Integrating Low-level and Semantic Features for Object Consistent Segmentation

Hao Fu\textsuperscript{a,}\textsuperscript{*}, Guoping Qiu\textsuperscript{a}

\textsuperscript{a}School of Computer Science, University of Nottingham, Nottingham, UK

Abstract

The aim of semantic segmentation is to assign each pixel a semantic label. Numerous methods for semantic segmentation have been proposed in recent years and most of them chose pixel or superpixel as the processing primitives. However, as the information contained in a pixel or a superpixel is not discriminative enough, the outputs of these algorithms are usually not object consistent. To tackle this problem, we introduce the concept of object-like regions as a new and higher level processing primitive. We first experimentally showed that using groundtruth segments as processing primitives can boost semantic segmentation accuracy, and then proposed a novel method to produce regions that resemble the groundtruth regions, which we named them as ‘object-like regions’. We achieve this by integrating state of the art low-level segmentation algorithms with typical semantic segmentation algorithms through a novel semantic feature feedback mechanism. We present experimental results on the publicly available image understanding dataset MSRC21 and stanford background dataset, showing that the new method can achieve relatively good semantic segmentation results with far fewer processing primitives.

Keywords: Semantic segmentation, Object-like regions, Feedback mechanism

1. Introduction

Holistic image understanding is always the Holy Grail in computer vision. To achieve this goal, a natural method is to adopt a segment-then-recognize strategy, i.e. first segment an image into different objects, then recognize each object one by one. However, it is generally believed that segmentation is an ill posed problem. Taking the image in Fig.1 as an example, the head of the sheep (black color) and the body of the sheep (white color) are totally different from each other. But still they belong to one object. Therefore, we could not expect our segmentation module which are purely based on low-level features can produce semantic consistent regions.

To circumvent this problem, recent methods tend to bypass the segmentation module. On one hand, sliding window based object recognition paradigm has achieved remarkable success in some specific areas, like face detection [1]; on the other hand, pixel or superpixel based semantic segmentation methods [2, 3, 4] has become the mainstream in holistic image understanding.

Are we gradually losing faith in image segmentation? The answer is of course not. In [5], the authors experimentally confirmed that segment based recognition can achieve a higher accuracy than sliding window based recognition; although one segmentation is always prone to make mistakes, [6] used multiple segmentations.

Although the use of multiple segmentations increases our chance of finding object-like regions, it’s still a low-level segmentation. A better model can be expect-
ed if we could incorporate top down information into the low-level segmentation module. We named this kind of segmentation as supervised segmentation. Taking the image in Fig.1 as an example, how can the segmentation algorithm consider the head of the sheep and the body of the sheep to be one object? The only way is to incorporate supervised information into the segmentation module. As in our training phase, probably we have already seen a sheep with black head and white body, thus it is possible to utilize these training images to guide our segmentation algorithm. Therefore, the key problem is how to train our segmentation module to achieve this.

In this paper, we proposed a method that treats the output of typical semantic segmentation algorithm as top down information to guide the bottom up unsupervised segmentation. More precisely, we treat the output of typical semantic segmentation algorithm as semantic features. These semantic features are combined with low-level features, which together determine the similarity between neighboring nodes. In this case, although the head of the sheep and the body of the sheep differs a lot in their low-level feature, they may share similar semantic features, thus making it possible to consider these two parts belonging to one object.

In summary, we made two contributions in this paper. Firstly, we emphasized the importance of choosing regions as processing primitives in image understanding. Secondly, we propose to combine state of the art low-level segmentation with typical semantic segmentation algorithm to produce regions that resemble the groundtruth object regions, which we called them as ‘object-like regions’. Based on these ‘object-like regions’, we can achieve relatively good semantic segmentation results on public image understanding dataset.

2. Related work

The problem we are interested here is holistic image understanding, which means given an image, our task is to assign a semantic label to every pixel of the image. There exists a large literature in this domain. A common aspect shared by most of the existing algorithms is that they usually choose pixel or superpixel [2, 9, 4] as their processing primitives. However, one direct consequence of taking such strategy is that the output of the algorithm is usually not object consistent (as shown in Fig.1), and each object is just a group of pixels or superpixels with the same semantic labels.

One notable exception beyond this theme is the work in [10], where the authors directly chose the region as their processing primitives, and they achieved the best segmentation accuracy on the recent PASCAL challenge [11]. Compared with choosing pixel or superpixel as processing primitives, choosing region enjoys many benefits: the information contained in a region is much more comprehensive than that contained in a superpixel, and the shape of the region is beneficial for recognizing some shape dominate objects.

Although the strategy of choosing region as primitives is appealing, it is very difficult to segment images to semantic consistent regions in practice. [10] circumvents this problem by generating multiple figure-ground hypothesis. However, the segmentation module they adopted is still low-level cue based. Although they performed a supervised ranking after the hypothesis generation, the errors that occurred in the low-level segmentation module will not be remedied. After all, if we want to achieve a semantic consistent segmentation, we need to incorporate information beyond the image itself. While the authors in [12] used a shape prior to guide the segmentation, classcut [13] or co-segmentation [14] utilize information from other images. Another interesting work in [15] retrieves similar images from the internet, and these retrieved images are treated as additional information to guide the segmentation.

Different from all those previous works, in this paper we aim to obtain the additional information from the training data. In a typical low-level spectral segmentation algorithm, the core module is to define the affinity weights between neighboring nodes. In order to make the low-level segmentation module produce semantic consistent regions, we had to incorporate high-level knowledge in defining these weights. As we aim to obtain these high-level knowledge from the training data, we find typical semantic segmentation algorithms exactly to meet our needs: their posterior outputs can just define our semantic likeliness between neighboring nodes.

In essence, the semantic segmentation algorithm is still based on low-level cues, our algorithm can be considered as having introduced a feedback from the classifier output to the input level. We believe this idea is an important contribution of our work. As we have also noticed similar ideas being adopted in some other scenarios, we believe it is an important philosophy and could be generally adopted in many scenarios.

In [16], the outputs of various object detectors are treated as new mid-level features; in [17], the authors learned a bank of weak classifiers from the web-retrieved images, and the output of these weak learners are also treated as mid-level features. In this theme, the output of semantic segmentation algorithm used in our
low-level segmentation module can also be considered as mid-level features. In [18], the authors first learned a classifier on the local patch, then the output of the classifier is considered as the context information of the target patch. By concatenating them together, the author retrained the system. A similar strategy is also adopted in [19], where the authors use the geometric context detector to detect the layout of similar images, then the average of those outputs are considered as a prior, based on which the author retrained the system. From those previous works, we came to a conclusion that the output of classifier contains useful information. They can enhance the performance of the original system through a carefully design. This also justifies the applicability of the framework we proposed in the next section.

3. Combining low-level and semantic segmentation for object proposal

3.1. The advantage of choosing regions as processing primitives

Before we introduce our newly proposed framework, we would like to first emphasize the importance and advantage of choosing regions as processing primitives. We choose MSRC21 as our test bed. MSRC21 contains 591 images from 21 semantic classes. Following [4], the dataset is divided into 276 images for training, 59 images for validation, and the remaining 256 images for testing. For the object regions, we directly use the clean groundtruth segmentations\(^1\) on this dataset. These groundtruth segmentations are directly fed into our region labeling module to be described in section 3.6. We obtain an overall global pixel accuracy of 90.5%, outperforming any state-of-art methods [2, 3] on this dataset. This experiment clearly shows the advantage of choosing the regions as our processing primitives. However, the challenge is how to automatically generate such object regions. We present such a method in the following subsections: section 3.2 will present a general picture of the whole framework, then several of its important modules will be introduced in the subsequent subsections.

3.2. The proposed framework

Given an input image, several kinds of low-level features can be extracted, and these features are fed into a low-level spectral segmentation module. These low-level segmentation algorithms can produce low-level consistent regions, which we call them superpixel here. A typical semantic segmentation algorithm can either choose these superpixel or directly choose the image pixel as the processing primitives, and their outputs can be considered as semantic features.

Those semantic features are then combined with the low-level features and fed back to the low-level segmentation algorithm again. This time, as the features contain semantic feature, we call the segmentation module mid-level segmentation. Ideally, they will segment images into regions that are both low-level and semantic-level consistent, i.e. they are object-like regions. Based on those object-like regions, typical classification algorithms can be used to achieve the goal of image understanding. The flow chart of this procedure is shown in Fig.2.

3.3. Spectral methods for low-level segmentation

For the low-level spectral segmentation module, we adopt the multi-layer graph method proposed in [8], as it produces the state of the art results on BSDS dataset [20]. A multi-layer graph is represented by \(G^* = (V^*, E^*)\), where the nodes \(V^*\) are a set of pixels and superpixels, and edges \(E^*\) exist between neighboring pixels, neighboring superpixels, and also between the superpixel and all the pixels it includes. The edge weights are defined as:

\[
 w_{ij} = \begin{cases} 
 \exp(-\theta_g \parallel g_i - g_j \parallel) & \text{if } i, j \in \text{pixels} \\
 \exp(-\theta_g \parallel \bar{g}_i - \bar{g}_j \parallel) & \text{if } i, j \in \text{superpixels} \\
 \text{const} & \text{otherwise}
\end{cases}
\]  
(1)

where \(g_i\) is the color value (in Lab space) of pixel \(i\), and \(\bar{g}_i\) represents the mean color of all the inner pixels contained in a superpixel \(i\). \(\theta_e\) and \(\theta_g\) are constants that control the strengths of the weight. They can be specified either manually or using cross-validation techniques. For details of this algorithm, please refer to [8].

3.4. Pixel or superpixel based semantic segmentation

For the semantic segmentation module, we adopt the typical two order Conditional Random Field (CRF) [7] based methods. Mathematically speaking, a CRF defines a posterior distribution over hidden random variables \(Y\) (labels), given observed image features \(X\), in a factored form:

\[
 p(Y|X) = \frac{1}{Z} \exp(-\sum_{c \in C} \psi_c(Y_c, X))
\]  
(2)

---

\(^1\)http://www.cs.cmu.edu/~tmalisie/projects/bmvc07/
where $Z$ is a normalizing constant, and $C$ is the set of all cliques. When the size of the clique $c$ is one, $\psi_c(Y_c, X)$ corresponds to the node potential. Accordingly, when its size is two, $\psi_c(Y_c, X)$ corresponds to the pairwise potential.

Given a set of images and its corresponding groundtruth labels, the training procedure of a CRF aims to make the energy of the groundtruth label assignment corresponds to the minimum of the energy function. After the model is trained, for a new test image, its most probable of labeling $Y^*$ is defined as

$$Y^* = \arg \max_{Y \in L} p(Y|X)$$

where $L$ corresponds to any kinds of possible labeling.

Besides the most probable labeling $Y^*$, we can also get the marginal posterior distribution $p(y_i)$ of any node $i$. In this paper, we decide to use this marginal distribution as our semantic feature. Compared with directly choosing the MAP assignment as the semantic feature, it clearly enjoys some benefits. For example, for two neighboring nodes, we can not only judge if they are most likely to belong to one specific semantic class, but also we could say whether they are all dissimilar with another semantic class.

In our experiments, for the pixel based semantic segmentation, we adopted the Textonboost method \cite{4} and used their publicly available code\(^2\); for superpixel based semantic segmentation, we adapted the code from the STAIR Vision Library\(^3\), which uses piecewise training method \cite{21} to train the CRF and uses max-product propagation \cite{22} for inference.

### 3.5 Mid-level segmentation with semantic feature feedback

We propose to introduce the semantic features generated by CRF into the above mentioned low-level spectral segmentation module, enabling the spectral segmentation algorithm to produce both low-level and semantically consistent regions.

One intuitive way of achieving this goal is to redefine the edge weight function of equation (1). Note that in equation (1), the weight between neighboring nodes only depends on their low-level features. By introducing our semantic segmentation module, we can also obtain the semantic feature of each node. Thus, we redefine the weight between neighboring nodes as a combination of their low-level feature similarity and their semantic feature similarity:

$$w_{ij} = \begin{cases} a \exp(-\theta_{g} \| g_i - g_j \|) + (1 - a) \exp(-\theta_{s} \| s_i - s_j \|) & \text{if } i, j \in \text{pixels} \\ a \exp(-\theta_{g} \| g_i - g_j \|) + (1 - a) \exp(-\theta_{s} \| s_i - s_j \|) & \text{if } i, j \in \text{superpixels} \\ \text{const} & \text{otherwise} \end{cases}$$

where $s_i$ corresponds to the semantic feature of node $i$, here it is equivalent to the marginal probability $p(y_i)$ of node $i$ in our CRF paradigm. $\alpha \in (0, 1)$ is a trade-off parameter between low-level feature and semantic feature. When it equals to 1, (4) degrades to (1); when it equals to 0, (4) will generate exactly the same regions as the semantic output, and our algorithm can be considered as a verification step after the CRF. We set $\alpha = 0.5$ in all our experiments to make the semantic features and low-level features contribute equally to the segmentation module. $\| \|$ represents the norm of the vector.

\(^2\)http://jamie.shotton.org/work/code.html
\(^3\)http://robotics.stanford.edu/~gould/svl/
3.6. Object-like primitives labeling

Having produced object-like regions using the spectral segmentation algorithm with semantic feature feedback, we now extract features of those object-like regions and build a classifier to label them. As the object-like regions are usually large and contain many pixels, we believe that their histogram features are more robust and discriminative. Similar to [24], several kinds of histogram features, including Texton Histogram, Color Histogram and pHOG Histogram are extracted from each groundtruth region among all the training images. All these histograms are obtained by vector-quantizing and pooling the corresponding features. We use $\chi^2$ kernel to measure the similarities among those histograms. These kernels can then be added together, and a plain SVM can be adopted as the classifier based on this combined kernel.

To enhance the performance of the classifier, we have adopted two additional techniques: the first one is to use the Multiple Kernel Learning approach\(^4\) to replace the plain SVM. MKL differs from plain SVM in that it assigns a different weight to each kernel, and the algorithm learns these weights together with the parameters of the final classifier simultaneously in a principled framework. Mathematically speaking, let $(x_i, y_i), i = 1, 2, ..., N$ be $N$ instances consisting of regions $x_i \in X$ and their semantic class labels $y_i \in Y$; let $f_m \in \mathbb{R}^{d_m}$, $m = 1, 2, ..., F$, where $d_m$ denotes the dimensionality of the $m$-th feature, represents the different kinds of histogram features. Then the task of Multiple Kernel Learning is to learn a linear combined kernel $k^*$:

$$
k^*(x, x') = \sum_{m=1}^{F} \beta_m k_m(x, x') \quad (5)$$

where the linear coefficients $\beta_m$ need to satisfy the constraints $\beta_m \geq 0$ and $\sum_{m=1}^{F} \beta_m = 1$. These coefficients are learned together with the parameters $\alpha$ and $b$ of an SVM. The final SVM objective function is [25]:

$$\min_{\alpha, b, \beta} \frac{1}{2} \sum_{m=1}^{F} \beta_m \alpha^T K_m \alpha + C \sum_{i=1}^{N} L(y_i, b + \sum_{m=1}^{F} \beta_m k_m(x_i^T \alpha))$$

s.t. $\sum_{m=1}^{F} \beta_m = 1, \quad \beta_m \geq 0, \quad m = 1, ..., F \quad (6)$

where $L(y, t) = \max(0, 1 - yt)$ is the Hinge loss.

The second technique we adopted to augment the training set. As we select the regions as processing primitives, and the number of the groundtruth regions contained in the training set is always very limited. For example, there are only about 10 regions for some semantic class in MSRC21 dataset. Therefore it is very easy for the classifier to get overfitted. This is in contrast to selecting superpixel or pixel as processing primitives, where we can obtain tens of thousands of training samples from the training set. Therefore, it is necessary and beneficial if we can augment the training set. We achieved this through the following procedure: we make the low-level segmentation module to perform on the training images. If they generate some pure regions according to the groundtruth semantic labeling, then these regions are also considered as positive training samples and augmented the training set.

This strategy is in the similar spirit with the ‘mining hard negatives’ technique widely used in the object detection literature [26]. Both methods advocate reevaluating the algorithm on the training set to obtain more training samples. Note that in our scenario, we are not limited to any specific kinds of segmentation algorithms. Instead, we can use any kind of low-level segmentation algorithm, and we can even utilize our full model to perform on the training set. As long as these segmentation algorithms generate some pure regions according to the groundtruth semantic labeling, we can augment these regions to the training set.

We will show in the experiment section that both of these two techniques will help to boost the performance.

4. Experiments

4.1. Segmentation quality

To assess the quality of the regions generated by our model, we adopted the segmentation covering [27] as the accuracy measure. The covering of a segmentation $S$ by a segmentation $S'$ is defined as

$$C(S, S') = \frac{1}{N} \sum_{R \in S} |R| \cdot \max_{R' \in S'} O(R, R')$$

where $N$ denotes the total number of pixels in the image, $|R|$ represents the number of pixels in the region $R$, and $O$ is the overlap.

For each test image, we firstly segmented it into a fixed number (3 to 12) of regions using different segmentation algorithms. Then we compared these segments with the groundtruth segmentation to compute the segmentation covering score. Here we compared Normalized Cut [28] and Full Pairwise Affinity Model in [8] with our full model. The results are shown in

\(^4\)Downloadable from http://www.di.ens.fr/~obozinski/SKMsmo.tar
Figure 3: Segmentation covering score of different methods. X axis represents dividing an image into # regions, while Y axis corresponds to the accuracy. NCut means regions generated by normalized cut, FPA means Full Pairwise Affinity model in [8].

Fig.3. From there we can see our full model consistently performs better than the other two algorithms.

Furthermore, as we don’t know how many regions we should partition an image into, we consider all these generated segments as a segmentation pool, and recompute the segmentation covering score. The results are shown in Table.1. Again, we can see an obvious advantage of our full model over its counterparts. Besides, the results we achieved is very close to a state of the art method [27]. However, in [27], it generates a hierarchy of segments of each image. Usually it generates hundreds of segments, whilst our results are obtained based only on a total of 75 (which is the sum of 3+4+...+12) regions.

<table>
<thead>
<tr>
<th>method</th>
<th>NCut</th>
<th>FPA</th>
<th>Ours</th>
<th>gPb-owt-ucm [27]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover</td>
<td>0.60</td>
<td>0.73</td>
<td>0.77</td>
<td>0.78</td>
</tr>
</tbody>
</table>

tatistics of experimental results are shown in Fig.5 and Table.2. We can see that our full model consistently produced better results than other two methods, no matter we use the plain SVM classifier or the MKL classifier. Besides, we could see an improvement on MKL classifier over plain SVM for all the three segmentation methods. This suggests that the usage of an advanced classifier can help to improve the performance.

Compared with most state of the art results ([2]: 77%, [3]: 76.4%) on this dataset, we are getting very close. Although the hierarchical CRF model of [9] demonstrates superior performance: 86%, it should be noted that their pixel-wise classifier can obtain an overall accuracy of 81%, which suggests their use of much more discriminative features [30]. However, the main difference between our algorithm and theirs lies in that their algorithms are superpixel based whilst ours is object-like region based. For a typical image in MSRC21, it is usually segmented into hundreds of superpixels [3, 2]. However, in our case, we only need to divide the image into a few (3-12) regions and we achieved similar accuracy. It clearly shows the advantages of choosing object-like regions as processing primitives.

Fig.6 shows an illustrative example. It can be seen that whilst superpixel based methods divide the image into many small patches, our model divides it into only
Table 2: Semantic labeling accuracy on MSRC21 when each test image is partitioned into a fixed 8 regions. While ‘Global’ means the overall pixelwise labeling accuracy, ‘Average’ means the average of classwise accuracy. The results are shown in terms of percentages.

<table>
<thead>
<tr>
<th></th>
<th>building</th>
<th>grass</th>
<th>tree</th>
<th>cow</th>
<th>sheep</th>
<th>sky</th>
<th>aeroplane</th>
<th>water</th>
<th>face</th>
<th>car</th>
<th>bicycle</th>
<th>flower</th>
<th>sign</th>
<th>bird</th>
<th>book</th>
<th>chair</th>
<th>road</th>
<th>cat</th>
<th>dog</th>
<th>body</th>
<th>boat</th>
<th>Global</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4]</td>
<td>62</td>
<td>98</td>
<td>86</td>
<td>58</td>
<td>50</td>
<td>83</td>
<td>60</td>
<td>53</td>
<td>74</td>
<td>63</td>
<td>75</td>
<td>63</td>
<td>35</td>
<td>19</td>
<td>92</td>
<td>15</td>
<td>86</td>
<td>54</td>
<td>19</td>
<td>62</td>
<td>7</td>
<td>72</td>
<td>58</td>
</tr>
<tr>
<td>[29]</td>
<td>77</td>
<td>93</td>
<td>70</td>
<td>58</td>
<td>64</td>
<td>92</td>
<td>57</td>
<td>70</td>
<td>61</td>
<td>69</td>
<td>67</td>
<td>74</td>
<td>70</td>
<td>47</td>
<td>80</td>
<td>53</td>
<td>73</td>
<td>53</td>
<td>56</td>
<td>47</td>
<td>40</td>
<td>75</td>
<td>65</td>
</tr>
<tr>
<td>NCut</td>
<td>55</td>
<td>86</td>
<td>87</td>
<td>56</td>
<td>27</td>
<td>90</td>
<td>30</td>
<td>62</td>
<td>46</td>
<td>53</td>
<td>60</td>
<td>62</td>
<td>57</td>
<td>24</td>
<td>41</td>
<td>36</td>
<td>72</td>
<td>0</td>
<td>21</td>
<td>48</td>
<td>9</td>
<td>67</td>
<td>49</td>
</tr>
<tr>
<td>FPA</td>
<td>61</td>
<td>88</td>
<td>87</td>
<td>63</td>
<td>39</td>
<td>93</td>
<td>36</td>
<td>71</td>
<td>42</td>
<td>43</td>
<td>58</td>
<td>68</td>
<td>49</td>
<td>32</td>
<td>48</td>
<td>45</td>
<td>79</td>
<td>21</td>
<td>32</td>
<td>44</td>
<td>12</td>
<td>70</td>
<td>53</td>
</tr>
<tr>
<td>Ours</td>
<td>77</td>
<td>89</td>
<td>85</td>
<td>75</td>
<td>52</td>
<td>80</td>
<td>29</td>
<td>76</td>
<td>70</td>
<td>65</td>
<td>75</td>
<td>65</td>
<td>53</td>
<td>33</td>
<td>71</td>
<td>53</td>
<td>81</td>
<td>44</td>
<td>33</td>
<td>62</td>
<td>10</td>
<td>75</td>
<td>61</td>
</tr>
</tbody>
</table>

Figure 5: Global pixel accuracy on MSRC21

Figure 6: From left to right: original image; object-like regions generated by our full model; superpixels commonly used in conventional semantic segmentation methods

a few object like regions. Besides this, our algorithm inherently enjoys the advantage of producing object-consistent outputs as clearly illustrated in Fig.1.

4.3. Experiments on Stanford background dataset

To further verify the effectiveness of our approach, we also did experiments on Stanford background dataset [3]. This dataset consists of 715 images belonging to eight semantic classes including sky, tree, road, grass, water, building mountain and foreground object. The images in this dataset are collected from several well-known datasets including PASCAL, LabelMe, and Geometric Context. Each image in this dataset contains about 11 distinct object regions on average, while MSRC21 only contains about 3 object regions. Therefore, this dataset is much more challenging than MSRC21. Following previous work [3, 31] on this dataset, the overall 715 images are divided into 572 images for training and 143 images for testing.

Firstly, we also use the groundtruth segmentation to see how much accuracy we could achieve, which can be considered as the upper bound for the subsequent experiments. This time, we achieve a global pixel accuracy of 85.64%, which is again better than any state of the art methods. For example, [3] reported an accuracy of 76.4%, while [31] achieved 77.5%. [32] reported 79.4% which is the highest score on this dataset until the submission of this paper. All these three methods chose superpixel as the processing primitive and falls into the conventional CRF paradigm. This once again shows the advantage of choosing regions as processing primitives.

The following experiments follow exactly the same procedure as in MSRC21. The results we have achieved are shown in Fig.7. In that figure, we have also shown the effect of augmenting the training set strategy which was described in section 3.6. Here we have only augmented about 10000 regions for the whole dataset due to the space limitation caused by the kernel SVM, that is about 20 regions per training image. This is still much lower then conventional pixel or superpixel based methods, where the training set is usually as large as hundreds of thousands. From Fig.7, we can see that the strategy of augmentation do help improve the performance, and we can expect a further performance boost when augmenting more training samples.

4.4. Computational complexity analysis

From the above experiments, we could see that our algorithm exhibits a better performance than its counterparts: NCut and FPA. However, this improvement relies on its increased computational burden. For the segmentation algorithm itself, it shares the same complexity with FPA, as we have only redefined the weight function...
As shown in Equ.4. Therefore, the increased computational complexity arises from the computation of semantic features. Although most of the current semantic segmentation algorithm are time consuming, such as the method we adopted in this paper [4], there do exist some methods that can perform the semantic segmentation in real-time [33]. How to design a faster semantic segmentation algorithm is beyond the scope of this paper. It is important to point out that the work we presented in this paper is a general framework for combining semantic segmentation and low-level segmentation, thus it can be seamlessly combined with any kind of semantic segmentation algorithms.

5. Hard Integration

Besides the framework introduced above on fusing semantic segmentation with low-level segmentation, we have also tried another method to achieve this. That is to treat the semantic segmentation module as an unreliable ‘teacher’, which generates some tokens to guide the low-level segmentation module. Thus the whole system performs in an interactive segmentation fashion. We called this kind of integration hard integration, in comparison with the above framework which can be called soft integration.

The rationale behind the hard integration is that if the semantic segmentation module consistently considers a relatively large region belonging to one semantic class with high probability, then it has a high chance to be correct. Therefore, we can treat these large regions as highly reliable regions and generate tokens from it. The procedure of generating the tokens is shown in Fig.8. The low-level segmentation module then treat these tokens as ‘human’ input, and tries to segment images satisfying these token constraints. Here, we adopt the method introduced in [34] to perform the low-level interactive segmentation. Some visual examples are shown in Fig.9.

Figure 7: Semantic class accuracy on Stanford background dataset. ‘Ours+Augment’ represents results obtained by augmenting the training set.

Figure 8: The procedure for generating the tokens. Based on the outputs from the semantic segmentation algorithm and their predicted semantic classes, the highly probable large regions went through a series of morphological operations including gaussian filtering, threshold, thinning and dilation.

Figure 9: From left to right: original image; outputs from the semantic segmentation algorithm; the predicted semantic class; automatically generated tokens; segmentation results generated by [34] based on the tokens.

Although the idea of hard integration seems appealing, it can not perform on its own in practice. This is simply because not all images can generate reliable tokens. Besides this, the interactive segmentation algo-
rithm is still not mature. However, no matter it is hard integration or soft integration, we believe that there still exists a large room for improvement. Future works will try to design new frameworks to make the semantic segmentation module and low-level segmentation module really ‘interact’ with each other.

6. Concluding Remarks

In this paper, we first experimentally highlighted the importance of object-level image understanding. To produce the object-like regions, we then propose to unify the state-of-art low-level segmentation algorithms with typical semantic segmentation algorithms by introducing the semantic feature feedback. Experiments on MSRC21 and stanford background dataset confirmed the effectiveness of our new method.

Future works will develop better segmentation methods to produce object-consistent regions. Another problem is how to automatically decide the number of objects contained in an image. In the illustrative example of Fig.1, the appropriate number of regions should be 4. Dirichlet Process Mixture Model\(^5\) may be a promising tool to deal with this problem.

References


\(^5\)http://people.csail.mit.edu/jacobe/software/dpnn.tar.gz