Optimal spatio-temporal path discovery for video event detection

Conference Paper *in* Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition · July 2011

DOI: 10.1109/CVPR.2011.5995416 · Source: IEEE Xplore

2 authors:

Du Tran
Nanyang Technological University
6 PUBLICATIONS **696** CITATIONS
SEE PROFILE

Junsong Yuan
Nanyang Technological University
285 PUBLICATIONS **9,955** CITATIONS
SEE PROFILE

Optimal Spatio-Temporal Path Discovery for Video Event Detection

Du Tran, Junsong Yuan
School of EEE, Nanyang Technological University, Singapore
{dutran,jsyuan}@ntu.edu.sg

Abstract

We propose a novel algorithm for video event detection and localization as the optimal path discovery problem in spatio-temporal video space. By finding the optimal spatio-temporal path, our method not only detects the starting and ending points of the event, but also accurately locates it in each video frame. Moreover, our method is robust to the scale and intra-class variations of the event, as well as false and missed local detections, therefore improves the overall detection and localization accuracy. The proposed search algorithm obtains the global optimal solution with proven lowest computational complexity. Experiments on realistic video datasets demonstrate that our proposed method can be applied to different types of event detection tasks, such as abnormal event detection and walking pedestrian detection.

1. Introduction

Sliding window-based approaches have been quite successful in searching objects in images, such as face and pedestrian detections [12, 19]. However, its extension to searching for spatio-temporal sliding windows in videos remains a challenging problem. Although several methods have been proposed recently [11, 22] to search spatio-temporal video patterns with applications like video event and human action detection, they are confronted with two unsolved problems.

First, most of the current spatio-temporal sliding window search methods only support sliding windows of constrained structure, i.e., the 3-dimensional (3D) bounding box. Unfortunately, unlike object detection where a bounding box works reasonably well in many applications, the 3D bounding box is quite limiting for video pattern detection. To illustrate this, Figure 1a shows a cycling event. The cyclist starts at the left side of the screen and rides to the right side of the screen. To detect this event, because of the bounding box constraint, one can only locate the whole event using a large video subvolume, which covers not only the cycling event, but also a significantly large portion of the backgrounds (Figure 1a). In such a case, the detection score of the video event is negatively affected by the cluttered and dynamic backgrounds. Instead of providing a global bounding box that covers the whole event, more often than not, it is preferable to provide an accurate spatial location of the video event and track it in each frame. As a result, a more accurate spatio-temporal localization is desirable to detect the video event, as shown in Figure 1b.

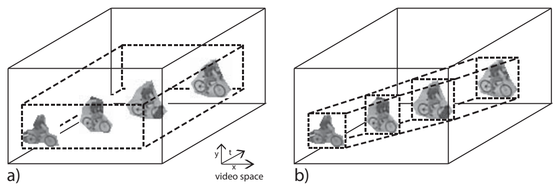

Figure 1. **Detection of the cycling event** a) Event localization by 3-dimensional bounding box. b) More accurate spatio-temporal localization of the event.

Moreover, as the video space is much larger than the image space, it becomes very time consuming to search 3D sliding windows. For example, given a video sequence of size $w \times h \times n$, where $w \times h$ is the spatial size and n is its length, the total number of 3D bounding boxes is of $O(w^2h^2n^2)$, which is much larger compared with the image space of only $O(w^2h^2)$ 2D boxes. Although some recent methods have been proposed to handle the large video space [22], the worst case complexity is still of $O(w^2h^2n)$. In general, it is challenging to search videos of high spatial resolutions. Even worse, if we relax the bounding box constraint of the sliding windows, the number of candidates will further increase. Thus a more efficient search method is required.

To address the above problems, we propose a novel spatio-temporal localization method which relaxes the 3D bounding box constraint and formulates the video event detection as a spatio-temporal path discovery problem. Suppose a discriminative classifier can assign a local detection score to every 2D sliding window in each frame. To fuse these local evidences and connect them to establish a spatio-temporal path, we build a spatio-temporal trellis which presents all of smooth spatio-temporal paths, where a target event will correspond to one of them. By finding the optimal path in the trellis with the highest detection score, our formulation is a generalization of the 3D bounding box search in [22]: we do not reinforce the fixed spatial location of the video event, but track the event as it moves across multiple frames. Because the discovered path precisely contains the video event, it minimizes the affection of the cluttered and dynamic backgrounds.

Although the search space of our new formulation is much larger than searching 3D bounding boxes, we propose an efficient search method that can obtain the global optimal solution with proven lowest search complexity, which is only linear to the video volume size, i.e. $O(whn)$. Experiments on abnormal video event detection and walking pedestrian detection validate the following advantages of our new formulation of video event detection:

1. By discovering the optimal spatio-temporal path, our method determines the start and the end of the video event automatically, and can precisely localize the event in each video frame. It is robust to the false and missed local detections, thus can effectively handle heavy occlusions;
2. As both positive and negative training examples are utilized for a discriminative training, our method is robust to intra-class variations of the video events and the cluttered and dynamic backgrounds;
3. Our proposed method can be easily extended to handle spatial scale variations of the event, and can detect multiple events simultaneously.

1.1. Previous Work

Video event detection is an important topic in computer vision, with extensive applications in video surveillance, content-based video search, multimedia retrieval, etc. The later two have seen increasing demands due to the exploding number of internet videos (*e.g.* YouTube). At the same time, the problem becomes more challenging when dealing with realistic videos because of intra-class variations, complex background motions, scale changes, and occlusions, not to mention the high dimensional search space inherent to videos.

One traditional approach for event detection is to track the actors, stabilize these figures, and then recognize them [7]. Such methods highly rely on the quality of the tracking results, hence suffer from unreliable trackers. This limitation motivates methods that handle detection and recognition simultaneously, normally accomplished by spatio-temporal video volume matching, including action-MACH [16], volumetric features [11], segment-based features [10], spacetime orientation [5], etc. To localize events, these methods have to apply the sliding subvolume scheme which is ineffective and time-consuming. Rather than sliding subvolume, Boiman and Irani [1] proposed ensembles of patches to detect irregularities in images and videos. Hu *et al* used multiple-instance learning to localize the best video subvolume [8]. Recently, with the success of branch-and-bound subwindow search [12], Yuan *et al* extended this method to subvolume search [22] with some speed improvements. However, these approaches are still constrained by the 3D subvolume. Finally, Zhang *et al* [24] relaxed the subwindow rectangle constraint to free-shape subwindow search based on contour refinement. This approach works well for object localization but is still difficult to extend to video search due to its complexity.

Our approach also shares some properties with tracking-by-detection methods [2, 14, 18]. These methods had shown their effectiveness compared to traditional tracking methods thanks to the success of object detectors. Similarly, our approach considers joining detection outputs to maximize the discriminative scores while keeping the smoothness of the movement trajectories. In contrary, our proposed method is not limited to object detectors, but can be also applied to more general discriminative confidence maps of event, action, motion, or keypoint detectors. This flexibility makes it more general, and thus applicable to a broader range of video event detection problems. To our best knowledge, our *Maximum Path* algorithm is novel to video event detection and proven to be globally optimal with the lowest computational complexity. Interestingly, this problem has not been discussed in discrete algorithm literature although the *Maximum Subarray* problem had been raised and solved long time ago [9].

2. Problem Formulation

We denote a video sequence as $\mathcal{S} = \{I_1, I_2, \ldots, I_n\}$, where I_k is a $w \times h$ image frame. Treating the video as a spatio-temporal data volume, for each spatio-temporal location $v = (x, y, t)$, we denote by $W(v)$ the local window or subvolume centered at v. Without loss of generality, we suppose all of the windows are of a fixed scale, we further denote by $M(W(v))$, or $M(v)$ for short, the discriminative score of the local window centered at v. A high positive score of $M(v)$ implies a high likelihood that the event occurs at the local position v, while a negative score indicates a low likelihood of the occurrence. There are many ways to obtain the score $M(v)$ using different types of features. For example, one can use the 2D window [3, 20] to slide over the video sequence and get the local scores of each window. Alternatively, individual spatio-temporal interest points [6, 13] can vote for the video event [22], then the score of a local window is the summation of the interest point scores.

By treating each window $W(v)$ as a node, we obtain a 3-dimensional trellis to represent all $W(v)$ in the video. Given a 3D trellis $\mathcal{G}_M$ with a size of $w \times h \times n$, $p = \{v_i\}_{i=i_1}^{i_k}$ is a *path* in $\mathcal{G}_M$ if it satisfies (1) the *path connectivity constraints*: $x_i - 1 \le x_{i+1} \le x_i + 1$, $y_i - 1 \le y_{i+1} \le y_i + 1$,

$t_{i+1} = t_i+1$ and (2) the *boundary constraints*: $1 \leq x_i \leq w$, $1 \leq y_i \leq h$, and $1 \leq t_{i_1} \leq t_{i_k} \leq n$. The first constraint set shows that each node $v = (x, y, t)$ has 9 incoming and 9 outgoing neighbors as showed in Figure 2a. The second constraint set indicates that the path can start and end at any position in the 3D array as long as the ending point occurs later than the starting point. Let $p = \{v_i\}_{i=i_1}^{i_k}$ be a path in $\mathcal{G}_M$, to evaluate its likelihood, we compute the accumulated score of the path p in Eq. 1.

$$M(p) = \sum_{i=i_1}^{i_k} M(v_i) \tag{1}$$

As each video event is characterized by a smooth spatio-temporal path in the 3D trellis, to detect the video event, the problem becomes to find the optimal path p^* with the highest accumulated score:

$$p^* = \underset{p \in path(\mathcal{G})}{\text{argmax}}\ M(p) \tag{2}$$

Solving the *Maximum Path* problem is difficult (Figure 2b) because of the large search space: we do not know the start location (x_s, y_s, t_s) or the end location (x_e, y_e, t_e) of the event, as well as all of the intermediate states. The search space of all possible paths is exponential: $O(whnk^n)$, where whn is the size of the video volume, k is the number of incoming edges per node. Thus exhaustive search is infeasible. Although the maximum path problem can be addressed by the traditional shortest path search algorithm, *e.g.*, the Floyd-Warshall algorithm to find the shortest paths between all pairs of vertices, the search complexity is still quite high. The complexity of the Floyd-Warshall algorithm is to the cube of the number of vertices $O(|V|^3)$. Thus, it becomes $O(w^3h^3n^3)$ to solve Eq. 2, which is very time consuming for a large video volume. Other related work includes the *Maximum Subarray* problem which was posed by Ulf Grenander in 1977 and the 1D case was solved by Jay Kadane in 1984 [9]. Although it works perfect for the 1D trellis [23], the problem is more complicated with higher dimension, *e.g.*, for our 3D trellis. Although the branch-and-bound search has proven to be efficient in searching 2D and 3D bounding boxes [12, 22], they cannot be applied to more flexible structures. To propose an efficient search that can find the global solution in Eq. 2, we firstly present an approach based on dynamic programming, followed by our proposed search method with proven lowest complexity.

3. Optimal Path Discovery

3.1. Efficient Max-Path Search via Dynamic Programming

Before addressing the Max-Path discovery problem, we first study a simplified version of the problem. We assume

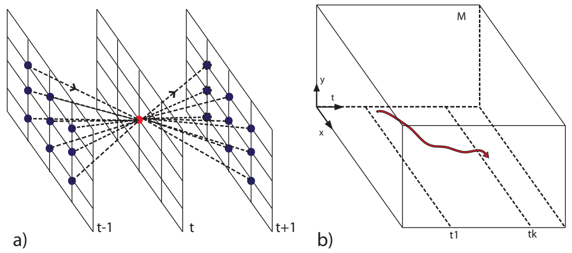

Figure 2. **Maximum Path problem** a) 9 incoming and 9 outgoing neighbors for a node in $\mathcal{G}_M$. b) The visualization of one path. Searching for the maximum path in spatio-temporal space is difficult due to an exponential number of possible paths with arbitrary lengths.

that the best path starts somewhere in the first frame and ends at the last frame. The following dynamic programming algorithm will find the best path.

Let $S_{i,j,t}$ be the maximum accumulated score of the best path starting somewhere from the first frame and leading to (i, j, t). For short, we denote $u = (i, j)$ and $v = (x, y)$ are 2D indices (*e.g.* $S_{i,j,t} = S_{u,t}$). We note that these notions are slightly different from the previous section where v is a 3D index. And $N(u)$ is the set of neighbors of u in the previous frame. Eq. 3 gives a solution for the *Max-Path* search problem.

$$S_{u,t} = \begin{cases} M_{u,t}, & t = 1 \\ \max_{v \in N(u)} \{S_{v,t-1} + M_{u,t}\}, & t > 1. \end{cases} \tag{3}$$

This dynamic programming can be completed in $O(whn)$ to compute S, another $O(n)$ to trace backward to identify the best path, and uses $O(whn)$ memory space.

However, to automatically determine the starting and ending locations of the paths, we need to try different combinations and perform the dynamic programming many times. To improve this, let $S_{u,t,s}$ be the accumulated scores of the best path starting from the s-th frame to the end location (u, t). S can be computed by Eq. 4.

$$S_{u,t,s} = \begin{cases} -\infty, & s > t \\ M_{u,t}, & s = t \\ \max_{v \in N(u)} \{S_{v,t-1,s} + M_{u,t}\}, & s < t. \end{cases} \tag{4}$$

Different from the previous dynamic programming in computing the matrix S, this new algorithm stores all possible solutions from all starting frames. When S is computed, the algorithm traces back for the best solution with all possible starting and ending points. This extension makes the complexity of the extended-algorithm $O(whn^2)$ to construct S and another $O(n)$ to search the best path, and needs $O(whn^2)$ memory. Taking the advantage of the trellis structure, the search complexity now is reduced to linear to the volume size times the length of the video. Based on this result, we will show how to further improve the search to reach the lowest complexity in the next section.

```
Input: $M(u,t)$: the local discriminative scores;
Output: $S(u,t)$: the accumulated scores of the best
        path leads to $(u,t)$;
$P(u,t)$: the best path record for tracing back;
$S^*$ : the accumulated score of the best path;
$l^*$ : the ending location of the best path;
begin
    $S(u,1) = M(u,1), \forall u$;
    $P(u,t) = null, \forall (u,t)$;
    $S^* = -\infty$;
    $l^* =$ null;
    for $i \leftarrow 2$ to $n$ do
        foreach $u \in [1..w] \times [1..h]$ do
            $v_0 \leftarrow \text{argmax}_{v \in N(u)} S(v, i-1)$;
            if $S(v_0, i-1) > 0$ then
                $S(u,i) \leftarrow S(v_0, i-1) + M(u,i)$;
                $P(u,i) \leftarrow (v_0, i-1)$;
            else
                $S(u,i) \leftarrow M(u,i)$;
            end
            if $S(u,i) > S^*$ then
                $S^* \leftarrow S(u,i)$;
                $l^* \leftarrow (u,i)$;
            end
        end
    end
end
```

Algorithm 1: Message forwarding algorithm

3.2. Our Proposed Max-Path Discovery

We now propose a new algorithm with message passing mechanism for the Max-Path discovery problem, with the complexity of only $O(whn)$. The algorithm consists of two steps: *message forwarding* and *path back-tracing*. The Algorithm 1 shows the message forwarding process. Following the notations, let $M(x,y,t)$ be the output predictions of the video. The message passing starts at $t = 1$, then propagates the information from the current frame to the next. Each node needs to store a message value $S(x,y,t)$, which is the maximum accumulated score of the best possible path up to (x,y,t). $P(x,y,t)$ is the previous node that leads to (x,y,t) in the best possible path. These values can be computed by collecting information from each node's neighbors and its local value $M(x,y,t)$. When the message reaches a node, the algorithm looks for the best value S of its neighbors from the previous frame. If this value is positive, then the path continues to grow from existing best path and stores the accumulated score and the previous position. Otherwise, the algorithm starts a new path from the current position. Figure 3 illustrates a concrete example of the algorithm.

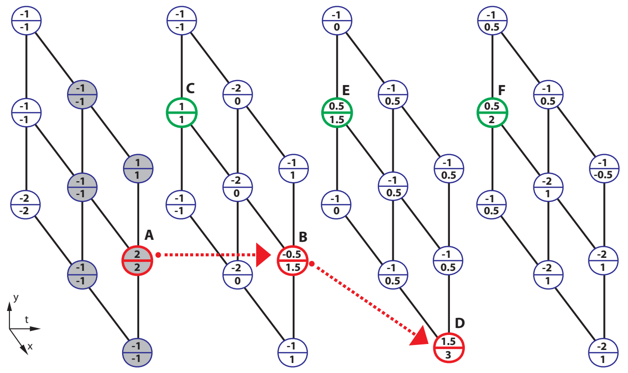

Figure 3. **A message passing example**: an example of Max-Path algorithm applied to a $3 \times 3 \times 4$ video. Each node is denoted with a local discriminative score (upper number), and the best accumulated score (lower number). In the first frame, all the best accumulated scores are initialized by their corresponding local discriminative scores. In the second frame, B can grow further from A which has the best accumulated score among B's neighbors (shaded nodes), while C needs to start a new path. The final best path is A-B-D (red nodes), and C-E-F is the second best path (green nodes).

Lemma 1. *$S(x,y,t)$ resulted from Algorithm 1 is the accumulated sum of the best path that leads to (x,y,t).*

Lemma 1 confirms the correctness of Algorithm 1. The formal proof of *Lemma 1* is provided in the appendix. Algorithm 1 will result in the best path value S^* and the ending point of the best path l^*. The localization of the best path is straightforward by looking at the values stored in P and tracing back until reaching a $null$ node. Overall, it takes $O(whn)$ to compute S, $O(n)$ to trace back the path, and uses $O(whn)$ memory to store S and P. The algorithm gives exactly the same results as the dynamic programming algorithm but reduces both computational and storage requirement.

As the size of the trellis is $w \times h \times n$, and one cannot find the maximum path sum without reading every element, $O(whn)$ is the lowest complexity we can expect. Together with *Lemma 1*, we thus have the following theorem.

Theorem 1. *Algorithm 1 results in the global optimal solution with a complexity of $O(whn)$, which is the lowest complexity of the Max-Path problem.*

3.3. Further Extensions of the Algorithm

Handling Multiple Scales: when the events appear across a wide range of scales, we can extend the sliding window scheme to multiple scales. Instead of sliding a fixed-scale window, at the same location v, one can use windows $W(v)$ with different scales. As a result, since each node is coupled with multiple windows with different scales, the trellis $\mathcal{G}_M$ becomes a 4D array with a size of $w \times h \times m \times n$ (m is the number of scales). The problem is now posed in 4D, but still can be solved by the same algorithm 1. One

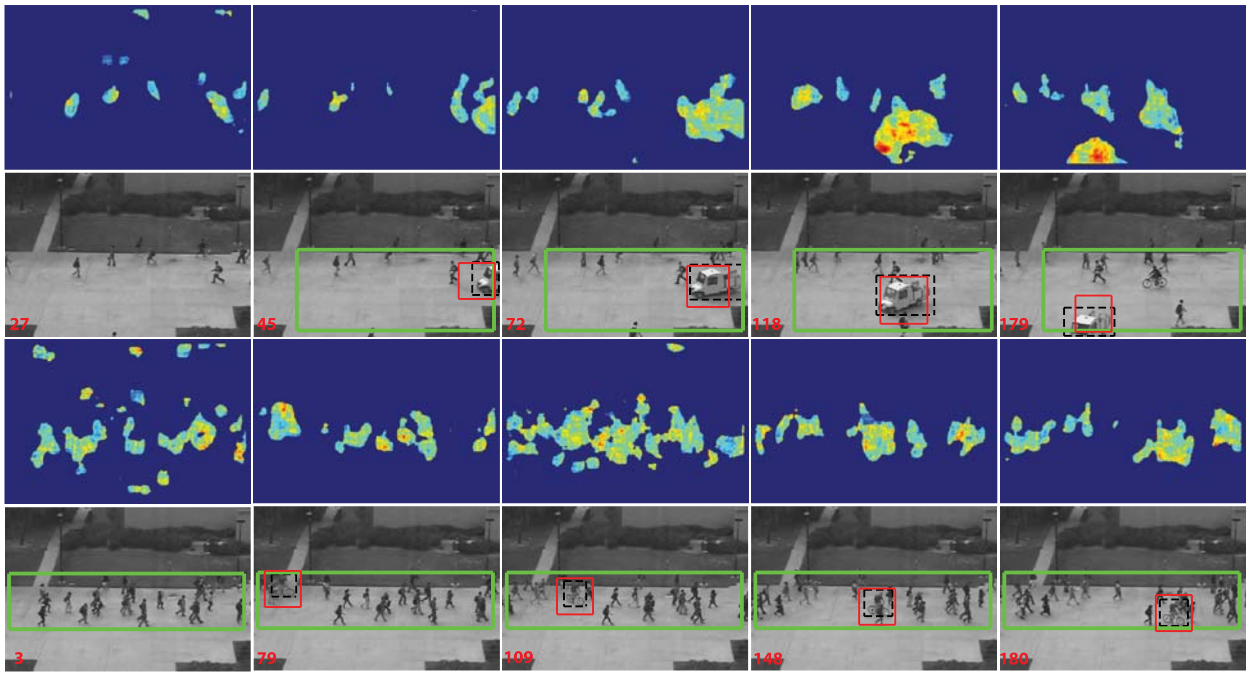

Figure 4. **Abnormal event detection**: demonstration of abnormal event detection. The odd rows are confidence maps, even rows are localization results. The results of subvolume search [22] are visualized in green boxes; the results of Max-Path search are in red; and groundtruth is in dashed-black. The subvolume search covers a large portion of the video. Max-Path locates events more accurately by relaxing the constraint, and it can automatically discover the starting and ending points of events.

difference is that the trellis is changed because each node now has neighbors not only from its same scale but also from its two nearest scales. More specifically, if a node has up to 9 neighbors for the single scale setting, it now may have 27 neighbors including 9 from its same scale and two other 9s from its two adjacent scales. In general, the algorithm's complexity and space cost will both be increased to $O(whmn)$.

Discovery of Multiple Paths: similar to non-maximum suppression or branch-and-bound [12, 22], this algorithm can also be applied repeatedly to locate multiple instances. After obtaining p^*, one can remove it from M and restart the process to search for the next best max-path.

Moving Speed Adaptation: one can instead use a larger neighborhood region to accommodate fast motions of the event. Edges of neighbors can be weighted by a Gaussian mask to control the smoothness of the spacial movement.

4. Application 1: Anomaly Event Detection

Datasets: we use UCSD abnormal event detection dataset [15] for evaluation. The dataset consists of two sections of two different scenarios. We use section 2, which consists of 16 training and 12 test sequences. Each sequence has about 180 frames. The training videos capture only normal motions of walking crowd, while the testing ones have abnormal motions such as bikers, skaters, and small carts. Only 8 of 12 testing sequences are provided with pixel-level binary mask groundtruth. As our target is to discover and to locate the events, only sequences with pixel-level groundtruth are evaluated.

Training: we firstly extract features at locations with notable motions in the training dataset, because abnormal events cannot happen without movements. The features we used are Histogram of Oriented Gradients (HOG) [3] and Histogram of Oriented Flows (HOF) [4] with 16×16 patches. Feature quantization is then performed using k-means clustering. These cluster centers are used as codewords.

Testing: on testing, at any location with motions, we compute features and the distance to its nearest codeword. These distances are then used as prediction values for the local pixels. A great distance implies a high likelihood of being abnormal. The pixels with no motion are assigned zero distances. To introduce negative values, we subtract these distances by a threshold. This distance map is now a 3D array of positive and negative scores which can be passed to the subvolume search [22] for event localization. For our approach, we assume that the abnormal events will occur across m different scales (*e.g.* $d_1..d_m$). We use *integral image* [19] to compute the sum scores of different scales local windows (*e.g.* squares with d_i-long sides) at each frame. This process results in a 4D discriminative score map which is then inputted to our Max-Path algorithm to discover ab-

Algorithm	Subvolume[22]	**Our Max-Path**
Average Accuracy	23.98	**60.20**

Table 1. **Abnormal event localization results**. Our Max-Path algorithm significantly improves the localization accuracy over subvolume search [22] thanks to the constraint relaxation.

normal events.

Results: for evaluations, we build the groundtruth by drawing bounding boxes around the provided masks of abnormal events. We use PASCAL metric (*e.g.* the overlapped area divided by the union of predicted and groundtruth boxes) to evaluate the localization accuracy. At every frame, if both prediction and groundtruth are positive, then the PASCAL metric is applied to compute the localization score. If both of them are negative, then the score is assigned 1, otherwise 0. Table 1 shows the average accuracy of abnormal event detection and localization. Our Max-Path algorithm significantly outperforms subvolume search more than 35% as a result of relaxing the 3D bounding box constraint. Figure 4 compares the results of our Max-Path search and the subvolume search [22]. The first two rows are from a relatively simple sequence while the last two rows are from a more difficult one with noisy motions of the walking crowd. In both cases, subvolume search predicts large volumes covering most of the video. Even though with a very noisy confidence map, our Max-Path search can locate events accurately. This is true because the false positives appear randomly at inconsistent spacial locations. In the long run, their accumulated scores cannot compete with those of the true event paths. On the other hand, the short-run missed or weak detections caused by occlusions can be resolved and linked to the main path as long as the final score can be further improved after the drops. Finally, experimental results showed that Max-Path search can automatically discover the starting and ending points of events.

5. Application 2: Walking Person Localization

Datasets: we use two datasets, *TUD-MotionPairs* [21] for training and our *YouTube Walking* for testing. *TUD-MotionPairs* is a fully annotated dataset containing image pairs of outdoor walking pedestrians for evaluating pedestrian detection algorithms that employ motion information. These image pairs include 1092 positive pairs, 192 negative pairs, and 26 additional negative pairs for further bootstrapping training. Our *Walking* dataset contains 2 long video sequences (800-900 frames per sequence) and 25 short video sequences (100-150 frames per sequence) downloaded from YouTube making a total of 4083 annotated bounding boxes. These videos are real-world sequences including outdoor walking and indoor fashion shows of catwalk models. The sequences are very challenging due to their low quality with compression artifacts, appearing in crowded scenes, many

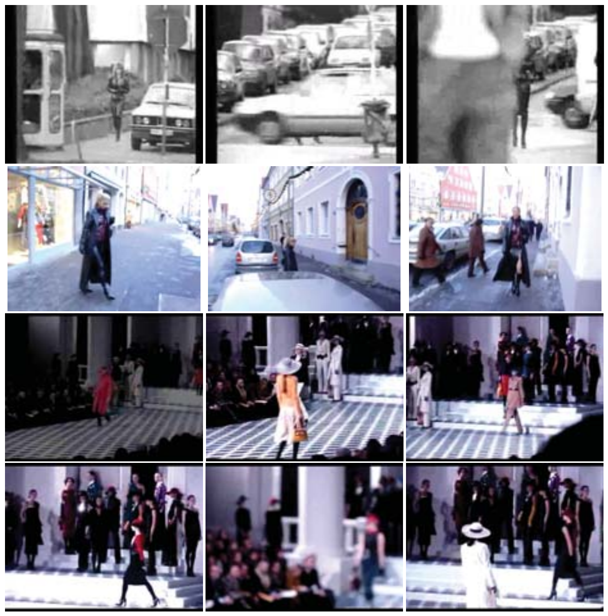
Figure 5. **Our walking dataset**: 27 realistic video sequences downloaded from YouTube. The two upper rows are snapshots of outdoor sequences. The two lower rows are those from indoor fashion shows. These realistic videos are at low quality, captured in crowded scenes with occlusions and complex background motions.

partly and fully occlusions, significant scale changes, different lighting conditions, noisy background motions, camera shaking motions (Figure 5).

Training a walker detector: we use a global representation of pedestrian by HOG [3], HOF (more specifically IMHd2), and simple Self-Similarity [20]. These features are then trained on a linear SVM with one more additional bootstrapping round on hard negative set of *TUD-MotionPairs*.

Walking localization algorithms: we slide the trained detector over the test sequence at multiple scales. The sliding process results in a 4D output prediction map, which is then passed to a localization algorithm to process. This map does often contain false positives and missed detections due to the imperfect base detector. For quantitative evaluations, we implement two baseline algorithms to locate walking. The first one is to simply choose the maximum detection score at every frame over all scales. It is actually a variant of non-maximum suppression, and it is more reasonable than non-maximum suppression provided that there is one walking person in the sequence. We call this algorithm *greedy-suppression*. Another baseline algorithm is the *spatiotemporal smoothing* which is straightforwardly averaging k-consecutive boxes, which are results from the greedy-suppression algorithm. Besides baseline algorithms, we also compare our framework to a tracking algorithm. We use the *Incremental Learning Tracking* (IL-Tracking)

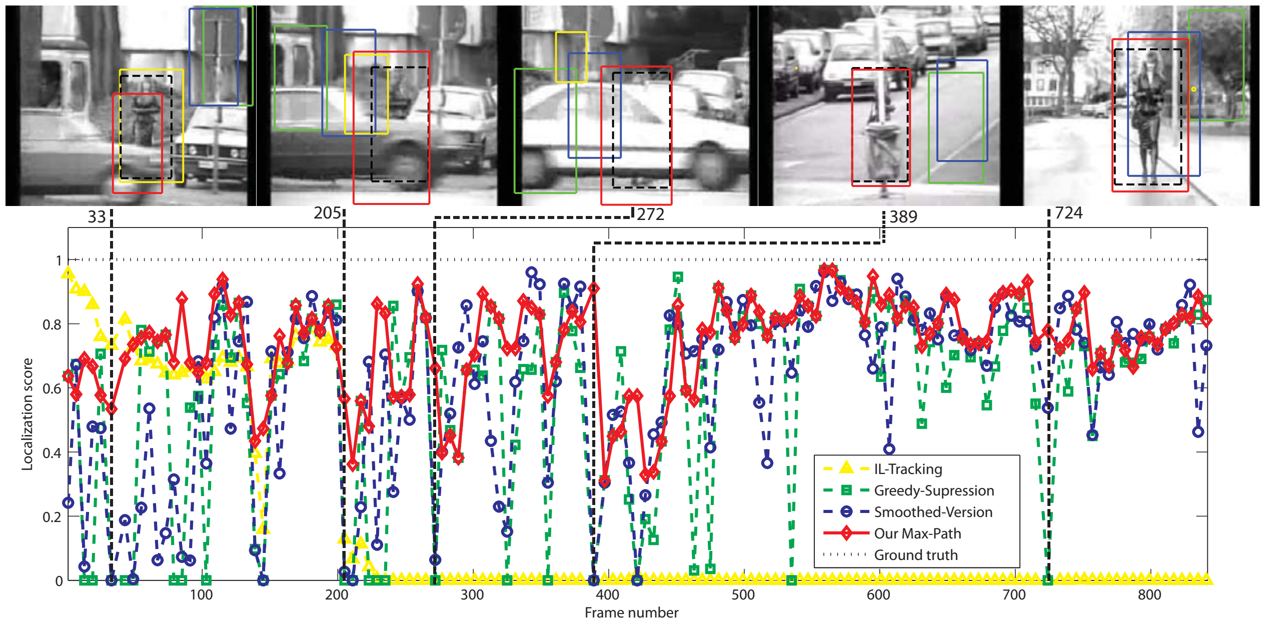

Figure 6. **Detection and localization results**: the plots of localization scores from different algorithms on an outdoor walking sequence with visualized snapshots. IL-Tracking [17] works only at the beginning, then loses the targets when occlusions occur. Greedy-suppression and smoothed-version perform poorly due to false positives and missed detections. Max-Path significantly outperforms the other algorithms with the globally optimized solution. The data points are dropped by ratio $1:7$ for better representation (best viewed in color).

with the source code provided by Ross *et al*[17]. The IL-Tracking algorithm is initialized by the ground-truth bounding box of the first frame. We note that the IL-Tracking is not directly comparable to the baselines and our max-path algorithm, because first it requires initialization and second it does not use the prediction map. These algorithms are then compared to our proposed Max-Path algorithm to demonstrate the effectiveness and robustness of our algorithm. In this experiment, we use the max-path algorithm with the multiple-scale extension. The node's neighbors are its 9-connected neighbors.

Results: we evaluate the localization accuracy in each frame by PASCAL metric, and report the average accuracy in Table 2. We also compare the behaviors of different algorithms on a long outdoor walking sequence in Figure 6. Our Max-Path algorithm improves 24-27% from the baseline algorithms. The IL-Tracking algorithm works only for a short time at the beginning, then loses the targets when occlusions occur. It works better on some other higher quality sequences but still loses the targets when partial occlusions present. The greedy-suppression algorithm suffers from false positives and missed detections. The spatiotemporal smoothing cannot make any difference from greedy-suppression, if not making it worse, due to highly noisy false detections. Finally, Max-Path algorithm provides significant improvements, thanks to its global optimal solution over all spatiotemporal and scale spaces.

Algorithm	Average accuracy
[17] Incremental Learning Tracking*	30.30
[20]+Greedy-Suppression	50.11
[20]+Spatiotemporal-Smoothing	47.47
Our Max-Path	**73.98**

Table 2. **Walking localization accuracy**. Our Max-Path algorithm improves 24-27% of accuracy compared to the baseline algorithms. *The tracking algorithm is not directly comparable.

Detecting of multiple walking pedestrians: we collect 2 sequences from YouTube consisting of 672 frames for evaluating this extension. These realistic sequences are the television news of the event in which President Barack Obama was walking into the White House on his inauguration day. The news videos contain many walking people. We apply the detector in section 5 with our proposed algorithm repeatedly to find the top $k = 5$ paths. Results are shown in Figure 7. It is worth noting that unlike multiple object tracking, we do not identify different persons due to the lack of appearance modeling. Instead, we only discover the top 5 best paths in the video.

6. Conclusions

We have proposed a novel algorithm for event detection and localization. Our proposed Max-Path discovery algorithm is proven to be very efficient, robust, and has many potential applications. The benefits of our method are three-fold. First, its proven lowest complexity makes it possible

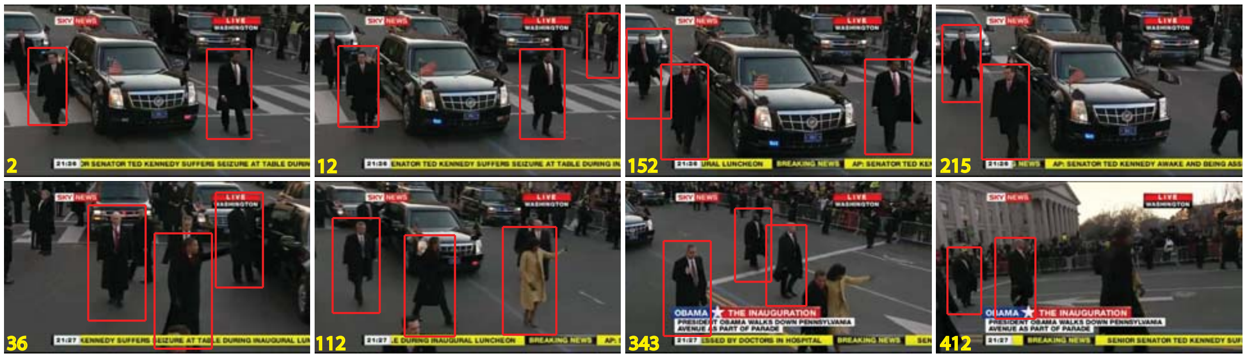

Figure 7. **Pedestrian detection in videos**: the sequences are challenging due to complex camera and background motions.

to search for the best path over a large 4D search space. Second, its global optimal solution guarantees the smoothness of event throughout the video, hence eliminates the false positives and alleviates missed or weak detections. Last but not least, the relaxation from the spatio-temporal subvolumes to spatio-temporal paths is more flexible for various applications. Our experiments on realistic videos have demonstrated favorable results.

Appendix: we prove the *Lemma 1* here. Let us define $\mathcal{Q}(t) \triangleq$ "*$S(x, y, t)$ as the maximum accumulated sum of the best path leading to* (x, y, t)". We will prove that $\mathcal{Q}(t)$ is true $\forall t \in [1..n]$ by induction. We initialize $S(x, y, 1) = M(x, y, 1), \forall(x, y)$, hence $\mathcal{Q}(1)$ is true. Assume that $\mathcal{Q}(k-1)$ is true, we now show $\mathcal{Q}(k)$ is also true. If a node u at frame k has m directly connected neighbors, then there are $m + 1$ possible paths leading to it. These paths include m paths going through its neighbors with an accumulated scores of $S(v, k-1) + M(u, k), v \in N(u)$ and another one starting by itself with a score of $M(u, k)$. From Algorithm 1, we have:

$$v_0 = \operatorname*{argmax}_{v \in N(u)} S(v, k-1) \tag{5}$$

$$\Rightarrow S(v_0, k-1) \geq S(v, k-1), \forall v \in N(u) \tag{6}$$

$$\Rightarrow S(v_0, k-1) + M(u, k) \geq S(v, k-1) + M(u, k), \quad \forall v \in N(u) \tag{7}$$

And also from the Algorithm 1, the *If* statement for assigning values to S indicates two cases that

$$S(u, k) = \begin{cases} S(v_0, k-1) + M(u, k), & S(v_0, k-1) > 0 \\ M(u, k), & \text{otherwise.} \end{cases} \tag{8}$$

$$\Rightarrow S(u, k) = \max\{S(v_0, k-1) + M(u, k), M(u, k)\} \tag{9}$$

From (7) and (9), we have shown that $S(u, k)$ is always the best accumulated sum compared to all $m + 1$ paths that can lead to u. This confirms that $\mathcal{Q}(k)$ is true.

Acknowledgements: This work is supported in part by the Nanyang Assistant Professorship (SUG M58040015) to Dr. Junsong Yuan.

References

[1] O. Boiman and M. Irani. Detecting irregularities in images and in video. *IJCV*, 2007.

[2] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Gool. Robust tracking-by-detection using a detector confidence particle filter. *ICCV*, 2009.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[4] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. *ECCV*, 2006.

[5] K. Derpanis, M. Sizintsev, K. Cannons, and P. Wildes. Efficient action spotting based on a spacetime oriented structure representation. *CVPR*, 2010.

[6] P. Dollr, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. *ICCV VS-PETS*, pages 65–72, 2005.

[7] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. *ICCV*, pages 726–733, 2003.

[8] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong, and T. S. Huang. Action detection in complex scenes with spatial and temporal ambiguities. *ICCV*, 2009.

[9] B. Jon. Programming pearls: algorithm design techniques. *Communications of the ACM*, 27(9):865–873, 1984.

[10] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. *ICCV*, 2007.

[11] Y. Ke, R. Sukthankar, and M. Hebert. Volumetric features for video event detection. *IJCV*, 2010.

[12] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2009.

[13] I. Laptev and T. Lindeberg. Space-time interest points. *ICCV*, 2003.

[14] B. Leibe, K. Schindler, and L. Gool. Coupled detection and trajectory estimation for multi-object tracking. *ICCV*, 2007.

[15] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. *CVPR*, 2010.

[16] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: A spatio-temporal maximum average correlation height filter for action recognition. *CVPR*, 2008.

[17] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 2008.

[18] S. Stalder, H. Grabner, and L. V. Gool. Cascaded confidence filtering for improved tracking-by-detection. *ECCV*, 2010.

[19] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

[20] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. *CVPR*, 2010.

[21] C. Wojek, S. Walk, and B. Schiele. Multi-cue onboard pedestrian detection. *CVPR*, 2009.

[22] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. *CVPR*, pages 2442–2449, 2009.

[23] J. Yuan, J. Meng, Y. Wu, and J. Luo. Mining recurring events through forest growing. *IEEE Trans. on Circuits and Systems for Video Technology*, 18(11):1597–1607, 2008.

[24] Z. Zhang, Y. Cao, D. Salvi, K. Oliver, J. Waggoner, and S. Wang. Free-shape subwindow search for object localization. *CVPR*, 2010.