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 [5] 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 [5] that runs 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 explore 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 [13] 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.

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. There have been many attempts to addressing the scalability issues associated with this method. [5] optimizes the layout of features in memory to minimize cache misses. [4] speeds up template matching using distance transform that group templates together to avoid matching all templates. [12] uses image segmentation to avoid exhaustive matching. Other approaches [2, 11] utilize relatively cheap root filters in a cascade detection framework. Our first technique of speeding up the template matching procedure draws mostly on these ideas.

Generalized Hough Transform Models. Hough Transform [6] is a classical Computer Vision algorithm that was originally used for line detection. In recent years, Hough-based methods have been successfully adapted to problem of part-based object detection, where they constitute an implicit shape model. [1, 3, 7, 8, 10]. 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.

3. Approach

In this section, we first describe the multimodal feature extraction and matching algorithms. These steps are similar to those described in [5]. 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

The RGB image and a corresponding depth map are converted to a set of three feature 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

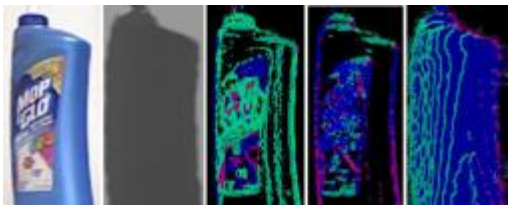

Figure 1. Visualization of the features computed from an image. From left to right: color image, depth image, color modality, gradient orientation modality, and depth orientation modality. A different color is used for every possibly value that a modality can take on (one of 0 to n_0. Here n_0 = 8 is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

obtained from the rgb image by computing absolute value of the orientation of the gradients at every pixel. Locations with 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.

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 neighbourhood of size T centered 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 neighbourhood to the most similar feature in the input image. This formulation is therefore invariant to small changes in the input patch.

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 by quickly rejecting most parts of the image. 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, a downsampled image is first filtered with lower-resolution templates and only positions that score above the threshold τ are further investigated as potential positives using the full-resolution templates.

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.

Concretely, 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 $P(O_n, x|\mathcal{C}_i)$ is learned from 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.

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 sparse, repeatable locations on the object. Specifically, we choose to use Harris corner keypoints for this purpose.

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 a threshold.

This method has several attractive properties. Mainly, both learning and detection can be implemented very efficiently as they only require us to increment appropriate variables. All distance calculations can also be implemented efficiently using Approximate k-Nearest Neighbour techniques such as kd-trees. Efficient implementations exist [9].

4. Dataset Collection

The data used in our experiments are collected using the Microsoft Kinect sensor. Depth holes in the raw Kinect depth map

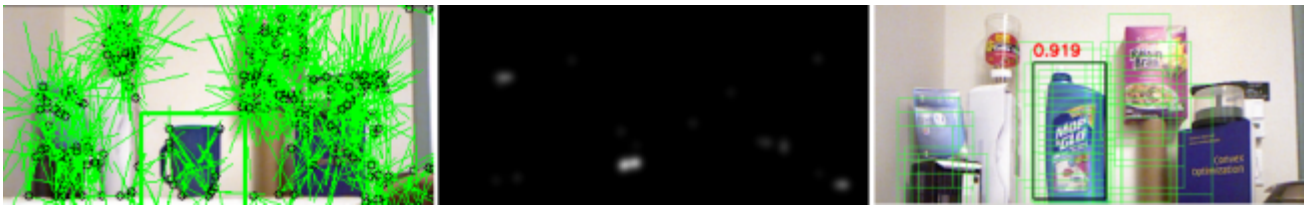

Figure 2. **Left**: Visualization of Hough voting method. Every black circle is a weakly-detected part, and green lines indicate votes. The green lines meeting at the center of the tea cup give rise to a noticeable peak in the response map for that object (**Center**). **Right**: Visualization of the 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.

Figure 3. **Left**: Example of an image from the turntable dataset. **Center**: Example image from the "in the wild" dataset. **Right**: The two bottles we attempt to detect. The blue bottle is partially occluded by a yellow marker in the test set.

are inpainted using algorithm described in [14].

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. We also place a number of distractor objects in the background, which can produce false positives for low detection thresholds.

Objects "in the wild" tests 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 placed two of the objects on the turntable in turn and moved the Kinect freely around the object to acquire many images from many possible views, scales, rotations, out of plane rotations, and tilts. On the order of 300 such color, depth, and mask images were obtained from this procedure for each object. Examples of acquired images are shown in 3. The precise mask segmentation was automatically computed by calibrating the camera using RANSAC with respect to a square calibration pattern placed on the turntable.

To collect the test set we placed both objects in a cluttered environment and partially occluded one with a marker. We moved 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 with 2 GB of RAM.

5.0.1 Turntable dataset

The images in the dataset are split randomly to 410 training and 143 testing images. 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 0.2 with 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 4. The root filtering approach can significantly improve the speed (6.3x, to about 2fps) if one is willing to sacrifice a small amount of recall. We explore this trade-off further below.

The Hough voting method achieves a lower performance, but runs relatively quickly (14.3x, about 4fps). Note that the method performs almost perfectly on some objects, and poorly on others. Concretely, 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. More gen-

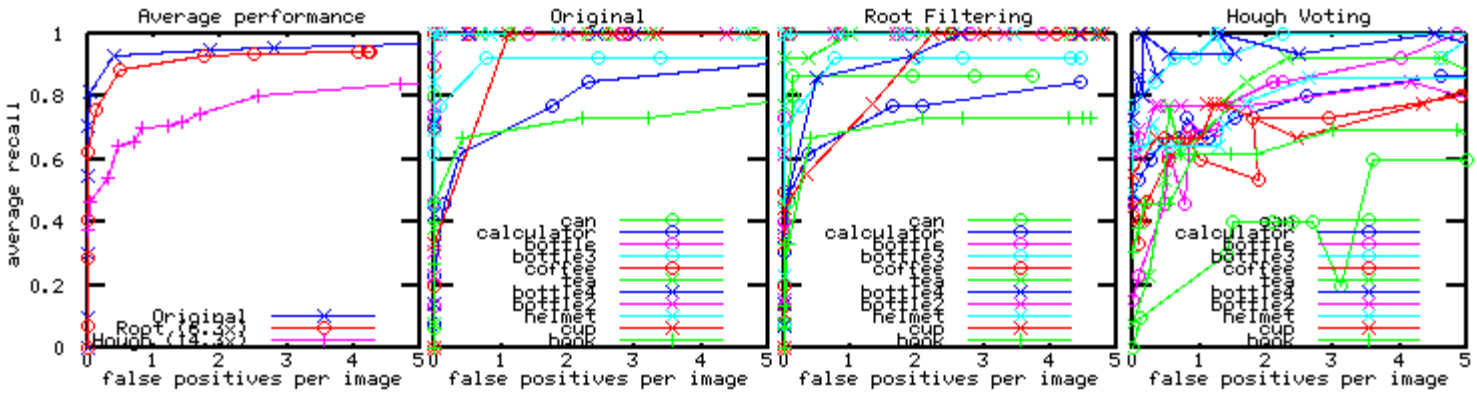

Figure 4. Detection results on the turntable dataset. The original runs at 0.3 frames per second, Root Filtering 6.3x faster, and Hough voting 14.3x faster. Left: average recall across classes. To the right: individual performances.

erally, from manual inspection it appears that this method performs poorly 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 5 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 downsample 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 two objects, and 200 testing images that contain them both in a cluttered scene. 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 5. The 3 scales of the image can be processed at about 2 frames per second per object. The blue bottle turns out to be 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 6.

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.

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 are applied. For instance, 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 response is too low.

References

[1] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Computer Vision, 2009 IEEE 12th*

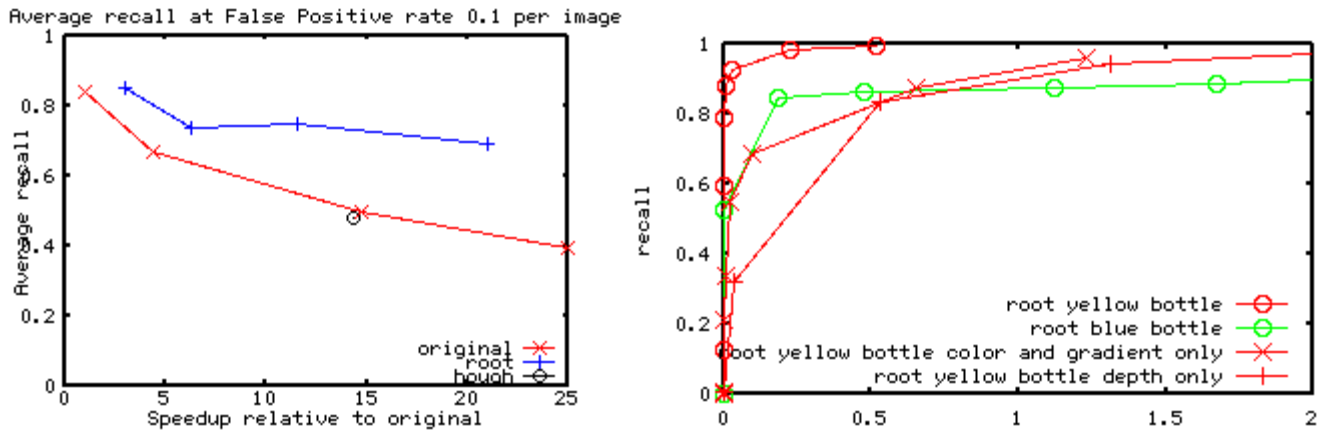

Figure 5. **Left:** Speed vs accuracy trade-offs. **Right**: Performance of the root filtering approach on "in the wild" dataset.

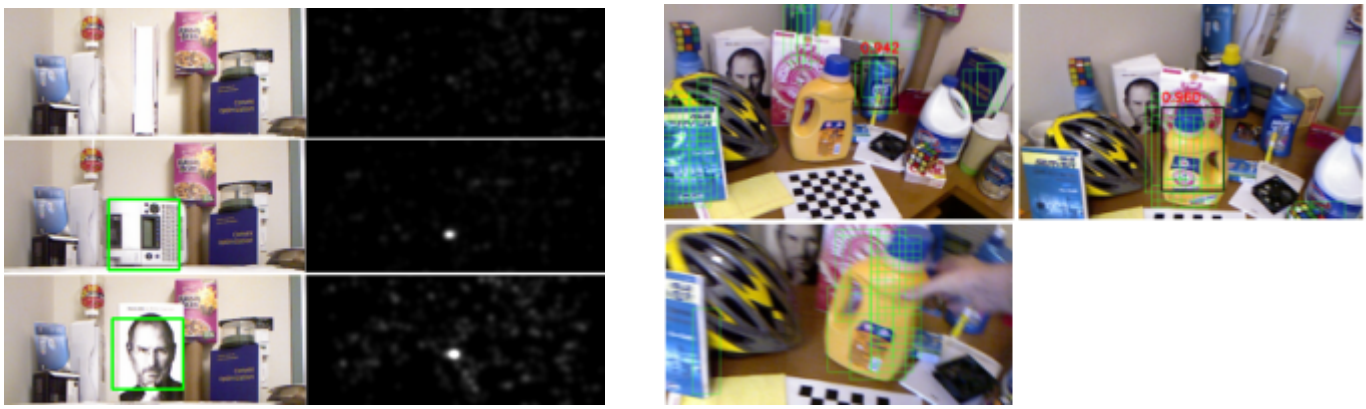

Figure 6. **Left:** Example detections using the Hough voting scheme. Top: a typical failure case, when not much of the object is visible. **Right:** Detections "in the wild". Proposed detections by root filter are shown in green. From left: blue bottle correctly detected, yellow bottle correctly detected, example of typical failure due to motion blur.

International Conference on, pages 1365–1372. IEEE, 2009.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

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