Track validation using gradient-based normalised cross-correlation

Darren Caulfield
Darren.Caulfield@cs.tcd.ie
Kenneth Dawson-Howe
Kenneth.Dawson-Howe@cs.tcd.ie

GV2 Group
School of Computer Science
Trinity College Dublin
Ireland

Abstract

We develop a gradient-based normalised cross-correlation tracker that is as robust as brute-force template matching while being significantly more computationally efficient. The technique serves as the basis of our *track validation* algorithm: by tracking an object forwards in time, reinitialising at the end of the sequence and then tracking backwards in time, we can determine whether or not the object has been followed correctly – the forwards and backwards trajectories will be very different for tracking failures. If such a failure occurs, we iteratively attempt to validate shorter portions of the video sequence until validation is achieved. The algorithm provides a means of determining whether or not an object was tracked successfully without the need for ground truth data.

1 Introduction

Normalised cross-correlation (NCC) is often used as the similarity measure in template matching trackers. However, NCC, when used with a brute-force search strategy, is computationally expensive. In this paper we develop a gradient ascent version of NCC; it retains the robustness of its brute-force counterpart, but it executes much more quickly. Indeed, its speed is comparable to that of another data-driven tracker – the mean-shift method – but it is much less likely to lose track of its target.

In real-world systems, it is often important to know when tracking has failed so that we may take corrective action. Ideally, such notifications will not rely on ground truth data or human monitoring of the system. We propose the alternative approach of using our gradient-based NCC technique as the basis of a *track validation* algorithm. By tracking a target forwards in time through the sequence, reinitialising the tracker with a new model taken from the end of the video, and following the target backwards in time, we can judge, in the absence of ground truth data, whether or not the tracking was successful: a large divergence between the forwards and backwards trajectories (*failed validation*) indicates that the object was lost by the tracker at some point. In such a situation the algorithm iteratively attempts to validate shorter subsequences of the video. A successful validation allows the method to switch to a new model of the target, which it uses in an effort to validate the object's trajectory in the remainder of the sequence. We apply the algorithm to all of the videos in our test set of 21 sequences; the results show that it enables a target to be tracked even as it

BMVC 2011 http://dx.doi.org/10.5244/C.25.70

undergoes severe appearance changes, and it also draws attention to parts of the video and objects that are proving difficult to track.

The main contributions of the paper are: the development of an efficient, robust, gradient-based NCC tracker; a quantitative assessment of the robustness of trackers based on mean shift, NCC and SSD; and the specification of an algorithm that uses the new technique to determine whether or not the tracking of a target was successful without resorting to ground truth data.

Section 2 describes related research in the areas of data-driven tracking and track validation. In section 3 we derive our gradient-based NCC tracker and compare its performance to mean shift and to SSD-based approaches. Our *track validation* algorithm is presented in section 4, along with an assessment of its performance. Finally, we discuss possible applications of the techniques and suggest some future research directions.

2 Related work

2.1 Data-driven tracking

Data-driven, or bottom-up, tracking techniques are characterised by the absence of high-level information, such as motion models, to follow targets through a video sequence. In this paper, we focus on region-based object representations used in data-driven tracking.

At one end of the spectrum, a region can be represented in a very detailed manner, e.g., by its template. *Template matching* can then be used to track the region over time. Lewis [9] presents a number of optimisations that can be applied when calculating the normalised cross-correlation (NCC) of a template and an image. To further lower the computational cost, Schweitzer *et al.* [15] develop a method that uses the "integral images" of Viola and Jones [19] to find a fast approximation to NCC. However, basic template matching of this kind can only accommodate translational image motion. In contrast, the approach of Hager and Belhumeur [7] extends the Lucas–Kanade optical flow method [10] to account for scaling and rotation.

At the other end of the spectrum, an image region can be summarised by its histogram, a less detailed but more compact representation. Histogram-based approaches are also amenable to gradient-based tracking methods, e.g., mean shift [5], which are computationally efficient. Bradski's *CAMSHIFT* system [1] was one of the first to use the mean-shift technique [4] for tracking. Later, Comaniciu *et al.* [5] extended mean shift to use spatial kernels

In section 3 we develop a gradient-based tracker that represents the image region to be tracked by its template. It combines the computational efficiency of mean shift with the detailed object representation (and therefore enhanced robustness) afforded by a template.

2.2 Track validation

Our implementation of track validation is based on a simple principle: tracking an object forwards in time through a sequence, stopping, reversing the sequence and tracking the object backwards in time should yield two very similar trajectories, provided the object has been tracked correctly. Here, we review some related approaches.

Given the starting and ending locations (and models) for an object to be tracked, Sun *et al.* [17] determine the full trajectory of the object between the two points in spite of occlusions. They first extract short track segments that they hypothesise correspond to the

object. A segment is only accepted as being part of the final trajectory if it can be joined with others to form a smooth path in space-time. Wu *et al.* [20] develop the "time-reversibility constraint" and apply it to the KLT tracker [16]. A tracked image point is retained by the method only if it moves in a similar direction when it is tracked forwards and backwards in time. The constraint is only applied to pairs of frames, and not to the entire video sequence. A related approach was developed be Kalal *et al.* [8]

The approach most closely related to our work is the "recurrent tracking" of Pan *et al.* [14] An object is first tracked forwards in time through the sequence. Tracking is then reinitialised with a new model taken from tracker's final location, and the object is followed backwards in time through the sequence. A significant difference between the forwards and backwards trajectories signifies that tracking has failed at some point. In section 4 we present our *track validation* algorithm. Unlike Pan's approach, it attempts to concatenate short sections of the video sequence so that the entire trajectory of the object can be validated.

3 Gradient-based normalised cross-correlation

The robustness of normalised cross-correlation (NCC) comes at the expense of relatively high computational complexity, especially when compared to mean-shift tracking. We therefore seek to develop a hybrid technique that combines the efficient gradient ascent search strategy of mean shift with the robustness of NCC. In the following section, we present the derivation of such a tracker and describe the iterative tracking algorithm that it employs to converge to its target in each frame of the video sequence. Next, we show how the tracker relates to popular types of brute-force template matching, and also to the Lucas–Kanade optical flow method. We run versions of these trackers on our video test set, measuring their robustness, accuracy and speed.

3.1 Derivation

We begin our derivation by defining certain entities, following the image sequence notation of Hager and Belhumeur [7]. Let $I(\mathbf{x},t)$ be the image at time t, where $\mathbf{x} \triangleq (x,y)$ is a point in the image. Let $Q \triangleq I(\mathbf{x},0)$ be the target (image template) we wish to track, and let $I \triangleq I(\mathbf{x}+\mathbf{u},t)$ be a candidate image region in the current frame. The similarity $O(\mathbf{u})$ of a template Q and an image patch I displaced from the template by $\mathbf{u}$ is:

$$O(\mathbf{u}) \triangleq \frac{1}{n}\sum_{\mathbf{x}\in R}(I-\bar{I})(Q-\bar{Q}) = \frac{1}{n}\sum_{\mathbf{x}\in R} I(Q-\bar{Q}) \tag{1}$$

where n is the number of pixels in the image patch and R is the set of pixel locations in the template. (See the supplemental material for a mathematical justification for disregarding $\bar{I}$.) The above formula is the same as that for normalised cross-correlation, except that we do not divide by the standard deviations of Q and I. Furthermore, equation 1 applies to single-channel images; for multiple-channel data we simply sum the contributions from the individual channels.

We can approximate I by the low-order terms of its Taylor series around $(\mathbf{x},t)$:

$$I = I(\mathbf{x}+\mathbf{u},t) \approx I(\mathbf{x},t) + u_1 I_x(\mathbf{x},t) + u_2 I_y(\mathbf{x},t) + (t-t)\times I_t(\mathbf{x},t) \tag{2}$$

where $\mathbf{u} \triangleq (u_1, u_2)$. The last term in the above equation contains the factor $(t-t)$. We have included this term, which evaluates to zero, to emphasise that the Taylor series expansion is indeed performed around the point $(\mathbf{x}, t)$, even though the final approximation does not make use of any past or future frames. The expressions

$$I_x(\mathbf{x},t) \triangleq \frac{\partial I(\mathbf{x},t)}{\partial x}; \qquad I_y(\mathbf{x},t) \triangleq \frac{\partial I(\mathbf{x},t)}{\partial y}; \qquad I_t(\mathbf{x},t) \triangleq \frac{\partial I(\mathbf{x},t)}{\partial t} \tag{3}$$

are the spatial and (unneeded) temporal image derivatives, respectively. We use image differences to approximate the required derivatives:

$$I_x(\mathbf{x},t) \triangleq I(\mathbf{x},t) - I\Big((x+1,y),t\Big); \qquad I_y(\mathbf{x},t) \triangleq I(\mathbf{x},t) - I\Big((x,y+1),t\Big) \tag{4}$$

Simplifying equation 2, we obtain:

$$I(\mathbf{x}+\mathbf{u},t) \approx I(\mathbf{x},t) + u_1 I_x(\mathbf{x},t) + u_2 I_y(\mathbf{x},t) \tag{5}$$

Our goal is to find a local maximum of the similarity measure O as we vary the displacement $\mathbf{u}$; therefore, we differentiate O with respect to $\mathbf{u} = (u_1, u_2)$:

$$\frac{\partial O}{\partial u_1} \approx \frac{1}{n}\sum_{\mathbf{x}\in R} I_x(\mathbf{x},t)\left(Q-\bar{Q}\right); \qquad \frac{\partial O}{\partial u_2} \approx \frac{1}{n}\sum_{\mathbf{x}\in R} I_y(\mathbf{x},t)\left(Q-\bar{Q}\right) \tag{6}$$

At each iteration we move the tracker to the neighbouring (integer) pixel location "pointed to" by the gradient $\nabla(O)$:

$$\nabla(O) = \left(\frac{\partial O}{\partial u_1}, \frac{\partial O}{\partial u_2}\right) \tag{7}$$

We terminate the iterations as soon as a move results in a decrease in the similarity measure. (The last tracker move, which led to the decrease, is also reversed.)

3.2 Comparison with other trackers

The tracker that we derived above can be seen as a gradient-based version of brute-force template matching; both approaches use normalised cross-correlation as the similarity measure. An analogous pair of tracking methods based on the sum-of-squared-differences (SSD) measure also exists: the Lucas–Kanade optical flow method [10] has its brute-force counterpart in SSD-based template matching. Both are based on the SSD similarity measure P:

$$P(\mathbf{u}) = \frac{1}{n}\sum_{\mathbf{x}\in R} (I-Q)^2 \tag{8}$$

The relationship between the four methods is shown in table 1.

In order to isolate the impact of a particular similarity measure on tracking performance, we have implemented a gradient-based SSD tracker that is analogous to our NCC technique developed in the previous section. We are therefore not testing the performance of Lucas–Kanade optical flow directly; instead, we are using the SSD similarity measure on which it is based to determine the translational (as opposed to affine) motion which the image region undergoes.

Search strategy \ Similarity measure	SSD	NCC
Brute-force	Template matching	Template matching
Gradient-based	Lucas–Kanade optical flow	Gradient-based NCC

Table 1: Classification of various trackers by similarity measure and search strategy

3.3 Dataset, metrics and experiments

We have used 21 short video sequences in order to perform a quantitative evaluation of each of the trackers described above. Fourteen of the sequences come from the CAVIAR dataset[1] and the remainder come from the PETS 2007 dataset[2]. (See the supplementary material for further details.) Ground truth data was available for the CAVIAR sequences, but it was necessary to create such data for the PETS videos using the ViPER toolkit [6, 10]. At each frame, we determine the target's scale by specifying the horizon line in the scene and exploiting the effects of perspective [3][2, pp. 57–60].

We assess a tracker's *robustness* by counting the number of sequences in which it successfully follows its target. A *lost track* is recorded whenever the overlap between the tracker's rectangle and the object's ground truth bounding box falls below 10% of the area of the latter [13].

For computational efficiency, we only evaluate the brute-force trackers on an 11×11 grid of image locations, centred on the tracker's position in the previous frame. Adjacent grid locations are separated by 3 pixels in the horizontal and 9 pixels in the vertical direction.

3.4 Results

3.4.1 Robustness

We compare the performance of our gradient-based normalised cross-correlation technique against three others: the brute-force NCC and SSD trackers (equations 1 and 8, respectively) and the gradient-based SSD tracker of the previous section. We attempt to track designated targets in each of the 21 CAVIAR and PETS video test sequences (section 3.3). For each of the four techniques, operating in the RGB colour space, we record the number of times it lost track of the object it was following (a measure of its *robustness*). The results of the tests are presented in table 2.

Search strategy \ Similarity measure	SSD	NCC
Brute-force	14	4
Gradient-based	13	3

Table 2: Count of lost tracks for each of the four trackers operating on the 21 video sequences in the dataset

[1] The datasets come from the EC Funded CAVIAR project/IST 2001 37540, found at http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

[2] The PETS 2007 datasets can be found at http://www.cvg.rdg.ac.uk/PETS2007/

It is clear that the choice of similarity measure has a large impact on tracking performance. Both of the normalised cross-correlation techniques are significantly more robust than the SSD-based trackers. (Trucco and Verri point out the close relationship between SSD and cross-correlation, but emphasise that the *normalised* version of the latter is needed to avoid biases caused by very bright or very dark image regions [18, p. 147].) For comparison, mean shift loses track on 9 of the sequences.

3.4.2 Speed

Figure 1 compares the execution time of the mean-shift technique and the gradient-based and brute-force normalised cross-correlation trackers on each of the 21 video sequences in the dataset. Our gradient-based NCC tracker is, on average, 4.8 times faster than the brute-force version, while being slightly more robust.[3] (The brute-force trackers must evaluate the similarity measure on a grid of 11×11 image locations.) The speed difference is more pronounced for the (lower-resolution) CAVIAR videos, where the gradient-based tracker generally requires fewer iterations to reach a local maximum. It is also seen that the mean-shift approach, which is much less robust than the other methods, is only 20% faster, on average, than our technique.

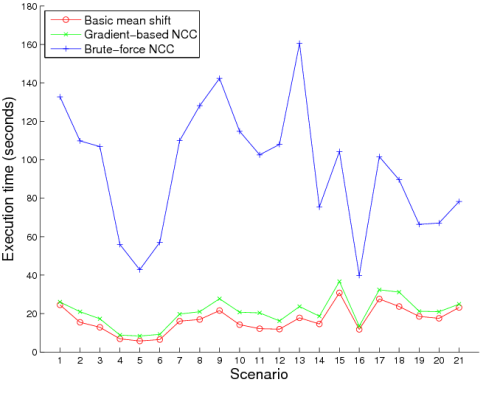

Figure 1: Execution time of the mean-shift, gradient-based NCC and brute-force NCC trackers on each of the 21 videos in the dataset

4 Track validation

The results of section 3.4.1 show that trackers based on normalised cross-correlation, whether using a brute-force or a gradient ascent search, can track most of the targets in our dataset of CAVIAR and PETS sequences. However, we only know that a target has been successfully followed by comparing the locations returned by the tracker to the ground-truth data for that target. Such data will not be available in a real-world system. This is the motivation behind the present section: it is apparent that tracking will sometimes fail; a mechanism that can detect such failures without using ground truth, which we refer to as *track validation*, would prove of great use.

[3] The experiments were performed in MATLAB on a computer with a 3.2GHz Pentium 4 CPU and 3GB of RAM.

4.1 Algorithm

We have implemented a track validation algorithm that is based on forwards–backwards trajectory mismatches. However, it differs from the work of Pan *et al.* [14] in the actions it takes when a tracking failure is detected: whereas Pan's approach is to shorten the video sequence one frame at a time until the forwards and backwards trajectories are sufficiently similar, we reduce the length of the sequence by *half* at each iteration. And in contrast to Pan, we then attempt to track the object through the remainder of the video, using a new model taken from the end of the validated portion.

A pseudocode version of our approach is given in algorithm 1. The inner loop performs the forwards–backwards validation on a section of the sequence. The outer loop attempts to find a collection of these subsequences that, when concatenated, yield a trajectory for the object in the entire video clip. Two thresholds are required by the algorithm: `min_traj_length` is the minimum number of frames which each of the subsequences must contain, and `max_traj_diff` is the the maximum allowable average per-frame distance, in pixels, between forwards and backwards trajectories that are to be considered as matching (and hence validated). In our experiments we have set the thresholds to 25 frames and 5 pixels, respectively.

```
// We have validated tracking up to pos1 in the sequence
pos1 = start;
while pos1 ≠ end do
    pos2 = end ; // Attempt to track up to pos2 in the sequence
    repeat
        track forwards from pos1 to pos2;
        reinitialise tracker at pos2; track backwards from pos2 to pos1;
        diff = average per-frame distance between fwds and bkwds trajectories;
        pos2_old = pos2;
        pos2 = (pos1 + pos2)/2;
    until (diff ≤ max_traj_diff) or (pos2_old − pos2 < min_traj_length);
    if diff > max_traj_diff then
        print('Unable to validate any further than ' + pos1); break;
    else
        print('Validated up to ' + pos2_old);
        pos1 = pos2_old;
    end
end
```

Algorithm 1: *Track validation and extension*

4.2 Results

We have compared the effectiveness of track validation when using two different underlying techniques: mean-shift tracking and gradient-based normalised cross correlation (section 3). The proportion of each of the 21 video sequences in the test set (section 3.3) that is successfully validated by our algorithm is shown in figure 2, for both of the tracker types.

It is clear from the results that gradient-based NCC significantly outperforms the mean-

shift method when used for track validation. The former technique is able to validate three-quarters or more of the track's length in 17 of the 21 sequences in the dataset, whereas mean shift only reaches this level of validation on nine occasions. Indeed, with mean shift there are a further nine videos where none of the tracker's trajectory can be validated at all – something which never happens with gradient-based NCC. Figure 3 shows examples of successful and failed track validation.

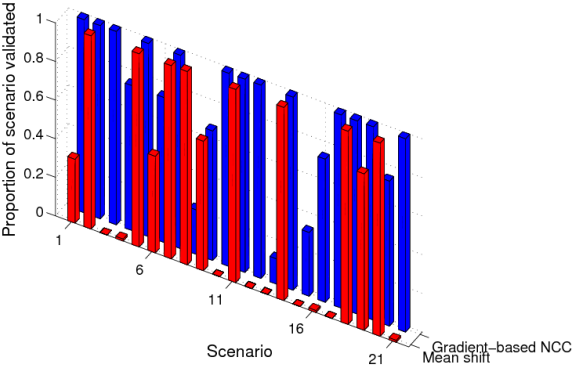

Figure 2: Proportion of each video sequence validated by our algorithm for both mean-shift and gradient-based NCC trackers.

Figure 3: (Left) An instance of successful track validation (using the gradient-based NCC tracker). (Right) An instance where tracking is not validated (using the mean-shift tracker). The forwards trajectory (red) and the backwards trajectory (green) should match very closely for validation to occur.

We attribute the disparity in track validation performance between the two methods to the underlying difference in their robustness and accuracy. At any given frame, the gradient-based NCC tracker is more likely than mean shift to be positioned accurately on its target. This is a necessary condition for track validation to occur: at the moment of reinitialisation, the tracker's location must correspond well to the location of its target. Otherwise, the tracker will be reinitialised with a model that does not accurately represent the appearance of the object it is supposed to follow – for example, the new model may show half of the person being tracked, with the other half being occupied by the background of the scene. Table 3 shows, for each video sequence, the successive models used by the track validation algorithm as it attempts to construct the longest-possible trajectory for the target. (Gradient-based NCC was used as the tracking method.) Naturally, there are different numbers of models

shown for each sequence; this is a consequence of the varying difficulty of the videos: some require more reinitialisations than others. (See the supplementary material for videos of the algorithm in operation.)

Table 3: Various models found and used by the track validation algorithm for each of the 21 scenarios.

Looking at the models in table 3 used by the track validation algorithm, it is apparent that some of them do not accurately represent the target: the tracker was badly placed at the moment of reinitialisation, but the forwards and backwards trajectories matched sufficiently well for track validation to occur. Scenario 5 demonstrates this effect, sometimes called the "template update problem": each successive model drifts further away from its target [17]. It can be seen that once such an inaccuracy has arisen, it is likely to remain present in the subsequent models. In one instance (scenario 3), the inaccuracy is compounded to the point where a portion of the track is validated using an entirely incorrect model (in this case, a model of the background). It should be borne in mind, however, that the inaccurate reinitialisations and false validations of the kinds described above occur only rarely in our video test set.

5 Discussion

Our gradient-based NCC tracker displays computational efficiency comparable to mean shift while retaining the robustness of NCC-based template matching. The track validation algorithm, which uses the gradient-based tracker, possesses a number of desirable properties. Firstly, it allows us to follow objects whose appearance changes drastically over time: table 3 contains many examples of lighting changes, articulation and out-of-plane rotation,

which make tracking a person through an entire sequence using only a single model very challenging. The approach can therefore be regarded as a form of "unsupervised model building" [12], where the model adapts to changes in the target's appearance without manual intervention. Secondly, even when our algorithm is unable to track an object for the full length of a video clip, it is providing valuable information to the higher-level processes that invoked it. Specifically, a failure of track validation indicates that tracking has become very difficult in that particular section of the video, and that other techniques and strategies should be considered. We note finally that there are ways of reducing the chance of a false validation occurring. As it stands, our algorithm only uses a certain selection of frames (shortening the sequence by half, stopping when the subsequence contains less than 25 frames) in its attempt to find forwards and backwards trajectories that match. A more exhaustive search through the sequence, or a comparison of the outcomes of runs that used different values for the thresholds, has the potential to reduce the number of false validations further.

References

[1] G.R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 2(2):12–21, 1998.

[2] D. Caulfield. *Mean-Shift Tracking for Surveillance: Evaluations and Enhancements*. PhD thesis, University of Dublin, 2011.

[3] D. Caulfield and K. Dawson-Howe. Evaluation of multi-part models for mean-shift tracking. *International Machine Vision and Image Processing Conference*, pages 77–82, Sept. 2008. doi: 10.1109/IMVIP.2008.14.

[4] D. Comaniciu and P. Meer. Robust analysis of feature spaces: Color image segmentation. In *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997. Proceedings.*, pages 750–755, 1997.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577, 2003.

[6] D. Doermann and D. Mihalcik. Tools and techniques for video performance evaluation. *Proceedings of the 15th International Conference on Pattern Recognition*, 4, 2000.

[7] G. D. Hager and P. N. Belhumeur. Real-time tracking of image regions with changes in geometry and illumination. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*, pages 403–410. IEEE Computer Society, 1996. ISBN 0-8186-7258-7.

[8] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *ICPR'10*, pages 2756–2759, 2010.

[9] J.P. Lewis. Fast template matching. In *Vision Interface*, pages 120–123, 1995.

[10] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, volume 3, page 3, 1981.

[11] V.Y. Mariano, J. Min, J.H. Park, R. Kasturi, D. Mihalcik, H. Li, D. Doermann, and T. Drayer. Performance Evaluation of Object Detection Algorithms. *International Conference on Pattern Recognition*, pages 965–969, 2002.

[12] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):810 –815, june 2004. ISSN 0162-8828. doi: 10.1109/TPAMI.2004.16.

[13] J.C. Nascimento and J.S. Marques. Performance evaluation of object detection algorithms for video surveillance. *Multimedia, IEEE Transactions on*, 8(4):761–774, Aug. 2006. ISSN 1520-9210. doi: 10.1109/TMM.2006.876287.

[14] Pan Pan, Fatih Porikli, and Dan Schonfeld. Recurrent tracking using multifold consistency. In *Proceedings of the Eleventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2009.

[15] H. Schweitzer, JW Bell, and F. Wu. Very Fast Template Matching. In *Proceedings of the 7th European Conference on Computer Vision*, volume 4, page 372. Springer-Verlag, 2002.

[16] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[17] Jian Sun, Weiwei Zhang, Xiaoou Tang, and Heung-Yeung Shum. Bidirectional tracking using trajectory segment analysis. In *Tenth IEEE International Conference on Computer Vision*, volume 1, pages 717–724 Vol. 1, Oct. 2005. doi: 10.1109/ICCV.2005.49.

[18] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998. ISBN 0132611082.

[19] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:511–518, 2001.

[20] Hao Wu, Rama Chellappa, Aswin C. Sankaranarayanan, and Shaohua Kevin Zhou. Robust visual tracking using the time-reversibility constraint. *Proceedings of the Eleventh IEEE International Conference on Computer Vision*, 2007. ISSN 1550-5499. doi: 10.1109/ICCV.2007.4408956.