Depth from Camera Motion and Object Detection

Brent A. Griffin
University of Michigan
griffb@umich.edu

Jason J. Corso
Stevens Institute for Artificial Intelligence
jcorso@stevens.edu

Abstract

This paper addresses the problem of learning to estimate the depth of detected objects given some measurement of camera motion (e.g., from robot kinematics or vehicle odometry). We achieve this by 1) designing a recurrent neural network (DBox) that estimates the depth of objects using a generalized representation of bounding boxes and uncalibrated camera movement and 2) introducing the Object Depth via Motion and Detection Dataset (ODMD). ODMD training data are extensible and configurable, and the ODMD benchmark includes 21,600 examples across four validation and test sets. These sets include mobile robot experiments using an end-effector camera to locate objects from the YCB dataset and examples with perturbations added to camera motion or bounding box data. In addition to the ODMD benchmark, we evaluate DBox in other monocular application domains, achieving state-of-the-art results on existing driving and robotics benchmarks and estimating the depth of objects using a camera phone.

1. Introduction

With the progression of high-quality datasets and subsequent methods, our community has seen remarkable advances in image segmentation [27, 54], video object segmentation [35, 38], and object detection [4, 16, 32], sometimes with a focus on a specific application like driving [5, 11, 43]. However, applications like autonomous vehicles and robotics require a three-dimensional (3D) understanding of the environment, so they frequently rely on 3D sensors (e.g., LiDAR [10] or RGBD cameras [8]). Although 3D sensors are great for identifying free space and motion planning, classifying and understanding raw 3D data is a challenging and ongoing area of research [25, 28, 33, 39]. On the other hand, RGB cameras are inexpensive, ubiquitous, and interpretable by countless vision methods.

To bridge the gap between 3D applications and progress in video object segmentation, in recent work [14], we developed a method of video object segmentation-based visual servo control, object depth estimation, and mobile robot
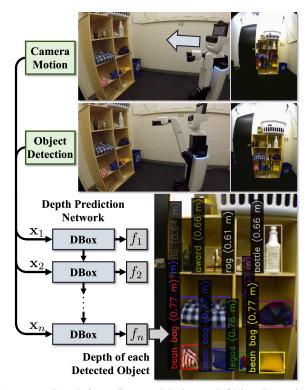

Figure 1. **Depth from Camera Motion and Object Detection**. Object detectors can reliably place bounding boxes on target objects in a variety of settings. Given a sequence of bounding boxes and camera movement distances between observations (e.g., from robot kinematics, top), our network (DBox) estimates each object's depth (bottom). This result is from the ODMD Robot Set.

grasping using a single RGB camera. For object depth estimation, specifically, we used the optical expansion [22, 46] of segmentation masks with z-axis camera motion to analytically solve for depth [14, VOS-DE (20)]. In subsequent work [15], we introduced the first learning-based method (ODN) and benchmark dataset for estimating Object Depth via Motion and Segmentation (ODMS), which includes test sets in robotics and driving. From the ODMS benchmark, ODN improves accuracy over VOS-DE in multiple domains, especially those with segmentation errors.

Motivated by these developments [14, 15], this paper ad-

dresses the problem of estimating the depth of objects using uncalibrated camera motion and bounding boxes from object detection (see Figure 1), which has many advantages. First, a bounding box has only four parameters, can be processed quickly with few resources, and has less domain-specific features than an RGB image or segmentation mask. Second, movement is already measured on most autonomous hardware platforms and, even if not measured, structure from motion is plausible to recover camera motion [23, 34, 45]. Third, as we show with a pinhole camera and box-based model (see Figure 2) and in experiments, we can use x-, y-, or z-axis camera motion to estimate depth using optical expansion, motion parallax [9, 42], or both. Finally, our detection-based methods can support more applications by using boxes *or* segmentation masks, which we demonstrate in multiple domains with state-of-the-art results on the segmentation-based ODMS benchmark [15].

The first contribution of our paper is deriving an analytical model and corresponding solution (Box_{LS}) for uncalibrated motion and detection-based depth estimation in Section 3.1. To the best of our knowledge, this is the first model or solution in this new problem space. Furthermore, Box_{LS} achieves the best analytical result on the ODMS benchmark.

A second contribution is developing a recurrent neural network (RNN) to predict **D**epth from motion and bounding **Box**es (DBox) in Section 3.2. DBox sequentially processes observations and uses our normalized and dimensionless input-loss formulation, which improves performance across domains with different movement distances and camera parameters. Thus, using a single DBox network, we achieve the best Robot, Driving, and overall result on the ODMS benchmark and estimate depth from a camera phone.[1]

Inspired by ODMS [15], a final contribution of our paper is the **O**bject **D**epth via **M**otion and **D**etection (ODMD) dataset in Section 3.3.[2] ODMD is the first dataset for motion and detection-based depth estimation, which enables learning-based methods in this new problem space. ODMD data consist of a series of bounding boxes, x, y, z camera movement distances, and ground truth object depth.

For ODMD training, we continuously generate synthetic examples with random movements, depths, object sizes, and three types of perturbations typical of camera motion and object detection errors. As we will show, training with perturbations improves end performance in real applications. Furthermore, ODMD's distance- and box-based inputs are 1) simple, so we can generate over 300,000 training examples per second, and 2) general, so we can transfer from synthetic training data to many application domains.

Finally, for an ODMD evaluation benchmark, we create four validation and test sets with 21,600 examples, including mobile robot experiments locating YCB objects [3].

---

[1]Supplementary video: https://youtu.be/GruhbdJ2l7k
[2]Dataset website: https://github.com/griffbr/ODMD

2. Related Work

Object Detection predicts a set of bounding boxes and category labels for objects of interest. Many object detectors operate using regression and classification over a set of region proposals [2, 41], anchors [31], or window centers [47]. Other detectors treat detection as a single regression problem [40] or, more recently, use a transformer architecture [48] to directly predict all detections in parallel [4]. Given the utility of locating objects in RGB images, detection supports many downstream vision tasks such as segmentation [17], 3D shape prediction [13], object pose estimation [36], and even single-view metrology [55], to name but a few.

In this work, we provide depth for "free" as an extension of object detection in mobile applications. We detect objects on a per-frame basis, then use sequences of bounding boxes with uncalibrated camera motion to find the depth of each object. One benefit of our approach is that depth accuracy will improve with future detection methods. For experiments, we use Faster R-CNN [41], which has had many improvements since its original publication, runs in real time, and is particularly accurate for small objects [4]. Specifically, we use the same Faster R-CNN configuration as Detectron2 [50] with ResNet 50 [18] pre-trained on ImageNet [6] and a FPN [30] backbone trained on COCO [32], which we then fine-tune for our validation and test set objects.

Object Pose Estimation, to its fullest extent, predicts the 3D position and 3D rotation of an object in camera-centered coordinates, which is useful for autonomous driving, augmented reality, and robot grasping [44]. Pose estimation methods for household objects typically use a single RGB image [24, 36], RGBD image [49], or separate setting for each [1, 51]. Alternatively, recent work [29] uses multiple views to jointly predict multiple object poses, which achieves the best result on the YCB-Video dataset [51], T-LESS dataset [20], and BOP Challenge 2020 [21].

To predict object depth without scale ambiguity, many RGB-based pose estimation methods learn a prior on specific 3D object models [3]. However, innovations in robotics are easing this requirement. Recent work [37] uses a robot to collect data in cluttered scenes to add texture to simplified pose training models. Other work [7] uses a robot to interact with objects and generate new training data to improve its pose estimation. In our previous work [14, 15], we use robot motion with segmentation to predict depth and grasp objects, which removes 3D models entirely.

In this work, we build off of these developments to improve RGB-based object depth estimation without any 3D model requirements. Instead, we use depth cues based on uncalibrated camera movement and general object detection or segmentation. A primary benefit of our approach is its generalization across domains, which we demonstrate in robotics, driving, and camera phone experiments.

3. Depth from Camera Motion and Object Detection

We design an RNN to predict **D**epth from camera motion and bounding **Box**es (DBox). DBox sequentially processes each observation and uses optical expansion and motion parallax cues to make its final object depth prediction. To train and evaluate DBox, we introduce the **O**bject **D**epth via **M**otion and **D**etection Dataset (ODMD). First, in Section 3.1, we derive our motion and detection model with analytical solutions, which is the theoretical foundation for this work and informs our design decisions in the remaining paper. Next, in Section 3.2, we detail DBox's input-loss formulation and architecture. Finally, in Section 3.3, we explain ODMD's extensible training data, validation and test sets for evaluation, and DBox training configurations.

3.1. Depth from Motion and Detection Model

Motion and Detection Inputs. To find a detected object's depth, assume we are given a set of $n \geq 2$ observations $\mathbf{X} := \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, where each observation $\mathbf{x}_i$ consists of a bounding box for the detected object and a corresponding camera position. Specifically,

$$\mathbf{x}_i := \begin{bmatrix} x_i, y_i, w_i, h_i, \mathbf{p}_i^\mathsf{T} \end{bmatrix}^\mathsf{T}, \tag{1}$$

where x_i, y_i, w_i, h_i denote the two image coordinates of the bounding box center, width, and height and

$$\mathbf{p}_i := \begin{bmatrix} \mathrm{C}_{Xi}, \mathrm{C}_{Yi}, \mathrm{C}_{Zi} \end{bmatrix}^\mathsf{T} \tag{2}$$

is the relative camera position for each observation $\mathbf{x}_i \in \mathbb{R}^7$. Notably, we align the axes of $\mathbf{p}_i$ with the camera coordinate frame and the model is most accurate without camera rotation, but the absolute position of $\mathbf{p}_i$ is inconsequential. Observations can be collected as a set of images or by video.

Camera Model. To infer 3D information from 2D detection, we relate an object's bounding box image points to 3D camera-frame coordinates using the pinhole camera model

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix} \begin{bmatrix} \frac{X}{Z_i}, \frac{Y}{Z_i}, 1 \end{bmatrix}^\mathsf{T}, \tag{3}$$

where f_x, f_y and c_x, c_y are the camera's focal lengths and principal points and X, Y correspond to image coordinates x, y in the 3D camera frame at depth Z_i. Notably, Z_i is the distance along the optical axis (or depth) between the camera and the visible perimeter of the detected object. To specify individual x, y image coordinates we simplify (3) to

$$x = \frac{f_x X}{Z_i} + c_x, \quad y = \frac{f_y Y}{Z_i} + c_y. \tag{4}$$

Notably, although we include pinhole camera parameters in the model, we do not use them to solve depth at inference.
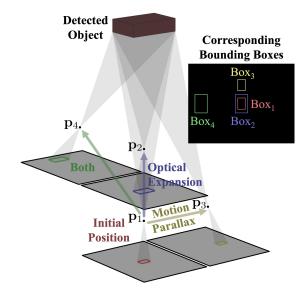

Figure 2. **Motion and Detection Model**. We show an object's bounding box (Box$_i$) for four pinhole camera positions ($\mathbf{p}_i$). Boxes scale inversely with object depth (optical expansion) and offset with lateral position (motion parallax). Using relative camera movements with these cues, we can solve for an object's depth.

Depth from Optical Expansion & Detection. In an ideal model, we can find object depth using z-axis motion between observations and corresponding changes in bounding box scale (e.g., Box$_1$ to Box$_2$ in Figure 2). First, we use (4) to relate bounding box width w_i to 3D object width W as

$$w_i = x_{\mathrm{R}i} - x_{\mathrm{L}i} = \frac{f_x X_{\mathrm{R}i}}{Z_i} + c_x - \frac{f_x X_{\mathrm{L}i}}{Z_i} - c_x = \frac{f_x W}{Z_i}, \tag{5}$$

where $x_{\mathrm{R}i}, x_{\mathrm{L}i}$ are the right and left box image coordinates with 3D coordinates $X_{\mathrm{R}i}, X_{\mathrm{L}i}$ respectively. Object width W is constant, so we use (5) to relate two observations as

$$w_i Z_i = w_j Z_j = f_x W. \tag{6}$$

Notably, (5)-(6) also apply to object height H. Specifically, $h_i = \frac{f_y H}{Z_i}$ from (5) and $h_i Z_i = h_j Z_j = f_y H$ from (6).

To use the camera motion ($\mathbf{p}_i$), we note that changes in depth Z_i between observations of a static object are only caused by changes in camera position C_{Zi} (2). Thus,

$$Z_i + \mathrm{C}_{Zi} = Z_j + \mathrm{C}_{Zj} \implies Z_j = Z_i + \mathrm{C}_{Zi} - \mathrm{C}_{Zj}. \tag{7}$$

Finally, we use (7) in (6) and solve for object depth Z_i as

$$w_i Z_i = w_j(Z_i + \mathrm{C}_{Zi} - \mathrm{C}_{Zj}) = f_x W$$
$$\implies Z_i = \frac{\mathrm{C}_{Zj} - \mathrm{C}_{Zi}}{1 - \left(\frac{w_i}{w_j}\right)}. \tag{8}$$

In (8), we find Z_i from two observations i, j using optical expansion, i.e., the change in bounding box scale $(\frac{w_i}{w_j})$ relative to z-axis camera motion ($\mathrm{C}_{Zj} - \mathrm{C}_{Zi}$). To measure the change in box scale using height, we replace $\frac{w_i}{w_j}$ with $\frac{h_i}{h_j}$.

Depth from Motion Parallax & Detection. If there is x- or y-axis camera motion, we can solve for object depth using corresponding changes in bounding box location (e.g., Box_1 to Box_3 in Figure 2). For brevity, we provide this derivation and comparative results in the supplementary material.

Using all Observations to Improve Depth. In real applications, camera motion measurements and object detection will have errors. Thus, we make detection-based depth estimation more robust by incorporating all n observations.

For the n-observation solution, we reformulate (8) for each observation as $w_j Z_i - f_x W = w_j(\mathbf{C}_{Zj} - \mathbf{C}_{Zi})$ for width and $h_j Z_i - f_y H = h_j(\mathbf{C}_{Zj} - \mathbf{C}_{Zi})$ for height. Now we can estimate Z_i over n observations in $\mathbf{Ax} = \mathbf{b}$ form as

$$
\begin{bmatrix}
w_1 & 1 & 0 \\
h_1 & 0 & 1 \\
w_2 & 1 & 0 \\
\vdots & \vdots & \vdots \\
h_n & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\hat{Z}_i \\
-f_x \hat{W} \\
-f_y \hat{H}
\end{bmatrix}
=
\begin{bmatrix}
w_1(\mathbf{C}_{Z1} - \mathbf{C}_{Zi}) \\
h_1(\mathbf{C}_{Z1} - \mathbf{C}_{Zi}) \\
w_2(\mathbf{C}_{Z2} - \mathbf{C}_{Zi}) \\
\vdots \\
h_n(\mathbf{C}_{Zn} - \mathbf{C}_{Zi})
\end{bmatrix}.
\tag{9}
$$

In this work, we solve for $\hat{Z}_i$ as a least-squares approximation of Z_i. In Section 4, we refer to this solution as Box_{LS}.

3.2. Depth from Motion and Detection Network

Considering the motion and detection model and Box_{LS} solution in Section 3.1, we design DBox to use all n observations for robustness and full x, y, z camera motion to utilize both optical expansion and motion parallax cues. Additionally, to improve performance across domains, we derive a normalized and dimensionless input-loss formulation.

Normalized Network Input. As in Section 3.1, assume we have a set of $n \geq 2$ observations $\mathbf{X}$ to predict depth. We normalize the bounding box coordinates of each $\mathbf{x}_i$ (1) as

$$
\bar{\mathbf{b}}_i := \left[\frac{x_i}{W_I}, \frac{y_i}{H_I}, \frac{w_i}{W_I}, \frac{h_i}{H_I} \right]^{\mathsf{T}},
\tag{10}
$$

where W_I and H_I are the box image's width and height. We normalize the camera position $\mathbf{p}_i$ (2) of each $\mathbf{x}_i$ as

$$
\bar{\mathbf{p}}_i := \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{\|\mathbf{p}_n - \mathbf{p}_1\|},
\tag{11}
$$

where $\|\mathbf{p}_n - \mathbf{p}_1\|$ is the overall Euclidean camera movement range, $\mathbf{p}_i - \mathbf{p}_{i-1}$ is the incremental camera movement, and we set initial condition $\mathbf{p}_0 = \mathbf{p}_1 \implies \bar{\mathbf{p}}_1 = \begin{bmatrix} 0, 0, 0 \end{bmatrix}^{\mathsf{T}}$.

From (10) and (11), we form the normalized network input $\bar{\mathbf{X}} := \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \cdots, \bar{\mathbf{x}}_n\}$, where each $\bar{\mathbf{x}}_i \in \mathbb{R}^7$ is defined

$$
\bar{\mathbf{x}}_i := \begin{bmatrix} \bar{\mathbf{b}}_i^{\mathsf{T}}, \bar{\mathbf{p}}_i^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}.
\tag{12}
$$

Normalized Network Loss. A straightforward loss for learning to estimate object depth is direct prediction, i.e.,

$$
\mathcal{L}_{\text{Abs}}(\mathbf{W}) := Z_n - f_{\text{Abs}}(\mathbf{X}, \mathbf{W}),
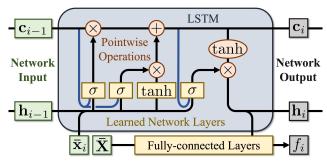\tag{13}
$$

Figure 3. **DBox Network**. DBox is a Long Short-Term Memory network with sequential input (green), five pointwise operations (red), five learned components (yellow), three σ-gate peephole connections (blue), and a sequential depth output (gray).

where $\mathbf{W}$ are the trainable network parameters, Z_n is the ground truth object depth at $\mathbf{p}_n$, and $f_{\text{Abs}} \in \mathbb{R}$ is the predicted absolute object depth. For the input $\mathbf{X}$ in (13), we use the $\bar{\mathbf{b}}_i$ bounding box format (10) and make each camera position relative to final prediction position $\mathbf{p}_n = \begin{bmatrix} 0, 0, 0 \end{bmatrix}^{\mathsf{T}}$.

To use dimensionless input $\bar{\mathbf{X}}$ (12), we modify (13) as

$$
\mathcal{L}_{\text{Rel}}(\mathbf{W}) := \frac{Z_n}{\|\mathbf{p}_n - \mathbf{p}_1\|} - f_{\text{Rel}}(\bar{\mathbf{X}}, \mathbf{W}),
\tag{14}
$$

where $\frac{Z_n}{\|\mathbf{p}_n - \mathbf{p}_1\|}$ is the ground truth object depth at $\mathbf{p}_n$ made dimensionless by its relation to the overall camera movement range and $f_{\text{Rel}} \in \mathbb{R}$ is the corresponding relative depth prediction. To use this dimensionless relative output at inference, we simply multiply f_{Rel} by $\|\mathbf{p}_n - \mathbf{p}_1\|$ to find Z_n.

As we will show in experiments in Section 4, by using a normalized and dimensionless input-loss formulation (14), DBox can predict object depth across domains with vastly different image resolutions and camera movement ranges.

Network Architecture. The DBox architecture is shown in Figure 3. DBox is a modified Long Short-Term Memory (LSTM) RNN [19] with peephole connections for each σ-gate [12] and fully-connected (FC) layers that use each LSTM output ($\mathbf{h}_i$) within the context of all n observations ($\bar{\mathbf{X}}$) to predict object depth (f_i). Using initial conditions $\mathbf{c}_0, \mathbf{h}_0 = \mathbf{0}$, each intermediate input ($\mathbf{c}_{i-1}, \mathbf{h}_{i-1}, \bar{\mathbf{x}}_i$) and output ($\mathbf{c}_i, \mathbf{h}_i, f_i$) is unique. DBox operates sequentially across all n observations to predict final object depth f_n.

We design the network to maximize depth prediction performance with low GPU requirements. The LSTM hidden state capacity is 128 (i.e., $\mathbf{c}_i, \mathbf{h}_i \in \mathbb{R}^{128}$). Outside the LSTM, $\mathbf{h}_i$ is fed into the first of six FC layers, each with 256 neurons, ReLU activation, and $\bar{\mathbf{X}} \in \mathbb{R}^{7n}$ concatenated to their input. The output of the sixth FC layer is fed to a fully-connected neuron, $f_i \in \mathbb{R}$, the depth output. Using a modern workstation and GPU (GTX 1080 Ti), a full DBox forward pass of 1×10^3 10-observation examples uses 677 MiB of GPU memory and takes 4.3×10^{-3} seconds.

3.3. Depth from Motion and Detection Dataset

To train and evaluate our DBox design from Section 3.2, we introduce the ODMD dataset. ODMD enables DBox to learn detection-based depth estimation from any combination of x, y, z camera motion. Furthermore, although the current ODMD benchmark focuses on robotics, our general ODMD framework can be configured for new applications.

Generating Motion and Detection Training Data. To generate ODMD training data, we start with the n random camera movements ($\mathbf{p}_i$ (2)) in each training example $\mathbf{X}$. Given a minimum and maximum movement range as configurable parameters, $\Delta\mathbf{p}_{\min}$ and $\Delta\mathbf{p}_{\max}$, we define a uniform random variable for the overall camera movement as

$$\Delta\mathbf{p} := \mathbf{p}_n - \mathbf{p}_1 \sim \mathcal{U}[\Delta\mathbf{p}_{\min}, \Delta\mathbf{p}_{\max}] \circ \mathbf{k}, \quad (15)$$

where $\Delta\mathbf{p} \in \mathbb{R}^3$ is the camera movement from $\mathbf{p}_1$ to $\mathbf{p}_n$ and $\mathbf{k} \in \{-1, 1\}^3$ uses a Rademacher distribution to randomly assign the movement direction of each axis. As in (13), we choose $\mathbf{p}_n = [0, 0, 0]^\mathsf{T} \implies \mathbf{p}_1 = -\Delta\mathbf{p}$ in (15). For the $1 < i < n$ intermediate camera positions, we use $\mathbf{p}_i \sim \mathcal{U}[\mathbf{p}_1, \mathbf{p}_n]$ then sort the collective $\mathbf{p}_i$ values along each axis so that movement from $\mathbf{p}_1$ to $\mathbf{p}_n$ is monotonic.

After finding the camera movement, we generate a random object with a random initial 3D position. To make each object unique, we randomize its physical width and height each as $W, H \sim \mathcal{U}[s_{\min}, s_{\max}]$, where $s_{\min}, s_{\max}$ are configurable. Using parameters $Z_{1\min}, Z_{1\max}$, we similarly randomize the object's initial depth as $Z_1 \sim \mathcal{U}[Z_{1\min}, Z_{1\max}]$. To ensure the object is within view, we use Z_1 to adjust the bounds of the object's random center position X_1, Y_1. Thus,

$$X_1 \sim \mathcal{U}[X_{1\min}(Z_1), X_{1\max}(Z_1)]$$
$$Y_1 \sim \mathcal{U}[Y_{1\min}(Z_1), Y_{1\max}(Z_1)], \quad (16)$$

and $[X_1, Y_1, Z_1]^\mathsf{T}$ are the object's 3D camera-frame coordinates at camera position $\mathbf{p}_1$. For completeness, we provide a detailed derivation of (16) in the supplementary material.

After finding size and position, we project the object's bounding box onto the camera's image plane at all n camera positions. As in (7), assuming a static object, all motion is from the camera. Thus, the object's position at each $\mathbf{p}_i$ is

$$[X_i, Y_i, Z_i]^\mathsf{T} = [X_1, Y_1, Z_1]^\mathsf{T} - (\mathbf{p}_i - \mathbf{p}_1). \quad (17)$$

In other words, object motion in the camera frame is equal and opposite to the camera motion itself. Using X_i, Y_i, Z_i, W, H in (4) and (5), we find bounding box image coordinates x_i, y_i, w_i, h_i (1) at all n camera positions, which completes the random training example $\mathbf{X}$.

As a final adjustment for greater variability, because the range of final object position $[X_n, Y_n, Z_n]^\mathsf{T}$ is greater than that of the initial position $[X_1, Y_1, Z_1]^\mathsf{T}$, we randomly reverse the order of all observations in $\mathbf{X}$ with probability 0.5 (i.e., $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ changes to $\{\mathbf{x}_n, \mathbf{x}_{n-1}, \cdots, \mathbf{x}_1\}$).

Learning which Observations to Trust. We add perturbations to ODMD-generated data as an added challenge. As we will show in Section 4, this decision improves the performance of ODMD-trained networks in real applications with object detection and camera movement errors.

For perturbations that cause camera movement errors, we modify (2) by adding noise to each $\mathbf{p}_i$ for $1 < i \leq n$ as

$$\mathbf{p}_{pi} := \mathbf{p}_i + [p_{Xi}, p_{Yi}, p_{Zi}]^\mathsf{T}, \; p_{Xi}, p_{Yi}, p_{Zi} \sim \mathcal{N}(0, \sigma^2), \quad (18)$$

where $\mathbf{p}_{pi}$ is the perturbed version of the camera position $\mathbf{p}_i$ and $\mathcal{N}(0, \sigma^2)$ is a Gaussian distribution with $\mu = 0$, $\sigma = 1 \times 10^{-2}$ that is uniquely sampled for each p_{Xi}, p_{Yi}, p_{Zi}. Note that σ can be configured to reflect the anticipated magnitude of errors for a specific application domain.

We use two types of perturbations for object detection errors. First, similar to (18), we modify (10) by adding noise ($\sigma = 1 \times 10^{-3}$) to each bounding box $\bar{\mathbf{b}}_i$ for $1 \leq i \leq n$ as

$$\bar{\mathbf{b}}_{pi} := \bar{\mathbf{b}}_i + [p_{xi}, p_{yi}, p_{wi}, p_{hi}]^\mathsf{T}$$
$$p_{xi}, p_{yi}, p_{wi}, p_{hi} \sim \mathcal{N}(0, \sigma^2). \quad (19)$$

For the second perturbation, we randomly replace one $\bar{\mathbf{b}}_i$ with a *completely* different bounding box with probability 0.1. This random replacement synthesizes intermittent detections of the wrong object, and the probability of replacement can be configured for a specific application. Ground truth labels remain the same when we use perturbations.

ODMD Validation and Test Sets. We introduce four ODMD validation and test sets using robot experiments and simulated data with various levels of perturbations. This establishes a repeatable benchmark for ablative studies and future methods. All examples include $n = 10$ observations.

The robot experiment data evaluates object depth estimation on a physical platform using object detection on real-world objects. We collect data with camera movement using a Toyota Human Support Robot (HSR). HSR has an end effector-mounted wide-angle grasp camera, a 4-DOF arm on a torso with prismatic and revolute joints, and a differential drive base [52, 53]. Using HSR's full kinematics, we collect sets of 480×640 grasp-camera images across randomly sampled camera movements ($\Delta\mathbf{p}$ (15)) with target objects in view (see example in Figure 1).

For the robot experiment objects, we use 30 custom household object for the Validation Set and 30 YCB objects [3] for the Test Set (see Figure 4) across six different scenes. We detect objects using Faster R-CNN [41], which we fine-tune on each set of objects using custom annotation images outside of the validation and test sets. The camera movement range ($\Delta\mathbf{p}$) varies between $[-0.34, -0.33, -0.43]^\mathsf{T}$ to $[0.33, 0.36, 0.37]^\mathsf{T}$ m, and the final object depth (Z_n), which we measure manually, varies between 0.11-1.64 m.

Figure 4. **Robot Experiment Objects**. We use 30 custom objects in the Validation Set (left) and 30 YCB objects in the Test Set (right). Object size spans from 16 mm (die) to 280 mm (shirt).

Altogether, we generate 5,400 robot object depth estimation examples (2,400 validation and 3,000 test), and we show a few challenging examples in the supplementary material.

We generate a set of normal and two types of perturbation-based data for simulated objects. The Normal Set approximates HSR's operation without any object detection or camera movement errors. Using ODMD's random data-generation framework, we choose configurable parameters: $s_{\min} = 0.01, s_{\max} = 0.175$ m for object size; $\Delta\mathbf{p}_{\min} = [0, 0, 0.05]^{\mathsf{T}}, \Delta\mathbf{p}_{\max} = [0.25, 0.175, 0.325]^{\mathsf{T}}$ m for camera movement (15); and $Z_{1\min} = 0.55, Z_{1\max} = 1$ m for initial object depth. For the camera model, we use factory-provided intrinsics for HSR's 480×640 grasp camera, specifically, $f_x, f_y = 205.5, c_x = 320.5, c_y = 240.5$. This configuration satisfies camera view constraints in (16).

The two perturbation-based sets approximate HSR's operation with only object detection or camera movement errors. For the Camera Motion Perturbation Set, we use the same configuration as the Normal Set with camera movement noise added (18). For the Object Detection Perturbation Set, we use the normal configuration with bounding box noise and random box replacement (19). We generate 5,400 depth estimation examples for each of the simulated object configurations (2,400 validation and 3,000 test).

ODMD Training Configurations for DBox. To test the efficacy of new concepts and establish best practices, we train DBox using multiple ODMD training configurations. The first configuration is the same as ODMD's **N**ormal validation and test **S**et (DBox$_{NS}$) but trains on continuously-generated data. Similarly, the second configuration (DBox$_p$) is based on both perturbation sets and trains on new data with camera movement errors (18), object detection errors (19), and random replacement.

DBox$_{NS}$ and DBox$_p$ both learn to predict object depth relative to camera movement using the dimensionless loss $\mathcal{L}_{Rel}$ (14). Alternatively, the third configuration (DBox$_{Abs}$)

learns to predict absolute object depth using $\mathcal{L}_{Abs}$ (13). DBox$_{Abs}$ uses the same training perturbations as DBox$_p$.

We train each network using a batch size of 512 randomly-generated training examples with $n = 10$ observations per prediction. We train each network for 1×10^7 iterations using the Adam Optimizer [26] with a 1×10^{-3} learning rate, which takes approximately 2.8 days using a single GPU (GTX 1080 Ti). We also train alternate versions of DBox$_p$ for 1×10^6 (DBox$_p^{1M}$) and 1×10^5 iterations (DBox$_p^{100K}$), which takes 6.7 and 0.7 hours respectively.

For compatibility with the ODMS benchmark [15], we train three final configurations with *only z*-axis camera motion and more general camera parameters. The first configuration (DBox$_p^z$) is the same as DBox$_p$ but with $\Delta\mathbf{p}_{\max} = [0, 0, 0.4625]^{\mathsf{T}}$ m and $f_x, f_y = 240.5$. The other configurations are the same as DBox$_p^z$ but either remove perturbations (DBox$_{NS}^z$) or use loss $\mathcal{L}_{Abs}$ (DBox$_{Abs}^z$). Because single-axis camera movement is simple, we train each configuration for only 1×10^4 iterations, which takes less than four minutes.

4. Experimental Results

4.1. Setup

For the first set of experiments, we evaluate DBox on the new ODMD benchmark in Section 4.2. We find the number of training iterations for the results using the best overall validation performance, which we check at every hundredth of the total training iterations. Like the ODMS benchmark [15], we evaluate each method using the mean percent error for each test set, which we calculate for each example as

$$\text{Percent Error} = \left| \frac{Z_n - \hat{Z}_n}{Z_n} \right| \times 100\%, \qquad (20)$$

where Z_n and $\hat{Z}_n$ are ground truth and predicted object depth at final camera position $\mathbf{p}_n$. Finally, for all of our results, if an object is not detected at position $\mathbf{p}_i$, we use the bounding box from the nearest position with a detection.

We compare DBox to state-of-the-art methods on the object segmentation-based ODMS benchmark in Section 4.3. Because ODMS provides only z-axis camera motion and segmentation, we preprocess inputs for DBox. First, we set all x- and y-axis camera inputs to zero. Second, we create bounding boxes around the object segmentation masks using the minimum and maximum x, y pixel locations. For instances where a mask consists of multiple disconnected fragments (e.g., from segmentation errors), we use the segment minimizing $\frac{\text{center offset}}{\text{\# of pixels}}$, which generally keeps the bounding box to the intended target rather than including peripheral errors. Finally, as with ODN [15], we find the number of training iterations for each test set using the best corresponding validation performance, which we check at every hundredth of the total training iterations.

Table 1. **Object Depth via Motion and Detection Results**. ODMD inputs have full x, y, z motion and bounding boxes.

Config. ID	Object Depth Method	Train Data	Mean Percent Error (20)				
				Perturb			All
			Norm.	Camera Motion	Object Detect.	Robot	Sets
DBox_p	f_{Rel} (14)	Perturb	1.7	2.5	2.5	**11.2**	**4.5**
DBox_{Abs}	f_{Abs} (13)	Perturb	1.1	**2.1**	**1.8**	13.3	4.6
DBox_p^{1M}	f_{Rel} (14)	Perturb	1.7	2.6	2.6	11.5	4.6
DBox_p^{100K}	f_{Rel} (14)	Perturb	2.2	3.0	3.0	11.7	5.0
DBox_{NS}	f_{Rel} (14)	Normal	0.5	3.9	6.4	12.5	5.8
Box_{LS}	$\hat{Z}_n$ (9)	N/A	**0.0**	4.5	21.6	21.2	11.8
DBox_p^z	f_{Rel} (14)	z-axis	12.9	12.5	15.0	22.0	15.6

Table 2. **Object Depth via Motion and Segmentation Results**. ODMS inputs have z-axis motion and binary segmentation masks.

Object Depth Method	Vision Input	Mean Percent Error (20)				All
		Simulated				
		Normal	Perturb	Robot	Driving	Sets
Learning-based Methods						
DBox_p^z	Detection	11.8	20.3	**11.5**	**24.8**	**17.1**
$\text{ODN}_{\ell r}$ [15]	Segmentation	8.6	17.9	13.1	31.7	17.8
$\text{ODN}_{\ell p}$ [15]	Segmentation	11.1	**13.0**	22.2	29.0	18.8
ODN_ℓ [15]	Segmentation	**8.3**	18.2	19.3	30.1	19.0
$\text{DBox}_{\text{NS}}^z$	Detection	9.2	31.6	39.3	37.3	29.3
$\text{DBox}_{\text{Abs}}^z$	Detection	21.3	25.5	20.4	53.1	30.1
Analytical Methods						
Box_{LS} (9)	Detection	13.7	36.6	**17.6**	**33.3**	**25.3**
VOS-DE [14]	Segmentation	**7.9**	**33.6**	32.6	36.0	27.5

We test a new application using DBox with a camera phone in Section 4.4. We take sets of ten pictures with target objects in view, and we estimate the camera movement between images using markings on the ground (e.g., sections of sidewalk). Like the ODMD Robot Set in Section 3.3, the ground truth object depth is manually measured, and we detect objects using Faster R-CNN fine-tuned on separate training images. Overall, we generate 46 camera phone object depth estimation examples across a variety of settings.

4.2. ODMD Dataset

We provide ablative ODMD Test Set results in Table 1. We evaluate the five full-motion DBox configurations, z-axis only DBox_p^z, and analytical solution Box_{LS} (9). All results use $n = 10$ observations (same in Sections 4.3-4.4), and "All Sets" is an aggregate score across all test sets.

DBox_p has the best result for the Robot Set and overall. DBox_{Abs} comes in second overall and has the best result for the Perturb Camera Motion and Object Detection sets. However, DBox_{Abs} is the least accurate of any full-motion DBox configuration on the Robot Set. Essentially, DBox_{Abs} gets a performance boost from a camera movement range- and depth-based prior (i.e., $\mathbf{p}_i, \mathbf{X}$ and f_{Abs} in (13)) at the cost of generalization to other domains with different movement and depth profiles. In Section 4.3, this trend becomes most apparent for $\text{DBox}_{\text{Abs}}^z$ in the driving domain.

Analytical solution Box_{LS} is perfect on the error-free Normal Set but performs worse relative to other methods on sets with input errors, especially object detection errors. Similarly, DBox_{NS} has a great result on the Normal Set but is the least accurate overall of the full-motion DBox configurations. Intuitively, DBox_{NS} trains without perturbations, so it is more susceptible to input errors. In Section 4.3, this trend becomes most apparent for $\text{DBox}_{\text{NS}}^z$ in the robot domain, which has many object segmentation errors [15].

DBox_p^z uses only z-axis camera motion and is the least accurate overall. Thus, for applications with full x, y, z motion, this result shows the importance of training on examples with motion parallax and using all three camera motion inputs to determine depth. We provide more results comparing depth estimation cues in the supplementary material.
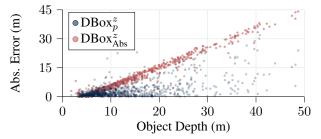

Figure 5. **Absolute Error vs. Object Depth for the Driving Set**.

We find that we can train the state-of-the-art DBox_p much faster if needed. DBox_p^{1M} trains for one tenth the iterations of DBox_p and performs almost as well. Going further, training for one hundredth the iterations is more of a trade off, as DBox_p^{100K} has an overall 11% increase in error.

4.3. ODMS Dataset

We provide comparative results for the ODMS benchmark [15] in Table 2. We evaluate the three z-axis DBox configurations, Box_{LS}, and current state-of-the-art methods.

DBox_p^z achieves the best Robot, Driving, and overall result on the segmentation-based ODMS benchmark despite training only on detection-based data. Furthermore, while the second best method, $\text{ODN}_{\ell r}$, takes 2.6 days to train on segmentation masks [15], DBox_p^z trains in under four minutes by using a simpler bounding box-based input (12).

For analytical methods, Box_{LS} achieves the best Robot, Driving, and overall result. Box_{LS} improves robustness by using width- and height-based changes in scale (9), generating twice as many solvable equations as VOS-DE [14], but performs worse on Simulated sets due to mask conversion.

State-of-the-art DBox_p^z is significantly more reliable across applications than other DBox configurations. $\text{DBox}_{\text{NS}}^z$ is the least accurate Robot Set method, and $\text{DBox}_{\text{Abs}}^z$ is the least accurate Driving Set method, while DBox_p^z is the best method for both. Considering the DBox configuration differences, we attribute DBox_p^z's success to using three types of perturbations during training (18)-(19) and a dimensionless loss ($\mathcal{L}_{\text{Rel}}$ (14)). Notably, perturbations

Figure 6. **Example Depth Results for Driving**.

Figure 7. **Example Depth Results using a Camera Phone**.

from previous work are less effective (ODN$_{\ell p}$, ODN$_\ell$ [15]).

To compare DBox$_p^z$ and DBox$_{\text{Abs}}^z$ on the Driving Set, we plot the absolute error ($|Z_n - \hat{Z}_n|$) vs. ground truth object depth (Z_n) in Figure 5. Because DBox$_{\text{Abs}}^z$ predictions are biased toward short distances within its learned prior (f_{Abs} (13)), DBox$_{\text{Abs}}^z$ errors offset directly with object depth. Alternatively, DBox$_p^z$ predictions are dimensionless (f_{Rel} (14)) and less affected by long distances. We show four Driving results for DBox$_p^z$ in Figure 6, which have an error of 0.95 m (truck), 1.5 m (car), and 1.0 m and -0.59 m (pedestrians).

4.4. Depth using a Camera Phone

We check the performance of DBox$_p^z$ on all camera phone examples at every hundredth of the total training iterations, which achieves a mean percent error (20) of 6.7 in under four minutes of training. Overall, DBox$_p^z$'s robustness to imprecise camera movement and generalization to a camera with vastly different parameters than the training model is highly encouraging. We show six results in Figure 7, which have an error of 0.30 m (garden light), 0.31 m (bird house), 0.16 m (lamppost), and 0.10 m, 0.85 m, and 7.3 m (goal yellow, goal white, and goal orange).

4.5. Final Considerations for Implementation

For applications with full x, y, z camera motion, DBox$_p$ is state-of-the-art. Alternatively, for applications with only z-axis motion, DBox$_p^z$ is state-of-the-art. Thus, although each network is broadly applicable, we suggest configuring ODMD training to match an application's camera movement profile for best performance. Finally, because DBox's

prediction speed is negligible for many applications, we suggest combining predictions over many permutations of collected data to improve accuracy [15, Section 6.2].

5. Conclusions

We develop an analytical model and multiple approaches to estimate object depth using uncalibrated camera motion and bounding boxes from object detection. To train and evaluate methods in this new problem space, we introduce the **O**bject **D**epth via **M**otion and **D**etection (ODMD) benchmark dataset, which includes mobile robot experiments using a single RGB camera to locate objects. ODMD training data are extensible, configurable, and include three types of perturbations typical of camera motion and object detection. Furthermore, we show that training with perturbations improves performance in real-world applications.

Using the ODMD dataset, we train the first network to estimate object depth from motion and detection. Additionally, we develop a generalized representation of bounding boxes, camera movement, and relative depth prediction, which we show to improve general applicability across vastly different domains. Using a single ODMD-trained network with object detection *or* segmentation, we achieve state-of-the-art results on existing driving and robotics benchmarks and accurately estimate the depth of objects using a camera phone. Given the network's reliability across domains and real-time operation, we find our approach to be a viable tool to estimate object depth in mobile applications.

References

[1] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and carsten Rother. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[3] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics Automation Magazine*, 22(3):36–52, 2015.

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

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[7] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox. Self-supervised 6d object pose estimation for robot manipulation. In *The IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[8] M. Ferguson and K. Law. A 2d-3d object detection system for updating building information models with mobile robots. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[9] Steven H. Ferris. Motion parallax and absolute distance. *Journal of Experimental Psychology*, 95(2):258–263, 1972.

[10] L. Gan, R. Zhang, J. W. Grizzle, R. M. Eustice, and M. Ghaffari. Bayesian spatial kernel smoothing for scalable dense semantic mapping. *IEEE Robotics and Automation Letters (RA-L)*, 5(2), 2020.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[12] F. A. Gers and J. Schmidhuber. Recurrent nets that time and count. In *The IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, 2000.

[13] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *The IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[14] Brent Griffin, Victoria Florence, and Jason J. Corso. Video object segmentation-based visual servo control and object depth estimation on a mobile robot. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[15] Brent A. Griffin and Jason J. Corso. Learning object depth from camera motion and video object segmentation. In *The European Conference on Computer Vision (ECCV)*, 2020.

[16] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[20] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[21] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, Caner Sahin, Fabian Manhardt, Federico Tombari, Tae-Kyun Kim, Jiri Matas, and Carsten Rother. Bop: Benchmark for 6d object pose estimation. In *The European Conference on Computer Vision (ECCV)*, 2018.

[22] William H. Ittelson. Size as a cue to distance: Radial motion. *The American Journal of Psychology*, 64(2):188–202, 1951.

[23] Y. Kasten, M. Galun, and R. Basri. Resultant based incremental recovery of camera pose from pairwise matches. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[24] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[25] Salman H. Khan, Yulan Guo, Munawar Hayat, and Nick Barnes. Unsupervised primitive discovery for improved 3d generative modeling. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014.

[27] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollar. Panoptic segmentation. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[28] Sudhakar Kumawat and Shanmuganathan Raman. Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[29] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic. Cosypose: Consistent multi-view multi-object 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[30] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *The European Conference on Computer Vision (ECCV)*, 2014.

[33] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics (T-RO)*, 2017.

[35] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *The IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[36] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *The IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[37] Kiru Park, Timothy Patten, and Markus Vincze. Neural object learning for 6d pose estimation using a few cluttered images. In *The European Conference on Computer Vision (ECCV)*, 2020.

[38] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[39] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[40] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28 (NIPS)*, 2015.

[42] Brian Rogers and Maureen Graham. Motion parallax as an independent cue for depth perception. *Perception*, 8(2):125–134, 1979.

[43] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[44] Caner Sahin, Guillermo Garcia-Hernando, Juil Sock, and Tae-Kyun Kim. A review on object pose recovery: From 3d bounding box detectors to full 6d pose estimators. *Image and Vision Computing*, 96:103898, 2020.

[45] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[46] Michael T. Swanston and Walter C. Gogel. Perceived size and motion in depth from optical expansion. *Perception & Psychophysics*, 39:309–326, 1986.

[47] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *The IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[49] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martin-Martin, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[51] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[52] Ui Yamaguchi, Fuminori Saito, Koichi Ikeda, and Takashi Yamamoto. Hsr, human support robot as research and development platform. *The Abstracts of the international conference on advanced mechatronics : toward evolutionary fusion of IT and mechatronics : ICAM*, 2015.6:39–40, 2015.

[53] Takashi Yamamoto, Koji Terada, Akiyoshi Ochiai, Fuminori Saito, Yoshiaki Asahara, and Kazuto Murase. Development of human support robot as the research platform of a domestic mobile manipulator. *ROBOMECH Journal*, 6(1):4, 2019.

[54] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[55] Rui Zhu, Xingyi Yang, Yannick Hold-Geoffroy, Federico Perazzi, Jonathan Eisenmann, Kalyan Sunkavalli, and Manmohan Chandraker. Single view metrology in the wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.