A Segmentation-aware Object Detection Model with Occlusion Handling
Supplementary Material

Tianshi Gao1 Benjamin Packer2 Daphne Koller2
1 Department of Electrical Engineering, Stanford University
2 Department of Computer Science, Stanford University
{tianshig,bpacker,koller}@cs.stanford.edu

1. Introduction
In this supplementary material, we provide more detailed deriviations of the bound used in our efficient inference with branch-and-bound introduced in section 4.2 of the submitted paper. Furthermore, for readers who are not familiar with the branch-and-bound algorithm provide the detailed branch-and-bound procedure in a pseudo-code level as described in section 4.2 of the paper.

2. Derivation of the bound
The objective function for our inference is as follows:

\[ F(x_k, y; w) + \Delta(y_k, y) \]

where the definitions of different terms are described in section 4.1 in the paper. The inference task given a candidate space \( \mathcal{P} \times \mathcal{V} \) is:

\[ \arg\max_{(p, v) \in \mathcal{P} \times \mathcal{V}} F(x_k, y; w) + \Delta(y_k, y) \]  

We group the objective 1 into three parts: (i) terms that involve the visibility variables \( v \); (ii) the term for empty cell count, i.e., \( F^c(x_k, p) \); (iii) the term for the loss, i.e., \( F^{\text{loss}}(x_k, p, p) \).

Specifically, the first part is:

\[ F^v(x_k, p, v) = F^h(x_k, p, v) + F^h(x_k, p, v) + F^\text{prior}(v) + F^\text{pair}(v) \]  

In the branch-and-bound algorithm, we need an upper bound for the scores of all candidates \( (p, v) \) in a candidate space \( \mathcal{P} \times \mathcal{V} \), i.e., we want to find a \( U : \mathcal{P} \times \mathcal{V} \rightarrow \mathbb{R} \) such that

\[ \max_{(p, v) \in \mathcal{P} \times \mathcal{V}} F(x_k, y; w) + \Delta(y_k, y) \leq U(\mathcal{P} \times \mathcal{V}) \]  

As described in the paper, we derive the bound by pushing the maximization into the three components of Equation 1:

\[ \max_{(p, v) \in \mathcal{P} \times \mathcal{V}} F(x_k, y; w) + \Delta(y_k, y) \]

\[ = \max_{(p, v) \in \mathcal{P} \times \mathcal{V}} \left( F^v(x_k, p, v) + F^c(x_k, p) + F^{\text{loss}}(x_k, p, p) \right) \]

\[ \leq \max_{v} \left( \max_{p} F^h(x_k, p, v) + \max_{p} F^c(x_k, p) + F^\text{prior}(v) + F^\text{pair}(v) \right) \]  

(5)

We’ve derived the bounds for the second (the term corresponding to the empty cell count) and third (the term corresponding to the loss) components, i.e., \( U^c(\mathcal{P}) \) and \( U^{\text{loss}}(\mathcal{P}) \) as shown in equation (9) and equation (10) respectively in the paper. Here, we provide the derivation of the bound for the first part (terms involving \( v \)), i.e., \( U^v(\mathcal{P} \times \mathcal{V}) \) as shown in equation (8) in the paper. Specifically,

\[ \max_{(p, v) \in \mathcal{P} \times \mathcal{V}} F^v(x_k, p, v) \]

\[ = \max_{(p, v) \in \mathcal{P} \times \mathcal{V}} \left( F^h(x_k, p, v) + F^c(x_k, p) + F^\text{prior}(v) + F^\text{pair}(v) \right) \]

\[ \leq \max_{v} \left( \max_{p} F^h(x_k, p, v) + \max_{p} F^c(x_k, p) + F^\text{prior}(v) + F^\text{pair}(v) \right) \]  

(6)

Furthermore,

\[ \max_{p \in \mathcal{P}} F^h(x_k, p, v) = \max_{p \in \mathcal{P}} \sum_{i=1}^{wh} F^h_i(x_k, p, v_i) \]

\[ \leq \sum_{i=1}^{wh} \max_{p \in \mathcal{P}} \left( w^T_{h,i} \phi_{i}(x_k, p) \right) \cdot v_i \]  

(7)
Similarly,

\[
\max_{p \in \mathcal{P}} F^h(x_k, p, v) = \max_{p \in \mathcal{P}} \sum_{i=1}^{wh} F_i^h(x_k, p, v_i) \\
\leq \sum_{i=1}^{wh} \max_{p \in \mathcal{P}} \left( w_{h,i}^T \phi_i(x_k, p) \right) \cdot (1 - v_i)
\]

These two inequalities mean that for each cell, we use the best HOG response among all bounding boxes \( p \in \mathcal{P} \) with respect to the model. This maximization can be done very efficiently by caching the cell response. Combining Equation 7 and Equation 8 with \( F^{prior}(v) \) and \( F^{pair}(v) \), the right hand side of the inequality of Equation 6 becomes:

\[
U^v(\mathcal{P} \times \mathcal{V}) = \max_v \left( \sum_{i=1}^{wh} \max_p \left( w_{h,i}^T \phi_i(x_k, p) \right) \cdot v_i \right.
\]

\[
+ \sum_{i=1}^{wh} \max_p \left( w_{h,i}^T \phi_i(x_k, p) \right) \cdot (1 - v_i)
\]

\[
+ F^{prior}(v) + F^{pair}(v)
\]

Based on Equation 6, we have \( \max_{(p,v) \in \mathcal{P} \times \mathcal{V}} F^v(x_k, p, v) \leq U^v(\mathcal{P} \times \mathcal{V}) \). Therefore, \( U^v(\mathcal{P} \times \mathcal{V}) \) is the bound for the first component of the objective. Furthermore, to compute the bound, i.e., solving the MAP problem over \( v \), a single graph cut is sufficient. The cost of this computation is independent of \(|\mathcal{P}| \) (number of bounding boxes in \( \mathcal{P} \)).

Finally, combining the bounds for three parts of the objective, we get the bound for the scores of a candidate space \( \tilde{\mathcal{P}} \times \mathcal{V} \):

\[
\max_{(p,v) \in \tilde{\mathcal{P}} \times \mathcal{V}} F(x_k, y; w) + \Delta(y_k, y)
\]

\[
\leq U^v(\tilde{\mathcal{P}} \times \mathcal{V}) + U^v(\mathcal{P}) + U^{loss}(\mathcal{P})
\]

\[
def U(\tilde{\mathcal{P}} \times \mathcal{V})
\]

3. Efficient inference with branch-and-bound

For readers who are not familiar with the branch-and-bound procedure applied to window search [1], we elaborate the description of the algorithm (section 4.2 in the paper) by providing a high-level pseudo code description as shown Algorithm 1. For the parameterization of the space, a bounding box space \( \tilde{\mathcal{P}} \) is specified by a subspace of the image pyramids \([s_1, s_2] \times [w_1, w_2] \times [h_1, h_2] \) where \( s, w \) and \( h \) are the scale, width and height (normalized to \([0, 1]\)). The full image pyramid space is then defined as \([0, 1] \times [0, 1] \times [0, 1] \). In our implementation, when

Algorithm 1 Efficient Inference with Branch-and-Bound

\textbf{Require:} Image \( I \)

\textbf{Require:} Score bounding function \( U \) (Equation 10)

1: Initialize \( P \) as an empty priority queue

2: Initialize \( \mathcal{P} \) as the whole image pyramid space, i.e., \( \mathcal{P} = \{s_1, s_2\} \times [w_1, w_2] \times [h_1, h_2] = [0, 1] \times [0, 1] \times [0, 1] \)

3: \textbf{repeat}

4: split \( \mathcal{P} \rightarrow \mathcal{P}_1 \cup \mathcal{P}_2 \)

5: push \((U(\mathcal{P}_1 \times \mathcal{V}), \mathcal{P}_1)\) and \((U(\mathcal{P}_2 \times \mathcal{V}), \mathcal{P}_2)\) into \( P \)

6: retrieve the top state \( \mathcal{P}, \) i.e., \( U(\mathcal{P} \times \mathcal{V}) \) is the largest, from \( P \)

7: \textbf{until} \( \mathcal{P} \) consists of only one element, i.e., \( \mathcal{P} = \{p^*\} \)

\textbf{Return:} \( p^* \) and it’s corresponding \( v^* \)

References