European Conference on Computer Vision 2008 (ECCV'08)

Finding Actions Using Shape Flows

Hao Jiang and David R. Martin

Computer Science Department, Boston College, Chestnut Hill, MA 02467, USA
{hjiang,dmartin}@cs.bc.edu

Abstract. We propose a novel method for action detection based on a new action descriptor called a *shape flow* that represents both the shape and movement of an object in a holistic and parsimonious manner. We find actions by finding shape flows in a target video that are similar to a template shape flow. Shape flows are largely independent of appearance, and the match cost function that we propose is invariant to scale changes and smooth nonlinear deformation in space and time. The problem of matching shape flows is difficult, however, yielding a large, non-convex, integer program. We propose a novel relaxation method based on *successive convexification* that converts this hard program into a vastly smaller linear program: By using only those variables that appear on the 4D lower convex hull of the matching cost volume, most of the variables in the linear program may be eliminated. Experiments confirm that the proposed shape flow method can successfully detect complex actions in cluttered video, even with self-occlusion, camera motion, and intra-class variation.

1 Introduction

An action can be characterized by the movement and deformation of a shape. A *flow line*, which is the space-time line formed by a tracked object point over time, provides a compact representation of motion. An object's *shape flow*, which we define simply as an assembly of flow lines, represents not only the object's motion but also its shape and deformation over time. Flow lines have previously been used for motion visualization [5]. We propose the shape flow as a representation for actions.

Consider the shape flows shown in Fig. 1. One can readily identify complex actions based on the shape flow alone, suggesting that it is a discriminative and descriptive representation. The shape flow itself is quite simple to compute. The challenge, which is the focus of this paper, is how to use the shape flow as a representation for actions. In particular how can we efficiently search for a template shape flow in a cluttered single-view video? And how can we do the search in a manner that is invariant to scale changes and nonlinear shape deformations, and also tolerant of occlusion and intra-class variation? Although shape flow matching is certainly an NP-hard problem because of the loopy relations between flow lines, we show that a novel linear relaxation based on *successive convexification* [6] yields both an efficient and accurate matching algorithm for finding actions in video.

There is much related work on detecting actions in video. The first dimension of related work consists of methods that use multi-view stereo to reliably access 3D spatial information. Parameswaran & Chellappa [12] use multiple cameras to capture joint

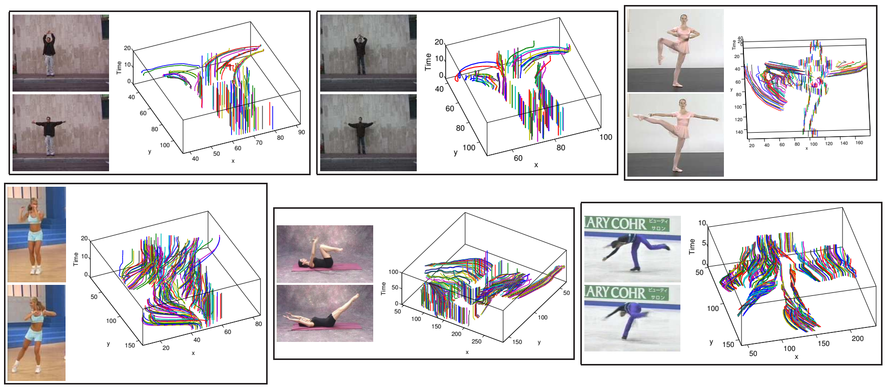


Fig. 1. Example shape flows and first/last frames for a variety of actions. Even though individual flow lines are noisy, the shape flow represents the holistic shape *and* movement of the object reliably. In this paper, we show how to efficiently search for a template shape flow in a target video. *Please note that the figures in this paper are best viewed in color.*

locations via motion capture, while Yilmaz & Shah [20] use manually labeled joint locations. Weinland et al. [18] use multiple cameras to achieve reliable background subtraction, using 3D silhouettes of foreground objects to describe actions. These methods are only tangentially related to this paper, however, as they rely on multi-view stereo.

The second dimension of related work includes methods that rely on background subtraction to formulate action features from silhouettes of foreground objects either in 2D or in 3D. Weinland et al. [18] use 3D silhouettes from multi-view stereo, as mentioned above. Both Yilmax & Shah [19] and Blank et al. [2] use background subtraction in 2D images to describe actions as 3D space-time volumes. Bobick & Davis [3] project space-time silhouettes into the image to get 2D action silhouettes. In this paper, we address the problem of finding actions in cluttered single-view video with a moving camera and moving background objects. Silhouette based features, which rely on high quality background subtraction, are difficult to use in this regime.

The third dimension of related work involves using labeled joint trajectories to detect human actions. In this class-specific approach, joint locations may be labeled in different ways. Parameswaran & Chellappa use motion capture and manual labeling for joint locations in 2D and 3D in their work [12]. Yilmaz & Shah [20] use manually labeled joint trajectories in 3D, and Sheikh & Shah [17] use manually labeled joint trajectories in 2D. Recognizing actions is challenging even with clean joint trajectories. Although automatic human pose tracking is recently much improved [11, 13] and so could be used to extract (unclean) joint trajectories from unlabeled video, we pursue a non-parametric approach of analyzing the whole motion field rather than the motion of distinguished high-level feature points.

The final dimension of related work consists of non-parametric action models, and contrasts methods that rely on sparse descriptors located at interest points versus methods that use a dense motion or gradient field. These methods typically operate on uncalibrated single-view video, do not require background subtraction, and do not require manual labeling of foreground objects. Laptev & Lindeberg [9] have extended the notion of Harris extrema to find space-time interest points. Schuldt et al. [14] use motion and gradient histograms around these interest points to recognize actions using SVMs. Scovanner et al. [15] extend the popular SIFT descriptor to space-time volumes for the purpose of action recognition. Instead of constructing features that are tolerant to clutter, one may instead use dense feature fields along with clutter-tolerant matching. Efros et al. [4] recognize actions using optical flow fields from single frames, carefully smoothed and rectified. Shechtman & Irani [16] cleverly match space-time volumes directly, without any explicit motion estimate. Ke et al. [7] have extended that work using superpixels and part-based matching. Laptev & Prez [10] match histograms of gradients from space-time cubes using a boosted cascade.

These non-parametric methods represent a recent trend toward using volumetric space-time descriptors that combine shape *and* motion information, along with a matching framework that tolerates both foreground deformation and background clutter. We adopt this general approach to finding actions, although our proposed shape flow descriptor is neither an interest point method nor a dense field approach. As is clear in Fig. 1, the shape flow preserves much information about both the shape and movement of an object in a manner that is largely independent of the object's appearance. This enables us to match both shape and motion in a uniform framework.

The outline of the paper is as follows. In §2 we describe a simple method to compute shape flows. In §3, we formulate the shape flow matching problem and show how this NP-hard problem may be solved efficiently using a novel relaxation scheme. We present experimental results in §4 and conclude in §5.

2 The Shape Flow of an Action

We desire a compact yet expressive descriptor of both the shape and motion of an object. In addition, we seek a general method that may be applied to any object class, so we do not attempt to track distinguished points (such as joints) or to impose a shape model a priori. Instead, we seek a non-parametric shape and motion representation. We adapt the technique of flow fields from motion visualization [5]. If points on an object may be tracked in 2D video, then their positions through time form a flow field of lines in 3D. Flow lines will inevitably be individually unreliable, but the collection of flow lines—the *shape flow*, as shown in Fig. 1—is a compact and descriptive action representation.

We use a greedy scheme to compute flow lines based on iterative conditional modes (ICM) [1] to estimate the sparse point motion between adjacent frames. ICM is based on an MPEG-like local motion estimation search; it computes motion vectors that both minimize an image match cost and maximize the the motion consistency of neighboring points. We apply ICM to edge pixels that surpass a Canny detector threshold; neighborhood relations are defined by the Delaunay triangulation of these edge pixels. The resultant sparse motion field is then interpolated across Delaunay cells to produce a dense motion field. The frame-by-frame motion fields produced by this procedure are then simply concatenated to form 3D flow lines in the space-time video volume. There are no constraints to prevent flow lines from intersecting.

This method for computing flow lines is designed to produce flow lines that are good enough on average to generate a coherent flow field, since our robust matching procedure tolerates flow line errors. Although the method is greedy, one can see from

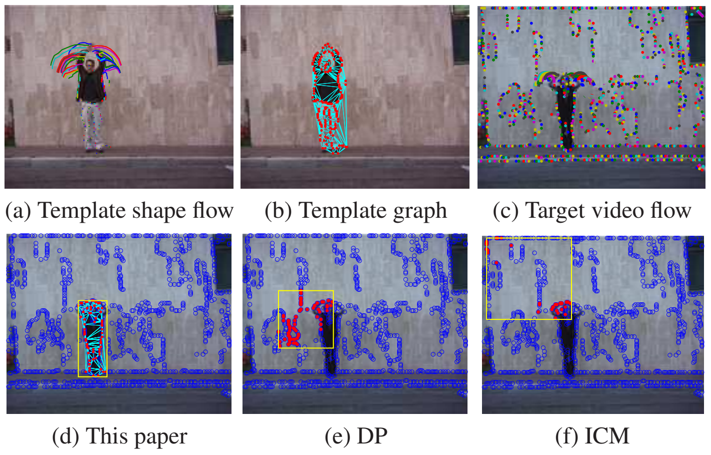


Fig. 2. Matching shape flows. A template shape flow (a) and its neighbor relation graph (b) are matched to a target video (c). Note that stationary points in (a) and (c) produce flow lines orthogonal to the page, which are depicted as dots. The matching result using the proposed method of this paper (d) is superior to the result achieved either by dynamic programming (e) or iterative conditional modes (f). In (d-f), the blue dots denote the start points of target candidate flow lines.

Fig. 1 that the flow lines are actually quite clean. More accurate but computationally more expensive methods such as [8] could be used to improve the flow line extraction.

Although the collection of flow lines that we compute is dense, for computational considerations we sample the field in both the template and the target. The spatial localization of flow lines in a shape flow is therefore somewhat imprecise. However, any localization error introduced by this sampling process is overwhelmed by variation caused by viewpoint changes and intra-class shape and motion variation. We now turn to the shape flow matching procedure.

3 Matching Shape Flows for Action Detection

In this section, we address the problem of matching a shape flow to a target video. We desire the matching to be scale invariant, operate in significant clutter, and also to be tolerant to spatial and temporal deformations caused by viewpoint variation and intra-class variation. The problem is made more difficult by the fact that the relationships between flow lines in a shape flow, which are defined by a Delaunay triangulation, form a loopy graph. We present a novel relaxation method that can accomplish effective and efficient matching using a compact linear program.

3.1 An Example

Before describing the matching procedure, we present the matching example shown in Fig. 2 in order to elucidate both the problem statement and the shape flow representation. Figs. 2(a,b) show the top-down view (projected along the time axis) of a template shape flow for a person waving both arms. Note that the shape flow consists of a collection of flow lines (a) related by a neighborhood graph (b) given by the Delaunay

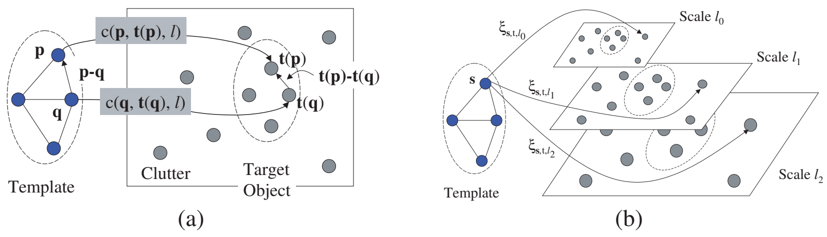


Fig. 3. (a) Matching a template shape flow to a target video frame. See Eq. 1. (b) Linearizing the problem with respect to scale. By quantizing the range of scales, the nonlinear scale factor is replaced by scale-specific indicator variables ξ. This quantization of scale is not a problem because the matching cost can tolerate some scale difference between the template and target.

triangulation of the flow line start points. Fig. 2(c) shows the top-down view of the flow lines in the target video volume, which are sampled from edge pixels in the first frame. Note that the target is a different person at a different scale, and that, individually, the target flow lines differ significantly from those in the template.

The problem is to find an assignment between flow lines in the template and flow lines in the target so that (1) the paired flow lines are similar, and (2) the spatial relationships between flow lines as given by the neighbor graph are respected. Fig. 2(d) shows the matching result using the method proposed by this paper. Not only is the target detected and well localized, but the assignment of flow lines between the template and target is sensible.

Our proposed method's success relies on the combination of matching the loopy relation graph and doing a robust global search. We can see the result of removing one or the other of these elements. Fig. 2(e) shows the result of our method if the loopy graph is replaced by a chain. In this case, without cycles, the matching may be done efficiently using dynamic programming (DP). Fig. 2(f) shows the result of using the fully loopy graph, but using ICM instead of our proposed linear relaxation. The DP and ICM results are poor, despite the fact that both methods were given the advantage of having the template pre-scaled to the target scale.

3.2 The Matching Problem

The template shape flow consists of a set of flow lines. These flow lines originate from randomly selected edge points on the object. Pairwise neighbor relationships between flow lines are given by the Delaunay triangulation of the flow line start points. Thus, the neighbor graph has cycles. All of the flow lines in the template have the same temporal extent, but vary in length due to motion.

The target search domain is a space-time video volume having the same temporal extent as the template action. The search is performed over a randomly selected subset of flow lines anchored by edge points in the first frame of the volume. The goal is to find a consistent assignment of flow lines in the template to flow lines in the target. Matched flow lines should be similar, and the spatial arrangement in the target should match that of the template. We assume that there is little change in rotation between the template and target, but that there may be a large change in scale.

Let $\mathcal{S}$ denote the template point set and $\mathcal{N}$ the set of neighbor point pairs in the template. We formulate shape flow matching as the following optimization problem:

$$\min_{\mathbf{t},l} \left\{ \sum_{\mathbf{s}\in\mathcal{S}} c(\mathbf{s},\mathbf{t}(\mathbf{s}),l) \;+\; \lambda \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} ||(\mathbf{p}-\mathbf{q}) - (l\cdot\mathbf{t}(\mathbf{p}) - l\cdot\mathbf{t}(\mathbf{q}))|| \right\} \tag{1}$$

where $\mathbf{t}(\mathbf{s})$ maps template point $\mathbf{s}$ to the target point, and $c(\mathbf{s},\mathbf{t},l)$ is the cost of matching $\mathbf{s}$ and $\mathbf{t}$ with scale l. The first term in the energy function is the cost of matching individual flow lines. The second term penalizes assignments that violate the relative positions of flow lines given by the neighbor graph $\mathcal{N}$. λ weighs the relative importance of the two terms. We wish to find an assignment $\mathbf{t}(\cdot)$ and scale l so that the template shape flow matches a target shape flow with small cost in a spatially consistent manner.

The matching cost c between a pair of flow lines is computed as follows. Flow lines are normalized to a common spatial diameter (but regularized to account for stationary flow lines, which have zero diameter). Each flow line has the same number of sample points, given by the number of frames in the template. The matching cost is given simply by the Euclidean distance between the two vectors formed by the normalized flow lines' 2D spatial coordinates. Because of the spatial normalization, the flow line matching cost is scale invariant. One could use scale-specific matching costs, but we found this simpler method to be sufficient.

As stated, this optimization problem is difficult to solve. It is discrete, nonlinear, and highly non-convex. In addition, the loopy relation graph precludes efficient search. We are, however, able to solve this optimization problem effectively. In the following subsections, we show how to remove the nonlinearities, how to deal with the non-convexity, and how to dramatically reduce size of the search space, yielding a fast and small linear program solution.

3.3 Linearization

As stated, the optimization problem is a nonlinear integer program. The first step in transforming the problem into a linear program is to remove the nonlinearities. As shown in Fig. 3, we may remove the nonlinear scale factor from the energy function by searching over a range of m discrete scales $L = \{l_0, l_1, ..., l_{m-1}\}$. We use $m = 7$ scales, equally spaced between half and double the template's natural scale. Let $\xi_{\mathbf{s},\mathbf{t},l}$ be binary indicator variables that take the value 1 when template point $\mathbf{s}$ matches target point $\mathbf{t}$ at scale l. The nonlinear scale factor is replaced by the scale-specific indicator variables ξ. Quantizing scale in this manner works because the matching tolerates small differences in scale between the template and target shape flows. The first term of Eq. 1 may therefore be linearized as follows:

$$\sum_{\mathbf{s}\in\mathcal{S},\ \mathbf{t}\in\mathcal{T}(\mathbf{s}),\ l\in L} c(\mathbf{s},\mathbf{t},l)\cdot\xi_{\mathbf{s},\mathbf{t},l} \tag{2}$$

where $\mathcal{T}(\mathbf{s})$ is the target candidate point set for template point $\mathbf{s}$. In order to linearize the second term in the energy function, we use the L_1 vector norm and a standard linear program trick to linearize the absolute value function. By introducing some pairs of auxiliary variables, the second term may be linearized as follows:

$$\lambda \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} (u^+_{\mathbf{pq}} + u^-_{\mathbf{pq}} + v^+_{\mathbf{pq}} + v^-_{\mathbf{pq}}) \tag{3}$$

where $u^+_{\mathbf{pq}}$, $u^-_{\mathbf{pq}}$, $v^+_{\mathbf{pq}}$ and $v^-_{\mathbf{pq}}$ are non-negative auxiliary variables. Let $x(.)$ and $y(.)$ be functions that extract the x and y coordinates of a point. The auxiliary variables follow the constraints $\forall\{\mathbf{p},\mathbf{q}\} \in\mathcal{N}$:

$$u^+_{\mathbf{pq}} - u^-_{\mathbf{pq}} = u_{\mathbf{p}} - u_{\mathbf{q}} - x(\mathbf{p}) + x(\mathbf{q}) \qquad v^+_{\mathbf{pq}} - v^-_{\mathbf{pq}} = v_{\mathbf{p}} - v_{\mathbf{q}} - y(\mathbf{p}) + y(\mathbf{q}) \tag{4}$$

and where $u_{\mathbf{p}}$ and $v_{\mathbf{p}}$ are x and y coordinates of the scaled target point $l \cdot \mathbf{t}(\mathbf{p})$:

$$u_{\mathbf{s}} = \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{s}),\, l\in L} l \cdot x(\mathbf{t}) \cdot \xi_{\mathbf{s},\mathbf{t},l} \qquad v_{\mathbf{s}} = \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{s}),\, l\in L} l \cdot y(\mathbf{t}) \cdot \xi_{\mathbf{s},\mathbf{t},l} \tag{5}$$

If the objective function is optimized given these constraints, at least one of $u^+_{\mathbf{pq}}$ or $u^-_{\mathbf{pq}}$ and at least one of $v^+_{\mathbf{pq}}$ or $v^-_{\mathbf{pq}}$ vanishes. The proof of this fact is by contradiction: If it were not the case, then we could subtract the smaller value in each pair from u^+ and v^-, producing a feasible solution at lower cost. Based on this observation, $u^+_{\mathbf{pq}} + u^-_{\mathbf{pq}} = |u_{\mathbf{p}} - u_{\mathbf{q}} - x(\mathbf{p}) + x(\mathbf{q})|$ and $v^+_{\mathbf{pq}} + v^-_{\mathbf{pq}} = |v_{\mathbf{p}} - v_{\mathbf{q}} - y(\mathbf{p}) + y(\mathbf{q})|$, and therefore $u^+_{\mathbf{pq}} + u^-_{\mathbf{pq}} + v^+_{\mathbf{pq}} + v^-_{\mathbf{pq}} = ||(\mathbf{p}-\mathbf{q}) - (l\cdot\mathbf{t}(\mathbf{p}) - l\cdot\mathbf{t}(\mathbf{q}))||$. The original energy minimization problem of Eq. 1 is now linear.

Additional constraints enforce that the matching is an assignment, where each template point matches a single target point:

$$\sum_{\mathbf{t}\in\mathcal{T}(\mathbf{s}),\, l\in L} \xi_{\mathbf{s},\mathbf{t},l} = 1,\ \xi_{\mathbf{s},\mathbf{t},l} = 0 \text{ or } 1,\ \forall \mathbf{s}\in\mathcal{S} \tag{6}$$

And we constrain each pair in the assignment to select the same scale w, so that the template scales uniformly:

$$\sum_{\mathbf{t}\in\mathcal{T}(\mathbf{s}),\, l\in L} l\cdot\xi_{\mathbf{s},\mathbf{t},l} = w,\ \forall \mathbf{s}\in\mathcal{S} \tag{7}$$

The matched target points are given by:

$$\hat{\mathbf{t}}_x = \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{s}),\, l\in L} x(\mathbf{t}) \cdot \xi_{\mathbf{s},\mathbf{t},l} \qquad \hat{\mathbf{t}}_y = \sum_{\mathbf{t}\in\mathcal{T}(\mathbf{s}),\, l\in L} y(\mathbf{t}) \cdot \xi_{\mathbf{s},\mathbf{t},l} \tag{8}$$

The original nonlinear integer problem has been transformed into a linear integer program without any approximations: The linearized version solves the original problem exactly, apart from the effects of quantizing scale. The problem is still integer, however, and still non-convex. The following subsection addresses both of these issues.

3.4 Relaxation and Convexification: The Lower Convex Hull Property

Due to the computational expense of integer programming, we relax the indicator variables $\xi_{\mathbf{s},\mathbf{t},l}$ so that instead of taking discrete binary values they take continuous values

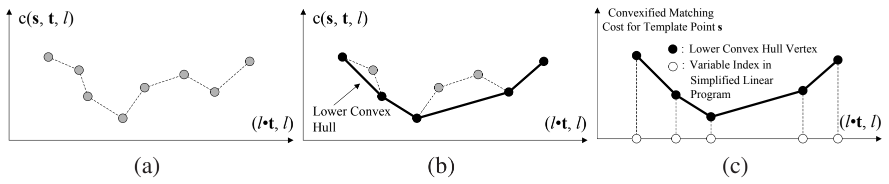

c(s, t, l)
(l•t, l)
(a)
c(s, t, l)
Lower Convex Hull
(l•t, l)
(b)
Convexified Matching Cost for Template Point s
● : Lower Convex Hull Vertex
○ : Variable Index in Simplified Linear Program
(l•t, l)
(c)

Fig. 4. Convexification using the 4D lower convex hull. The actual cost surface of the integer program is non-convex (a). Relaxing the ξ yields a linear program, the solution for which lies on the lower convex hull (b) of the original cost surface. Any variables corresponding to points above this surface are redundant in the linear program, so they may be pruned (c). In our formulation, the cost surface is 3D, and so the lower convex hull is 4D. As the dimensionality increases, pruning becomes proportionally more effective.

in the unit interval. In this now linear program, the ξ correspond to a kind of likelihood that a template flow line matches a target flow line at a particular scale. The program is very large, however: The number of variables $\xi_{\mathbf{s},\mathbf{t},l}$ is proportional to the number of template points, the number of candidate points, and the number of discrete scales. We can greatly reduce the number of ξ variables without changing the solution of the linear program by taking advantage of the following property.

Property: We need keep only those $\xi_{\mathbf{s},\mathbf{t},l}$ that correspond to the vertices of the 4D lower convex hull of the point clouds $(l \cdot \mathbf{t}, l, c(\mathbf{s}, \mathbf{t}, l))$ with respect to the last coordinate for each $\mathbf{s}$.

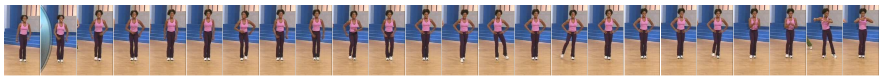
As depicted in Fig. 4, the solution to a linear program may be found by exploring the lower convex hull of the cost surface. Variables corresponding to interior points, depicted as gray dots in Fig. 4(b), are algebraically redundant because they may be represented as linear combinations of points on the hull. These "interior variables" may be safely removed from the linear program without affecting its solution.

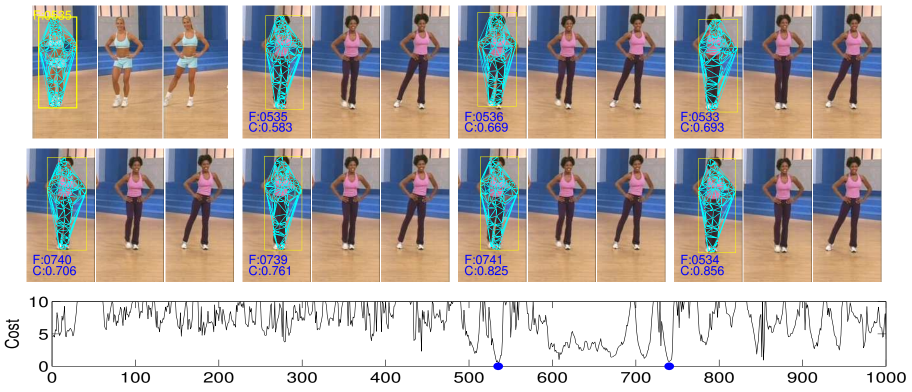
This convexification procedure can reduce the size of the linear program and therefore greatly speed up its solution—by a factor of 100 or more in our experiments in this problem domain. Note that if $c(\mathbf{s}, \mathbf{t}, l)$ is convex over $(l \cdot \mathbf{t}, l)$ for each $\mathbf{s}$ in the original integer program, the relaxed linear program will find the global optimum.

3.5 Iterative Refinement

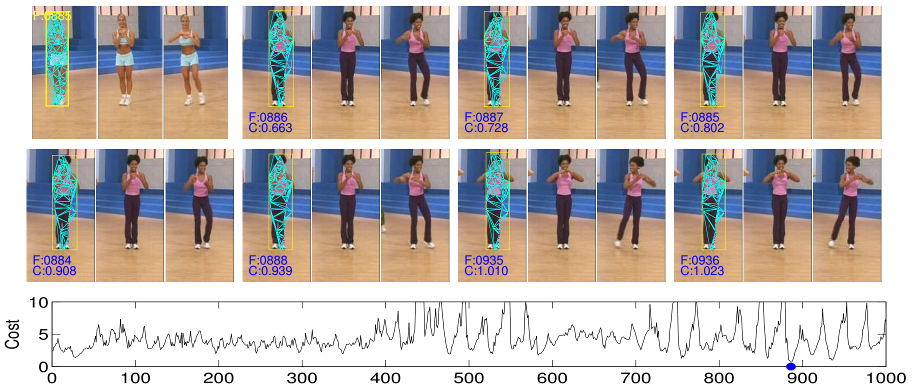
In general, the cost surface $c(\mathbf{s}, \mathbf{t}, l)$ is non-convex over the domain $(l \cdot \mathbf{t}, l)$. Because of the convexification, and because of the interactions between variables from $\mathcal{N}$, the linear relaxation gives an approximate solution. The magnitude of the approximation error depends on the size of the domain, however, because the convex approximation will be more accurate when the domain is smaller. This suggests an iterative approach known as *successive convexification* [6].

Initially, the search window, or *trust region*, covers the entire target domain: Every point in the target frame is a possible matching candidate for each template point at every scale. In this case, the matching will still likely find the correct target, but the convexification introduces localization errors. Given a solution, we may iteratively shrink the trust region over the domain $(\mathbf{t}, l)$, each time repeating the convexification and linear relaxation procedure. As the trust region shrinks, the lower convex hull fits the original cost surface more accurately, and the template is localized in the target more accurately.


(a) Sampled frames from the 1000-frame *fitness* video.



(b) Template #1, top 7 matches, and per-frame match cost.



(c) Template #2, top 7 matches, and per-frame match cost.

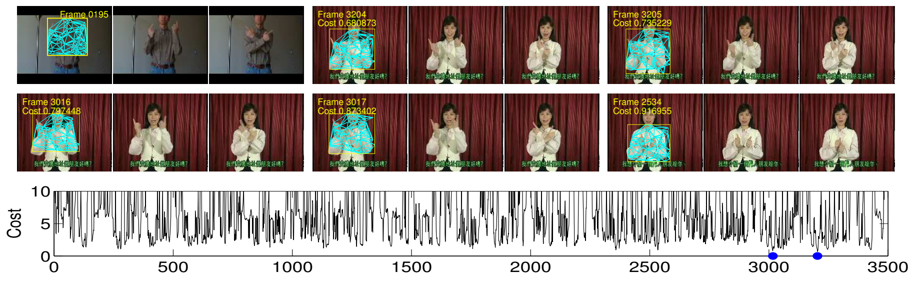
Fig. 5. Detection of two 15-frame actions in the *fitness* video. The graphs show the per-frame match cost for the template; blue dots mark the locations of true positives. For each template and match, there are three images: (1) the first frame of the action with the template's neighbor graph, frame number F, and match cost C; (2) the second frame of the action; and (3) the last frame of the action. In (b), the top 6 matches correspond to the 2 true positives for a *right leg out* action. In (c), the top 5 matches correspond to the 1 true positive for a *right arm out, left knee out* action. In these examples, which include significant camera motion, there are no errors.

At most 5 iterations are required, and the program shrinks geometrically in size at each iteration.

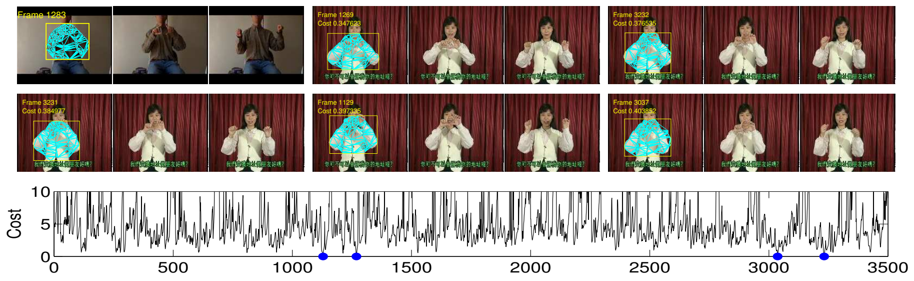
4 Empirical Evaluation

We show experimental results for the proposed action detector on a wide range of videos and on the action dataset of Blank et al. [2]. The output of the optimization described above is the best match for a template in each frame of the target video, whether that

(a) Sampled frames from the 3501-frame *sign* video.



(b) Template #1, top 5 matches, and per-frame match cost.



(c) Template #2, top 5 matches, and per-frame match cost.

Fig. 6. Compare with Fig. 5. In (b), the top 4 matches correspond to the 2 true positives for a 7-frame *exchange* action. In (c), the top 5 matches correspond to the 4 true positives for a 15-frame *message* action; the *message* template is partitioned into left/right halves. In these examples, which involve self-occlusion, there are no errors.

match is good or bad. We require a method of ranking the matches by quality, across frames, to generate a top-list for action detection.

4.1 Scoring Action Matches and Ranking Results

The energy function that we minimize to find an action in a video frame is effective at locating the best match within a frame. However, the energy values cannot be meaningfully compared across frames. There are two reasons for this. First, the flow line match cost in the linear program is totally scale invariant, which is too lenient for a cross-frame match score. And second, the energy will be artificially low when the template is matched against a partially similar action; for example, a template of a person waving one arm will match well to a target waving two arms, but should not be scored high.

To address these issues, we formulate a more robust similarity measure between the template shape flow and the matched target shape flow. First, we normalize the flow lines within each shape flow by the mean flow line length, and translate each flow line start point to the origin. The distance between these two bundles of normalized and shifted flow lines is defined as the average minimum distance between individual flow lines across bundles, which is a symmetric measure. The distance between individual

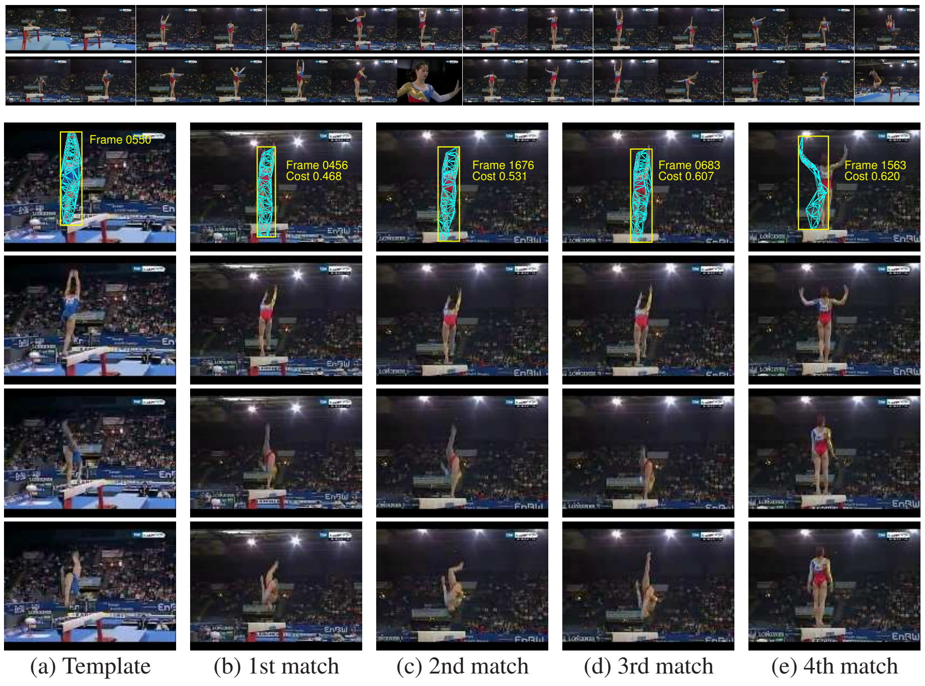


Fig. 7. Detection of a complex 15-frame action in the 2091-frame *gymnastics* video. The action is the first half or a forward flip on the balance beam. The top row of images shows frames sampled uniformly in time from the video. Column (a) shows the action template along with three frames (beginning, middle, end) from the action. Columns (b-e) show the top 4 matches. The sequence contains 3 true positives, which are the top 3 matches. In this example, which involves rapid and complex object motion as well as camera motion and background clutter, there are no errors.

flow lines is again defined as the Euclidean distance between the flow lines' spatial coordinate vectors.

As stated, this shape flow distance measure discards all information regarding the relative position of flow lines. For most actions, for ranking results, this is not a problem. However, consider the highly symmetric actions shown in Fig. 6(c). Because the flow lines on the left arm of one action will have low distance to the flow lines on the right arm of the other action, the total distance will be very low. Consequently, for such actions, we break the symmetry by partitioning the template into halves. Normalization is still performed on the entire template, but the distance is computed for each part separately. The total distance for a partitioned template is simply the average distance of the parts, weighted by the number of flow lines in each part. All of the results in this paper use a whole single-part template; only the action in Fig. 6(c) required a (left-right) template partition.

4.2 Results

We first present action detection results for single actions in four extended videos. Fig. 5 shows results for the *fitness* video; Fig. 6 for the *sign* video; Fig. 7 for the *gymnastics* video; and Fig. 8 for the *golf* video. In all cases, the top matches are determined using

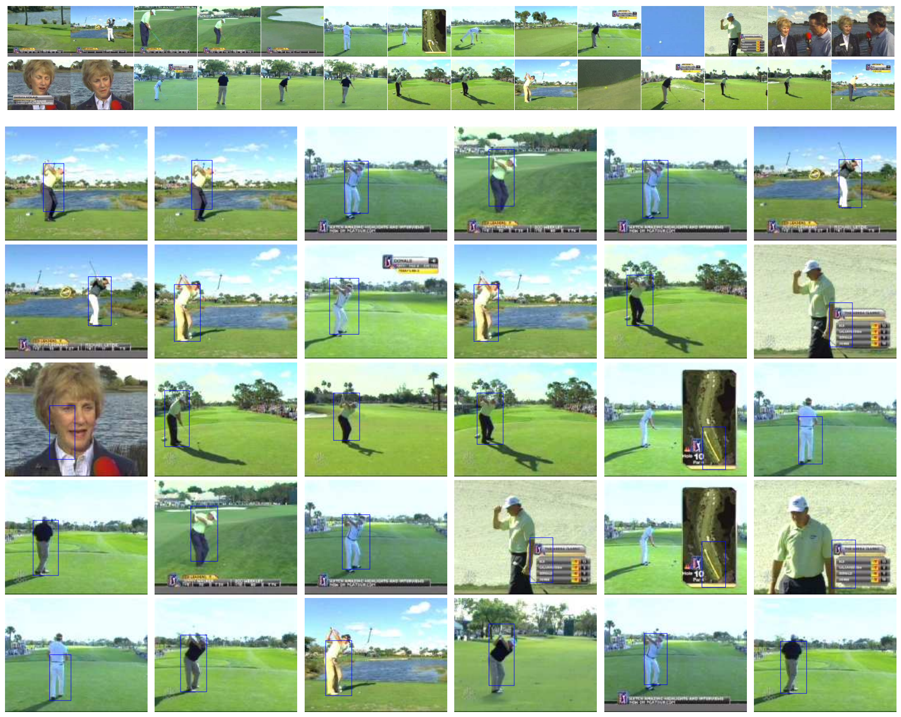


Fig. 8. The top 30 detections for a 8-frame *swing* template action in an 8000-frame video *golf*. There are 10 hits (with duplicates) of 14 true positives and 11 false positives, yielding 71% recall and 63% precision. This is a difficult video with few true positives compared to negatives, and much camera motion.

the shape flow distance measure described in the previous subsection. These videos involve fast object motion, camera motion, and background clutter. In addition, the template and target are always of different people, which introduces scale variation, pose variation, and intra-class variation.

Fig. 5 shows match results for two 15-frame actions in the 1000-frame *fitness* video. All true positives from the video appear at the front of the list of top matches. Fig. 6 shows the same perfect result for the detection of two 15-frame actions in the 3501-frame *sign* video. In both videos, there are multiple positives per true positive because the top list ranks matches from all frames in the video. The *fitness* video involves a moving camera and background clutter; the *sign* video involves background clutter (there are many stationary flow lines in the background) as well as complex self occlusion. Despite these challenges, the actions are detected successfully.

Fig. 7 shows match results for a complex 15-frame action in the 2091-frame *gymnastics* video. After applying non-minima suppression in the time dimension to the per-frame match scores, the three true positives appear as the top three matches. Non-minima suppression is not necessary, but prunes duplicate matches from the top list. Despite background clutter, constant camera motion, and significant intra-class variation, the proposed method is again successful.

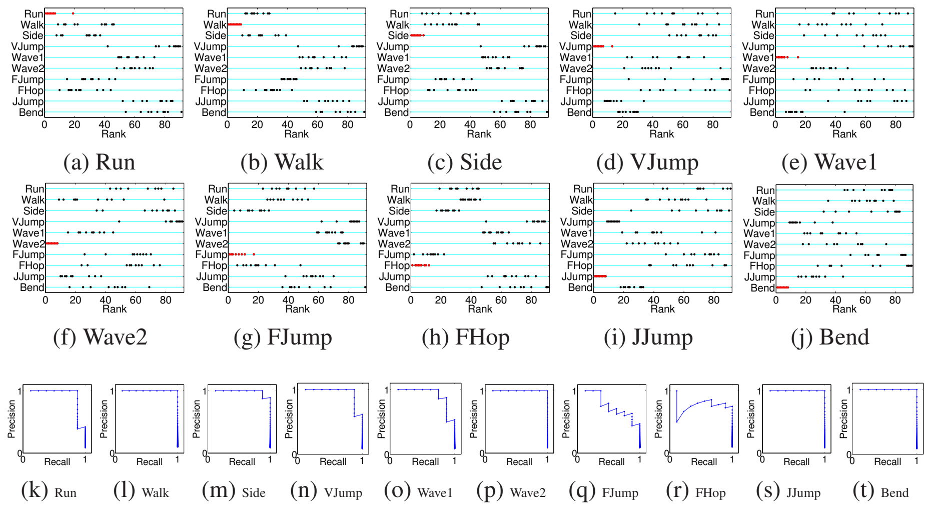


Fig. 9. Detecting actions in the video dataset of Blank et al. [2]. The dataset consists of 93 single action video clips for 10 actions. The 10 actions are: running (*Run*), walking (*Walk*), side stepping (*Side*), jumping in place (*VJump*), waving one arm (*Wave1*), waving two arms (*Wave2*), forward jumping (*FJump*), forward hopping (*FHop*), jumping jack (*JJump*), and bending (*Bend*). We use a single action template for each action class. Graphs (a-j) show, for each action, all 92 clips (template clip excluded) sorted by match score when matched against that action template; the y-axis places each clip into its ground truth category. Most of the same category target clips (red dots) are ranked first, which is the goal. Graphs (k-t) show the corresponding precision recall (PR) curves for the 10 actions. We achieve high precision and recall for 8 of 10 actions; the *FJump* and *FHop* actions are extremely similar visually, and panels (g,h) show that they get confused with each other.

Fig. 8 shows results for a 8-frame *swing* action in the 8000-frame *golf* video. This is the most challenging of the four videos, and demonstrates some match failures. The top 30 matches are shown over the entire sequence. There are 10 hits of 14 true positives with 11 false positives, yielding 71% recall and 63% precision (accounting for duplicate hits). This sequence involves much camera motion, a variety of individuals as targets, and highly variable background clutter. In addition, there are many distractor frames in which there is no relevant object present.

We also report results for the action recognition dataset of Blank et al. [2] in Fig. 9. The dataset consists of 93 single action video clips for 10 actions performed by various subjects. We extract a single 15-frame shape flow template for each action by randomly choosing a 15-frame sequence from a randomly chosen clip. Each template is then matched against each frame in the set of test clips (excluding the template clip). The match cost for a clip is taken as the minimum match cost across frames in the clip. Fig. 9 shows that the top matching clips have the correct class, yielding high performance precision recall curves for 8 of 10 classes; the *FJump* and *FHop* classes are not distinguished well by our detector, but that is because they are visually extremely similar. The equal precision-recall point across the dataset is 90%. Excluding the *FJump* and *FHop* categories, the equal PR point is at 95%.

5 Conclusion

We have proposed a novel representation for actions called the *shape flow*, which incorporates both the shape *and* motion of an object in an assembly of easily computed flow lines. We have formulated action detection and localization as an energy minimization problem, yielding a difficult nonlinear, non-convex integer program. We show how to linearize this energy function, and how to use successive convexification of the energy cost surface to relax the program and vastly reduce the size of the linear program without approximation. Experimental results on difficult videos and a standard action detection dataset confirm that the proposed method can accommodate complex object motion and self-occlusion, camera motion, background clutter, scale changes, pose variation, and intra-class variation. The method is non-parametric, and requires neither background subtraction nor accurate trajectories. We believe that this method is a useful tool for automatic action detection.

References

1. Besag, J.: On the Statistical Analysis of Dirty Pictures. Journal of the Royal Statistical Society, Series B 48(3):259–302, 1986.
2. Blank M., Gorelick L., Shechtman E., Irani M., Basri R.: Actions as Space-Time Shapes. ICCV, 2005.
3. Bobick, A., Davis, J.: The Recognition of Human Movement Using Temporal Templates. IEEE Trans. PAMI 23(3):257-267, 2001.
4. Efros, A., Berg, A., Mori, G., Malik, J.: Recognizing Action at a Distance. ICCV, 2003.
5. Interrante, V.,Grosch, C.: Visualizing 3D Flow. IEEE Computer Graphics and Applications 18(4):49–53, 1998.
6. Jiang, H., Drew, M.S., Li, Z.N.: Matching by Linear Programming and Successive Convexification, IEEE Trans. on PAMI 29(6):959–975, 2007.
7. Ke, Y., Sukthankar R., Hebert, M.: Event Detection in Crowded Videos. ICCV, 2007.
8. Kenwright, D., Mallinson, G.: A 3-D Streamline Tracking Algorithm Using Dual Stream Functions. Visualization 62–68, 1992.
9. Laptev, I., Lindeberg, T.: Space-Time Interest Points. ICCV, 2003.
10. Laptev, I., Prez, P.: Retrieving Actions in Movies. ICCV, 2007.
11. Mori, G., Ren, X.F., Efros, A., and Malik, J.: Recovering Human Body Configurations: Combining Segmentation and Recognition, CVPR 326–333, 2004.
12. Parameswaran, V., Chellappa, R.: Human Action-Recognition Using Mutual Invariants. CVIU 98(2):295–325, 2005.
13. Ramanan, D., Forsyth, D.A., Zisserman, A.: Tracking People by Learning Their Appearance, IEEE Trans. on PAMI 29(1):65–81, 2007.
14. Schuldt, C., Laptev, I., Caputo, B.: Recognizing Human Actions: A Local SVM Approach. ICPR, 2004.
15. Scovanner, P., Ali, S., Shah, M.: A 3-Dimensional SIFT Descriptor and its Application to Action Recognition. ACM Multimedia, Sept. 2007.
16. Shechtman, E., Irani, M.: Space-Time Behavior Based Correlation. CVPR, 2005.
17. Sheikh, Y., Shah, M.: Exploring the Space of an Action for Human Action Recognition. ICCV, 2005.
18. Weinland, D., Ronfard, R., Boyer, E.: Free Viewpoint Action Recognition using Motion History Volumes. CVIU, Nov./Dec. 2006.
19. Yilmaz, A., Shah, M.: Actions as Objects: A Novel Action Representation. CVPR, 2005.
20. Yilmaz, A., Shah, M.: Recognizing Human Actions in Videos Acquired by Uncalibrated Moving Cameras. ICCV, 2005.