IEEE Conference on Computer Vision and Pattern Recognition 2012 (CVPR'12)

# Scale Resilient, Rotation Invariant Articulated Object Matching

Hao Jiang
Boston College

Tai-Peng Tian
GE Global Research

Kun He, Stan Sclaroff
Boston University

## Abstract

*A novel method is proposed for matching articulated objects in cluttered videos. The method needs only a single exemplar image of the target object. Instead of using a small set of large parts to represent an articulated object, the proposed model uses hundreds of small units to represent walks along paths of pixels between key points on an articulated object. Matching directly on dense pixels is key to achieving reliable matching when motion blur occurs. The proposed method fits the model to local image properties, conforms to structure constraints, and remembers the steps taken along a pixel path. The model formulation handles variations in object scaling, rotation and articulation. Recovery of the optimal pixel walks is posed as a special shortest path problem, which can be solved efficiently via dynamic programming. Further speedup is achieved via factorization of the path costs. An efficient method is proposed to find multiple walks and simultaneously match multiple key points. Experiments show that the proposed method is efficient and reliable and can be used to match articulated objects in fast motion videos with strong clutter and blurry imagery.*

## 1. Introduction

We are interested in recovering key points of an object in images, for example, localizing the hands and feet of a human subject. The positions of these key points can be used in computer vision applications such as human computer interaction, movement analysis and activity recognition. The object(s) of interest may undergo significant deformations, articulations, rotations and scale changes; this can make key point localization challenging.

We propose a matching approach to recover the positions of key points in images. We assume a single exemplar image of the object is given and we will mark out pixel paths between key points by placing a few strokes on the image. Each stroke in the exemplar image defines the appearance properties of pixels along a path between key points. Given these paths in the exemplar image, we find the optimal paths that match the key points in novel images. For example, in Fig. 1, we use this approach to localize the head, hands and feet of a gymnast in cluttered images.

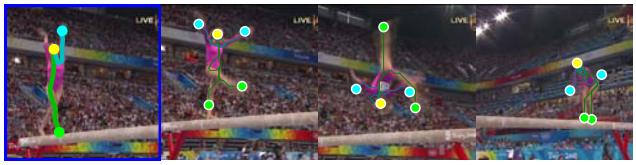


Figure 1. We match a single exemplar to target objects in cluttered images to find key points, e.g., head, hands and feet. The leftmost image shows the exemplar annotated with strokes that define an appearance model. This model is used to match target objects in the 2nd, 3rd and 4th image that have undergone significant deformations, articulations, motion blur, rotation, and scale changes.

The proposed method is based on identifying pixel paths with similar appearance to paths in an exemplar image. We model each path as a directed walk on pixels. Each step in a walk is guided by the similarity of image local appearance with the exemplar path, image local orientation, curvature, and step size. Coupling multiple walks on an image, we are able to model a complex articulated object and match multiple key points on the target. Our model is rotation invariant as well as articulation and scale resilient, i.e., it matches paths with different lengths, orientations and articulations in an efficient single-pass optimization. Our method is also versatile. In the experiments we demonstrate detecting key points on not only human subjects but also different deformable objects such as a fish, cloth, and a goose.

Our work is related to past work with Pictorial Structures [1, 2, 10, 11, 15] and with the Chains Model [4]. The Pictorial Structures (PS) model is commonly used for detecting articulated objects and the Chains Model incorporates linear context into the detection. Differently from these previous methods, the proposed method is able to efficiently optimize on a large number of local parts, and find the optimal matching in a single pass for a huge set of template patterns in foreshortening and scaling. Even though PS methods and the Chains model can be extended to include large number of parts, optimization on these models with each possible template configuration would be very slow.

While our method uses "linear structures", it has important differences with the Chains model [4]. First the Chains model requires a large set of training samples to learn an accurate model for a specific problem, whereas our method requires one exemplar and can handle arbitrary objects. Sec-

978-1-4673-1228-8/12/$31.00 ©2012 IEEE

ond, the Chains model formulation is not rotation invariant and multiple scales must be tested if object sizes change. While a reference part can be used to determine the scale and rotation in the Chains model, it may be difficult to reliably detect the reference part in complex videos. In contrast, our method is rotation invariant and it just uses a single pass optimization to handle a large range of scale changes.

The performance of previous methods also depends critically on the accuracy of the part detectors. If the human subject is moving quickly, e.g., a sprinter running, then motion blur may occur at the limbs and/or head, and the corresponding body part detectors may not detect these body parts reliably. In our method, we avoid costly part detection. Instead, we use the appearance of the directed walks between key points on the template to match the corresponding points in the target image. Matching the appearance of the directed walk on image pixels tends to be more resilient to motion blur than using body part detectors in a Pictorial Structure or Chains model. This observation is verified in our experiments with sports video sequences.

Our approach is also related to other methods of finding linear structures in images, e.g., to find roads [5, 3] and hands [7]. There is a key difference between finding a "walk on pixels" in our formulation and fitting a line in [5, 3]. In [5, 3] the stiffness of a curve is controlled by the curvature computed from consecutive non-overlapping pixel positions. However, this does not work for our model since a walk is allowed to revisit the same pixel multiple times– an undesirable property for line fitting but a critical setting in our method to simulate part foreshortening. At the same time, we penalize the walk along a path according to the duration of stay to discourage the walk shrinking to a single point. These properties require that the walk has longterm memory and forms a higher-order model, unlike [7] where the move along a path only relates to the previous state. In this paper, we show that the more complex model can still be solved efficiently by finding shortest paths on a properly constructed trellis.

Our method is also loosely related to random walk methods for image segmentation [13] and finding edges in cluttered images [6]. In these methods, the image cues used to bias the walks are generally simple, e.g. intensity boundaries. Instead, in our work we perform walks in a more guided setting, using richer information to control the quality of a walk, such as appearance, image gradients, and higher-order smoothness.

The key contribution of this work is a scale and articulation resilient, as well as rotation invariant, directed walk method for matching key points on articulated objects. This method efficiently searches over a huge number of template configurations in a single pass. We also propose an efficient method to find multiple walks. Experiments show that the proposed method gives several times smaller average matching error than competing methods in localizing key points on articulated objects in cluttered sports sequences.

## 2. Method

Given an exemplar image, the user will specify multiple pixel paths on the articulated object, where each path connects two key points on the articulated structure. On target images, we find similar paths by simulating a walk on image pixels. The sequence of footsteps on the walk matches the shape of the target structure.

After specifying a path on the exemplar image, i.e., a sequence of $N$ pixels, we extract a local appearance feature descriptor $\mathbf{t}_n$ for each pixel and we create the appearance template $\mathbf{T} = \{\mathbf{t}_1, \cdots, \mathbf{t}_N\}$. Our goal is to find the best matching path in the target image. Once the best matching path is found, then the corresponding end points of the path on the target image are deemed the key points on the articulated structure. For ease of exposition, we will first discuss our technique for optimizing a single walk and then we will propose an efficient method to find multiple walks simultaneously.

### 2.1. The Cost of a Walk

The cost of a walk is determined by the similarity of the pixel appearance on the path to the template walk ($E_a$), the conformance of the walk direction to the image local dominant direction ($E_d$), the smoothness of the walk ($E_t$), and how many times the walks stops ($E_s$).

We define some notation. An $N$ step walk $w$ is defined by $\{\mathbf{x}_1, \cdots, \mathbf{x}_N, \mathbf{d}_1, \cdots, \mathbf{d}_N\}$, where at the $n$th step, $\mathbf{x}_n$ is the pixel $xy$ coordinate in the target image and $\mathbf{d}_n$ is a unit vector representing the walk direction. We quantize the direction $\mathbf{d}_n$ into eight directions. The direction vector $\mathbf{d}_n$ is used to maintain *direction persistence*, i.e., if a step falls at the same pixel as the previous one, then $\mathbf{d}$ remembers the direction of the previous step that leads to the current pixel. We also define the neighborhood $\mathcal{N}(\mathbf{x}_n)$ of a pixel and each step should be within a neighborhood of the previous step.

The cost of a walk with $N$ steps is defined as:

$$E(\mathbf{x}_1, \cdots, \mathbf{x}_N, \mathbf{d}_1, \cdots, \mathbf{d}_N) = E_a + \alpha E_d + \beta E_t + \gamma E_s \quad (1)$$

where $\alpha, \beta$ and $\gamma$ are constant coefficients that determine the weight among different terms.

By minimizing $E$, the matching path on the target has four desirable properties. First, the appearance of the target path should be similar to the template. Second, the path should follow image local dominant orientations. Third, a path should not contain excessive twists and turns, and it should be laid out as straight as possible. Last, the *sojourn property* enables the path to consecutively revisit the same pixel thereby making the matched path visually shorter in the target image. This path shortening property allows our

method to be scale resilient and it is critical for handling object scale changes as well as body part foreshortening.

In the following, we cast the above energy minimization problem into finding shortest paths problem on a properly constructed trellis. At the same time, we define each cost term in Eq. (1).

### 2.2. Walk on a Trellis

We reduce the energy minimization problem to a special shortest path problem on a trellis. We will construct a trellis $G$ with a set of start vertices $S$ and end vertices $T$, so that a path starting from a vertex in $S$ and ending in a vertex in $T$ corresponds to a walk in the target image. Furthermore, we require that the cost of the "dynamic" path on the trellis should equal the cost of the corresponding walk in the target image.

Given a template path $\{\mathbf{t}_1, \cdots, \mathbf{t}_N\}$ and a target image $I$ with $|I|$ pixels, we define the structure of the trellis $G = (V, E)$. Let $Q$ be the set of quantized step direction vectors. The vertex set $V = \{v_{i,j,n}\}$ includes $|I| \times |Q| \times N$ vertices, where $i$ is the index for the pixels in the target image $I$, $j$ is the index for the quantized step direction vectors in $Q$, and $n$ is the position on the length $N$ template path. We define the set of edges as follows: there is an edge from $v_{k,l,n-1}$ to $v_{i,j,n}$ if

$$\mathbb{Q}(\mathbf{p}_i - \mathbf{p}_k) = \mathbf{q}_l, i \neq k, \mathbf{q}_l \in Q \text{ and } \mathbf{p}_i \in \mathcal{N}(\mathbf{p}_k) \quad (2)$$

where $\mathbf{p}_i$ is the $i$th pixel coordinate in image $I$ and $\mathbf{q}_l$ is the $l$th quantized unit direction vector in $Q$. $\mathbb{Q}(.)$ is a quantization function that maps a vector to the closest unit vector in $Q$, i.e., $\mathbb{Q}(\mathbf{x}) = \text{argmin}_{\mathbf{q}\in Q}\|\mathbf{x}/\|\mathbf{x}\| - \mathbf{q}\|$. In this paper, $|Q| = 8$. This indicates that if the direction from one point to another point conforms to the step direction vector and the two points are in a neighborhood, we link the corresponding vertices in the trellis $G$. We also add the edges $(v_{k,l,n-1}, v_{i,j,n})$ when $i = k$, $j = l$ and this corresponds to the case when the step stays at the same pixel. Note that for stationary steps, the walk direction is kept the same between the linked vertices in two successive trellis layers. This setting is important because it enforces the direction persistence property. The set of start vertices $S = \{v_{i,j,1}\}$ and end vertices $T = \{v_{i,j,N}\}$.

To complete the trellis construction, we assign weights to the vertices and edges based on the walk costs. The terms $E_a$ and $E_d$ correspond to weights on vertices:

$$\begin{aligned} &\pi(v_{i,j,n}) = E_a + \alpha E_d, \\ &E_a = e(\mathbf{c}(\mathbf{p}_i), \mathbf{t}_n),\ E_d = |\mathbf{q}_j \cdot \mathbf{g}(\mathbf{p}_i)| \end{aligned} \quad (3)$$

where $\mathbf{c}(\mathbf{p}_i)$ is the feature vector at pixel location $\mathbf{p}_i$. And, recall that $\mathbf{t}_n$ is the template path feature vector at stage $n$. $e(.)$ is a distance function. In this paper, we use the Euclidean distance function and simple pixel values as the features; other features can also be used in the same formulation. The dominant gradient orientation $\mathbf{g}(\mathbf{p}_i)$ at point $\mathbf{p}_i$ is set as the direction of the long axis of the local structure tensor. The coefficient $\alpha$ is defined in Eq. (1). The node weight approaches zero if the walk follows template appearances and image local orientations. The walk smoothness $E_t$ corresponds to the weight on the edges

$$\pi(v_{k,l,n-1}, v_{i,j,n}) = \beta E_t, \quad E_t = \|\mathbf{q}_j - \mathbf{q}_l\|, \quad (4)$$

if $i \neq k$. Here we use the notation of $\pi$ for both node weight and edge weight. The edge weight quantifies the walk direction changes, where $\beta$ is the constant coefficient defined in Eq. (1). Such a setting makes the walk mostly straight but it allows a small number of turns.

For the trellis edges $(v_{k,l,n-1}, v_{i,j,n})$ corresponding to stationary steps, we have $i = k, j = l$. For these edges, we set the weight to a small constant $\gamma$, which is the coefficient for $E_s$ in Eq. (1). A step thus may copy itself or stay at the same location but it involves a constant penalty. The penalty will accumulate if a walk stays at the same point for too long and is encouraged to take some action. This is equivalent to penalizing the total number of stationary steps, using the term

$$E_s = \sum_{n=2}^{N} \mathbb{I}(\|\mathbf{x}_n - \mathbf{x}_{n-1}\|) \quad (5)$$

where the indicator function $\mathbb{I}(x) = 1$ if $x = 0$ and otherwise $\mathbb{I}(x) = 0$.

Based on the above settings, the directed walks in the target image and the paths from $S$ to $T$ in the trellis form a one-to-one mapping. And, the cost of each walk equals the summation of the node weights and edge weights along the corresponding path in the trellis.

### 2.3. Optimization

We compute the minimum cost of walks that end at each image point by finding the "dynamic" shortest paths that terminate at vertices in the last layer of $G$. Let cost $C_{i,j,n}$ be the min-cost of walks that terminate at node $v_{i,j,n}$, then

$$\begin{aligned} C_{i,j,n} &= \min_{\forall (k,l)\in\mathcal{P}} \{\pi(v_{k,l,n-1}, v_{i,j,n}) + u_{i,j,k,l,n} + C_{k,l,n-1}\} \\ u_{i,j,k,l,n} &= \begin{cases} D_{k,l,n-1} & \text{if } i = k \text{ and } j = l \\ \pi(v_{i,j,n}) & \text{otherwise} \end{cases} \end{aligned} \quad (6)$$

where $\mathcal{P} = \{(k,l)|v_{k,l,n-1} \text{ connects to } v_{i,j,n}\}$ and $D$ memorizes the stationary step vertex costs. Similar to standard dynamic programming for finding shortest paths on a trellis, we keep a backward pointer to indicate the vertex selection in the minimization. Apart from $C$, the memory map $D$ also needs to be updated. We set $D_{i,j,n} = D_{i,j,n-1}$ if the backwards pointer of vertex $v_{i,j,n}$ points to $v_{i,j,n-1}$, and otherwise $D_{i,j,n} = \pi(v_{i,j,n})$. This ensures that the matching

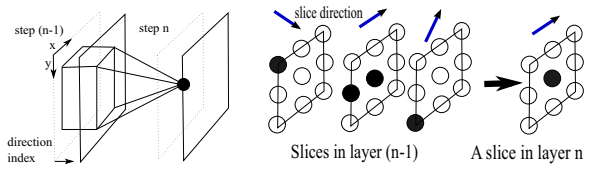


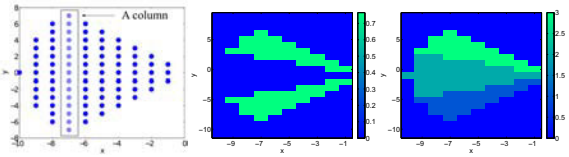

Figure 2. Left: Backward links in min-convolution. Right: An example that shows the walk direction slices. The black vertices are linked together in the trellis.

algorithm can skip steps along the walk template if shortening the walk is needed, e.g., due to foreshortening. Note that due to the penalty accumulation for stationary steps, the walk will not shrink to a single point. To initialize, we set $C_{i,j,1} = D_{i,j,1} = \pi(v_{i,j,1})$. This indicates that at these starting points we prefer pixels that have similar appearance to the template start point. The first steps in the walk have equal chance to go in all possible directions.

Let's take a closer look at the computation to obtain $C_{i,j,n}$. The computation of $\pi(v_{k,l,n-1}, v_{i,j,n})$ takes two different forms: if $k \neq i$, $\pi$ quantifies the walk direction changes and if $k = i$ and $l = j$, $\pi$ equals a small constant. We thus need to find all the neighborhood points that can reach point $i$ in one step and we add their costs with a predefined edge link weight defined by the direction difference penalty or a small stationary penalty. We then find the minimum of the weighted costs. Fig. 2 illustrates the range of nodes involved in the computation for $C$, when a rectangular neighborhood is used. If we replace pixel index $i$ with its $xy$ coordinates, the computation is a 3D min-convolution [8]. The 3D min-convolution kernel in the $xy$ dimensions has the same size as the maximum step size and its third dimension indicates the walk directions. In a simple implementation, the min-convolution's complexity is proportional to the square of the radius of the pixel neighborhood. In the following we show how to speed up the computation by using the special structure of the problem.

We consider $|Q|$ kernels, each of which corresponds to one quantized direction $j$ at a vertex $v_{i,j,n}$. These kernels are related to each other by rotations. We thus only need to show that there is an efficient method to compute the min-convolution for a canonical kernel. Here, we use the canonical kernel that corresponds to a node with a left-to-right step direction. We further restrict the turning angle at each step to be at most 45 degrees. We now have a reduced pie-slice kernel. Fig. 3(a) shows an example kernel that corresponds to a maximum step size of 10. Note that the kernel does not include the point at the current step location. To merge it with the min-convolution result, we need one additional min operation. Fig. 3(b) shows the values of the min-convolution kernel and Fig. 3(c) shows the correspondence of the kernel elements to the rotation directions.

Since we have quantized the walk directions, the min-convolution kernel has a piecewise constant structure. For fast computation, we decompose the kernel into columns, e.g., the columns of the kernel in Fig. 3(a). Each column




(a) (b) (c)

Figure 3. (a) The top-down view of a min-convolution kernel with radius 10. (b) The values of the min-convolution kernel. (c) The direction indices of the min-convolution kernel. The shading in (c) indicates the slice indices.

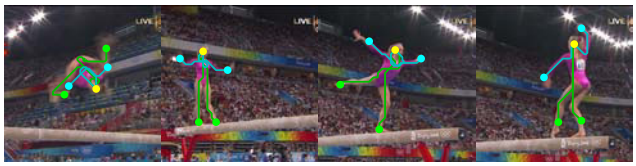


Figure 4. Matching a gymnast using the exemplar in Fig. 1. Key point matching results are color coded (yellow: head, cyan: hands, and green: feet). The lines are optimal walk paths.

kernel corresponds to a set of walk direction slices and for each slice the kernel values are constant due to quantization. We perform the 1D min-convolution on each slice and merge the result by a final minimization to obtain the 3D min-convolution result. By using [12], the 1D per-slice min-convolution has a complexity independent of the 1D kernel length. Therefore, the complexity of the proposed method is $O(r)$, where $r$ is the radius of the pixel neighborhood, in contrast to $O(r^2)$ when we directly compute the 3D min-convolution.

### 2.4. Finding Multiple Structures

We have proposed methods to match a single walk to an articulated structure in an input image and determine the end points. In many situations, we need to determine the end points of multiple linear structures simultaneously. For instance, finding two hands or two feet requires that we jointly find two walks that have low costs and at the same time tend to be apart from each other. Using the terminal potential map of walk $i = 1...k$, we extract $M_i$ candidate walks using non-minimum suppression and then we select $k$ optimal walks from them, one from each of the $M_i$ candidates. Due to the coupling among walks, when $M_i$ and $k$ are large, exhaustive search soon becomes impractical. In the following, we propose an efficient implicit enumeration method to find $k$ optimal walks.

Finding multiple walks can be formulated as an assignment problem, in which we associate candidate walk $f_i$ to structure $i$. We minimize the overall local and coupling cost,

$$\min_{f}\{\sum_{i=1}^{k} c(i, f_i) + \sum_{\{i,j\}\in\mathcal{B}} h(f_i, f_j)\}. \qquad (7)$$

The first term is the unary term, where $c(i, f_i)$ is the cost of assigning walk $f_i$ to structure $i$. $c(.)$ is obtained from

the walk cost map $C$. The second term incorporates two costs: the first is the distance between the start point of two walks $f_i$ and $f_j$ if structure $i$ and $j$ should start from the same point, and the second is a positive penalty if walks $f_i$ and $f_j$ have a distance less than a threshold, otherwise it is zero. $\mathcal{B}$ is the set of structure pairs. The pairwise regularization term $h(.)$ pushes structures apart and encourages the selection of walks at different locations. Note that the pairwise constraint is soft and therefore allows overlapping walks. Here, the distance between two walks is defined as $\min(\min d(\mathbf{e}(f_i), \mathbf{p}(f_j)), \min d(\mathbf{e}(f_j), \mathbf{p}(f_i)))$, where $\mathbf{e}(f_i)$ is the end point of walk candidate $f_i$ and $\mathbf{p}(f_j)$ denotes all points on the walk of $f_j$ and $d(.)$ is the distance function. Thus, if one walk embeds in the other, their distance is zero. We convert the optimization into a linear one.

We introduce variable $\xi_{i,n}$ to indicate whether structure $i$ maps to the candidate path $n$. $\xi_{i,n}$ is 1, if the mapping is true and 0 otherwise. If the cost of mapping structure $i$ to $n$ is $c_{i,n}$, the total local mapping cost is $\sum_{i=1}^{k}\sum_{n=1}^{M_i} c_{i,n}\xi_{i,n}$, where $c_{i,n}$ is obtained from $c(.)$ in Eq. (7). Since each linear structure on the target needs to map to a walk candidate, $\xi$ is constrained by $\sum_{n=1}^{M_i} \xi_{i,n} = 1, \forall i$.

To linearize the pairwise penalty term, we introduce another variable $\eta_{i,j,n,m}$ that is 1 if structure $i$ selects candidate $n$ and $j$ selects $m$, and otherwise 0. $\eta$ is thus a pairwise indicator variable. It is related to $\xi$ by $\eta_{i,j,n,m} \geq \xi_{i,n} + \xi_{j,m} - 1, \eta_{i,j,n,m} \leq \xi_{i,n}, \eta_{i,j,n,m} \leq \xi_{j,m}$, which enforces that $\eta_{i,j,n,m}$ is 1 if both $\xi_{i,n}$ and $\xi_{j,m}$ are 1, and otherwise 0. With $\eta$, the pairwise term in the objective function can be written as $\sum_{i,j,m,n} h_{i,j,n,m}\eta_{i,j,n,m}$. Here, $h_{i,j,n,m}$ is the penalty term and can be computed from $h(.)$ in Eq. (7).

Combining the linear local cost term and the pairwise cost term with the constraints on $\xi$ and $\eta$, we obtain an exact reformulation of the optimization.

$$\min\{\textstyle\sum_{i,n} c_{i,n}\xi_{i,n} + \sum_{i,j,n,m} h_{i,j,n,m}\eta_{i,j,n,m}\} \quad (8)$$
$$\text{s.t.} \quad \textstyle\sum_{n=1}^{M_i} \xi_{i,n} = 1, \forall i,\ \eta_{i,j,n,m} \geq \xi_{i,n} + \xi_{j,m} - 1,$$
$$\eta_{i,j,n,m} \leq \xi_{i,n},\ \eta_{i,j,n,m} \leq \xi_{j,m},\ \forall\{i,j\} \in \mathcal{B},$$

where $\xi$ and $\eta$ are binary. The binary integer linear program is not submodular but we still would like to find its global optimal solution. One option is Balas' implicit enumeration method [9]. However, Balas' method is not the most efficient due to the large number of $\eta$ variables.

Using the special structure of the problem, we construct a special branch and bound method for implicit enumeration. The basic idea is to implicitly enumerate on only the $\xi$ variables or equivalently enumerate all the possible walk assignments. Since we do not enumerate $\eta$ variables, our method is more efficient than the standard Balas' method. Algorithm 1 illustrates the special branch and bound algorithm in a depth first search fashion.

In Algorithm 1, array $w$ stores the chosen candidate for each walk. At the beginning, index $i$ is set to 0. The upper

**Algorithm** 1. Finding Multiple Walks

```
global input/output variable: walks, ubound
procedure multiwalk(w[1..k], k, i)
if i = k then
    Update ubound and the optimal walks, return
else
    lbound = cost of determined walk assignment
                 in w[1..i]
    if lbound > ubound then
        The branch is pruned away, return
    else
        Increment i by 1
        foreach walk candidate j in 1..M_i do
            w[i] ← j
            multiwalk(w, k, i)
        end
    end
end
```

bound $ubound$ is initially computed using the lowest cost candidate from each walk candidate set, and $walks$ are initialized accordingly. The upper bound $ubound$ and optimal $walks$ are updated when a leaf node is reached in the search tree. The lower bound $lbound$ at each search tree node is computed using Eq. (8). We set $\xi = 1$ for walks that are determined and $\xi = 0$ otherwise. We set $\eta_{i,j} = x_i \& x_j$, where $\&$ is the logical bitwise "and" operator. The cost of the partial labeling is a lower bound because all the coefficients in the cost function are non-negative, which means any way of expanding the search tree from the current search tree node to include more walk assignments will increase the energy. The search terminates and backtracks if (1) the estimated lower bound is greater than the current upper bound, or (2) we complete $k$ levels and reach the leaf node. The method terminates when the search tree is exhausted. This method is efficient even though the worst case complexity is exponential; it terminates in a fraction of a second for $k = 10$ and $N = 100$ on a 2.8GHz machine.

Fig. 4 illustrates the proposed method matching a gymnast in a fast motion balance bar sequence. The exemplar has two walk templates as shown in Fig. 1: One from head to a foot and the other from head to a hand. Due to the very cluttered background and similar foreground colors, simple skin color matching is not able to find the target point. The proposed method reliably finds the end points in spite of scaling, rotation, articulation and strong motion blur.

## 3. Experiments

We applied our proposed method to detect key points on a variety of deformable objects in video sequences. In each video sequence, one video frame is chosen at random for the user to define a template walk. Given this template, its key points are then automatically detected in all the other frames of the video sequence. As formulated in Section 2 our method is applied independently in each frame, i.e., there is no temporal model employed.

Figure 5. Matching objects using the proposed method (row 1, 3, 5) and deformable matching [14] (row 2, 4, 6). The detection rates of this paper for the 570-frame sock, 193-frame goose and 930-frame fish sequences are 98%, 96% and 94% respectively, while the detection rates of [14] are 16%, 15% and 52% for the three sequences.

Figure 6. Examples of hand matching using our method on the gesture sequences of [4].

**Three Test Videos of Deformable Objects:** The first experiment considers three video sequences of different deformable objects: a cloth sock, a moving goose, and a swimming fish. All these sequences contain complex deformation, scaling, and rotation of the target objects. Fig. 5 shows sample results of the proposed method (odd rows) and the method of [14] (even rows). In [14], pixel color and SIFT are used as features. In this experiment, a key point is deemed correctly localized if it is within 10 pixels from the ground truth. The performance of deformable matching [14] degrades when object deformation increases. We achieve a 96% average detection rate for the test sequences, while the average detection rate for [14] is 36%. The proposed method matches the target reliably even though there are large deformation, rotation and scaling.

**Hand Gesture Videos:** The second experiment considers the gesture dataset of [4], where we use their self-training (ST) sequences. In this experiment, the walks are from head to hand. Sample frames are shown in Fig. 6. These are grayscale videos; therefore, the appearance term $E_a$ is computed based on pixel intensity values. For this dataset, our method achieves a detection rate of 84%, which is close to the 87% detection rate reported in [4]. In the self-trained scenario of [4], the first 1000-1500 frames of each movie sequence in the dataset were used for training and the rest were used for testing. In contrast, our method only requires a single exemplar image and a few user strokes to define the template walks.

| Matching Mean Error (Pixels) | | | | | | | |
|---|---|---|---|---|---|---|---|
| V | Part | TP | MC | [11] | [2] | [10] | [15] |
| B-I | Head | **3.40** | 11.97 | 20.15 | 58.26 | 63.49 | 5.65 |
| | Hands | **10.56** | 12.12 | 49.93 | 88.05 | 77.11 | 16.48 |
| | Feet | **5.82** | 20.55 | 44.56 | 58.39 | 85.65 | 10.54 |
| B-II | Head | **3.18** | 7.42 | 21.69 | 66.09 | 94.95 | 8.80 |
| | Hands | **9.43** | 11.56 | 55.05 | 74.06 | 101.14 | 17.74 |
| | Feet | **9.97** | 19.25 | 38.89 | 122.98 | 98.49 | 14.01 |
| Gym | LT-Joint | **2.44** | 2.59 | 59.78 | 45.11 | 36.77 | 8.11 |
| | Feet | **7.18** | 13.19 | 61.24 | 121.43 | 77.67 | 11.15 |
| Dive | Head | **3.96** | 25.27 | 71.58 | 52.49 | 73.12 | 22.26 |
| | Feet | **6.30** | 12.80 | 92.26 | 77.42 | 92.79 | 21.89 |
| LJump | Head | **4.26** | 6.85 | 28.63 | 55.26 | 71.30 | 12.49 |
| | Knee | **6.19** | 11.84 | 64.30 | 58.85 | 71.95 | 12.19 |
| Dance | Head | 1.61 | 5.75 | 5.36 | 43.27 | 62.99 | **1.41** |
| | Hands | **3.19** | 9.40 | 33.32 | 56.18 | 58.48 | 6.99 |
| | Feet | **2.78** | 8.54 | 19.82 | 15.32 | 76.23 | 3.08 |

| Matching Error Standard Deviation (Pixels) | | | | | | | |
|---|---|---|---|---|---|---|---|
| V | Part | TP | MC | [11] | [2] | [10] | [15] |
| B-I | Head | **5.14** | 12.35 | 32.06 | 46.15 | 64.54 | 9.49 |
| | Hands | **7.47** | 8.28 | 30.78 | 44.24 | 54.49 | 10.73 |
| | Feet | **5.99** | 10.51 | 48.82 | 26.31 | 41.65 | 9.67 |
| B-II | Head | **5.45** | 6.39 | 35.03 | 47.95 | 61.06 | 15.84 |
| | Hands | **7.20** | 8.13 | 29.03 | 43.95 | 55.96 | 14.31 |
| | Feet | **10.24** | 10.51 | 41.28 | 46.87 | 48.20 | 18.92 |
| Gym | LT-Joint | **2.41** | 2.73 | 31.51 | 28.99 | 27.84 | 13.19 |
| | Feet | **5.33** | 9.26 | 50.79 | 39.09 | 50.97 | 18.74 |
| Dive | Head | **4.78** | 13.92 | 45.46 | 55.24 | 41.66 | 18.58 |
| | Feet | **7.39** | 10.24 | 57.61 | 60.27 | 59.27 | 21.02 |
| LJump | Head | **4.26** | 6.85 | 28.63 | 55.26 | 71.30 | 12.49 |
| | Knee | **6.19** | 11.84 | 64.30 | 58.85 | 71.95 | 12.19 |
| Dance | Head | 0.82 | 8.38 | 8.53 | 18.81 | 37.70 | **0.74** |
| | Hands | **3.45** | 6.31 | 22.27 | 23.53 | 24.42 | 5.64 |
| | Feet | 3.95 | 5.95 | 26.25 | 7.99 | 31.66 | **2.78** |

Figure 8. Comparison of mean errors and standard deviation on the six sports sequences. TP is the proposed method and MC is the modified Chains method.

**Sports and Dancing Videos:** The final experiment considers videos that depict fast moving dancers and athletes. These videos are particularly challenging, due to uncontrolled camera movements, abrupt video cuts, large lighting changes and a lot of motion blur. Within a single video sequence, the dancers or athletes can undergo significant rotations and their scales can change substantially. Motions in these sequences can be abrupt; therefore, employing motion continuity constraints in tracking is unadvisable.

Example images from these test sequences are shown in Fig. 9. The first image in each row (shown with blue border) depicts the exemplar image and stroke template with keypoints. Subsequent images in each row show the key points detected using our method. The few failure cases are mostly due to similar structure in the background.

Using this same dataset, we have conducted a quantitative comparison with five state-of-the-art methods in the literature: a modified version of the Chains method of [4], the Pictorial Structures (PS) detector of [11], a tracker based

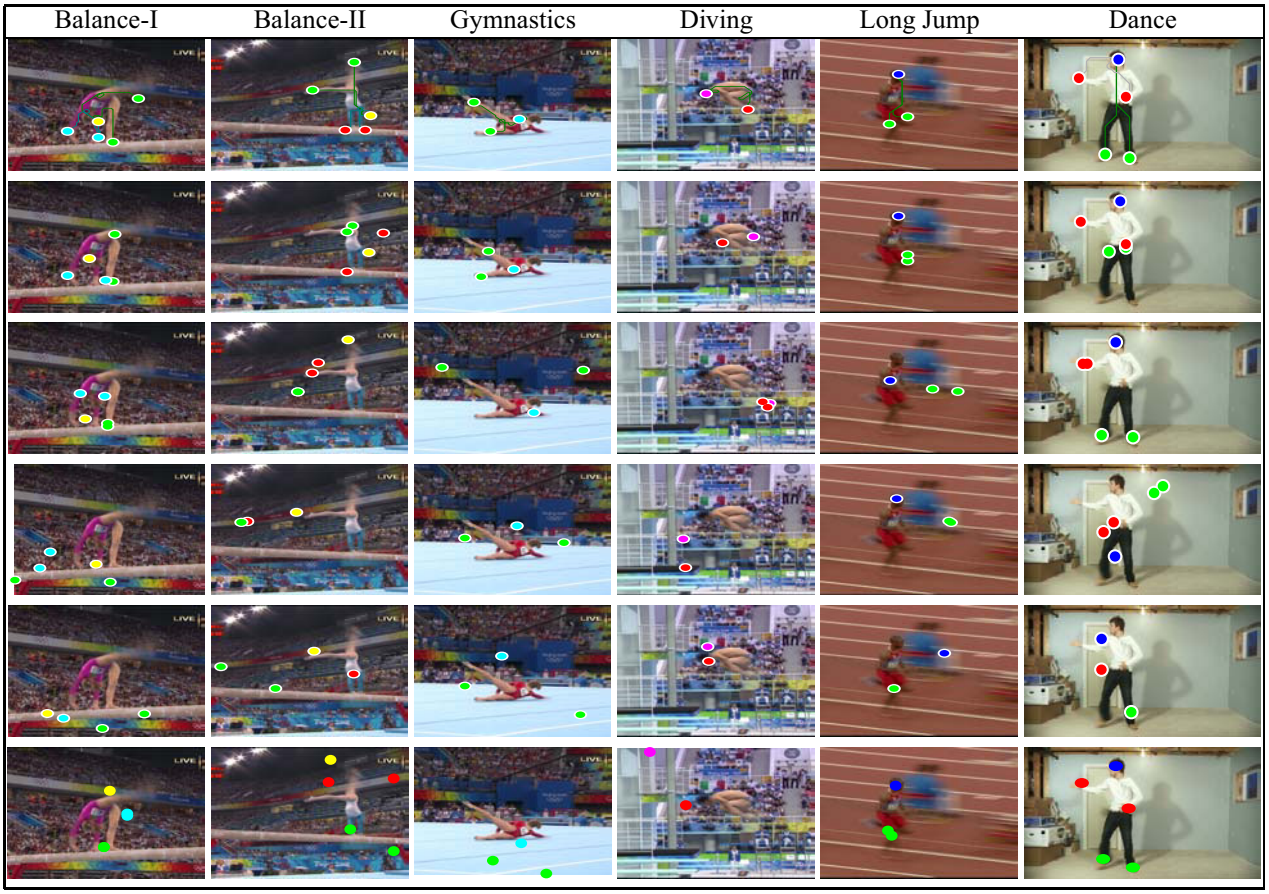



Figure 7. Sample results on finding key points on human subjects in the six test videos. Row 1: Our method. Row 2: Modified chains method [4]. Row 3: Person detector [11]. Row 4: Person tracker [2]. Row 5: Pose detector [10]. Row 6: A recent 26-part person detector[15].

on PS [2], a PS detector that incorporates a two-stage optimization [10], and a recent PS method that uses 26 body parts [15]. Before presenting the results of this comparison, we give some details of the experimental setup.

Since the Chains method [4] cannot be applied when there is a single exemplar, we only use its optimization model. Our modified version of the Chains method employs a small set of chains with different lengths and all the chains use the same features as our approach. The spatial transitional probability from vertex to vertex is set so that it is high if the model appearance is matched. We run the chain matching on different length models and marginalize the probability.

Details of the experimental setup for the PS models are as follows. The first PS method of [11] uses strong body part detectors. For a fair comparison, we incorporate color information into the local part detection. We linearly combine the color matching map with the shape context matching map to form the local part detection map for this method. We adjust the combining ratio to achieve its best performance. The second PS method of [2] uses the exemplar for each sequence to train the color model. The PS method of [15] uses 26 body parts and to be fair, color appearance is included in the PS model local part detectors. Thus, except for [10] all the competing methods we tested use the color model from the same exemplar as the proposed method. The PS models we compared are scale dependent except the 26-part PS model [15]. In our experiments we assume that these PS methods have the advantage of knowing the correct scale and use it to normalize the image sizes before part detection and matching.

Sample results of the competing methods and our method are shown in Fig. 7. A quantitative comparison of the mean matching errors and their standard deviation on each sequence for each method are shown in Fig. 8. Our method achieves consistently better average matching error in nearly all the tests. In the dance sequence, the dancer has mostly upright poses and the 26-part PS method has better head detection than the proposed method, while the proposed method has lower mean errors on the more challenging hand and foot detection.

The proposed method is efficient. The typical running time for detecting head, feet and hands of a human subject is 5 seconds per frame in our experiments on a 2.8GHz machine, which is comparable to [15] that runs in 4 seconds, while the method of [2] takes about 10 seconds, [10] takes

Figure 9. Sample matching results of the proposed method on sports and dance sequences. Row 1: 410-frame *balance-I*. Row 2: 459-frame *balance-II*. Row 3: 170-frame *gymnastics*. Row 4: 238-frame *diving*. Row 5: 178-frame *long jump*. Row 6: 426-frame *dance*. The first image in each row shows the exemplar image overlaid with template walk paths. These test sequences have complex body part articulation, large rotation, scale changes, cluttered backgrounds and blurred imagery. The proposed method matches reliably in spite of these challenges.

60 seconds and [11] takes 90 seconds for each scale.

## 4. Conclusion

We propose a novel method for matching articulated objects in cluttered videos by walking on pixels. The new formulation enables scale resilient and rotation invariant matching. Working directly on pixels, it combines feature extraction and structure matching in an integrated framework. Our experiments on a variety of videos confirm that the proposed method is reliable and efficient. It is able to find key points on fast moving articulated objects in very cluttered videos and shows superior performance to competing methods. It is also much faster than most pictorial structure methods. We believe the proposed method is useful for many applications including object tracking, movement analysis and action recognition.

**Acknowledgment**: This research is supported by US NSF grants 1018641, 1029430 and 0855065.

## References

[1] P.F. Felzenszwalb, D.P. Huttenlocher, "Pictorial structures for object recognition", IJCV 61(1), Jan. 2005. 1

[2] D. Ramanan, D. A. Forsyth, A. Zisserman. "Strike a pose: tracking people by finding stylized poses." CVPR 2005. 1, 6, 7

[3] N. Merlet, J. Zerubia, "A curvature-dependent energy function for detecting lines in satellite images", 8th SCIA Conf. 1993. 2

[4] L. Karlinsky, M. Dinerstein, D. Harari, S. Ullman, "The chains model for detecting parts by their context", CVPR 2010. 1, 6, 7

[5] A. Gruen, H. Li, "Semi-automatic linear feature extraction by dynamic programming and LSB-snakes", Photogrammetric engineering and remote sensing, vol. 63, no. 8, 1997. 2

[6] Q. Zhu, G. Song, J. Shi, "Untangling cycles for contour grouping", ICCV 2007. 2

[7] J. Wang, V. Athitsos, S. Sclaroff, M. Betke, "Detecting objects of variable shape structure with hidden state shape models", TPAMI, vol.30, no.3, 2008. 2

[8] L. Babai, P. Felzenszwalb, "Computing rank convolutions with a mask", ACM Transactions on Algorithms, Vol. 6, No. 1, 2009. 4

[9] E. Balas, "An additive algorithm for solving linear programs with zero-one variables", Operations Research, 13(1965), 517-545. 5

[10] D. Ramanan, "Learning to parse images of articulated objects", NIPS 2006. 1, 6, 7

[11] M. Andriluka, S. Roth, B. Schiele, "Pictorial structures revisited: people detection and articulated pose estimation", CVPR 2009. 1, 6, 7, 8

[12] J. Gil, M. Werman, "Computing 2-D min, median, and max filters", TPAMI, vol.15, no.5, 1993. 4

[13] L. Grady, "Random walks for image segmentation", TPAMI, vol.28, no.11, 2006. 2

[14] H. Jiang, S.X. Yu and D.R. Martin, "Linear Scale and Rotation Invariant Matching", TPAMI, vol.33, no.7, 2011. 6

[15] Y. Yang, D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts", CVPR 2011. 1, 6, 7