Structured Prediction of Unobserved Voxels From a Single Depth Image

Anonymous CVPR submission

Paper ID 1437

Abstract

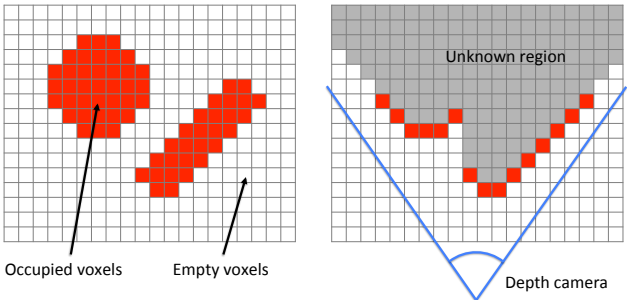
Building a complete 3D model of a scene is underconstrained given a single camera view. To gain a full volumetric model, one typically needs either multiple views, or a single view together with the strong assumption that one has a library of unambiguous training instances that fit the shape of each individual object in the scene.

We hypothesize that objects of dissimilar semantic classes often share similar shape components, enabling a limited dataset to model the shape of a wide range of objects, and hence estimate the hidden geometry of various scenes. Exploring this hypothesis, we have implemented a system that can complete the unobserved geometry of a scene using a library of simple volumetric elements learned from training data. Using a novel feature representation, we train a model to map from observed depth values to an estimate of surface shape in the neighborhood of a 3D location. Through the prototype, we validate our approach qualitatively and quantitatively on a range of indoor scenes.

1. Introduction

We broadly categorize space in our world as being 'occupied' and opaque, or 'vacant' and transparent. Depth cameras such as the Microsoft Kinect are able to give an estimate of which regions of a scene are composed of free, vacant space. However, each pixel in a depth image only makes an estimate of occupancy in front of the first solid surface encountered along that camera ray (Figure 1). The 'occlusion phenomenon' prevents any information from being measured about the occupancy of space beyond that first surface.

There are many applications, however, which critically require a complete representation of the world geometry. When a robot sets out to grasp an unknown object in an unknown scene, a 3D model is required to get there and prevent collision with the object or nearby clutter. Separately, in photo-editing, the full geometry would enable realistic shadows from a new light source to be automatically added to an image after it has been captured.



(a) Voxel world model (b) Depth rendering of world

Figure 1. We model our world as a grid of voxels, each of which is either occupied or empty (vacant). An overhead view of a coarse 2D representation of this is shown in (a). When observed by a depth camera, only the first voxel along each ray is seen. This leaves a region of unknown occupancy extending beyond the depth surface (b). The aim of our algorithm is to predict the state of the voxels in this unknown region.

Much computer vision research has been invested in reconstructing a full 3D world model from images of a scene captured from multiple viewpoints, thus coping with the effects of occlusion (e.g. [18]). Instead, we focus on the task of classifying each voxel in a 3D scene as being either 'occupied' or 'vacant' given just a single depth image from one viewpoint.

1.1. Our approach and contributions

Given a single depth image, our system predicts whether each voxel in the scene is occupied by a solid impermeable object, or free and empty. In effect, we strive to predict the voxelized output of KinectFusion [18], but having test-time input of only a single view of the scene instead of multiple views.

We achieve this by learning a mapping from local and novel semi-regional features on a depth image to a structured prediction of geometry in the region of a query point, using a general-purpose collection of training objects and scenes. We take inspiration from recent work which has segmented objects from images using silhouettes learned from different object classes [21]. This work showed that

shape can transcend class categories, enabling shape predictions to be made regardless of the accuracy of semantic classifiers. Because we care about shape, independently of semantic understanding, we are free to use training objects which differ from the objects being modeled in the scene.

The key contributions that underpin our novel depth image to voxel geometry framework are:

- *Voxlets*, a representation of multi-voxel geometry in the region of a point in a scene. We use a Random Forest to learn a mapping from a point in a depth image to a structured prediction of geometry in the region near the point.
- A novel feature representation for a point in a depth image, which captures both local and region-level shape for a single point in a depth image.

2. Related work

Previous works on completing unknown regions of visual data can be categorized against several different criteria. Some of these are given here, and we italicize the categories into which our approach falls.

Works can be classed according to their application domain, such as operating on meshes [14, 30], 2D images [12] or in *voxel space* [20]. Some works make use of highly specialist hardware to capture images e.g. [38], while we use *hardware available to consumers*. Some approaches only aim to get a result that looks plausible to a human, while we strive for *accuracy in comparison to the ground truth*. In comparison to works which use heuristics, we make use of *training data*. Such supervised approaches can be further classed according to whether the data used come from within the same image (e.g. symmetry [23]) or from a *database of training data*.

We now outline some of these previous approaches in more detail, and compare them to our approach.

Fitting 3D models If prior knowledge is available about the objects present in the scene, in the form of 3D models, then an instance-level model can be fitted to a 3D scene. This gives a good recovery of missing geometry of that object [17, 10]. Some works focus on the broader problem where an exact match is not present in the training set. Shen et al. [32] complete the missing regions of a single object using an assembly of parts from several different models in a CAD database. Cocias et al. [5] directly deform a mesh to fit to a point cloud, while Prisacariu and Reid [28] fit a class-level manifold model to the image data.

All these methods, however, rely on the availability of some form of specific prior model, and this completion method relies on an accurate detector to find and classify each object in the scene. In our approach, we set out to get as much shape information as possible *without semantics*, thus remaining free of its associated machinery and limitations.

Symmetry Where one can accurately detect symmetry, it can be leveraged to complete some types of objects (e.g. [24, 37, 23]). However, using symmetry for object completion is brittle. If symmetry cannot be detected at all, then no prediction can be made. This is the case for example when viewing a cuboid end-on.

Voxel space reasoning The two algorithms which bear the most similarity to ours both make predictions of full scene geometry from a single depth image.

Kim et al. [20] use a CRF model over a voxel representation of a scene to simultaneously predict occupancy, visibility and semantic labeling of voxels from an RGBD image. For training, they use manually labeled top-down views of the scene. Their final labelings are found from graph cuts over the CRF. They primarily model the probability of a voxel being occupied as a Gaussian centered on the first observed voxel along a camera ray. However, high-order-terms in the CRF are used to enforce planar structures and for 'objects' to remain contiguous.

Like [20], Zheng et al. [39] go from a single depth image to a voxel representation of a scene. Their core algorithm consists of two parts. Firstly they complete missing voxels by extruding visible points in the detected Manhattan World directions of the scene. This part bears resemblance to a similar method used for completions by [23]. Secondly, they use physics-based reasoning to fill in missing data to ensure connectedness and stability. The physics-based reasoning is clearly useful in complex scenes where, for example, a table leg may be occluded. However the voxel completion by extrusion is limited by the Manhattan World assumption, and the extent of visible voxels in the scene.

In contrast to these methods, we are able to make *structured* predictions in 3D space, thus being able to make predictions about shape learned directly from training data.

Surface completion Silberman et al. [35] tackle the completion of an incomplete multi-view reconstruction as a surface completion problem. By detecting planes, they can complete their contours in a 2D projection using a novel CRF method. This method, however, relies both on a piecewise-planar scene, and on beginning with an almost-complete scan as input.

Davis et al. [7] complete surfaces by operating directly on the *signed distance field*, the zero level-set of which defines the surface location. They diffuse the signed distance field across holes in the mesh to fill in the gaps. [14] use a data-driven approach, finding matches in the mesh to the missing region.

All of these completion methods are only suitable where the set of missing data is small relative to the size of the observed data. In our case, we are completing the geometry of an unknown surface which is typically at least as large as

the observed surface.

Image completion and super-resolution Image completion works such as [15, 6] typically use region-based data-driven approaches as it is very difficult to form true generative models over image appearance. For example, [15] look up possible completion regions in a large database of similar images, while [6] in-paint by selecting and combining multiple plausible patches from other regions of the input image. This differs from our task, as image completion typically aims for a visually plausible output, irrespective of the accuracy compared to ground truth. Similar to image *completion* is image *super-resolution*, e.g. [26]. This, like our problem, is ill-posed and relies on the repeatability and predictability of the world in order to improve the local geometry of an image. Our problem differs from that of super-resolution because we make predictions about the state of the world outside of the region directly observable by the camera.

Using 3D primitives for recognition 'Geons' are proposed by [1] as a set of 3D primitives such as cylinders and cuboids used by humans in their recognition of object shapes. While in theory, geons could be used by computers as features to describe natural objects, in practice, this was found to be challenging [8] due to their "idealized nature", requirement for part segmentation, labeling errors, and the coarseness of features used to extract geons in the first place.

However, fitting bounding boxes has recently become a popular method to explain the arrangement of objects in a scene. Recent work has successfully incorporated high-level information such as gravity and stability [31, 19], and made use of training data to accurately detect bounding box locations [16]. Gupta et al. [12] estimate voxel occupancy from a 2D image, which is regularized using cuboid bounding box hypotheses. The obvious problem with bounding box style methods is that they can only give coarse shape information, which is not suitable for many applications of geometry completion.

In our work, we make use of 3D primitives. However, unlike geons which are fixed in shape, we learn a distribution of shapes from training data. We are also able to make more fine-grained predictions than bounding boxes.

Structured learning for vision *Structured learning* has become a popular method for some computer vision tasks. As with standard supervised learning, structured learning finds a mapping from a feature space to a label space. In contrast to a traditional one-dimensional label space, however, a structured label space is *multidimensional*. Feature space is typically limited to a local descriptor of the image, while the label space may be surface normals [11], human poses [2], or semantic labels. This family of works provides inspiration for our approach.

There are many different ways of finding the mapping from feature to structured label space. For example, [2] cluster human poses, while [11] use an SVM-like formulation to find primitives which are both discriminative in feature space and informative in label space.

In our work, we make use of Random Forests [3]. Originally proposed for regression and then single-label classification problems, they have since been adapted to make structured predictions for tasks such as semantic labelling [22] and edge detection [9]. Structured prediction can make faster and more regularized predictions than a single dimensional predictor. In our case, we benefit from a structured prediction as we are able to make predictions beyond the location of the input data point.

3. Approach formulation

We model the geometry of a scene as a regular grid of voxels $\mathcal{V} = \{v_i\}$. Following works such as [18, 28], each $v_i \in [-d_{\max}, d_{\max}]$ represents the *truncated signed distance function* (TSDF) of the surface of the scene. Each $|v_i|$ gives the distance from v_i to the nearest surface, truncated to a maximum value of $d_{\max}$, which is a parameter. v_i is negative if voxel i is inside solid opaque matter, and positive if it is in free space. The zero level-set of $\mathcal{V}$ therefore represents the surface.

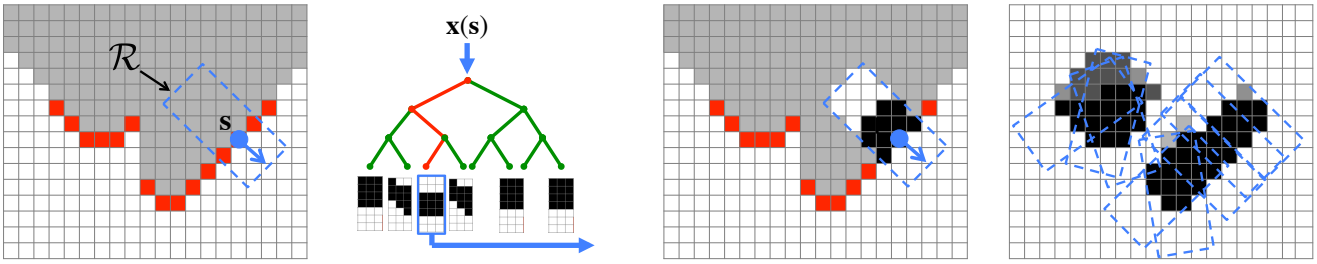
Our system maps a point $\mathbf{s}$ on an input depth image $\mathcal{D}$ to a prediction of the TSDF in a set of voxels in the neighborhood of $\mathbf{s}_p$, which we use to denote the 3D reprojection of $\mathbf{s}$. The aggregation of multiple such predictions gives our final TSDF prediction for the scene.

Support regions The *support region* $\mathcal{R} \subset \mathcal{V}$ is a set of voxels in the neighborhood of $\mathbf{s}_p$ for which our model can make a prediction of the TSDF. Each $\mathcal{R}$ is a fixed-size cuboid of voxels, whose x-axis aligned with the normal direction at $\mathbf{s}$ (Figure 2(a)).

In a 2D world, the location of $\mathbf{s}$ and the direction of its normal can unambiguously define the location and orientation of $\mathcal{R}$. However, in 3D there is a degree of freedom unconstrained as the rotation of the cuboid about the axis of the normal is unspecified. We resolve this by aligning the cuboid such that its z direction is coincident with the world z-axis, i.e. the 'up' direction of the scene. The top and bottom face of each cuboid region $\mathcal{R}$ is therefore parallel with the world's ground plane.

Making a single prediction At test time, we map a pixel $\mathbf{s}$ to a feature representation $\mathbf{x}(\mathbf{s})$, as described in Section 4. Using a structured Random Forest, we can make a prediction of the geometry inside of $\mathcal{R}$. We call this prediction of geometry a *voxlet*. The voxlet, which comes out of the forest in canonical alignment, is then transformed from its local coordinate system into world space to fill the voxels in $\mathcal{R}$.

The accumulation of multiple such predictions forms our



(a) Query point (b) Forest prediction (c) Insertion into voxel grid (d) Multiple predictions

Figure 2. **Overview of our algorithm, shown diagrammatically in a 2D representation.** (a) At test time, we define a cuboid region of voxels $\mathcal{R}$ around a query point $\mathbf{s}$. This cuboid is aligned with the normal at $\mathbf{s}$. (b) A feature representation $\mathbf{x}(\mathbf{s})$ is put through our pre-trained Random Forest. The forest makes a prediction for the values of each of the voxels in $\mathcal{R}$. (c) This prediction is transformed into the scene and used to update the values of the voxels. (d) The aggregation of multiple such predictions forms our final prediction of the TSDF, which can be converted to voxel occupancy or a surface representation.

final prediction of our TSDF. This can then be converted to a prediction of surface geometry, described in Section 5.2.

Training At training time, we similarly define a region $\mathcal{R}$ around each point $\mathbf{s}$, again of a fixed size. In this case, the ground truth values of $\mathcal{V}$ and hence $\mathcal{R}$ are known. We use these ground truth TSDF values, together with $\mathbf{x}(\mathbf{s})$, to train a Random Forest, as explained in Section 5.

4. Feature representation

Our feature representation $\mathbf{x}(\mathbf{s})$ at location $\mathbf{s}$ in the input depth image $\mathcal{D}$ describes aspects of its neighborhood. We use local and regional features extracted from the depth image. The Random Forest–based model weights feature space to make structured predictions of the truncated signed distance function in the surrounding region.

$\mathbf{x}(\mathbf{s})$ is composed of two parts. The *offset* feature captures the shape of the depth surface in the immediate neighborhood of $\mathbf{s}$, and is made up of several local depth comparisons. The *spider* feature captures the size and shape of the region in the 2D image in which $\mathbf{s}$ resides.

4.1. Offset feature

The 'offset feature' x_{off} is a simple pairwise feature, used with success in recent work such as [33], capturing the surface shape in the immediate neighborhood of pixel $\mathbf{s}$. The feature computes the difference between the depth from the camera at $\mathbf{s}$ and the depth at a predetermined offset Δ (Figure 3). We define

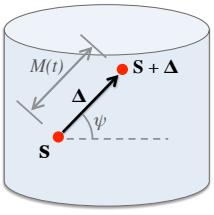
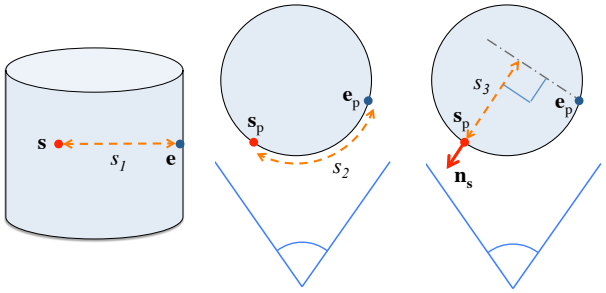


Figure 3.

$$x_{\text{off}}(\mathbf{s},\psi,t) = \mathcal{D}(\mathbf{s}) - \mathcal{D}(\mathbf{s}+\Delta), \text{ where} \quad (1)$$

$$\Delta = M(t)(\sin(\psi),\cos(\psi)). \quad (2)$$



(a) s_1 (b) s_2 (Top view) (c) s_3 (Top view)

Figure 4. The spider features $x_s(\mathbf{s},\mathbf{e},\psi) = (s_1, s_2, s_3)$ provides three measures of the distance between $\mathbf{s}$ and the edge point $\mathbf{e}$. (a) s_1 measures the distance between pixel locations $\mathbf{s}$ and $\mathbf{e}$. (b) s_2 measures the distance between 3D points $\mathbf{s}_p$ and $\mathbf{e}_p$ along the surface of $\mathcal{D}$. (c) s_3 gives an estimate of the depth of the object at the 3D location $\mathbf{s}_p$, using the surface normal $\mathbf{n_s}$.

Here $M(t) = (t \cdot f)/\mathcal{D}(\mathbf{s})$ uses the focal length f to map a distance in world space to a pixel offset.

For a single point $\mathbf{s}$ we compute $x_{\text{off}}(\mathbf{s},\psi,t)$ for a range of different values of ψ and t, and concatenate all the values into a feature vector $\mathbf{x}_{\text{off}}$. For our experiments we use $\psi,\ t \in \{0°, 45°, \ldots, 315°\} \times \{0.02m, 0.04m, 0.06m, 0.08m\}$. $\mathbf{x}_{\text{off}}$ is therefore 24-dimensional.

4.2. The spider feature

Local features, such as the offset feature, have proved discriminatory where the output labeling is on a per-pixel basis, such as labeling each pixel as a part of a human body [33]. However, when it comes to predicting the geometry

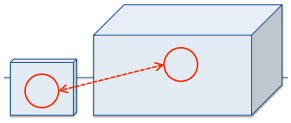


Figure 5.

of the scene we cannot observe, the local shape of the depth image around a point does not provide enough information. An example of this can be seen in Figure 5. The two regions circled in the image have identical *local* appearance, but the 3D geometry around each region is very different due to the different thicknesses of the two boxes. We use this observation to motivate the 'spider feature'.

We desire a feature which captures the size and shape of the region in which $\mathbf{s}$ resides, along with the location of $\mathbf{s}$ in that region. To achieve this we take inspiration from the features between a point and a binary edge map used by [10].

Computing a binary edge map We first compute a binary edge map for the image, where each pixel takes the value 1 at a large change in depth or surface normal direction, and is 0 otherwise. We first use the method of [9] to compute a real-valued edge map for the image. Next we apply the Canny filtering algorithm [4] to convert this to a binary edge map. We mitigate against poor quality depth data around discontinuities by dilating this edge map with a 3x3 disc-shaped structuring element.

Casting lines from $\mathbf{s}$ From $\mathbf{s}$, we then cast a line across the edge map at angle ψ, where ψ is a parameter of the feature computation. We denote the point on the edge map where the line first hits a pixel with value 1 as $\mathbf{e}$. We additionally denote the 3D reprojections of the 2D pixel coordinates $\mathbf{s}$ and $\mathbf{e}$ as $\mathbf{s}_p$ and $\mathbf{e}_p$ respectively.

Extracting the spider feature The spider feature for a single point with a line cast in direction ψ is $x_s(\mathbf{s}, \mathbf{e}, \psi) = (s_1, s_2, s_3)$. Each element of this tuple is a distance between $\mathbf{s}$ and $\mathbf{e}$, computed as follows:

- $s_1 = \mathcal{D}(\mathbf{s})\|\mathbf{s} - \mathbf{e}\|_2/f$. This is the distance between $\mathbf{s}$ and $\mathbf{e}$ in pixel units $\|\mathbf{s} - \mathbf{e}\|_2$ scaled by $\mathcal{D}(\mathbf{s})/f$ to convert to a real-world distance (Figure 4(a)).

- $s_2 = \sum_{i=2}^{|\mathbf{l}|} \|\mathbf{l}(i) - \mathbf{l}(i-1))\|_2$, where $\mathbf{l}$ is an ordered list of the 3D locations of the points on the line between $\mathbf{s}$ and $\mathbf{e}$. s_2 approximates the arc length between $\mathbf{s}$ and $\mathbf{e}$ along the surface of $\mathcal{D}$ (Figure 4(b)).

- $s_3 = (\mathbf{s}_p - \mathbf{e}_p) \cdot \mathbf{n}_\mathbf{s}$. This is the Euclidean distance between 3D points $\mathbf{s}_p$ and $\mathbf{e}_p$ along the direction of the normal $\mathbf{n}_\mathbf{s}$ at $\mathbf{s}$ (Figure 4(c)).

We compute $x_s(\mathbf{s}, \mathbf{e}, \psi)$ for $\psi \in \{0°, 45°, \ldots, 315°\}$, and concatenate the results into a 24-dimensional vector $\mathbf{x}_{\text{spi}}$. The concatenation of $\mathbf{x}_{\text{off}}$ and $\mathbf{x}_{\text{spi}}$ forms the final 56D feature vector $\mathbf{x}(\mathbf{s})$ for point $\mathbf{s}$.

Implementation The spider feature is efficient to compute; we can compute the 24D spider feature for every point in a 640×480 image in less than 0.01s using our C++ implementation.

5. Learning a mapping from features to voxlets

In this paper we pose unobserved geometry estimation given partial observed information as a supervised learning problem. More specifically, our goal is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$, which maps each observed feature vector $\mathbf{x} \in \mathcal{X}$ to the label space $\mathbf{v} \in \mathcal{Y}$ representing the corresponding 3D geometry in the region $\mathcal{R}$ around $\mathbf{x}$. Unlike standard classification where the goal is to predict a category label for each $\mathbf{x}$, our label space is a multi-dimensional structured vector $\mathbf{v} \in \mathbb{R}^{w \times d \times h}$ which encodes the TDSF values in a local region. Inspired by the recent work of Dollár and Zitnick [9] we use a structured Random Forest to learn the function f.

Training To train the forest we pass a bagged subset of the training set $\{(\mathbf{x}_1, \mathbf{v}_1), ..., (\mathbf{x}_n, \mathbf{v}_n)\}$ to each node in the tree starting at the root. Each node is then tasked with splitting the examples so that the ones sent to its children are as similar as possible in label space. Instead of minimizing the structured loss directly, Dollár and Zitnick [9] approximate this loss at each node using a classification loss. To convert the structured problem into a classification one we sample a different random subset of the dimensions of each $\mathbf{v}_i$ at the node, reduce their dimensionality to N dimensions, and then cluster them into a discrete number of classes K (here we set $N = 20$ and $K = 2$). Then a standard classification loss can be used on this new discretization to evaluate the quality of different candidate splits for each $\mathbf{x}_i$. In practice, this dimensionality reduction and clustering can be efficiently performed using randomized PCA [13]. To cluster, a training example is assigned to one of the two possible clusters K based on the sign of the values in its first principal component.

This process is repeated until we cannot split the data any further. Finally, each leaf node stores the medoid of all the examples that have arrived there, which we refer to as a voxlet, see Figure 6.

Testing Evaluating each tree at test time is extremely efficient. For each location in the input depth image we simply traverse the tree until we reach a leaf node and return the voxlet stored at that location, see Figure 2(b).

Implementation Given the large dimensionality of $\mathcal{Y}$ we perform an initial dimensionality reduction to a 50 dimensional space using PCA. Due to the large amount of redundancy in each $\mathbf{v}$ we found this to have little impact on the quality of our results, and yet it provides a large improvement to computational tractability. We use an ensemble of 50 trees, which are grown to a maximum depth of 15 or until there is a minimum of 2 examples at a node. We use simple axis aligned feature splits at each node.

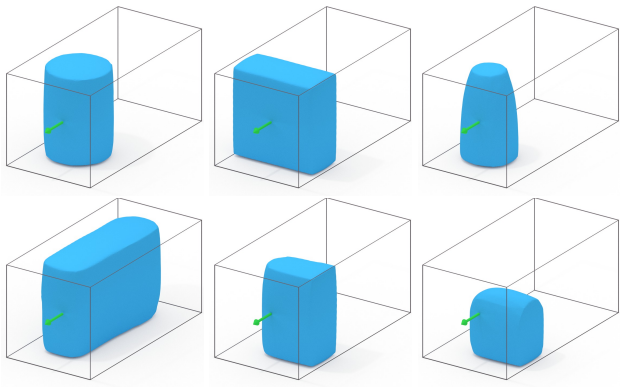


Figure 6. Representative voxlets from the dataset. Here we show cluster centroids after running K-Means on a collection of 20,000 voxlets, with K=200. The green arrow represents the vector which is aligned with $\mathbf{n_s}$ (see Figure 2(c)). Each voxlet can be seen to capture a section of the geometry of an object.

5.1. The voxlet dimensions

The fixed-size regions $\mathcal{R}$ in the voxel grid are set in our experiments to have dimensions of $15 \times 30 \times 15$ voxels. We make the voxlets longer in the y-direction as it is this direction that is approximately parallel to the normal at $\mathbf{s}_p$. This allows the voxlet to make a larger prediction *backwards* into the scene than *sideways* into a region which typically already has observed data. We set each voxlet to have real-world dimensions $0.1m \times 0.2m \times 0.1m$, meaning each voxel inside a voxlet has edge lengths of $(0.1/15)$m. We define a local coordinate system for the voxlet, as shown in red in Figure 7. The position in the voxlet which we align with the 3D point $\mathbf{s}_p$, as shown in Figure 2(c), is the point $x_v = y_v = z_v = 0.05$m.

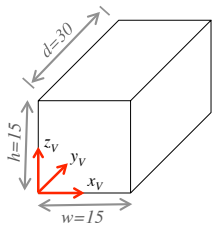


Figure 7.

5.2. Aggregating predictions

For each voxel in the output grid $\mathcal{V}$, we accumulate the predictions of the multiple overlapping TSDF predictions at that location. A number of different strategies could be employed to recover our final values, e.g. filtering techniques, loopy belief propagation *etc*. In our work we simply take the mean of all the predictions at that voxel. We note that the truncation of the signed distance function helps to make this style of accumulation robust. A single incorrect estimation at a voxel can only be wrong by a maximum amount of $2d_{\max}$, where $d_{\max}$ is the level at which the distance function is truncated.

Some voxels in $\mathcal{V}$ do not have any predictions made for them at all. We assign to these voxels the value $d_{\max}$.

The final TSDF estimation can be converted to a mesh by finding the level-set of zero, for example using marching cubes [25].

6. Experiments

We evaluate our algorithm from the perspective of its ability to reconstruct objects in isolation, and then further its ability to reconstruct more cluttered scenes. All of the experiments are performed on data captured from the Microsoft Kinect.

6.1. Experimental procedure

In our experiments we first smooth the noisy depth data using cross-bilateral filtering, using the implementation provided by [34]. We then detect the largest approximately horizontal plane in the image, and use this to estimate the 'up' direction of the scene. We make an estimate of geometry using the 3D points which lie above this detected planar surface.

We can choose at test time how many pixels to run through the forest to make a prediction of voxel occupancy. In our experiments we use 200 pixels. The forest predictions for these pixels are then aggregated as described in Section 5.2.

We use two different datasets in our experiments. Our first, quantitative experiment tests the reconstruction of objects in isolation. Secondly, we perform qualitative evaluation of our algorithm on cluttered tabletop scenes.

Objects in isolation For our experiments on objects in isolation, we make use of the 'Bigbird' dataset [36]. This dataset consists of 125 household objects, captured on a turntable from multiple viewing angles. The camera poses are well calibrated, and a complete object mesh is provided for each object. This mesh can be converted to provide us with a ground truth voxel occupancy for quantitative evaluation.

In our train/test split of this dataset, we note that there are many 'near duplicate' items in the dataset. For example, two different varieties of the same product may be included. To ensure a fair split we group such items together to ensure that each group ends up as a whole in either the training or the test section. Our final training set consists of 80 objects, and our test set 45 objects. We use 45 views of each object.

Objects in clutter We perform qualitative evaluations on images from the Object Segmentation Dataset [29]. This dataset consists of household objects arranged in various arrangements on a table surface. For our experiments on the Object Segmentation Dataset, we use the model learned from the Bigbird turntable data.

6.2. Quantitative evaluation

Evaluation criteria The aim of our algorithm is to accurately classify free space around objects and in scenes.

Method	Precision	Recall
Bounding box	0.550	0.683
Zheng et al. [39]	0.570	0.466
K-means forest	0.922	0.944
Structured forest (offset only)	0.921	0.948
Structured forest (spider only)	0.855	0.949
Structured forest (offset + spider)	0.926	0.948

Table 1. A comparison of the accuracy of geometry estimation algorithms on the Bigbird dataset. Details of the algorithms used are given in Section 6.2.

Therefore, we report the per-voxel precision and recall over all the test data. For our algorithms, we use the sign of the accumulated TSDF to form our final binary prediction of occupancy.

Baselines and algorithm variants We compare to two main baseline algorithms: a naive bounding box approach and the method of Zheng et al. [39]. The details of these baseline implementations, and of the details of variants of our method, are:

(a) Bounding box We fit a minimum-area bounding box to the 3D points belonging to the object, as defined by the ground truth object segmentation mask provided by [36]. We are careful to remove 'flying pixels', as they can have a large adverse effect on bounding box predictions. Finally, the prediction of voxel occupancy is simply all voxels inside the bounding box are predicted to be occupied, while those outside are predicted to be empty.

(b) Zheng et al. [39] We implemented the algorithm of Zheng et al. [39] for reconstructing voxel occupancy as described in Section 2 of their paper. We find the Manhattan axes of the scene from the minimum area bounding box aligned with the ground truth voxel grid. We then perform their axis-aligned voxel search for each unobserved voxel, marking voxels as 'filled' where more than two Manhattan directions hit a voxel directly observed by the camera.

(c) K-means forest We train a standard classification Random Forest, the leaf nodes of which vote for items in a dictionary of 200 voxlets formed by running K-means on the training set of voxlets. The transformation into the scene and the aggregation steps are equivalent to our core algorithm.

(d) Structured (offset only) Our full structured forest, but only using the offset features

(e) Structured (spider only) Our full structured forest, but only using the spider features

(f) Structured (offset + spider) Our full structured forest using both offset and spider features.

The results from our quantitative analysis on the Bigbird turntable dataset are shown in Table 1. We present some views of individual results in Figure 8. We notice that the naive bounding box prediction suffers from a poor rate of recall. This is expected, as typically the bounding box predictions under-predict the volume due to a lack of observed pixels to complete the shape of the object (8). Our results successfully capture the overall geometry of each of the objects. The main, general point of failure is a smoothing effect which affects the edges of the objects.

While our full forest outperforms the other methods, we notice only a small change in the variants of our approach using different features. The lower precision and higher recall for the spider feature suggests that it is over-predicting the size of objects, although the performance is maximized when it is coupled with the offset feature. The K-means Forest does almost as well as our full structured forest. However, as it has discretized the label space before training we would not expect it to generalize well to larger and more diverse training data.

6.3. Qualitative scene results

In our qualitative analysis, we run the full structured prediction model from the previous section on views from the object segmentation dataset. Qualitative results from this evaluation are shown in Figure 9. We note that the geometry of larger objects is recovered well, while the geometry of smaller regions can be missed. Where objects are occluded and cluttered, our algorithm can fail to recover the most occluded sections. See for example the result in row three, column three.

6.4. Limitations

We can see that while we have success reconstructing well-observed objects, our algorithm can fail to recover geometry in occluded regions. We propose that a physics-based-reasoning post-processing step, such as is proposed by [39, 31], would help the completion of ambiguous regions.

7. Conclusions and future work

In this paper we have presented an algorithm to successfully recover 3D geometry given just a single depth image. Key to this approach is the voxlet, a set of voxels which can be learned from training data and then used at test time to recover shape.

The primary direction for our future work is to apply the algorithm to larger and more natural scenes. In particular, we believe that training on real-world scenes in addition to turntable data will yield a large improvement to the results.

An interesting potential application of our method is to use the predicted completion as a prior for structure-from-motion. Such a system would up its prediction of occupancy with observed data as it arrives from the capture device. Additionally, using our prediction as a prior could form the basis of a *next-best-view* algorithm [27].

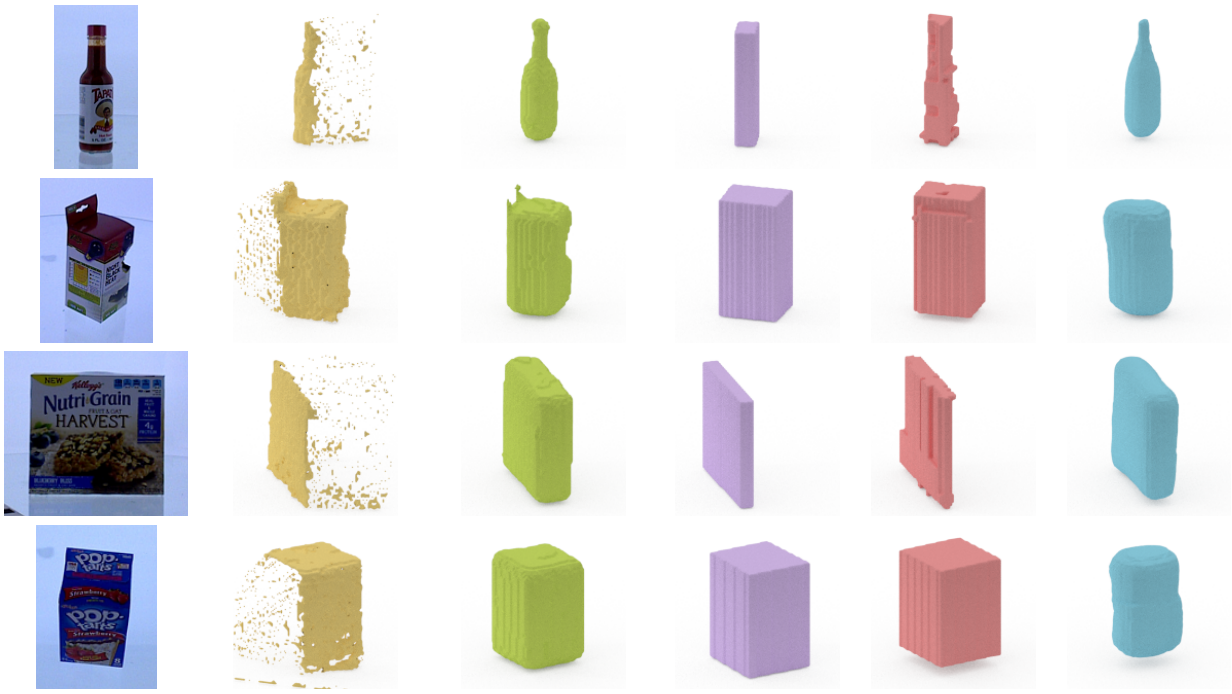


Figure 8. Results from the Bigbird turntable dataset. Each row shows a different object, while each column shows the result of a different algorithm. The first column shows the RGB image of the input view, while the second column shows the reprojected input depth pixels from a side view. The following columns, all shown from the same viewpoint as the input depth pixels view, demonstrate the different baselines and algorithm variants. More views and objects can be seen in the supplementary material.

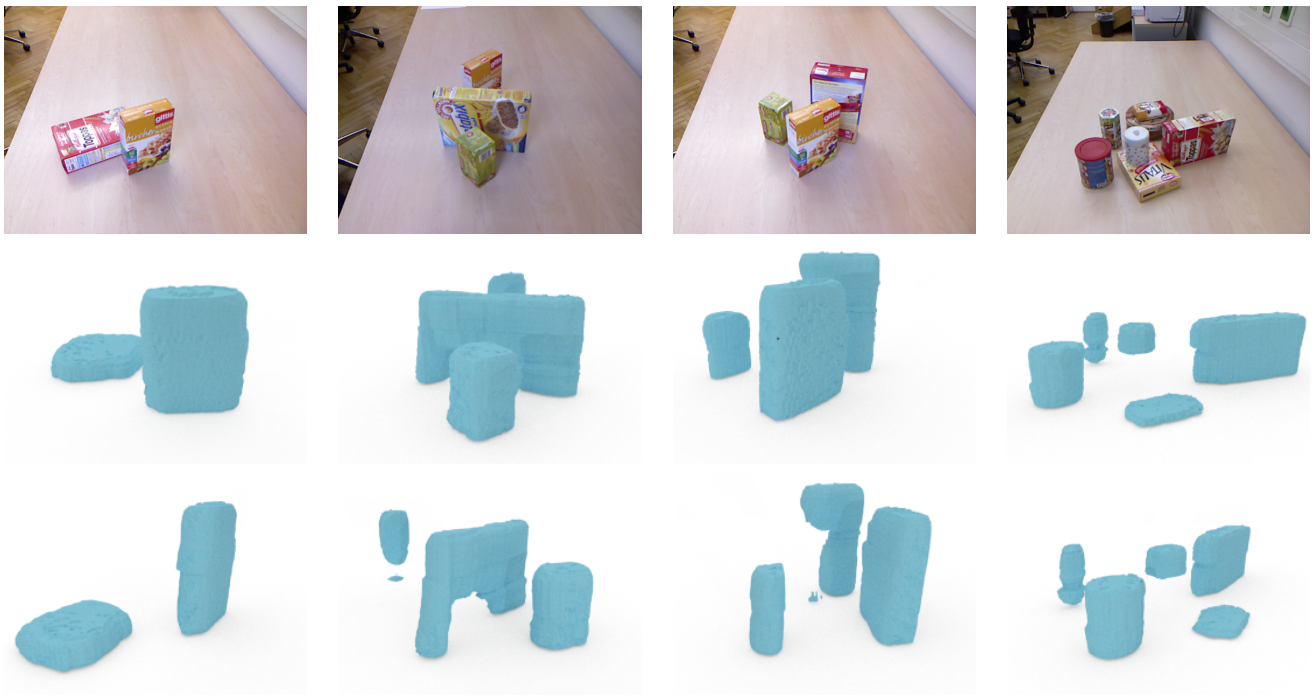


Figure 9. Qualitative results from the Object Segmentation Dataset. Each column shows the result of our agorithm on a different image from the database. We note that our training from the turntable dataset successfully recovers the shape of many objects. It is in areas of occlusions, heavy clutter and specular objects that the algorithm misses areas (e.g. column 4).

References

[1] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 1987. 3

[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *International Conference on Computer Vision (ICCV)*, 2009. 3

[3] L. Breiman. Random Forests. *Machine learning*, 45(1), 2001. 3

[4] J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence (PAMI)*, 1986. 5

[5] T. T. Cocias, F. Moldoveanu, and S. M. Grigorescu. Generic fitted primitives (GFP): Towards full object volumetric reconstruction for service robotics. In *Computer Graphics, Visualization and Computer Vision*, 2013. 2

[6] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition (CVPR)*, 2003. 3

[7] J. Davis, S. Marschner, M. Garr, and M. Levoy. Filing holes in complex surfaces using volumetric diffusion. In *3DPVT*, 2009. 2

[8] S. J. Dickinson, R. Bergevin, I. Biederman, J.-O. Eklundh, R. Munck-Fairwood, A. K. Jain, and A. Pentland. Panel report: The potential of geons for generic 3-D recognition. *Image and Vision Computing*, 1997. 3

[9] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *International Conference on Computer Vision (ICCV)*, 2013. 3, 5

[10] B. Drost and S. Ilic. 3D object detection and localization using multimodal point pair features. In *3DIMPVT*, 2012. 2, 5

[11] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *International Conference on Computer Vision (ICCV)*, 2013. 3

[12] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3D scene geometry to human workspace. In *Computer Vision and Pattern Recognition (CVPR)*, 2011. 2, 3

[13] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 2011. 5

[14] G. Harary and A. Tal. Context-based coherent surface completion. *TOG*, 2013. 2

[15] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics*, 2007. 3

[16] V. Hedau, D. Hoiem, and D. Forsyth. Recovering free space of indoor scenes from a single image. In *Computer Vision and Pattern Recognition (CVPR)*, 2012. 3

[17] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, , and N. Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conference on Computer Vision (ACCV)*, 2012. 2

[18] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *ACM symposium on User interface software and technology*, 2011. 1, 3

[19] Z. Jia, A. Gallagher, A. Saxena, and T. Chen. 3D-based reasoning with blocks, support, and stability. In *Computer Vision and Pattern Recognition (CVPR)*, 2013. 3

[20] B.-s. Kim, P. Kohli, and S. Savarese. 3D scene understanding by voxel-CRF. In *International Conference on Computer Vision (ICCV)*, 2013. 2

[21] J. Kim and K. Grauman. Shape sharing for object segmentation. In *European Conference on Computer Vision (ECCV)*, 2012. 1

[22] P. Kontschieder, S. R. Bulo, H. Bischof, and M. Pelillo. Structured class-labels in random forests for semantic image labelling. In *International Conference on Computer Vision (ICCV)*, 2011. 3

[23] O. Kroemer, H. B. Amor, M. Ewerton, and J. Peters. Point cloud completion using extrusions. In *International Conference on Humanoid Robots (HUMANOIDS)*, 2012. 2

[24] A. J. Law and D. G. Aliaga. Single viewpoint model completion of symmetric objects for digital inspection. *Computer Vision and Image Understanding*, 2011. 2

[25] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm, 1987. 6

[26] O. Mac Aodha, N. D. Campbell, A. Nair, and G. J. Brostow. Patch based synthesis for single depth image super-resolution. In *European Conference on Computer Vision (ECCV)*, 2012. 3

[27] C. Potthast and G. S. Sukhatme. A probabilistic framework for next best view estimation in a cluttered environment. *Journal of Visual Communication and Image Representation*, 2014. 7

[28] V. A. Prisacariu and I. Reid. Shared shape spaces. In *International Conference on Computer Vision (ICCV)*, 2011. 2, 3

[29] A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *Intelligent Robots and Systems (IROS)*, 2012. 6

[30] R. Schnabel, P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. In *Computer Graphics Forum*, 2009. 2

[31] T. Shao, A. Monszpart, Y. Zheng, B. Koo, W. Xu, K. Zhou, and N. J. Mitra. Imagining the unseen: Stability-based cuboid arrangement for scene understanding. *ACM Transactions on Graphics*, 2014. 3, 7

[32] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu. Structure recovery by part assembly. *ACM Transactions on Graphics*, 2012. 2

[33] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR)*, 2011. 4

[34] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *European Conference on Computer Vision (ECCV)*, 2012. 6

[35] N. Silberman, L. Shapira, R. Gal, and P. Kohli. A contour completion model for augmenting surface reconstructions.

In *European Conference on Computer Vision (ECCV)*, 2014. 2

[36] A. Singh, J. Sha, K. Narayan, T. Achim, and P. Abbeel. A large-scale 3D database of object instances. In *International Conference on Robotics and Automation (ICRA)*, 2014. 6, 7

[37] S. Thrun and B. Wegbreit. Shape from symmetry. In *International Conference on Computer Vision (ICCV)*, 2005. 2

[38] A. Velten, T. Willwacher, O. Gupta, A. Veeraraghavan, M. G. Bawendi, and R. Raskar. Recovering three-dimensional shape around a corner using ultra-fast time-of-flight imaging. *Nature Communications*, 2012. 2

[39] B. Zheng, Y. Z. andJoey C. Yu, K. Ikeuchi, and S.-C. Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *Computer Vision and Pattern Recognition (CVPR)*, 2013. 2, 7