Pictorial Strucutures for Articulated Human Pose Estimation
Technical Report

Vahid Kazemi
vahidk@kth.se

September 6, 2011

1 Introduction

In this report we describe a framework to efficiently solve part matching problems in the form a pictorial structure. The framework can be used to solve a range of problems in Computer Vision including object detection, recognition and 2D pose estimation.

The problem of human pose estimation which we are going to focus on in this report is, given a set of images of humans with annotated parts, find the location of these parts in novel images. Figure 1 shows a sample image from our dataset with annotations. As you can see each image in the training set comes with 10 bounding boxes outlining the major parts of the body. The aspect ratio of the bounding boxes are assumed to be fixed, therefore we only need to find 4 parameters including the position (x, and y), angle, and scale of the parts.

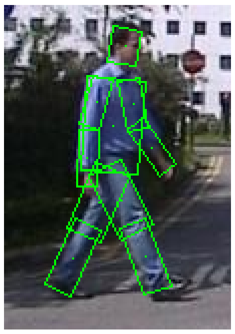

Figure 1: Sample image from IIP dataset [5] with annotated parts.

To find the location of parts we train individual models for each part by extracting PHOG descriptors [2] from the image and feeding them to a SVM classifier. The naive approach of finding the location of parts by taking the maximum likelihood of each part individually doesn't work well (Figure 2). Therefore we need a way to take into account not only the local appearance of each part, but the global configuration of parts as well. One way to formulate this is to use the pictorial structure [4] model. Pictorial structure represents an object as a graph ($G = [V, E]$) where each node represents a part of the object. To simplify

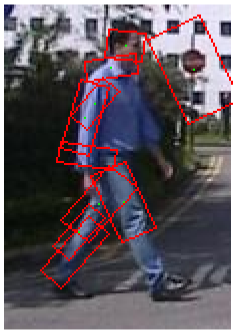

Figure 2: Taking the maximum likelihood of each part individually doesn't give good results.

the problem appearance of parts are considered to be independent. The penalty for a configuration L is given by the sum of deformation score (d_{ij}) between all connected nodes plus the mismatch score (m_i) of appearance of each part.

$$penalty(L) = \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) + \sum_{v_i \in V} m_i(l_i) \tag{1}$$

This is what we want to minimize over all the possible locations. The complexity of a brute-force approach to solve this is $O(H^n)$ where H is the number possible location for each part, and n is the number of parts. It's easy to see that this solution is not tractable even for small images. In fact the method that we present in this report is able to find the optimal solution in linear time, that is $O(Hn)$, with the simplification of using a tree instead of a generic graph, and by approximating the distribution of relative location of pairs of nodes with Gaussian distributions.

2 Efficient Minimization

In this section we describe how we can use dynamic programming to efficiently solve the described minimization problem efficiently. The basic idea is to calculate the contribution of each node of the graph from the leaves up to the root of the tree, and take the minimum on the location of root to get the global minimum. The contribution of jth node in the total configuration score is given by:

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) + \sum_{(v_j,v_c)\in E} B_c(l_j) \right) \tag{2}$$

The optimal location of a leaf node can be simply expressed as a function of only its parents:

$$B_j(l_i) = \min_{l_j} \left(m_j(l_j) + d_{ij}(l_i, l_j) \right) \tag{3}$$

The optimal location of every other node (except root) is dependent on the location of parent in addition to the optimal location of children nodes. Therefore we can simply solve the problem starting from the maximum depth, calculating the contribution of the leaf nodes for all possible locations in a grid, and then doing the same for the lower depth. In the next section we describe how to calculate the value of B_j efficiently using the generalized distance transform [3].

3 Generalized Distance Transform

Consider we have a set of points (B) on a grid ($\mathcal{G}$), and we want to find for every point on the grid, the distance to the closest point in the set.

$$\mathcal{D}_B(x) = \min_{y \in B} \rho(x, y) \tag{4}$$

In the most simple case consider a one dimensional grid, and a euclidean distance function. Distance transform in this case can be efficiently computed in two passes. First we initialize the values on the grid with zero where the points are and infinities in the other locations. Then there would be a pass from left to right, and one from right to left, where we replace the value of each location with the maximum of current value and the last value plus the difference between two locations which is 1 in this case:

sequence:	0	0	1	0	0	1	0	0	0
initialization:	∞	∞	0	∞	∞	0	∞	∞	∞
left to right:	∞	∞	0	1	2	0	1	2	3
right to left:	2	1	0	1	1	0	1	2	3

The distance transform formulation can be extended to transform arbitrary functions. To understand this, first consider the indicator function $1_{B(y)}$, that is:

$$1_{B(y)} = \begin{cases} 0 & y \in B \\ \infty & otherwise \end{cases} \tag{5}$$

Using this we can rewrite equation 4 as:

$$\mathcal{D}_B(x) = \min_{y \in G} \rho(x, y) + 1_B(y) \tag{6}$$

Similar concepts can apply to an arbitrary choice of function instead of the indicator function, this gives the formulation of the generalized distance transform:

$$\mathcal{D}_f(x) = \min_{y \in G} \rho(x, y) + f(y) \tag{7}$$

Getting back to the original problem, we can define the deformation function in equation 2 in the form of the Mahalanobis distance of transformed locations:

$$d_{ij}(l_i, l_j) = (T_{ij}(l_i) - T_{ji}(l_j))^T M_{ij}^{-1} (T_{ij}(l_i) - T_{ji}(l_j)) \tag{8}$$

where M_{ij} is a diagonal covariance matrix, and T_{ij} and T_{ji} are transformation functions. We can then define the transformation function in such a way that enables us to use the generalized distance transform to efficiently calculate $B_j(l_i)$:

$$B_j(l_i) = \mathcal{D}_f(T_{ij}(l_i)) \tag{9}$$

Using 2 we can find f as follows:

$$f(y) = m_j(T_{ji}^{-1}(y)) + \sum_{(v_j, v_c) \in E} B_c(T_{ji}^{-1}(y)) \tag{10}$$

Note that to ensure that M_{ij} is a diagonal matrix, different dimensions of $T_{ij}(l_i) - T_{ji}(l_j)$ should be uncorrelated. Furthermore it should be zero mean (for 8 to hold). Assuming that $l_i - l_j$ has a mean equal to s_{ij} and the covariance Σ_{ij}, the transformation can be defined to satisfy the constraints as follows:

$$T_{ij}(l_i) = U_{ij}^T (l_i - s_{ij}) \tag{11}$$

$$T_{ji}(l_j) = U_{ij}^T l_j \tag{12}$$

where $\Sigma_{ij} = U_{ij} D_{ij} U_{ij}^T$. Therefore the joint distribution of l_i and l_j will have zero mean with diagonal covariance:

$$p(l_i, l_j) \sim \mathcal{N}(T_{ij}(l_i) - T_{ji}(l_j), 0, D_{ij}) \tag{13}$$

Now the question remains how to calculate the generalized distance transform with the distance function from equation 8. The fact the Gaussian distribution is assumed to be zero mean, with diagonal covariance enables us to calculate the distance transform efficiently for any number of dimensions. Figure 3 shows that the generalized distance transform for a Mahalanobis distance function can be calculated by taking the lower envelope of n parabolas (where n is the length of the grid). Note how this satisfies the definition of the generalized distance transform (7), the value of distance transform at each point on the grid is equal to the minimum of value of that point on the grid and the value of each other point plus the distance between the pair of points.

To find the lower envelope of the parabolas we only need to find the intersection of them. Two parabolas have one intersection point. Consider two points p and q on the grid, If q is less than p, parabola centered on q will be under the parabola centered on p before the intersection point, and over it after the intersection point. This gives the idea of a simple algorithm that calculates the distance transform on the grid in $O(n)$. The intersection point of two parabolas can be simply calculated as follows:

$$f(q) + (s - q)^2 = f(p) + (p - s)^2 \tag{14}$$

$$s = \frac{(f(p) + p^2) - (f(q) + q^2)}{2(p - q)} \tag{15}$$

The complete algorithm for calculating the distance transform comes in the following:

```
Algorithm DT(f)
k =  0
v[0] = 0
z[0] = $-\infty$
z[1] = $+\infty$
for q = 1 to n-1
        s  = ((f(q) + q2) - (f(v[k]) + v[k]^2))/(2q - 2v[k])
        if s $\leq$ z[k] then
                k  = k - 1
                goto 6
```

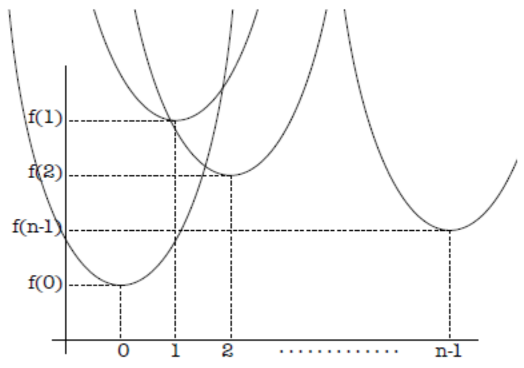

Figure 3: This figure shows that the generalized distance transform for a Mahalanobis distance function can be calculated by taking the lower envelope of n parabolas. (Source: [3])

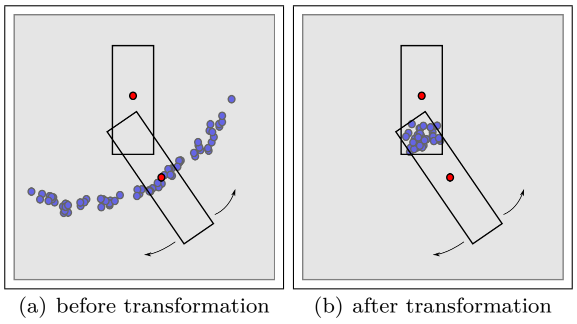

Figure 4: In these figures you see the distribution of relative location of two parts (centered around the parent part center). It can be seen that after transforming the location of the child node to the location of the joint ((b)), the Gaussian distribution will be a far better estimate.

```
        else
                k =  k + 1
        end
        v[k] = q
        z[k] = s
        z[k + 1] = $\infty$
end
k = 0
for q = 0 to n-1
        while z[k + 1] < q
                k = k + 1
        end
        Df (q) =  (q - v[k])^2 + f(v[k])
end
```

Pay attention that since we ensured that the different dimensions of f are uncorrelated we can simply extend the algorithm to arbitrary number of dimensions by applying the 1D distance transform to each dimension individually.

4 Optimal Joint Locations

In the last section we assumed that the deformation function can be estimated with a Gaussian distribution, but looking at figure 4(a) you can see that the Gaussian distribution may not be a good choice if we have the joints far from the center of the parts. This problem can be solved with a simple transformation

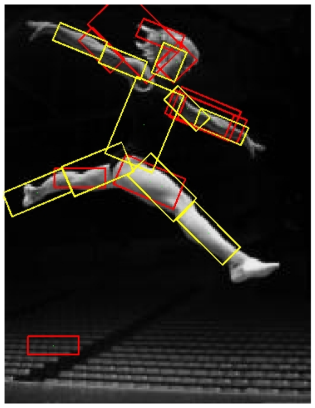

Figure 5: A sample result from IIP dataset. Yellow lines show the result of pictorial structures, and red lines show the result of finding each part independently.

to the joint location [1], as is shown in figure 4(b):

$$\begin{pmatrix} x' \\ y' \\ \theta' \\ s' \end{pmatrix} = \begin{pmatrix} x_i + s_i p_x^{ij} \cos\theta_i - s_i p_y^{ij} \sin\theta_i \\ y_i + s_i p_x^{ij} \sin\theta_i + s_i p_y^{ij} \cos\theta_i \\ \theta_i + \bar{\theta}_{ij} \\ s_i \end{pmatrix} \tag{16}$$

$$\begin{pmatrix} x' \\ y' \\ \theta' \\ s' \end{pmatrix} = \begin{pmatrix} x_j + s_j p_x^{ji} \cos\theta_j - s_j p_y^{ji} \sin\theta_j \\ y_j + s_j p_x^{ji} \sin\theta_j + s_j p_y^{ji} \cos\theta_j \\ \theta_j \\ s_j \end{pmatrix} \tag{17}$$

where (p_x^{ij}, p_y^{ij}) is the relative location of intersection joint of part i and j in respect to the center of part i, and (p_x^{ji}, p_y^{ji}) is the relative location of the joint in respect to the center of part j. We can find the optimal joint locations by solving an overdetermined set of equations using least squares as follows:

$$\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + s_i \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix} \begin{pmatrix} p_x^{ij} \\ p_y^{ij} \end{pmatrix} \tag{18}$$

$$\begin{pmatrix} x_j' \\ y_j' \end{pmatrix} = \begin{pmatrix} x_j \\ y_j \end{pmatrix} + s_j \begin{pmatrix} \cos\theta_j & -\sin\theta_j \\ \sin\theta_j & \cos\theta_j \end{pmatrix} \begin{pmatrix} p_x^{ji} \\ p_y^{ji} \end{pmatrix} \tag{19}$$

We will find p_x^{ji} and p_y^{ji} in such a way that the difference between transformed positions is minimized:

$$\min_{p^{ij}, p^{ji}} \left(\begin{pmatrix} x_i' \\ y_i' \end{pmatrix} - \begin{pmatrix} x_j' \\ y_j' \end{pmatrix} \right)^2 \tag{20}$$

this results to an overdetermined set of problems:

$$\begin{pmatrix} x_i - x_j \\ y_i - y_j \\ \vdots \end{pmatrix} = \begin{pmatrix} s_j \cos\theta_j & -s_j \sin\theta_j & -s_i \cos\theta_i & s_i \sin\theta_i \\ s_j \sin\theta_j & s_j \cos\theta_j & -s_i \sin\theta_i & -s_i \cos\theta_i \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} p_x^{ji} \\ p_y^{ji} \\ p_x^{ij} \\ p_y^{ij} \end{pmatrix} \tag{21}$$

which we can simply solve to get the get the optimal location of joints.

Figure 5 shows a sample result from IIP dataset.

5 Results

S	J	B	0.96	0.83	0.91	0.90	0.88	0.94	0.88	0.91	0.89	0.90
S	I	-	**0.97**	**0.91**	**0.93**	**0.93**	**0.95**	**0.95**	**0.91**	**0.94**	**0.96**	**0.93**
S	I	B	0.97	0.91	0.92	0.93	0.94	0.95	0.90	0.93	0.96	0.93

References

[1] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. 2009.

[2] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. CIVR, 2007.

[3] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions.

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[5] D. Ramanan. Learning to parse images of articulated bodies. In *In NIPS 2007*. NIPS, 2006.

[6] W. Yang, T. Duan, and L. Zicheng. Learning hierarchical poselets for human parsing. 2011.