A DETAILS OF SST-SWAPTREE ALGORITHM

Our tree construction algorithm begins with an arbitrary tree structure and seeks to iteratively improve it by making moves within the space of tree structures. We consider swaps, in which the subtree rooted at a node in the tree is exchanged with the subtree rooted at another node. Refer to Figure 4 for an illustration of a sample swap.

We now present the details of how we efficiently compute and score allowable swaps in order to decide which move to make. This procedure continues for a fixed number of iterations chosen based on validation performance.

A.1 COMPUTATION OF ALLOWABLE SWAPS

We consider only swaps that satisfy two criteria. The first is that a swap must not be between a node and an ancestor of that node. We do not consider such swaps because the subtree rooted at each of two nodes overlap and therefore the position of any shared descendants after the swap is ill-defined. For example, a swap between node B and node I in Figure 4 would not be allowable.

The second criterion is that the swap must preserve the property that the children of any nonterminal node in the tree are neighboring. We call this property the *neighboring-region* property. Informally, the neighboring-region property ensures that for two children of a parent node k to be siblings, at least one of the superpixels corresponding to the left sibling must be adjacent to one of the superpixels corresponding to the right. If this condition is satisfied, we say that Neigh$(C(\ell(k)), C(r(k)))$. More precisely, the neighboring-region property is satisfied by a tree if for every non-terminal node k in the tree, $N(C(\ell(k))) \cap C(r(k)) \neq \varnothing$, where $N(S) = \left(\bigcup_{s \in S} \text{Adj}(s)\right) \setminus S$ denotes the set of superpixels directly adjacent to the superpixels contained in set S but not including the elements of S itself, and Adj(s) denotes the set of superpixels directly adjacent to superpixel s. Note that the neighboring-region property is symmetric in that it could equally well be expressed as $N(C(r(k))) \cap C(\ell(k)) \neq \varnothing$ for each non-terminal node k.

Finding swaps that satisfy the first criterion is simple. We begin by iterating over each node i in the tree. For each node i, we can search for potential swap partners j by traversing the path from i to the root. For each node k along this path (excluding i itself), we search the the subtree rooted at the child of k not belonging to that path for potential swap partners j. For example, if B were being considered for a swap in Figure 4, the path to the root (excluding B) would be E $\rightarrow$ I $\rightarrow$ K. We would therefore search the subtrees rooted at A, F, and J for nodes that could be swapped with B.

Satisfying the second criterion relies on two observations that hold for all swaps between nodes i and j: (a) any node not on the path from i to j will continue to satisfy the neighboring-region property after the swap, and (b) the lowest common ancestor $k = LCA(i, j)$ will maintain the neighboring-region property after the swap. Observation (a) is due to the fact that the regions corresponding to these nodes do not change as a result of the swap. For example, in Figure 4, the path from node B to C is B $\rightarrow$ E $\rightarrow$ I $\rightarrow$ K $\rightarrow$ J $\rightarrow$ H $\rightarrow$ C. A swap between B and C will not affect the subtrees rooted at A, F, G, or D, and therefore we do not need to check the neighboring-region property for these nodes. In addition, if there were any nodes above K in the example tree, they would not need to be checked either since the region corresponding to K in the tree does not change as a result of the swap. Furthermore, the subtrees rooted at i and j themselves do not change as a result of the swap and thus do not need to be checked.

To show observation (b) holds, assume without loss of generality that i is a descendant of $\ell(k)$ and j is a descendant of $r(k)$ (the right child of k). We know that prior to the swap, Neigh$(C(\ell(k)), C(r(k)))$ is true. Thus Neigh$((C(\ell(k)) \setminus C(i)) \cup C(i), (C(r(k)) \setminus C(j)) \cup C(j))$ is also true. For k to satisfy the neighboring-region property after the swap, it must be the case that Neigh$((C(\ell(k)) \setminus C(i)) \cup C(j), (C(r(k)) \setminus C(j)) \cup C(i))$. If $\ell(k) \neq i$, then we know that Neigh$(C(\ell(k)) \setminus C(i), C(i))$, otherwise the subtree rooted at $\ell(k)$ would not have satisfied the neighboring-region property prior to the swap. Similarly, if $r(k) \neq j$, then we know that Neigh$(C(r(k)) \setminus C(j), C(j))$. If both $\ell(k) = i$ and $r(k) = j$, then the neighboring-region property is satisfied trivially after the swap, because otherwise Neigh$(C(\ell(k)), C(r(k)))$ would not be true prior to the swap.

Therefore we must only check the nodes along the path from i to j that are not either the $LCA(i, j)$ or i or j themselves. Let m be such a node along the path from i to $LCA(i, j)$. Without loss of generality, assume that i is a descendant of $\ell(m)$. Define the set of *critical superpixels* in i relative to a node h to be Crit$(i, h) = C(i) \cap N(C(h))$. For m to have satisfied the neighboring-region property prior to the swap, it must be the case that Crit$(\ell(m), r(m)) \neq \varnothing$. If Crit$(\ell(m), r(m)) =$ Crit$(i, r(m))$, i.e. all critical superpixels of $\ell(m)$ relative to $r(m)$ are also contained in $C(i)$, then we say i is *critical* relative to $r(m)$.

If i is critical relative to $r(m)$, we know m will not satisfy the neighboring-region property after a swap involving i unless the region of the replacement node j is neighboring to the region of $r(m)$. We can thus add a *critical constraint* on j due to m that $C(j) \cap N(r(m)) \neq \varnothing$. Any such constraints can be accumulated as we traverse the path from i to $LCA(i,j)$, and we must check that the candidate node j satisfies all constraints. Similarly we must check that i satisfies all critical constraints collected along the path from j to $LCA(i,j)$. If all such constraints are satisfied then the swap between i and j is allowable.

A.2 SWAP SCORING

In this section we outline the algorithm used to assign scores to potential swaps. We first examine the minimum achievable Hellinger distance for a given tree structure. Then we show how to compute the change in minimum achievable Hellinger distance resulting from a swap. Finally, we derive an approximation to the change in Hellinger distance that can be computed given only information known at test time.

A.2.1 Minimum Achievable Hellinger Distance

When iteratively making swaps, we seek to decrease Hellinger distance achievable by the tree. In other words, we wish to decrease

$$\begin{aligned}\min_{\mathbf{p}} D(\pi, q) = 1 - &\frac{1}{\binom{N}{2}} \sum_k \sum_{\substack{i \in C(\ell(k)) \\ j \in C(r(k))}} \left[\sqrt{\pi_k q_{ij}} + \sqrt{(1-\pi_k)(1-q_{ij})} \right] \\ &- \frac{1}{\binom{N}{2}} \sum_m \sum_{\substack{i,j \in C(m) \\ i<j}} \sqrt{q_{ij}}, \end{aligned} \tag{5}$$

where the summation involving k is over nonterminal nodes in the tree, the summation involving m is over leafs of the tree, $\mathbf{p}$ is a vector of probabilities (one for each nonterminal node), and $\pi_k = p_k \pi_{\ell(k)} \pi_{r(k)}$ is the marginal probability of node k.

When scoring swaps it is more convenient to consider minimization directly over $\boldsymbol{\pi}$ because the summations decompose over nodes:

$$1 - \frac{1}{\binom{N}{2}} \sum_m \sum_{\substack{i,j \in C(m) \\ i<j}} \sqrt{q_{ij}} - \sum_k \max_{\pi_k} \left[\sqrt{\pi_k} \left(\frac{1}{\binom{N}{2}} \sum_{\substack{i \in C(\ell(k)) \\ j \in C(r(k))}} \sqrt{q_{ij}} \right) + \sqrt{1-\pi_k} \left(\frac{1}{\binom{N}{2}} \sum_{\substack{i \in C(\ell(k)) \\ j \in C(r(k))}} \sqrt{1-q_{ij}} \right) \right] \tag{6}$$

$$= 1 - \frac{1}{\binom{N}{2}} \sum_m \sum_{\substack{i,j \in C(m) \\ i<j}} \sqrt{q_{ij}} - \sum_k \max_{\pi_k} \left[Q(\ell(k), r(k)) \sqrt{\pi_k} + \tilde{Q}(\ell(k), r(k)) \sqrt{1-\pi_k} \right], \tag{7}$$

where we define

$$Q(s_1, s_2) \equiv \frac{1}{\binom{N}{2}} \sum_{\substack{i \in C(s_1) \\ j \in C(s_2)}} \sqrt{q_{ij}}, \text{ and } \tilde{Q}(s_1, s_2) \equiv \frac{1}{\binom{N}{2}} \sum_{\substack{i \in C(s_1) \\ j \in C(s_2)}} \sqrt{1-q_{ij}}. \tag{8}$$

The value π_k^* that minimizes Equation 7 is

$$\pi_k^* = \frac{Q(\ell(k), r(k))^2}{Q(\ell(k), r(k))^2 + \tilde{Q}(\ell(k), r(k))^2}, \tag{9}$$

and the term of the summation involving k resulting from substituting in π_k^* becomes

$$\max_{\pi_k} \left[Q(\ell(k), r(k))\sqrt{\pi_k} + \tilde{Q}(\ell(k), r(k))\sqrt{1-\pi_k} \right] = \sqrt{Q(\ell(k), r(k))^2 + \tilde{Q}(\ell(k), r(k))^2} \tag{10}$$

$$= \|\boldsymbol{z}(\ell(k), r(k))\|_2, \tag{11}$$

where $\boldsymbol{z}(s_1, s_2) \in \mathbb{R}^2$ and is defined to be

$$\boldsymbol{z}(s_1, s_2) \equiv [Q(s_1, s_2), \tilde{Q}(s_1, s_2)]. \tag{12}$$

A.2.2 Effect of a Swap on Minimum Achievable Hellinger Distance

Most terms in the expression for $D(\pi, q)$ from Equation 5 will not change as a result of a swap of s_1 and s_2. The terms that do change are those corresponding to nodes along the path from s_1 to s_2, not including s_1 and s_2 themselves. Let k be a node along the path from s_1 to $LCA(s_1, s_2)$, not including the endpoints. Define $u(k)$ to be the child of k that is not an ancestor of s_1, and $g(k)$ to be the child of k that is an ancestor of s_1. Then the decrease in $D(\pi, q)$ due to k from making the swap is:

$$\Delta_k(s_1, s_2) = \|\boldsymbol{z}(g(k) \setminus s_1, u(k)) + \boldsymbol{z}(s_2, u(k))\| - \|\boldsymbol{z}(g(k) \setminus s_1, u(k)) + \boldsymbol{z}(s_1, u(k))\| \tag{13}$$

The change in $D(\pi, q)$ due to a node along the path from s_2 to $LCA(s_1, s_2)$ is similar to Equation 13 but with the roles of s_1 and s_2 reversed.

At the $LCA(s_1, s_2)$ the change is

$$\begin{aligned}\Delta_{LCA}(s_1, s_2) =& \|\boldsymbol{z}(g(LCA) \setminus s_1, h(LCA) \setminus s_2) + \boldsymbol{z}(s_1, s_2) + \boldsymbol{z}(g(LCA) \setminus s_1, s_1) + \boldsymbol{z}(s_2, h(LCA) \setminus s_2)\| - \\ & \|\boldsymbol{z}(g(LCA) \setminus s_1, h(LCA) \setminus s_2) + \boldsymbol{z}(s_1, s_2) + \boldsymbol{z}(g(LCA) \setminus s_1, s_2) + \boldsymbol{z}(s1, h(LCA) \setminus s_2)\|,\end{aligned} \tag{14}$$

where LCA is used as shorthand for $LCA(s_1, s_2)$. The total change in $D(\pi, q)$ is then

$$\sum_{k \in path(s_1, LCA(s_1, s_2))} \Delta_k(s_1, s_2) + \sum_{k \in path(s_2, LCA(s_1, s_2))} \Delta_k(s_2, s_1) + \Delta_{LCA}(s_1, s_2). \tag{15}$$

A.2.3 Approximating the Effect of a Swap

The quantity in Equation 15 depends on the values of q_{ij} for the relevant pixel pairs, which will be unknown at test time. We note that from Equations 9 and 12 that

$$\boldsymbol{z}(s_1, s_2) = \|\boldsymbol{z}(s_1, s_2)\|_2 [\sqrt{\pi_k^*}, \sqrt{1-\pi_k^*}]. \tag{16}$$

We choose to bound this based on the product of the size of the regions:

$$\|\boldsymbol{z}(s_1, s_2)\|_2 \leq \frac{|C(s_1)||C(s_2)|}{\binom{N}{2}}, \tag{17}$$

with equality iff either $q_{ij} = 1$ for all $i \in C(s_1), j \in C(s_2)$ or $q_{ij} = 0$ for all $i \in C(s_1), j \in C(s_2)$. If we have an estimate $\hat{\pi}(s_1, s_2)$ of any two regions s_1 and s_2, we can combine Equations 16 and 17 to obtain an estimate for $\boldsymbol{z}(s_1, s_2)$:

$$\hat{\boldsymbol{z}}(s_1, s_2) \approx \frac{|C(s_1)||C(s_2)|}{\binom{N}{2}}[\sqrt{\hat{\pi}(s_1, s_2)}, \sqrt{1 - \hat{\pi}(s_1, s_2)}]. \tag{18}$$

At test time, we can therefore use model predictions of $\boldsymbol{\pi}$ to estimate the change in Hellinger distance by plugging the approximate $\hat{\boldsymbol{z}}$ values from Equation 18 into the change in Hellinger distance from Equation 15.

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 ADDITIONAL DATASET DETAILS

We generated four synthetic ground truth segmentations per example in the Penn-Fudan pedestrian dataset, each of which represents a different prototypical parse of the image. The original semantic segmentation labels contained thirteen classes: background, hair, face, upper and lower clothes, and left and right arms, hands, legs, and shoes. The first synthetic ground truth treats each of these semantic classes as a separate region. The second merges hair with face, hand with corresponding arm, and shoe with corresponding leg. The third further merges all lower body classes, upper body classes, and head classes together. The fourth and final ground truth segmentation distinguishes only between background and non-background semantic classes. When forming the ground truths, regions were split as necessary to ensure that each is spatially contiguous. Visualizations of several ground truths and corresponding images are shown in Figure 9.

B.2 ADDITIONAL SST TRAINING DETAILS

Our network for estimating marginal probabilities used two sets of features as input. The first are visual features extracted from VGG-19 (Simonyan & Zisserman, 2014). Each image was run through the net, and the activations just prior to the second max-pooling operation were recorded (fourth convolutional layer). These activations were then upsampled back to the original image size using nearest neighbor interpolation, thereby producing 128 features for each pixel. These features were average-pooled within the base-level regions to give 128 visual features for each. The second set were geometric features regarding the regions and their relations. These features included: centroids of the parent, left, and right regions; difference and absolute difference between the left and right centroids; coordinates of the top left and bottom right bounding box corners of each region; height and width of the regions; normalized number of pixels of each region (maximum of 1); the difference and absolute difference of the normalized number of pixels in the child regions; and the difference between the parent normalized number of pixels and minimum (respectively maximum) of child normalized number of pixels. All coordinates were expressed relative to the image size such that both x and y coordinates ranged from -1 to 1. This resulted in 35 additional geometric features, yield 163 features total.

Each set of features has the benefit that the features of a parent region can efficiently be computed given features of the child regions. All features were normalized to have zero mean and unit variance across the training examples.

For BSDS500, the first layer of the marginal estimation network had 128 hidden units and the second layer had 64. For Penn-Fudan the first layer had 32 hidden units and the second had 16. ReLU activations with dropout were used for the first and second layers. The final layer had a single sigmoid output and no dropout.

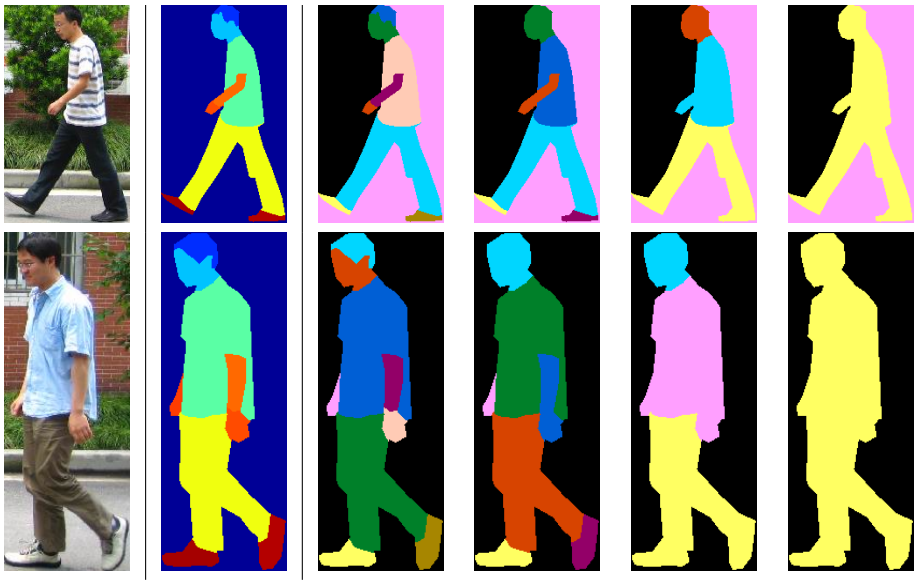

Figure 9: Example synthetic ground truths created for the Penn-Fudan dataset. From left to right: image, semantic segmentation labels, ground truths 1-4. The colors shown in the ground truths are not semantically meaningful: they are chosen to simply be maximally distinguishable for ease of visualization.

B.3 ADDITIONAL DETAILS REGARDING BOYKOV-JOLLY BASELINE

In BOYKOV-JOLLY, a strength is computed between any two adjacent base-level regions s_1 and s_2 as the mean of $1 - \frac{\exp(-(I_i - I_j)^2)}{2\sigma^2}$, where pixel $i \in s_1$ and pixel $j \in s_2$. Here $I_i \in [0, 1]$ denotes the mean intensity at pixel i across color channels. Merges are then sequentially made between regions in order of ascending strength. The strength between two arbitrary regions r_1 and r_2 is defined to be the minimum superpixel strength between two adjacent superpixels s_1 and s_2 such that $s_1 \in r_1$ and $s_2 \in r_2$. Merges continue in this way until a single region remains. This sequence of merges induces a segmentation tree, which can be thresholded by any strength from 0 to 1 in order to form a segmentation. In the results presented in Tables 1 and 2 of the main paper, a single threshold and σ with the best validation performance were chosen via grid search.

B.4 ADDITIONAL SAMPLES

We show additional samples from the SST-UTREE algorithm trained on BSDS 500 in Figure 10 and representative samples from SST-UTREE on Penn-Fudan in Figure 11.

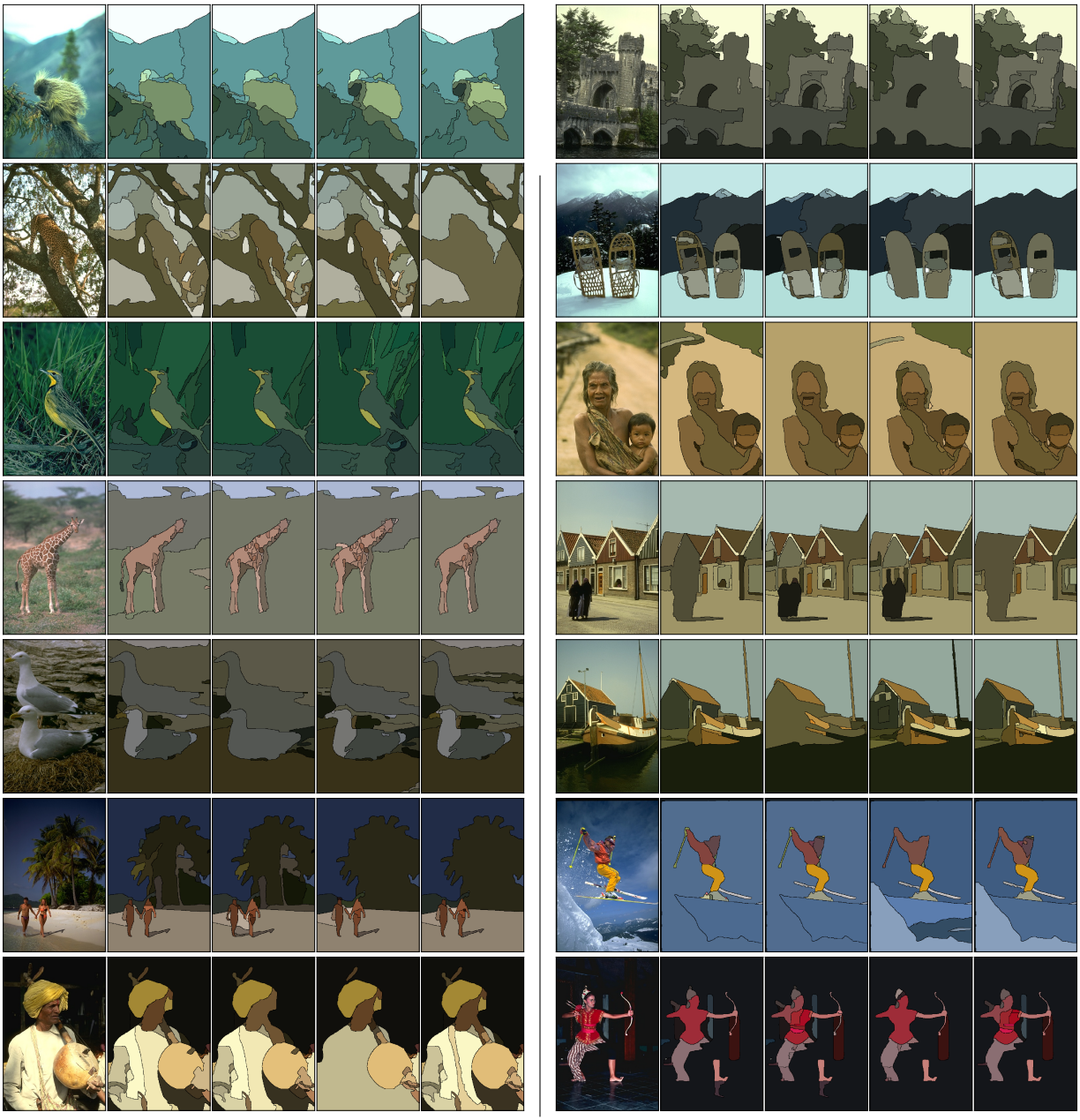

Figure 10: Samples from the SST-UTREE model on BSDS500. From left to right: image, samples 1-4 from the model. All images are from the test set.

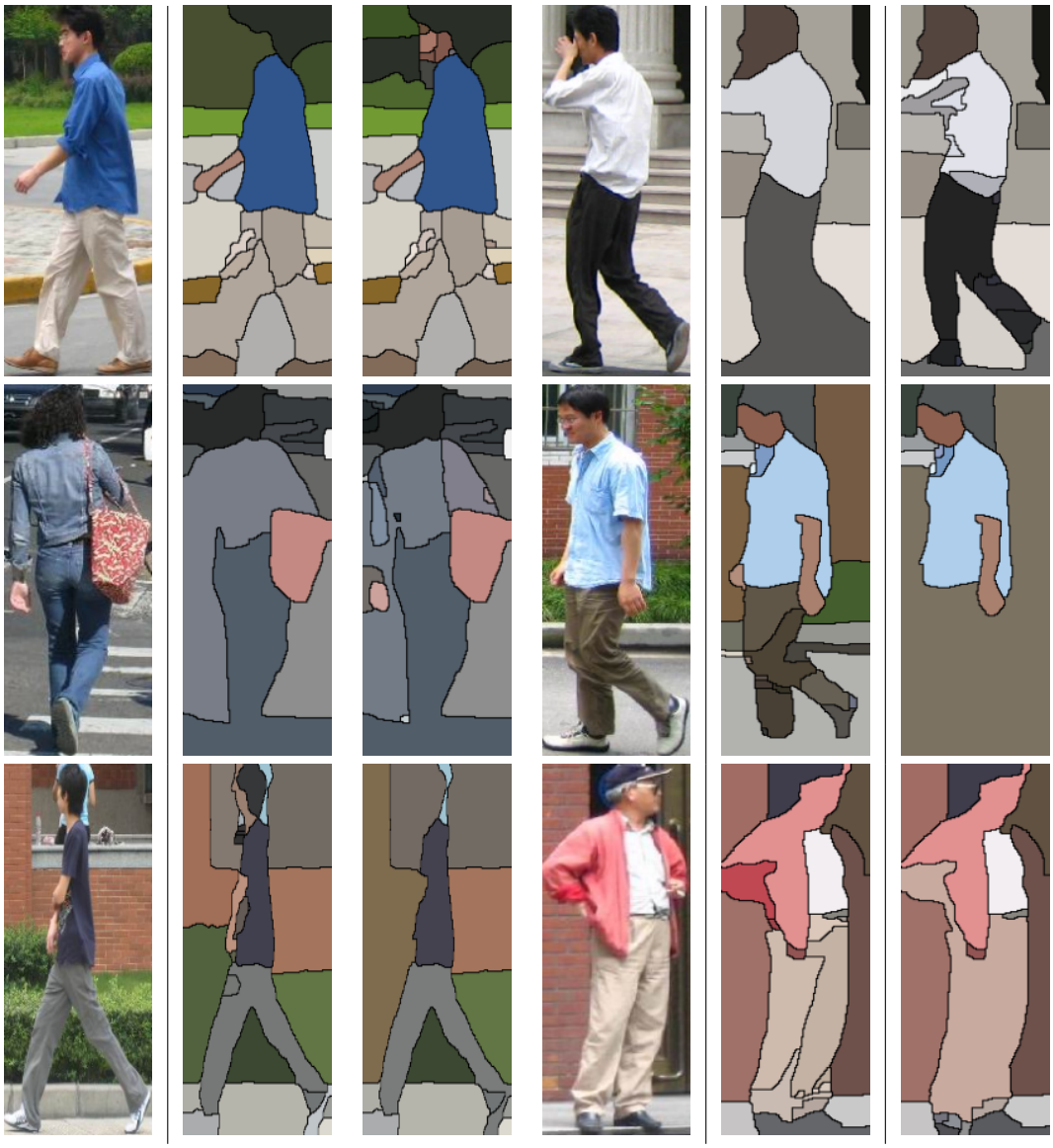

Figure 11: Representative samples from the SST-UTREE model on Penn-Fudan. Within each pane, from left to right: image, samples 1-2 from the corresponding test model. Note that for visualization purposes, the images and segmentations have been resized to the same dimensions; in the actual dataset the sizes vary.