Learning Object Depth from Camera Motion and Video Object Segmentation

Brent A. Griffin and Jason J. Corso

University of Michigan
{griffb,jjcorso}@umich.edu

Abstract. Video object segmentation, i.e., the separation of a target object from background in video, has made significant progress on real and challenging videos in recent years. To leverage this progress in 3D applications, this paper addresses the problem of learning to estimate the depth of segmented objects given some measurement of camera motion (e.g., from robot kinematics or vehicle odometry). We achieve this by, first, introducing a diverse, extensible dataset and, second, designing a novel deep network that estimates the depth of objects using only segmentation masks and uncalibrated camera movement. Our data-generation framework creates artificial object segmentations that are scaled for changes in distance between the camera and object, and our network learns to estimate object depth even with segmentation errors. We demonstrate our approach across domains using a robot camera to locate objects from the YCB dataset and a vehicle camera to locate obstacles while driving.

Keywords: Depth Estimation, Video Object Segmentation, Robotics

1 Introduction

Perceiving environments in three dimensions (3D) is important for locating objects, identifying free space, and motion planning in robotics and autonomous vehicles. Although these domains typically rely on 3D sensors to measure depth and identify free space (e.g., LiDAR [12] or RGBD cameras [10]), classifying and understanding raw 3D data is a challenging and ongoing area of research [21, 23, 30, 40]. Alternatively, RGB cameras are less expensive and more ubiquitous than 3D sensors, and there are many more datasets and methods based on RGB images [8, 17, 27]. Thus, even when 3D sensors are available, RGB images remain a critical modality for understanding data and identifying objects [11, 52].

To identify objects in a sequence of images, video object segmentation (VOS) addresses the problem of densely labeling target objects in video. VOS is a hotly studied area of video understanding, with frequent developments and improving performance on challenging VOS benchmark datasets [25, 38, 39, 49, 55]. These algorithmic advances in VOS support learning object class models [36, 47], scene parsing [28, 48], action recognition [31, 43, 44], and video editing applications [5].

Given that many VOS methods perform well in unstructured environments, in this work, we show that VOS can similarly support 3D perception for robots

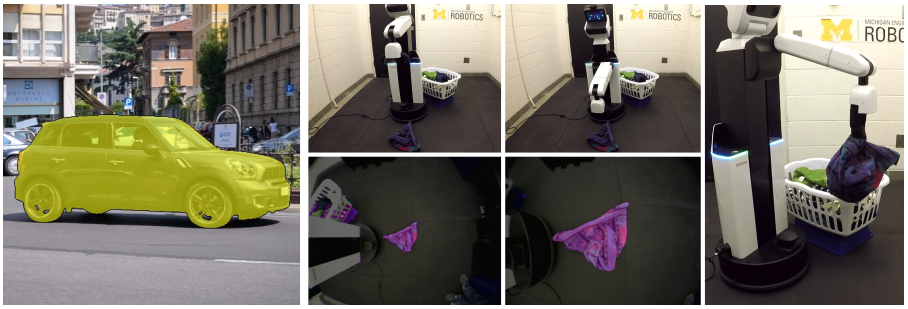

Fig. 1. Depth from Video Object Segmentation. Video object segmentation algorithms can densely segment target objects in a variety of settings (DAVIS [38], *left*). Given object segmentations and a measure of camera movement (e.g., from vehicle odometry or robot kinematics, *right*), our network can estimate an object's depth

and autonomous vehicles. We take inspiration from work in psychology that establishes how people perceive depth motion from the optical expansion or contraction of objects [19, 46], and we develop a deep network that learns object depth estimation from uncalibrated camera motion and video object segmentation (see Fig. 1). We depict our optical expansion model in Fig. 2, which uses a moving pinhole camera and binary segmentation masks for an object in view. To estimate an object's depth, we only need segmentations at two distances with an estimate of relative camera movement. Notably, most autonomous hardware platforms already measure movement, and even hand-held devices can track movement using an inertial measurement unit or GPS. Furthermore, although we do not study it here, if hardware-based measurements are not available, structure from motion is also plausible to recover camera motion [20, 34, 42].

In recent work [14], we use a similar model for VOS-based visual servo control, depth estimation, and mobile robot grasping. However, our previous analytic depth estimation method does not adequately account for segmentation errors. For real-world objects in complicated scenes, segmentation quality can change among frames, with typical errors including: incomplete object segmentation, partial background inclusion, or segmenting the wrong object. Thus, we develop and train a deep network that learns to accommodate segmentation errors and reduces object depth estimation error from [14] by as much as 59%.

The first contribution of our paper is developing a learning-based approach to object depth estimation using motion and segmentation, which we experimentally evaluate in multiple domains. To the best of our knowledge, this work is the first to use a learned, segmentation-based approach to depth estimation, which has many advantages. First, we use segmentation masks as input, so our network does not rely on application-specific visual characteristics and is useful in multiple domains. Second, we process a series of observations simultaneously, thereby mitigating errors associated with any individual camera movement or segmentation mask. Third, our VOS implementation operates on streaming video and

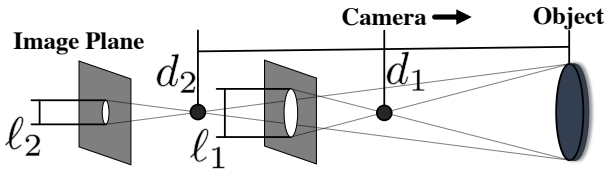

Fig. 2. Optical Expansion and Depth. An object's projection onto the image plane scales inversely with the depth between the camera and object. We determine an object's depth (d) using video object segmentation, relative camera movement, and corresponding changes in scale (ℓ). In this example, $d_1 = \frac{d_2}{2}$, $\ell_1 = 2\ell_2$, and $d_1\ell_1 = d_2\ell_2$

our method, using a single forward pass, runs in real-time. Fourth, our approach only requires a single RGB camera and relative motion (no 3D sensors). Finally, our depth estimation accuracy will improve with future innovations in VOS.

A second contribution of our paper is the **O**bject **D**epth via **M**otion and **S**egmentation (ODMS) dataset.[1] This is the first dataset for VOS-based depth estimation and enables learning-based algorithms to be leveraged in this problem space. ODMS data consist of a series of object segmentation masks, camera movement distances, and ground truth object depth. Due to the high cost of data collection and user annotation [3, 50], manually collecting training data would either be cost prohibitive or severely limit network complexity to avoid overfitting. Instead, we configure our dataset to continuously generate synthetic training data with random distances, object profiles, and even perturbations, so we can train networks of arbitrary complexity. Furthermore, because our network input consists simply of binary segmentation masks and distances, we show that domain transfer from synthetic training data to real-world applications is viable. Finally, as a benchmark evaluation, we create four ODMS validation and test sets with over 15,650 examples in multiple domains, including robotics and driving.

2 Related Work

We use video object segmentation (VOS) to process raw input video and output the binary segmentation masks we use to estimate object depth in this work. Unsupervised VOS usually relies on generic object motion and appearance cues [9, 16, 24, 37, 53, 54], while semi-supervised VOS segments objects that are specified in user-annotated examples [1, 6, 15, 32, 35, 58]. Thus, semi-supervised VOS can learn a specific object's visual characteristics and reliably segment dynamic *or* static objects. To segment objects in our robot experiments, we use One-Shot Video Object Segmentation (OSVOS) [2]. OSVOS is state-of-the-art in VOS, has influenced other leading methods [33, 51], and does not require temporal consistency (OSVOS segments frames independently). During robot experiments, we apply OSVOS models that have been pre-trained with annotated examples of each object rather than annotating an example frame at inference time.

[1] Dataset and source code website: `https://github.com/griffbr/ODMS`

We take inspiration from many existing datasets in this work. VOS research has benefited from benchmark datasets like SegTrackv2 [49, 25], DAVIS [38, 39], and YouTube-VOS [55], which have provided increasing amounts of annotated training data. The recently developed MannequinChallenge dataset [26] trained a network to predict dense depth maps from videos with people, with improved performance when given an additional human-mask input. Among automotive datasets, Cityscapes [7] focuses on *semantic segmentation* (i.e., assigning class labels to all pixels), KITTI [13] includes benchmarks separate from segmentation for single-image depth completion and prediction, and SYNTHIA [41] has driving sequences with simultaneous ground truth for semantic segmentation and depth images. In this work, our ODMS dataset focuses on **O**bject **D**epth via **M**otion and **S**egmentation, establishing a new benchmark for segmentation-based 3D perception in robotics and driving. In addition, ODMS is arbitrarily extensible, which makes learning-based methods feasible in this problem space.

3 Optical Expansion Model

Our optical expansion model (Fig. 2) forms the theoretical underpinning for our learning-based approach in Section 4 and ODMS dataset in Section 5. In this section, we derive the complete model and analytic solution for segmentation-based depth estimation. We start by defining the inputs we use to estimate depth. Assume we are given a set of $n \geq 2$ observations that consist of masks

$$\mathsf{M} := \{\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_n\} \tag{1}$$

segmenting an object and corresponding camera positions on the optical axis

$$\mathbf{z} := \{z_1, z_2, \cdots, z_n\}. \tag{2}$$

Each binary mask image $\mathbf{M}_i$ consists of pixel-level labels where 1 indicates a pixel belongs to a specific segmented object and 0 is background. For the solutions in this work, the optical axis's origin and absolute position of $\mathbf{z}$ is inconsequential.

3.1 Relating Depth and Scale

We use changes in scale of an object's segmentation mask to estimate depth. As depicted in Fig. 2, we relate depth and scale across observations using

$$d_i \ell_i = d_j \ell_j \implies \frac{\ell_j}{\ell_i} = \frac{d_i}{d_j}, \tag{3}$$

where ℓ_i is the object's projected scale in $\mathbf{M}_i$, d_i is the distance on the optical axis from z_i to the visible perimeter of the segmented object, and $\frac{\ell_j}{\ell_i}$ is the object's change in scale between $\mathbf{M}_i$ and $\mathbf{M}_j$. Notably, it is more straightforward to track changes in scale using area (i.e., the sum of mask pixels) than length measurements. Thus, we use Galileo Galilei's Square-cube law to modify (3) as

$$a_j = a_i \left(\frac{\ell_j}{\ell_i}\right)^2 \implies \frac{\ell_j}{\ell_i} = \frac{\sqrt{a_j}}{\sqrt{a_i}} = \frac{d_i}{d_j}, \tag{4}$$

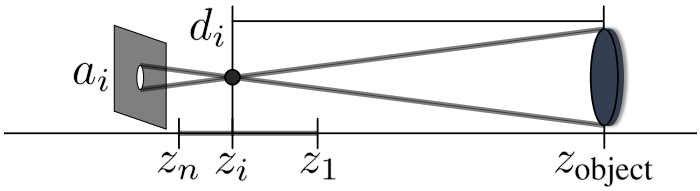

Fig. 3. Calculating Object Depth. First, we define d_i in terms of its component parts z_{object} and z_i (6). Second, we relate measured changes in camera pose z_i and segmentation area a_i across observations (7). Finally, we solve for z_{object} using (8)

where a_i is an object's projected area at d_i and $\frac{\sqrt{a_j}}{\sqrt{a_i}}$ is equal to the change in scale between $\mathbf{M}_i$ and $\mathbf{M}_j$. Combining (3) and (4), we relate observations as

$$d_i\sqrt{a_i} = d_j\sqrt{a_j} = c, \tag{5}$$

where c is a constant corresponding to an object's orthogonal surface area.

3.2 Object Depth Solution

To find object depth d_i in (5), we first redefine d_i in terms of its components as

$$d_i := z_i - z_{\text{object}}, \tag{6}$$

where z_{object} is the object's static position on the optical axis and $\dot{z}_{\text{object}} = 0$ (see Fig. 3). Substituting (6) in (5), we can now relate observations as

$$(z_i - z_{\text{object}})\sqrt{a_i} = (z_j - z_{\text{object}})\sqrt{a_j} = c. \tag{7}$$

From (7), we can solve z_{object} from any two unique observations ($z_i \neq z_j$) as

$$z_{\text{object}} = \frac{z_i\sqrt{a_i} - z_j\sqrt{a_j}}{\sqrt{a_i} - \sqrt{a_j}} = \frac{z_i - z_j\frac{\sqrt{a_j}}{\sqrt{a_i}}}{1 - \frac{\sqrt{a_j}}{\sqrt{a_i}}}. \tag{8}$$

Substituting z_{object} in (6), we can now find object depth d_i at any observation.

4 Learning Object Depth from Camera Motion and Video Object Segmentation

Using the optical expansion model from Section 3, we design a deep network, **O**bject **D**epth **N**etwork (ODN), that learns to predict the depth of segmented objects given a series of binary masks M (1) and changes in camera position $\mathbf{z}$ (2). To keep ODN broadly applicable, we formulate a normalized relative distance input in Section 4.1. In Sections 4.2 and 4.3, we derive three unique losses for learning depth estimation. After some remarks on using intermediate observations in Section 4.4, we detail our ODN architecture in Section 4.5.

4.1 Normalized Relative Distance Input

To learn to estimate a segmented object's depth, we first derive a normalized relative distance input that increases generalization. As in Section 3, assume we are given a set of n segmentation masks M with corresponding camera positions $\mathbf{z}$. We can use M and $\mathbf{z}$ as inputs to predict object depth, however, a direct $\mathbf{z}$ input enables a learned prior based on absolute camera position, which limits applicability at inference. To avoid this, we define a relative distance input

$$\Delta\mathbf{z} := \{z_2 - z_1, z_3 - z_1, \cdots, z_n - z_1\}, \tag{9}$$

where $z_1, z_2, \cdots, z_n$ are the sorted $\mathbf{z}$ positions with the minimum z_1 closest to the object (see Fig. 3) and $\Delta\mathbf{z} \in \mathbb{R}^{n-1}$. Although $\Delta\mathbf{z}$ consists only of relative changes in position, it still requires learning a specific SI unit of distance and enables a prior based on camera movement range. Thus, we normalize (9) as

$$\bar{\mathbf{z}} := \left\{ \frac{z_i - z_1}{z_n - z_1} | z \in \mathbf{z}, 1 < i < n \right\}, \tag{10}$$

where $z_n - z_1$ is the camera move range, $\frac{z_i - z_1}{z_n - z_1} \in (0, 1)$, and $\bar{\mathbf{z}} \in \mathbb{R}^{n-2}$.

Using $\bar{\mathbf{z}}$ as our camera motion input increases the general applicability of ODN. First, $\bar{\mathbf{z}}$ uses the relative difference formulation, so ODN does not learn to associate depth with an absolute camera position. Second, $\bar{\mathbf{z}}$ is dimensionless, so our trained ODN can use camera movements on the scale of millimeters or kilometers (it makes no difference). Finally, $\bar{\mathbf{z}}$ is made a more compact motion input by removing the unnecessary constants $\frac{z_1 - z_1}{z_n - z_1} = 0$ and $\frac{z_n - z_1}{z_n - z_1} = 1$ in (10).

4.2 Normalized Relative Depth Loss

Our basic depth loss, given input masks M (1) and relative distances $\Delta\mathbf{z}$ (9), is

$$\mathcal{L}_d(\mathbf{W}) := \left| d_1 - f_d(\mathsf{M}, \Delta\mathbf{z}, \mathbf{W}) \right|, \tag{11}$$

where $\mathbf{W}$ are the trainable network parameters, d_1 is the ground truth object depth at z_1 (6), and $f_d \in \mathbb{R}$ is the predicted depth. To use the normalized distance input $\bar{\mathbf{z}}$ (10), we modify (11) and define a normalized depth loss as

$$\mathcal{L}_{\bar{d}}(\mathbf{W}) := \left| \frac{d_1}{z_n - z_1} - f_{\bar{d}}(\mathsf{M}, \bar{\mathbf{z}}, \mathbf{W}) \right|, \tag{12}$$

where $\frac{d_1}{z_n - z_1}$ is the normalized object depth and $f_{\bar{d}}$ is a dimensionless depth prediction that is in terms of the input camera movement range. To use $f_{\bar{d}}$ at inference, we multiply the normalized output $f_{\bar{d}}$ by $(z_n - z_1)$ to find d_1.

4.3 Relative Scale Loss

We increase depth accuracy and simplify ODN's prediction by learning to estimate relative changes in segmentation scale. In Section 4.2, we define loss

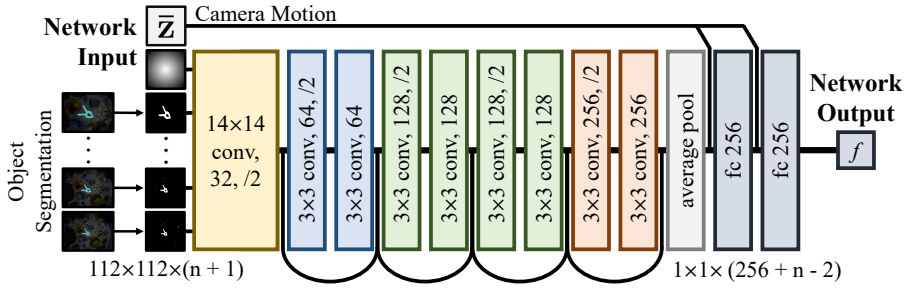

Fig. 4. Object Depth Network Architecture

functions that use a similar input-output paradigm to the analytic solution in Section 3. However, training ODN to directly predict depth requires learning many operations. Alternatively, if ODN only predicts the relative change in segmentation scale, we can finish calculating depth using (8). Thus, we define a loss for predicting the relative scale as

$$\mathcal{L}_\ell(\mathbf{W}) := \left| \frac{\ell_n}{\ell_1} - f_\ell(\mathsf{M}, \bar{\mathbf{z}}, \mathbf{W}) \right|, \tag{13}$$

where $\frac{\ell_n}{\ell_1} = \frac{d_1}{d_n} \in (0, 1)$ (3) is the ground truth distance-based change in scale between $\mathbf{M}_n$ and $\mathbf{M}_1$ and f_ℓ is the predicted scale change. To use f_ℓ at inference, we output $f_\ell \approx \frac{\ell_n}{\ell_1}$ and, using (4) to substitute $\frac{\ell_j}{\ell_i}$ for $\frac{\sqrt{a_j}}{\sqrt{a_i}}$ in (8), find z_{object} as

$$z_{\text{object}} = \frac{z_1 - z_n f_\ell}{1 - f_\ell} \approx \frac{z_1 - z_n \left(\frac{\ell_n}{\ell_1}\right)}{1 - \left(\frac{\ell_n}{\ell_1}\right)}. \tag{14}$$

After finding z_{object} in (14), we use (6) to find object depth as $d_1 = z_1 - z_{\text{object}}$.

4.4 Remarks on using Intermediate Observations

Although the ground truth label d_1 in (11)-(12) is determined only by camera position z_1 and label $\frac{\ell_n}{\ell_1}$ in (13) is determined only by endpoint masks $\mathbf{M}_n$, $\mathbf{M}_1$, we emphasize that intermediate mask and distance inputs are still useful. Consider that, first, the ground truth mask scale monotonically decreases across all observations (i.e., $\forall i, \ell_{i+1} < \ell_i$). Second, the distance inputs make it possible to extrapolate d_1 and $\frac{\ell_n}{\ell_1}$ from intermediate changes in scale. Third, if z_1, z_n, $\mathbf{M}_1$, or $\mathbf{M}_n$ have significant errors, intermediate observations provide the best prediction for d_1 or $\frac{\ell_n}{\ell_1}$. Finally, experiments in Section 6.1 show that intermediate observations improve performance for networks trained on (11), (12), or (13).

4.5 Object Depth Estimation Network Architecture

Our ODN architecture is shown in Fig. 4. The input to the first convolution layer consists of n 112×112 binary segmentation masks and, for three configurations

in Section 6.1, a radial image. The first convolution layer uses 14×14 kernels, and the remaining convolution layers use 3×3 kernels in four residual blocks [18]. After average pooling the last residual block, the relative camera position (e.g., $\bar{\mathbf{z}}$) is included with the input to the first two fully-connected layers, which use ReLU activation and 20% dropout for all inputs during training [45]. After the first two fully-connected layers, our ODN architecture ends with one last fully-connected neuron that, depending on chosen loss, is the output object depth $f_d(\mathsf{M}, \Delta\mathbf{z}, \mathbf{W}) \in \mathbb{R}$ using (11), normalized object depth $f_{\bar{d}}(\mathsf{M}, \bar{\mathbf{z}}, \mathbf{W})$ using (12), or relative scale $f_\ell(\mathsf{M}, \bar{\mathbf{z}}, \mathbf{W})$ using (13).

5 ODMS Dataset

To train our object depth networks from Section 4, we introduce the **O**bject **D**epth via **M**otion and **S**egmentation dataset (ODMS). In Section 5.1, we explain how ODMS continuously generates new labeled training data, making learning-based techniques feasible in this problem space. In Section 5.2, we describe the robotics-, driving-, and simulation-based test and validation sets we develop for evaluation. Finally, in Section 5.3, we detail our ODMS training implementation.

5.1 Generating Random Object Masks at Scale

Camera Distance and Depth We generate new training data by, first, determining n random camera distances (i.e., $\mathbf{z}$ (2)) for each training example. To make ODMS configurable, assume we are given a minimum camera movement range ($\Delta z_{\min}$) and minimum and maximum object depths ($d_{\min}$, $d_{\max}$). Using these parameters, we define distributions for uniform random variables to find the endpoints

$$z_1 \sim \mathcal{U}[d_{\min}, d_{\max} - \Delta z_{\min}], \tag{15}$$

$$z_n \sim \mathcal{U}[z_1 + \Delta z_{\min}, d_{\max}], \tag{16}$$

and, for $1 < i < n$, the remaining intermediate camera positions

$$z_i \sim \mathcal{U}(z_1, z_n). \tag{17}$$

Using (15)-(17) to select $\mathbf{z} = \{z_1, \cdots, z_n\}$ ensures that the random camera movement range is independent of the number of observations n. For the object depth label d_1, we choose an optical axis such that $z_{\text{object}} = 0$ and $d_1 = z_1$ (6). We generate data in this work using $\Delta z_{\min} = d_{\min} = 0.1$ m and $d_{\max} = 0.7$ m.

Random Object Contour and Binary Masks After determining $\mathbf{z}$, we generate a random object with n binary masks (i.e., M (1)) scaled for each distance in $\mathbf{z}$ (see Fig. 5). To make each object unique, we randomly select parameters that change the object's size ($s_{\mathbf{p}}$), number of contour points ($n_{\mathbf{p}}$), and contour smoothness (r_B, ρ_B). In this work, we randomly select $s_{\mathbf{p}}$ from

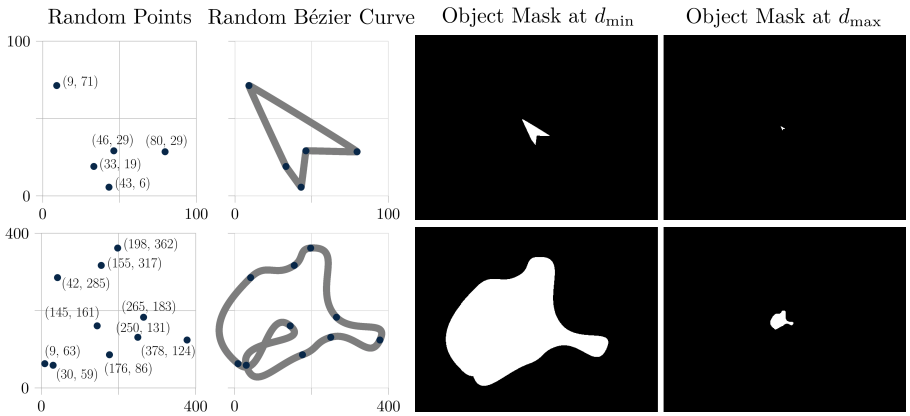

Fig. 5. Generating Random Object Masks at Scale. Initializing from a random number of points within a variable boundary (*left*), random curves complete the contour of each simulated object (*middle left*). These contours are then scaled for each simulated distance and output as a filled binary mask (*right*). Each generated object is unique

$\{100, 200, 300, 400\}$ and $n_{\mathbf{p}}$ from $\{3, 4, \cdots, 10\}$. Using $s_{\mathbf{p}}$ and $n_{\mathbf{p}}$, we select each of the random initial contour points, $\mathbf{p}_i \in \mathbb{R}^2$ for $1 \leq i \leq n_{\mathbf{p}}$, as

$$\mathbf{p}_i = [x_i, y_i]', \; x_i \sim \mathcal{U}[0, s_{\mathbf{p}}], \; y_i \sim \mathcal{U}[0, s_{\mathbf{p}}]. \tag{18}$$

To complete the object's contour, we use cubic Bézier curves with random smoothness to connect each set of adjacent coordinates $\mathbf{p}_i$, $\mathbf{p}_j$ from (18). Essentially, r_B and ρ_B determine polar coordinates for the two intermediate Bézier control points of each curve. $\arctan(\rho_B)$ is the rotation of a control point away from the line connecting $\mathbf{p}_i$ and $\mathbf{p}_j$, while r_B is the relative radius of a control point away from $\mathbf{p}_i$ (e.g., $r_B = 1$ has a radius of $\|\mathbf{p}_i - \mathbf{p}_j\|$). In this work, we randomly select r_B from $\{0.01, 0.05, 0.2, 0.5\}$ and ρ_B from $\{0.01, 0.05, 0.2\}$ for each object. In general, lower r_B and ρ_B values result in a more straight-edged contour, while higher values result in a more curved and widespread contour. As two illustrative examples in Fig. 5, the top "straight-edged" object uses $r_B = \rho_B = 0.01$ and the bottom "curved" object uses $r_B = 0.5$ and $\rho_B = 0.2$.

To simulate object segmentation over multiple distances, we scale the generated contour to match each distance $z_i \in \mathbf{z}$ from (15)-(17) and output a set of binary masks $\mathbf{M}_i \in \mathsf{M}$ (1). We let the initial contour represent the object's image projection at $d_{\min}$, and designate this initial scale as $\ell_{\min} = 1$. Having chosen an optical axis such that $z_{\text{object}} = 0$ in (6) (i.e., $d_i = z_i$), we modify (3) to find the contour scale of each mask, ℓ_i for $1 \leq i \leq n$, as

$$\ell_i = \frac{d_{\min}\ell_{\min}}{d_i} = \frac{d_{\min}}{z_i}. \tag{19}$$

After finding ℓ_i, we scale, fill, and add the object contour to each mask $\mathbf{M}_i$. In this work, we position the contour by centering the scaled boundary $(\ell_i s_{\mathbf{p}})$ in a 480×640 mask. Our complete object-generating process is shown in Fig. 5.

Fig. 6. Robot Experiment Data. HSR view of validation (yellow bin) and test set objects (blue bin) using head-mounted RGBD camera (*left*). Unfortunately, the depth image is missing many objects (*middle left*). However, using 4,400 robot-collected examples (*middle right*), we find that segmentation-based object depth works (*right*)

5.2 Robotics, Driving, and Simulation Validation and Test Sets

We test object depth estimation in a variety of settings using four ODMS validation and test sets. These are based on robot experiments, driving, and simulated data with and without perturbations and provide a repeatable benchmark for ablation studies and future methods. All examples include $n \geq 10$ observations.

Robot Validation and Test Set Our robot experiment data provide an evaluation for object depth estimation from a physical platform using video object segmentation on real-world objects in a practical use case. We collect data using a Toyota Human Support Robot (HSR), which has a 4-DOF manipulator arm with an end effector-mounted wide-angle grasp camera [56, 57]. Using HSR's prismatic torso, we collect 480×640 grasp-camera images as the end effector approaches an object of interest, with the intent that HSR can estimate the object's depth using motion and segmentation. We use 16 custom household objects for our validation set and 24 YCB objects [4] for our test set (Fig. 6, left). For each object, we collect 30 images distanced 2 cm apart of the object in isolation and, as an added challenge, 30 more images in a cluttered setting (see Fig. 6, middle right). The ground truth object depth (d_1) is manually measured at the closest camera position and propagated to the remaining images using HSR's kinematics and encoder values, which also measure camera positions ($\mathbf{z}$). To generate binary masks (M), we segment objects using OSVOS [2], which we fine-tune on each object using three annotated images from outside of the validation and test sets. We vary the input camera movement range between 18-58 cm and object depth (d_1) between 11-60 cm to generate 4,400 robot object depth estimation examples (1,660 validation and 2,240 test).

Driving Validation and Test Set Our driving data provide an evaluation for object depth estimation in a faster moving automotive domain with greater camera movement and depth distances. Our goal is to track driving obstacles using an RGB camera, segmentation, and vehicle odometry. Challenges include changing object perspectives, camera rotation from vehicle turning, and moving objects. We collect data using the SYNTHIA Dataset [41], which includes ground

Fig. 7. Driving Test Set Examples and Results. The ODN$_\ell$ object depth error is -6 and -23 cm for the pedestrians, -10 cm for the bicycle, and -4 cm for the car

truth semantic segmentation, depth images, and vehicle odometry in a variety of urban scenes and weather conditions. To generate binary masks (M), we use SYNTHIA's semantic segmentation over a series of 760×1280 frames for unique instances of pedestrians, bicycles, and cars (see Fig. 7). For each instance, the ground truth object depth (d_1) is the mean depth image values contained within corresponding mask $\mathbf{M}_1$. As the vehicle moves, we track changes in camera position ($\mathbf{z}$) along the optical axis of position z_1. With an input camera movement range between 4.2-68 m and object depth (d_1) between 1.5-62 m, we generate 1,250 driving object depth estimation examples (500 validation and 750 test).

Simulation Validation and Test Sets Finally, we generate a set of normal and perturbation-based data for simulated objects. The normal set and the continuously-generated training data we use in Section 5.3 both use the same mask-generating procedure from Section 5.1, so the normal set provides a consistent evaluation for the type of simulated objects we use during training.

To test robustness for segmentation errors, we also generate a set of simulated objects with random perturbations added to each mask, $\mathbf{M}_i$ for $1 \leq i \leq n$, as

$$p_i \sim \mathcal{N}(0,1),\ \mathbf{M}_{i,p} = \begin{cases} \text{dilate}(\mathbf{M}_i, \lfloor p_i + 0.5 \rfloor) & \text{if } p_i \geq 0 \\ \text{erode}(\mathbf{M}_i, \lfloor p_i + 0.5 \rfloor) & \text{if } p_i < 0 \end{cases}, \tag{20}$$

where $\mathcal{N}(0,1)$ is a Gaussian distribution with $\mu = 0$, $\sigma^2 = 1$, p_i randomly determines the perturbation type and magnitude, and $\mathbf{M}_{i,p}$ is the perturbed version of initial mask $\mathbf{M}_i$. Notably, the sign of p_i determines a dilation or erosion perturbation, and the rounded magnitude of p_i determines the number of iterations using a square connectivity equal to one. When generating perturbed masks $\mathbf{M}_{i,\text{p}}$, we make no other changes to input data or ground truth labels.

We generate 5,000 object depth estimation examples (2,000 validation and 3,000 test) for both the normal and perturbation-based simulation sets.

5.3 Training Object Depth Networks using ODMS

Using the architecture in Section 4.5, we train networks for depth loss $\mathcal{L}_d$ (11), normalized relative depth loss $\mathcal{L}_{\bar{d}}$ (12), and relative scale loss $\mathcal{L}_\ell$ (13). We call these networks ODN$_d$, ODN$_{\bar{d}}$, and ODN$_\ell$ respectively. We train each network

Table 1. ODMS Test Set Results

Config. ID	Object Depth Method	n Input Masks	Mean Percent Error (21)				
			Robot Objects	Driving Objects	Simulated Objects		All Sets
					Normal	Perturb	
ODN_ℓ	$\mathcal{L}_\ell$ (13)	10	19.3	**30.1**	8.3	18.2	**19.0**
$\text{ODN}_{\bar{d}}$	$\mathcal{L}_{\bar{d}}$ (12)	10	18.5	30.9	8.2	18.5	19.0
ODN_d	$\mathcal{L}_d$ (11)	10	**18.1**	47.5	**5.1**	**11.2**	20.5
VOS-DE	[14]	10	32.6	36.0	7.9	33.6	27.5

with a batch size of 512 randomly-generated training examples using the framework in Section 5.1 with $n = 10$ observations per prediction. We train each network for 5,000 iterations using the Adam Optimizer [22] with a 1×10^{-3} learning rate, which takes 2.6 days using a single GPU (GTX 1080 Ti). Notably, the primary time constraint for training is generating new masks, and we can train a similar configuration with $n = 2$ for 5,000 iterations in 15 hours.

6 Experimental Results

Our primary experiments and analysis use the four ODMS test sets. For each test set, the number of network training iterations is determined by the best validation performance, which we check at every ten training iterations. We determine the effectiveness of each depth estimation method using the mean percent error for each test set, which is calculated for each example as

$$\text{Percent Error} = \left| \frac{d_1 - \hat{d}_1}{d_1} \right| \times 100\%, \tag{21}$$

where d_1 and $\hat{d}_1$ are ground truth and predicted object depth at final pose z_1.

6.1 ODMS Test Results

Object depth estimation results for all four ODMS test sets are provided in Table 1 for our three ODN configurations and VOS-DE [14]. We use $n = 10$ observations, and "All Sets" is an aggregate score across all test sets. Notably, VOS-DE uses only the largest connected region of each mask to reduce noise.

The relative scale-based ODN_ℓ performs best on the Driving set and overall. We show a few quantitative depth estimation examples for ODN_ℓ in Fig. 6 and Fig. 7. Normalized depth-based $\text{ODN}_{\bar{d}}$ comes in second overall, and depth-based ODN_d performs best in three categories but worst in driving. Basically, ODN_d gets a performance boost from a camera movement range- and depth-based prior (i.e., $\Delta\mathbf{z}$ and f_d in (11)) at the cost of applicability to other domains where the scale of camera input and depth will vary. On the other hand, the generalization of $\text{ODN}_{\bar{d}}$ and ODN_ℓ from small distances in training to large distances in Driving is highly encouraging. VOS-DE performs the worst overall, particularly on test sets with segmentation errors or moving objects. However, VOS-DE does perform well on normal simulated objects, which only have mask discretization errors.

Table 2. ODMS Test Set Results vs. Number of Observations

Config. ID	Depth Method	Overall Mean Percent Error				Average Training Iterations			
		$n = 2$	$n = 3$	$n = 5$	$n = 10$	$n = 2$	$n = 3$	$n = 5$	$n = 10$
ODN_{ℓ}	$\mathcal{L}_{\ell}$ (13)	**20.4**	**19.9**	20.0	**19.0**	**2,590**	**3,460**	**3,060**	**3,138**
$\text{ODN}_{\bar{d}}$	$\mathcal{L}_{\bar{d}}$ (12)	22.7	20.9	**19.9**	19.0	3,993	4,330	3,265	3,588
ODN_{d}	$\mathcal{L}_{d}$ (11)	21.6	21.2	20.5	20.5	4,138	4,378	4,725	3,300
VOS-DE	[14]	50.3	29.7	27.6	27.5	N/A	N/A	N/A	N/A

Table 3. Test Results with Perturb Training Data and Radial Input Image

Config. ID	Object Depth Method	Radial Input Image	Type of Training Data	Mean Percent Error (21)				
				Robot	Driving	Simulated		All Sets
						Normal	Perturb	
Perturb Training Data								
$\text{ODN}_{\ell p}$	$\mathcal{L}_{\ell}$ (13)	No	Perturb	22.2	**29.0**	11.1	13.0	18.8
$\text{ODN}_{\bar{d}p}$	$\mathcal{L}_{\bar{d}}$ (12)	No	Perturb	25.8	31.4	11.1	13.2	20.4
ODN_{dp}	$\mathcal{L}_{d}$ (11)	No	Perturb	20.1	60.9	7.3	**8.2**	24.1
Radial Input Image								
$\text{ODN}_{\ell r}$	$\mathcal{L}_{\ell}$ (13)	Yes	Normal	**13.1**	31.7	8.6	17.9	**17.8**
$\text{ODN}_{\bar{d}r}$	$\mathcal{L}_{\bar{d}}$ (12)	Yes	Normal	15.2	30.9	8.4	18.5	18.3
ODN_{dr}	$\mathcal{L}_{d}$ (11)	Yes	Normal	13.4	48.6	**5.6**	11.2	19.7

Results on Changing the Number of Observations Object depth estimation results for varied number of observations are provided in Table 2. We repeat training and validation for each new configuration to learn depth estimation with less observations. As n changes, each test set example uses the same endpoint observations (i.e., $\mathbf{M}_1, \mathbf{M}_n, z_1, z_n$). However, the $n-2$ intermediate observations are evenly distributed and do change (e.g., $n = 2$ has none). Notably, at $n = 2$, VOS-DE is equivalent to (8) and $\bar{\mathbf{z}} \in \mathbb{R}^{n-2}$ (10) gives no input to $\text{ODN}_{\bar{d}}$, ODN_{ℓ}.

ODN_{ℓ} has the most consistent and best performance for all n settings, aside from a second place to $\text{ODN}_{\bar{d}}$ at $n = 5$. ODN_{ℓ} also requires the fewest training iterations for all n. In general, $\text{ODN}_{\bar{d}}$ and ODN_{d} performance starts to decrease for $n \leq 3$. VOS-DE performance decreases most significantly at $n = 2$, having 2.5 times the error of ODN_{ℓ} at $n = 2$. Amazingly, all $n = 2$ ODN configurations outperform $n = 10$ VOS-DE. Thus, even with significantly less input data, our learning-based approach outperforms prior work.

Results with Perturbation Training Data We train each $n = 10$ ODN on continuously-generated perturbation data (20) from Section 5.2. As shown in Table 3, this improves performance for each ODN on the Perturb test set, and demonstrates that we can learn robust depth estimation for specific errors. The perturbed ODN_{ℓ} configuration, $\text{ODN}_{\ell p}$, improves performance overall and has the best Driving result of any method.

Results with Radial Input Image For our final ODMS results in Table 3, we train each $n = 10$ ODN with an added input radial image for convolution. Pixel values $\in [0, 1]$ are scaled radially from 1 at the center to 0 at each corner (see Fig. 4). This serves a similar purpose to coordinate convolution [29] but simply focuses on how centered segmentation mask regions are. This improves overall performance for each ODN, particularly on the Robot test, where objects are generally centered for grasping and peripheral segmentation errors can be ignored. Notably, $\text{ODN}_{\ell r}$ has the best Robot and overall result of any method.

6.2 Robot Object Depth Estimation and Grasping Experiments

As a live robotics demonstration, we use $\text{ODN}_{\ell r}$ to locate objects for grasping. Experiments start with HSR's grasp camera approaching an object while generating segmentation masks at 1 cm increments using pre-trained OSVOS. Once ten masks are available, $\text{ODN}_{\ell r}$ starts predicting depth as HSR continues approaching and generating masks. Because $\text{ODN}_{\ell r}$'s prediction speed is negligible compared to HSR's data-collection speed, we use the median depth estimate of multiple permutations of collected data to improve robustness against segmentation errors. Once $\text{ODN}_{\ell r}$ estimates the object to be within 20 cm of grasping, HSR stops collecting data and grasps the object at that depth. Using this active depth estimation process, we are able to successfully locate and grasp consecutive objects at varied heights in a variety of settings, including placing laundry in a basket and clearing garbage off a table (see Fig. 1). We show these robot experiments in our Supplementary Video at: `https://youtu.be/c90Fg_whjpI`.

7 Conclusions

We introduce the **O**bject **D**epth via **M**otion and **S**egmentation (ODMS) dataset, which continuously generates synthetic training data with random camera motion, objects, and even perturbations. Using the ODMS dataset, we train the first deep network to estimate object depth from motion and segmentation, leading to as much as a 59% reduction in error over previous work. By using ODMS's simple binary mask- and distance-based input, our network's performance transfers across sim-to-real and diverse application domains, as demonstrated by our results on the robotics-, driving-, and simulation-based ODMS test sets. Finally, we use our network to perform object depth estimation in real-time robot grasping experiments, demonstrating how our segmentation-based approach to depth estimation is a viable tool for real-world applications requiring 3D perception from a single RGB camera.

Acknowledgements

We thank Madan Ravi Ganesh, Parker Koch, and Luowei Zhou for various discussions throughout this work. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

1. Bao, L., Wu, B., Liu, W.: CNN in MRF: video object segmentation via inference in A cnn-based higher-order spatio-temporal MRF. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
2. Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: One-shot video object segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
3. Caelles, S., Montes, A., Maninis, K., Chen, Y., Gool, L.V., Perazzi, F., Pont-Tuset, J.: The 2018 DAVIS challenge on video object segmentation. CoRR **abs/1803.00557** (2018)
4. Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., Dollar, A.M.: Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. IEEE Robotics Automation Magazine **22**(3), 36–52 (2015)
5. Chen, D.J., Chen, H.T., Chang, L.W.: Video object cosegmentation. In: ACM International Conference on Multimedia (2012)
6. Chen, Y., Pont-Tuset, J., Montes, A., Van Gool, L.: Blazingly fast video object segmentation with pixel-wise metric learning. In: Computer Vision and Pattern Recognition (CVPR) (2018)
7. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
9. Faktor, A., Irani, M.: Video segmentation by non-local consensus voting. In: British Machine Vision Conference (BMVC) (2014)
10. Ferguson, M., Law, K.: A 2d-3d object detection system for updating building information models with mobile robots. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) (2019)
11. Florence, V., Corso, J.J., Griffin, B.: Robot-supervised learning for object segmentation. In: The IEEE International Conference on Robotics and Automation (ICRA) (2020)
12. Gan, L., Zhang, R., Grizzle, J.W., Eustice, R.M., Ghaffari, M.: Bayesian spatial kernel smoothing for scalable dense semantic mapping. IEEE Robotics and Automation Letters (RA-L) **5**(2) (2020)
13. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
14. Griffin, B., Florence, V., Corso, J.J.: 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. Griffin, B.A., Corso, J.J.: Bubblenets: Learning to select the guidance frame in video object segmentation by deep sorting frames. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
16. Griffin, B.A., Corso, J.J.: Tukey-inspired video object segmentation. In: IEEE Winter Conference on Applications of Computer Vision (WACV) (2019)
17. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-cnn. In: The IEEE International Conference on Computer Vision (ICCV) (2017)

18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
19. Ittelson, W.H.: Size as a cue to distance: Radial motion. The American Journal of Psychology **64**(2), 188–202 (1951)
20. Kasten, Y., Galun, M., Basri, R.: Resultant based incremental recovery of camera pose from pairwise matches. In: IEEE Winter Conference on Applications of Computer Vision (WACV) (2019)
21. Khan, S.H., Guo, Y., Hayat, M., Barnes, N.: Unsupervised primitive discovery for improved 3d generative modeling. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2014)
23. Kumawat, S., Raman, S.: Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
24. Lee, Y.J., Kim, J., Grauman, K.: Key-segments for video object segmentation. In: IEEE International Conference on Computer Vision (ICCV) (2011)
25. Li, F., Kim, T., Humayun, A., Tsai, D., Rehg, J.M.: Video segmentation by tracking many figure-ground segments. In: The IEEE International Conference on Computer Vision (ICCV) (2013)
26. Li, Z., Dekel, T., Cole, F., Tucker, R., Snavely, N., Liu, C., Freeman, W.T.: Learning the depths of moving people by watching frozen people. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
27. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision – (ECCV). pp. 740–755. Springer International Publishing, Cham (2014)
28. Liu, B., He, X.: Multiclass semantic video segmentation with object-level active inference. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
29. Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., Yosinski, J.: An intriguing failing of convolutional neural networks and the coordconv solution. In: Advances in Neural Information Processing Systems 31 (NIPS)
30. Liu, Y., Fan, B., Xiang, S., Pan, C.: Relation-shape convolutional neural network for point cloud analysis. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
31. Lu, J., Xu, R., Corso, J.J.: Human action segmentation with hierarchical supervoxel consistency. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
32. Luiten, J., Voigtlaender, P., Leibe, B.: Premvos: Proposal-generation, refinement and merging for video object segmentation. In: Asian Conference on Computer Vision (ACCV) (2018)
33. Maninis, K., Caelles, S., Chen, Y., Pont-Tuset, J., Leal-Taix, L., Cremers, D., Gool, L.V.: Video object segmentation without temporal information. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)
34. Mur-Artal, R., Tards, J.D.: Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. IEEE Transactions on Robotics (T-RO) (2017)
35. Oh, S.W., Lee, J.Y., Sunkavalli, K., Kim, S.J.: Fast video object segmentation by reference-guided mask propagation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)

36. Oneata, D., Revaud, J., Verbeek, J., Schmid, C.: Spatio-temporal object detection proposals. In: European Conference on Computer Vision (ECCV) (2014)
37. Papazoglou, A., Ferrari, V.: Fast object segmentation in unconstrained video. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2013)
38. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
39. Pont-Tuset, J., Perazzi, F., Caelles, S., Arbelaez, P., Sorkine-Hornung, A., Gool, L.V.: The 2017 DAVIS challenge on video object segmentation. CoRR **abs/1704.00675** (2017)
40. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
41. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: 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)
42. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
43. Soomro, K., Idrees, H., Shah, M.: Action localization in videos through context walk. In: IEEE International Conference on Computer Vision (ICCV) (2015)
44. Soomro, K., Idrees, H., Shah, M.: Predicting the where and what of actors and actions through online action localization. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
45. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**, 1929–1958 (2014)
46. Swanston, M.T., Gogel, W.C.: Perceived size and motion in depth from optical expansion. Perception & Psychophysics **39**, 309–326 (1986)
47. Tang, K., Sukthankar, R., Yagnik, J., Fei-Fei, L.: Discriminative segment annotation in weakly labeled video. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2013)
48. Tighe, J., Lazebnik, S.: Superparsing: scalable nonparametric image parsing with superpixels. International Journal of Computer Vision (2012)
49. Tsai, D., Flagg, M., Nakazawa, A., Rehg, J.M.: Motion coherent tracking using multi-label mrf optimization. International journal of computer vision **100**(2), 190–202 (2012)
50. Vijayanarasimhan, S., Grauman, K.: What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
51. Voigtlaender, P., Leibe, B.: Online adaptation of convolutional neural networks for video object segmentation. In: British Machine Vision Conference (BMVC) (2017)
52. Wang, C., Xu, D., Zhu, Y., Martin-Martin, R., Lu, C., Fei-Fei, L., Savarese, S.: Densefusion: 6d object pose estimation by iterative dense fusion. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
53. Wehrwein, S., Szeliski, R.: Video segmentation with background motion models. In: British Machine Vision Conference (BMVC) (2017)
54. Xu, C., Corso, J.J.: Libsvx: A supervoxel library and benchmark for early video processing. International Journal of Computer Vision **119**(3), 272–290 (2016)

55. Xu, N., Yang, L., Fan, Y., Yue, D., Liang, Y., Yang, J., Huang, T.S.: Youtube-vos: A large-scale video object segmentation benchmark. CoRR **abs/1809.03327** (2018)
56. Yamaguchi, U., Saito, F., Ikeda, K., Yamamoto, T.: 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)
57. Yamamoto, T., Terada, K., Ochiai, A., Saito, F., Asahara, Y., Murase, K.: Development of human support robot as the research platform of a domestic mobile manipulator. ROBOMECH Journal **6**(1), 4 (2019)
58. Yang, L., Wang, Y., Xiong, X., Yang, J., Katsaggelos, A.K.: Efficient video object segmentation via network modulation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)