Tracking Accuracy and Performance of a Basic ℓ_1-tracker

Saxon Jensen
University of Western Australia
email: 20945267@student.uwa.edu.au

May 14, 2015

Abstract

Object tracking is becoming an increasingly important topic in the field of computer vision. With the better availability of powerful computers and use of camera hardware more prevalent today, the ability to be able to track objects and extract useful information from the growing quantity of video data is very important. One such algorithm for tracking objects in a video sequence is the ℓ_1-tracker. In this paper, a basic implementation of a ℓ_1-tracker is tested on a set of video sequences, each presenting occlusions and light variation to challenge the algorithm. The performance of the algorithm is also noted, and key weaknesses of the algorithm highlighted.

Keywords: ℓ_1-tracking, ℓ_1-optimisation, tracking, video

1 Introduction

Visual object tracking aims to estimate the pose (position, size and orientation) of an object in a video sequence. Increased availability of powerful computers and more affordable video hardware has lead to an ever growing volume of video data, generating interest in more effective methods of analysis. One such way to extract information from a video sequence is object tracking. Object tracking involves detecting an object of interest, tracking it from frame to frame, and analysing the behaviour of the object. Object tracking therefore has uses in automated surveillance, detecting events of interest; object recognition, such as detecting human gait; human computer interaction.

Real life tracking scenarios are difficult for a number of reasons. Appearance of other similar objects, illumination changes and object occlusions can introduce error into the template set producing a less accurate tracker. Real time processing is often a requirement of object trackers, and so there often must be a compromise between tracking accuracy and speed.

A number of object tracking techniques have been proposed, each different in how objects are represented. Point trackers represent objects as the centroid of the object or feature points, and track an object by matching points between frames [8]. Silhouette trackers represent an object by it's silhouette, edges or some shape-based property, such as a histogram [2]. Kernel trackers [4] represent objects by templates and search for the closest matching image patches. Candidate targets are usually generated with a particle filter.

The focus of this paper is evaluating the tracking accuracy and performance of a class of kernel trackers called ℓ_1-trackers. ℓ_1-trackers represent a target object as a set of templates. A candidate target is then taken to be a linear combination of elements from the template set. An object is then the sparse representation of templates that minimises error.

Tracking accuracy of three ℓ_1-trackers will be evaluated by computing the error between the tracking result and annotated ground truth on each frame of seven video sequences. Run time is simply compared by noting the average frames per second each video sequence was run at.

In the evaluation of each algorithm, it is made apparent that while advances in run-time a made, tracking accuracy is not as significantly improved.

2 Previous and related work

A number of ℓ_1-trackers have been proposed, each improving on the original ℓ_1-tracker proposed by Mei and Ling [6].

2.1 Original ℓ_1-tracker

The ℓ_1-tracker upon which other ℓ_1-trackers is based is the tracker proposed by Mei and Ling [6].

The ℓ_1-trackers works on the assumption that a target candidate in the current frame can be reconstructed as a sparse linear combination of elements from the template set. Simply put, a target candidate is chosen as the candidate whose sparse representation minimises error.

Given a template set $T = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_2] \in \mathbb{R}^{n \times d}$ containing n templates, where each template $\boldsymbol{t}_i \in \mathbb{R}^d$ is a 1D vector constructed by stacking the template image columns. A target candidate $\boldsymbol{y} \in \mathbb{R}^d$ can then be represented as a linear combination of these templates, that is

$$\boldsymbol{y} \approx a_1\boldsymbol{t}_1 + a_2\boldsymbol{t}_2 + \cdots + a_n\boldsymbol{t}_n$$

where $\boldsymbol{a} = (a_1, a_2, \ldots, \boldsymbol{a}_n)^\top \in \mathbb{R}^n$ is the sparse target coefficient vector.

In order to allow for variances introduced by occlusions and illumination changes, some error factor must be incorporated into this linear combination. Trivial templates are included as an extension to the template set. Trivial templates are templates with only one non-zero entry. Thus

$$\boldsymbol{y} = [T, I]\begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{e} \end{bmatrix}$$

where $I \in \mathbb{R}^{2d\times d}$ is the trivial template set and $e \in \mathbb{R}^{2d}$ is the trivial coefficient vector. With $B = [T, I] \in \mathbb{R}^{d\times(n+2d)}$ and $\boldsymbol{c} = (\boldsymbol{a}, \boldsymbol{e})^\top \in \mathbb{R}^{n+2d}$ we can define the ℓ_1-minimisation problem as

$$\min_{\boldsymbol{c}} \|B\boldsymbol{c} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{c}\|_1.$$

That is, the sparse coefficient $\boldsymbol{c}$ that minimises the reconstruction error of $\boldsymbol{y}$ must be determined.

From a tracking result $\boldsymbol{y}_t$ in frame t, a particle filter is used to generate a set of target candidates for the current frame. The sparse representation of each candidate is found, and the candidate with the greatest confidence is chosen as the object in that frame.

2.2 Minimum Error bounded ℓ_1-tracker

The previous algorithm is improved upon in another algorithm proposed by Mei et al. [7]. The previous algorithm computed the ℓ_1-optimisation problem for each particle generated by the particle filter. This is computationally expensive and is the largest factor in the poor run-time of the algorithm. Additionally, the previous algorithm only added elements to the template set if they were sufficiently different from the template set. This allowed occlusions to be added to the template set, reducing the accuracy of the tracker.

It is noted [7] that the ℓ_1-optimisation problem has an upper bound, so not all target candidates generated must be tested. Let $\boldsymbol{x}_t$ be the target candidate vector and $\boldsymbol{z}_t$ be the vector representing the observation at time t. The target vector $\boldsymbol{x}_t$ can be a state vector in a particle filter framework. Then the likelihood $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ can be expressed as:

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t) = \frac{1}{\rho}\exp\left(-\alpha\|T\boldsymbol{a} - \boldsymbol{y}\|\right)$$

where α controls the shape of the Gaussian kernel and ρ is a normalisation factor. The template coefficient $\hat{\boldsymbol{c}}$ giving the lowest reconstruction error is given by

$$\hat{\boldsymbol{c}} = \arg\min_{\boldsymbol{c}} \|B\boldsymbol{c} - \boldsymbol{y}\|^2.$$

This means that $\|B\boldsymbol{c} - \boldsymbol{y}\|^2 \geq \|B\hat{\boldsymbol{c}} - \boldsymbol{y}\|^2$. Therefore, define an upper bound on $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ as

$$q(\boldsymbol{z}_t|\boldsymbol{x}_t) = \frac{1}{\rho} \exp -\alpha \|B\hat{\boldsymbol{c}} - \boldsymbol{y}\|$$

where $p(\boldsymbol{z}_t|\boldsymbol{x}_t) < q(\boldsymbol{z}_t|\boldsymbol{x}_t)$. This means that only particles with a value of $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$ within some threshold are checked.

Furthermore, this algorithm improves upon the previous algorithm by detecting occlusions and preventing their inclusion in the template set. Trivial templates are chosen when the target candidate cannot be approximated using only the template set. The trivial templates form a 2D image patch that can be mapped to each pixel in the candidate image patch, then thresholded. If a large enough proportion of the target is white (above the threshold), then the object is taken as occluded, and not included in the template set.

3 Methodology

The previous tracker improved upon the original ℓ_1-tracker Mei and Ling [6] in a few ways. Namely, the algorithm run-time was targeted by restricting the number of particles the ℓ_1-optimisation problem is solved on. Additionally, tracking accuracy is also improved upon by preventing occlusions from being included in the template set.

The Accelerated proximal gradient (APG) ℓ_1-tracker [3] improves upon the accuracy of the last tracker by modifying the ℓ_1-norm calculation. Trivial templates are included in the template library for representing occlusions, background and noise. However, since parts of the object being tracked can also be represented with the trivial templates, the previous trackers often detected regions that were not the object at all. There are two different situations that the previously defined ℓ_1-minimisation calculation doesn't allow for. The target in the next frame should be well approximated by the template set and have small trivial coefficients if there are no occlusions. If there are noticeable occlusions, the target cannot be well approximated by a sparse combination of templates, so the trivial coefficients are high.

In order to compensate for the effect of the trivial templates in the two scenarios, the ℓ_1-minimisation is reformulated as

$$\min_{\boldsymbol{c}} \frac{1}{2}\|B\boldsymbol{c}-\boldsymbol{y}\|_2^2+\lambda\|\boldsymbol{c}\|+\frac{\mu_t}{2}\|\boldsymbol{e}\|_2^2$$

where μ_t is a parameter to control the energy of the trivial templates.

The APG ℓ_1-tracker improves upon the previous ℓ_1-trackers [6, 7] by replacing the interior point method [5] used to solve the minimisation problem. The APG method is used instead. The APG solver is much faster than the interior point method, and so using this fixes a major performance bottleneck that was apparent in the previous trackers.

4 Implementation

The three ℓ_1-trackers are implemented in MATLAB and use the SPAMS [1] package for the minimisation calculations. The performance of the three ℓ_1-trackers are evaluated on several publicly available videos, each displaying different challenges to the tracking algorithms.

α controlling the shape of the Gaussian kernel and regularisation parameter λ are set to 40 and 0.01 respectively. For the APG ℓ_1-tracker, the μ_t parameter is set to 10.

In the first two ℓ_1-trackers proposed by Mei and Ling [6], Mei et al. [7], the tracking error was simply the Euclidean distance between centre points of the ground truths and target. As seen in Figure 1, this matrix does not penalise a tracking result for being a difference size to the ground truth. Bao et al. [3] normalises the Euclidean distance between the two centre points by the size of the target from the ground truth. Whilst it is not stated exactly how this is done, simply taking the size of the target from the ground truth could lead to division by 0 in a perfect tracking result.

From the problems that exist in the previous error measurements, the following error definition is used. Let $\boldsymbol{x}_t$ be the target state vector for frame t and let $\bar{\boldsymbol{x}}_t$ be the ground truth. $A(\cdot)$ is a function returning the area of the bounding box of a state vector. Centroid$(\cdot)$ gives the vector representing the centroid of a state. The error, ϵ of a state vector $\boldsymbol{x}$ is given by

$$\epsilon=\frac{\|\text{Centroid}(\bar{\boldsymbol{x}}_t)-\text{Centroid}(\boldsymbol{x}_t)\|}{A(\boldsymbol{x}_t)+A(\bar{\boldsymbol{x}}_t)+|A(\boldsymbol{x}_t)-A(\bar{\boldsymbol{x}}_t)|}$$

5 Results

Table 2 demonstrates the accuracy of the trackers. Each algorithm was run on a series of seven video sequences, each demonstrating different challenges

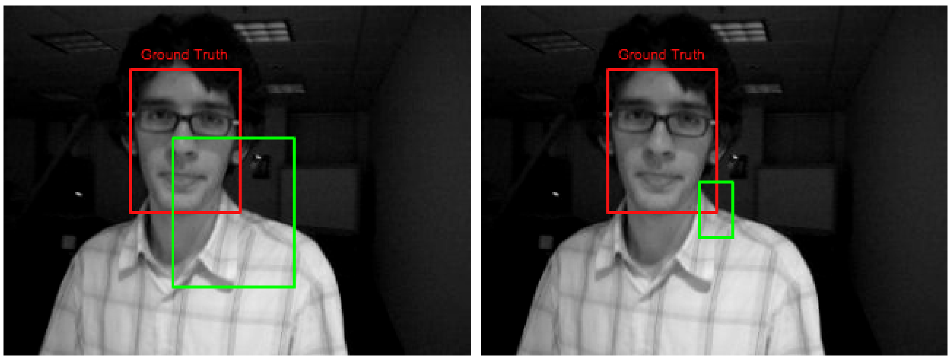

Figure 1: Both sets of boxes would have the same error using purely Euclidean distance. With the proposed error measure, 1a is rightly labeled to have smaller error.

to the tracking methods, such as large illumination changes and significant changes of pose. The error of the tracking result from the ground truth for each frame is graphed.

The original ℓ_1-tracker does not incorporate any mechanism for preventing occlusions from being added to the template set. This may account for the very large fluctuations in error. The APG tracker and bounded ℓ_1-tracker are mostly the same, the key differences being the faster ℓ_1-optimisation method used and a reformulated optimisation problem. This is seen in a generally lower degree of error for the APG tracker.

The ℓ_1-trackers varied only slightly in their accuracy. As expected, the APG ℓ_1-tracker generally performed better.

The *car4* video sequence presents no significant occlusions for the trackers to handle. At frame 150 of this sequence, the car being tracked passes into a shadow, changing the illumination on the object. The original ℓ_1-tracker can be seen to become less accurate after this point. The APG tracker appears to handle this illumination change significantly better than the other trackers.

The *david indoors* video sequence displays changes in pose of the object being tracked, in this case a persons head. All ℓ_1-trackers seemed to fail to keep track of the target through these pose changes, as can be seen by the spikes at 150 - 170 frames, when the object changes pose dramatically in a short space of time.

The *faceocc2* sequence is the first video sequence that presents partial occlusions to the trackers. Whilst the first occlusion at 150 frames does not significantly effect the trackers, a rapid pose change and movement at 350 frames can be seen as a spike in error across all trackers. The APG

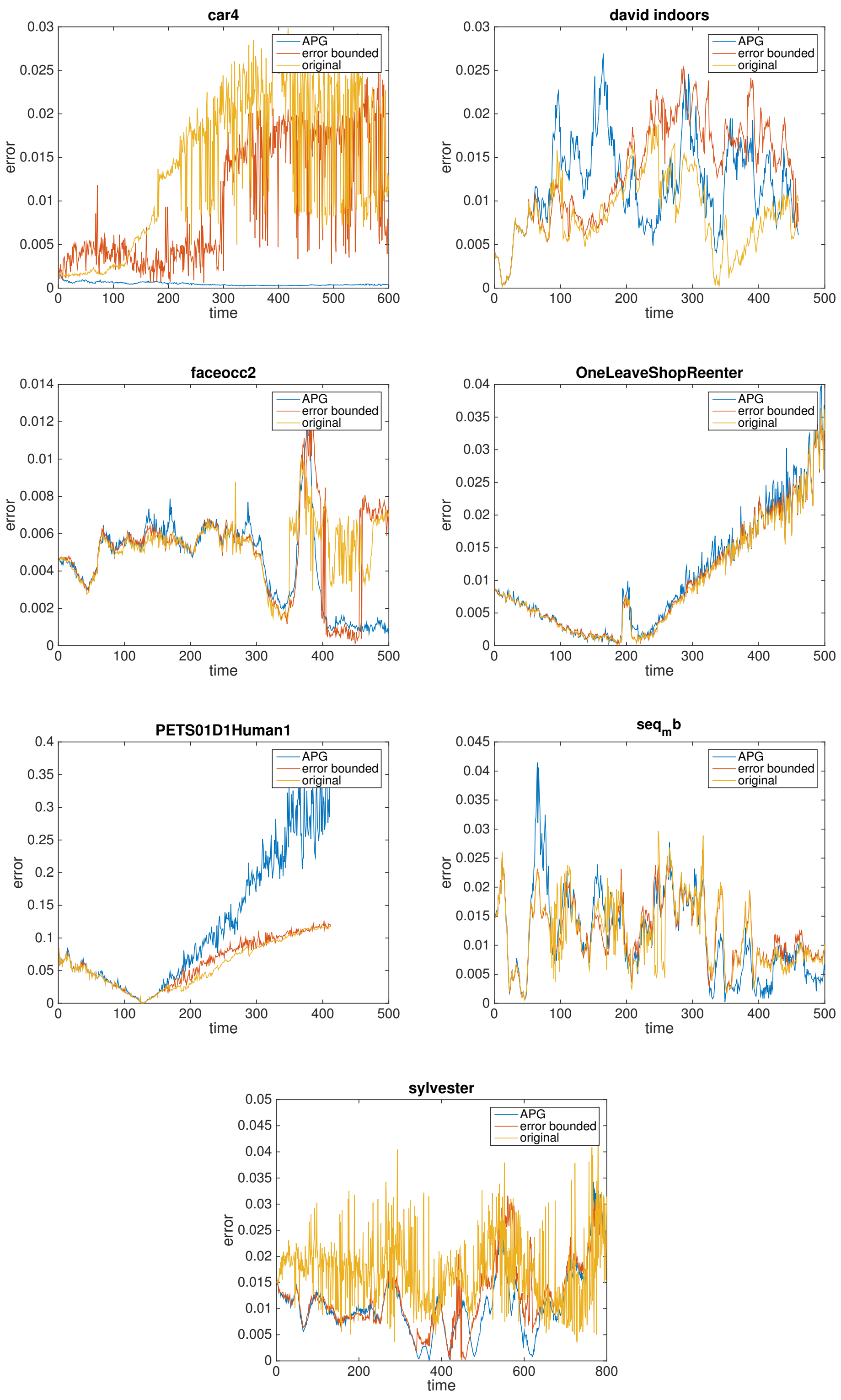

Figure 2: The error over the duration of each video sequence for the three tested ℓ_1-trackers.

tracker however recovers best afterwards, while the original tracker continues to perform badly after this, possibly due to the inclusion of a bad template into the template set.

The *OneLeaveShopReenter2cor* video sequence shows the object changing size gradually over time, shrinking into the distance. At 200 frames, there is a spike in the error, corresponding directly to the instance where the object is occluded.

The *PETS01DHuman1* is an interesting example of the trackers failing to follow the target. At 100 frames, the object being tracked passes behind a post, occluding the target totally. After this point, the tracker is unable to accurately follow the target. This could be due to a contamination of the template set, that is, the template set was updated with an occluded object. Since the occlusion is of a similar colour to the dominant background of the scene, this could lead to the background being chosen more often as the best candidate. It is also noted the the APG method did the worst in this test, despite including the occlusion detection mechanism. This indicates that while the optimisation calculation can be adapted to compensate for noise and occlusion, the method used to update the template library is critical to the trackers accuracy.

Finally, the *Sylvester* video and *seq mb* sequences show the APG tracker performing the best, and the original tracker preforming the worst, as expected. The *Sylvester* sequence incorporated a number of rapid and significant pose changes under a very directional light, meaning the template library must be often updated. Since occlusions do not happen in this sequence, the occlusion detection incorporated into the bounded ℓ_1-tracker does not much improve tracking accuracy. The APG ℓ_1-tracker performs the best possibly due to it's dynamic adjustment of the energy of the trivial templates.

As Figure 2 shows, with the exception of two of the video sequences, the three ℓ_1-trackers performed very similarly in terms of tracking accuracy. However, the most significant advances have been made in the run-time of the tracking algorithms. As seen in Table 1, the performance of the APG tracker is significantly better, while the performance of the error-bounded ℓ_1-tracker is only slightly better than the original ℓ_1-tracker, indicating that the ℓ_1-minimisation calculation is the most expensive element of the algorithm.

6 Conclusion

The three ℓ_1-trackers tested show a steady improvement of the ℓ_1-tracking framework. The original ℓ_1-tracker [6] uses a simple template updating scheme without occlusion detection, and so occlusions are included in the

-	APG	Bounded	Original
car4	10.32	0.80	0.68
david indoors	7.29	0.38	0.41
face occlusion	8.46	0.79	0.69
leaving shop	11.28	0.78	0.57
PETS01DHuman1	13.91	0.48	0.74
seq mb	9.65	0.73	0.5
sylvester	9.94	0.67	0.57

Table 1: Average fps counts.

template set, reducing its accuracy. Furthermore, the ℓ_1-optimisation calculation is performed on every particle generated by the particle filter framework, this being a major performance bottleneck. Indeed, performing consistently slower than 1 fps means that the original ℓ_1-tracker cannot possibly perform in real-time, as is often required in real applications.

The minimum error bounded ℓ_1-tracker [7] improved upon the original ℓ_1-tracker in two ways. An occlusion detection mechanism is included to prevent occlusions contaminating the template set, producing more accurate tracking results as demonstrated. Additionally, the bound on the error of target candidates means that the expensive ℓ_1-optimisation calculation is not performed on every particle. This allowed for a consistently higher, but still rather poor frame-rate.

Finally, the APG ℓ_1-tracker Bao et al. [3] replaced the ℓ_1-optimisation procedure used in the previous trackers with the much faster APG approach, producing a much higher frame-rate on all the video sequences tested. Furthermore, the factors included in the optimisation calculation to allow for intensity of trivial templates produced slightly more accurate tracking results.

Whilst important improvements were made to the ℓ_1-tracking framework, the tests indicate that there are still problems. All of the tested trackers used a similar template updating strategy, assigning weights to each template and replacing the lowest weighted template with a newer higher weighted one. What was often observed however, was a very small number of the templates were often updated to reflect changes in the object being tracked, leaving the remaining templates out of date and no longer capturing the object. With this in mind, the template update procedure would be the next element of the ℓ_1-tracking framework to improve upon.

References

[1] URL `http://spams-devel.gforge.inria.fr/downloads.html`.

[2] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 798–805, June 2006. doi: 10.1109/CVPR.2006.256.

[3] C. Bao, Y. Wu, H. Ling, and H. Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1830–1837, June 2012. doi: 10.1109/CVPR.2012.6247881.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5): 564–577, May 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1195991.

[5] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):606–617, Dec 2007. ISSN 1932-4553. doi: 10.1109/JSTSP.2007.910971.

[6] X. Mei and H. Ling. Robust visual tracking using ℓ_1-minimization. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1436–1443, Sept 2009. doi: 10.1109/ICCV.2009.5459292.

[7] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai. Minimum error bounded efficient ℓ_1 tracker with occlusion detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1257–1264, June 2011. doi: 10.1109/CVPR.2011.5995421.

[8] C. Veenman, M. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(1):54–72, Jan 2001. ISSN 0162-8828. doi: 10.1109/34.899946.