Review: Geometric Context from a Single Image

Samuel Teeter
wustl-key@wustl.edu

Abstract

In their 2005 article "Geometric Context from a Single Image," Derek Hoiem et al describe a system for inferring the coarse geometry of a scene based on a single camera image. In this paper I will analyze the usefulness of this task for applications such as object detection, describe the system itself in detail, and finally explore improvements and further applications for the algorithm.

1 Introduction

The objective of this algorithm is to break a single outdoor image into cohesive regions and classify each region as part of either the ground plane, the sky, or a vertical surface. Vertical surfaces are then subclassified by their orientation relative to the camera, and to their texture: they can be left-facing, right-facing, center-facing, solid, or porous. According to Hoiem, this task is meant to address a shortcoming in contemporary object detection systems: that they attempt to recognize targets based purely on small, self-contained regions of the image plane, ignoring potentially valuable information about the target's context. As Hoiem states, "Our ultimate goal is to recover a 3D 'contextual frame' of an image, a sort of theater stage representation containing major surfaces and their relationships to each other. Having such a representation would then allow each object to by physically 'placed' within the frame and permit reasoning between the different objects and their 3D environment." [3] Inferring the geometric class of each region in the image is intended as a first step towards this goal.

2 Background & Related Work

Hoiem cites an object detector developed by Murphy et al [5] as an example of an object detection system that benefits from geometric context data. Interestingly, this system also incorporates data from the entire image in order to recognize objects in localized regions. In brief, it combines the strategies of individual object recognition (based on small image regions), scene recognition (applying classes to the entire image), and joint object recognition (learning the likelihood of objects co-occurring in a scene) in order to improve its accuracy. However, as Hoiem points out, none of these methods considers the image as a representation of a 3D scene. In the Results section, we will examine the impact of including coarse geometric context as a feature in object recognition.

Current state-of-the-art object detection systems tend to rely on one or two stages of feature extraction followed by deep learning with convolutional neural networks. Jarrett et al give a good overview of these techniques in [4]. The feature extraction stages used in these systems do not rely on segmenting and classifying the whole image in the way that Hoiem proposes; rather, most apply filters such as oriented edge detection to small regions of the image. There is a certain philosophical similarity to Hoiem's work in that both methods involve integrating geometric features from multiple locales across the image; contemporary methods simply achieve this integration through the neural network itself rather than the cluster-like technique that Hoiem describes.

Finally, geometric classification remains an important component in contemporary systems for 3D scene reconstruction. I mention this because Hoiem's focus on noisy outdoor images suggests a particular focus on this task, and because we will see in the Results section that the geometric classes provided by the system can indeed be used to construct a crude "pop-up" 3D model of a scene. In [2], Christian Hane et al describe solving image segmentation and 3D scene reconstruction as joint problems. In a nutshell, part of their algorithm identifies cohesive surfaces within the image and identifies their orientation, and the other part attempts to coordinate all these surfaces within a 3D depth map. While the segmentation/orientation part of this algorithm works on a much finer scale than Hoiem's system, the overall task is very similar.

3 The Algorithm

3.1 Overview

The system begins by segmenting the image into superpixels using a graph-based method described by Felzenszwalb et al [1]. All further computation uses these superpixels as the finest units of the image. The superpixels are augmented with the responses of various filters re-

lated to color, texture, shape, location, edge detection, etc. See 3.2 for the full list. The algorithm then generates a large sample of regional segmentation hypotheses for the superpixels–that is, it creates groupings of the superpixels into 3, 4, 5, 7, 9, 15, 20, and 25 regions. Each region assigns a) a geometric classification, and b) a confidence measure to each superpixel. The final geometric classification hypothesis for each superpixel is generated by taking the weighted sum of the classifications from each region given their confidence measures.

3.2 Generating Segmentations

Since the main purpose of this review is to assess the effectiveness of the learning methods used for generating geometric classes, I will not go into detail about how the superpixels or their features are computed. Once these computations are complete, the superpixels are grouped into regions using a greedy algorithm. Its steps are to order the superpixels randomly, assign the first n superpixels to different regions, and then iteratively assign the remaining pixels to regions based on a learned pairwise affinity function. Given two superpixels, this function returns the log likelihood that they belong in a region together. During region segmentation, each pixel is assigned to the region with which it has the maximum average affinity. This last step is repeated several times. Note that this is not as computationally intensive as it sounds. At the beginning of segmentation, the algorithm simply computes a matrix of pairwise affinities for all superpixels in the image (there are about 550); it can then quickly calculate the overall likelihood of a superpixel being in any number of regions.

The pairwise affinity function was learned using AdaBoost with eight-node decision trees as weak learners. Given the absolute distance between two superpixels *in a single dimension of their features space*, it returns the probability of their being co-regional. The probabilities given each feature are then summed to produce the overall log likelihood, like so:

$$f_m(x_1, x_2) = \sum_i^{n_f} \log \frac{P(y_1 = y_2 | x_{1i} - x_{2i}|)}{P(y_1 \neq y_2 | x_{1i} - x_{2i})} \quad (1)$$

I found it very interesting that, in determining whether two superpixels belong in a region together, the algorithm only compares them one feature at a time. On the one hand, this solution is very elegant at reducing computational complexity; instead of having to apply a classifier to each pair of superpixels in turn, we can simply compute the differences in their features and find the corresponding likelihoods in a lookup table. On the other hand, this means that the two superpixels are never directly compared by the learner, meaning that the actual log likelihood estimation is a very naive one. While analyzing this part of the algorithm, I became intrigued by the possibility of comparing each superpixel with a "representative" superpixel for each cluster, and generating the regional segmentations using k-means, a concept I will explore further in the Experiments section.

Feature Descriptions	Num
Color	**16**
C1. RGB values: mean	3
C2. HSV values: C1 in HSV space	3
C3. Hue: histogram (5 bins) and entropy	6
C4. Saturation: histogram (3 bins) and entropy	4
Texture	**15**
T1. DOOG filters: mean abs response of 12 filters	12
T2. DOOG stats: mean of variables in T1	1
T3. DOOG stats: argmax of variables in T1	1
T4. DOOG stats: (max - median) of variables in T1	1
Location and Shape	**12**
L1. Location: normalized x and y, mean	2
L2. Location: norm. x and y, 10^{th} and 90^{th} pctl	4
L3. Location: norm. y wrt horizon, 10^{th}, 90^{th} pctl	2
L4. Shape: number of superpixels in region	1
L5. Shape: number of sides of convex hull	1
L6. Shape: $num\ pixels/area(convex\ hull)$	1
L7. Shape: whether the region is contiguous $\in \{0, 1\}$	1
3D Geometry	**35**
G1. Long Lines: total number in region	1
G2. Long Lines: % of nearly parallel pairs of lines	1
G3. Line Intsctn: hist. over 12 orientations, entropy	13
G4. Line Intsctn: % right of center	1
G5. Line Intsctn: % above center	1
G6. Line Intsctn: % far from center at 8 orientations	8
G7. Line Intsctn: % very far from center at 8 orient.	8
G8. Texture gradient: x and y "edginess" (T2) center	2

Table 1: Features computed on superpixels (C1-C2,T1-T4,L1) and on regions (all). The "Num" column gives the number of features in each set. The boosted decision tree classifier selects a discriminative subset of these features.

3.3 Geometric Labeling

For each regional segmentation hypothesis (i.e., number of regions=2), the regions are augmented with additional features computed over all their member superpixels (see 3.2). These features are then run through a second classifier (again, trained using Adaboost with 8-node decision trees) which returns a geometric classification for the region. Each superpixel is then assigned a final classification based on the weighted sum of the classifications from each regional hypothesis:

$$C(y_i = v|x) = \sum_j^n P(y_j = v|x, h_{ji}) P(h_{ji}|x) \quad (2)$$

where C is the confidence level for a geometric label with value v for superpixel x. h_{ji} are the hypotheses for each region that x might belong to, and $P(h_{ji}|X)$ is the confidence level for each hypothesis.

4 Experiments

While analyzing the system, I decided to try modifying the clustering technique used to generate the regions for different segmentation hypotheses. This was motived by two observations:

1. As Hoiem's results show, the features computed on regions are crucial to the algorithm's success. When Hoiem et al tried training a geometric classifier based solely on superpixel features, its performance dropped considerably. Likewise, when they tried feeding the algorithm the "correct" regions (that is, setting the regional segmentation hypotheses based on the ground truth labels), performance was dramatically improved. See the Results section for details.

2. As previously noted, the current regional segmentation hypotheses are based solely on the distance between superpixels in individual feature space dimensions, which severely limits the power of the region classifier. Furthermore, the regional classifications are not actually part of the algorithm's final hypothesis; rather, they are used to generate predictions and confidence measures for the geometric labels of superpixels, and then discarded. Given the importance of region-based features, I thought that the algorithm might perform better if it made more of an earnest attempt to find the "correct" regional segmentations.

4.1 k-Means Clustering with City Block Distances

My first experiment was to replace Hoiem's `sp2regions` function with a simple k-means clustering algorithm based on the city block distance between superpixels in feature space. I chose this distance measure because it is the same distance measure used in the original algorithm to compute the pairwise log likelihoods. After some experimentation, I decided to add weight to the location-based features of the superpixels, reasoning that spatially contiguous regions might be closer to the ground truth labels. Overall, my algorithm returned K clusters according to the function

$$L, C = \text{argmin}_{L,C} \sum_{k=1}^{K} \sum_{\mathbf{x}:L(\mathbf{x})=k} \mathbf{w}^T |\mathbf{x} - C_k| \quad (3)$$

where L is the set of labels for $\mathbf{x}$, C is the set of centroids, and $\mathbf{w}$ is a fixed set of weights for the features of $\mathbf{x}$.

4.2 k-Means With Pairwise Log Likelihood as a Distance Function

The goal of this experiment was to see whether the original pairwise log likelihood function would prove more effective as a distance measure. Note that this is fairly similar to the paper's original method, except that the original method computes an average affinity between a candidate superpixel and every superpixel in the cluster, whereas the k-means method computes the affinity only for the candidate superpixel and the cluster centroid. The question is therefore whether we can improve the computational complexity of this method without sacrificing accuracy.

5 Results

5.1 Results from the Original Algorithm

The original geometric classification system achieved 86% accuracy for vertical classes and 52% accuracy for horizontal subclasses on a sample of the images. Without repeating the entire results section from the original paper, I would like to call attention to a few interesting findings. One is the importance of the intermediate regional segmentation step. 1 shows the accuracy of the algorithm with varying degrees of structural estimation. OneH refers to one regional hypothesis, while MultiH refers to the multiple hypothesis model used by the system. As the figure shows, accuracy is significantly improved by the intermediate regional clustering step.

Secondly, the algorithm proved to be an effective component for both 3d reconstruction and object recognition systems. 2 shows the difference in accuracy achieved by a car detector with and without the confidence values generated by the geometric context system as input. The version of the detector trained with the context data shows considerable improvement over the original version. **??** shows a 3D "pop-up" image constructed by using the system's ground/vertical labels to draw edges around vertical surfaces. Note that because this image does not include any sort of depth estimation, it is not a 3D reconstruction per se; but it is easy to see how this technique could be combined with data from a binocular stereo rig or depth camera to create a coarse 3D model of an outdoor scene. These results show that while classifying surfaces geometrically may not be much of an application in itself, it can be an extremely useful precursor for other computer vision tasks.

5.2 k-Means with City Block Distance

In order to choose a weight for the location-based features of superpixels in my k-means algorithm, I wrote software

Intermediate Structure Estimation						
	CPrior	Loc	Pixel	SPixel	OneH	MultiH
Main	49%	66%	80%	83%	83%	86%
Sub	34%	36%	43%	45%	44%	52%

Figure 1: Classification accuracy for different levels of structure estimation. "Main" refers to the main vertical classes while "Sub" refers to the horizontal subclasses.

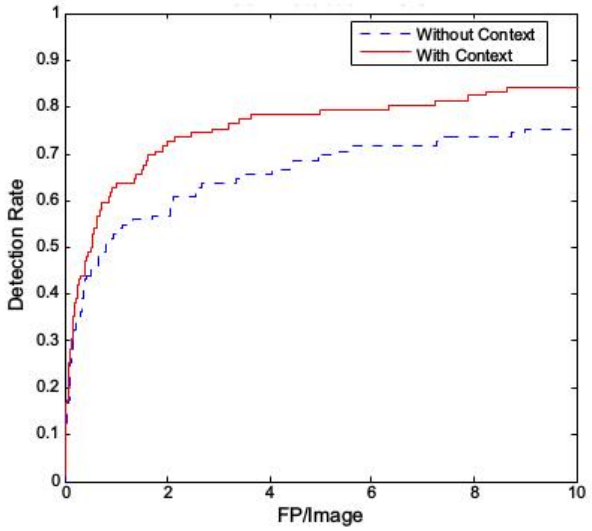

Figure 2: Accuracy of an object detection system with and without geometric context features.

Figure 3: 3D popup image constructed using geometric context labels.

Figure 4: The input image

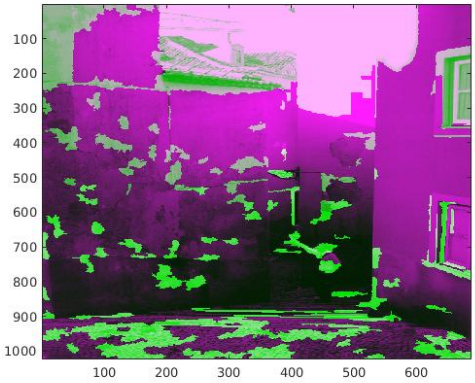

Figure 5: Regions returned by the original algorithm

to visualize the regions returned by the algorithm and increased the spatial weight until I got regions that were fairly contiguous. 6 and 5 show the 3 regions returned by k-means with a spatial weight of 100 and the regions generated by the original algorithm, respectively. 4 shows the original input image.

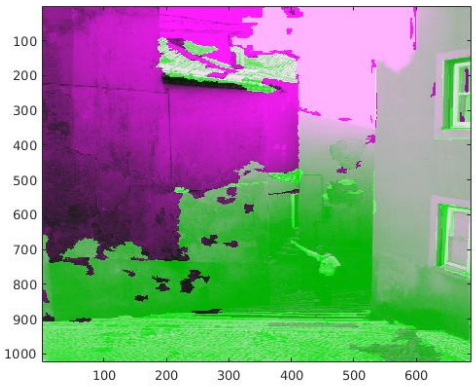

Figure 6: Regions from kmeans with spatial weight of 100

I then evaluated the geometric classes returned by each method on a small subset of the training data. This is by no means intended to be a comprehensive analysis of the algorithm's performance; given more time, I would have selected the best spatial weight after numerous rounds of cross-validation and evaluated the final hypothesis on a much larger test set. Nevertheless, these results do demonstrate that for noisy outdoor images, we can achieve comparable performance by taking a more rigorous approach to the regional segmentation hypotheses, without even having to train a learner. It is also worth noting that the time taken to compute these regions was noticeably less than the original algorithm.

Tables 3 and 4 show the confusion matrices for the vertical and horizontal classes using the k-means algorithm; figure 1 and 2 show the confusion matrices for the original algorithm. Note that in particular, the confusion between the ground and sky classes is markedly improved. Overall the accuracy for the control algorithm was 0.75 for vertical classes and 0.26 for horizontal subclasses. For the k-means algorithm it was 0.78 for vertical classes and 0.23 for horizontal subclasses. In the horizontal classes, the '–' class was an artifact of computation that was not explained by the paper or any of the comments in the code; I assume that it identifies superpixels for which no ground truth horizontal class exists. I therefore subtracted it from the total number of pixels when computing overall accuracy.

5.3 k-Means with Pairwise Affinity as a Distance Measure

I evaluated my k-means algorithm using the learned pairwise affinity function as a distance measure on the same subset of data as the previous experiment. 7 gives a visualization of the regions it generated for a 3-region hypothesis. 5 and 6 show the confusion matrices for vertical and horizontal classes for this test. Overall I found that it performed worse than the control, with an accuracy of .68 for the vertical classes and .11 for horizontal subclasses (again, ignoring unlabeled examples). Overall this is not surprising, since this version of the algorithm fits clusters essentially in the same way as the original, but with only one imaginary centroid pixel as a reference point, rather than the entire cluster. A more productive approach would be to train a learner that compared superpixels in all fea-

Control Vertical Classes			
	Ground	Wall	Sky
Ground	89	174	0
Wall	2	443	0
Sky	0	24	62

Table 1: Confusion matrix for vertical classes in the control algorithm

Control Horizontal Subclasses						
	Left	Center	Right	Porous	Solid	–
Left	41	25	0	12	11	0
Center	0	0	0	4	5	0
Right	0	201	36	7	10	0
Porous	0	0	0	34	2	0
Solid	0	14	0	37	6	0
–	0	190	0	56	103	0

Table 2: Confusion matrix for horizontal subclasses in the control algorithm.

k-Means Vertical Classes			
	Ground	Wall	Sky
Ground	135	128	0
Wall	5	437	3
Sky	0	36	50

Table 3: Confusion matrix for vertical classes in the k-means algorithm

k-Means Horizontal Subclasses						
	Left	Center	Right	Porous	Solid	–
Left	51	14	0	2	22	0
Center	0	0	0	5	4	0
Right	8	205	7	18	16	0
Porous	0	0	0	33	3	0
Solid	0	11	0	34	12	0
–	0	103	6	43	197	0

Table 4: Confusion matrix for horizontal subclasses in the k-means algorithm.

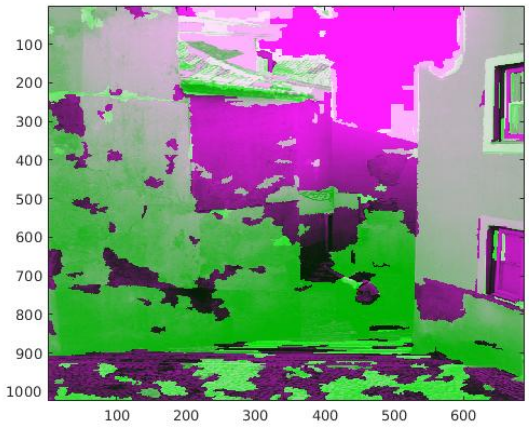

Figure 7: k-Means regions with pairwise affinity as distance measure

Density-based k-Means Vertical Classes			
	Ground	Wall	Sky
Ground	57	206	0
Wall	6	439	0
Sky	0	42	44

Table 5: Confusion matrix for vertical classes in the k-means density algorithm

Density-based k-Means Horizontal Subclasses						
	Left	Center	Right	Porous	Solid	–
Left	1	71	0	0	17	0
Center	0	0	0	5	4	0
Right	1	241	0	5	7	0
Porous	0	0	0	34	2	0
Solid	0	16	0	26	15	0
–	0	207	0	49	93	0

Table 6: Confusion matrix for horizontal subclasses in the k-means density algorithm.

ture dimensions at once, and use that as the distance measure; see Future Experiments for details.

5.4 Future Experiments

There were a couple of experiments that I wanted to try with this algorithm but was not able to complete, due to my limited time and access to labeled training data. One would be to train a learner that computes the pairwise affinity of superpixels by comparing them directly, rather than computing a naive density distribution based on the absolute distance in feature space. This would increase the power of the learner, and therefore the danger of overfitting, but I believe it could potentially return more accurate regional hypotheses than the current method.

The second idea I had would be to maximize the joint likelihood of the regional hypotheses and the geometric labels for each superpixel by treating the image as a random Markov field. Each superpixel would be a node with possible states corresponding to the regional hypotheses, and the geometric labels would be treated as outputs from each state. The pairwise log likelihood function could be used to establish a conditional probability for each state at each node based on its neighbors; the overall log likelihood of state assignments could then be maximized using a Markov random field optimization algorithm. This would allow the algorithm to optimize both the regional hypothesis and the geometric labelling hypothesis for the image in a more sophisticated way than computing one as a simple linear combination of the other.

Acknowledgments

The experiments in this paper were performed by modifying the publicly available implementation of this project at http://dhoiem.cs.illinois.edu/projects/context/

References

[1] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, Sept. 2004.

[2] C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys. Joint 3d scene reconstruction and class segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[3] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 654–661 Vol. 1, Oct 2005.

[4] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153, Sept 2009.

[5] K. P. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 1499–1506. MIT Press, 2004.