Object Detection in RGB-D Indoor Scenes

Edmund Shanming Ye
Jitendra Malik, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-3
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-3.html

January 14, 2013

Acknowledgement

I'd like to thank my advisor, Jitendra Malik, for being an invaluable source of wisdom and direction in my thesis project, and for supporting my time in the vision lab. I'd also like to thank Ross Girshick and Saurabh Gupta for their immeasurable help. Thanks also to Pulkit Agrawal, Bharath Hariharan, Pablo Arbelaez, Jon Barron, and Georgia Gkioxari for their substantial guidance along the way.

Finally, and perhaps most importantly, I'd like to thank my family and friends. Y'all made it worthwhile.

Object Detection in RGB-D Indoor Scenes

1 Introduction

With the arrival of the Microsoft Kinect, obtaining depth maps of interior spaces has become remarkably easy. The Kinect is equipped with an 8-bit RGB VGA resolution (640x480 pixel) video camera, and also features an IR-triangulation based depth sensor with reports of accuracy within $q(z) = 2.73z^2 + 0.74z - 0.58[mm]$, with z the depth in meters [11]. The Kinect's low cost and portability make it an attractive instrument for robotics and mapping. We have witnessed a boon of large datasets originating from such Kinect-style cameras, and an associated development in algorithms for SLAM-like tasks. While the potential for this data is vast, one immediate application is incorporating the depth data into a more robust object detector.

Object detection is a well-studied problem in computer vision. One of the basic tasks, as framed by the Pascal Visual Object Classes Challenge [6], is to draw tight bounding boxes around instances of various target classes in a set of images. The accuracy of each proposed bounding box is evaluated by some function of its overlap and non-overlap with ground truth. Computer vision literature has primarily focused on intensity, with less emphasis on depth data. In this report we address the challenge of detecting 10 common household items (bed, chair, etc) in RGB-D images obtained using the Kinect. We operate on the recently released NYU-Depth V2 (NYUD2) dataset [14], which contains 1449 images from several different indoor settings. Our algorithm augments the deformable parts model by adding a set of vector quantized depth features that are, to the best of our knowledge, novel on this dataset.

The remainder of this report is organized as follows: Part 2 describes previous work done in the area of object recognition and depth data. Part 3 describes the dataset. Part 4 gives our approach to the problem. Part 5 gives results of our method. Part 6 provides closing remarks.

2 Related Work

A wide variety of techniques for object recognition have proliferated in recent years. While even state-of-the-art performance is far removed from the human eye, much progress has been made in the area of feature descriptors and incorporating semantically meaningful information into the detection pipeline. Many

methods employ the use of Amazon Mechanical Turk (AMT) to harness the power of human annotations as input to supervised learning algorithms. We provide an abbreviated overview of seminal works that use the "sliding window" approach.

2.1 Histogram of Oriented Gradients (HOG)

HOG presented a novel feature descriptor on image patches. As a first step, it computes discretized gradients by convolving the image patch with a filter. Two 1-D tap filters, consisting of $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}$ and $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}^T$, were empirically found to perform best. Next the image patch is segmented into a dense grid of uniformly spaced cells. Within each cell, a histogram of gradients is computed. Each pixel casts a vote weighted by the strength of its gradient and distance to the center of the cell, and each vote is cast towards a certain gradient orientation range corresponding to a bin in the histogram. Finally, each histogram is contrast normalized over spatial neighbors [5].

2.2 Non-Rigid Detectors

Some models improve on HOG by better capturing intra-class variation. These models break down an object into its constituent parts. We present two independently developed methods that have been successful in Pascal.

2.2.1 Poselets

The poselet detection pipeline attempts to better localize the configuration of objects in an image patch. The intuition is that semantic keypoints within an object accurately capture its pose. For each object class, a set of meaningful keypoints is selected by hand. For example, a "person" may be described by keypoints such as {"head", "shoulder", "feet", etc.}, and AMT workers are asked to identify these keypoint in a training set. Random seed image patches containing keypoints are used as cluster centers, and nearest neighbors in keypoint configuration space are identified. Each cluster, referred to as a "poselet", serves as positive examples for a support vector machine (SVM) trained to find that particular poselet. At test time, each poselet SVM provides a q-score activation, which is then converted to a Q-score taking into account neighboring poselet activations [3].

While this approach has proven successful in the past, scalability is a large concern. First, keypoints must be chosen for each class, some of which may not even possess enough structure to have clearly defined keypoints. In addition, AMT can be a tenuous tool. Workers often have questionable motives for performance, and quality control methods are often manual and tedious. Industrial-strength systems (Kinect) can skirt this problem by allocating dedicated workers to hand annotations, but in academic settings this is not always feasible.

2.2.2 Deformable Parts Model (DPM)

We give an outline of the DPM methodology, as it provides a baseline and framework for our work with depth features. DPM [7] augments rigid HOG templates and avoids the overhead of additional annotations. It trains a latent SVM scored by a function of the form:

$$f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z) \tag{1}$$

where β is a vector of model parameters, z are latent values, and $\phi(x, z)$ is a feature vector.

Model

The model learns a "root" filter, several higher-resolution internal "part" filters, and deformation costs associated with those part filters. The score of a $w \times h$ linear filter F at a position (x, y) in a feature map G is given by the "dot product" of the filter and a subwindow of the feature map with top-left corner at (x, y):

$$\sum_{x',y'} F(x', y') \times G(x + x', y + y') \tag{2}$$

Furthermore, let H be a feature pyramid and $p = (x, y, l)$ specify a position (x, y) in the l-th level of the pyramid. Let $\phi(H, p)$ denote the vector obtained by concatenating the feature vectors in the $w \times h$ subwindow of H with top-left corner at p in row-major order, and let F' be the vector obtained by concatenating the weight vectors in F in row-major order. In this context, (2) can be written as:

$$F' \cdot \phi(H, p)$$

The score of a model at a particular position and scale within an image is the score of the root filter at the given location plus the sum over parts of the maximum, over placements of that part, of the part filter score on its location minus a deformation cost measuring the deviation of the part from its ideal location relative to the root. Hence, a model with n parts consists of a $(n + 2)$-tuple $(F_0, P_1, \ldots, P_n, b)$ where F_0 is a root filter, P_i is a model for the i-th part and b is a real-valued bias term. Each P_i is defined by a 3-tuple (F_i, v_i, d_i) where F_i is a filter for the i-th part, v_i is a two-dimensional vector specifying an "anchor" position for part i relative to the root position, and d_i is a four-dimensional vector specifying coefficients of a quadratic function defining a deformation cost of the placement of the part relative to the anchor position. The score of a particular choice of filter placements $(p_0, \ldots, p_n)$ is given by:

$$score(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i^{'} \cdot \phi(H, p) - \sum_{i=1}^{n} d_i \cdot \phi_d(dx_i, dy_i) + b$$

where

$$(dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i)$$

gives the displacement of the i-th part relative to its anchor position and

$$\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)$$

are deformation features. The score of a hypothesis z can be expressed in terms of a dot product:

$$\beta \cdot \psi(H, z)$$

where

$$\beta = (F_0^{'}, \ldots, F_n^{'}, d_1, \ldots, d_n, b)$$
$$\psi(H, z) = (\phi(H, p_0), \ldots, \phi(H, p_n), -\phi_d(dx_1, dy_1), ..., -\phi_d(dx_n, dy_n), 1)$$

This naturally fits the latent SVM formulation (1).

Latent SVM

Consider a classifier that scores an example x with a function of the form (1). We train β from labeled examples $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $y_i \in \{-1, 1\}$ by minimizing the objective function:

$$L_D(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i f_\beta(x_i))$$

Let Z_p specify a latent value for each positive example in D. We define an auxiliary objective function $L_D(\beta, Z_p)$ by restricting the latent values for the positive examples. Note that:

$$L_D(\beta) = \min_{Z_p} L_D(\beta, Z_p)$$

In practice, $L_D(\beta, Z_p)$is minimized using a "coordinate descent" approach:

1. *Relabel positive examples*: Optimize $L_D(\beta, Z_p)$ over Z_p by selecting the highest scoring latent value for each positive example $z_i = \arg\max_{z \in Z(x_i)} L_D(\beta, Z_p)$
2. *Optimize beta*: Optimize $L_D(\beta, Z_p)$ over β by solving the convex optimization problem.

2.3 Depth

Literature involving depth data for object recognition is less extensive. We present an overview of some previous work.

Many RGB-D datasets include humans and are tied to the problems of human recognition and human pose estimation. These tasks are relevant to industrial applications, and feature descriptors are commonly tailored to the human

limb structure [21]. [22],[9] integrate multiple classifiers and binocular disparity features. Depth data is often noisy as triangulation methods are prone to quantization error, and [2] introduces a pose estimation algorithm that copes with this. In addition, it is not always clear how best to combine depth and RGB, and [18] suggests one way to do so.

The challenge of object detection in standard indoor environments is closely associated with robotics. As such, much emphasis is placed on developing speedy algorithms that may be executed in real time. Depth maps are often sparse and the objects small; [8] develops a multi-modal object detector to deal with this. [19] uses a viewpoint feature histogram for simultaneous recognition of object and its pose. [15] use independent reflectance and depth data from a single sensor to detect known objects. Some work has been done in using point clouds to recognize geometric primitives [16], or detecting large novel objects [4].

Indoor scene segmentation on RGB-D, a closely related task, has also recently become quite popular. [17] found success by using kernel descriptors, and by combining MRF with segmentation tree. [10],[12] reason about 3D geometry of rooms and objects. [1] use a graphical model and contextual information to semantically label 3D points clouds.

3 Dataset

NYUD2 consists of 1449 images taken from a variety of commercial and residential buildings in three different US cities across 26 scene classes. Per-pixel instance and class annotations were produced by AMT workers. The annotations are extremely specific, with 35,064 distinct objects spanning 894 classes. Due to time-synchronization errors between the Kinect RGB and depth video streams, we empirically discovered a contour disparity of up to 5 pixels on certain images in the dataset.

4 Our Approach

Our pipeline is as follows:

1. Raw depth maps are first transformed to surface normal maps by fitting planes to support patches.

2. Surface normals are clustered using a standard k-means on cosine distance metric.

3. Features are constructed by segmenting an image patch into 8pixel $\times$ 8pixel cells. Cell histograms are produced by vector quantizing surface normals to the nearest centroid produced during the k-means step, and $l1$-normalized.

4. These features are used in a variant of the deformable parts model.

4.1 Depth

Our pipeline for generating depth feature descriptors is somewhat similar to that of HOG. However, our intuition is that surface normals are a more natural way of capturing the curvature of objects, and hence a better descriptor, than depth gradients. We precompute surface normals from a raw depth map and align them with an estimated gravity vector. The aligned normals are clustered via k-means and spatially histogrammed.

4.1.1 Surface Normals

We note that a dense set of surface normals reveals important information about the shape of an object. In particular, they are a better descriptor for shape than depth gradients alone. To extract a surface normal at each pixel, we follow [13] by fitting a plane to a support patch of raw depths.

The equation of a plane is given by

$$\begin{aligned} disparity = \frac{1}{z} &= (\frac{x}{z})n_x + (\frac{y}{z})n_y + n_z \\ &= \boldsymbol{n}\phi \end{aligned}$$

where $\phi = \begin{bmatrix} \frac{x}{z} & \frac{y}{z} & 1 \end{bmatrix}^T$ and $\boldsymbol{n} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}$. For a given point cloud, we solve for $\boldsymbol{n}$ such that $l2$ loss in disparity is minimized.

Let S be a $m \times m$ support patch of raw depths with top left corner at (x_0, y_0) in the image. For each pixel in S, we first translate the patch to center at $(0, 0)$. That is, $\forall (x, y) \in S$ define $x' \triangleq x - x_0 - \frac{m}{2}$ and $y' \triangleq y - y_0 - \frac{m}{2}$.

We also create the matrices for our least-squares problem. Let z_{xy} be the depth at (x, y) and define

$$\begin{aligned} T_S &: (x, y) \mapsto (\frac{x'}{z_{xy}}, \frac{y'}{z_{xy}}, 1) \\ U_S &: (x, y) \mapsto \frac{1}{z_{xy}} \end{aligned}$$

Let $O = \{(x_1, y_1), \ldots, (x_{m^2}, y_{m^2})\}$ be any sequence of the points in S. Let Φ be a $m^2 \times 3$ matrix whose first row is $T_S(O_1)$, second row is $T_S(O_2)$, and so on. Let A be a $m^2 \times 1$ vector whose first entry is $U_S(O_1)$, second entry is $U_S(O_2)$, and so on.

We then solve

$$\min_{\boldsymbol{n}} \|\Phi\, \boldsymbol{n} - A\|_2$$

via standard least squares.

We experimented with $m \in \{3, 7, 11, 21, 31, 41\}$. Smaller m values capture finer resolution, which is useful for smaller objects.

4.1.2 Extracting a Geocentric Coordinate Frame

We observe that the gravity vector g imposes much structure on the real world, particularly within indoor settings (the floor and other supporting planes are usually orthogonal to g, whereas walls are aligned with g). Since the Kinect is not robust to biases resulting from camera rotation, it is critical to produce a gravity vector estimate $\hat{g}$ for each image.

We follow [13] to estimate the direction of gravity from the surface normals obtained previously. Intuitively, the algorithm tries to find the direction which is either most aligned to or most orthogonal to the surface normals at as many points as possible. We start by setting our initial estimate g_0 to be the Y-axis and refine the estimate with each iteration i

1. Use the current estimate $\hat{g}_{i-1}$, make hard assignments of surface normals to aligned set $\mathcal{S}_i^+$ and orthogonal set $\mathcal{S}_i^\perp$, (based on a threshold d on the angle between the surface normal and $\hat{g}_{i-1}$). Stack the vectors in $\mathcal{S}_i^+$ to form a matrix S_i^+, and similarly in $\mathcal{S}_i^\perp$ to form $S_i^\perp$.

   $$\mathcal{S}_i^+ \leftarrow \{\boldsymbol{n} | \theta(\boldsymbol{n}, \hat{g}_{i-1}) < d \text{ or } \theta(\boldsymbol{n}, \hat{g}_{i-1}) > 180^\circ - d\}$$
   $$\mathcal{S}_i^\perp \leftarrow \{\boldsymbol{n} | 90^\circ - d < \theta(\boldsymbol{n}, \hat{g}_{i-1}) < 90^\circ + d\}$$
   $$\text{where } \theta(x, y) \triangleq \text{angle between } x \text{ and } y \text{ constrained to } [0^\circ, 180^\circ]$$

   Intuitively, $\mathcal{S}^+$ should contain surface normals that are aligned with g (points on the floor and tables) while $\mathcal{S}^\perp$ should contain surface normals that are orthogonal to g (points on the wall).

2. Estimate a new $\hat{g}_i$ that maximizes both alignment to surface normals in $\mathcal{S}_i^+$ and orthogonality to surface normals in $\mathcal{S}_i^\perp$. We solve the optimization problem

   $$\hat{g}_i \leftarrow \arg \min_{g:\|g\|_2=1} \sum_{\boldsymbol{n} \in \mathcal{S}_i^\perp} \cos^2(\theta(\boldsymbol{n}, g)) + \sum_{\boldsymbol{n} \in \mathcal{S}_i^+} \sin^2(\theta(\boldsymbol{n}, g))$$

   This simplifies into finding the the smallest eigenvector of the 3×3 matrix: $S_i^\perp (S_i^\perp)^T - S_i^+ (S_i^+)^T$

We run 5 iterations with $d = 45^\circ$ followed by 5 iterations with $d = 15^\circ$. Using this estimate, we rotate our normals such that the y-axis is roughly in the direction of g.

4.1.3 K-Means

Using surface normal estimates at each pixel, we follow HOG's structure by histogramming them. A natural approach is to run k-means over the surface normals and treat the cluster centroids as bins. k-means is an algorithm that partitions a set of data points into k different clusters in a way that minimizes the within-cluster sum of squares.

Given n 3D surface normals $x_1, \ldots, x_n \in \mathbb{R}^3$, we wish to assign corresponding cluster labels $z_1, \ldots, z_n \in \{1, \ldots, k\}$ to those normals. We define $S_i = \{x_j | z_j = i\}$ and k cluster centroids $\mu_1, \ldots, \mu_k$, where $\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$. We initialize $\mu_i^{(0)} = x_j$ with j random. k-means consists of 2 iterative steps at each time step t:

1. $\forall i \in \{1, \ldots, n\} : z_i^{(t+1)} \leftarrow \arg\min_{j \in \{1,\ldots,k\}} d(\mu_j^{(t)}, x_i)$ where d is some distance metric. In practice, we let d be the cosine distance.
2. $\forall i \in \{1, \ldots, k\} : \mu_i^{(t+1)} \leftarrow \frac{1}{|S_i^{(t+1)}|} \sum_{x_j \in S_i^{(t+1)}} x_j$

Fig. 1 gives a visualization of the clustered means.

4.1.4 Feature Extraction

Let $\hat{n}(x, y)$ be the surface normal estimate at a pixel (x, y) in an image, and let $\mu_1, \ldots, \mu_k$ be the centroids obtained from k-Means. The surface normal estimate at each pixel is discretized into one of k values via vector quantization

$$v(x, y) \triangleq \arg \min_{i \in \{1,\ldots,k\}} d(\mu_i, \hat{n}(x, y))$$

We define a pixel-level feature map that specifies a sparse histogram at each pixel. Let $c \in \{1, \ldots, k\}$ range over cluster centers. The feature vector at (x, y) is

$$F(x, y)_c = \begin{cases} 1 & \text{if } c = v(x, y) \\ 0 & \text{otherwise} \end{cases}$$

We define a dense grid of $n \times n$ "cells". We wish to spatially aggregate values from F, a $w \times h$ pixel-level feature map, to produce a cell-based feature map C, with feature vectors $C(i, j)$ for $0 \leq i \leq \lfloor (w-1)/k \rfloor$ and $0 \leq j \leq \lfloor (h-1)/k \rfloor$. This aggregation provides invariance to small deformations and inconsistencies and reduces the size of the feature map. We follow HOG and use a "soft binning" approach where each pixel-level feature contributes to the cell-level feature vectors in the four cells around it using bilinear interpolation. After constructing these cell histograms, we $l1$ normalize them.

4.2 Configuring DPM

In practice, we let $k = 36$ and mimic DPM by letting our cells be 8pixels $\times$ 8pixels. Fitting our depth descriptors into the DPM pipeline consists of a few changes. Most importantly, we must consider the question of how to use the obtained features. Since the cell grid structure in our depth framework is identical to HOG, it is natural to stack the features. In particular, let $G(i, j)$ be the 31 dimensional feature vector on RGB at cell location (i, j) produced by DPM [7], and let $G'(i, j)$ be our depth features at the same location. The stacked cell features to be used in the pipeline are $H(i, j) \triangleq \left[G(i, j); \quad G'(i, j)\right]$ with $|H(i, j)| = 31 + 36 = 67$.

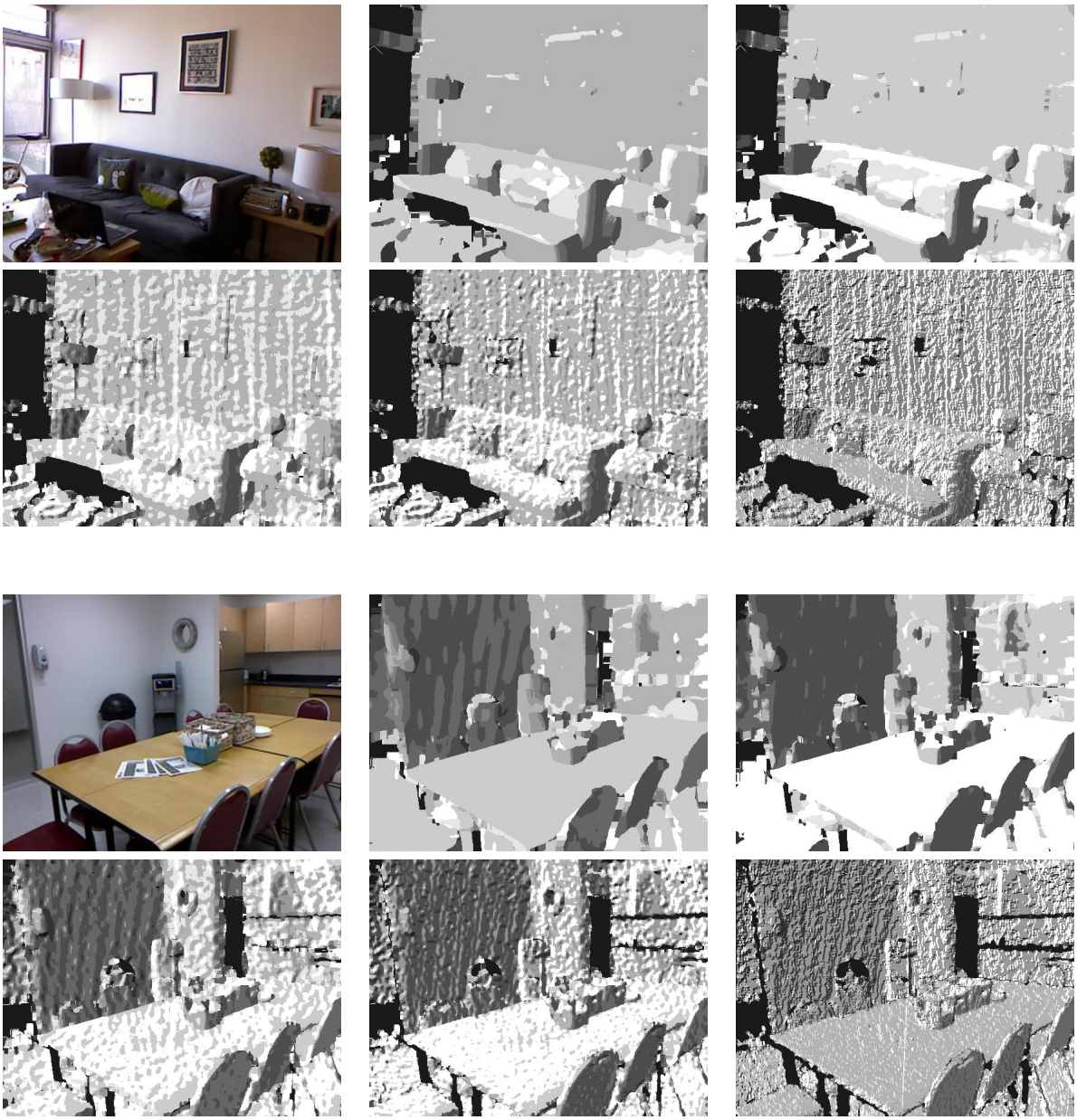
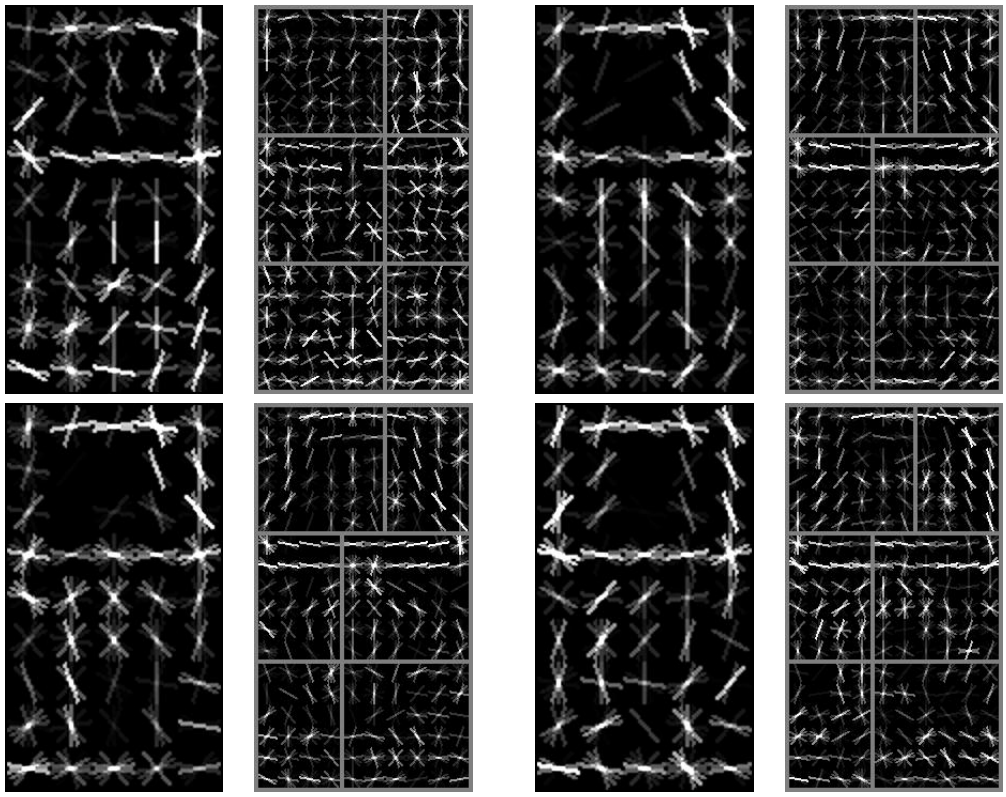
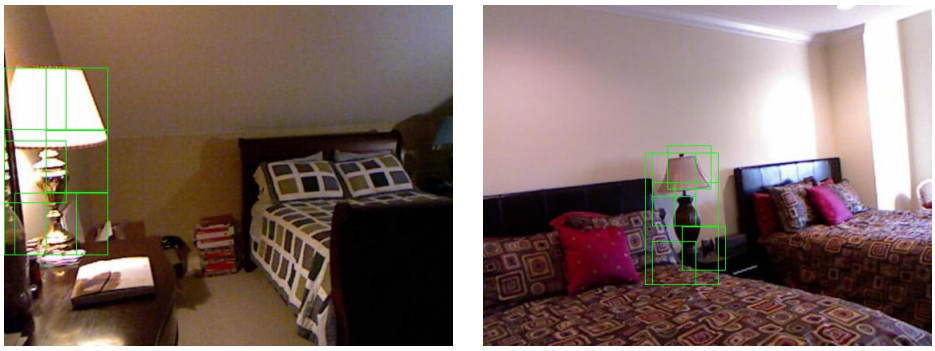


Fig. 1: Visualizing Vector Quantized Normals

We demonstrate a fine to course grained progression of our vector quantized surface normal estimates. RGB is presented in the top left corner of each image group. The ensuing images show the vector quantized normals, with each gray shade representing a distinct cluster center. The patch size m ranges from 31, 21, 11, 7, to 3.



a) b)



c)

Fig. 2: Models and Detections for Lamp Class

a) An RGB model with 2 components, with visualizations for root and part filters
b) Same as a), but trained on RGB-D and patch size m=3. For this visualization, cluster means are limited to x, y and snapped to a θ orientation as in HOG
c) 2 sample detections

We also modify the design choice for image patch resizing when computing features at different resolutions in the pyramid. In standard DPM, image scaling is carried out in a fast implementation that interpolates a pixel's spatial neighbors. This is less sensible for depth features, so we simply take the nearest neighbor when scaling.

Fig. 2 shows an example learned model.

5 Empirical Results

We evaluated our system on NYUD2 using the PASCAL protocols for evaluation. At test time, the goal is to predict the bounding boxes of all objects of a given class in an image (if any). In practice our system will output a set of bounding boxes with corresponding scores, and we threshold these scores at various levels to obtain a precision-recall curve. For a particular threshold the precision is the percentage of the reported bounding boxes that are correct, while recall is the percentage of the objects found.

A predicted bounding box is considered correct if it overlaps more than 50% with the ground truth, otherwise it is considered a false positive. We use a system's average precision (AP) across precision-curves as a measure of its performance.

NYUD2 provides instance and class labels, but not bounding box annotations. To extract bounding box annotations for training and testing, we take the min and max x and y coordinates over each instance, and set these to be the boundaries of the box for that particular instance. We use the same splits as provided by NYUD2. There are 795 training images and 654 test images. Each image contains a subset of the 10 evaluation classes. We note that scarcity of data may contribute to sub-optimal performance of our system.

We compare the impact of our depth features on performance. In the overwhelming majority of cases, we see an improvement over just the RGB data by adding our features. We also compare the effect of using just the root filter vs. with parts, components vs. no components, and varying the patch size used to compute surface normals.

In Tab. 1, we find that adding parts gives a substantial boost. Using parts in the DPM framework allows the system to capture intra-class variability.

Adding additional components does not appear to significantly impact performance. This finding may be the result of two factors. Due to the nature of data collection in indoor environments there is little variation among camera viewpoints and object poses, implying that enforcing multiple components on an object class is not necessarily the correct thing to do. Also, a lack of sufficient training examples prevents the components from taking on clearly defined structure.

[20] appears to suggest that adding components, in addition to a few minor modifications, can achieve performance comparable to the boost from parts. Our results do not affirm this conclusion, although the datasets are very different.

We observe that monotonically changing patch size does not monotonically

	bed	blinds	chair	counter	garbage bin	lamp	pillow	sink	sofa	window
a) !d	0.169	0.0667	0.0758	0.0703	0.1	0.191	0.071	0.0782	0.0368	0.0863
b) base	0.243	0.0483	0.0821	0.0708	0.0773	0.166	0.123	0.196	0.0794	0.123
c) p	0.375	0.135	0.13	0.0866	0.116	0.165	0.126	0.184	0.153	0.132
d) c2	0.219	0.0429	0.105	0.0636	0.0936	0.163	0.141	0.209	0.0646	0.122
e) m=3	0.314	0.0404	0.0849	0.121	0.112	0.203	0.143	0.197	0.0924	0.139
f) m=7	0.227	0.0506	0.0702	0.0813	0.164	0.205	0.139	0.176	0.0672	0.149
g) m=11	0.231	0.0485	0.0771	0.0864	0.137	0.195	0.143	0.181	0.0868	0.129
h) m=21	0.237	0.0396	0.0809	0.0871	0.133	0.19	0.131	0.19	0.106	0.16
i) m=31	0.235	0.0434	0.0704	0.0818	0.117	0.174	0.116	0.166	0.0938	0.129
j) p; $m = \arg\max_m$ AP	0.36^{3}	0.194^{3}	0.151^{3}	0.164^{3}	0.155^{3}	0.234^{7}	0.168^{7}	0.193^{3}	0.154^{21}	0.171^{7}
k) p; c2; $m = \arg\max_m$ AP	0.32^{3}	0.195^{7}	0.147^{31}	0.151^{3}	0.128^{3}	0.211^{3}	0.169^{21}	0.23^{31}	0.155^{21}	0.177^{3}

Tab. 1: Average Precision on NYUD2

The "base" configuration uses: depth data, root filter only (no parts), 1 component, and patch size m=41. The other listed configurations modify one or more of these parameters. !d indicates no depth (hence, patch size is irrelevant). p indicates parts are used. c2 indicates 2 components. m indicates a different patch size. For h) and i), the patch size m is written as a superscript.

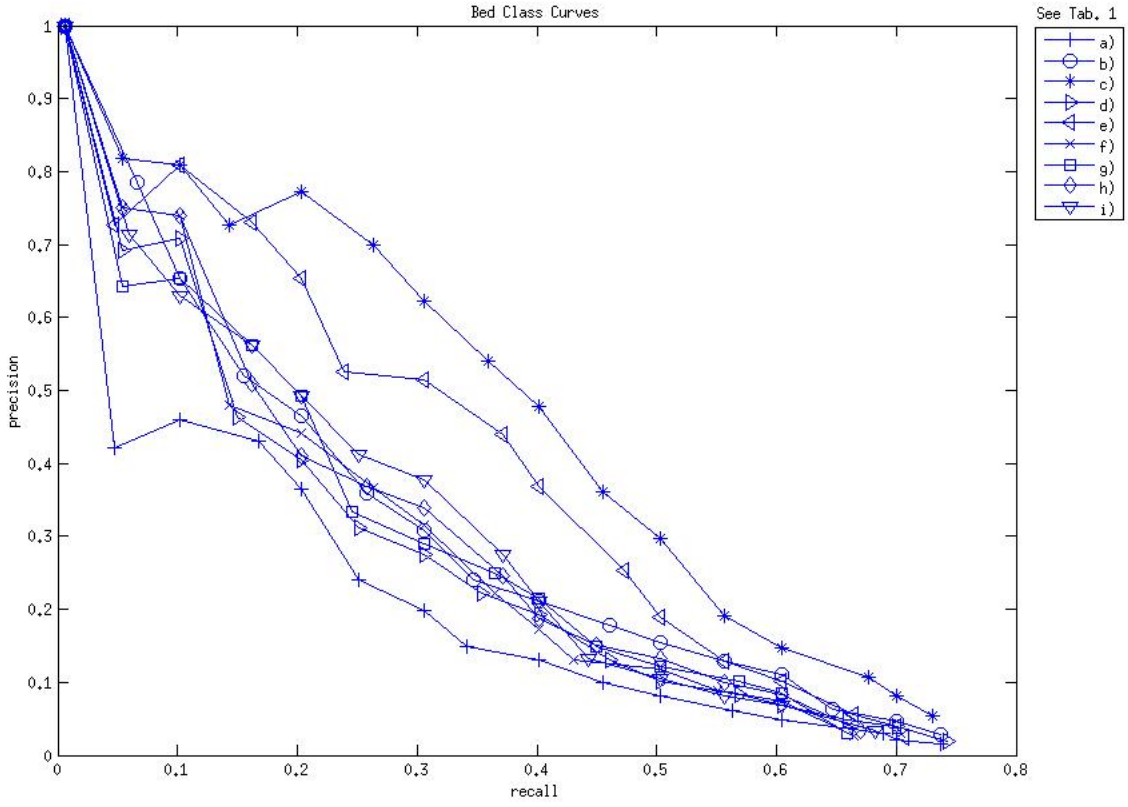


Figure 3: Precision Recall

change performance. Using a smaller patch size to estimate surface normals is useful for capturing the smaller objects, as it captures a finer-grained resolution. However, it is also more susceptible to errors due to noise, as the estimates are less smoothed. This tradeoff suggests that a combination of all patch sizes may achieve the best performance.

6 Conclusion

In this report, we have suggested a new method of using raw depth data to augment HOG-based object detectors. Our depth descriptors are efficiently computed and may be easily "bolted on" to existing methods that compute HOG on RGB. While our features empirically provide a dramatic boost to only RGB, it is important to note that the depth features are computed independently of RGB. In the future we would like to combine RGB and depth channels at the feature extraction stage, as opposed to considering them separately.

References

[1] Abhishek Anand, Hema Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for 3d point clouds. 2012.

[2] Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *IEEE 13th International Conference on Computer Vision (ICCV)*, pages 1092–1099. IEEE, November 2011.

[3] Lubomir Bourdev, Subhransu Maji, Thomas Brox, and Jitendra Malik. Detecting people using mutually consistent poselet activations. In *European Conference on Computer Vision (ECCV)*, 2010.

[4] Hua Chen, Oliver Wulf, and Bernardo Wagner. Object detection for a mobile robot using mixed reality. In *Proceedings of the 12th international conference on Interactive Technologies and Sociotechnical Systems*, VSMM'06, pages 466–475, Berlin, Heidelberg, 2006. Springer-Verlag.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886 –893 vol. 1, june 2005.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[8] Stephen Gould, Paul Baumstarck, and Morgan Quigley. Integrating visual and range data for robotic object detection.

[9] Hiroshi Hattori, Akihito Seki, Manabu Nishiyama, and Tomoki Watanabe. Stereo-based pedestrian detection using multiple patterns. In *BMVC'09*, pages –1–1, 2009.

[10] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1), October 2007.

[11] Tomas Pajdla Jan Smisek, Michal Jancosek. 3D with Kinect. *ICCV 2011*, November 2011.

[12] David Changsoo Lee, Martial Hebert, and Takeo Kanade. Geometric reasoning for single image structure recovery. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2009.

[13] Saurabh Gupta Pablo Arbelaez Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgbd images. Under review, CVPR 2013.

[14] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[15] Andreas Nuchter, Kai Lingemann, Joachim Hertzberg, and Hartmut Surmann. Accurate object localization in 3d laser range scans. In *In Proceedings of the 12th IEEE International Conference on Advanced Robotics (ICAR '05), pages 665 - 672*, 2005.

[16] Tahir Rabbani and Frank Van Den Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds.

[17] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2759–2766. IEEE, 2012.

[18] Marcus Rohrbach, Markus Enzweiler, and Dariu M. Gavrila. High-level fusion of depth and intensity for pedestrian classification. In *In Proc. DAGM*, pages 101–110, 2009.

[19] R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155 –2162, oct. 2010.

[20] Martial Hebert Santosh K. Divvala, Alexei A. Efros. How important are 'deformable parts' in the deformable parts model? In *Parts and Attributes Workshop, ECCV 2012*, October 2012.

[21] 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 *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1297–1304, Washington, DC, USA, 2011. IEEE Computer Society.

[22] Stefan Walk, Konrad Schindler, and Bernt Schiele. Disparity statistics for pedestrian detection: combining appearance, motion and stereo. In *Proceedings of the 11th European conference on Computer vision: Part VI*, ECCV'10, pages 182–195, Berlin, Heidelberg, 2010. Springer-Verlag.