Boston University Computer Science Technical Report Number BUCS-TR-2010-007. To appear in proceedings of CVPR 2010

Fast Globally Optimal 2D Human Detection with Loopy Graph Models

Tai-Peng Tian   Stan Sclaroff [*]
Boston University
{tian,sclaroff}@cs.bu.edu

Abstract

This paper presents an algorithm for recovering the globally optimal 2D human figure detection using a loopy graph model. This is computationally challenging because the time complexity scales exponentially in the size of the largest clique in the graph. The proposed algorithm uses Branch and Bound (BB) to search for the globally optimal solution. The algorithm converges rapidly in practice and this is due to a novel method for quickly computing tree based lower bounds. The key idea is to recycle the dynamic programming (DP) tables associated with the tree model to look up the tree based lower bound rather than recomputing the lower bound from scratch. This technique is further sped up using Range Minimum Query data structures to provide $O(1)$ cost for computing the lower bound for most iterations of the BB algorithm. The algorithm is evaluated on the Iterative Parsing dataset and it is shown to run fast empirically.

1. Introduction

Recovering the 2D configuration of a human figure from an image is difficult because human bodies are capable of a large variety of postures. A common approach is to model the human figure in terms of body parts, e.g., using Pictorial Structures (PS) [5]. When coupled with strong body part detectors, the PS model provides good results [1].

The PS model captures kinematic constraints between limbs using a tree structured graphical model. The model has been extended to include additional constraints such as appearance symmetry of clothing [16] and spatial constraints between body parts [8]. However, the associated graphical models are loopy graphs, i.e., graphs with loops, and the time complexity for recovering the global solution scales exponentially in the size of the largest clique in the graph.

Our goal is to recover the optimal solution for a loopy graph in a reasonable amount of time in practice. This is useful for comparing the performance of different models for detecting human figures and estimating body poses. Approximation algorithms are not useful in this case because we cannot ascribe a model's performance to modeling errors, approximation errors or a mixture of both. In contrast, we can exclude approximation errors from consideration when using exact algorithms.

Existing algorithms for detecting 2D human figures using loopy graphical models, both exact and approximate algorithms, suffer from the following drawbacks:

1. *Algorithm does not scale.* To keep the algorithm fast [2] or prevent an explosion of variables in the problem encoding [8, 16], the number of candidate locations for each body part is kept small ($\sim 10^3$ or less for each body part). These algorithms typically require a thresholding step to remove low scoring candidates and this thresholding heuristic is sub-optimal because candidates are removed without considering their relation to other body parts or the global cost function.

2. *Specialized edge cost.* The edge cost functions are assumed to have a specific form in order to accelerate the inference algorithm, e.g., variables are assumed to be jointly Gaussian in [10]. This limits the applicability of these models because useful constraints such as appearance symmetry [16] or positional exclusion [8] cannot be modeled using Gaussian distributions.

We present an approach for computing the exact solution that overcomes the aforementioned drawbacks. It is a global optimization algorithm based on Branch and Bound (BB) (Sec. 4). The BB algorithm uses a novel technique to quickly compute tree based lower bounds on the cost function and it only incurs an $O(1)$ look up cost for most of the BB iterations (Sec. 4.2). Consequently, this boost in speed enables our method to handle problem sizes that are multiple orders of magnitude greater than previous work ($\sim 10^6$ locations per body part compared to $\sim 10^3$ previously). Asymptotically, in the worst case, our algorithm requires exponential time, but our experiments show that the algorithm recovers the optimal solution in a reasonable amount of time in practice (Sec. 5).

[1] This research was funded in part through grant NSF IIS-0713168.

2. Related Work

We broadly categorize the related work based on the type of model used (tree vs. loopy models) and the class of inference algorithm used (approximate vs. exact).

Tree Model + Exact Inference In this class of work [1, 5, 14, 15], the state space is discrete and exact inference can be performed efficiently using the belief propagation algorithm. One drawback of the tree model is the over-counting of evidence problem where legs are often localized to the same region. In our work, we demonstrate how to ameliorate this problem using a loopy graph model that encodes constraints across different branches of the tree and we also provide a practical algorithm that performs exact inference quickly in practice.

Loopy Model + Approximate Inference In addition to the underlying kinematic tree model, useful pairwise constraints were proposed in [8, 16] and the resulting models contain loops in the associated graphs. The time complexity for exact inference in loopy graphs scales exponentially in the size of the largest clique in the graph and approximation algorithms are commonly used for inference.

Inference in loopy graph models can be transformed into Integer Programs [8] or Integer Quadratic Programs [16] and solved using approximate algorithms that depend on general purpose linear program solvers. Unfortunately, these approaches do not scale well to larger sized problems; e.g., both [8, 16] use less than 500 candidate locations for each body part. In comparison, our method performs exact inference by exploiting structure within the combinatorial problem and it is capable of handling more than 10^6 candidates for each body part.

Other approximate methods, such as Tree Reweighing (TRW) [18] or Loopy Belief Propagation (LBP) are an order of magnitude slower than our exact method. TRW and LBP require multiple rounds of message passing and when messages cannot be created using the distance transform trick [5], then creating a message requires $O(h^2)$ time, where h is the number of candidates for a body part. For example, the distance transform trick cannot be used to create messages along arcs that enforce constraints such as the Appearance Symmetry and Evidence Scaling constraints used in our experiments (Sec. 5). In our experience, computing a message involving one of these constraints takes 22 minutes and multiple rounds of message passing requires many hours of computation time. In contrast, our method avoids this problem by using a fast branch and bound algorithm that uses a novel lower bounding technique. Our method runs faster empirically and on top of that, it recovers the optimal solution.

A model that uses a convex combination of spanning trees was proposed in [19]. This model can be viewed as an approximation to a loopy graph model. In contrast, our work recovers the optimal solution to the original loopy graph model.

Loopy Model + Exact Inference The Common Factor Model (CFM) [10] was proposed as an alternative model to graphical models with dense cliques and the CFM assumes a jointly Gaussian distribution among the variables within a clique. In contrast, our method allows more complex relationships between variables. For example, we use the Region Covariance constraint in our experiments to compute the difference between two appearance patches and this cannot be handled in the CFM.

Bergtholdt, et al. [2] used the A^* search algorithm and compute an admissible cost using the Belief Propagation (BP) algorithm. In each iteration of the A^* search, the BP algorithm requires $O(nh^2)$ time for h labels in each of the n nodes. In comparison, our lower bounding technique requires a constant $O(1)$ time complexity for most iterations. This computational advantage allows our method to handle label sizes orders of magnitudes larger than [2] and thus obviates the need to threshold feature detection in [2], which potentially could lead to a sub-optimal solution.

Exact methods based on Non Serial Dynamic Programming [3], Junction Trees (see e.g., [9]) or Bucket Elimination [4] have memory requirements that are too large for the problem sizes we are handling. Consider the case where each variable has 10^6 candidate locations and further assume that we are only dealing with pairwise functions. If we are eliminating a variable within a clique of size three, then we need to store a table of size $10^6 \times 10^6 = 10^{12}$. If each label is stored as a four byte integer, then the table requires an astounding four terabytes of memory. Therefore, these algorithms are unsuitable for our problem size.

Others Other works do not explicitly optimize the energy function of a graphical model, e.g., [12] which uses over segmentation to assemble body parts and in [7], detection is formulated as a consistent max covering problem.

Branch and Bound Branch and bound is used in Bayesian Networks with large numbers of random variables with a small domain [11, 13]. The search proceeds by instantiating each of the random variables sequentially. For our problem, the situation is reversed. We have a small number of random variables but a large set of values for each variable ($\sim 10^6$). Rather than instantiating the variables sequentially (which is too slow for our problem size), we choose to prune away large regions of the solution space quickly through the use of fast upper and lower bounding techniques.

3. Human Model

We adopt the commonly used ten part model consisting of the torso, head, upper and lower arms, as well as upper and lower legs. The 2D configuration of a human figure $X = \{x_1, \dots, x_{10}\}$ consists of parameters x_i for each body part. The conditional probability of configuration X given

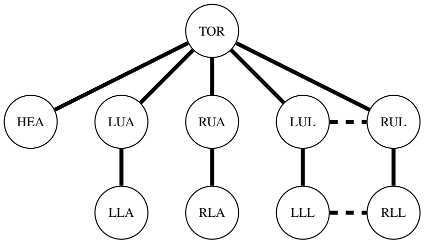

Figure 1. Ten body parts model. HEA : Head, TOR : Torso, LUA : Left upper arm, etc. Solid edges represent kinematic constraints between body parts and dotted edges represent additional constraints such as appearance symmetry constraints, spatial exclusion constraints, etc.

an image I, is

$$p(X|I) \propto p(I|X)p(X), \tag{1}$$

where $p(I|X)$ denotes the likelihood and $p(X)$ is the spatial prior over the configuration X.

3.1. Tree Model

The tree model was proposed in [5] and it assumes independence of appearance among the body parts. The likelihood is factorized as

$$p(I|X) \propto \prod_{i \in V} \phi_i(x_i), \tag{2}$$

where V represents the set of body parts (corresponding to vertices of the graph in Fig. 1) and the unary terms $\phi_i(x_i)$ are functions of the body part detector scores. The spatial prior is tree structured and it is factorized as

$$p(X) \propto \prod_{ij \in E_t} \phi_{ij}(x_i, x_j), \tag{3}$$

where E_t denotes the set of kinematic constraints between body parts (corresponding to solid edges in Fig. 1).

3.2. Non-Tree Model (Loopy Graph Model)

In the loopy graph model, additional edges are added to the basic tree model. The likelihood models additional appearance dependency between body parts and factorizes as

$$p(I|X) \propto \prod_{i \in V} \phi_i(x_i) \prod_{ij \in E_a} \psi_{ij}(x_i, x_j), \tag{4}$$

where E_a denotes the set of appearance constraint edges and the potential function $\psi_{ij}(x_i, x_j)$ models appearance dependency between body parts i and j.

The model also allows constraints between the spatial positions of the body parts. The spatial prior factorizes as

$$p(X) \propto \prod_{ij \in E_t} \phi_{ij}(x_i, x_j) \prod_{ij \in E_s} \sigma_{ij}(x_i, x_j), \tag{5}$$

where E_s denotes the set of additional spatial constraint edges and the set of functions $\sigma_{ij}(x_i, x_j)$ model spatial constraints between between parts i and j. The full model is

$$\begin{split} p(X|I) \propto \prod_{i \in V_1} \phi_i(x_i) \prod_{\substack{i \,\in\, V \,-\, V_1 \\ ij \,\in\, E_s}} \sigma_{ij}(x_i, x_j) \\ \prod_{ij \in E_t} \phi_{ij}(x_i, x_j) \prod_{ij \in E_a} \psi_{ij}(x_i, x_j), \end{split} \tag{6}$$

where V_1 represents the set of vertices that are not connected by any of the ψ_{ij} or σ_{ij} edges.

4. Optimization

We recover the optimal solution by minimizing the negative log of the posterior, i.e.,

$$X^* = \arg \min_X U(X), \tag{7}$$

where the energy $U(X) \propto -\log P(X|I)$. The potentials are exponential functions, where $\phi_{ij} \propto \exp(-U_{ij}(x_i, x_j))$, and the energy $U(X)$ has the form,

$$U(X) = \sum_{i \in V} U_i(x_i) + \sum_{ij \in E_t \bigcup E_a \bigcup E_s} U_{ij}(x_i, x_j). \tag{8}$$

The energy minimization proceeds by first discretizing the parameter space for X (following [5]), then the energy function is minimized using a Branch and Bound (BB) algorithm. The BB algorithm recursively subdivides the search space into disjoint regions. These regions are ranked according to a lower bounding function and the region with the smallest lower bound is chosen for the next subdivision step (details are shown in Algorithm 1).

4.1. Lower Bound

The performance of the Branch and Bound algorithm depends on the quality of the lower bounding function. We define the following lower bound that performs well in practice. Given a region Ω that contains one or more solutions, a lower bound $LB(\Omega)$ satisfies the property $LB(\Omega) \leq U(X_i)$ for all $X_i \in \Omega$. The lower bound for a region Ω is

$$LB(\Omega) = \begin{cases} U(X) & \text{If } \Omega \text{ contains only } X, \\ \min_{X \in \Omega} U_{tree}(X) & \text{otherwise,} \end{cases} \tag{9}$$

where

$$U_{tree}(X) = \sum_{i \in V} U_i(x_i) + \sum_{ij \in E_t} U_{ij}(x_i, x_j) \tag{10}$$

is the energy associated with the tree model.

When defining Eq. 9, we require that the tree model cost is less than the original model cost, i.e., $U_{tree}(X) \leq U(X)$.

Algorithm 1 Branch and Bound algorithm

```
Set $\Omega$ as initial solution space and set priority queue $Q$ to empty
$X^* = \arg\min_{X \in \Omega} U_{tree}(X)$
$UB = U(X^*)$
Insert $(\Omega, LB(\Omega))$ into $Q$
while true do
    $\Omega$ = pop($Q$)
    if $\Omega$ contains only a single solution $X$ then
        Return $X$
    else
        $(\Omega_1, \Omega_2)$ = split($\Omega$)
        $X_1^* = \arg\min_{X \in \Omega_1} U_{tree}(X)$
        $X_2^* = \arg\min_{X \in \Omega_2} U_{tree}(X)$
        $UB = \min\{U(X_1^*), U(X_2^*), UB\}$
        If $LB(\Omega_1) \leq UB$, insert $(\Omega_1, LB(\Omega_1))$ into $Q$
        If $LB(\Omega_2) \leq UB$, insert $(\Omega_2, LB(\Omega_1))$ into $Q$
    end if
end while
```

Algorithm 2 (Ω_1, Ω_2) = split(Ω)

```
Input : $\Omega = \langle \omega_1 \times \cdots \times \omega_{10}, RMQ(d), d \rangle$ where $\omega_i$ is the candidate list for body part $i$, with $d$ as the index of the current body part's candidate list being split in topological ordering and $RMQ(d)$ denotes the range minimum query data structure built over the cost table $D_d(x_d)$ (Eq. 16).
if $|\omega_d| == 1$ then
    $\Omega' = \langle \omega_1 \times \cdots \times \omega_{10}, RMQ(d+1), d+1 \rangle$
    $(\Omega_1, \Omega_2)$ = split($\Omega'$)
    Return $(\Omega_1, \Omega_2)$
end if
Suppose $\omega_d = [p_1, \cdots, p_k]$.
$\omega'_d = [p_1, \ldots, \frac{1}{2}(p_1 + p_n)]$
$\omega''_d = [\frac{1}{2}(p_1 + p_n) + 1, \ldots, p_n]$
$\Omega_1 = \langle \omega_1 \cdots \times \omega'_d \times \cdots \times \omega_{10}, RMQ(d), d \rangle$
$\Omega_2 = \langle \omega_1 \cdots \times \omega''_d \times \cdots \times \omega_{10}, RMQ(d), d \rangle$
Return $(\Omega_1, \Omega_2)$
```

This can be satisfied by assuming that all the unary and pairwise cost functions in the energy $U(X)$ are non-negative. This assumption is easily satisfied in practice. The unary costs are normalized to the range $[0, 1]$ using a softmax transformation and the pairwise functions are typically distance functions, which are non-negative by definition.

4.2. Computing the Lower Bound $LB(\cdot)$ Efficiently

Computing the lower bound involves a minimization over the region Ω. Naively, if we compute the minimum cost using dynamic programming on the tree structure then it requires a time complexity of $O(nh^2)$ for n nodes with h labels each. This is a prohibitive cost for each branch and bound iteration because the number of labels is huge ($h \sim 10^6$ in the problem size we are handling). We outline an approach that avoids computing the lower bound from scratch at each iteration and instead it recovers the lower bound from a look up table.

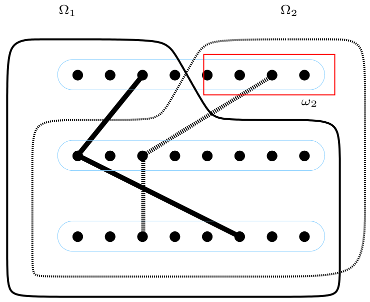

Figure 2. Region partitioning by making use of the dynamic programming trellis. The original domain is partitioned into Ω_1 and Ω_2 by splitting the domain for the body part at the root into half. The region Ω_1 retains the original tree model solution and the optimal tree model solution has to be recomputed for Ω_2, which is efficiently computed by making use of a range minimum search data structure.

We describe the key ingredients of our technique using a toy example. The loopy model consists of three variables $X = \{x_1, x_2, x_3\}$, with cost function

$$U(X) = \sum_{i \in \{1,2,3\}} U_i(x_i) + \sum_{ij \in \{12,13,23\}} U_{ij}(x_i, x_j). \quad (11)$$

In a preprocessing step, we select the spanning tree with x_1 as the root, x_2 and x_3 are the child and grandchild respectively. The associated tree cost is

$$U_{tree}(X) = \sum_{i \in \{1,2,3\}} U_i(x_i) + \sum_{ij \in \{12,23\}} U_{ij}(x_i, x_j). \quad (12)$$

Next, we apply dynamic programming to obtain a set of DP tables $B_j(x_i)$ [5]. These tables are recursively defined for non-root nodes i as

$$B_j(x_i) = \min_{x_j} U_j(x_j) + U_{ij}(x_i, x_j) + \sum_{k \in \text{child}(j)} B_k(x_k), \quad (13)$$

where child(j) is the set of child nodes for j. Fig. 2 shows the familiar dynamic programming trellis, where each row of black dots represents the state space of the variable (with the root node at the top). The solid path from the root to the leaf is the minimum cost solution. Once the programing table is constructed, we recover the lower bound

$$LB(\Omega) = \min_{x_1} U_1(x_1) + B_2(x_1). \quad (14)$$

Given the initial solution space $\Omega = S_{(1,h)} \times S_{(1,h)} \times S_{(1,h)}$, where $S_{(1,h)} = \{1, \ldots, h\}$ is an index set over all possible configurations of a body part, the first iteration of the branch and bound partitions the region Ω into two smaller regions Ω_1 and Ω_2 and then the algorithm computes the lower bounds $LB(\Omega_1)$ and $LB(\Omega_2)$ for the new regions. Naively, these bounds are computed using dynamic programming but we can speed up the computation by selecting an advantageous partitioning for Ω.

We propose partitioning the original region Ω by splitting the domain of the root node in half, i.e., $\Omega_1 = S_{(1,\lfloor h/2 \rfloor)} \times S_{(1,h)} \times S_{(1,h)}$ and $\Omega_2 = S_{(\lfloor h/2 \rfloor+1,h)} \times S_{(1,h)} \times S_{(1,h)}$. This partitioning scheme allows us to reuse the optimal tree solution from the larger domain Ω. In the toy example, the optimal tree solution for the larger region Ω is contained within the left smaller region Ω_1 (Fig. 2). We can easily verify that $LB(\Omega_1) = LB(\Omega)$.

The remaining task is to compute the optimal tree solution for the other region Ω_2. The algorithm scans the second region Ω_2 root level entries, in the set ω_2, for the minimum cost entry. The lower bound for the second region is

$$LB(\Omega_2) = \min_{x_1 \in \{\lfloor h/2 \rfloor+1,h\}} U_1(x_1) + B_2(x_1). \tag{15}$$

Once the minimum cost entry is found, we can reconstruct the optimal tree solution by back tracking through the rest of the DP table.

In the toy example, when the domain for the root is reduced to a single element, i.e., when $\Omega = S_{(p,p)} \times S_{(1,h)} \times S_{(1,h)}$, then the child node will be split next. In general, we use a topological sorted ordering of the tree nodes such that no child nodes come before a parent node and this ordering allows the cost entries in DP tables to be reused. Before splitting the new node, some housekeeping is required to update the cost entries of the new node. If there are n variables and the domains for the first $k-1$ variables have been reduced to a single element, i.e.,

$$\Omega = S_{(p_1,p_1)} \times \cdots \times S_{(p_{k-1},p_{k-1})} \times S_{(1,h)} \cdots \times S_{(1,h)},$$

then the cost entries for node k are defined as

$$\begin{aligned} D_k(x_k) =& C_k(X_k^*) + U_k(x_k) + U_{kk'}(x_k, x_{k'}[p_{k'}]) \\ &+ \sum_{i \in child(k)} B_i(x_k), \end{aligned} \tag{16}$$

where k' denotes the parent of k, $child(k)$ are the children of node k, the notation $x_i[p_i]$ refers to looking up the discretized value for body part i using the index p_i, and $C_k(\cdot)$ denotes the fixed tree cost for node k which is defined as

$$C_k(X_k^*) = \sum_{i \in \overline{V}} U_i(x_i[p_i]) + \sum_{\substack{ij \in E_t \\ i,j \in \overline{V}}} U_{ij}(x_i[p_i], x_j[p_j]), \tag{17}$$

and X_k^* denotes tree solution obtained by fixing the first $k-1$ values and identifying the values for the other parts by backtracking in the DP tables. The set of vertices $V(k)$ includes node k and its descendants and $\overline{V} = V - V(k)$. Intuitively, the fixed tree cost is a partial cost of the lower bound that does not change with respect to splitting the domain of node k.

Speedup Using Range Minimum Query (RMQ) Data Structure Computationally, we require linear time to search for the minimum cost entry in the set ω_2. This linear search is repeat over the same list but over different ranges of the list when computing the lower bound. We propose building a RMQ data structure [6] over the list such that querying for the minimum cost entry within a given range only requires $O(1)$ processing time. Building the RMQ data structure requires $O(h)$ time for processing h entries in a list. At the start of the branch and bound algorithm, the RMQ structure is built over the DP table of the root node, i.e., $B_1(x_1)$ and subsequently, the RMQ table is built over the cost table $D_k(x_k)$ (Eq. 16) when splitting node k.

Pruning by Upper Bounding A pruning step is added to the Branch and Bound algorithm (Algorithm 1). In each iteration, the region Ω is split into Ω_1 and Ω_2. On top of recovering the lower bounds, we also recover the actual configurations X_1^* and X_2^* that correspond to the tree costs (this is done with a constant cost by backtracking using the back pointers in the DP tables). We compute the actual loopy graph cost $U(X_1^*)$ and $U(X_2^*)$ and pick the smaller value as an upper bound UB on the optimal solution, i.e., $UB = \min\{U(X_1^*), U(X_2^*)\}$. In general, we keep the smallest upper bound encountered so far in the branch and bound. Regions with lower bounds that exceed the current upper bound are pruned away, i.e., these regions will not be inserted into the priority queue. This is safe because the optimal solution is not within these regions.

4.3. Correctness of Algorithm 1

We prove that Algorithm 1 always terminates and returns the optimal solution. The solution space Ω is discrete and we can map the subdivision process onto a tree. The root of the tree represents the entire solution space, the child nodes are the subdivided regions and the leaves are singleton sets, i.e., sets with only one solution. Since there is a finite number of leaf nodes, the algorithm will terminate and in the worst case it visits all the leaf nodes. To prove that the algorithm recovers the optimal solution, we argue that the first singleton set popped off the priority queue is the optimal solution. In the case where the top of the priority queue contains the singleton set $\Omega = \{X\}$ then its lower bound is the smallest among all the other regions Ω' in the priority queue, i.e., $LB(\{X\}) \leq LB(\Omega')$. Since $LB(\Omega') \leq U(Y)$ for all $Y \in \Omega'$ and $U(X) = LB(\{X\})$ then $U(X) \leq U(Y)$ for all $Y \in \Omega'$. □

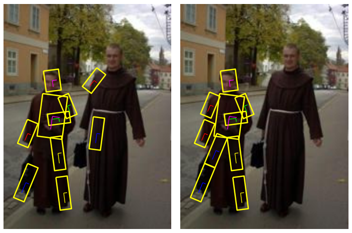

Figure 3. **Left:** Andriluka et al. (AN) [1] solution does not tightly group the body parts and body parts are localized on two different human figures. **Right:** Our result enforces kinematic constraints on AN's solution. This is achieved by performing a MAP inference using AN's posterior marginal as unary cost and adding a tree structured kinematic constraint.

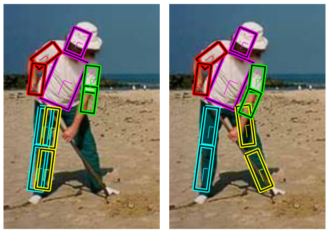

Figure 4. Resolving over-counting of evidence problem. **Left:** Using Andriluka et al.'s [1] algorithm, both legs are localized onto the same region. **Right:** Our method finds a better solution with legs apart after enforcing the Evidence Scaling constraints that rescale detector scores based on the overlapping regions.

5. Experiments

Currently, [1] reports the best 2D human detection results on the Iterative Parsing dataset [14], but the inference method has two drawbacks. Firstly, the body parts are not tightly grouped together (see Fig. 3). Secondly, the solutions suffer from the over-counting of evidence problem common in tree models (see Fig. 4). Our method ameliorate these two problems.

For comparison with [1], we use the same dataset, i.e., the Iterative Parsing dataset [14]. The standard tree model [5] is chosen as the spanning tree to compute the lower bound (shown as the tree with solid edges in Fig. 1). Following [1], each body part x_i consists of three parameters, namely, rotation and (x, y) positions.

Running Time We use the same state space discretization as [1], i.e., 24 rotation angles and all image positions are considered. An image size of 167×251, has slightly over a million candidate locations for each body part. Currently, the algorithms are implemented in Matlab and the computation-intensive parts are implemented using mex files. On average, it takes about one minute to compute the DP tables and another minute for the branch and bound algorithm to converge. We independently implemented the algorithm of [1] using Matlab, and the detection accuracy differs slightly different from results published in [1] (see Table. 1, second row).

Kinematic Constraints We adopt the kinematic constraints of [5] and the pairwise potential is given as

$$\phi_{ij}(x_i, x_j) = \exp\left\{-\frac{\lambda}{2}(x'_i - x'_j)^T M_{ij}^{-1}(x'_i - x'_j)\right\}, \tag{18}$$

where $x'_i = T_{ij}(x_i)$ and $x'_j = T_{ji}(x_j)$ are transformed coordinates and the diagonal covariance matrix M_{ij} can be learned from training samples (see [5] for more details).

Unary Cost The strong body parts detector proposed by Andriluka, et al. [1] provides good results for detection. Each body part i is detected by maximizing the marginal posterior $p(x_i|I)$. These marginal posteriors can be reinterpreted as re-weighted body part detector scores. We use the marginal posteriors as the unary cost for our model, i.e.,

$$\phi_i(x_i) = p(x_i|I). \tag{19}$$

In the absence of other pairwise terms ϕ_{ij}, ψ_{ij} and σ_{ij}, our model reduces to Andriluka et al.'s model. In this case, the best solution for each body part will be the maximum of the posterior marginal, but such an inference algorithm is unable to group the body parts tightly. This results in dismemberment of the human figure (Fig. 3 left). This can be rectified by adding kinematic constraints ϕ_{ij} (Fig. 3 right).

Appearance Symmetry Humans tend to wear clothing with symmetrical appearance [16] and we include such constraints in our model. The constraints are shown as dotted edges in Fig. 1. We use the Region Covariance (RC) descriptor [17] to describe the appearance of a rectangular patch associated with a body part configuration x. We extract the spatial coordinates (u, v) and red, green and blue color intensity (r, g, b) for each pixel location i within the patch, i.e., $F(y_i) = [u_i, v_i, r_i, g_i, b_i]$. The RC descriptor is the covariance matrix of the collection of $F(y_i)$. Given two covariance matrices C_1 and C_2, a distance metric is $\rho(C_1, C_2) = \sqrt{\sum_i \ln^2 \lambda_i(C_1, C_2)}$ where λ_i are the generalized eigenvalues. The potential is defined as

$$\psi_{ij}(x_i, x_j) = \exp(-\rho(C_1, C_2)). \tag{20}$$

Evidence Scaling Tree models suffer from the over-counting of evidence problem. Limbs are often placed close together in a region with high detection scores. This is further exacerbated by the symmetric appearance constraint since stacking one limb on top of the other will give identical appearance. We address this problem by scaling the unary costs for limbs participating in a symmetry appearance term (see Fig. 4). For a limb x_i constrained to have symmetric appearance with another limb x_j we scale the body part detector score with

$$\sigma_{ij}(x_i, x_j) = \frac{|R(x_i) \setminus R(x_j)| + \frac{1}{2}|R(x_i) \bigcap R(x_j)|}{|R(x_i)|} \tag{21}$$

where $R(x)$ denotes the 2D region of the limb and $|.|$ is the area of the region. The scaling $\sigma_{ij}(x_i, x_j)$ is in the range of $[\frac{1}{2}, 1]$. The evidence scaling function is not symmetrical, i.e., $\sigma_{ij}(x_i, x_j) \neq \sigma_{ji}(x_j, x_i)$. In practice, it suffices to include one of these terms and this has the effect of penalizing the intersection between the two areas. The area of intersection between two rectangles is computed by first clipping one rectangle against the other using the Sutherland Hodgman algorithm and the resulting intersection polygon's area can be computed using the surveyor's formula.

The full model is shown in Fig. 1. The solid edges form the kinematic tree and the dotted edges are the additional pairwise constraints. Intuitively, our model discourages overlapping parts and rewards finding body parts with similar appearance.

Dimensionality of RMQ. The body part configurations in the DP table are 3D but we use a 1D Range Minimum Query algorithm [6] to reduce memory usage in the BB algorithm. The DP table is flattened into a 1D array (ordering does not matter). This flattening does not affect the correctness of the Branch and Bound algorithm as the solution space is not modified.

Results We compare our algorithm against the method in [1] and the quantitative comparison is summarized in Table 1. A body part is correctly localized when both endpoints of the body part are within half the body part length of the ground truth (following [1]). We compare the tree structured maximum a posteriori solution [5] (row 1) with our model (row 5), and we achieve an improvement of $\sim$ 6% in the average detection rate. When compared with [1] (row 2), our results (row 5) achieve an improvement of $\sim$ 1% in the average detection rate with a noticeable improvement in the localization of both the upper and lower legs. Our Evidence Scaling and Appearance Symmetry constraints are effective in curbing the dismemberment problem and the over-counting of evidence problem.

Performance of Branch and Bound We conducted two experiments to assess the performance of the Branch and Bound algorithm. See Fig. 6 and Fig. 7 for details.

6. Discussion and Conclusion

We proposed a Branch and Bound algorithm to compute the global optimal solution for a non-tree model and the algorithm converges quickly in practice. As demonstrated in the experiments, enforcing Evidence Scaling and Appearance Symmetry in a loopy model helps to improve the accuracy of the state of the art 2D human detector. Furthermore, our optimization technique is general since the algorithm only requires the constraint functions to return a scalar value when evaluating the upper bound and these functions can effectively be treated as black boxes.

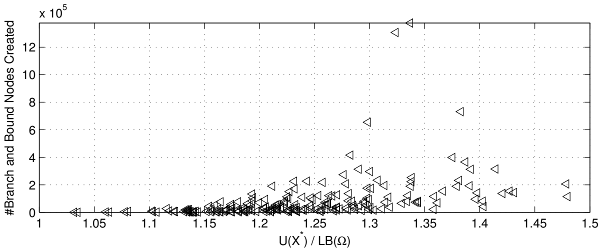

Figure 6. We examine the relationship between the total number of BB nodes explored during the search (vertical axis) and quality of the lower bound. The quality of the lower bound which is computed as the ratio of the loopy graph cost for the optimal solution X^* and the first lower bound computed in the BB, i.e., $U(X^*)/LB(\Omega)$, where Ω is the entire solution domain. Each triangle in the plot represents a test image from the Iterative Parsing dataset. For 97% (199 out of 205) of the test cases, the BB converges after visiting at most 4×10^5 nodes (or at most 100 secs.). Empirically, the optimal cost $U(X^*)$ is at most 1.5 greater than the initial tree based lower bound $LB(\Omega)$, and this good approximation allows our BB algorithm to converge quickly.

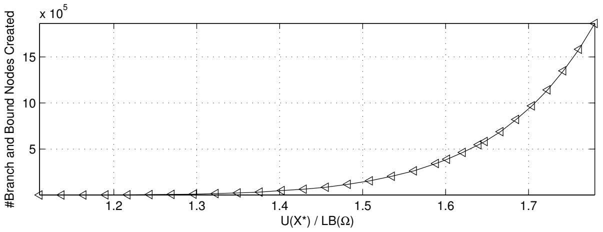

Figure 7. We probe the performance of the algorithm when the lower bound deteriorates. For a fixed image, we increased the ratio $U(X^*)/LB(\Omega)$ by increasing the weights on the Evidence Scaling and Region Covariance constraints. As the lower bound deteriorates, the number of BB nodes explored increases gradually but after a certain threshold it increases sharply in a non-linear fashion and our server (with 32GB RAM) quickly runs out of memory to store the priority queue. This behavior is typical for all the images but the threshold varies among the images.

The spanning tree chosen for the lower bound consists of all the kinematic edges in the Pictorial Structure model. This is primarily for efficiency reasons: the DP table is constructed in linear time [5] instead of quadratic time for general DP. The quadratic time complexity is impractical in our case because of the large number of candidates for each body part ($\sim 10^6$). But our technique works for any choice of spanning tree. In future work, algorithms could be developed to choose the spanning tree by optimizing the tightness of the resulting lower bound. Potentially, these other spanning trees may incur the quadratic computational cost in computing the DP table. Thus, the choice of spanning tree must balance between complexity of computing the DP tables and the tightness of the bound. In our experiments, we found that the spanning tree we propose strikes a good balance and yields the optimal detection and pose estimation in a reasonable amount of time, i.e., within minutes, rather than hours or days.

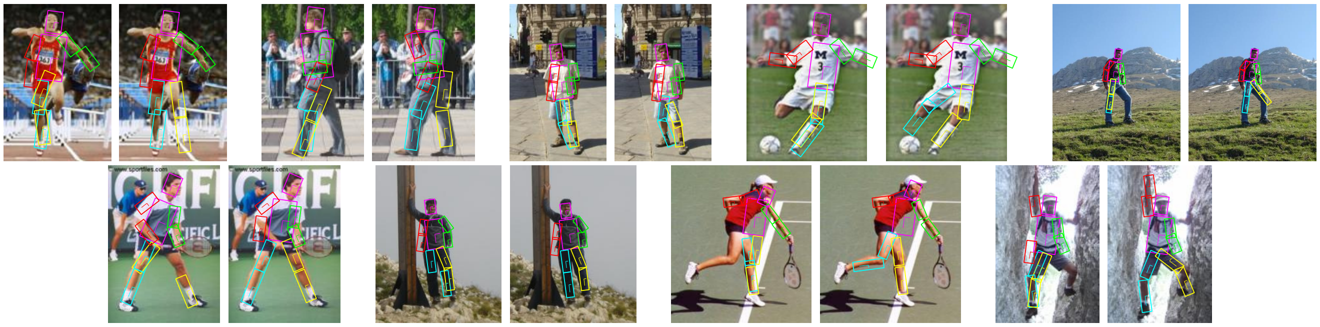

Figure 5. **First Row Over-counting of Evidence Problem:** When using the method in [1], two legs are frequently localized to the same region (left image). By making use of the Evidence Scaling and Appearance Symmetry constraints, our model tries to recover a solution where the left and right legs are similar in appearance but with less overlap between the legs. **Second Row, Dismemberment Problem:** The body parts are not tightly grouped when using [1] (left image) and this is corrected in our model (right image).

	Torso	Upper Arms		Upper Legs		Lower Arms		Lower Legs		Head	Avg
		Left	Right	Left	Right	Left	Right	Left	Right		
FH [5]	73.7	43.4	41.5	62.0	54.1	35.6	29.3	57.0	50.2	59.5	50.6
AN [1]	78.0	45.4	49.3	65.9	60.0	37.6	35.6	60.0	52.2	66.3	55.0
AN + KC	78.5	46.8	48.8	66.3	61.0	39.5	32.2	61.0	54.6	67.3	55.6
AN + KC + ES	80.0	46.3	49.3	67.3	63.4	39.0	32.7	61.0	57.0	67.8	56.4
AN + KC + RC + ES	80.0	46.8	49.3	68.8	61.5	39.5	32.7	62.0	55.6	67.3	56.4

KC : Kinematic Constraints RC : Region Covariance ES : Evidence Scaling

Table 1. Body part detection accuracy in percentages. A body part is correctly localized when both ends of the limb are within half the part's length from the ground truth. **Row 1:** Using FH [5] dynamic programming. **Row 2:** Using the sum product belief propagation for inference as suggested in AN [1]. **Row 3-5:** Combining different constraints with the original AN model. Note that the AN row differs slightly from the published result of [1] because we used our implementation.

References

[1] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited : People detection and articulated pose estimation. In *CVPR*, 2009. 1, 2, 6, 7, 8

[2] M. Bergtholdt, J. Kappes, S. Schmidt, and C. Schnorr. A study of part-based object class detection using complete graphs. *IJCV*, 28(3):416–431, 2009. 1, 2

[3] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972. 2

[4] R. Dechter. Bucket Elimination : A unifying framework for probabilistic inference. In *Proc. UAI*, 1996. 2

[5] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005. 1, 2, 3, 4, 6, 7, 8

[6] J. Fischer and V. Heun. Theoretical and practical improvements on the RMQ-problem with applications to LCA and LCE. In *Proc. of the 17th Annual Symposium on Combinatorial Pattern Matching*, 2006. 5, 7

[7] H. Jiang. Human pose estimation using consistent max-covering. In *ICCV*, 2009. 2

[8] H. Jiang and D. R. Martin. Global pose estimation using non-tree models. In *CVPR*, 2008. 1, 2

[9] D. Koller and N. Friedman. *Probabilistic Graphical Models Principles and Techniques*. MIT Press, 2009. 2

[10] X. Lan and D. P. Huttenlocher. Beyond trees: Common-factor models for 2D human pose recovery. In *ICCV*, 2005. 1, 2

[11] R. Marinescu and R. Dechter. AND/OR branch-and-bound for graphical models. In *IJCAI*, 2005. 2

[12] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *CVPR*, 2004. 2

[13] J. D. Park and A. Darwiche. Solving MAP exactly using systematic search. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003. 2

[14] D. Ramanan. Learning to parse images of articulated objects. In *NIPS*, 2006. 2, 6

[15] D. Ramanan, D. A. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *PAMI*, 29(1):65–81, 2007. 2

[16] X. Ren, A. C. Berg, and J. Malik. Recovering human body configurations using pairwise constraints between parts. In *ICCV*, 2005. 1, 2, 6

[17] O. Tuzel, F. Porikli, and P. Meer. Region covariance : A fast descriptor for detection and classification. In *ECCV*, 2006. 6

[18] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005. 2

[19] Y. Wang and G. Mori. Multiple tree models for occlusion and spatial constraint in human pose estimation. In *ECCV*, 2008. 2