Multiple View Semantic Segmentation for Street View Images

Jianxiong Xiao Long Quan
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{csxjx, quan}@cse.ust.hk

Abstract

We propose a simple but powerful multi-view semantic segmentation framework for images captured by a camera mounted on a car driving along streets. In our approach, a pair-wise Markov Random Field (MRF) is laid out across multiple views. Both 2D and 3D features are extracted at a super-pixel level to train classifiers for the unary data terms of MRF. For smoothness terms, our approach makes use of color differences in the same image to identify accurate segmentation boundaries, and dense pixel-to-pixel correspondences to enforce consistency across different views. To speed up training and to improve the recognition quality, our approach adaptively selects the most similar training data for each scene from the label pool. Furthermore, we also propose a powerful approach within the same framework to enable large-scale labeling in both the 3D space and 2D images. We demonstrate our approach on more than 10,000 images from Google Maps Street View.

1. Introduction

Understanding the semantic content of images is a fundamental and challenging problem in computer vision. In this work, we are especially interested in simultaneously learning object class models and performing segmentation on multi-view images captured along streets in outdoor environments. This problem has many potential applications, such as to automatic vehicles in the DARPA Urban Challenge, extensions to earth maps, and city modeling [37, 30].

1.1. Related works

In recent years, many methods have been proposed for simultaneous single-view multi-class object classification and segmentation, such as [5, 6, 20, 27, 26, 28, 39, 29]. In our setting, we have multi-view images of the same scene to improve the performance.

For object recognition tasks, several multi-view systems have been proposed such as [31, 24, 16, 36, 13]. In these methods, either multi-view or 3D information is utilized during training. However, all of these methods focus on single view recognition during testing, while our problem is to recognize and segment multiple views during both training and testing.

[4] proposed a system making use of multi-view information during testing for instance-level retrieval. However, they focus on distributed systems in which the computation power and transmission bandwidth are limited. [38] proposed a joint affinity propagation method for both automatic segmentation and interactive refinement. Although multi-view information is used, this clustering-based approach cannot infer semantic meaning for object classes.

The work most related to this paper is probably [3], where Brostow *et al.* proposed to utilize 3D information for street image segmentation. Compared with [3], our approach differs in the following aspects. First, we propose a graph-based optimization approach to enforce consistency of the segmentation result across multiple views. Second, we adaptively select the training data from a large label pool. Furthermore, we propose a simple yet powerful approach for data labeling tasks, while [3] releases a high-quality labeled data set. Our approach works on practical data collected by Google Street View without human intervention. The data are very noisy and have strong glare. Posner *et al.* [19] also worked on similar problems although using range data.

1.2. Overview

We propose a multi-view semantic segmentation framework for images captured by a camera mounted on a car driving along the street. In Section 2, we illustrate how to set up the image capturing system. The pixel correspondences are then obtained across multiple views. Structure from Motion is used to reconstruct the scene geometry and prune incorrect correspondences. With both 2D and 3D information available, in Section 3, we lay out a Markov Random Field (MRF) across multiple images. Nodes in MRF represent superpixels from images, while edges represent smoothness across either neighboring superpixels in

2009 IEEE 12th International Conference on Computer Vision (ICCV)
978-1-4244-4419-9/09/$25.00 ©2009 IEEE

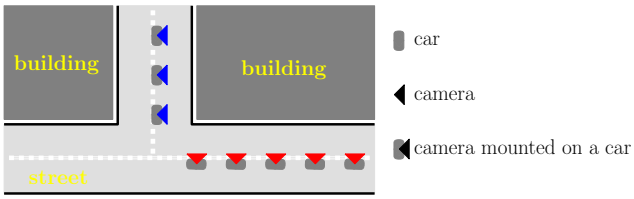


Figure 1. Top view of the camera motion. The car drives along the street and makes a 90-degree turn at the corner. Therefore, we break the sequence down into two different sequences, denoted in red and blue at the turn.

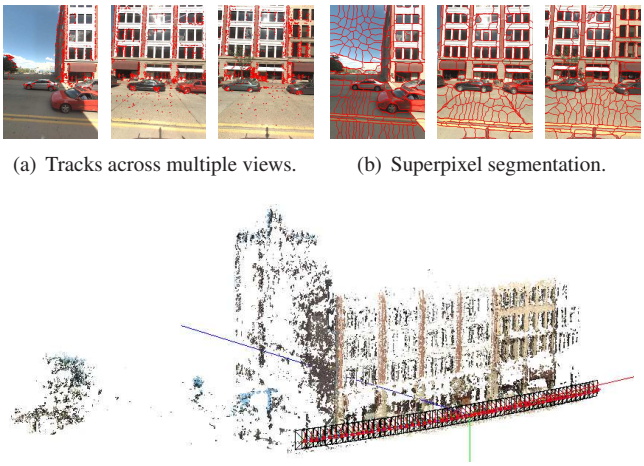
the same images or from different images linked by pixel correspondences. Section 3.1 gives the definition of the unary data term, while Section 3.2 defines the smoothness term. To improve performance by scene alignment, Section 3.3 illustrate the organization of the label pool to ease similar context and learning transference. In Section 4, we propose a approach to enable labeling of many images at the same time using the available geometry and color information. Finally, we demonstrate our approach in Section 5.

2. Preprocessing

We use a camera that usually faces the building façade and moves laterally along streets. The camera should be preferably held straight and the neighboring two views should have sufficient overlapping. The top view of the camera motion is illustrated in Figure 1. With the captured images, we first compute pixel-to-pixel correspondences between two adjacent images using a robust uncalibrated matching algorithm [15, 35]. Taking an image in sequence as a bridge, we can obtain feature tracks of three neighboring views for projective reconstruction. We merge all the triplets by estimating the transformation between those with two common images [38, 36] and metrically upgrade to Euclidian space. In each step, bundle adjustment is used to minimize the geometric errors, and feature tracks are merged and linked to cover more views.

Figure 2 shows an example of feature tracks across multiple view images and corresponding 3D reconstruction. We not only recover a set of 3D points representing the scene, but also all camera poses and parameters. We denote a feature track as $\mathbf{t} = \langle \mathbf{x}, (x_i, y_i, i), (x_j, y_j, j), \dots \rangle$, where $\mathbf{x} = (x, y, z)$ is the coordinate for the corresponding 3D point, and (x_i, y_i, i) is the 2D projection (x_i, y_i) on the i-th image, I_i.

We work on sequences with about 100 images and break down at the turn in the driving path as exemplified in Figure 1. To ease the description of the 3D geometry, the right-hand coordinate system is rotated to align with the average down vector of all reconstructed cameras to be in the y direction, and the camera path is roughly on the x axis, while



(a) Tracks across multiple views. (b) Superpixel segmentation.

(c) 3D reconstruction by Structure from Motion. The x, y and z axes are indicated in red, green and blue, respectively.

Figure 2. Preprocessing.

the orientations of the cameras are roughly the same as the $+z$ direction. To improve the segmentation accuracy and speed up the process, we over-segment [21, 17] each input image, I_i, into about 200 superpixels $\{p_j\}$.

3. Multi-view semantic segmentation

Since street view data usually contain multiple images, we define a Markov Random Field for the entire sequence to improve the segmentation consistency across different views. For each image, I_i, we build a graph, $\mathcal{G}_i = \langle \mathcal{V}_i, \mathcal{E}_i \rangle$, on the over-segmentation results. Each vertex, $p \in \mathcal{V}_i$, in the graph is one superpixel in the over-segmentation, while the edges, $\mathcal{E}_i$, denote the neighboring relationship between superpixels. The graphs $\{\mathcal{G}_i\}$ from multiple images in the same sequence are merged into a large graph, $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, by adding the edges between two superpixels in correspondence across different views. The superpixels, p_i and p_j, from images I_i and I_j are in correspondence if and only if there is at least one feature track, $\mathbf{t} = \langle \mathbf{x}, (x_i, y_i, i), (x_j, y_j, j), \dots \rangle$, with projection (x_i, y_i) lying inside superpixel p_i in image I_i, and projection (x_j, y_j) lying inside superpixel p_j in image I_j. To limit the graph size, there is at most only one edge, e_{ij}, between any superpixel, p_i and p_j, in the final graph, $\mathcal{G}$.

The labeling problem is to assign a unique label, l_i, to each node, $p_i \in \mathcal{V}$. The solution, $L = \{l_i\}$, can be obtained by minimizing a Gibbs energy [10]

$$E(L) = \sum_{p_i \in \mathcal{V}} \psi_i(l_i) + \rho \sum_{e_{ij} \in \mathcal{E}} \psi_{ij}(l_i, l_j). \quad (1)$$

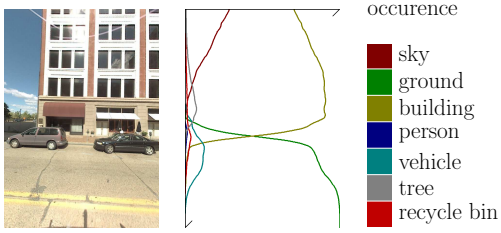
Since the smoothness costs defined in Section 3.2 satisfy the metric requirement, after the cost has been computed, GraphCut-based alpha expansion [2] can be used to obtain a local optimized label configuration, L.

3.1. Unary potential

To define the unary potential function, $\psi_i(\cdot)$, we extract features for superpixels to train discriminative classifiers.

3.1.1 2D features

For each superpixel, p_i, we compute a 192-dimensional feature description vector, $\mathbf{f}_i^A$, based on the 2D image-based appearance. Built on [1, 11, 34, 27], for each superpixel, p_i, the feature vector, $\mathbf{f}_i^A$, contains the median, deviation, skewness and kurtosis statistics over the superpixel, p_i, of the RGB and Lab color-space components, as well as the texture features drawn from filter bank responses. The filter bank we used is made of three Gaussians, four Laplacians of Gaussians and four first-order derivatives of Gaussians. This filter bank has been shown to achieve good performance in [34] among a number of different filter combinations of derivatives of Gaussians and Gabor kernels. In addition, following [1], we compute the size and shape of each superpixel. The shape features consist of the ratio of the region area to the perimeter square, the moment of inertia about the center of the mass, and the ratio of the area to the bounding rectangle area. As in [11], we also append to the description vector the average of the descriptor over the neighbors for each superpixel weighted by the number of pixels for the neighbors.



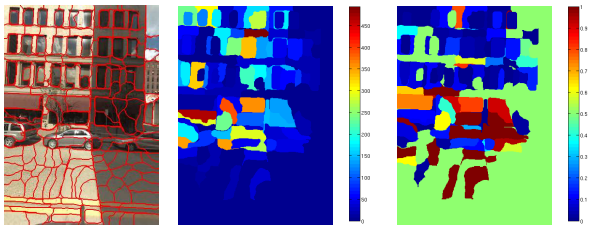
(a) Sample Image (b) Distribution

Figure 3. Pixel location statistics.

Because of the way we captured images, we can roughly learn the location for each class of objects. For example, the sky is always in the upper part of the image, while the ground is always in the lower part. Since our camera moves laterally along the street, each pixel position at the same height in the image space should have the same chance to be a specific class. To illustrate the idea, we compute the accumulated frequency of different classes from all labeled data and plot the distribution in Figure 3. Based on this observation, we only use the vertical position of the superpixel as the one-dimensional feature vector, $\mathbf{f}_i^P$.

3.1.2 3D features

We define the superpixel orientation and the 3D point density as our geometric features, $\mathbf{f}_i^G$. We do not use the absolute height above the camera and the absolute distance to the camera path as in [3], because an extra setup for the capturing system to measure the absolute dimensions is needed. We also do not use the back projection residual since it strongly depends on the implementation of Structure from Motion.



(a) Superpixel (b) Density (c) Direction

Figure 4. An example of 3D geometric features.

Let $\mathcal{T}_i$ denote all tracks that have projection in p_i, and let $\mathbf{m}_i$ be the medians of three components of all 3D points in $\mathcal{T}_i$. For each superpixel, p_i, the patch normal, $\mathbf{n}_i$, is provided by the symmetric 3×3 positive semidefinite matrix, $\sum_{\mathbf{x} \in \mathcal{T}_i} (\mathbf{x} - \mathbf{m}_i) \otimes (\mathbf{x} - \mathbf{m}_i)$. Among the eigenvectors, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$, associated with the eigenvalues, $\lambda_1 \geq \lambda_2 \geq \lambda_3$, respectively, we choose $\mathbf{n}_i$ to be either $\mathbf{v}_3$ or $-\mathbf{v}_3$. The sign is chosen to have a greater than 180-degree angle between $\mathbf{n}_i$ and the camera orientation. In the experiments, we only estimate the normal direction for regions containing at least five 3D points. The estimated normal direction, $\mathbf{n}_i$, is projected onto the yz-plane. The dot product of the normalized unit projected vector and the $-y$ direction is defined to be the orientation descriptor. For diluted regions without sufficient points for normal estimation, we let this feature value to be 0.5. This definition of the geometric features is very useful for classification between the ground with normals roughly the same as $-y$, and other objects such as buildings. For objects that are textureless, such as the sky, we use the density $|\mathcal{T}_i|$ of the feature tracks to distinguish them.

3.1.3 The boosting classifier

The collection of all feature descriptors are then whitened to give a zero mean and unit covariance. We learn a series of one-vs-all AdaBoost classifiers [25] for each class label l. Here, we take as positive examples the superpixels that belong to that class in the ground-truth labeling and as negative examples all superpixels with ground-truth labels of different classes. We apply the AdaBoost classifier that we have learned for each class, l, to the descriptors. The estimated confidence value can be reinterpreted as a probability distribution using softmax transformation:

$$P_i\left(l_i|\mathbf{f}_i^A, \mathbf{f}_i^P, \mathbf{f}_i^G\right) = \frac{\exp\left(H\left(l_i, \mathbf{f}_i^A, \mathbf{f}_i^P, \mathbf{f}_i^G\right)\right)}{\sum_l \exp\left(H\left(l, \mathbf{f}_i^A, \mathbf{f}_i^P, \mathbf{f}_i^G\right)\right)}, \quad (2)$$

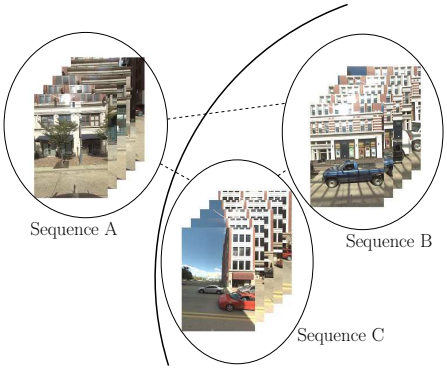


Figure 5. Affinity clustering in the label pool.

where $H\left(l, \mathbf{f}_i^A, \mathbf{f}_i^P, \mathbf{f}_i^G\right)$ is the output of the AdaBoost classifier for class l. We then define the unary potential as $\psi_i\left(l_i\right) = -\log P_i\left(l_i | \mathbf{f}_i^A, \mathbf{f}_i^P, \mathbf{f}_i^G\right)$.

3.2. Smoothness

For edge $e_{ij} \in \mathcal{E}_k$ in the same image, I_k, the smoothness cost is defined as

$$\psi_{ij}\left(l_i, l_j\right) = \left[l_i \neq l_j\right] \cdot g\left(i, j\right), \tag{3}$$

where

$$g\left(i, j\right) = \frac{1}{\zeta \left\|\mathbf{c}_i - \mathbf{c}_j\right\|^2 + 1} \tag{4}$$

and $\left\|\mathbf{c}_i - \mathbf{c}_j\right\|^2$ is the $L2$-Norm of the RGB color difference of two superpixels, p_i and p_j. Note that $\left[l_i \neq l_j\right]$ allows us to capture the gradient information only along the segmentation boundary. In other words, ψ_{ij} is a penalty term when adjacent nodes are assigned with different labels. The more similar the colors of the two nodes are, the larger ψ_{ij} is, and thus the less likely the edge is on the segmentation boundary.

For edge $e_{ij} \in \mathcal{E}$ across two images, the smoothness cost is defined as

$$\psi_{ij}\left(l_i, l_j\right) = \left[l_i \neq l_j\right] \cdot \lambda \left|\mathcal{T}_{ij}\right| g\left(i, j\right) \tag{5}$$

where $\mathcal{T}_{ij} = \left\{\mathbf{t} = \left\langle \mathbf{x}, \left(x_i, y_i, i\right), \left(x_j, y_j, j\right), \ldots \right\rangle\right\}$ is the set of all feature tracks with projection $\left(x_i, y_i\right)$ lying inside the superpixel, p_i, in image I_i, and projection $\left(x_j, y_j\right)$ lying inside the superpixel, p_j, in image I_j. This definition encourages two superpixels with more feature track connections to have the same semantic segmentation label, since the cost to have different labels is high due to large $\left|\mathcal{T}_{ij}\right|$.

3.3. Adaptive training

For each testing sequence, we only select a subset of labeled images that are similar with the input sequence as the training data for that sequence [7]. We define the distance between two images as the distance between their respective

(a) Strokes (b) Segment regions (c) Info source likelihood

Figure 6. Labeling in 2D. Color code: ■ sky, ■ vehicle.

Gist descriptors [18]. The Gist descriptor is used because it has been shown to work well for retrieving semantically and structurally similar scenes. We create a Gist descriptor for each image with 4 by 4 spatial resolution where each bin contains that image region's average response to Steerable filters at 4 scales with 8,8,4 and 4 orientations, respectively.

To speed up the training and prediction process, we cluster the labeled sequences in the pool based on affinity. As shown in Figure 5, we regard each label sequence as one node in a graph. The weight of the edge between each pair of sequences is defined to be the minimal Gist distance between any image in a sequence and any image in another sequence. Given this graph, we use Affinity Propagation [9] to cluster 40 labeled sequences into 7 clusters. We then learn 7 models respectively by first training the AdaBoost classifier in Section 3.1.3, and learning ρ in Equation 1, ζ in Equation 4, and λ in Equation 5 by piecewise training [27].

Given a testing sequence, we can compute all the distances between the labeled images and each image in the input sequence. We may define the distance between the testing sequence and one cluster as the minimal Gist distance between any image in the testing sequence and any image from the cluster. However, this process is very time consuming. Therefore, we approximate by using the middle image in each sequence. In this way, we only need to compute 40 distances between the Gist of the middle image in the testing sequence, and the pre-computed Gist descriptors for images at the middle positions of the 40 labeled sequences. We find the most similar cluster for the testing sequence and use the corresponding model for prediction.

4. Large-scale labeling

Labeling a sufficient number of examples is a fundamental requirement for any supervised learning method. Many researchers in this field have proposed methods to ease labeling tasks, such as LabelMe [23], ESP game [32] and Peekaboom [33]. However, most of these methods tried to make the labeling become interesting and online to encourage people to do more labeling, while the efficiency has not yet been greatly addressed, possibly due to the single image nature of these tasks. In our multiple view setting, it is possible to enable large-scale labeling with little interaction.

(a) Labeling at the front view. (b) Labeling at the top view.

Figure 7. Labeling in 3D. Color code: ☐ ground, ☐ building, ☐ vehicle, ☐ tree.

We first reconstruct the 3D scene for each sequence with about 100 images, and let the user label the 3D points in the 3D space. Using labels of 3D points, we are able to segment the 2D images at the same time. In this way, labeling once gives us about 100 labeled images. This significantly improves the efficiency.

In detail, after the 3D scene is reconstructed, as shown in Figure 7, the user can draw rectangular or polygonal regions to indicate the semantic meaning of the point clouds. Note that the user may not want to, or may not be able to, identify and label all 3D points. The task is to use these non-perfect labeled 3D points to segment multiple view images. The same framework that we proposed previously can be naturally used in our labeling task. In Equation 1, the same smoothness defined in Section 3.2 can also be used since it involves no information gained from training data. The unary potential is defined as follows.

First of all, a superpixel, p_i, has a set, $\mathcal{T}_i$, of 2D projections of 3D points inside the superpixel region. The more points in $\mathcal{T}_i$ that are labeled, the more confidence we gain about the label of the region. Therefore, we define

$$P_i^{3D}\left(l_i = l\right) \propto \left|\mathcal{T}^l \cap \mathcal{T}_i\right| + \frac{\left|\mathcal{T}^{\text{unknown}} \cap \mathcal{T}_i\right|}{n}, \qquad (6)$$

where $\mathcal{T}^l = \{\mathbf{t}|\mathbf{t}$ is a track labeled as class l by the user in 3D$\}$, $\mathcal{T}^{\text{unknown}}$ is the set of feature tracks that have no label information from the user, and n is the total number of all possible labels. This definition will put a uniform uncertainty on each class if there are unlabeled 3D points with projections in the superpixel region. However, it cannot characterize the projection density in each superpixel region. A superpixel region with more labeled projections should have more influence on the neighboring regions. The superpixel that has lower uncertainty should also contribute more. Therefore, we define the unary potential to be

$$\psi_i\left(l\right) = -\frac{\left|\mathcal{T}_i - \mathcal{T}^{\text{unknown}}\right| + \epsilon}{H\left(P_i\left(\cdot\right)\right) + \varepsilon} \log P_i\left(l\right), \qquad (7)$$

where $\left|\mathcal{T}_i - \mathcal{T}^{\text{unknown}}\right|$ is the number of labeled feature tracks with projections in superpixel p_i, $H\left(P_i\left(\cdot\right)\right)$ is the entropy of the distribution $P_i\left(\cdot\right)$, and ϵ and ε are two small positive values to avoid 0. $P_i\left(l\right)$ is set to be $P_i^{3D}\left(l\right)$.

Ideally, this method works well to identify regions with sufficient texture. However, for the classes lacking texture, such as the sky, where almost no 3D points are reconstructed, it is impossible to label them in 3D. Therefore, we also provide mechanism to draw strokes on one or more 2D images. When a superpixel in one image is covered by the strokes drawn by the user to be class l, the corresponding unary potential is set to $\psi_i\left(l_i = l\right) = -\infty$ and $\psi_i\left(l_i \neq l\right) = +\infty$. By adding these hard constraints, together with the definition of smoothness based on the color difference in Section 3.2, the labeling results can be obtained by MRF optimization.

In the same city block, the color distributions across instances of the same class are quite similar, while those across instances of different classes are always different. To draw as few strokes as possible, we want to make use of the strokes in one image, I_k, to segment the other images. To integrate this idea into our framework, for a superpixel, p_i, in an image, I_j, without strokes, we first distinguish whether the labeling information of p_i should come from 2D color or 3D points. Illustrated in Figure 6, we segment some regions in image I_k with the strokes and compute the color statistics of the regions, $\mathcal{R}$, belonging to the same set of classes for the strokes. The color distribution of all pixels in $\mathcal{R}$ is approximated by a Gaussian mixture model, $M_{\mathcal{R}}$, on the RGB color space. Furthermore, the color distribution of all pixels in $I_k - \mathcal{R}$ is approximated by another Gaussian mixture model $M_{I_k - \mathcal{R}}$. For a superpixel, p_i, in an image, I_j, without strokes, we determine the likelihood that the label information of p_i with mean color $\mathbf{c}_i$ comes from the 2D color from

$$P_i^{col}\left(\mathbf{c}_i\right) = \frac{P\left(\mathbf{c}_i|M_{\mathcal{R}}\right)}{P\left(\mathbf{c}_i|M_{\mathcal{R}}\right) + P\left(\mathbf{c}_i|M_{I_k - \mathcal{R}}\right)}. \qquad (8)$$

The probability is defined accordingly by

$$P_i\left(l\right) = P_i^{col}\left(\mathbf{c}_i\right) P_i^{2D}\left(l\right) + \left(1 - P_i^{col}\left(\mathbf{c}_i\right)\right) P_i^{3D}\left(l\right), \qquad (9)$$

where $P_i^{3D}\left(l\right)$ is defined in Equation 6, and $P_i^{2D}\left(l\right)$ is the color likelihood computed from Gaussian mixture model of pixel colors in the stroke-covered regions of image I_k that belong to class l. This definition is then used in Equation

	sky	ground	building	person	vehicle	tree	recycle bin
sky	**94.6**	-	5.1	-	-	-	0.1
ground	-	**97.6**	1.0	-	1.1	0.1	0.2
building	-	0.6	**98.6**	-	0.2	0.4	0.2
person	-	6.6	85.0	**8.3**	-	-	-
vehicle	-	15.5	3.8	1.6	**78.7**	0.2	0.2
tree	0.1	1.2	5.7	-	0.1	**92.8**	0.1
recycle bin	-	6.1	25.7	-	-	-	**68.1**

Table 1. Accuracy of our approach to the evaluation data set in percentage. This confusion matrix shows the pixel-wise recall accuracy for each class (rows) and is row-normalized to sum to 100%. Row labels indicate the true class and column labels the predicted class.

7. If multiple views are labeled with 2D strokes, we just put all the pixels from the multiple views together to define $M_{\mathcal{R}}$ and $M_{I-\mathcal{R}}$ accordingly.

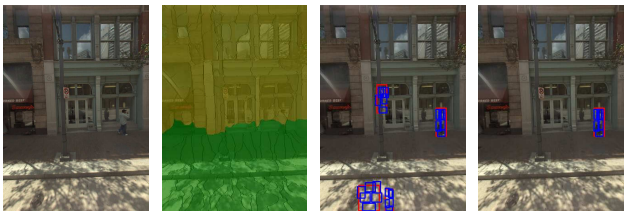
5. Experiment

In our experiment, we use more than 10,000 images captured in the downtown of Pittsburgh by Google Street View. We want to segment the images into seven classes: sky, ground, building, person, vehicle, tree and recycle bin. All of our input images are at 640×905 resolution. We train our models with a small computer cluster composed of seven desktop PCs. Each sequence contains about 200,000 3D points. It takes about 2 days for training using all labeled sequences excluding over-segmentation. For testing, each image takes 25.7 seconds on average in one desktop PC excluding over-segmentation.

We first manually label every image in one sequence. Then, we label this sequence in the 3D space with a few strokes in 2D images and learn ρ in Equation 1, ζ in Equation 4, and λ in Equation 5 for large-scale labeling by piecewise training [27]. Now, we use the large-scale labeling method to randomly label another 39 sequences. Together with the previous one, we have totally 40 labeled sequences and 3877 labeled images in the label pool.

5.1. Evaluation

For performance evaluation, we manually label 320 images sampled uniformly from the testing data set. The segmentation performance is measured as the global accuracy, i.e., the total proportion of pixels that are correct. The global pixel-wise prediction accuracy is 94.7% for the full model, 91.1% for the model from a random cluster (also below), 83.3% for the model without cross-view consistency, 81.2% for the model without smoothness, 75.4% for the model without 3D geometry features and smoothness, 69.9% for 2D appearance features together with the AdaBoost classifiers. Some example results predicted by the full model are shown in Figure 9. Note that there may be several images between two adjacent images not shown due to space limits. The confusion matrix of full model is presented in Table 1 for the pixel-wise recall accuracy per class.



(a) Input Image (b) Our result (c) Detection [8] (d) Pruning

Figure 8. An example of person detection. Color code: ■ ground, ■ building, □ person, □ parts of body.

5.2. Discussion

We can see that the accuracy for the sky, ground, building and tree classes are very high. However, for the person class, the results are unsatisfactory for several reasons. First, objects like persons are difficult to reconstruct by Structure from Motion, since they are dynamically deforming and unclear due to small pixel coverage, or there is a large disparity due to the close proximity to the camera. There are very few 3D points for these objects and it is difficult for the user to label them in 3D. Hence, very few of our labeled data contains good labels of these objects. Second, these small objects are not suitable for our super-pixel representation. The traditional sliding windows approach can better encode the profile shape for such small objects.

This paper tries to propose a consistent framework for multi-view segmentation. Although it is beyond the scope of this paper, we can easily incorporate existing detection methods to handle these difficult classes. For example, in Figure 8, we use a state-of-the-art model [8] to obtain a bounding box of the person. With our segmentation results for background objects, such as buildings and the ground, we can naturally put the boxes in perspective [12, 14] in order to prune incorrect predictions. For example, a person must stand on the ground and has a head usually at the pixel-position of the building. Finally, we can obtain the silhouette from the bounding box by methods such as GrabCut [22] or Active Contours.

6. Conclusion

We propose a multiple view framework for semantic object segmentation and demonstrate our approach on large-scale data sets from Google Street View images. Interesting future work will consider real-time implementation for prediction, better handling of small objects such as the illustration in Section 5.2, and extend the method to more general contexts beyond Street View.

Acknowledgements

This work was supported by Hong Kong RGC Grants 618908, 619107, 619006, and RGC/NSFC N-HKUST602/05. We thank Qiang Bi for labeling some data and the anonymous reviewers and the area chair for constructive comments that helped to improve this work. The data set was kindly provided by Google.

References

[1] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *JMLR*, 3:1107–1135, 2003.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[3] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2008.

[4] C. Christoudias, R. Urtasun, and T. Darrell. Unsupervised feature selection via distributed coding for multi-view object recognition. In *CVPR*, 2008.

[5] J. Corso, A. Yuille, and Z. Tu. Graph-shifts: Natural image labeling by dynamic hierarchical computing. In *CVPR*, 2008.

[6] G. Csurka and F. Perronnin. A simple high performance approach to semantic segmentation. In *BMVC*, 2008.

[7] S. K. Divvala, A. A. Efros, and M. Hebert. Can similar scenes help surface layout estimation? In *IEEE Workshop on Internet Vision at CVPR*. 2008.

[8] P. F. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[9] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[10] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *PAMI*, 6(6):721–741, 1984.

[11] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *IJCV*, 2008.

[12] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, 2006.

[13] A. P. IV, P. Mordohai, and K. Daniilidis. Object detection from large-scale 3d datasets using bottom-up and top-down descriptors. In *ECCV*, 2008.

[14] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool. Dynamic 3d scene analysis from a moving vehicle. In *CVPR*, June 2007.

[15] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *PAMI*, 27:418–433, 2005.

[16] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3D feature maps. In *CVPR*, 2008.

[17] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: combining segmentation and recognition. In *CVPR*, 2004.

[18] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Visual Perception, Progress in Brain Research*, 155, 2006.

[19] I. Posner, D. Schroeter, and P. Newman. Describing composite urban workspaces. In *ICRA*, 2007.

[20] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *ICCV*, 2007.

[21] X. Ren and J. Malik. Learning a classification model for segmentation. *ICCV*, 2003.

[22] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. *ACM TOG*, 23(3):309–314, 2004.

[23] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008.

[24] S. Savarese and L. Fei-Fei. 3D generic object categorization, localization and pose estimation. In *ICCV*, 2007.

[25] R. E. Schapire and Y. Singer. *Machine Learning*, (3):297–336, 1999.

[26] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[27] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1):2–23, 1 2009.

[28] G. Stephen, R. Jim, C. David, E. Gal, and K. Daphne. Multi-class segmentation with relative location prior. *IJCV*, 80(3):300–316, 2008.

[29] E. B. Sudderth and M. I. Jordan. Shared segmentation of natural scenes using dependent pitman-yor processes. In *NIPS*, 2008.

[30] P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan. Single image tree modeling. *ACM TOG*, 27(5):1–7, 2008.

[31] A. Thomas, V. Ferrari, B. Leibe, T. Turtelaars, B. Schiele, and L. V. Gool. Towards multi-view object class detection. In *CVPR*, 2006.

[32] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *ACM CHI*, pages 319–326, 2004.

[33] L. von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *ACM CHI*, pages 55–64, 2006.

[34] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005.

[35] J. Xiao, J. Chen, D.-Y. Yeung, and L. Quan. Learning two-view stereo matching. In *ECCV*, 2008.

[36] J. Xiao, J. Chen, D.-Y. Yeung, and L. Quan. Structuring visual words in 3d for arbitrary-view object localization. In *ECCV*, 2008.

[37] J. Xiao, T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan. Image-based façade modeling. *ACM TOG*, 27(5):1–10, 2008.

[38] J. Xiao, J. Wang, P. Tan, and L. Quan. Joint affinity propagation for multiple view segmentation. In *ICCV*, 2007.

[39] L. Zhu, Y. Chen, Y. Lin, and A. Yuille. A hierarchical image model for polynomial-time 2d parsing. In *NIPS*, 2008.

Figure 9. Example Results. Color code: sky, ground, building, person, vehicle, tree, recycle bin.