Linear Time Offline Tracking and Lower Envelope Algorithms

Steve Gu
Computer Science
Duke University
steve@cs.duke.edu

Ying Zheng
Computer Science
Duke University
yuanqi@cs.duke.edu

Carlo Tomasi
Computer Science
Duke University
tomasi@cs.duke.edu

Abstract

Offline tracking of visual objects is particularly helpful in the presence of significant occlusions, when a frame-by-frame, causal tracker is likely to lose sight of the target. In addition, the trajectories found by offline tracking are typically smoother and more stable because of the global optimization this approach entails. In contrast with previous work, we show that this global optimization can be performed in $O(MNT)$ time for T frames of video at $M \times N$ resolution, with the help of the generalized distance transform developed by Felzenszwalb and Huttenlocher[13]. Recognizing the importance of this distance transform, we extend the computation to a more general lower envelope algorithm in certain heterogeneous l_1-distance metric spaces. The generalized lower envelope algorithm is of complexity $O(MN(M+N))$ and is useful for a more challenging offline tracking problem. Experiments show that trajectories found by offline tracking are superior to those computed by online tracking methods, and are computed at 100 frames per second.

1. Introduction

The state of the art of visual object tracking has advanced significantly in the past 30 years [25, 32, 21, 10, 1, 4, 22, 26, 35, 15, 16, 5, 30, 18]. Visual tracking is often cast "online," that is, solved as frames become available, because of the need for instant response to environmental changes or real time control and surveillance.

On the other hand, when a video is already recorded (e.g., on YouTube), online tracking is not necessary. "Offline" tracking can then use all the information available in the video for optimization, and is expected to result in smoother and more stable trajectories that can recover from more severe occlusions than the online methods can.

To cast tracking as an optimization problem, a measure $E(\mathbf{x}_i, \mathcal{M})$ is defined for the degree of *inconsistency* between an appearance model $\mathcal{M}$ and a rectangular window centered at $\mathbf{x}_i \in \Omega_i$ where Ω_i is the i^{th} image frame domain of size $M \times N$. Motion along the trajectory is assumed to be smooth, so that offline tracking can be cast as the problem (omitting explicit mention of the model $\mathcal{M}$ for brevity)

$$\hat{\Gamma} = \arg \min_{\Gamma \in \Omega_1 \times \cdots \times \Omega_T} \sum_{i=1}^{T} E(\mathbf{x}_i) \tag{1}$$

with the constraint that the distance $d(\mathbf{x}_{i+1}, \mathbf{x}_i)$ between $\mathbf{x}_{i+1}$ and $\mathbf{x}_i$ is less than ε for $1 \leq i \leq T-1$. Here and throughout the paper, d represents the l_1 distance metric. The number ε is a parameter that quantifies how much a window can move between consecutive frames, and the optimization variable $\Gamma = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ ranges over the set of all possible trajectories of length T bounded by the $M \times N$ pixel image rectangle.

This problem can be solved with dynamic programming, and the computation is fast in practice when ε is small. With increasing ε, on the other hand, the time complexity for the optimization approaches $O(M^2N^2T)$. This quadratic complexity in the number of pixels makes the algorithm rather slow. Imposing a tighter constraint ε is difficult, because such a hard constraint is inevitably arbitrary.

1.1. The Objective

In a recent paper [36], Uchida *et al.* replace the hard constraint by a regularization term:

$$\hat{\Gamma} = \arg \min_{\Gamma \in \Omega_1 \times \cdots \times \Omega_T} F(\mathbf{x}_1, \ldots, \mathbf{x}_T) \tag{2}$$

where the object takes the following form:

$$F(\mathbf{x}_1, \cdots, \mathbf{x}_T) = \sum_{i=1}^{T} \underbrace{E(\mathbf{x}_i)}_{\text{visual inconsistency}} + \lambda \sum_{i=1}^{T-1} \underbrace{d(\mathbf{x}_{i+1}, \mathbf{x}_i)}_{\text{motion discontinuity}} \tag{3}$$

In this formulation, the parameter λ balances the contributions from visual consistency and motion continuity, and this is preferable to a hard constraint. Uchida *et al.* [36] then states that problem (2) has time complexity $O(M^2N^2T)$. In order to accelerate the computation, the authors introduce the assumption that the inconsistency measure $E(\mathbf{x})$ is quadratic and constant.

1.2. Our Contributions

We improve on Uchida *et al.*[36] by showing that problem (2) can be solved without assumptions in time $O(MNT)$, using the generalized distance transform [13].

This transform can be viewed as a special lower envelope algorithm. We extend this class of computations to solve a more general optimization problem that is encountered in the offline tracking framework. We show that this extended version of offline tracking can be solved in $O(MN(M+N)T)$ time for the l_1 distance.

The literature of offline tracking is rich, ranging from single [6, 3, 33, 12, 38, 36] to multi-object tracking [20, 40, 27, 23]. Our work stands out in two aspects: First, in contrast to object tracking systems [40, 27, 23], our tracker works at the pixel level and does not use any object detectors (e.g. person). Second, our tracker is arguably the fastest among all the others, both asymptotically and practically.

2. Efficient Offline Tracking

We first write the state equations for the optimization (2) and show that they can be solved in $O(MNT)$ time. We then describe our design of the visual inconsistency measure $E(\mathbf{x}_i)$. Careful readers will notice that the generalized distance transform is used twice, for two different purposes.

2.1. State Equations

We first exhibit the structure of the optimization:

$$\min_{\Gamma} F(\mathbf{x}_1, \cdots, \mathbf{x}_T) = \min_{\mathbf{x}_T} \left[\underbrace{\min_{\mathbf{x}_1, \cdots, \mathbf{x}_{T-1}} F(\mathbf{x}_1, \cdots, \mathbf{x}_T)}_{G(\mathbf{x}_T)} \right]$$

$$= \min_{\mathbf{x}_T} \left\{ E(\mathbf{x}_T) + \underbrace{\min_{\mathbf{x}_{T-1}} \left[\underbrace{G(\mathbf{x}_{T-1})}_{\text{to be expanded}} + d(\mathbf{x}_T, \mathbf{x}_{T-1}) \right]}_{H(\mathbf{x}_T)} \right\}$$

The dynamic programming is then reduced to the following two alternating steps for $j = 1, 2, \cdots, T$ and each $\mathbf{x} \in \Omega_j$:

Step 1: $H(\mathbf{x}) \leftarrow \min_{\mathbf{y} \in \Omega_{j-1}} [G(\mathbf{y}) + \lambda d(\mathbf{x}, \mathbf{y})]$;

Step 2: $G(\mathbf{x}) \leftarrow E(\mathbf{x}) + H(\mathbf{x})$;

Clearly, step 2 takes $O(MN)$ time. Felzenszwalb and Huttenlocher [14] showed an $O(MN)$ algorithm for computing step 1 by using the generalized distance transform, which is equivalent to computing the lower envelope of a set of 2D cones with equal slopes (See Figure 1). Dynamic programming process lasts T steps, so that the overall time complexity is $O(MNT)$.

Figure 1. The lower envelope of a set of 1D homogeneous cones.

2.2. Evaluating Visual Inconsistency

We now describe the object model $\mathcal{M}$ and how to compute the inconsistency measure $E(\mathbf{x}; \mathcal{M})$. Our model is simple but effective, as experiments in Section 4 demonstrate. Of course, any available technique (e.g. [5, 11, 37, 34, 41, 28, 30, 39, 9]) can be used to achieve a desired goal and a compromise is typically necessary between running time and quality. For instance, a face detector could be used to score each window position in every frame. Also, visual inconsistency could be precomputed for each window position in each frame if desired.

For general object tracking, we use a simple tracking module from the literature [18], but introduce some variations to simplify the assignment of pixel costs. The idea for constructing object models and score evaluation is intuitive:

Step 1: Given an arbitrary frame and a specified window, compute the set of SIFT [24] features (other features [32, 7, 31, 17] can be used as well) and put all the features within the window into $\mathcal{M}$ and the rest into a background model $\mathcal{B}$.

Step 2: For each feature v in each frame, compute the cost $S(v)$ of selecting the feature v as follows:

$$S(v) = \frac{\min_{u \in \mathcal{M}} \|u - v\|}{\min_{u \in \mathcal{B}} \|u - v\|} . \tag{4}$$

The rationale behind this formula is that the cost of v is low if v resembles some object feature more than it does any background feature, and is high if the situation is reversed. Note that this cost assignment is parameter free.

Step 3: For each pixel p in each frame, assign the cost:

$$\tilde{S}(p) = \min_{q \in F} [S(q) + \xi d(p, q)] \tag{5}$$

where F is the set of SIFT features. The benefit of this cost model is that each pixel receives a cost that is determined not only by its closeness to extracted features but also to their own costs. Because the lower envelope is continuous, the cost function is continuous as well. This is (again!) in the form of the generalized distance

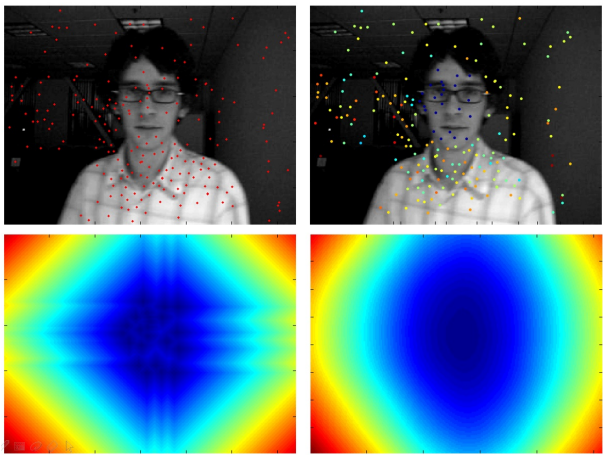

Figure 2. Top row. Left: the input image with extracted SIFT features. Right: the cost of each SIFT feature. The warmer the color, the higher the cost. Bottom row. Left: the 2D lower envelope induced by SIFT features. Right: the cost summation within each window. The pixel with lowest cost marks a tracked window.

transform described before, so the costs of each image can be computed in linear time with respect to the number of pixels.

Step 4: For each window W centered at $\mathbf{x}_i$, compute $E(\mathbf{x}_i) = \sum_{p \in W} \tilde{S}(p)$. This step can be computed in linear time with respect to the number of pixels in each frame because of the integral image representation and the evaluation step is only $O(1)$.

Figure 2 illustrates these four steps for cost assignment. In [18], scores where assigned with a binary threshold and each pixel either received a positive constant score or a negative constant score. In contrast, our more flexible score assignment is soft, and guarantees that the cost function is continuous. An additional advantage is that the entire process depends only on one parameter ξ.

In summary, we have presented not only a linear time algorithm for optimal trajectory estimation but also a fast way to evaluate visual consistency in a pre-processing step. The fast evaluation of visual consistency utilizes the modules from integral image representation and the generalized distance transform. The process is nearly parameter free except for ξ that is used in score evaluation.

3. Lower Envelope of Heterogeneous Cones

Recognizing the importance of the distance transform, we study how to extend it for more complex tracking scenarios. This section is mostly theoretical and readers who are interested in experiments can directly jump to Section 4 without affecting the flow of understanding.

In the generalized distance transform, all the 2D cones have equal slopes and differ only in their positions. We call these cones *homogeneous*. Equivalently, the regularization term λ has to be a single number for all the pixels in a video sequence. We study an adaptive regularization scheme so that λ is a general function defined on image grids.

Figure 3. Application scenarios where adaptive or pixel-by-pixel regularization is useful. From top to bottom: road, river and walkway. In each case, one can mark the specific region in order to encourage the target being tracked to stay within the region, thereby incorporating natural physical constraints.

3.1. Application Scenarios

There are at least two applications that can benefit from this generalization. First, consider surveillance applications where the background remains roughly static while the foreground object moves in a physically constraint fashion. For example, cars stay on the road, boats navigate a river, or pedestrians are required to stay on walkways. We can then allow a user to draw a curve that marks the centerline of road, river, or walkway so that λ is low near the marked curve and higher far away (See Figure 3). This does not rule out the possibility for the tracker to drift away from the centerline, but enforces physical constraints in a soft manner.

For a second application, suppose we are given an "objectness" measure (e.g. [2]) that tells how likely it is for each pixel to belong to an object rather than to the background. We can encode this information into λ so that the tracker integrates all information (motion continuity, visual consistency and objectness measure) into a single formula. Note that visual consistency and objectness measure provide two types of information that may not be correlated.

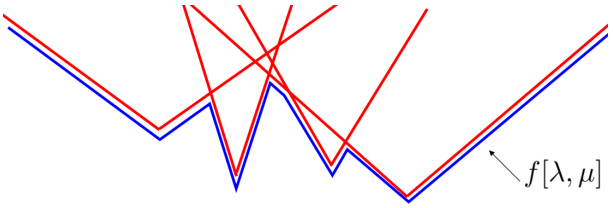

Figure 4. The lower envelope of a set of 1D heterogeneous cones.

3.2. The New Concept

We consider a more general distance transform, through the lens of a geometric lower envelope:

Definition 1 (Lower Envelope of Heterogeneous Cones)**.** *Let $\lambda : \Omega \to \mathbb{R}^+$ and $\mu : \Omega \to \mathbb{R}$ be two arbitrary functions. The lower envelope of a set of heterogeneous cones induced by λ and μ is a function $f[\lambda, \mu] : \Omega \to \mathbb{R}$ for each $p \in \Omega$:*

$$f[\lambda,\mu](p) = \min_{q\in\Omega} [\lambda(q)d(p,q) + \mu(q)] \tag{6}$$

Figure 4 shows the lower envelope of a set of 1D cones with different slopes. It is known [19] how to compute the 1D lower envelope in $O(n \log n)$ time. We show how to compute the 2D lower envelope in $O(n^{1.5})$ time in this paper. Our strategy is to first decompose the lower envelope of 2D cones to the lower envelope of 4 separate lower envelopes of $\frac{1}{4}$ cones. We then show the lower envelope of a set of $\frac{1}{4}$ 2D cones can be computed using results from 1D cones and hence save the computation. The resulting algorithm is also extremely simple to implement.

3.3. The Computation

We first divide the 2D cone $h_q(x) \triangleq \lambda(q)d(x,q) + \mu(q)$ into a composite of $\frac{1}{4}$ cones (as shown in Figure 5):

$$h_q = \min\left\{h_q^{++}, h_q^{+-}, h_q^{--}, h_q^{-+}\right\} \tag{7}$$

where $h_q^{++}(p) = h_q(p)$ if $p_x \geq q_x$ and $p_y \geq q_y$ and $h_q^{++} = +\infty$ otherwise. The other $\frac{1}{4}$ cones are defined similarly by taking care of the signs properly. We therefore can write:

$$f[\lambda,\mu] = \min_{q\in\Omega} h_q \tag{8}$$

$$= \min_{q\in\Omega} \min\left\{h_q^{++}, h_q^{+-}, h_q^{--}, h_q^{-+}\right\} \tag{9}$$

$$= \min\left\{\min_{q\in\Omega} h_q^{++}, \min_{q\in\Omega} h_q^{+-}, \min_{q\in\Omega} h_q^{--}, \min_{q\in\Omega} h_q^{-+}\right\} \tag{10}$$

In other words, the lower envelope of a set of 2D heterogeneous cones is equivalent to the lower envelope of the four lower envelopes of $\frac{1}{4}$ cones. It then suffices to understand how to compute the lower envelope of $\frac{1}{4}$ cones. We only study the properties of h_q^{++}, the rest being handled similarly.

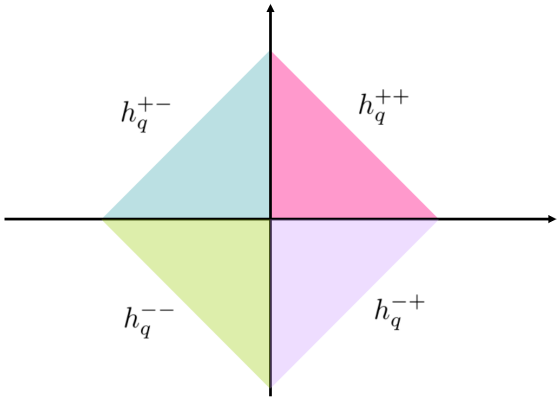

Figure 5. Division of a 2D cone (visualized above) into four separate $\frac{1}{4}$ cones with different colors.

Writing $q = (x_0, y_0)$, we note that $h_q^{++}(x,y)$ is a constant for $x + y = C, x \geq x_0$ and $y \geq y_0$. Let then $h_{q_1}^{++}, \cdots, h_{q_k}^{++}$ be a set of 2D cones with $q_i = (x_i, y_i)$. If we take the upper right corner $(\hat{x}, \hat{y})$ with $\hat{x} = \max_{i=1}^k x_i$ and $\hat{y} = \max_{i=1}^k y_i$, we then have the fact that $\min_{i=1}^k h_{q_i}^{++}(x,y)$ is a constant for $x + y = C, x \geq \hat{x}, y \geq \hat{y}$. In particular, this is true when these $\frac{1}{4}$ cones lie in a single column (See Figure 6).

We discuss how to use the above observation to design a more efficient algorithm than the naive one. First, we realize that for a pixel p located at (m, n), only those pixels lying to the lower left of p (including p itself) can affect the lower envelope at p by the definition of h^{++}. Let this rectangular set be $S = \{(i,j) \mid 1 \leq i \leq m, 1 \leq j \leq n\}$. We use $L(S,p)$ to represent the value of the lower envelope at p induced by S. We now decompose the set S into n columns $\{C_j\}_{j=1}^n$ so that $C_j = \{(i,j) \mid 1 \leq i \leq m\}$. By the definition of lower envelope we have:

$$L(S,p) = \min_{j=1}^{n} L(C_j, p) \tag{11}$$

But from the above observation we know that $L(C_j,p) = L(C_j,q)$ if $q = (m+n-j, j)$. The difference is that q lies in the same column as the pixels in C_j. Therefore, computing $L(C_j,q)$ is essentially a one dimensional problem and so is computing $L(C_j,p)$. To make this clear, we introduce a matrix A so that each column of A, namely $A(:,j)$ if we use MATLAB notation, stores the 1D lower envelope induced by C_j. We see that:

$$L(S,p) = \min_{j=1}^{n} A(m+n-j, j) \tag{12}$$

that is, the minimal value along the diagonal $x+y = m+n$. This explains how the 2D lower envelope can be reduced to

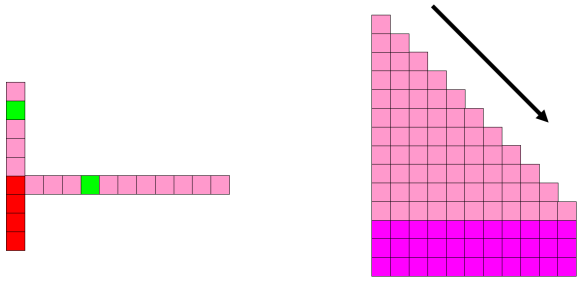

Figure 6. Left: the lower envelope induced by the dark red dots share the same value at two green dots . Right: the illustration of the construction of the 2D lower envelope from bottom to top.

alternatingly evaluating the 1D lower envelope and then taking the minimal values along the diagonals. This computation is sketched in Algorithm 1. The first inner double loop updates the 1D lower envelope of each column after a new row is inserted. The second inner double loop computes the values of the 2D lower envelope on the newly inserted row by taking the minimal values along the diagonals.

Algorithm 1 Compute the 2D lower envelope of $\frac{1}{4}$ cones given two $M \times N$ matrices λ and μ

```
Initialize a matrix A of size (M + N) × N to +∞;
for r = 1 to M do
  for c = 1 to N do
    for i = r to M + N do
      A(i, c) ← min {A(i, c), μ(r, c) + λ(r, c)(i − r)}
    end for
  end for
  for c = 1 to N do
    for i = 1 to c do
      A(r, c) ← min {A(r, c), A(r + c − i, i)}
    end for
  end for
end for
```

3.4. Analysis

Figure 6 illustrates the process of the algorithm. The first double inner for loop takes $O(MN(M+N))$ time and the second takes $O(MN^2)$ time. If we let $M = N = \sqrt{n}$, the overall time complexity is $O(n^{1.5})$. Once we compute the four separate lower envelopes of $\frac{1}{4}$ cones, merging them together takes only linear time because the min operation for each pixel is constant. Thus the overall time complexity for computing the lower envelope of a set of heterogeneous cones is $O(n^{1.5})$.

In contrast with the lower envelope of a set of 2D homogeneous cones, which can be computed in linear time, we can only compute the lower envelope of 2D heterogeneous cones in $O(n^{1.5})$ time where n is the number of pixels of an image. It remains an intellectual challenge to see if an $O(n \log n)$ algorithm is possible for the 2D case. Also note that our presented algorithm utilizes the special structure of the l_1 distance. It seems not obvious how to generalize the algorithm to the case of squared l_2 distances.

4. Experiments

Our experiments focused on the linear time offline tracking algorithm. The second algorithm – for the lower envelope of heterogenous cones – is of more theoretical interest and its implementation and experiments are available in our companion website (See Section 4.3).

The goal is to verify that our method produces smoother and more accurate results than online tracking methods. Our method allows performing this task in linear time without using any heuristics, and this is asymptotically as fast as it gets without introducing approximations.

4.1. Experimental Setting

Our experiments involve the two parameters ξ in equation (5) and λ in equation (3). ξ balances feature locations and costs when computing pixel matching costs. λ trades off the contributions from visual inconsistency (summation of pixel costs within a window) and motion discontinuity. In all the experiments, we fix $\xi = 0.01$ and $\lambda = 50$. The relative size of these parameters reflects our observation that a window contains on the order of 1000 pixels.

For simplicity and fair comparison, we do not use any advanced face detectors or previously learnt object models. In order to label object and background features, we randomly select 3 frames in a video sequence and ask a user to mark the object to be tracked. The object model and the background model are then the SIFT features within and outside the chosen windows, respectively. Models for online tracking methods can be updated frame by frame while for offline tracking both the object and background models are static. Even so, we show that the global trajectories from offline tracking are often better than those from online tracking, because both visual inconsistency and motion discontinuity are minimized globally rather than locally.

4.2. Comparisons

We test our linear time offline tracking algorithm on 8 video sequences from multiple data sets [8, 29, 1, 5, 30] in the literature and compare it to the following state-of-art online tracking methods: FragTracker [1], Multiple Instance Learning [5], PROST [30] and Nearest Neighbor [18]. This comparison is mainly proof-of-concept because the nature of two methods is quite different. However, given the good performance of these online methods, it is certainly nontrivial for our offline method to do any better because of the simplicity of our formulation of the cost function.

Table 1. Mean distance error. Bold: best. Underlined: second best.

Sequences	Frag	MIL	PROST	NN	Offline
Girl [8]	26.5	31.6	19.0	<u>18.0</u>	**12.7**
David [29]	46.0	<u>15.6</u>	15.3	<u>15.6</u>	**14.3**
Faceocc1 [1]	**6.5**	18.4	<u>7.0</u>	10.0	13.3
Faceocc2 [5]	45.1	14.3	17.2	<u>12.9</u>	**12.5**
Board [30]	90.1	51.2	37.0	<u>20.0</u>	**13.8**
Box [30]	57.4	104.6	**12.1**	16.9	<u>15.3</u>
Lemming [30]	82.8	**14.9**	25.4	79.1	<u>24.0</u>
Liquor [30]	30.7	165.1	21.6	<u>15.0</u>	**14.2**

The disadvantage of our offline tracking method is that both the object and background models are static and hence there is no adaptation to the appearance change. Online trackers typically adapt to the appearance change well by their design. Therefore, if our method can do any better, it must benefit from a combination of global motion continuity and visual consistency.

In the comparison, we directly quote previously reported results [30, 18]. This comparison is summarized in Table 1. The evaluation criterion is the same as what is used in [5, 30] except that we only compute the mean distance error e. This is because the scale of the tracked objects stay relatively the same so this measure of error reflects the true behavior of each tracker well. The mean distance error is:

$$e = \frac{1}{n}\sum_{i=1}^{n} \|O_i - O_i^g\| \tag{13}$$

where n is the number of frames and $\|O_i - O_i^g\|$ is the Euclidean distance between the tracked window centroid O_i and the ground truth window centroid O_i^g. The ground truth is manually labeled by different authors in the works cited.

Table 1 shows that our offline tracking algorithm does better than most online tracking algorithms, although there are some exceptions. The most significant improvements are in the 'girl' and 'board' sequences, which exhibit large occlusions and appearance changes. Our algorithm successfully extracts the trajectory for both video sequences. In other sequences, our tracker either wins, or loses by little. Figure 7 shows sample error plots for different trackers.

The advantage of our method is even more obvious from a computational point of view. For instance, finding the optimal trajectory for a video sequence of 500 frames of resolution 400×300 takes less than 3 seconds on a quad-core laptop computer, corresponding to a processing speed of more than 100 frames per second.

4.3. Reproducibility

All the algorithms and described steps in this paper can be implemented easily. Our experimental platform is written in MATLAB while the integral image and the 2D lower envelopes are computed in MEX/C++. The implementation and the test code that reproduces the results is available at `http://www.cs.duke.edu/~steve`.

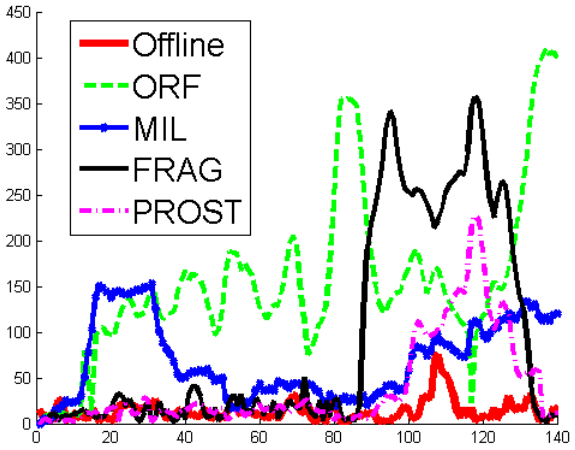

Figure 7. Error plots (measured in pixels) for the board sequence. The error against the ground truth is evaluated every 5 frames.

5. Conclusions and Future Work

The major contributions of this paper are twofold: First, we show that optimal offline tracking can be solved in linear time in the size of the video. This fact makes the global trajectory estimation highly efficient and practical. Second, we generalize the notion of distance transform to the lower envelope of a set of heterogeneous cones and show that this can be computed in $O(n^{1.5})$ where n is the number of pixels of an image. It remains an open question whether an $O(n \log n)$ algorithm exists for this task.

The disadvantage of the current formulation is that the appearance models are determined *a priori*, and hence our offline tracking method cannot adapt nimbly to changes of appearance over time. One simple remedy works as follows: First, find an initial trajectory with our algorithm and then use features on the entire trajectory to train a better object model. This process can possibly be repeated until convergence. Of course, such an approach will slow down computation considerably. How to achieve a good balance between result quality and computation cost is an interesting trade-off we plan to explore.

Acknowledgement: This work is supported by the National Science Foundation under Grant No. IIS-1017017 and by the Army Research Office under Grant No. W911NF-10-1-0387.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *IEEE CVPR*,

pages 798–805, 2006.

[2] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *IEEE CVPR*, pages 73–80, 2010.

[3] J. Arnold, S. Shaw, and H. Pasternack. Efficient target tracking using dynamic programming. *IEEE Trans. Aerospace and Electronic Systems*, 29:44–56, 1993.

[4] S. Avidan. Ensemble tracking. *IEEE PAMI*, 29(2):261–271, 2007.

[5] B. Babenko, M. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *IEEE CVPR*, pages 983–990, 2009.

[6] Y. Barnoiv. Dynamic programming solution for deteting dim moving target. *IEEE Trans. Aerospace and Electronic Systems*, 21:144–156, 1985.

[7] H. Bay, T. Tuytelaars, and L. Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417, 2006.

[8] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *IEEE CVPR*, pages 232–237, 1998.

[9] L. Breiman. Random forests. *Mach. Learning*, 45(1):5–32, 2001.

[10] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE CVPR*, pages 2142–, 2000.

[11] T. Dietterich, R. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artif. Intell.*, 89(1-2):31–71, 1997.

[12] P. Dreuw, T. Deselaers, D. Rybach, D. Keysers, and H. Ney. Tracking using dynamic programming for appearance-based sign language recognition. In *FG*, pages 293–298, 2006.

[13] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, 2004.

[14] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

[15] H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE CVPR*, pages 260–267, 2006.

[16] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, pages 234–247, 2008.

[17] S. Gu, Y. Zheng, and C. Tomasi. Critical nets and beta-stable features for image matching. In *ECCV*, pages 663–676, 2010.

[18] S. Gu, Y. Zheng, and C. Tomasi. Efficient visual object tracking with online nearest neighbor classifier. In *ACCV*, 2010.

[19] S. Gu, Y. Zheng, and C. Tomasi. Extended pairwise potentials. In *CVPR Workshop on Inference in Graphical Models with Structured Potentials*, 2011.

[20] M. Han, W. Xu, H. Tao, and Y. Gong. An algorithm for multiple object trajectory tracking. In *IEEE CVPR*, pages 864–871, 2004.

[21] M. Isard and A. Blake. A smoothing filter for condensation. In *ECCV*, pages 767–781, 1998.

[22] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *IEEE PAMI*, 30(10):1728–1740, 2008.

[23] Y. Li, C. Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *IEEE CVPR*, pages 2953–2960, 2009.

[24] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

[25] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981.

[26] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(3):448–461, 2010.

[27] A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *IEEE CVPR*, pages 666–673, 2006.

[28] C. Prakash, B. Paluri, S. Pradeep, and H. Shah. Fragments based parametric tracking. In *ACCV*, pages 522–531, 2007.

[29] D. Ross, J. Lim, R. Lin, and M. Yang. Incremental learning for robust visual tracking. *IJCV*, 77(1-3):125–141, 2008.

[30] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST Parallel Robust Online Simple Tracking. In *IEEE CVPR*, 2010.

[31] E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *IEEE CVPR*, 2007.

[32] J. Shi and C. Tomasi. Good features to track. In *IEEE CVPR*, pages 593 – 600, 1994.

[33] J. Sun, W. Zhang, X. Tang, and H. Shum. Bi-directional tracking using trajectory segment analysis. In *ICCV*, pages 717–724, 2005.

[34] M. Tian, W. Zhang, and F. Liu. On-line ensemble svm for robust object tracking. In *ACCV*, pages 355–364, 2007.

[35] C. Tomasi, S. Petrov, and A. Sastry. 3d tracking = classification + interpolation. In *ICCV*, pages 1441–1448, 2003.

[36] S. Uchida, I. Fujimura, H. Kawano, and Y. Feng. Analytical dynamic programming tracker. In *ACCV*, 2010.

[37] P. Viola, J. Platt, and C. Zhang. Multiple instance boosting for object detection. In *NIPS*, 2005.

[38] Y. Wei, J. Sun, X. Tang, and H. Shum. Interactive offline tracking for color objects. In *ICCV*, pages 1–8, 2007.

[39] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic huber-l1 optical flow. In *BMVC*, September 2009.

[40] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *IEEE CVPR*, 2008.

[41] X. Zhao and Y. Liu. Generative estimation of 3d human pose using shape contexts matching. In *ACCV*, pages 419–429, 2007.