Scaling for Multimodal 3D Object Detection

Andrej Karpathy
Stanford
karpathy@cs.stanford.edu

Abstract

We investigate two methods for scalable 3D object detection. We base our approach on a recently proposed template matching algorithm [8] for detecting 3D objects. First, we demonstrate that it is possible to gain significant increase in runtime performance of the algorithm at almost no cost in accuracy by quickly rejecting most regions of the image with low-resolution templates. Second, we investigate an implicit part-based model that uses fixed-sized template dictionary and a Generalized Hough Transform framework to detect objects. We present results on two separate datasets that we collected using the Kinect sensor.

1. Introduction

Real-time object learning and detection are important and challenging tasks in Computer Vision. Especially in the field of robotics, there is a need for algorithms that can enable autonomous systems to continuously learn to recognize new objects. For such time-critical applications, template matching is an attractive solution because new objects can be easily learned online, in contrast to statistical techniques that often require a time-consuming training stage.

An efficient template-based object detection algorithm has recently been proposed [8] that works in real-time, does not require a time consuming training stage, and can handle untextured objects. It is based on an efficient representation of templates that capture color, gradient and depth modalities. However, the template-based approach scales linearly with the number of objects and views, making it difficult to use in an application that requires detection of many objects and viewpoints. In this work, we use the same efficient feature extraction and matching algorithm, but address the scalability issues in two separate ways, each with its own trade-offs.

First, we propose a method of speeding up the template matching procedure by pre-filtering the image with low-resolution versions of all templates to quickly reject parts of the image that are unlikely to contain objects of interest.

Second, we show how to use use a fixed-size dictionary of random templates to detect parts of objects in the image. This approach is inspired by recent work [20] that suggests that even random filters can give rise to responses that can be used in a discriminative setting. We use the Generalized Hough Transform to accumulate votes for object center from weakly detected parts to detect whole objects.

The remainder of this report is structured as follows. We first discuss related work and introduce our approach. We then present results of experiments on two new datasets.

2. Related Work

Object detection is a widely studied topic in the literature. Below we briefly summarize the most common approaches to this problem.

Template Matching This technique is attractive due to its simplicity and its lack of assumptions about the background of objects. In general, it tends to work well in scenarios where a few rigid objects of fairly distinct appearance are concerned. However, the approach does not immediately scale to object detection that involves many classes, views and deformable objects due to computational concerns. There have been many attempts to addressing this problem. [8] optimizes the performance of template matching with a design of features and matching algorithm that can be computed very quickly, and by minimizing the number of cache misses during computation. Approaches such as [7] speed up template matching using distance transform that group templates together to avoid matching all templates in a brute force manner. Other approaches attempt to quickly reject parts of the image that are unlikely to contain objects of interest. [18] uses image segmentation, where as other approaches utilize relatively cheap root filters in a cascade detection framework [4, 17]. Our first technique of speeding up the template matching procedure draws mostly on these ideas.

Deformable Part-based Models. To address the exponential increase in the number of templates that are required for objects that can exhibit a large number of distinct appearances (such as deformable objects) many part-based approaches have been proposed that combine template matching with a part-based model that enforces soft constraints

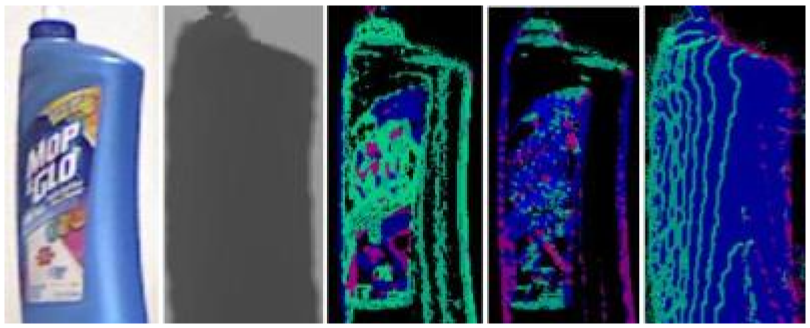


Figure 1. Visualization of the features computed from an image. From left to right: color image, depth image, color modality, gradient orientation modality, and depth orientation modality. A different color is used for every possibly value that a modality can take on (one of 0 to n_0. Here n_0 = 8 is used.)

on relative locations of parts. Popular approaches include Constellation models [12, 21], Star-shaped models [1, 13, 24, 26, 22, 19], and flexible part-based models learned in a latent structural large margin frameworks [5, 27]. In this work, we do not consider these explicit part-based models as most require a relatively expensive training stage, or take a long time to compute on a test image.

Generalized Hough Transform Models. Hough Transform [9] is a classical Computer Vision algorithm that was originally used for line detection. In recent years, however, Hough-based methods have been successfully adapted to problem of part-based object detection, where they constitute an implicit shape model. [3, 6, 11, 14, 16]. The main idea in these methods is to first detect parts of objects independently, and then use the detected parts to cast votes for the object center in the image. Our second approach is similar to these methods, except we use different features and matching algorithm to detect object parts.

Multimodal Detection. Especially in area of Robotics, the availability of RGBD sensors such as the Kinect have allowed researchers to utilize depth information and achieve much higher object detection rates. It has become relatively common, therefore, to see methods that combine multimodal data in a single detection framework [8, 25, 10, 2].

3. Approach

In this section, we first describe the multimodal feature extraction and matching algorithms. These steps are similar to those described in [8]. We then discuss two approaches we've investigated that serve to replace the naive, brute-force template matching scheme in the previous method.

3.1. Feature Extraction and Representation

To compute the feature representation, an RGB image and a corresponding depth map are converted to a set of three modalities, $\{\mathcal{O}_m\}_{m\in\mathcal{M}}$. Here, $m \in \mathcal{M}$ denotes one of three modalities: gradient orientation, depth orientation, and color. Each modality is a discrete-valued two-dimensional array of the size of the original image.

Gradient Orientation modality is computed from the image by calculating the orientation of the gradient at every point in the image. The orientation is computed on all three color channels separately, and only the orientation that corresponds to the channel with the highest gradient magnitude is retained at each location. To achieve additional invariance to illumination changes, only the absolute value of the orientation is computed at every point. Locations in the image that have a very low gradient magnitude are ignored.

Depth Orientation modality is identical to the gradient modality, except the gradients are computed on the depth map.

Color modality is computed by first converting the RGB image into HSV color space, and then keeping only the hue component. Locations in the image that have a very low gradient magnitude or very low saturation are ignored.

Finally, all modality maps are discretized into one of n_0 values. Examples of computed modalities are visualized in Figure 1. Throughout this work we set $n_0 = 8$, as this allows us to store every image as an array of 8-bit values with a 1-of-n_0 encoding for every location.

3.2. Template Matching

Given a set of aligned modalities $\{\mathcal{O}_m\}_{m\in\mathcal{M}}$, a template is defined as $\mathcal{T} = (\{\mathcal{O}\}_{m\in\mathcal{M}}, \mathcal{L})$. $\mathcal{L}$ is a list of pairs (r, m) where r is a location of a feature in modality m. The similarity measure between a template and an image patch is defined as:

$$\mathcal{E}(\{\mathcal{I}\}_{m\in\mathcal{M}}, \mathcal{T}, c) = \sum_{(r,m)\in\mathcal{L}} \left(\max_{t\in\mathcal{R}(c+r)} f_m(\mathcal{O}_m(r), \mathcal{I}_m(t)) \right)$$

where $\mathcal{R}(c+r) = [c+r-\frac{T}{2}, c+r+\frac{T}{2}] \times [c+r-\frac{T}{2}, c+r+\frac{T}{2}]$ defines the neighborhood of size T cen-

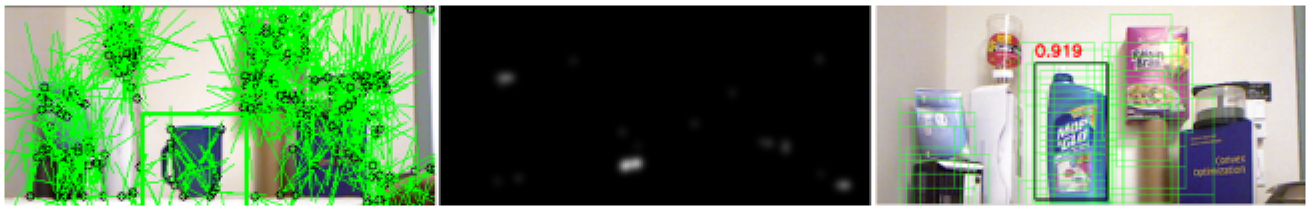


Figure 2. **Left**: Visualization of Hough voting scheme. Every black circle is a weakly-detected part, and green lines indicate votes. The green lines meeting at the center of the tea cup give rise to a noticeable peak in the response map for that object (**Center**). **Right**: Visualization of the root filter template matching. Proposed locations are shown in green and filtered further. The black rectangle is a true positive match obtained from subsequent filtering.

tered on location $c + r$ in the input image $\mathcal{I}_m$ and the function $f_m(\mathcal{O}_m(r), \mathcal{I}_m(t))$ computes the similarity score for modality m between the reference image at location r and the input image at location t. Thus, each feature in a template is aligned locally in a small neighborhood to the most similar feature in the input image. This formulation allows for small changes in the input patch by adding a slight amount of invariance to the matching algorithm.

The similarity measure is used to perform template matching with a simple sliding window approach together with non-maximum suppression.

3.3. Template Matching with Root Filters

In this section, we describe an algorithm to speed up the template matching approach of the original method. In its derivation, we draw inspiration from recent approaches that achieve significant boosts in time complexity without sacrificing performance by quickly rejecting most parts of the image.

Therefore, we re-define the similarity measure as follows:

$$\mathcal{E}_{p,\tau}(\{\mathcal{I}\}_{m\in\mathcal{M}}, \mathcal{T}, c) = \begin{cases} 0, & \mathcal{E}(\{\mathcal{I}^p\}_{m\in\mathcal{M}}, \mathcal{T}^p, pc) < \tau \\ \mathcal{E}(\{\mathcal{I}\}_{m\in\mathcal{M}}, \mathcal{T}, c), & \text{otherwise} \end{cases}$$

where $\mathcal{I}^p$ is the image downsampled by a factor of p, $\mathcal{T}^p$ are the templates downsampled by a factor of p, and τ is a threshold value that must be manually set. Intuitively, the image is first filtered against all templates on a lower resolution and only positions that score above the threshold τ are further investigated as potential positives using the full-resolution templates.

While this approach does not reduce the asymptotic complexity of the original method, it can still significantly reduce the runtime of the algorithm in practice if one is willing to tolerate a potential decrease in accuracy. A downside of the approach is that a naive implementation requires additional storage devoted to templates of lower resolution for all objects and views. One way to address this issue is to not create a new template if an existing template is sufficiently similar. This can naturally enforce an upper bound on the number of templates stored and establish a tunable trade-off between space complexity and accuracy.

3.4. Generalized Hough Transform for Implicit Part Model

Model. In the Generalized Hough Transform framework, an object is detected through a consensus vote of local, independent and weak object part detectors. Each weak detector maintains a distribution of the location of the object center relative to its location. For example, detecting a part similar to the top of a bottle can be an indicator of a bottle present below.

More specifically, we maintain a codebook $\mathcal{C}$ of size N, where each element $\mathcal{C}_i$ represents an object part. For every part there is an associated probability distribution $P(O_n, x|\mathcal{C}_i)$ over all objects O_n and locations in the image, x that are represented in the local coordinate system of the part. Assuming that every parts prediction is independent and equally important, the probability of $P(O_n, x)$ can be obtained by simply adding the probabilities $P(O_n, x|\mathcal{C}_i)$ for each detected part, offset by its location in the image.

The distribution $P(O_n, x|\mathcal{C}_i)$ can be learned from the training data by frequency counting. Every time a part O_n is detected on an object, the location of the object center relative to the part is recorded and stored in memory. Together, all records for a part represent a non-parametric model of the distribution that can be stored as a sparse matrix of size $K \times W \times H$ where K is the number of classes, and W, H are the size of a receptive field of each part. In this work we simply set W, H to be the size of the image.

Part Learning. We now describe the mechanism for learning and detecting object parts in the image. Each part $\mathcal{C}_i$ is a vector of responses for a set of m fixed, random templates $\mathcal{T}_i$, $i = 1..m$:

$$\mathcal{C}_i = [\mathcal{E}(\mathcal{I}_t, \mathcal{T}_1, c_i), ..., \mathcal{E}(\mathcal{I}_t, \mathcal{T}_m, c_i)]$$

where $\mathcal{I}_t$ is a training image of one of the objects, and c is a location inside the mask for that object. For efficiency, we only learn parts at repeatable locations in the image. Specifically, we choose to use Harris corner keypoints for this purpose.

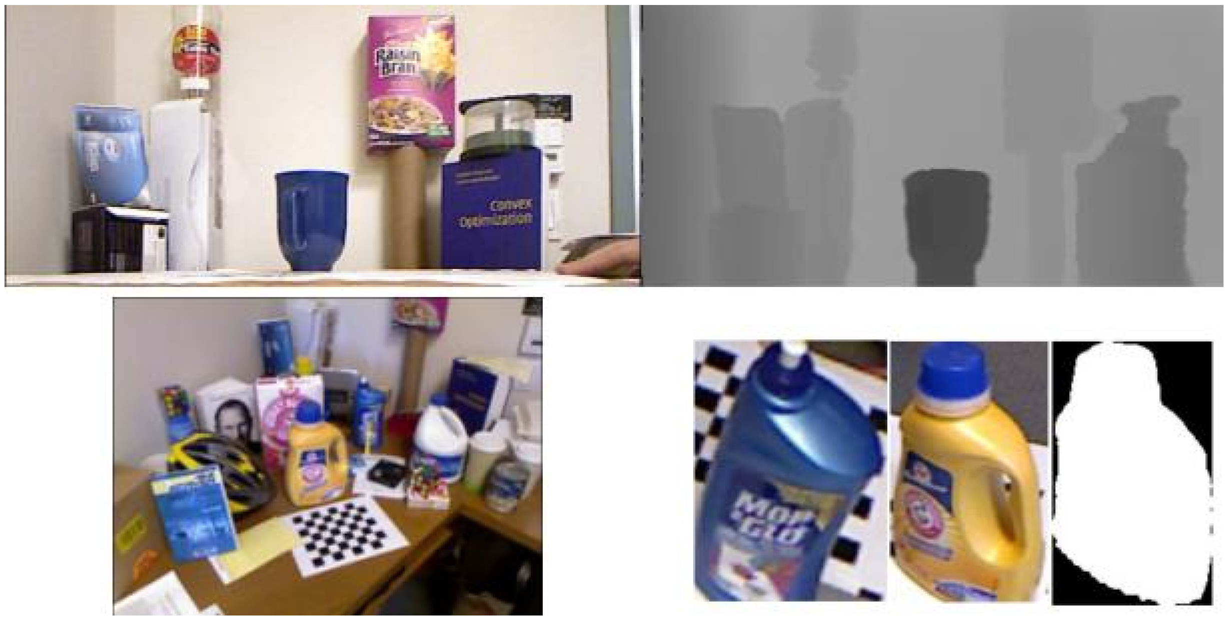


Figure 3. Example images from the collected dataset. **Top**: Example of the image and its depth from the turntable dataset. **Bottom left**: example image from the ”in the wild” dataset. **Bottom right**: The two bottles we attempt to detect, and an example mask for one of them. Note that the blue bottle is partially occluded by a yellow marker in the test set.

Detection. Given a location l in an image I, we can similarly form a vector of responses of every one of our random templates at that location, $[\mathcal{E}(\mathcal{I},\mathcal{T}_1,l),...,\mathcal{E}(\mathcal{I},\mathcal{T}_m,l)]$ and match the responses to those obtained during training using L2 distance. We consider a part to be detected at some location if the minimum L2 distance is below some threshold.

This method has several attractive properties:

- The complexity of the learning stage is very low, as every image requires us to merely store each detected part on the object and its offset from the object center. In practice, it is desirable to only create a new part if it is not too close to a part that already exists, but this simply reduces to a nearest neighbor lookup that can be implemented efficiently using Approximate Nearest Neighbor techniques, such as kd-trees. Efficient implementations such as the FLANN [15] library exist for this purpose. Thus, this approach can be used to train a new detector online.

- The number of templates, m, stays constant and does not grow linearly with the number of objects and views.

- Part detection in the image can be computed efficiently using Approximate Nearest Neighbor techniques, as every detection can be reduced to a k-nearest neighbor search.

- Detection of whole objects during test time given a set of detected parts reduces to simple addition of votes for each part. This can be efficiently implemented and lends itself to parallelization across both object parts and image regions.

4. Dataset Collection

The data used in our experiments are collected using the Microsoft Kinect sensor. The raw Kinect depth map is not complete due to occlusions and often contains holes in specular objects. In our experiments we address this technicality by in-painting the holes using the method described in [23]. This results in a smooth depth map with all holes filled, as can be seen in Figure 3.

4.1. Ten objects on turntable

The ”Ten objects on turntable” dataset is intended to test the discriminative power of the algorithm. It consists a set of 11 distinct household objects: book, four bottles, calculator, can of coke, coffee cup, tea cup, bicycle helmet, and tea box. These objects were chosen specifically to cover large variations in color, size, shape, and texture.

Each object is placed on a turntable and rotated 360 degrees. The Kinect sensor is placed at the height of the turntable and captures about 50 pairs of color and depth images. The ground truth mask for each object view is computed using a manually chosen depth threshold. Examples of images obtained are shown in Figure 3. As can be seen in the image, we also place a number of distractor objects in the background, which can produce false positives for low

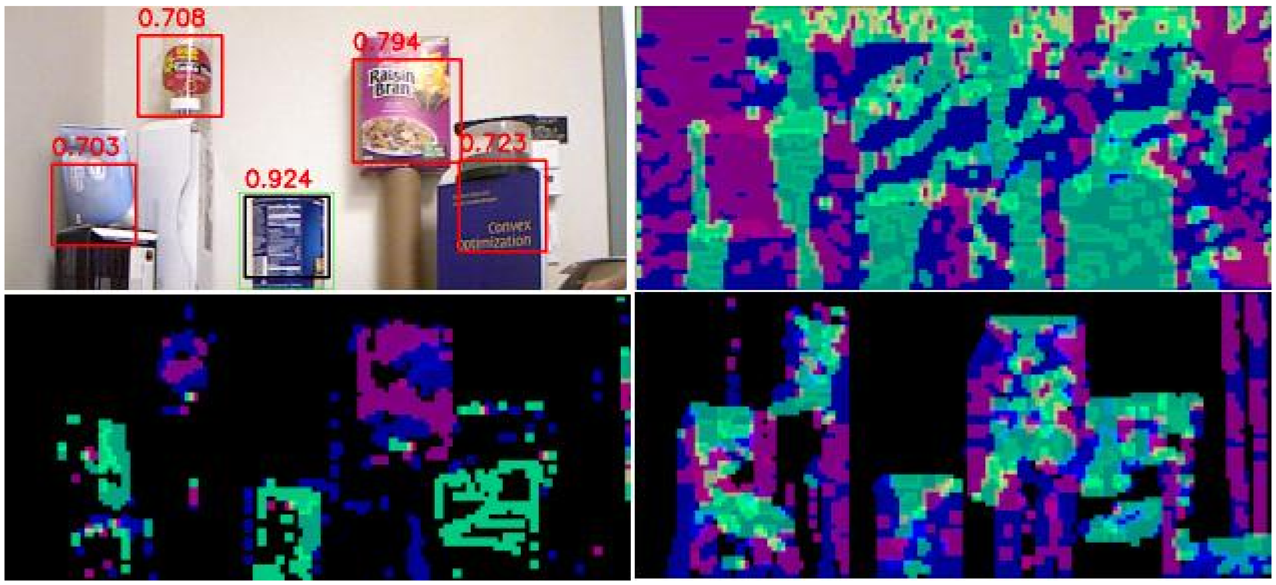


Figure 4. **top left**: Example detection and features. In this example, the threshold is set very low so several false positives appear, as depicted by the red bounding boxes. Note that the score of the true positive (0.924) is significantly higher than that of the false positives. Around the image is a visualization of the computed features on a downsampled image: depth (top right), color (bottom left), and gradient orientation (bottom right). Every one of n_0 bins is associated with a distinct color.

detection thresholds.

4.2. Objects "in the wild"

The Objects ”in the wild” dataset is intended to test the ability of the algorithm to detect objects in a heavily-cluttered environment from many viewpoints and with some occlusions.

To collect training images we place two of the objects on the turntable in turn and move the Kinect freely around the object to acquire many images from all possible views, scales, rotations, out of plane rotations, and tilts. On the order of 300 such color, depth, and mask images are obtained from this procedure for each object. Examples of acquired images are shown in 3.

The precise mask segmentation is automatically computed as follows. Sheets of paper with a black and white square calibration pattern are placed on the turntable next to objects. Next, corners in the calibration pattern are detected using the Harris corner detector. Since the corners are known to be collinear in the world coordinate frame and located on the turntable, we use RANSAC to fit an affine function to the depth of each corner as a function of its location in the image. The resulting affine function can be used to extrapolate the depth of the turntable plane, and every pixel that is closer to the camera than the plane obtained from the fit is defined to belong to the object.

Finally, to collect the test set we place both objects in a cluttered environment and partially occlude one with a marker. We move the Kinect freely around the table at different scales, tilts and rotations while saving color and depth images.

5. Results

We compare the results of the proposed methods on the two collected datasets discussed above. All experiments were performed on one processor of an Intel Core 2 Quad core CPU at 2.66 GHz and 2 GB of RAM. It is important to note that the exact implementation of the feature computation and matching algorithm used in this project is different from the one used in the original work [8]. In particular, our implementation is not optimized and is slower by more than an order of magnitude.

5.0.1 Turntable dataset

All images in the dataset are randomly split within each class such that three quarters are labelled as training set, and the rest are used for testing. This procedure yields a total of 410 training images and 143 testing images. Each image is downsampled by a factor of 4 to size of 160x120 pixels for efficiency. Every image contains exactly one of 11 classes, but this information is not used by the detector during test time.

The performance is evaluated by changing the detector threshold during template matching. Non-maximum suppression is performed on raw detection results such that any detection windows that overlap by more than a fraction of

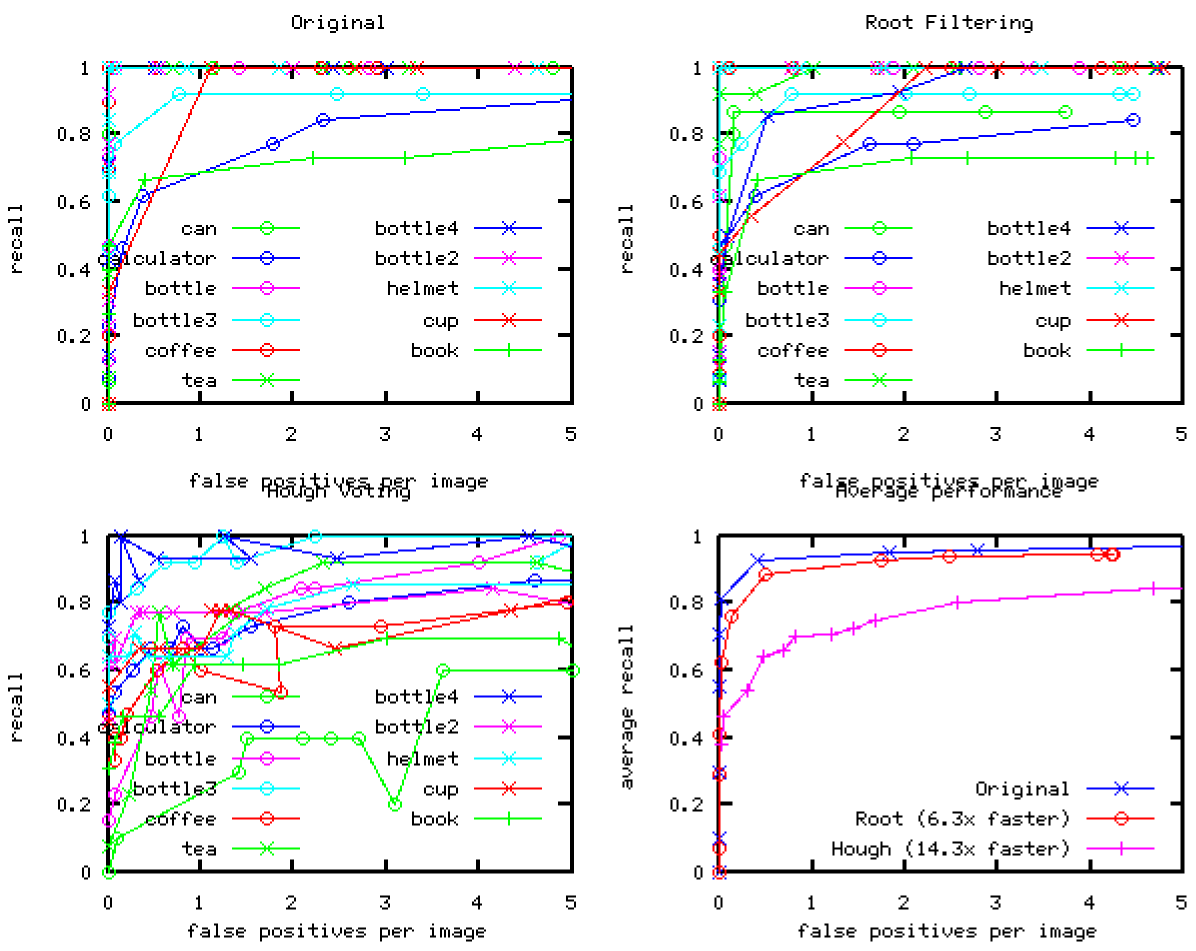


Figure 5. Detection results on the turntable dataset. The absolute running time for the original method is 0.3 frames per second. I apologize for the inability of Octave to produce simple plots. From bottom left, clockwise: Recall vs. false positives per image of Hough voting, original template matching, and Root Filtering. Bottom right: average recall for each method.

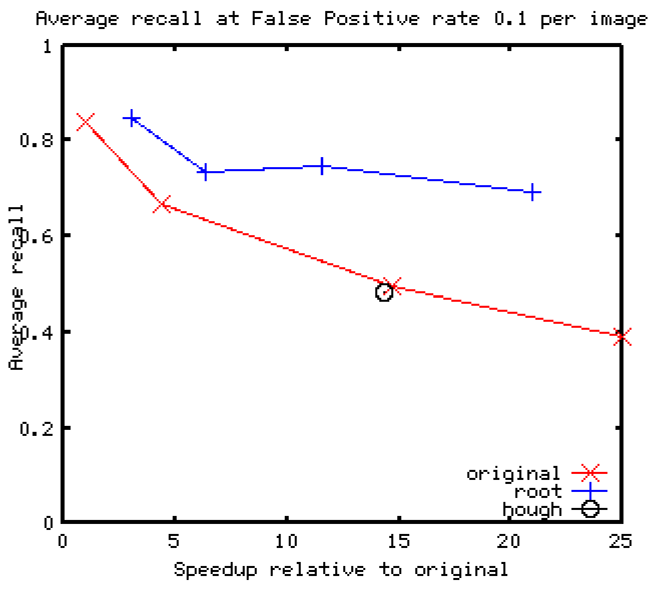
0.2 in the intersection over union metric are considered to be in conflict, and only the detection with higher score is retained.

For the Hough voting method we use $m = 120$ random templates of size 12. For approximate nearest neighbor we use 4 kd-trees and 100 checks. After all votes are cast, we smooth the response map using a gaussian with $\sigma = 3$.

Per class performance. The results are summarized in Figure 5. As can be seen in figure, the root filtering approach can significantly improve the speed (6.3x, to about 2fps) if one is willing to sacrifice a small amount of accuracy. We explore this trade-off further below.

The Hough Voting method is seen to achieve a lower performance, but runs relatively quickly (14.3x, about 4fps). It is also interesting to note that the method yields highly varied outcomes for different objects, performing almost perfectly on some, and poorly on others. In particular, the two objects it performs worst on are tea and can, which are the two smallest objects in the dataset. This suggests that these objects had trouble accumulating votes for their centers. In addition, from manually inspecting the false positives and negatives it appears that this method performs badly in general when relatively few pixels in the image belong to the object, such as when objects are viewed from the side. Accordingly, future work could potentially improve on these results by using a different detection threshold for each object or view, or scaling votes based on the size of each object. We attempted a few of these changes but were unable to significantly improve the performance.

Speed vs. Performance. In Figure 6 we explicitly investigate the speed vs. performance trade-offs for each method by fixing the rate of false detections per image we are willing to tolerate to 0.1, and plotting the average recall. To increase speed for the original method, we monotonically downsample all images and templates. To increase speed for the root filtering method, we monotonically downsam-



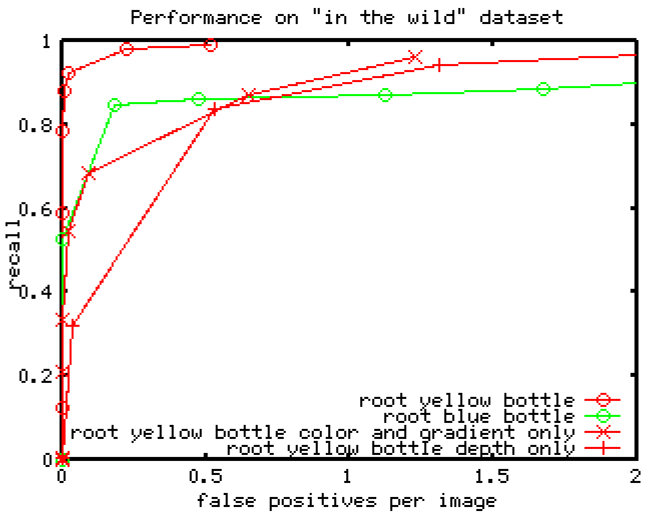


Figure 6. **Left:** Speed vs accuracy trade-offs. **Right**: Performance of the root filtering approach on ”in the wild” dataset.

ple both pre-filtering and post-filtering images by lower increments. We only evaluate the Hough voting method a single time because it is not immediately obvious how one could go about speeding it up. The figure shows that the root filtering approach to template matching can retain most of the recall with significant gains in performance (up to 20x and more). The Hough voting approach is seen to be comparable with downsampling images and running naive template matching.

5.0.2 Objects ”in the wild” dataset

The ”in the wild” dataset consists of 400 views for both objects, and 200 testing images. Template matching is done over 3 scales because the camera is not always at constant distance from the object. Results of the root filtering approach are shown in Figure 6. The 3 scales of the image can be processed at about 2 frames per second per object. As can be seen in the figure, both objects can be reliably detected. The blue bottle is harder to detect because it is smaller, less distinct from its environment, and partially occluded. Upon manual inspection, most false negatives can be attributed to motion blur. Example of these frames can be seen in Figure 7.

We also investigate the usefulness of modalities towards the final performance. As can be seen from the figure, including depth information significantly improves the performance of the detector. However, by itself depth performs on par with color and gradients. This indicates that depth captures useful information about the object that is orthogonal in nature to an ordinary image. To some degree, this is merely a quantitative support of an expected result.

6. Conclusion

We presented two extensions of an existing template matching scheme for 3D object detection. The idea of conducting the template matching procedure on two scales of the images for quick rejection has proven to be significantly more efficient without much loss in performance.

The approach that uses detection of parts in a Generalized Hough transform framework was found to perform comparably to template matching on low-resolution images in both speed and detection performance. By manual inspection of the classification results we speculate that this is in part because some views of objects (such as a book viewed from the side) contain too little evidence to reliably detect object parts. While this is in principle an issue for the template-based method as well, it was not found to affect the performance to such a high degree, potentially because together all pixels belonging to the object provide enough evidence for a confident match.

Future work could involve applying explicit part based models to this problem, such as start-shaped models or constellation models, but question remains if these models can be adapted to compete with the speed of root-based template matching for rigid objects. An interesting direction would be to try to obtain additional speedups in the root-based template matching method by not only rejecting image regions, but also rejecting templates before they get applied. For example, one could use the similarity of templates to reason about the correlation in their response values on a given patch, and potentially choosing to not apply a template if the expected score is too low.

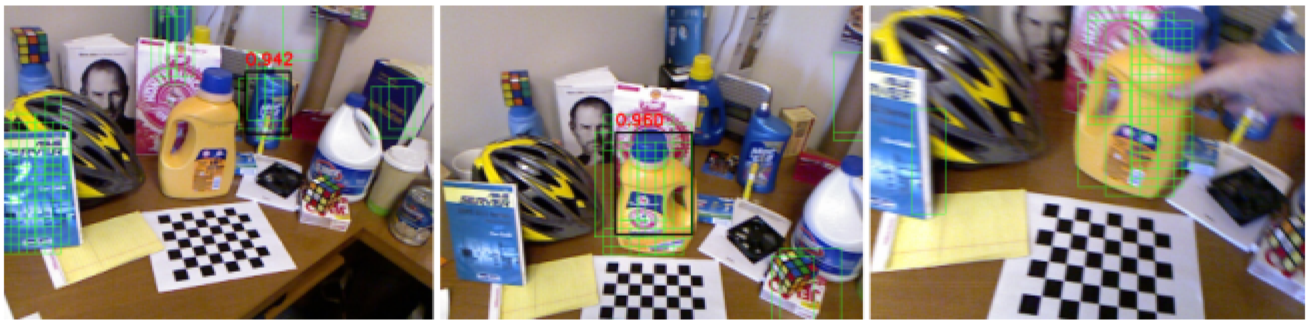


Figure 7. Examples of detection on ”in the wild” dataset. Proposed detections by root filter are shown in green, detections in black. **Left**: the blue bottle in the back is correctly detected despite the occluding yellow marker. Center: yellow bottle is correctly detected. Right: Example of an image where the yellow bottle was not correctly detected due to motion blur.

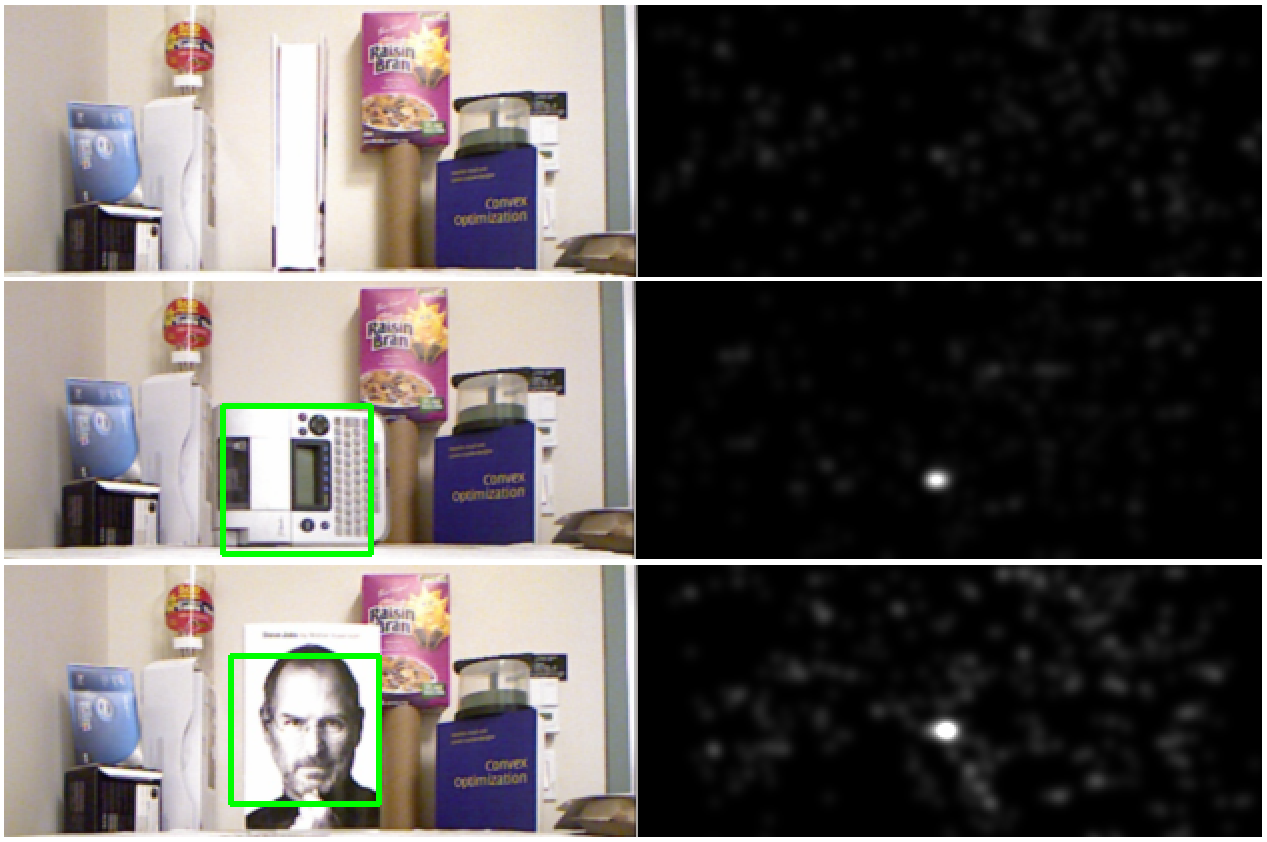


Figure 8. Example detections using the Hough voting scheme. Top: a typical failure case. Most failures occur when only few pixels of the object are visible, such as in this case of a book viewed from the side. When the objects are extended and clearly visible, maxima are usually easily found.

References

[1] M. Arie-Nachimson and R. Basri. Constructing implicit 3d shape models for pose estimation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1341–1348. IEEE, 2009.

[2] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *IEEE International Conference on Computer Vision and Pattern Recognition*, June 2011.

[3] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1365–1372. IEEE, 2009.

[4] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Computer*

vision and pattern recognition (CVPR), 2010 IEEE conference on, pages 2241–2248. IEEE, 2010.

[5] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://www.cs.brown.edu/ pff/latent-release4/.

[6] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1022–1029. Ieee, 2009.

[7] D. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 439 –444 vol.1, aug 1998.

[8] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. 2011.

[9] P. Hough. Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation*, volume 73, 1959.

[10] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[11] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 17–32, 2004.

[12] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, May 2008.

[13] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3d feature maps. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[14] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1038–1045. IEEE, 2009.

[15] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[16] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. *Computer Vision–ECCV 2006*, pages 575–588, 2006.

[17] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[18] O. Russakovsky and A. Ng. A steiner tree approach to efficient object detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1070 –1077, june 2010.

[19] S. Savarese and L. Fei-Fei. Multi-view object categorization and pose estimation. *Studies in Computational Intelligence-Computer Vision*, pages 205–231, 2010.

[20] A. Saxe, P. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Ng. On random weights and unsupervised feature learning. In *Workshop: Deep Learning and Unsupervised Feature Learning (NIPS)*, 2010.

[21] M. Stark, M. Goesele, and B. Schiele. Back to the future: Learning shape models from 3d cad data. In *British Machine Vision Conference*, pages 106–1. Citeseer, 2010.

[22] M. Sun, H. Su, S. Savarese, and L. Fei-Fei. A multi-view probabilistic model for 3d object classes. In *Proc. Computer Vision and Pattern Recognition*, 2009.

[23] A. Telea. An image inpainting technique based on the fast marching method. *journal of graphics, gpu, and game tools*, 9(1):23–34, 2004.

[24] A. Thomas, V. Ferrar, B. Leibe, T. Tuytelaars, B. Schiel, and L. Van Gool. Towards multi-view object class detection. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1589–1596. IEEE, 2006.

[25] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 794–801. IEEE, 2009.

[26] P. Yan, S. Khan, and M. Shah. 3d model based object class detection in an arbitrary view. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–6. IEEE, 2007.

[27] Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. 2011.