the query object to the class-mean. In 1DOF experiments, each query

object was given a random orientation obtained by sampling uniformly from $[0, \pi)$. In 3DOF experiments, each query object's orientation was given by by sampling a quaternion uniformly at random from the surface of the 4D hypersphere. As above, queries consisted of a single viewpoint looking down along the z-axis.

We found that while BEOs dramatically outperformed ICP—which tended to get stuck in local minima—BEO pose-estimation performance is significantly worse in 3DOF than it is in the smaller 1DOF case, likely due to the scaling challenges endemic to explicitly enumerating the search-space of poses.

3.3.3 Joint Pose, Class, and 3D Geometry Estimation

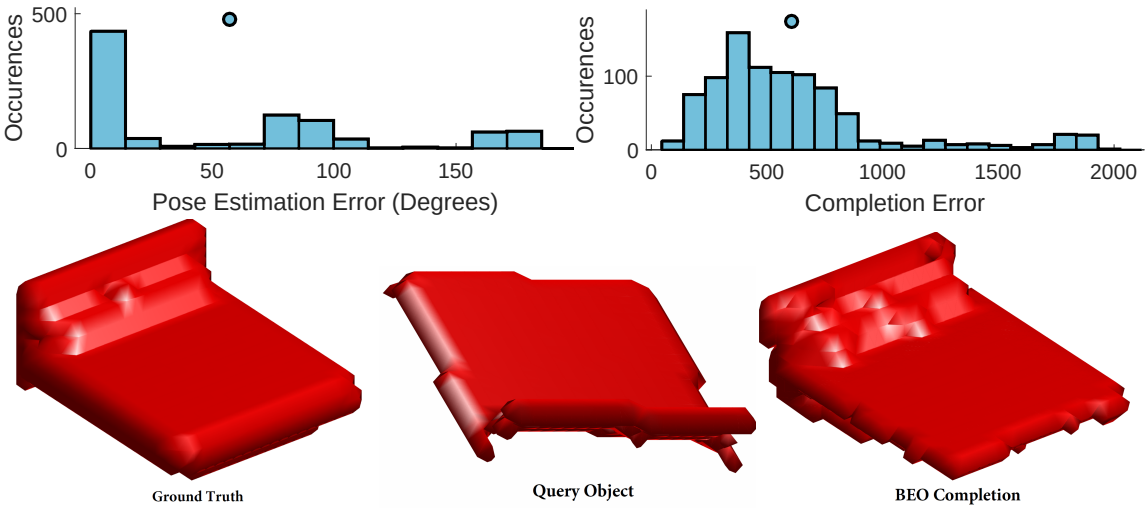
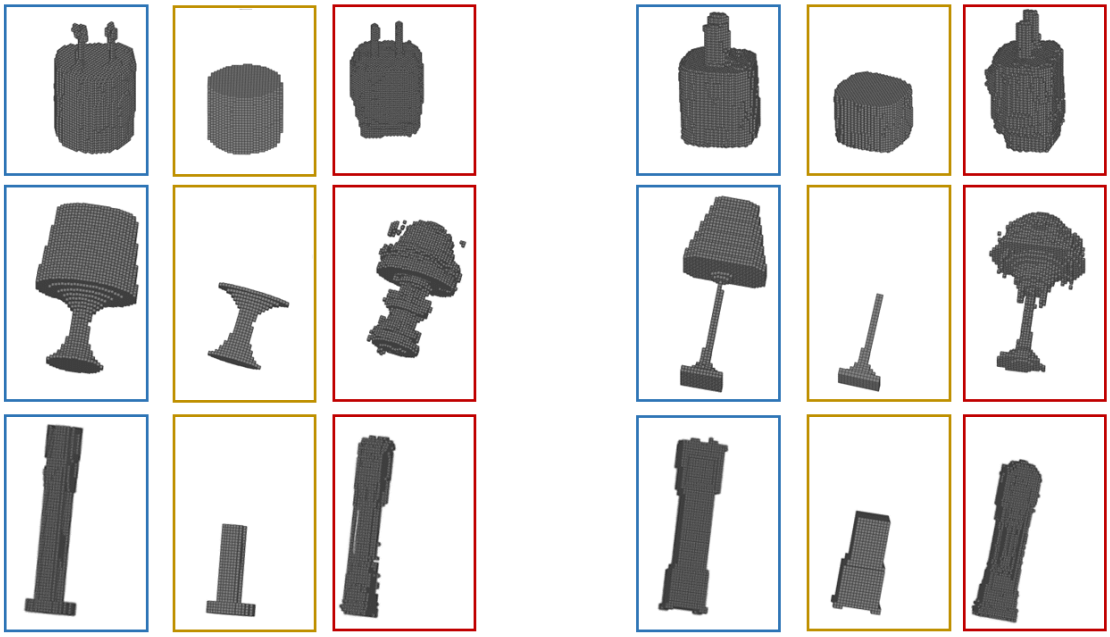


FIGURE 3.7: Full system performance.

We next evaluated our entire procedure of joint pose estimation, classification, and 3D completion. Input query objects were obtained by randomly rotating objects about their z-axis and extracting a single top-down view. While BEO performance degraded somewhat in this more challenging instance, we achieve equal classification performance with 3DShapeNets without employing knowledge of the object's pose. Table 3.1 contains our classification performance in this setting while Figure 3.7 shows our pose estimation and completion performance as well as an example query.

3.3.4 *High Resolution Output and Limited Training Data*

To demonstrate our approach's applicability to small training set sizes and high resolution output, we conducted experiments on three additional datasets, two synthetic and one composed of physical objects. The physical-object dataset contained 20 wall USB charging plugs—aligned to canonical orientation with the prongs directed upward—scanned on a MakerBot 3D scanner. The scanned mesh files were voxelized into $254 \times 254 \times 254$ objects, forming the dataset. The other two datasets were both sourced from ShapeNet (Chang et al., 2015) and consisted of of a random sampling of standing grandfather clocks and tabletop lamps. Both the lamp and clock datasets were randomly split into two sub-groups for training and testing. The lamp training set consists of 40 lamps while the lamp test set consists of 10. The clock training set consists of 35 clocks and the clock test set consists of 8. Objects in both the lamp and clock classes were voxelized into size $273 \times 273 \times 273$ objects. The clock and lamp class manifolds were then learned with 17 components while

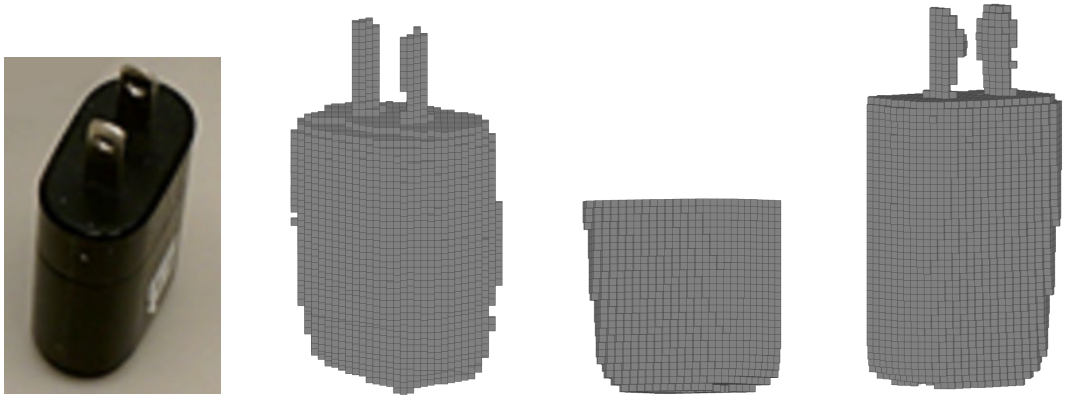


Left: ground truth. Center: partial observation. Right: BEO completion.

FIGURE 3.8: Example BEO Completions.

the plugs used 10. Figures 3.9 and 3.8 illustrate several example completions with a

Plug Dataset



Query Plug | True Shape | Partial Observation | Estimated Shape

FIGURE 3.9: Example high-resolution completion from a small training dataset.

voxelized visualization. At lower resolutions, recovering fine detail such as the shape of the USB plug prongs would be impossible. Note that due to the very small training-set size, not all completions are fully successful, but even in these failure cases, much of the low-frequency object structure is reproduced.

3.4 Discussion

We found that by using Variational Bayesian Principal Component Analysis to construct a low-dimensional multi-class object representation, we were successfully able to estimate the 3D shape, class, and pose of novel objects from limited amounts of training data. BEOs outperform prior work in joint classification and completion with queries of known pose, in both accuracy and classification performance, while also being significantly faster and scaling to higher resolution objects. Furthermore, BEOs are the first object representation that enables joint pose estimation, classification, and 3D completion of partially-observed novel objects with unknown orientations. A primary benefit of BEOs is their ability to perform partial object completion with limited training data. Because objects in real environments are rarely observable in their entirety from a single vantage point, the ability to produce even a rough estimate of the hidden regions of a novel object is mandatory. Additionally, being able to classify partial objects dramatically improves the efficiency of object-search tasks by not requiring the agent to examine all candidate objects from multiple viewpoints. Significantly however, BEOs require that observed objects be not only segmented, but also voxelized, a task that is generally quite onerous in practice. Furthermore, while pose estimation in 1 degree of freedom is reasonable using BEOs, because BEOs rely on pose-estimation by search, the process becomes infeasibly slow in higher dimensions. In the following chapter, we introduce an extension of BEOs that alleviate some of these limitations while also improving performance.

4

Hybrid Bayesian Eigenobjects

4.1 Introduction

While Bayesian Eigenobjects provides a useful foundation for object-centric perception, they have several important limitations, including that pose-estimation is significantly slower than realtime and a requirement that partially-observed input be voxelized. We now present an extension of BEOs, Hybrid Bayesian Eigenobjects (HBEOs), that addresses these limitations, and improves performance. In contrast to BEOs, HBEOs use a learned non-linear method—specifically, a deep convolutional network (LeCun and Bengio, 1995)—to determine the correct projection coefficients for a novel partially observed object. By combining linear subspace methods with deep convolutional inference, HBEOs draw from the strengths of both approaches.

Previous work on 3D shape completion employed either deep architectures which predict object shape in full 3D space (typically via voxel output) (Wu et al., 2015, 2016; Dai et al., 2017; Varley et al., 2017; Sun et al., 2018) or linear methods which learn linear subspaces in which objects tend to lie as we proposed for BEOs; however both approaches have weaknesses. End-to-end deep methods suffer from the high dimensionality of object

space; the data and computation requirements of regressing into $50,000$ or even million dimensional space are severe. Linear approaches, on the other hand, are fast and quite data efficient but require partially observed objects be voxelized before inference can occur; they also lack the expressiveness of a non-linear deep network. Unlike existing approaches, which are either fully linear or perform prediction directly into object-space, HBEOs have the flexibility of nonlinear methods without requiring expensive regression directly into high-dimensional space. Additionally, because HBEOs perform inference directly from a depth image, they do not require voxelizing a partially observed object, a process which requires estimating a partially observed object's full 3D extent and pose prior to voxelization. Empirically, we show that HBEOs outperform competing methods when performing joint pose estimation, classification, and 3D completion of novel objects.

4.2 Overview

HBEOs use an internal voxel representation, similar to both Wu et al. (2015) and BEOs, but use depth images as input, avoiding the onerous requirement of voxelizing input at inference time. Like BEOs, HBEOs learn a single shared object-subspace; however, HBEOs learn a mapping directly from depth input into the learned low-dimensional subspace and predict class and pose simultaneously, allowing for pose, class, and shape estimation in a single forward pass of the network.

The HBEO subspace is defined by a mean vector, $\boldsymbol{\mu}$ and basis matrix, $\mathbf{W}$. We find an orthonormal basis $\mathbf{W}' = orth([\mathbf{W}, \boldsymbol{\mu}])$ using singular value decomposition and, with slight abuse of notation, hereafter refer to $\mathbf{W}'$ as simply $\mathbf{W}$. Given a new (fully observed) object $\mathbf{o}$, we can obtain its embedding $\mathbf{o}'$ in this space via

$$\mathbf{o}' = \mathbf{W}^T \mathbf{o}, \tag{4.1}$$

a partially observed object will have its embedding estimated directly by HBEONet, and

any point in this space can be *back-projected* to 3D voxel space via

$$\hat{\mathbf{o}} = \mathbf{W}\mathbf{o}'. \tag{4.2}$$

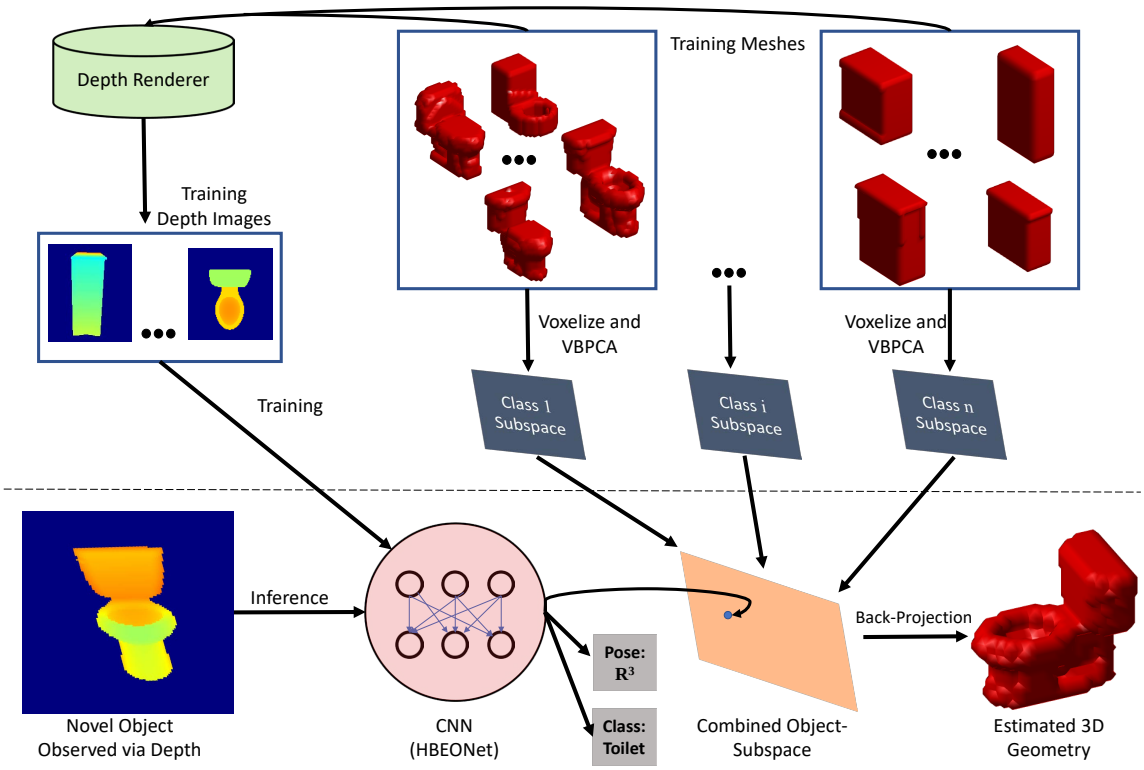
While HBEOs share the underlying subspace representation with BEOs, they have significant differences. Specifically:

- HBEOs operate directly on (segmented) input depth images.
- HBEOs use a learned non-linear mapping (HBEONet) to project novel objects onto an object subspace instead of Equation 3.19.
- HBEOs predict the subspace projection jointly with class and pose using a single forward pass through a CNN.

Figure 4.1 illustrates the complete training and inference pipeline used in HBEOs; note that portions above the dotted line correspond to training operations while the bottom area denotes inference.

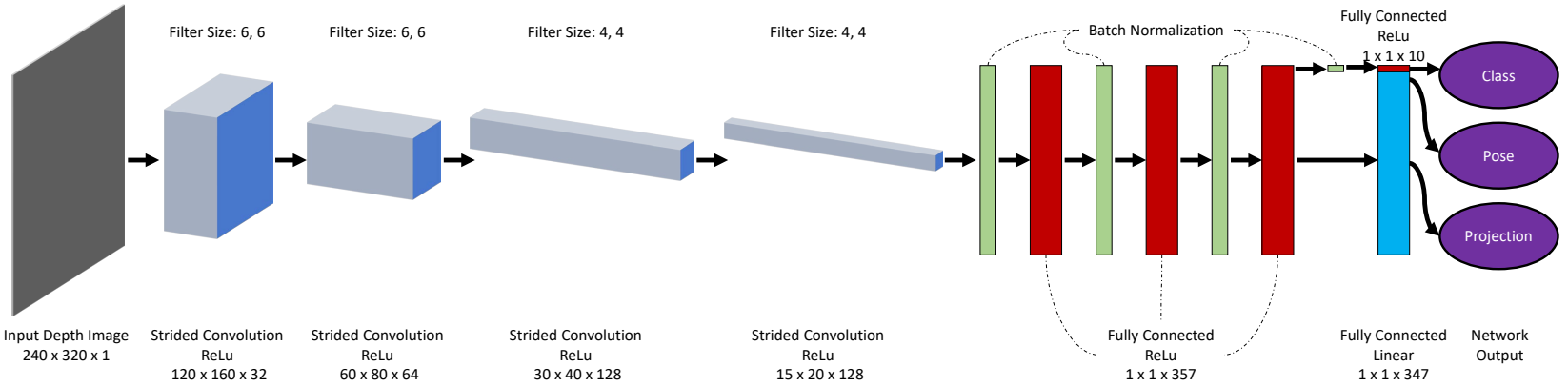
4.2.1 Learning a Projection into the Subspace

HBEOs employ a convolutional network (HBEONet) to jointly predict class, pose, and a projection into the low-dimensional subspace given a depth image. HBEONet consists of four shared strided convolutional layers followed by three shared fully connected layers with a final separated layer for classification, pose estimation, and subspace projection. Figure 4.2 provides an overview of HBEONet's structure, note that each convolution has a stride of 2x2 and pooling is not used. The non-trained softmax layer applying to the class output is not pictured. This shared architecture incentivizes the predicted class, pose, and predicted 3D geometry to be mutually consistent and ensures that learned low-level features are useful for multiple tasks. In addition to being fast, HBEOs leverage much more nuanced information during inference than BEOs. When BEOs perform object completion via Equation 3.19, each piece of object geometry treated as equally important; a voxel



HBEOs replace the projection step used in BEOs with a CNN that directly predicts BEO shape projections, class, and 3DOF pose.

FIGURE 4.1: Overview of the HBEO framework.



HBEONet layers with approximately 15 million total parameters.

FIGURE 4.2: The architecture of HBEONet.

representing the side of a toilet, for instance, is weighted equivalently to a voxel located in the toilet bowl. In reality however, some portions of geometry are more informative than others; observing a portion of toilet bowl provides more information than observing a piece of geometry on the flat side of the tank. HBEONet is able to learn that some features are far more germane to the estimated output than others, providing a significant performance increase. Because HBEONet predicts subspace-projections instead of directly producing 3D geometry (like end-to-end deep approaches), it need only produce several hundred or thousand dimensional output instead of regressing into tens or hundreds of thousands of dimensions. In this way, HBEOs combine appealing elements of both deep inference and subspace techniques.

4.2.2 Input-Output Encoding and Loss

HBEOs take a single pre-segmented depth image (such as that produced via a Kinect or RealSense sensor) at inference time and produce three output predictions: a subspace projection (a vector in $\mathbb{R}^d$), a class estimate (via softmax), and a pose estimate (via three element axis-angle encoding).

The loss function used for HBEONet is

$$\mathcal{L} = \gamma_c \mathcal{L}_c + \gamma_o \mathcal{L}_o + \gamma_p \mathcal{L}_p, \tag{4.3}$$

where $\mathcal{L}_c$, $\mathcal{L}_o$, and $\mathcal{L}_p$ represent the classification, orientation, and projection losses (respectively) and γ_c, γ_o, and γ_p weight the relative importance of each loss. Both $\mathcal{L}_O$ and $\mathcal{L}_P$ are given by Euclidean distance between the network output and target vectors while $\mathcal{L}_C$ is obtained by applying a soft-max function to the network's classification output and computing the cross-entropy between the target and soft-max output:

$$\mathcal{L}_c = -\sum_{c \in C} y_c log(\hat{y_c}) \tag{4.4}$$

where y_c is the true class label, and is 1 if the object is of class c and 0 otherwise, and $\hat{y_c}$ is the HBEONet-predicted probability (produced via softmax) that the object is of class c;

minimizing this classification loss can also be viewed as minimizing the Kullback-Leibler divergence between true labels and network predictions.

4.3 Experimental Evaluation

We evaluated the performance of HBEOs using the ModelNet10 dataset (Wu et al., 2015) as we did in the previous chapter. To obtain a shared object basis, each object mesh in ModelNet10 was voxelized to size $d = 30^3$ and then converted to vector form (i.e. each voxel object was reshaped into a $27,000$ dimensional vector). VBPCA was performed separately for each class to obtain 10 class specific subspaces, each with basis size automatically selected to capture 60 percent of variance in the training samples (equating to between 30 and 70 retained components per class). We also employed zero-mean unit-variance Gaussian distributions as regularizing hyperparameters during VBPCA. After VBPCA, the class specific subspaces were combined using SVD (via Equation 3.2.2) into a single shared subspace with 344 dimensions.

We then generated roughly 7 million synthetic depth images of size 320 by 240 from the objects in our training set by sampling multiple random viewpoints from each of the 3991 training objects. The ground truth subspace projection for each training object was obtained using Equation 4.1 and fed to HBEONet during training[1] along with the true pose and class of the object depicted in each depth image.

We compared HBEOs to vanilla BEOs as well as a baseline end-to-end deep method (3DShapeNets). An apples-to-apples comparison here is somewhat difficult; HBEOs, by their very nature, reason over possible poses due to their training regime while 3DShapeNets do not. Furthermore, BEO results in 3-DOF for combined classification and pose estimation proved to be computationally infeasible. As a result, we report 3DShapeNets results with known pose and BEO results with both known pose and 1-DOF unknown pose as

[1] HBEONet was implemented using TensorFlow 1.5 and required roughly 2 training epochs (16 hours on a single Nvidia GTX1070 GPU) to converge. The encoded and compressed depth-image dataset required roughly 200GB of storage space.

Table 4.1: ModelNet10 classification accuracy.

Queries are top-down views and accuracy is reported as a percentage.

Known Pose	Bathtub	Bed	Chair	Desk	Dresser	Monitor	Night Stand	Sofa	Table	Toilet	**Total**
BEO	48.0	**95.0**	**93.0**	**46.5**	64.0	**91.0**	**55.8**	**92.0**	**75.0**	80.0	**76.3**
3DShapeNets	**76.0**	77.0	38.0	22.1	**90.7**	74.0	38.4	57.0	1.0	79.0	54.4
Unknown Pose	Bathtub	Bed	Chair	Desk	Dresser	Monitor	Night Stand	Sofa	Table	Toilet	**Total**
BEO (1-DOF)	4.0	64.0	83.0	16.3	51.2	86.0	36.0	49.0	**76.0**	46.0	54.5
HBEO (3-DOF)	**91.3**	**86.4**	**84.1**	**57.6**	**79.7**	**97.9**	**81.3**	**75.4**	72.3	**92.3**	**81.8**

comparisons to HBEOs full 3-DOF results.

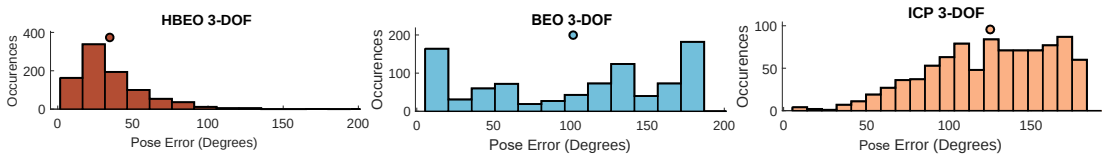
4.3.1 Classification

Despite HBEOs being required to solve a harder problem than 3DShapeNets and BEOs, classification performance was significantly better than both of them, outperforming (with unknown 3-DOF pose) 3DShapeNets and BEOs with known pose. Table 4.1 illustrates these results; HBEOs operating on data with unknown pose in 3-DOF has less than half of the misclassification rate of BEOs operating on data with only a single unknown pose DOF. One classification advantage HBEOs possess over BEOs is their ability to perform forward projection jointly with classification. Because BEOs first project into the learned subspace and then classify objects, small details which do not significantly affect shape, but are germane to determining object type, may be missed. This is particularly evident in the case of disambiguating desks, dressers, and nightstands: three classes which have similar overall shape and where small details are important for determining class. Because HBEOs learn to perform classification and subspace projection jointly, they perform much better in this scenario.

4.3.2 Pose Estimation

We evaluate the pose estimation performance of HBEOs by comparing with BEOs and an ICP baseline (see Figure 4.3). In the HBEO case, class, pose, and 3D geometry are estimated jointly as described in the preceding sections. Due to performance constraints, it was infeasable to directly compare to BEOs. Instead, we provided BEOs with the ground truth class and only required the BEO pipeline to estimate pose and 3D shape. The ICP baseline was also provided with the ground truth class and attempted to estimate pose by aligning the partial input with the mean training object in that class. Despite not having access to the input query's class, HBEOs significantly outperformed both BEOs and the baseline method, achieving a mean pose error less than half that of BEOs. Part of

the advantage HBEOs enjoy over BEOs is their direct prediction of pose; because BEOs employ pose estimation by search, they can only sample candidate poses at a relatively course resolution in 3-DOF due to computational constraints, even with known class. HBEOs do not suffer from this drawback as they predict pose parameters directly in a single inference step.



Mean error is indicated by circular dots.

FIGURE 4.3: Pose estimation error in 3-DOF.

4.3.3 3D Completion

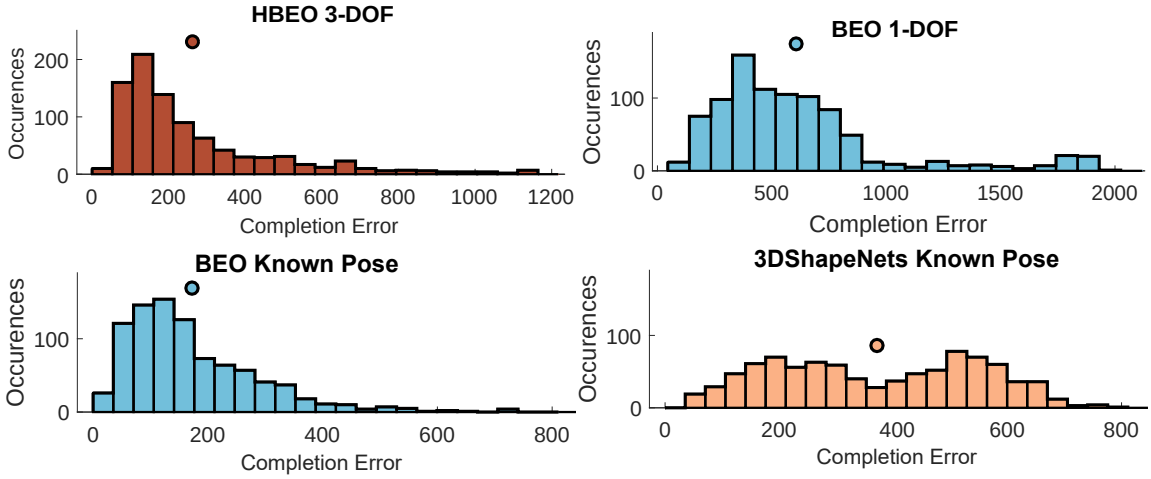


FIGURE 4.4: 3D completion error.

We follow the shape-completion evaluation methodology proposed in the previous chapter by extracting the (unsigned) Euclidean Distance Transform (EDT) (Maurer et al., 2003) from both the completed objects and the target objects. Given $\mathbf{D}$ and $\hat{\mathbf{D}}$ denoting the EDT of the target and estimated objects, our completion score is again:

$$e'(\mathbf{o}, \hat{\mathbf{o}}^{\mathbf{r}}) = ||\mathbf{D} - \hat{\mathbf{D}}||_2. \tag{4.5}$$

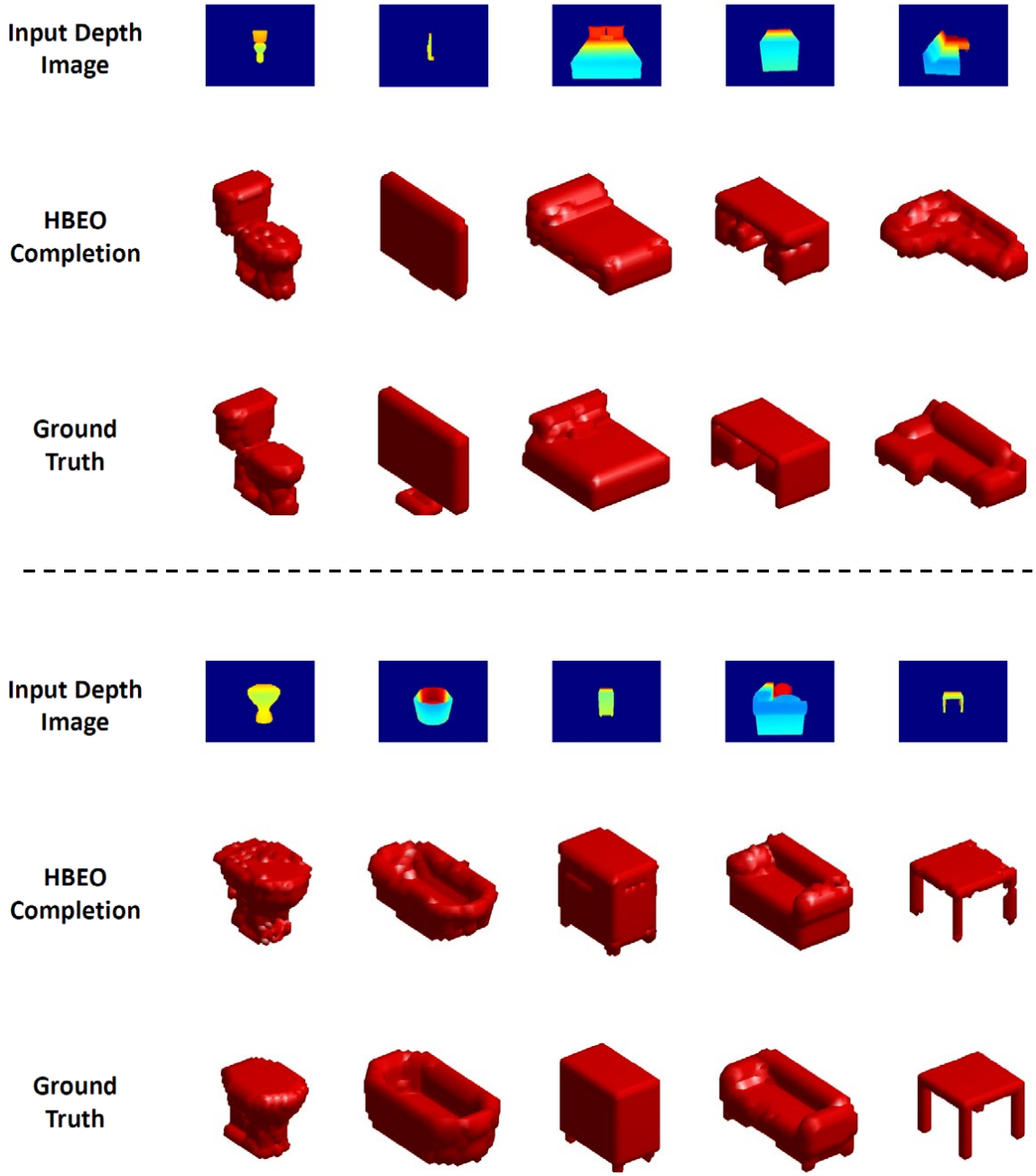


FIGURE 4.5: Sample completions from the ModelNet10 test set.

This EDT-based metric is sensitive to shape change but far less sensitive to small misalignment than Equation 3.20 which measures pairwise voxel correspondence.

Figure 4.4 illustrates the object completion performance of HBEOs relative to BEOs and 3DShapeNets, using the Euclidean Distance Metric defined in Equation 3.20. HBEOs, when performing inference directly from depth images with unknown 3-DOF pose, perform competitively with BEOs operating on perfectly voxelized and aligned input with known pose and significantly outperforms 3DShapeNets (with known pose) and BEOs (with unknown pose in a single DOF). Figure 4.5 illustrates several example HBEO object completions. In our experience, BEOs become brittle when the query object has unknown pose due to the alignment sensitivity of voxel representations. A small error in estimated pose causes a voxel representation to change dramatically while depth images change in a much smoother fashion. Because of this phenomenon, HBEO completion performance is far more robust to pose estimation errors.

4.3.4 Inference Runtime Performance

Table 4.2: Comparison of mean runtimes.

Method	Mean Runtime
3DShapeNets (Known Pose)	3.57s
BEO (Known Pose)	1.13s
BEO (1-DOF Pose)	672.97s
BEO (3-DOF Pose)	3529.88s
HBEO (3-DOF Pose)	0.01s

Comparing timing performance between methods can be difficult; differences in programming language and hardware can affect methods differently. For our testing, HBEOs were implemented in Python (with HBEONet trained using TensorFlow) while BEOs and 3DShapeNets are both implemented in Matlab. As a result, direct comparison of runtimes should be taken with a grain of salt, and small (i.e. 2x or 3x) speed differences between algorithms are not necessarily meaningful in this context. Furthermore, the HBEONet

portion of HBEOs is fully GPU accelerated while portions of 3DShapeNets and BEOs are not. Nevertheless, large speed differences (an order of magnitude or more) do highlight gross computational efficiency differences between approaches. Table 4.2 shows mean run-time performance of 3DShapeNets (for inference on objects of known pose) as well as BEOs and HBEOs. Critically, because HBEOs perform inference in a single forward pass of the network, they are able to estimate pose in 3-DOF without incurring the large speed penalty that the BEO approach of pose estimation by search produces. While the speed of BEOs can be tweaked to some degree (by adjusting the coarseness of the pose discretization), HBEOs are orders of magnitude faster in the full 3-DOF setting. HBEOs are fast enough for realtime use while BEOs—in 1-DOF or 3-DOF—are not.[2]

4.3.5 Pure Classification Evaluation

To provide context for the classification performance of HBEOs, we compared a modified version of HBEOs, with convolutional layers replaced by the convolutional portion of EfficientNet (Tan and Le, 2019), to a single-purpose depth-based classification network based on EfficentNet-b3. Both networks were identical with the exception of the final layer, which consisted of a softmax classification layer for the pure EfficentNet approach and the original HBEO output layer (used in the preceding section) for the HBEO approach. Note that because the original EfficentNet architecture is designed for RGB input, not depth, we reduced the number of input channels from three to one. At the time of writing (though not at the time of original HBEO publication), EfficentNet constitutes the state-of-the-art in pure classification performance, with the b3 version achieving over 81 percent top-1 ImageNet accuracy.

We evaluated both models on the ModelNet10 dataset used above, from both a top-down and side perspective. Ultimately, we found that the inclusion of pose and shape estimation had no significant effect on classification performance despite the vast majority of the

[2] Algorithms were evaluated on a 4-core Intel CPU with 32GB of RAM and an Nvidia GTX1070 GPU.

network being common to all three tasks in the HBEO case. As a result, we are able to perform all three tasks while retaining (though not exceeding) state-of-the-art classification performance. Table 4.3 illustrates these results.

Table 4.3: EfficientNet classification comparison.

Top-View	Classification Accuracy (percent)
EfficientNet (Tan and Le, 2019)	**83.3**
EfficientNet-HBEO	82.2
Side-View	**Classification Accuracy (percent)**
EfficientNet (Tan and Le, 2019)	86.6
EfficientNet-HBEO	**87.3**

4.3.6 Pix3D Evaluation

We also examined the shape completion and pose estimation performance of HBEOs against several RGB-based approaches on the recently released Pix3D dataset[3] (Sun et al., 2018). While this is not a like-to-like comparison with the depth-based HBEOs, it provides additional context due to the relative paucity of recent depth-based 3D completion methods. Note that Pix3D is heavily class imbalanced, with the only well-represented class consisting of chairs. As a result, performance evaluation on this dataset should be taken as somewhat of a noisy sample because methods are evaluated only on the chair class.

We trained HBEOs on the ShapeNet chair class similarly to the above experiments and evaluated on the 2894 non-occluded chairs in Pix3D. For each chair in the dataset we create a masked depth image, using the provided 3D object model, from the same perspective as the included RGB image. Table 4.5 contains the discrete pose estimation accuracy of our system while table 4.4 contains shape completion results. Because Pix3D and Render for CNN provide discrete pose predictions, we discretize the output of our system for comparison. Although HBEOs performed slightly more poorly than recent RGB-based

[3] Until the release of Pix3D in 2018, there existed no suitable dataset to compare depth-based and RGB-based shape completion and pose-estimation approaches.

Table 4.4: Pix3D shape completion performance.

Intersection over Union (IoU); higher is better.

Method	IoU
3D R2N2 (Choy et al., 2016)	0.136
3D-VAE-GAN (Wu et al., 2016)	0.171
DRC (Tulsiani et al., 2017)	0.265
MarrNet (Wu et al., 2017)	0.231
Pix3D (Sun et al., 2018)	**0.282**
HBEO	0.258

shape completion approaches, they provided significantly better pose estimates. While the causes for this performance difference are not immediately obvious, some poses may be ambiguous in 2D RGB space while more easily distinguishable using a depth image. Consider observing a chair directly from the front: it may be unclear in the RGB image if the observed surface is the chair's front or rear while depth-values trivially distinguish these two cases. It is also notable that the highest performing RGB shape completion approaches explicitly estimate object surface normals, while HBEOs do not, which may also be a contributing factor to their shape estimation performance differences. We have also noticed that BEO-based approaches seem to struggle most with objects consisting of thin structure (chairs being a particularly adversarial example) which we hypothesize is due to training-time object alignment; objects with thin details, such as chairs, are more sensitive to misalignment than objects with thicker geometry, such as cars and couches.

Table 4.5: Discretized Pix3D pose estimation performance.

Pose-bin classification accuracy (higher is better).

	Azimuth				Elevation		
Number of Bins	4	8	12	24	4	6	12
Render For CNN (Su et al., 2015b)	0.71	0.63	0.56	0.40	0.57	0.56	0.37
Pix3D (Sun et al., 2018)	0.76	0.73	0.61	0.49	0.87	0.70	0.61
HBEO	**0.87**	**0.76**	**0.69**	**0.53**	**0.96**	**0.91**	**0.71**

4.4 Discussion

Compared to their predecessors, BEOs, HBEOs are much easier to use in real-world settings because they perform inference in realtime and do not require voxelized input—instead operating directly on depth images. Fusing a learned linear-subspace with a convolutional projection module, HBEO representations retain the many of the advantageous properties of BEOs, including a low-dimensional representation and applicability to limited-data training regimes, while allowing for more robust inference with less brittleness to training-object misalignment or to pose-estimation errors. Experimentally, we found that HBEOs significantly outperformed BEOs—across all tasks—and constitute the state-of-the-art for depth-based 3D completion and pose-estimation. While on the Pix3D datataset of chairs, HBEOs do not quite achieve the shape completion performance of several RGB-based methods, their pose-estimation performance is state-of-the-art across all input modalities.

5

Probabilistic Pose Estimation with Silhouette Priors

5.1 Introduction

While HBEOs jointly predict pose and 3D object shape, they do not explicitly reason about consistency between input observations, predicted shape, and predicted pose. We propose incorporating such a consistency metric into the pose-estimation process to incentivize pose estimates which are consistent with the object's predicted shape and observed silhouette. This approach leverages the unique capabilities of unified object representations by using shape estimates to improve pose prediction. We also introduce the use of Mixture Density Networks (MDNs) for this task, both because they provide a principled way to incorporate shape-consistency into pose estimation and because MDNs are well suited to model object symmetry and artifacts endemic to our chosen axis-angle pose representation.

Recent work in category-level pose estimation consists primarily of discrete pose estimators (Sun et al., 2018; Su et al., 2015b; Fidler et al., 2012), which treat pose estimation as a classification task by predicting a single pose bucket (with width around 15 degrees). While BEOs and HBEOs produce continuous pose estimates, none of these approaches explicitly verify that their estimates of shape and pose are consistent with the observed

depth or RGB input. Our proposed approach predicts a distribution over possible object poses—from a single segmented depth-image—and constructs a pose consistency prior based on agreement between the predicted object silhouette and input image. To acomplish this, we modify the HBEO framework to produce a multimodal distribution over poses instead of a point-estimate, and develop an efficient 2D pose-consistency score by projecting shape-pose estimates back into the input image. This consistency score provides the basis of a prior probability over poses and allows the use of a sampling-based approximation to the *maximum a posteriori* estimate of 3D pose. We evaluate our approach empirically on several thousand 3D objects across three classes from the ShapeNet dataset (Chang et al., 2015) and on roughly three thousand chairs from the Pix3D dataset (Sun et al., 2018). Our results show a dramatic decrease in gross pose error ($\epsilon > 15^\circ$) compared to the previous state-of-the-art, HBEOs, and ablation analysis demonstrates a significant performance contribution from both the density prediction and input-verification components of our system. Furthermore, by dynamically varying the number of samples used to produce a pose-estimate, our approach becomes an any-time method; while it significantly outperforms existing methods with only a few samples, it produces increasingly high-quality pose estimates given a longer time budget.

5.1.1 Background: Mixture Density Networks

Mixture Density Networks (MDNs) were first proposed by Bishop (1994) as a method for predicting multimodal distributions, rather than point-estimates, when using neural networks for regression. Let x be inputs to the model and y be the desired regression target; a conventional regression network predicts $\hat{y}$ directly from x while an MDN predicts a conditional density function $P(y|x)$.

MDNs model $P(y|x)$ as a mixture of parameterized distributions with a common choice being the multivariate Gaussian mixture model (MV-GMM) with probability density

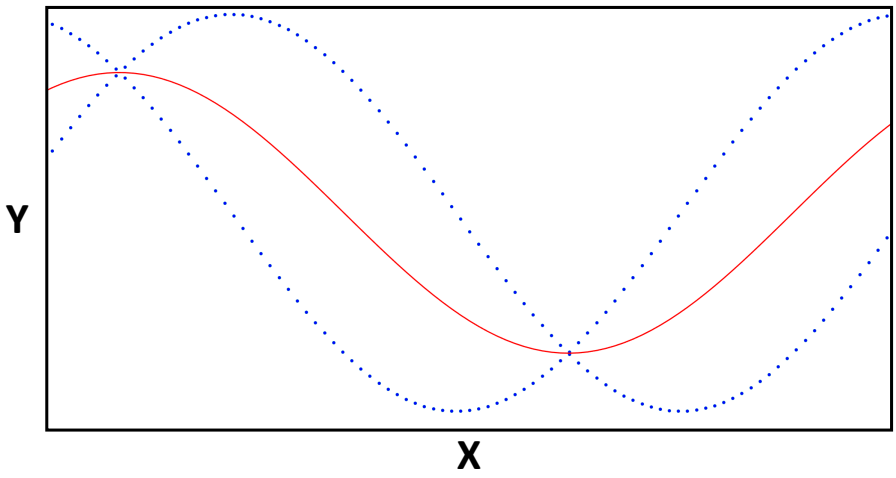
function:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \sum_{i=1}^{c} \alpha_i \mathcal{N}(y|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}), \tag{5.1}$$

where $\theta_i = \{\boldsymbol{\alpha_i}, \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}\}$ is the mixing coefficient, mean, and covariance of component i. The network predicts these mixture parameters by learning the function $\theta = f(x)$ while a conventional regression network learns $y = f(x)$. As a result, if $y \in \mathbb{R}^n$, a network that directly predicts y would have size n output while an MV-GMM-MDN would produce output of size $c(n^2 + n + 1)$, where c is the number of mixture components. To reduce the output size of MDNs, it is common to assume a diagonal covariance for each component, in which case the output size becomes $c(2n + 1)$. During training, each gradient update seeks to minimize the negative log-likelihood of observed data:

$$loss(y, \theta) = -ln \sum_{i=1}^{c} \alpha_i \mathcal{N}(y|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}). \tag{5.2}$$

MDN networks have some significant advantages over direct regression, including non-unimodality and measurable uncertainty. While direct regression learns to predict the conditional mean of the data, MDNs learn to model the shape of the underlying distribution. In contexts where the target is ill-posed (i.e. there are multiple valid mappings from x to y), the mean over good solutions may not actually be a reasonable solution itself. Figure 5.1 illustrates this in a simple univariate scenario; the function learned by direct regression with a least-squares loss is unable to accurately represent the underlying data. MDNs also allow multiple solutions to be sampled and provide a measure of confidence for each sample via the predicted conditional PDF. This explicit likelihood estimate provides a natural avenue for incorporating prior information into estimates in a straightforward and principled way.



Blue: training points. Red: $f(x)$ learned through direct regression.

FIGURE 5.1: Ill-posed regression example.

5.2 Predicting Pose Posteriors

We improve pose estimation in two primary ways: 1) predicting multimodal pose distributions instead of point-estimates and 2) incorporating a prior probability dependent on agreement between predicted pose, shape, and the input depth-image. We construct a distribution-predicting network using the MDN formulation and show how approximate maximum likelihood and *maximum a posteriori* estimates may be obtained via sampling from its output.

5.2.1 Using Predicted-Shape Priors for Predicting Pose Distributions

In order to incorporate pose priors, we require a probabilistic estimate of pose distribution conditioned on input observation. To obtain this, we modify the HBEO architecture to estimate a pose distribution by converting the final layer of the network to a multivariate Gaussian MDN layer. Let z_i be the sum of input signals to MDN-output neuron i and assume predicted components have diagonal covariance. Each covariance value is estimated using

a translated exponential-linear unit (ELU) at the final layer:

$$\sigma_{ii} = \left\{ \begin{array}{ll} z_i, & \text{if} x > 0 \\ \alpha\ exp(z_i) + \epsilon, & \text{if } x \leq 0 \end{array} \right\}.$$

This translated ELU ensures that elements along the diagonal of the predicted covariance matrix will be strictly positive and that the resulting matrix will be positive semidefinite. Component mean parameters are estimated straightforwardly with a linear layer and component mixing coefficients, $\alpha \in A$, are estimated with a softmax layer, ensuring A forms a valid discrete probability distribution:

$$\alpha_i = \frac{e^{z_i}}{\Sigma_{j=1}^{c} e^{z_j}}.$$

We fixed the number of components to $c = 5$ and used equation 5.2 to calculate the pose portion of our loss function:

$$loss_{pose}(y, \theta) = -ln \sum_{i=1}^{5} \alpha_i \frac{\exp\left(-\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu_i})^{\mathrm{T}} \boldsymbol{\Sigma_i}^{-1} (\mathbf{y} - \boldsymbol{\mu_i})\right)}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma_i}|}}, \tag{5.3}$$

where θ are the distribution parameters predicted by the network for input image x and y is the true pose of the object depicted in x. We maintain the same structure used in HBEONet for the lower layers of the network and for the class and shape output layers—softmax and linear layers, respectively. The full loss for the network is

$$\mathcal{L}(y) = \lambda_p loss_{pose}(y) + \lambda_s loss_{shape}(y) + \lambda_c loss_{class}(y) \tag{5.4}$$

where $loss_{shape}(y)$ is a Euclidean loss over subspace projection coefficients, $loss_{class}(y)$ is the multiclass categorical cross-entropy loss over possible classes, and λ_p, λ_s, λ_c are weighting coefficients over the pose, shape, and class terms.[1] Beyond allowing multiple

[1] In our experiments, we found that $\lambda_p = \lambda_s = \lambda_c = 1$ yielded good results.

possible poses to be sampled, HBEO-MDNs are more robust to training noise and object symmetry than HBEOs because they can explicitly model multiple pose modalities. Furthermore, MDNs naturally compensate for the representational discontinuity present in axis-angle formulations of pose. As an example, consider predicting only the z-axis rotation component of an object's pose. If the true pose z-component is π, and target poses are are in the range of $(-\pi, \pi]$, then the HBEO network would receive a small loss for predicting $p_z = \pi - \epsilon$ and a large loss for predicting $p_z = \pi + \epsilon$, despite the fact that both predictions are close to the true pose. While other loss functions or pose representations may alleviate this particular issue, they do so at the expense of introducing problems such as double coverage, causing the network's prediction target to no longer be well defined. By comparison, the HBEO-MDN approach suffers from none of these issues and can explicitly model object symmetry and representational discontinuities in prediction space by predicting multimodal distributions over pose.

5.2.2 Pose Priors from Shape and Segmentation

Although generative models that predict an object's 3D shape exist, those that also estimate object pose do not explicitly verify that these predictions are consistent with observed depth input and—while the shape estimate produced by such models is noisy—there is valuable information to be obtained from such a verification. Let D be a segmented depth-image input to such a model, $\hat{\mathbf{o}}$ be the predicted shape of the object present in D, and $\hat{\mathbf{R}}(\hat{\mathbf{o}})$ be the estimated 3DOF rotation transforming $\hat{\mathbf{o}}$ from canonical pose to the pose depicted in D. Assuming known depth camera intrinsic parameters, we can project the estimated shape and pose of the object back into a 2D depth-image via $\hat{D_R} = f(\hat{\mathbf{R}}(\hat{\mathbf{o}}))$ where the projection function $f(x)$ simulates a depth camera. Intuitively, if the shape and pose of the observed object are correctly estimated, and the segmentation and camera intrinsics are accurate, then $\Delta_D = ||D - \hat{D_R}|| = 0$ while errors in these estimates will result in a discrepancy between the predicted and observed depth-images. As prior work has shown pose to be the

least reliable part of the pipeline (Sun et al., 2018), we assume that error in $\hat{R}$ will generally dominate error in the other portions of the process and thus employ Δ_D to refine $\hat{R}$.

Let $T = SDF(D)$ be the 2D *signed distance field* (Osher and Fedkiw, 2003) calculated from D, we define an image-space error score between segmented depth-images as

$$e_R = ||SDF(D) - SDF(\hat{D_R})||_f \tag{5.5}$$

where $SDF(D)$ considers all non-masked depth values to be part of the object and $||\cdot||_f$ denotes the Frobenius norm. Figure 5.2 illustrates the masked depth input and SDF for an example object: the first column denotes the true 3D object (top), observed depth-image (middle) and resulting SDF (bottom) while the second and third columns depict estimated 3D object shape, depth-image, and resulting SDF. Note that the SDF corresponding to an accurate pose-estimate closely matches that of the observed input while the poor estimate does not. The calculation of error in depth-image-space has several advantages; because it operates in 2D image space, distance fields are both more efficient to calculate than in the 3D case and better defined because the observed input is partially occluded in 3D but fully observable from the single 2D perspective of the camera. Furthermore, by using the SDF instead of raw depth values, our error gains some robustness to sensor noise and minor errors in predicted shape. To transform this error into a pose prior, we take the quartic-normalized inverse of the score, producing the density function

$$p_{prior}(\hat{R}) = \frac{1}{e_R^4 + \epsilon}. \tag{5.6}$$

5.2.3 Sampling Pose Estimates

It is possible to obtain approximate maximum likelihood (MLE) and *maximum a posteriori* (MAP) pose estimates by sampling from the pose distribution induced by the predicted θ. Let $\mathbf{R}$ denote the set of n pose estimates sampled the HBEO-MDN network and $R_i \in \mathbf{R}$ be

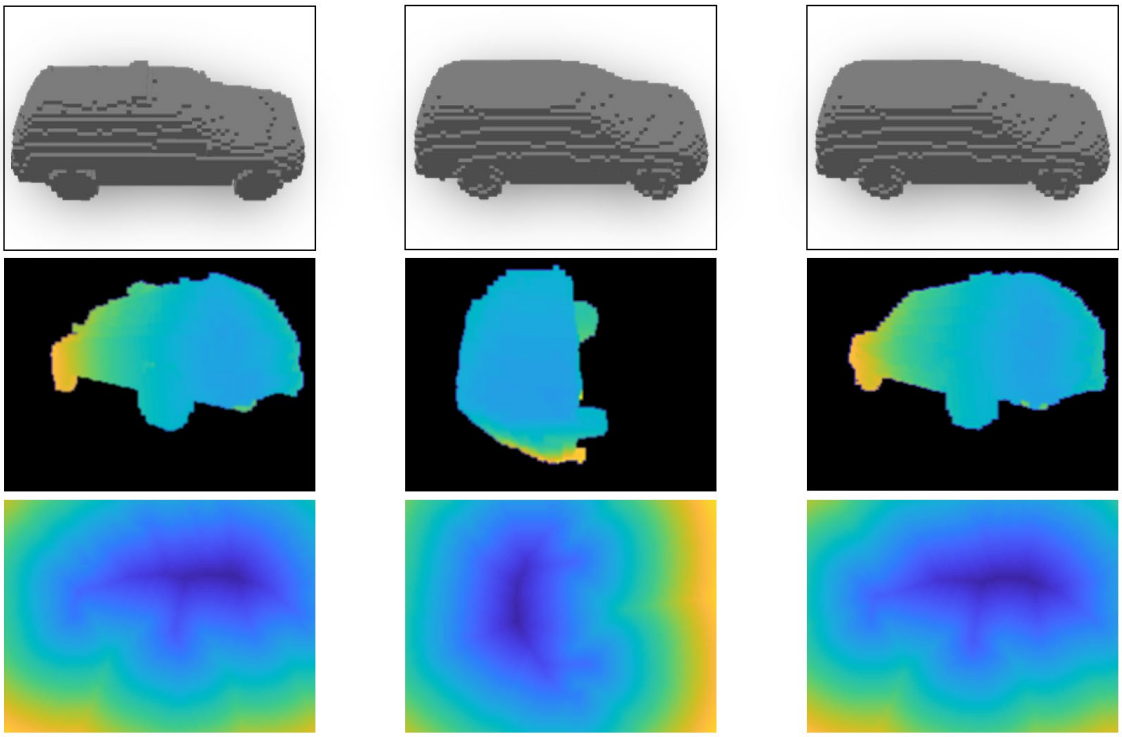


FIGURE 5.2: Example output of HBEO-MDN net evaluated on a car.

a single sampled pose. From equation 5.1, the approximate MLE pose estimate is

$$\hat{R}_{MLE} = argmax_{R_i \in \mathbf{R}} \sum_{i=1}^{c} \alpha_i \mathcal{N}(R_i | \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) \tag{5.7}$$

while incorporating equation 5.6 produces an approximate MAP pose estimate of

$$\hat{R}_{MAP} = argmax_{R_i \in \mathbf{R}} \frac{1}{e^4_{R_i} + \epsilon} \sum_{i=1}^{c} \alpha_i \mathcal{N}(R_i | \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}). \tag{5.8}$$

As $n \rightarrow \infty$, equations 5.7 and 5.8 approach the true MLE and MAP pose estimates, respectively. As a result, HBEO-MDN is a variable-time method for pose estimation, with prediction accuracy improving as computation time increases.

5.3 Experimental Evaluation

We evaluated our approach via an ablation analysis on three datasets consisting of cars, planes, and couches taken from ShapeNet (Chang et al., 2015) for a total of 6659 training objects and 3098 test objects. We also compared to two RGB-based approaches on the Pix3D chair dataset. Depth-based approaches were provided with segmented depth-image input and RGB-based approaches were given tight bounding boxes around objects; in the wild, these segmentations could be estimated using dense semantic segmentation such as MASK-RCNN (He et al., 2017). For the ablation experiments, HBEOs and HBEO-MDNs were trained for each category of object with 2798 couches, 2986 planes, and 875 cars used. During training, depth-images from random views[2] were generated for each object for a total of $2.7M$ training images. Evaluation datasets were constructed for each class containing 1500 views from 50 cars, 2300 views from 947 planes, and 2101 views from 368 couches. The HBEO and HBEO-MDN models used identical subspaces of size $d = 300$ for each object class, predicted size 64^3 voxel objects, and were trained for 25 epochs (both models converged at similar rates).[3]

We examined two forms of HBEO-MDN, an ablation that used the MLE approximation from equation 5.7 (HBEO-MDN Likelihood) and the full method which uses the posterior approximation defined in equation 5.8 (HBEO-MDN Posterior). The performance of HBEO-MDN Likelihood illustrates the contribution of the MDN portion of our approach while HBEO-MDN Posterior shows the efficacy of explicitly verifying possible solutions against the observed depth-image. To ablate the impact of the generative portion of our model, we also evaluated two baselines, Random Sample + Oracle, which uniformly sampled poses from $SO(3)$ and was provided with an oracle to determine which of the sampled poses was closest to ground truth, and Random Sample + SDF Error, which

[2] Azimuth and elevation were sampled across the full range of possible angles while roll was sampled from 0-mean Gaussian distribution with 99-percent mass within the range $[-25°, 25°]$.

[3] Models were trained using the Adam optimizer with $\alpha = 0.001$ and evaluated on an Nvidia 1080ti GPU.

Table 5.1: ShapeNet pose estimation performance—mean error and runtime

		Mean Angular Error (Lower is Better)			
Method	N	Car	Plane	Couch	Runtime
HBEO-MDN Likelihood	5	6.92°	11.43°	10.84°	0.07s
HBEO-MDN Likelihood	25	6.10°	10.81°	10.77°	0.13s
HBEO-MDN Likelihood	100	6.26°	10.64°	10.80°	0.36s
HBEO-MDN Posterior	5	4.83°	9.77°	9.59°	0.12s
HBEO-MDN Posterior	25	3.70°	8.45°	8.12°	0.32s
HBEO-MDN Posterior	100	**3.23**°	**8.06**°	**7.50**°	1.06s
HBEO	N/A	9.45°	13.29°	18.84°	**0.01**s

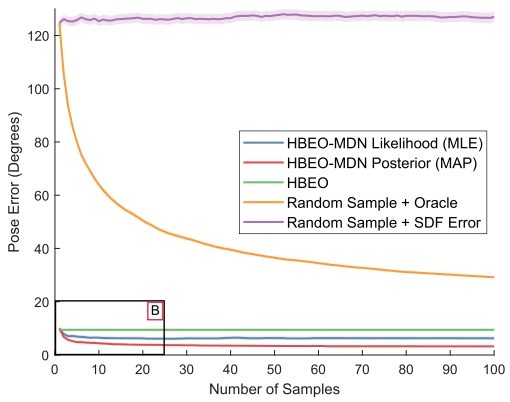
Table 5.2: ShapeNet pose estimation performance—gross-error incidence rate

		Error > 15° (Lower is Better)		
Method	N	Car	Plane	Couch
HBEO-MDN Likelihood	5	3.73 %	10.87 %	9.57 %
HBEO-MDN Likelihood	25	2.67 %	9.78 %	9.09 %
HBEO-MDN Likelihood	100	2.80 %	9.65 %	8.95 %
HBEO-MDN Posterior	5	1.80 %	8.13 %	8.09 %
HBEO-MDN Posterior	25	1.20 %	6.22 %	6.38 %
HBEO-MDN Posterior	100	**0.93** %	**5.78** %	**5.95** %
HBEO	N/A	6.87 %	13.57 %	31.27 %

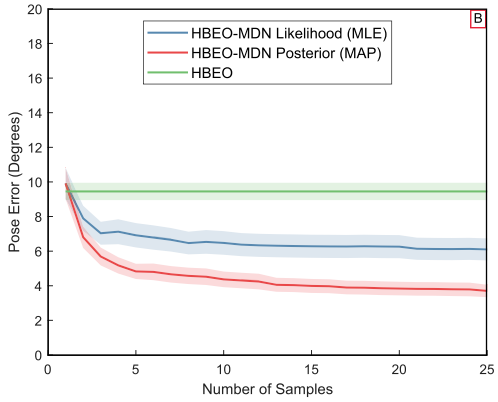
uniformly sampled poses from $SO(3)$ and used equation 5.5 to select a single pose estimate from these samples.

Figure 5.3 gives the performance of the approach across all three ShapeNet evaluation datasets as a function of the number of samples used, note that HBEOs do not sample solutions and thus produce a line with slope of zero. Error ranges indicate the 95 percent confidence estimate of the mean. Table 5.1 contains performance and inference-time at various sampling values while Table 5.2 contains the frequency of large pose errors of at least 15 degrees. HBEO-MDN Posterior substantially outperformed other approaches, even with a limited sample size.

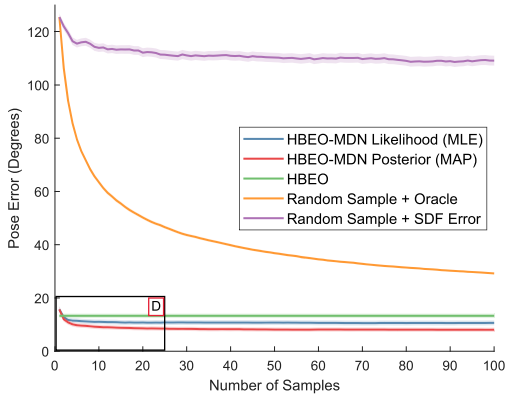
HBEOs, while having generally inferior performance to both HBEO-MDN varieties, struggled most significantly on the couch dataset. We hypothesize that this is due to the



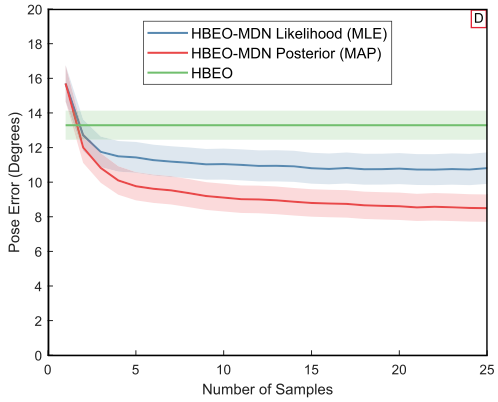
a: Car Dataset



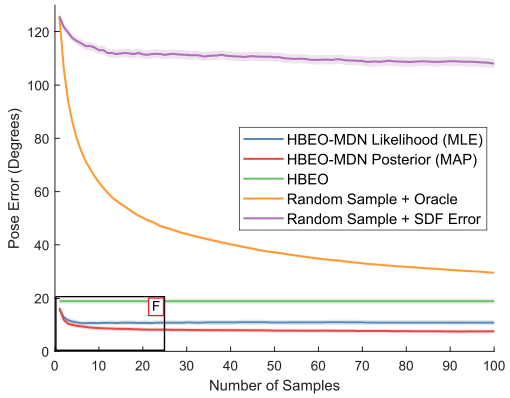
b: Car Dataset (Enlarged)



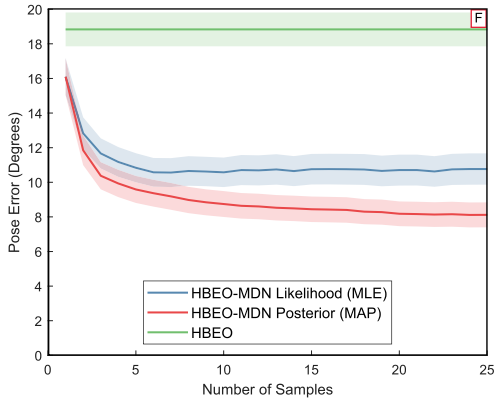
c: Plane Dataset



d: Plane Dataset (Enlarged)



e: Couch Dataset



f: Couch Dataset (Enlarged)

FIGURE 5.3: Mean pose error.

symmetry present in couches when viewed from the side, where only small aspects of the image disambiguate left from right. Because the MDN approaches can predict multimodal pose estimates, they can better model this symmetry. Interestingly, our baseline of Sample + SDF Error performed extremely poorly and only slightly better than selecting a pose estimate at random. It appears that because shape error is computed using predicted object shape, and not ground truth, it is too noisy of a signal to be directly useful—it can disambiguate between highly probable pose-candidates but is not suitable for disambiguating between arbitrary poses. Furthermore, while pose estimation with HBEO-MDNs is slower than with HBEOs, the most expensive HBEO-MDN Posterior method can still produce multiple predictions a second; if more time is available, a higher quality estimate can be obtained by increasing the number of evaluated samples. For a fixed time budget, the Posterior version of HBEO-MDNs outperformed the Likelihood variety, even though approximating the MAP will necessitate using fewer samples than the MLE.

We also compared against two RGB-based category-level pose estimation methods, Pix3D (Sun et al., 2018) and Render For CNN (Su et al., 2015b) on the Pix3D dataset. All methods were trained on the ShapeNet chair category and evaluated on the 2894 non-occluded chairs in Pix3D; Pix3D and Render for CNN used RGB images as their input while HBEOs and HBEO-MDNs were provided rendered depth-images from the same viewpoint. Because Pix3D and Render for CNN produce discretized poses, the output of HBEO and HBEO-MDN was discretized for comparison. Table 5.3 contains these results for multiple levels of discretization (a larger number of possible views equates to requiring more pose-estimation accuracy and the entries in the table indicate the proportion of objects with correctly estimated pose-bin). Our full method, HBEO-MDN-Posterior[4] achieved the best performance, along both azimuth and elevation, of all models when the number of bins was high, with HBEOs performing competitively as the bin size became larger. Interestingly, HBEOs slightly outperformed HBEO-MDN with very coarse bins

[4] HBEO-MDN variants utilized 100 samples.

Table 5.3: Discretized Pix3D pose estimation performance.

Pose-bin classification accuracy (higher is better).

	Azimuth				Elevation		
Number of Bins	4	8	12	24	4	6	12
Render For CNN (Su et al., 2015b)	0.71	0.63	0.56	0.40	0.57	0.56	0.37
Pix3D (Sun et al., 2018)	0.76	0.73	0.61	0.49	0.87	0.70	0.61
HBEO	**0.87**	0.76	0.69	0.53	0.96	0.91	0.71
HBEO-MDN-Likelihood	0.78	0.73	0.70	0.59	**0.97**	**0.93**	**0.75**
HBEO-MDN-Posterior	0.80	**0.76**	**0.73**	**0.62**	**0.97**	**0.93**	**0.75**

(90°) which we hypothesize is due to chair symmetry. While chairs are highly asymmetrical vertically, some variants of chairs lack arms and are thus fairly symmetrical rotationally. Because HBEOs learn to pick the average of good solutions, their predictions may be more likely to fall within 90 degrees of the true solution than HBEO-MDNs—which will tend to predict a mode of the distribution instead of the mean. This is primarily an artifact of using a sampling strategy to select a single pose instead of evaluating the entire MDN-predicted pose distribution.

5.4 Discussion

We found that explicitly incorporating consistency between observations, predicted 3D shape, and estimated pose provided significant pose-estimation performance. We also discovered that modeling pose via a multimodal distribution instead of a point-estimate significantly improved the reliability of our system, with only a moderate computational cost. Empirically, HBEO-MDNs significantly improved on the existing state-of-the-art, providing a significant reduction in average-case pose error and incidence of catastrophic pose-estimation failure. Furthermore, because we employ a sampling method to obtain a final estimate from this distribution, the algorithm becomes variable time and our experimental analysis suggests that in most cases, only a very small number of samples—on the order of three or four—need be obtained to outperform a point-estimate producing

approach. With additionally optimization of the sampling method employed, specifically batching the sampling operation, it should be possible to produce a reasonable number of pose samples with virtually zero computational overhead compared to producing a single point estimate. While our 2D input-pose-shape consistency prior does require some calculations, if time is highly constrained, we show that sampling an approximate maximum likelihood pose estimate–with no consistency prior—still outperforms direct regression.

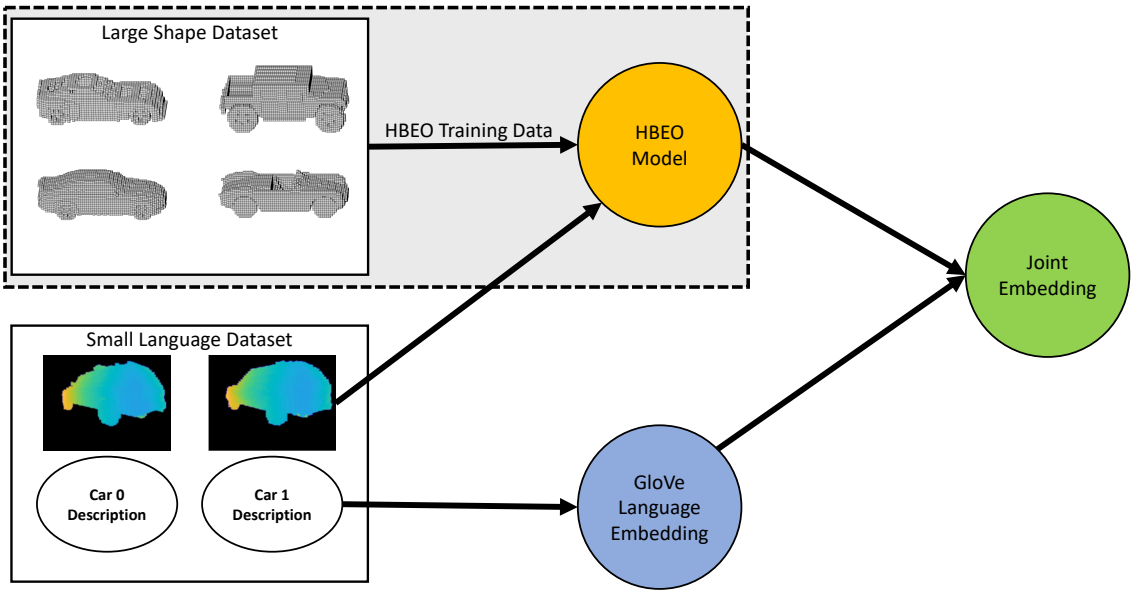
6

Conclusion

In this chapter, we discuss future work in object-centric robot perception and a recent application of BEOs natural language grounding. We begin by presenting a collaborative effort to ground natural language object descriptions to partially observed 3D shape (via depth images); by using BEOs as the underlying object representation, we were able to train our system successfully with only a small amount of language data describing depth images obtained from a single viewpoint. We also discuss future work: incorporation of multimodal input—including both RGB and depth images, joint segmentation prediction, refinement of perceptual estimates with multiple observations, detection of perceptual failure, and modeling arbitrary articulated and deformable objects.

6.1 Example BEO Application: Grounding Natural Language Descriptions to Object Shape

As robots grow increasingly capable of understanding and interacting with objects in their environments, a key bottleneck to widespread robot deployment in human-centric environments is the ability for non-domain experts to communicate with robots. One of the most sought after communication modalities is natural language, allowing a non-expert user



HBEOs are trained on non-language annotated data—pictured in the grey box—and the learned HBEO shape embedding is then used as a viewpoint-invariant compact representation of 3D shape.

FIGURE 6.1: An overview of our language grounding system.

to verbally issue directives. In collaboration with several colleagues at Brown University (Cohen et al., 2019),[1] we apply natural language to the task of object-specification—indicating which of several objects is being referred to by a user. This task is critically important when tasking a robot to perform actions such as retrieving a desired item.

Our system grounds natural-language object descriptions to object instances by combining HBEOs with a language embedding (Pennington et al., 2014); this coupling is achieved via a Siamese network (Bromley et al., 1994) which produces a joint embedding space for both shape and language. As a result, we are able to train the object-understanding portion of our system from a large set of non-language-annotated objects, reducing the need for expensive human-generated object attribute labels to be obtained for all the training data. Additionally, because the language model learns to predict language groundings from a

[1] Primary authorship was shared between Vanya Cohen, myself, and Thao Nguyen. My primary contribution to the work was constructing a viewpoint-invariant object representation using HBEOs and integrating the perceptual pipeline into the language portion of the system.

low-dimensional shape representation, instead of high-dimensional 2.5D or 3D input, the complexity of the language model—and amount of labeled training data required—is small. Finally, unlike a single monolithic system which would require human-annotated depth-images from all possible viewpoints, our approach allows a small number of annotated depth-images from a limited set of viewpoints to generalize to significantly novel partial views and novel objects.

We evaluate our system on a dataset of several thousand ShapeNet (Chang et al., 2015) objects across three classes (1250 couches, 3405 cars, and 4044 planes),[2] paired with human-generated object descriptions obtained from Amazon Mechanical Turk (AMT). We show that not only is our system able to distinguish between objects of the same class, even when objects are only observed from partial views. In a second experiment, we train our language model with depth-images obtained only from the front of objects and can successfully predict attributes given test depth-images taken from rear viewpoints. This view-invariance is a key property afforded by our use of an explicitly learned 3D representation—monolithic end-to-end depth to language approaches are not capable of handling this scenario. We demonstrate a Baxter robot successfully determining which object to pick based on a Microsoft Kinect depth-image of several candidate objects and a simple natural language description of the desired object as shown in Figure 6.3. For details our language model and some additional experimental results, please see the full paper (Cohen et al., 2019).

6.1.1 Learning a Joint Language and Shape Model

Our objective is to disambiguate between objects based on depth-images and natural language descriptions. The naive approach would be to directly predict an object depth-image given the object's natural language description, or vice versa. Such an approach would require language and depth-image pairs with a large amount of viewpoint coverage,

[2] We used a 70% training, 15% development, and 15% testing split.

an unreasonable task given the difficulty of collecting rich human-annotated descriptions of objects. Instead, we separate the language-modeling portion of our system from the shape-modeling portion. Our approach learns to reason about 3D structure from non-annotated 3D models—using HBEOs—to learn a viewpoint-invariant representation, of object shape. We combine this representation with a small set of language data to enable object-language reasoning.

Given a natural language phrase and a segmented depth-image, our system maps the depth-image into a compact viewpoint-invariant object representation and then produces a joint embedding: both the phrase and the object representation are embedded into a shared low-dimensional space. During training, we force this shared space to co-locate depth-image and natural language descriptions that correspond to each other while disparate pairs will embed further apart in the space. During inference, we compute similarity in this joint space between the input object-description and candidate 3D objects (observed via depth images) to find the nearby object that most closely matches the given description. This permits a larger corpus of 3D shape data to be used with a small set of human-annotated data. Because the HBEO module produces viewpoint-invariant shape predictions predictions from a single depth-image, the human annotations need not label entire 3D objects, but could instead be collected on images for which no true 3D model is known. These annotations could also be from only a very limited set of views because the HBEO shape representation provides for generalization across viewpoints. For our experiments, we used three classes of objects from ShapeNet: couches, cars, and airplanes and we collected object natural language text descriptions through Amazon Mechanical Turk (AMT).

6.1.2 Language Grounding Experiments and Results

We evaluate the ability of our system to retrieve the requested object—specified via a natural language description—from a pool of three possible candidates and show results

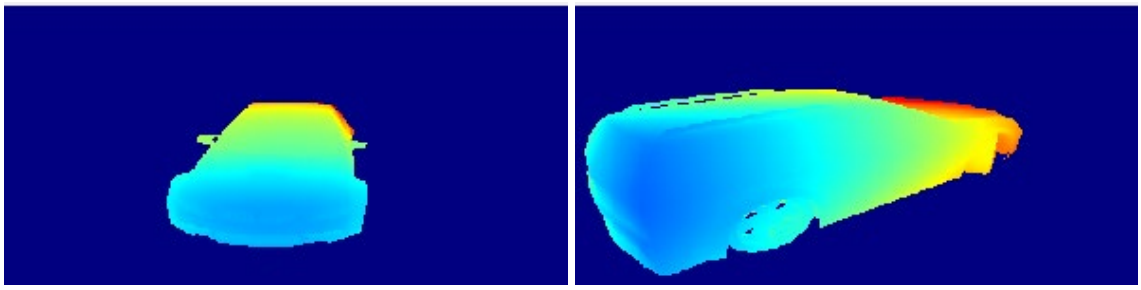


FIGURE 6.2: View-transfer experiment example.

for three training conditions: 1) *full-view*: a baseline where the language portion of the system is given a ground-truth 3D model for the object it is observing, 2) *partial-view*: the scenario where the system is trained and evaluated with synthetic depth images over a wide variety of possible viewpoints, 3) *view-transfer*: the system is identical to the previous *partial-view* case, except all training images come from a frontal viewpoint and all evaluation images are obtained from a side-rear view. In all experiments, the input HBEO embeddings of our network were trained directly from all meshes in the training dataset while language-descriptions were generated for a subset of these objects. In the partial-view and view-transfer cases, HBEONet was trained using 600 synthetically rendered depth-images, across a variety of viewpoints, from each 3D mesh.

Object Retrieval from Natural Language

To evaluate the retrieval performance of our model, we randomly selected a set of 10 depth-images for each object in our test set. Note that the full-view case used the 3D model for each object in the test set to provide an oracle-generated BEO projection for that object. In each retrieval test, we showed the system three different objects along with a natural language description of one of those objects and we report the resulting object-retrieval accuracy of our system in Table 6.1. We also evaluated the robustness of our method to substantial viewpoint differences between testing and training data by training our language model with only frontal views while evaluating model performance based only on rear-side views. Figure 6.2 shows an example training (left) and testing (right) depth-image from

our view-transfer experiment for a car instance; the underlying HBEO representation maps these substantially different viewpoints to the similar locations in the HBEO subspace.

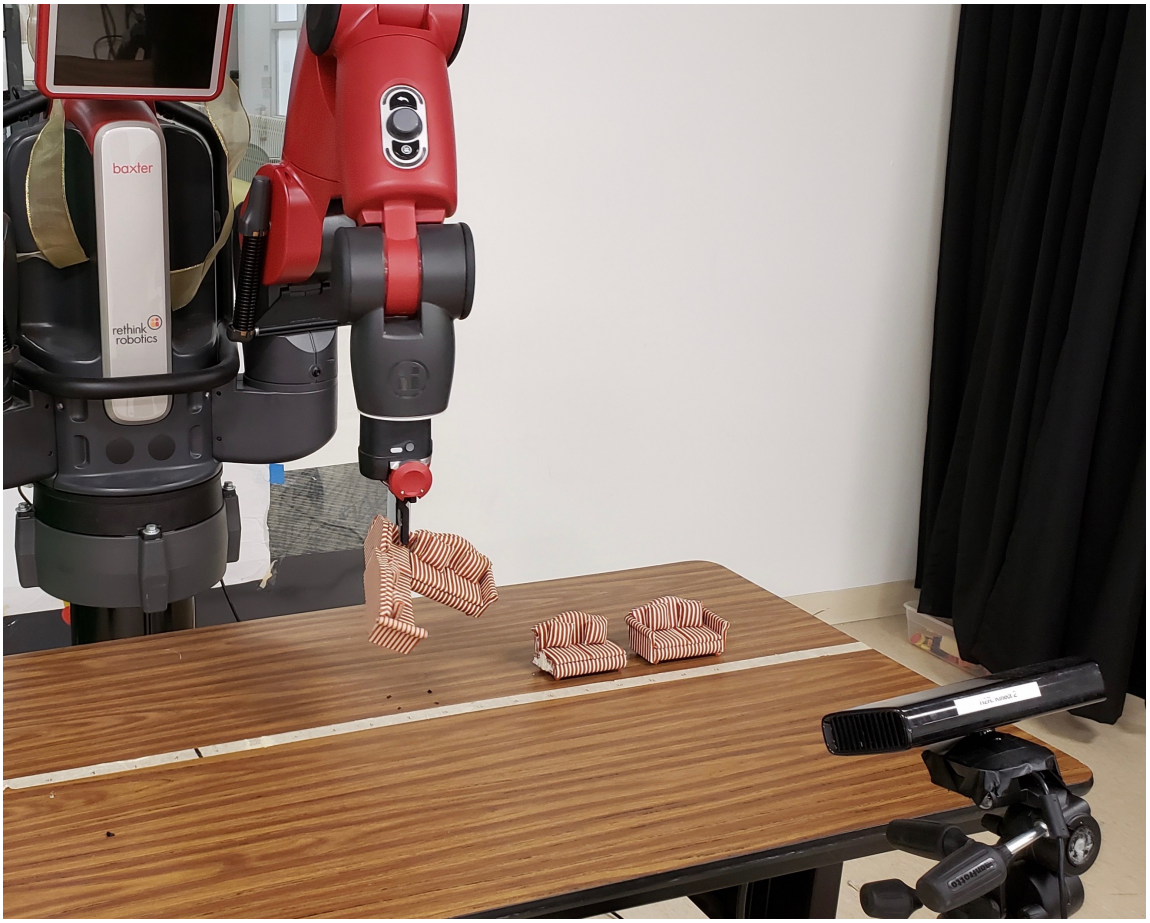
Table 6.1: Object Retrieval Results (Percent Correct)

Object Class	Full-view		Partial-view		View-transfer		Human-baseline
	Top-1	***Top-2***	***Top-1***	***Top-2***	***Top-1***	***Top-2***	***Top-1***
Couch	60.6	87.1	58.6	86.3	57.6	85.1	77.3
Car	74.6	93.8	73.5	93.4	72.2	93.1	76.0
Airplane	66.7	92.8	67.0	92.5	68.3	92.7	69.3

We also compare the performance of our system to a human baseline for the retrieval task. Humans are expert symbol grounders and are able to ground objects from incomplete descriptions rather well from an early age (Rakoczy et al., 2005). We showed 300 human users (via AMT) three objects and one language description, where the language was collected from AMT for one of the objects shown, and asked them to pick the object to which the language refers. These results are also shown in Table 6.1. We found human performance to be similar to our system's top-1 retrieval accuracy.

6.1.3 Picking Objects from Depth Observations and Natural Language Descriptions

We implemented our system on a Baxter robot: a mechanically compliant robot equipped with two parallel grippers. For this evaluation, we obtained realistic model couches (designed for use in doll houses) to serve as our test objects and used the same (synthetically trained) network used in the prior experiments, without retraining it explicitly on Kinect-generated depth-images. We passed a textual language description of the requested object into our model along with a manually-segmented and scaled Kinect-captured depth-image of each object in the scene. The robot then selected the observed object with the highest cosine-similarity with the language description and performed a pick action on it. Our system successfully picked up desired objects using phrases such as "Pick up the couch with no arms".



The system receives object depth images, the natural language command "Pick up the bent couch", and correctly retrieves the described couch.

FIGURE 6.3: Our language grounding system on the Baxter robot.

Additional Remarks

Our system was able to ground natural language descriptions to physical objects observed via depth image—with close to human-level performance in some instances and with only a limited amount of language data obtained from a restricted viewpoint—because our approach decoupled 3D shape understanding from language grounding. By using HBEOs to learn about the relationship between partially observed objects and their full 3D structure before introducing the language-grounding problem, the language portion of our system did not have to learn to reason in 3D, only to ground shape to a low-dimensional

feature vector. We believe this application serves as a model for how many robot tasks can be simplified given a general-purpose perceptual system; allow the perception system to learn a universally useful object representation and then use that representation for specific applications instead of retraining the entire system from scratch.

6.2 Future Work

While BEOs and their extensions form the basis of a general object-centric robot perception system, significant work is still required to enable fully robust perception. In this section, we discuss some of the promising avenues of exploration for further advancements in the field.

6.2.1 Multimodal Input

While existing object-centric perception systems typically utilize a single input modality, real robots are equipped with a variety of sensors which could be leveraged to improve perceptual accuracy and reliability. Some of the most commonly encountered sensor types include depth sensors, RGB cameras, event-based cameras, ultrasonic sensors, and tactile sensors. A particular challenge of multi-modal input is versatility—ideally a perceptual system should be able to make predictions based on any subset of the sensing modalities it is trained upon; a perception module that requires ultrasonic input, for example, will be useless on the large number of robots not equipped with such a sensor. Further complicating matters, in an environment where objects may move, temporally aligning sensors that capture data at different refresh rates is difficult—even before accounting for the effects of rolling or global shutters and synchronization of the sensors. One straightforward approach is to train individual perceptual systems for each desired sensing modality and then fuse the output, possibly via ensemble methods. This naive method has drawbacks however: a significant amount of training data is required for each type of sensor and sensing sub-modules are unable to share learned features between themselves. How to optimally fuse

multiple input modalities thus remains an open, and critically important, question.

6.2.2 Joint Segmentation

Current object-centric perception assumes the existence of a segmentation algorithm to pre-process input, either in the form of a pixel-level mask or an object bounding box. While recent years have seen significant advancements in such methods (He et al., 2017), segmentation is generally treated as a black box, despite having significant relevance to physical object characteristics. In the future, segmentation should be incorporated into object representations: given non-segmented input, image-level segmentation masks should be jointly estimated along with object shape and pose. This approach would ensure that shape, pose, and segmentation estimates are consistent, extending the approach taken by HBEO-MDNs. While HBEO-MDNs ensure that pose estimates are consistent with predicted shape and 2D masks, a fully joint method would jointly predict all three aspects.

6.2.3 Refining Estimates via Multiple Observations

Robots operating in the physical world obtain observations over a continuous period of time and generally—if the robot is in motion—from multiple viewpoints. To fully take advantage of this, perception systems should aggregate belief over time, from various viewpoints. Possible avenues of exploration include LSTM-based networks (Hochreiter and Schmidhuber, 1997), visual-attention-based transformer networks (Girdhar et al., 2019), and 3D convolutional networks, all of which are well-situated to reason about time-series data. Filtering approaches, such as Bayesian filters (Särkkä, 2013) could also be useful in this setting as they can naturally incorporate non-uniform prediction confidences, lending more weight to more certain observations. If objects besides the robot are moving in the scene, issues of observation correspondence also present themselves; future object-centric perceptual systems will have to either implicitly or explicitly estimate object associations across spatio-temporally separate observations.

6.2.4 Detecting Perception Failure

Because no perception system achieves perfect reliability, it is important to explicitly consider failure modes. In robotics particularly, explicitly reasoning about uncertainty is valuable; in the case of high-uncertainty, planning can be made more conservative and actions can be taken to gather more information. Recently, novelty-detection methods have been proposed to detect classification system failure (Hendrycks and Gimpel, 2016; Lee et al., 2018); The most straightforward of these methods examines the belief distribution produced by a classification system and labels relatively uniform distributions as novel cases while belief distributions with large density concentrations are determined to be of known class. A simple initial formulation of this is as follows: let C be a discrete probability distribution, over k possible classes, produced by the classification module where c_i is the predicted probability for class i. $H(C)$ is the entropy of this distribution:

$$H(C) = \sum_{i=0}^{k-1} c_i log \frac{1}{c_i},$$

allowing classifier output to be thresholded such that

$$object\_class = \begin{cases} \textbf{argmax}_\mathbf{i}\ \mathbf{c_i} \in \mathbf{C}, & \text{if } H(C) < \alpha \\ \textbf{unknown}, & \text{otherwise.} \end{cases}$$

While this particular approach is only applicable to classification tasks, and not regression, other methods exist, such as Bayesian Neural Networks (Neal, 2012) that are capable of providing such confidence estimates. Unfortunately, while such theoretical tools exist, their performance on real systems has tended to lag behind their theoretical promise and a silver bullet for quantifying prediction uncertainty remains elusive. A lack of prediction stability over time could also be a useful indicator of inference failure. If the system's belief about an objects class, shape, or pose is changing dramatically over time, it is a good indicator that the model has not produced a reliable prediction. While particular failure mode is only

one of multiple possible types of perceptual failure, it may still provide a useful basis for improving perceptual reliability.

6.2.5 Modeling Articulated and Deformable Objects

Many objects in the real world are not rigid; there are many virtually ubiquitous items in human homes—such as cabinets, microwaves, doors, pillows, and blankets—that are not well represented without modeling articulation or deformation. While this problem has been studied in the context of particular object classes such as human bodies (Ramakrishna et al., 2014) and human hands (Tompson et al., 2014; Carley and Tomasi, 2015), these existing methods can only predict the parameters of a parametric object model and cannot autonomously create such a model. Other work exists which segments an object into multiple rigid articulating parts, based on multiple input meshes corresponding to that object in several configurations (Anguelov et al., 2004) or RGBD images (Katz et al., 2013), but these approaches are either unable to generalize to novel objects or do not reason about occluded object geometry. Eventually, object-centric robot perception must be capable of discovering, through multiple observations and possibly interaction, that a particular object is articulated or deformable and once such a property has been discovered, the underlying object representation must be general enough to allow modeling of this articulation or deformation.

6.3 Final Remarks

This work presents a novel framework for representing 3D objects that is designed to be the foundation of a general-purpose object-centric robot perception system. We proposed our first contribution, BEOs, as a way to reduce the dimensionality of 3D completion. Furthermore, we hypothesised that performing classification, pose-estimation, and 3D completion jointly made sense, from both a computational efficiency perspective and a performance standpoint, as these tasks are highly interrelated. We found that the BEO

approach was data-efficient, able to learn from as few as 20 example objects, and scaled to high resolutions. We also demonstrated significant 3D shape completion improvements from the current state-of-the-art.

While BEOs were a step towards general object-centric 3D perception, they did not fully satisfy all of the requirements for a useful robotic perception system. Critically, BEOs struggled with pose-estimation in greater than one degree of freedom and required partially-observed input to be voxelized. Taking inspiration from the increasing performance CNN-based shape completion approaches, we extended BEOs to employ a non-linear convolutional projection module, creating HBEOs (Burchfiel and Konidaris, 2018). HBEOs retained the linear object subspace described in BEOs, but replaced the analytical projection step with a CNN capable of estimating class, pose, and a BEO subspace projection from a single input depth image. HBEOs exhibited higher performance than in every metric we examined, achieving state-of-the-art for depth-based methods in 3D completion, category-level pose estimation, and classification. Crucially, HBEOs are able to perform inference in realtime (roughly 100hz)—ensuring they are fast enough to not become a bottleneck when running on a real system. We then proposed to explicitly incentivize consistency between an observed depth image and and the depicted object's estimated 3D shape and pose, more fully taking advantage of the close relationship between an object's shape and pose, and an observation of that object. The resulting system, HBEO-MDN (Burchfiel and Konidaris, 2019), also introduced the use of a mixture density network architecture, producing a distribution over possible object poses instead of a single estimate. This multimodal distribution turned out to improve performance of the system, even without including our observation-consistency prior, which we hypothesize is due to the ill-posed nature of pose estimation with symmetrical objects and the artifacts that arise with axis-angle pose representations. While HBEO-MDNs outperformed HBEOs for category-level pose estimation, and constitute the current state-of-the-art across input modalities for this task, the concept behind them is general; any perceptual system capable of generating

3D shape predictions and pose estimates can be extended to include an MDN-based pose distribution and our observation-consistency prior.

One of the main insights we obtained from this work is that the tight relationship between pose, shape, and object type, benefit general approaches that reason jointly over these characteristics. With HBEO-MDNs in particular, the pose-estimation performance gains we observed would not have been possible if our method was not jointly estimating 3D shape. In the future, we suggest extending this principal to fully include 2D segmentation, class, pose, and 3D shape, estimating a full joint distribution over all of these attributes. We also gained an appreciation for explicitly representational invariance, in collaborative work on grounding natural language to 3D shape (Cohen et al., 2019), we took advantage the invariance of BEO shape representations, with respect to object pose pose, in order to dramatically reduce the amount of language-annotated training data our grounding system required. We believe that intelligently decomposing perceptual output into these selectively invariant representations will reduce the required complexity of higher-level perceptual systems that upon these representations.

While general-purpose and robust object-based 3D perception remains an open and challenging problem, this work has taken useful strides towards making such a system a reality. In the future, we expect general perceptual systems to become increasingly high-performance and robust, leveraging multiple input sensing modalities, reasoning about multiple observations from various spatiotemporal locations, and producing full joint belief distributions—across multiple object attributes—complete with prediction confidences. We further expect explicitly low-dimensional representations, be they linear or nonlinear, to continue to play a critical role in realizing such representations by allowing relatively simple machine learning models—that do not require enormous volumes of training data—to be employed for higher level reasoning and robot control.

Bibliography

Andersen, A. H., Gash, D. M., and Avison, M. J. (1999), "Principal component analysis of the dynamic response measured by fMRI: a generalized linear systems framework," *Magnetic Resonance Imaging*, 17, 795–815.

Anguelov, D., Koller, D., Pang, H., Srinivasan, P., and Thrun, S. (2004), "Recovering articulated object models from 3D range data," in *Conference on Uncertainty in artificial intelligence*, pp. 18–26.

Attene, M. (2010), "A lightweight approach to repairing digitized polygon meshes," *The Visual Computer*, 26, 1393–1406.

B. Browatzkiand, J. Fischer, G. B. H. H. B. and Wallraven, C. (2011), "Going into depth: Evaluating 2D and 3D cues for object classification on a new, large-scale object dataset," in *International Conference on Computer Vision Workshops*, pp. 1189–1195.

B. Drost, M. Ulrich, N. N. and Ilic, S. (2010), "Model globally, match locally: Efficient and robust 3D object recognition," in *Computer Vision and Pattern Recognition*, pp. 998–1005.

Bai, S., Bai, X., Zhou, Z., Zhang, Z., and Jan Latecki, L. (2016), "GIFT: A Real-Time and Scalable 3D Shape Search Engine," in *Computer Vision and Pattern Recognition*.

Bakry, A. and Elgammal, A. (2014), "Untangling object-view manifold for multiview recognition and pose estimation," in *European Conference on Computer Vision*, pp. 434–449.

Bergamo, A. and Torresani, L. (2010), "Exploiting weakly-labeled Web images to improve object classification: a domain adaptation approach," in *Advances in Neural Information Processing Systems*, pp. 181–189.

Besl, P. J. and McKay, N. D. (1992), "Method for registration of 3-D shapes," *Pattern Analysis and Machine Intelligence*, 14, 239–256.

Bishop, C. M. (1994), "Mixture density networks," Technical report, Aston University, Birmingham.

Bishop, C. M. (1999a), "Bayesian PCA," in *Advances in Neural Information Processing Systems*, pp. 382–388.

Bishop, C. M. (1999b), "Variational principal components," in *International Conference on Artificial Neural Networks*, pp. 509–514.

Bore, N., Ambrus, R., Jensfelt, P., and Folkesson, J. (2017), "Efficient retrieval of arbitrary objects from long-term robot observations," *Robotics and Autonomous Systems*, 91, 139–150.

Boutsidis, C., Garber, D., Karnin, Z., and Liberty, E. (2015), "Online principal components analysis," in *ACM-SIAM Symposium on Discrete Algorithms*, pp. 887–901.

Breiman, L. (2001), "Random forests," *Machine learning*, 45, 5–32.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994), "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, pp. 737–744.

Burchfiel, B. and Konidaris, G. (2018), "Hybrid Bayesian Eigenobjects: Combining Linear Subspace and Deep Network Methods for 3D Robot Vision," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6843–6850.

Burchfiel, B. and Konidaris, G. (2019), "Probabilistic Category-Level Pose Estimation via Segmentation and Predicted-Shape Priors," *arXiv: 1905.12079*.

Carley, C. and Tomasi, C. (2015), "Single-frame indexing for 3D hand pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 101–109.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015), "ShapeNet: An Information-Rich 3D Model Repository," Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017), "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFS," *IEEE transactions on pattern analysis and machine intelligence*, 40, 834–848.

Chen, W., Liu, Y., Kira, Z., Wang, Y., and Huang, J. (2019), "A closer look at few-shot classification," *arXiv preprint arXiv:1904.04232*.

Chen, Z. (2003), "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond," *Statistics*, 182.

Cheng, Z., Chen, Y., Martin, R. R., Wu, T., and Song, Z. (2018), "Parametric modeling of 3D human body shapeâĂŤA survey," *Computers & Graphics*, 71, 88–100.

Choi, C., Taguchi, Y., Tuzel, O., Liu, M. Y., and Ramalingam, S. (2012), "Voting-based pose estimation for robotic assembly using a 3D sensor," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1724–1731.

Choy, C. B., Xu, D., Gwak, J., Chen, K., and Savarese, S. (2016), "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *European conference on computer vision*, pp. 628–644.

Cohen, V., Burchfiel, B., Nguyen, T., Gopalan, N., Tellex, S., and Konidaris, G. (2019), "Grounding Language Attributes to Objects using Bayesian Eigenobjects," *arXiv:1905.13153*.

Crow, F. (1987), "The origins of the teapot," *IEEE Computer Graphics and Applications*, 7, 8–19.

D. Huber, A. Kapuria, R. D. and Hebert, M. (2004), "Parts-based 3D object classification," in *Computer Vision and Pattern Recognition*, vol. 2, pp. 82–89.

Dai, A., Qi, C., and Nießner, M. (2017), "Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis," in *Computer Vision and Pattern Recognition*.

Dalal, N. and Triggs, B. (2005), "Histograms of oriented gradients for human detection," in *international Conference on computer vision and Pattern Recognition*, vol. 1, pp. 886–893, IEEE Computer Society.

Daniels, M. and Kass, R. (2001), "Shrinkage Estimators for Covariance Matrices," *Biometrics*, pp. 1173–1184.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004), "Efficient graph-based image segmentation," *International journal of computer vision*, 59, 167–181.

Fidler, S., Dickinson, S., and Urtasun, R. (2012), "3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model," in *Advances in Neural Information Processing Systems 25*, pp. 611–619.

Gehler, P. and Nowozin, S. (2009), "On feature combination for multiclass object classification," in *International Conference on Computer Vision*, pp. 221–228.

Girdhar, R., Carreira, J., Doersch, C., and Zisserman, A. (2019), "Video action transformer network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 244–253.

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

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

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017), "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988.

Hegde, V. and Zadeh, R. (2016), "FusionNet: 3D Object Classification Using Multiple Data Representations," *rXiv:1607.05695*.

Hendrycks, D. and Gimpel, K. (2016), "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks," *CoRR*, abs/1610.02136.

Hochreiter, S. and Schmidhuber, J. (1997), "Long short-term memory," *Neural computation*, 9, 1735–1780.

Hu, J., Shen, L., and Sun, G. (2018), "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141.

Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q. V., and Chen, Z. (2018), "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism," *CoRR*, abs/1811.06965.

J. Glover, R. R. and Bradski, G. (2011), "Monte Carlo Pose Estimation with Quaternion Kernels and the Bingham Distribution," in *Robotics: Science and Systems*.

Joachims, T. (1998), "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*, pp. 137–142.

Kar, A., Tulsiani, S., Carreira, J., and Malik, J. (2015), "Category-specific object reconstruction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1966–1974.

Katz, D., Kazemi, M., Bagnell, J. A., and Stentz, A. (2013), "Interactive segmentation, tracking, and kinematic modeling of unknown 3D articulated objects," in *IEEE International Conference on Robotics and Automation*, pp. 5003–5010.

Kaufman, A. E. (1994), "Voxels as a Computational Representation of Geometry," in *in The Computational Representation of Geometry. SIGGRAPH*, p. 45.

Kim, V. G., Li, W., Mitra, N. J., Chaudhuri, S., DiVerdi, S., and Funkhouser, T. (2013a), "Learning part-based templates from large collections of 3D shapes," *ACM Transactions on Graphics*, 32, 70.

Kim, Y., Mitra, N. J., Yan, D. M., and Guibas, L. (2012), "Acquiring 3D Indoor Environments with Variability and Repetition," *ACM Transactions on Graphics*, 31, 138:1–138:11.

Kim, Y., Mitra, N. J., Huang, Q., and Guibas, L. (2013b), "Guided Real-Time Scanning of Indoor Objects," in *Computer Graphics Forum*, vol. 32, pp. 177–186.

Kirillov, A., He, K., Girshick, R., Rother, C., and Dollár, P. (2019), "Panoptic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9404–9413.

Korn, M. R. and Dyer, C. R. (1987), "3-D multiview object representations for model-based object recognition," *Pattern Recognition*, 20, 91–103.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012), "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105.

L. Nan, K. X. and Sharf, A. (2012), "A Search-Classify Approach for Cluttered Indoor Scene Understanding," *ACM Transactions on Graphics*, 31.

Laine, S. and Karras, T. (2010), "Efficient sparse voxel octrees," *IEEE Transactions on Visualization and Computer Graphics*, 17, 1048–1059.

Laumond, J. P. et al. (1998), *Robot motion planning and control*, vol. 229, Springer.

Learned-Miller, E. G. (2006), "Data driven image models through continuous joint alignment," *Pattern Analysis and Machine Intelligence*, 28, 236–250.

LeCun, Y. and Bengio, Y. (1995), "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, 3361, 1995.

Ledoit, O. and Wolf, M. (2015), "Spectrum estimation: A unified framework for covariance matrix estimation and PCA in large dimensions," *Journal of Multivariate Analysis*, 139, 360–384.

Lee, K., Lee, K., Lee, H., and Shin, J. (2018), "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, pp. 7167–7177.

Li, Y., Dai, A., Guibas, L., and Nießner, M. (2015), "Database-Assisted Object Retrieval for Real-Time 3D Reconstruction," in *Computer Graphics Forum*, vol. 34, pp. 435–446.

Li, Y., Wang, G., Ji, X., Xiang, Y., and Fox, D. (2018a), "DeepIM: Deep Iterative Matching for 6D Pose Estimation," *CoRR*, abs/1804.00175.

Li, Y., Wang, G., Ji, X., Xiang, Y., and Fox, D. (2018b), "Deepim: Deep iterative matching for 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 683–698.

Liang, X., Lin, L., Wei, Y., Shen, X., Yang, J., and Yan, S. (2017), "Proposal-free network for instance-level object segmentation," *IEEE transactions on pattern analysis and machine intelligence*, 40, 2978–2991.

Lin, G., Milan, A., Shen, C., and Reid, I. (2017), "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1925–1934.

Liu, M., Tuzel, O., Veeraraghavan, A., Taguchi, Y., Marks, T., and Chellappa, R. (2012), "Fast object localization and pose estimation in heavy clutter for robotic bin picking," *The International Journal of Robotics Research*, 31, 951–973.

Lowe, D. G. et al. (1999), "Object recognition from local scale-invariant features." in *Interational Conference on Computer Vision*, vol. 99, pp. 1150–1157.

Ma, C., Guo, Y., Yang, J., and An, W. (2019), "Learning Multi-View Representation With LSTM for 3-D Shape Recognition and Retrieval," *IEEE Transactions on Multimedia*, 21, 1169–1182.

Marini, S., Biasotti, S., and Falcidieno, B. (2006), "Partial matching by structural descriptors," in *Content-Based Retrieval*.

Maturana, D. and Scherer, S. (2015), "Voxnet: A 3D convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems*, pp. 922–928.

Maurer, C. R., Qi, R., and Raghavan, V. (2003), "A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions," *Pattern Analysis and Machine Intelligence*, 25, 265–270.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010), "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*.

Narayanan, V. and Likhachev, M. (2016), "PERCH: Perception via Search for Multi-Object Recognition and Localization," in *International Conference on Robotics and Automation*.

Neal, R. M. (2012), *Bayesian learning for neural networks*, vol. 118, Springer.

Nguyen, A. and Le, B. (2013), "3D point cloud segmentation: A survey," in *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pp. 225–230.

Osher, S. and Fedkiw, R. (2003), *Signed Distance Functions*, pp. 17–22, Springer New York.

Pennington, J., Socher, R., and Manning, C. (2014), "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Qi, C., Su, H., Niessner, M., Dai, A., Yan, M., and Guibas, L. (2016), "Volumetric and Multi-View CNNs for Object Classification on 3D Data," in *Computer Vision and Pattern Recognition*.

Quigley, M., Conley, K., P. Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Y. Ng, A. (2009), "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3.

Rakoczy, H., Tomasello, M., and Striano, T. (2005), "How children turn objects into symbols: A cultural learning account," *In L. L. Namy (Ed.), Emory symposia in cognition. Symbol use and symbolic representation: Developmental and comparative perspectives.*, pp. 67–97.

Ramakrishna, V., Munoz, D., Hebert, M., Bagnell, A. J., and Sheikh, Y. (2014), "Pose Machines: Articulated Pose Estimation via Inference Machines," in *Proceedings of the European Conference on Computer Vision (ECCV)*.

R.B. Rusu, G. Bradski, R. T. and Hsu, J. (2010), "Fast 3D recognition and pose using the Viewpoint Feature Histogram," in *International Conference on Intelligent Robots and Systems*, pp. 2155–2162.

Rios-Cabrera, R. and Tuytelaars, T. (2013), "Discriminatively Trained Templates for 3D Object Detection: A Real Time Scalable Approach," in *2013 IEEE International Conference on Computer Vision*, pp. 2048–2055.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015), "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, 115, 211–252.

Särkkä, S. (2013), *Bayesian filtering and smoothing*, vol. 3, Cambridge University Press.

Schäfer, J. and Strimmer, K. (2005), "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics," *Statistical applications in genetics and molecular biology*, 4, 32.

Schiebener, D., Schmidt, A., Vahrenkamp, N., and Asfour, T. (2016), "Heuristic 3D object shape completion based on symmetry and scene context," in *Intelligent Robots and Systems*, pp. 74–81.

Shen, C. H., Fu, H., Chen, K., and Hu, S. M. (2012), "Structure Recovery by Part Assembly," *ACM Transactions on Graphics*, 31, 180:1–180:11.

Shi, B., Bai, S., Zhou, Z., and Bai, X. (2015), "DeepPano: Deep Panoramic Representation for 3-D Shape Recognition," *Signal Processing Letters*, 22, 2339–2343.

Soltani, A., Huang, H., Wu, J., Kulkarni, T., and Tenenbaum, J. (2017), "Synthesizing 3D Shapes via Modeling Multi-view Depth Maps and Silhouettes with Deep Generative Networks," *Computer Vision and Pattern Recognition*, pp. 2511–2519.

Song, S. and Xiao, J. (2016), "Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images," in *Computer Vision and Pattern Recognition*.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015a), "Multi-view convolutional neural networks for 3D shape recognition," in *International Conference on Computer Vision*, pp. 945–953.

Su, H., Qi, C. R., Li, Y., and Guibas, L. J. (2015b), "Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views," in *The IEEE International Conference on Computer Vision (ICCV)*.

Su, J., Gadelha, M., Wang, R., and Maji, S. (2018), "A Deeper Look at 3D Shape Classifiers," in *Proceedings of the European Conference on Computer Vision (ECCV)*.

Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., Xue, T., Tenenbaum, J. B., and Freeman, W. T. (2018), "Pix3d: Dataset and methods for single-image 3D shape modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2974–2983.

Sung, M., Kim, V. G., Angst, R., and Guibas, L. (2015), "Data-driven Structural Priors for Shape Completion," *ACM Transactions on Graphics*, 34, 175:1–175:11.

Tan, M. and Le, Q. V. (2019), "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *arXiv preprint arXiv:1905.11946*.

Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2016), "Multi-view 3D models from single images with a convolutional network," in *European Conference on Computer Vision (ECCV)*, pp. 322–337.

Tatarchenko, M., Dosovitskiy, A., and Brox, T. (2017), "Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2107–2115.

Tibshirani, R. (1996), "Regression shrinkage and selection via the lasso," *The Royal Statistical Society*, pp. 267–288.

Tipping, M. E. and Bishop, C. M. (1999), "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61, 611–622.

Tompson, J., Stein, M., Lecun, Y., and Perlin, K. (2014), "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Transactions on Graphics (ToG)*, 33, 169.

Tulsiani, S. and Malik, J. (2015), "Viewpoints and keypoints," in *Computer Vision and Pattern Recognition*, pp. 1510–1519.

Tulsiani, S., Zhou, T., Efros, A., and Malik, J. (2017), "Multi-view supervision for single-view reconstruction via differentiable ray consistency," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 2626–2634.

Turk, M. and Pentland, A. (1991), "Face recognition using Eigenfaces," in *Computer Vision and Pattern Recognition*, pp. 586–591.

Varley, J., DeChant, C., Richardson, A., Ruales, J., and Allen, P. (2017), "Shape completion enabled robotic grasping," in *Intelligent Robots and Systems*, pp. 2442–2447.

Viola, P., Jones, M., et al. (2001), "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Wang, Y., Shi, T., Yun, P., Tai, L., and Liu, M. (2018), "Pointseg: Real-time semantic segmentation based on 3D lidar point cloud," *arXiv preprint arXiv:1807.06288*.

Wold, S., Esbensen, K., and Geladi, P. (1987), "Principal component analysis," *Chemometrics and intelligent laboratory systems*, 2, 37–52.

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. (2016), "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, pp. 82–90.

Wu, J., Wang, Y., Xue, T., Sun, X., Freeman, B., and Tenenbaum, J. (2017), "Marrnet: 3D shape reconstruction via 2.5 d sketches," in *Advances in neural information processing systems*, pp. 540–550.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015), "3D shapenets: A deep representation for volumetric shapes," in *Computer Vision and Pattern Recognition*, pp. 1912–1920.

Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2017), "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," *CoRR*, abs/1711.00199.

Yu, H., Yang, Z., Tan, L., Wang, Y., Sun, W., Sun, M., and Tang, Y. (2018), "Methods and datasets on semantic segmentation: A review," *Neurocomputing*, 304, 82–103.

Zhang, H., Fritts, J. E., and Goldman, S. A. (2008), "Image segmentation evaluation: A survey of unsupervised methods," *computer vision and image understanding*, 110, 260–280.

Zhang, Z., Fidler, S., and Urtasun, R. (2016), "Instance-level segmentation for autonomous driving with deep densely connected MRFS," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 669–677.

Zuendorf, G., Kerrouche, N., Herholz, K., and Baron, J. C. (2003), "Efficient principal component analysis for multivariate 3D voxel-based mapping of brain functional imaging data sets as applied to FDG-PET and normal aging," *Human Brain Mapping*, 18, 13–21.

Biography

Benjamin Burchfiel was born in Winchester, Massachusetts, a suburban town just outside of Boston. Beginning in 2008, Benjmain attended the University of Wisconsin-Madison, where he received his computer science Bachelor of Science degree in 2012. While an undergraduate student, Benjamin became interested in AI and computer vision and assisted in a research project to detect online bullying on social media under the supervision of Professor Charles Dyer and Professor Xiaojin Zhu.

Benjamin was admitted to the Computer Science Ph.D. program at Duke University in 2013 where he investigated robot learning from suboptimal demonstrations with Professor Carlo Tomasi and Professor Ronald Parr before joining Professor George Konidaris' Intelligent Robotic Lab where he developed his thesis on general object representations for 3D robot perception. In 2014 Benjamin was the recipient of the Department of Computer Science Excellence in teaching award before receiving his Master of Science degree in computer science from Duke University in 2016. Benjamin will defend his Ph.D. thesis in July 2019.

Benjamin's primary research interests lie in the intersection of robotics, machine learning, and computer vision with the ultimate goal of enabling the deployment of general-purpose robots in fully unstructured environments with minimal supervision. Beginning in the fall of 2019, Benjamin will join Brown University as a postdoctoral researcher in the department of computer science.

Benjamin's personal website may be found at `benburchfiel.com`.