Twisted Window Search for Efficient Shape Localization

Steve Gu
Duke University
Durham, NC 27708, USA
steve@cs.duke.edu

Ying Zheng
Duke University
Durham, NC 27705, USA
yuanqi@cs.duke.edu

Carlo Tomasi
Duke University
Durham, NC 27705, USA
tomasi@cs.duke.edu

Abstract

Many computer vision systems approximate targets' shape with rectangular bounding boxes. This choice trades localization accuracy for efficient computation. We propose twisted window search, a strict generalization over rectangular window search, for the globally optimal localization of a target's shape. Despite its generality, we show that the new algorithm runs in $O(n^3)$, an asymptotic time complexity that is no greater than that of rectangular window search on an image of resolution $n \times n$. We demonstrate improved results of twisted window search for localizing and tracking non-rigid objects with significant orientation, scale and shape change. Twisted window search runs at nearly 10 frames per second in our MATLAB/C++ implementation on images of resolution 240×320 on a quad-core laptop.

1. Introduction

Despite the use of a wide variety of features and scoring schemes, many algorithms in object recognition [31, 32, 20, 10, 11] and tracking [24, 8, 1, 3, 21, 26, 4, 25, 27, 22] can be summarized as finding a single rectangular window over which some additive score is maximized. The naive algorithm for this task takes $O(n^4)$ on an image of $n \times n$ pixels. Throughout the paper we assume that an image is square and contains n rows and n columns. Lampert et al. [20] accelerate the average running time of the exhaustive search to $O(n^2)$ using a branch and bound technique, although the worst case complexity remains $O(n^4)$. This algorithm was subsequently improved to $O(n^3)$ in worst case by An [2] and was further applied in human action categorization [35] and object tracking [16] with augmented priors.

A rectangle only approximates a target's shape loosely. Most daily life objects have much more complex shapes than rectangles. We propose to represent targets' shape using *twisted windows*, a generalization of rectangular windows. Twisted windows encompass the entire class of convex shapes and some non-convex families. Figure 1 illustrates that twisted windows are rich.

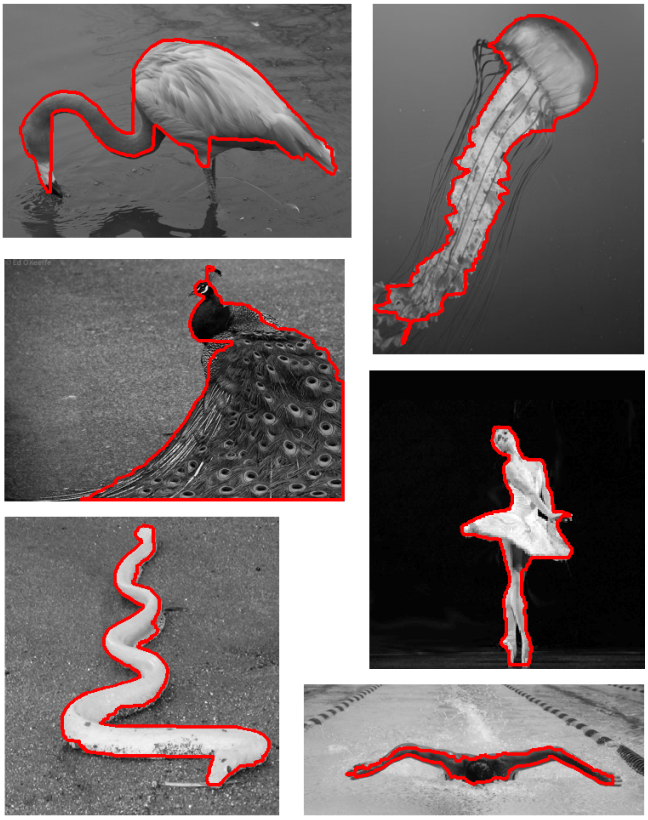

Figure 1. Example results of twisted window search on flamingo, jellyfish, peacock, dancer, snake and swimmer. Scores are generated in Section 4. All the images and videos used in this paper are downloaded from Google Image and YouTube. The global optimization takes less than 0.1 seconds for each image running on a quad-core laptop. The C++/MATLAB implementation is available at http://www.cs.duke.edu/~steve/twists.html

The literature in shape localization is rich in both non-rigid object tracking [18, 9, 28, 7, 19, 14, 17, 15] and object detection [34, 36, 30]. The techniques used in non-rigid object tracking include active contours, level sets [18, 9], background subtraction [17], Hough transform [14, 15], and

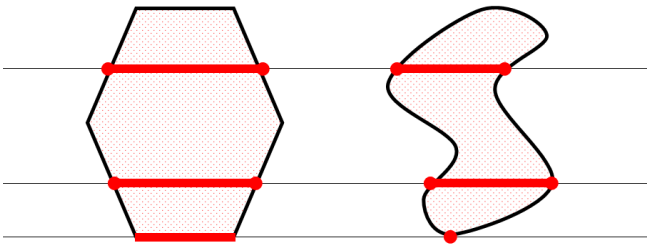

Figure 2. A twisted window intersects an arbitrary horizontal line in at most one connected component. This connected component may be a line segment lying in the shape interior or boundaries. It can also be degenerated to a single point on the boundary.

segmentation [7]. These methods typically rely on sampling techniques in local regions and do not exhaustively search for all the possible shapes in an image.

Shape localization methods have also been studied in object detection (e.g. [36, 30]). Most algorithms require intermediate representations and abstractions. In the work of [36], the shape is represented by a ratio contour [33]. Their search time complexity is super quadratic, too slow for near real time performance. In [30], a segment graph is first built from images and the optimization is run on the graph rather than on individual pixels.

Different from these works, we search for the optimal twisted window in the entire image domain and at the pixel level. This accounts for even extreme position drift and significant shape deformation. The worst case time complexity of twisted window search is practically identical to that of the optimal subwindow search [1], and is adequate for near real time performance on images of moderate size. Moreover, our method is independent of image segmentation and requires no intermediate abstractions.

Recently, Felzenszwalb and Veksler [13] gave a tiered scene labeling algorithm using dynamic programming. Although their objective and application scenarios are very different from ours, we find their algorithm closely related to twisted window search. The difference is that their objective is cast in a Markov Random Field and allows only local, pairwise smoothness penalties. In our formulation, the smoothness penalty is imposed globally and allows long-range shape deformation. Our algorithm follows the theme line of the optimal subwindow search, and utilizes Kadane's linear time algorithm for maximum-sum subarray [6] and the generalized distance transform [12] for finding an optimal twisted window with maximal score sum.

[1]The worst-case time complexity of optimal subwindow search is slightly better than $O(n^3)$ [29]. But that algorithm is of more theoretical than practical interest due to its leverage of fast matrix multiplication.

2. Twisted Window Search

We start with a formal definition in the continuous setting and conclude this section with a discrete version of twisted windows. The twisted window search is cast as a global discrete optimization on an image grid.

A twisted window is any connected 2D shape that intersects an arbitrary horizontal line (or an arbitrary vertical line) in at most one connected component (Figure 2). In this paper, we only look at horizontal lines but both horizontal and vertical cases are considered twisted windows.

Definition 1 (Twisted window) *A connected 2D shape $S \subset \mathbb{R}^2$ is a twisted window if $l \bigcap S$ has at most one connected component for an arbitrary horizontal line l.*

A twisted window is necessarily a shape without holes, since otherwise there are at least two connected components in $l \bigcap S$ if l passes through a hole. The shape class of twisted windows is quite general in the sense that any convex shape is a twisted window! Twisted windows also include many non-convex shapes. See Figure 1 and Figure 2.

2.1. Discrete Twisted Window

In the discrete setting an image function is defined on a grid Ω of n rows and n columns. The continuous twisted window then needs a proper translation to the discrete domain. We consider two mappings f and $g : [1, \cdots, n] \rightarrow [1, \cdots, n]$, which map row indices to column indices. We say f is dominated by g at interval $[T, B] \subseteq [1, n]$, denoted $f \prec_{[T,B]} g$, if $f[r] \leq g[r]$ for each $T \leq r \leq B$. A discrete twisted window is then described by its top T, bottom B, and two sides $f \prec_{[T,B]} g$. The definition follows:

Definition 2 (Discrete Twisted Window) *A set $S \subseteq \Omega$ is a discrete twisted window if there exists $T \leq B$ and $f \prec_{[T,B]} g$ so that $S = \{(r, c) \in \Omega \mid T \leq r \leq B, f[r] \leq c \leq g[r]\}$.*

The space of all twisted windows on the image grid Ω is denoted $\mathcal{T}(\Omega)$. By listing the top left and lower right corner of a rectangle, we know that there are in total $O(n^4)$ rectangular windows in Ω. The interesting question is how many discrete twisted windows there are on the image grid Ω. The answer is easy: For each (T, B) pair we have $\binom{n}{2}^{B-T+1}$ combinations for the choice of f and g, the total number of discrete twisted windows in Ω is therefore:

$$|\mathcal{T}(\Omega)| = \sum_{1 \leq T \leq B \leq n} \binom{n}{2}^{B-T+1} \sim O(n^{2n})$$

Despite the dramatic difference in their search spaces (polynomial versus exponential), we show below that twisted window search has the same (!) asymptotic time complexity as rectangular subwindow search using dynamic programming. We turn to the optimization part.

2.2. Global Optimization

Let $\xi : \Omega \to \mathbb{R}$ be a score function that takes both positive and negative values. The most intuitive definition for twisted window search would be:

$$\max_{T \le B, f \prec_{[T,B]} g} \sum_{T \le r \le B} \sum_{f[r] \le c \le g[r]} \xi(r, c) \qquad (1)$$

Unfortunately the cost $\xi(r, c)$ is imperfect in practice and a twisted window without proper constraints would be very sensitive to noisy score functions. We therefore use *regularization*, a commonly used strategy for constraining ill posed solutions, for controlling the smoothness of f and g. The smoothness of f is measured by the negative of total variation of f, defined as:

$$\sigma(f) = -\|\nabla f\|_1 = -\sum_{r=1}^{n-1} \mid f[r+1] - f[r] \mid \qquad (2)$$

Here we use the L_1 norm of the gradient of f although other measures such as the squared Euclidean works equally well. Clearly the larger $\sigma(f)$ is, the smoother f becomes.

The global optimization of the twisted window search therefore integrates both scores and smoothness considerations. We have the following definition:

Definition 3 (Twisted Window Search) *Let $\lambda > 0$ be a regularization parameter that balances the data scores and the smoothness measure. The twisted window search on a 2D image grid is cast as the following global optimization:*

$$\max_{f \prec_{[T,B]} g} \left\{ \lambda \left[\underbrace{\sigma(f) + \sigma(g)}_{\textit{smoothness}} \right] + \underbrace{\sum_{T \le r \le B} \sum_{f[r] \le c \le g[r]} \xi(r, c)}_{\textit{data scores}} \right\} \qquad (3)$$

We remark that as $\lambda \to +\infty$, the twisted window search is equivalent to the rectangular window search because then both f and g are enforced to be vertical line segments. It is in this sense that twisted window search is a strict generalization over rectangular window search.

3. Algorithm

We show how to do the twisted window search of Equation (3) in $O(n^3)$ time where n is the number of rows or columns of an input image. This time complexity is asymptotically equivalent to the subwindow search given in [2].

Our algorithm utilizes two dynamic programming procedures. The first is known as Kadane's algorithm for finding the maximum-sum subarray [6]. The second is known as the generalized distance transform [12], and is equivalent to computing the lower envelope of cones or parabolas on an image grid. Combined together they produce an efficient algorithm for twisted window search.

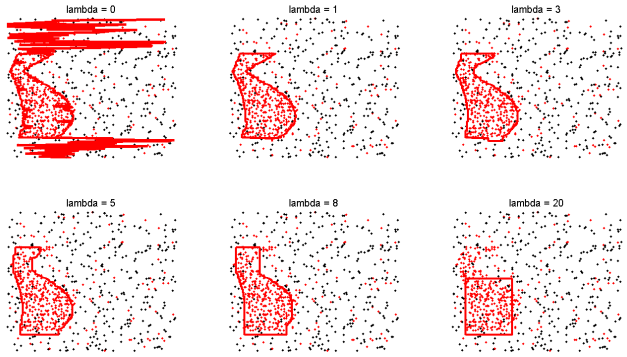

Figure 3. Twisted window search for $\lambda = 0, 1, 3, 5, 8, 20$. Red and black dots receive positive and negative scores respectively. $\lambda = 0$ corresponds to the twisted window search without regularization and is very sensitive to imperfect scores. When λ is large (e.g. $\lambda = 20$), the twisted window search is equivalent to rectangular window search. We fix $\lambda = 1$ throughout this paper.

We first note that the score sum of any interval along each row can be evaluated in $O(1)$ time if we precompute an integral image. That is, $\sum_{f[r] \le c \le g[r]} \xi(r, c) = F_r(f[r], g[r])$ where F_r is a $n \times n$ matrix and $F_r(a, b)$ memorizes the summation of scores from column a to column b at row r when $a \le b$. When $a > b$, we set $F_r(a, b) = -\infty$. Equation (3) can be expanded and simplified as:

$$\max_{T,B:1 \le T \le B \le n} \max_{f,g} \left\{ \sum_{r=T}^{B} F_r(f[r], g[r]) - \lambda \sum_{r=T}^{B-1} (|f[r+1] - f[r]| + |g[r+1] - g[r]|) \right\} \qquad (4)$$

where we have dropped the condition $f \prec_{[T,B]} g$ because the cost $F_r(a, b) = -\infty$ whenever $a > b$, and such a combination is avoided implicitly because of the maximization.

To further expose the structure of the optimization, it is best to view $(f[r], g[r])$ as a 2D point p_r rather than two separate numbers. The benefit is that the smoothness term $|f[r+1]-f[r]|+|g[r+1]-g[r]|$ is now simplified to $\|p_{r+1}-p_r\|_1$, the L_1 norm of the 2D vector difference $p_{r+1} - p_r$. Equation (4) can further be simplified and written as:

$$\max_{T \le B} \max_{\{p_r\}_{r=T}^{B}} \left\{ \sum_{r=T}^{B} F_r(p_r) - \lambda \sum_{r=T}^{B-1} \|p_{r+1} - p_r\|_1 \right\} \qquad (5)$$

3.1. Kadane's Idea

Let $E_r(p_r)$ be the optimal score at point p_r of row r. There are two choices: Either the top side of the twisted window starts at row r, and in this case the score is simply

Figure 4. Twisted window search vs. subwindow search [20, 2].

the data score $F_r(p_r)$ alone. Or the twisted window continues from the previous row and in this case the score is:

$$F_r(p_r) + \underbrace{\max_{p_{r-1}} \{E_{r-1}(p_{r-1}) - \lambda \|p_r - p_{r-1}\|_1\}}_{S_r(p_r)} \quad (6)$$

where $S_r(p_r)$ accumulates the previous data scores and the consecutive smoothness scores. The final score $E_r(p_r)$ is then the maximum of the scores produced by the two choices, and can be compacted into the following state equation:

$$E_r(p_r) = F_r(p_r) + \max\{S_r(p_r), 0\} \quad (7)$$

This equation means that whenever the score $S_r(p_r)$ drops below zero, we can safely discard the previous solution and place the top side of the twisted window at the current row. This idea is essentially the same as Kadane's algorithm for computing the maximum-sum subarray. To find the optimal solution one needs to memorize the p_r that receives the maximal score $E_r(p_r)$, and a linear scan suffices for tracing back the twisted window. $E_r(p_r)$ is evaluated for each p_r at each r ($O(n^3)$ positions in total) and each evaluation involves computing $S_r(p_r)$ which takes $O(n^2)$ time if computed naively. The overall time complexity is therefore $O(n^5)$. Next we improve the complexity to $O(n^3)$ using the generalized distance transform.

3.2. Felzenszwalb's Distance Transform

The challenge is to compute $S_r(p_r)$ in amortized $O(1)$ time instead of $O(n^2)$ time for each p_r. We recognize that S_r is in the form of a generalized distance transform:

$$\begin{aligned} S_r(p_r) &= \max_{p_{r-1}} \left\{ \underbrace{E_{r-1}(p_{r-1})}_{\triangleq -\mu(p_{r-1})} - \lambda \|p_r - p_{r-1}\|_1 \right\} \\ &= \min_{p_{r-1}} \{\mu(p_{r-1}) + \lambda \|p_r - p_{r-1}\|_1\} \end{aligned} \quad (8)$$

which is equivalent to computing the lower envelope of cones placed at a two dimensional grid. It is known [12] that this lower envelope can be computed in $O(n^2)$ time on a $n \times n$ matrix. The amortized complexity for evaluating each single $S_r(p_r)$ is therefore $O(1)$.

In summary, the optimization of Equation (3) can be computed in $O(n^3)$ time. We remark that this complexity remains the same even if we replace the L_1 norm in Equation (2) with the squared Euclidean metric or a truncation is applied because the generalized distance transform works for all three cases with the same time complexity. This concludes the computation for twisted window search.

4. Experiments

We apply twisted window search to the tracking and localization of non-rectangular, non-rigid objects. There is a single parameter λ in twisted window search. To understand its effect we first do the twisted window search on synthetically generated scores (Figure 3) with different choices of λ. It is clear that a tiny $\lambda \approx 0$ makes the localization sensitive to imperfect scores while a large λ reduces the twisted window search to optimal subwindow search. We find empirically that $\lambda \in [1, 3]$ works generally well provided that both positive and negative scores are of the same order of magnitude. We fix $\lambda = 1$ throughout the experiments.

4.1. Score Generation

In the first frame, we require a user to mark the object to be tracked by clicking 5 points in the interior of the object and an additional 5 points in the background. For each point we collect its R,G,B colors stacked as a 3D vector. We keep features simple because we expect the global optimization of twisted window search to compensate for this simplicity. More sophisticated, well engineered, high dimensional features such as SIFT [23] or SURF [5] should enhance the robustness of feature matching in object tracking, but with a greater running time. They fit into our framework for score generation as well, as exposed below.

Let $\mathcal{O}$ be the collection of object features and $\mathcal{B}$ be the collection of background features. Let v_p be the 3D color vector of pixel p. Then for each pixel p, we compute the ratio $\gamma(p)$ between the distances between v_p and its nearest neighbors in $\mathcal{B}$ and $\mathcal{O}$ respectively:

$$\gamma(p) = \frac{\min_{u \in \mathcal{B}} \|u - v_p\|}{\min_{u \in \mathcal{O}} \|u - v_p\|} \quad (9)$$

and we assign the score based on the ratio:

$$\xi(p) = \begin{cases} +1 & \text{if } \gamma(p) > \frac{3}{2} \\ -1 & \text{if } \gamma(p) < \frac{2}{3} \\ -\frac{1}{2} & \text{otherwise} \end{cases} \quad (10)$$

We remark that the numbers $\frac{3}{2}$ and $\frac{2}{3}$ are also used in SIFT [23] for the robust discriminative feature matching.

Table 1. Twisted window search vs. rectangular window search

	pheasant	plane	fish1	dolphin	diver
Baseline ([20, 2])	0.41	0.48	0.76	0.54	0.52
Twisted	**0.69**	**0.75**	**0.91**	**0.85**	**0.77**

	jellyfish1	jellyfish2	fish2	surfer
Baseline ([20, 2])	0.60	0.80	0.40	0.47
Twisted	**0.86**	**0.94**	**0.86**	**0.82**

We run twisted window search on the image with computed scores twice, once horizontally and once vertically. The final shape is the one that has the maximal score. In each successive frame, we compute the scores using the same formula of Equation (10). Both the object and background models are updated for adapting to appearance changes. This is realized by adding sampled features on the tracked object and background respectively, and excluding features that live beyond a chosen time stamp. The selected features anchored to the first frame are kept for reference. We use no motion or locality constraints to showcase that twisted window search is a global optimization that handles extreme position and shape changes.

4.2. Proof of Concept

We collect 9 simple videos from YouTube and provide detailed shape masks as ground truth for selected frames. Let M be the mask shape and T be a twisted window. The accuracy is measured by the ratio:

$$\rho(M,T) = \frac{|M \bigcap T|}{|M \bigcup T|} \tag{11}$$

It is clear that $\rho(M,T) = 1$ if $M = T$ and $\rho(M,T) = 0$ if $M \bigcap T = \emptyset$. Since we use particularly simple features, we focus on twisted window's ability to localize highly deformed objects. We treat subwindow search (e.g. [20, 2]) as the baseline method, and compare twisted window search to subwindow search using the same pixel scores.

Table 1 compares subwindow search and twisted window search in terms of average accuracy. It is clear that twisted window search leads to better shape localization in all the test cases. Since both subwindow search and twisted window search are computed on the same scores and have the same worst case complexity, the results showcase the advantages of twisted window search over rectangular window search for objects of complex shapes. Figure 4 gives a visual comparison of subwindow search and twisted window search. More results are in Figure 6.

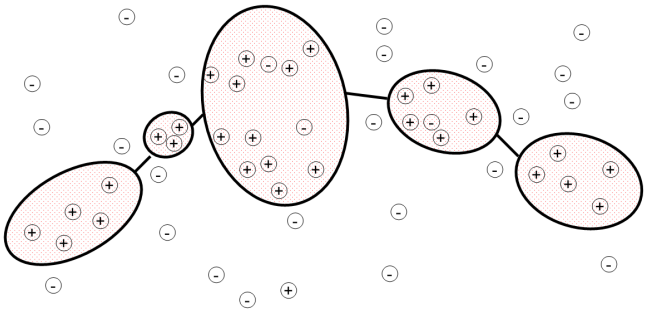

Figure 5. Twisted window search may connect multiple large positive regions into a single one. This presents difficulty for tracking a single object in the presence of multiple objects of similar appearance. One simple solution is to cut touching boundaries and take the connected component with the largest score summation.

4.3. Caveats and Remedies

We find that twisted window search works generally well when positive scores are highly *concentrated*. There is a subtle issue when positive scores are distributed in multiple locations (e.g. in the presence of multiple similar objects). The phenomenon is that the optimal twisted window tries to connect multiple positive regions with very thin strips of typical width only 1 pixel (Figure 5). This is logical because the overall score summation is maximized when multiple positive regions are combined with minimal negative paths. The easiest way to avoid this unnatural shape interpretation is perhaps to cut the found twisted window to multiple pieces. The cut is performed at 1-pixel width strips. This can be realized by traversing the boundary of the twisted window only once and the additional processing time is minimal compared to the $O(n^3)$ algorithm.

On the other hand, such a shape interpretation may be useful for tracking a group of objects as a whole rather than separately. We therefore let users decide whether a post processing step is necessary after twisted window search.

4.4. Running Time

Twisted window search is as efficient as the optimal subwindow search in terms of worst case complexity. In practice we find it runs in 10 frames per second on images of resolution 240×360 in our MATLAB/C++ implementation on a quad core computer. The code is available at `http://www.cs.duke.edu/~steve/twists.html`.

5. Conclusions

We propose twisted window search as a strict generalization of rectangular subwindow search and demonstrate improved localization of non-rectangular, non-rigid objects. We show that twisted window search, cast as a global

optimization at pixel level, can be solved in $O(n^3)$ time on an image of n rows and columns. We favor twisted window over rectangular subwindow search because it requires the same amount of computation but achieves better shape localization. It remains an open question whether there is a faster algorithm for twisted window search.

Acknowledgement: This work is supported by the Army Research Office under Grant No. W911NF-10-1-0387 and by the National Science Foundation under Grant IIS-10-17017.

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *IEEE CVPR*, pages 798–805, 2006. 1

[2] S. An, P. Peursum, W. Liu, and S. Venkatesh. Efficient algorithms for subwindow search in object detection and localization. In *IEEE CVPR*, pages 264–271, 2009. 1, 3, 4, 5

[3] S. Avidan. Ensemble tracking. *IEEE PAMI*, 29(2):261–271, 2007. 1

[4] B. Babenko, M. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *IEEE CVPR*, pages 983–990, 2009. 1

[5] H. Bay, T. Tuytelaars, and L. Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417, 2006. 4

[6] J. L. Bentley. *Programming pearls*. Addison-Wesley, 1986. 2, 3

[7] C. Bibby and I. Reid. Robust real-time visual tracking using pixel-wise posteriors. In *ECCV*, pages 831–844, 2008. 1, 2

[8] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE CVPR*, pages 2142–, 2000. 1

[9] D. Cremers. Dynamical statistical shape priors for level set-based tracking. *IEEE PAMI*, 28(8):1262–1273, 2006. 1

[10] M. Everingham, L. Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html. 1

[11] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE PAMI*, 32(9):1627–1645, 2010. 1

[12] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, 2004. 2, 3, 4

[13] P. Felzenszwalb and O. Veksler. Tiered scene labeling with dynamic programming. In *IEEE CVPR*, pages 3097–3104, 2010. 2

[14] J. Gall, N. Razavi, and L. V. Gool. On-line adaption of class-specific codebooks for instance tracking. In *BMVC*, pages 1–12, 2010. 1

[15] M. Godec, P. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In *ICCV*, 2011. 1

[16] S. Gu, Y. Zheng, and C. Tomasi. Efficient visual object tracking with online nearest neighbor classifier. In *ACCV*, pages 271–282, 2010. 1

[17] J. Henriques, R. Caseiro, and J. Batista. Globally optimal solution to multi-object tracking with merged measurements. In *ICCV*, 2011. 1

[18] M. Isard and A. Blake. A smoothing filter for condensation. In *ECCV*, pages 767–781, 1998. 1

[19] J. Kwon and K. Lee. Tracking of a non-rigid object via a patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *IEEE CVPR*, pages 1208–1215, 2009. 1

[20] C. Lampert, M. Blaschko, and T. Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE PAMI*, 31(12):2129–2142, 2009. 1, 4, 5

[21] 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. 1

[22] B. Liu, J. Huang, L. Yang, and C. Kulikowski. Robust tracking using local sparse appearance model and k-selection. In *IEEE CVPR*, pages 1313–1320, 2011. 1

[23] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999. 4

[24] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981. 1

[25] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE PAMI*, 32(3):448–461, 2010. 1

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

[27] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST Parallel Robust Online Simple Tracking. In *IEEE CVPR*, 2010. 1

[28] S. Shahed, J. Ho, and M. Yang. Visual tracking with histograms and articulating blocks. In *IEEE CVPR*, 2008. 1

[29] T. Takaoka. Efficient algorithms for the maximum subarray problem by distance matrix multiplication. *Electr. Notes Theor. Comput. Sci.*, 61:191–200, 2002. 2

[30] S. Vijayanarasimhan and K. Grauman. Efficient region search for object detection. In *IEEE CVPR*, pages 1401–1408, 2011. 1, 2

[31] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004. 1

[32] P. Viola, J. Platt, and C. Zhang. Multiple instance boosting for object detection. In *NIPS*, 2005. 1

[33] S. Wang, T. Kubota, J. Siskind, and J. Wang. Salient closed boundary extraction with ratio contour. *IEEE PAMI*, 27(4):546–561, 2005. 2

[34] T. Yeh, J. Lee, and T. Darrell. Fast concurrent object localization and recognition. In *CVPR*, pages 280–287, 2009. 1

[35] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *IEEE CVPR*, pages 2442–2449, 2009. 1

[36] Z. Zhang, Y. Cao, D. Salvi, K. Oliver, J. Waggoner, and S. Wang. Free-shape subwindow search for object localization. In *CVPR*, pages 1086–1093, 2010. 1, 2

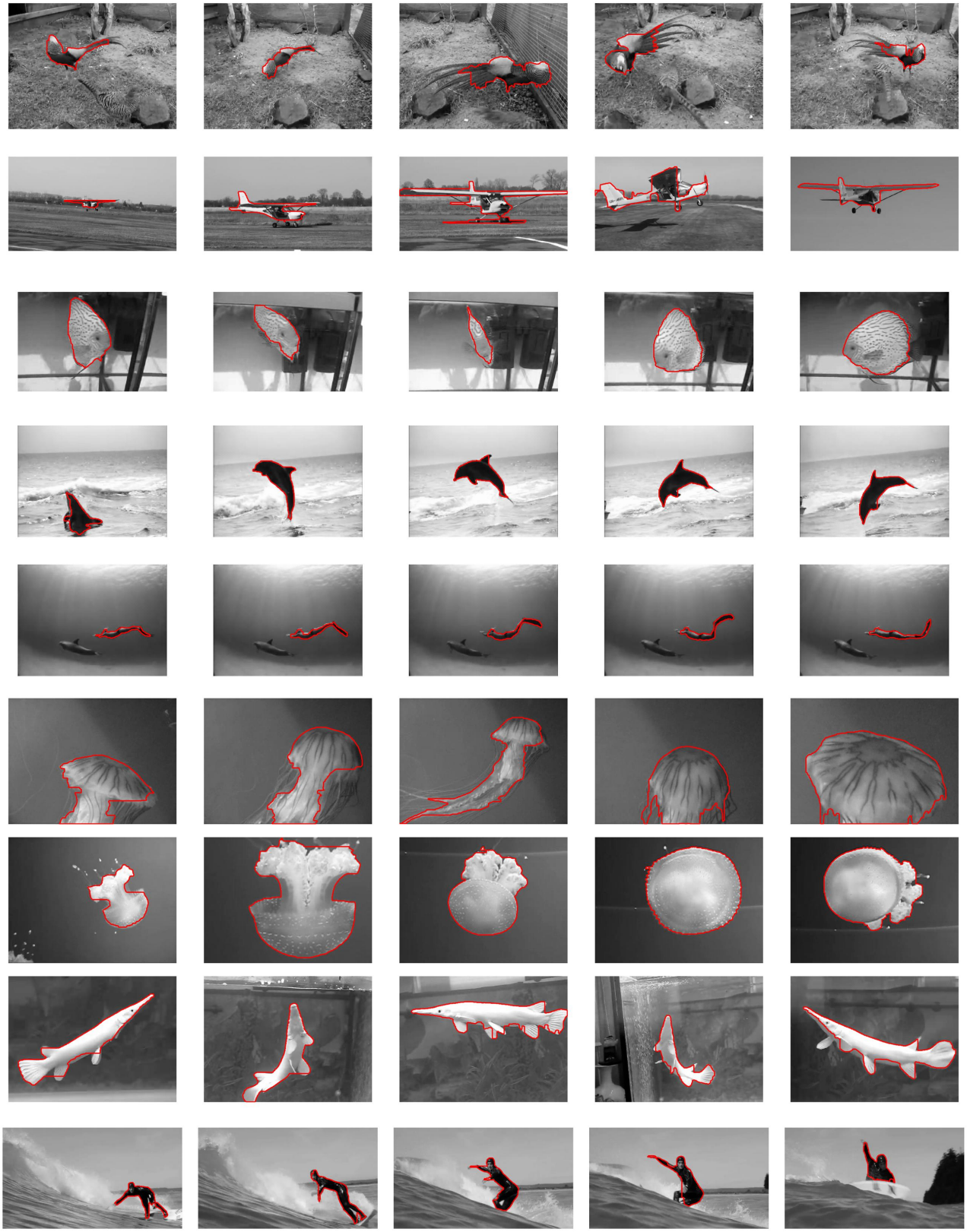

Figure 6. From top to bottom: pheasant, plane, fish1, dolphin, diver, jellyfish1, jellyfish2, fish2 and surfer. Twisted windows overlayed.