Object Tracking using the ℓ_1-tracker

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

07/04/2015

Abstract

Object tracking is becoming increasingly important in the field of computer vision. With the increased availability of powerful computers and the lowered cost of video equipment, the ability to automatically track objects of interest in video is even more important. Previously, object tracking algorithms were unable to track objects through occlusions and through changes of appearance, such as lighting. More recently, a group of object trackers known as ℓ_1-trackers have been proposed. While research into object representations has been conducted, relatively little work has been done in learning to track an object through various changes of pose. In this paper, the evolution of the ℓ_1-tracker is described, and the methods weakest points analyzed.

Keywords: tracking, computer vision

1 Introduction

Visual object tracking aims to estimate the pose (size and position) of a target in a video sequence. Increased availability of powerful computers and affordability of video cameras, combined with the increased need for automated video analysis due to increased use in surveillance and media has generated interest in object tracking. Object tracking involves detecting a target of interest, tracking it from frame to frame, and analyzing the behaviour of the target. This means that object tracking has many uses such as:

- object recognition, such as detecting human gait;

- automated surveillance, such as detecting events of interest;
- human computer interaction.

Visual object tracking is challenging for a number of reasons:

- object occlusions;
- complex or articulated shapes/objects;
- scene illumination changes;
- noise in the image; and
- real time processing requirements.

A number of object tracking techniques have been proposed, each different in how objects are represented. Point trackers represent objects as points, such as the centroid of the object. The tracker then tracks an object by matching points between frames. Silhouette trackers represent an object by it's silhouette or edges. Kernel trackers represent objects by templates and search for the closest matching image patches using methods such as the particle filter.

In this paper, we focus on kernel tracking, in particular, a class of kernel trackers called ℓ_1-trackers. ℓ_1-trackers represent a target object by a set of templates. In this tracking method, target candidate is assumed to be a linear combination of templates. An object state is predicted to be where the sparse representation of the object minimizes error.

From the first ℓ_1-tracker proposed by Mei and Ling [4], a number of improvements have been made targeting object representation and optimization, however the process of building and updating the template set has largely been left unaddressed. In this paper, the evolution of the ℓ_1-tracker is analyzed, and the importance of the template update problem made clear.

2 Previous Work

2.1 The First ℓ_1-tracker

2.1.1 ℓ_1-minimization

The first ℓ_1-tracker from which most other ℓ_1-trackers are based is the tracker first proposed by Mei and Ling [4]. The general ℓ_1-tracker can be described like so:

Given a target template set $T = [\boldsymbol{t}_1 \cdots \boldsymbol{t}_n] \in \mathbb{R}^{d \times n}$ where $d > n$ containing n target templates with each template $\boldsymbol{t}_i \in \mathbb{R}^d$. Templates $\boldsymbol{t}_i$ are vectors formed by stacking the template image columns into a 1D vector. The ℓ_1-tracker works on the premise of representing a tracking result $\boldsymbol{y} \in \mathbb{R}^d$ as a linear combination of templates from the template set as

$$\boldsymbol{y} \approx T\boldsymbol{a} = a_1\boldsymbol{t}_1 + \cdots + a_n\boldsymbol{t}_n$$

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

In order to incorporate some error due to occlusion and noise, the we can define $\boldsymbol{y}$ as

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

where I^+ and $I-$ are the positive and negative trivial template sets respectively (trivial templates have only one non-zero entry). e^+ and e^- are the co-efficient vectors for the positive and negative trivial template sets respectively.

With $B = [T, I^+, I^-] \in \mathbb{R}^{d \times (n+2d)}$, we can define the ℓ_1-minimization problem central to the algorithm as

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

That is, we find the sparse co-efficient matrix $\boldsymbol{c}$ that minimizes the error between $\boldsymbol{y}$ and the template set.

2.1.2 Template Building and Updating

An object is tracked through video by extracting templates from the first frame, and finding the object of interest in successive frames. However, in longer video sequences, an object of interest may change pose and go through different illumination changes. Thus, it is important that the template set be updated in some way. Template updating is a complex problem; if the template set is updated too often, small errors are quickly introduced and accumulate. If the template set is not updated, the tracker cannot adapt to changes in pose and illumination.

The tracker proposed by Mei and Ling [4] uses an intuitive update method. A weight $w_i = \|\boldsymbol{t}_i\|$ is assigned to each template $\boldsymbol{t}_i$. Intuitively, the weight corresponds to how *important* a template is. The greater the norm of $\boldsymbol{t}_i$, the smaller the corresponding co-efficient needed in the ℓ_1-minimization solution.

The template update process can be summarized simply as such. If a tracking result $\boldsymbol{y}$ is not similar to the current template set T, the least important template is replaced. Templates that are closely matched have their weight increased, meaning they are more important.

2.1.3 Shortcomings

This initial ℓ_1-tracker has some shortcomings. As was noted, trackers often have real time processing requirements. Since the optimization calculation in this algorithm must be solved many times for every frame, the algorithm cannot run in real time. This problem is addressed in later ℓ_1-trackers. Furthermore, whilst the template update method is intuitive, it does not make use of temporal information inherit in a video sequence.

2.2 Minimum Error Bounded ℓ_1-tracker

The ℓ_1-tracker previously proposed by Mei and Ling [4] had a number of shortcomings. In updating the template set, target candidates sufficiently different from the existing templates replaced templates with the lowest weight. However, this intuitive update strategy allows occlusions to contaminate the template set. Additionally, the algorithm solved the ℓ_1-optimization problem a number of times for each frame of the video sequence. This high computational load means that the algorithm is unable to perform at reasonable speeds. Both of these issues are addressed in an improved ℓ_1-tracker proposed by Mei et al. [5].

2.2.1 Run-time Improvements

It is noted that the ℓ_1-minimization problem has an upper bound, thus not all target candidates generated by the particle filter need be checked.

We define $p(\boldsymbol{x}_t|\boldsymbol{z}_t)$ as the confidence in a target candidate $\boldsymbol{x}_t$ taken from frame $\boldsymbol{z}_t$ where

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

where α controls the shape of the Gaussian kernel and ρ is a normalization factor.

The template co-efficient $\hat{a}$ giving the lowest reconstruction error is given by

$$\hat{a} = \arg\min_{b} \|T\boldsymbol{b} - \boldsymbol{y}\|^2$$

where $\boldsymbol{b} = [\boldsymbol{a}, \boldsymbol{e}^+, \boldsymbol{e}^-]^\top$. This means $\|T\boldsymbol{a} - \boldsymbol{y}\|^2 \geq \|T\boldsymbol{a} - \boldsymbol{y}\|^2$ therefore, we can define an upper bound on $p(\boldsymbol{x}_t|\boldsymbol{z}_t)$ as

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

where $p(\boldsymbol{x}_t|\boldsymbol{z}_t) \leq q(\boldsymbol{x}_t|\boldsymbol{z}_t)$

With this information, the previous algorithm is modified as such. A value of $q(\boldsymbol{x}_t|\boldsymbol{z}_t)$ is calculated for each target candidate $\boldsymbol{y}$. Then, only particles with a value of $q(\boldsymbol{x}_t|\boldsymbol{z}_t)$ within some threshold have their ℓ_1-minimization calculated.

2.2.2 Template Updating

In the previous tracker, patches significantly different from the template set replaced the least important template in the set. While this intuitive strategy allows the algorithm to adapt to changes in object appearance, it also allows occlusions to contaminate the template set, resulting in the tracker *drifting*.

Trivial templates are chosen when the pixel intensity cannot be approximated well using the target templates. The trivial templates form a 2D trivial image patch, which is then mapped to each pixel in the candidate image patch and thresholded. White pixels in the thresholded image represent occlusions, if a large enough proportion of the image is white, the object is taken as being occluded, and frames are not added to the template set.

2.3 The Real Time ℓ_1-tracker

The first ℓ_1-tracker proposed had a number of shortcomings, the most significant of which it processed only 1-2 frames per second [4]. The tracker proposed by Mei et al. [5] offered a number of improvements, reducing the number of particles that are processed by first calculating a bound on the ℓ_1-minimization problem. Whilst this offered a reported speed up of four to five times [5], it is still far slower than being a real time tracker. The tracker proposed by Bao et al. [1] improves upon the previous ℓ_1-trackers by offering real time tracking performance.

It is noted that the even the ℓ_1-tracker using the bounded minimization requires an ℓ_1-minimization problem be solved many times for every frame. Therefore, performance improvements can be made in reducing the cost of the minimization.

The previously proposed trackers use an interior point method [3] to solve the ℓ_1-norm minimization problem. This method is very slow. The accelerated proximal gradient method (APG) [7] is a much faster optimization method.

The ℓ_1-minimization problem solved in the previous algorithms is restructured as an unconstrained minimization problem. This algorithm is largely the same as the previous algorithm proposed by Mei et al. [5], with the interior point method for solving the ℓ_1-norm minimization replaced with the faster APG method.

2.3.1 Shortcomings

While the issue of real time performance is addressed specifically in this tracker, the problem of adapting the template set to changing object illumination and pose is left largely unaddressed.

2.4 Multi-Task Tracker

One aspect not yet investigated in detail in the previous trackers is the particle filter. The previous trackers solve the sparse coding problem for each particle independently.

Particles clustered around a target state are not independent however, and image patches generated by a particle filter are related in that they share a region in the frame. A ℓ_1-tracker proposed by Zhang et al. [9] makes use of the relationship between particles by utilizing a multi-task sparse learning.

2.4.1 Multi-Task Learning

The Multi-Task Learning (MTL) tracker makes use of the inherit relationships between particles to improve tracking performance and accuracy. The ℓ_1-minimization problem is adapted as follows.

Particle image patches $\boldsymbol{x}_i \in \mathbb{R}^d$ are represented in a matrix $X \in \mathbb{R}^{d \times n}$ with

$$X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$$

where there are n particles.

Similar to the previous algorithms, the template set T and the trivial templates I such that

$$X = [T, I] \begin{bmatrix} Z \\ E \end{bmatrix}$$

where Z is the sparse template coefficient matrix and E is the trivial template coefficient matrix. That is, the particles are a linear combination of the templates. The difference from the previous ℓ_1-trackers is that the minimization problem to solve becomes

$$\min_C \|X - BC\|_F^2 + \lambda \|C\|_{p,q}.$$

By finding a coefficient matrix C that most closely represents the particles represented by X in matrix form, the inherit similarities between particles are exploited. Furthermore, less ℓ_1-minimization problems need to be solved, since particles are looked at jointly, improving the performance of the algorithm.

2.4.2 Shortcomings

Whilst tracking performance and accuracy is improved by learning parameters by jointly solving between particles, the template set is still updated in the same intuitive manner described in [5]. Template updating methods still remain the least investigated aspect of the ℓ_1-tracking framework.

2.5 Low-rank Sparse Learning Tracker

The Low-rank Sparse Learning Tracker (LRST) proposed by Zhang et al. [8] is very similar to the tracker proposed in [9].

As in the previous algorithm, the particle samples are in the current frame are represented as linear combinations Z of object and trivial templates that make up the dictionary T. Particles are sampled around the previous object state, and thus particle samples are expected to have similar representations, so will be of low-rank.

This tracker is different in that the sparse representation of the particle sample is found by solving

$$\min_{Z,E} \lambda_1 \|Z\|_* + \lambda_2 \|Z\|_{1,1} + \lambda_3 \|E\|_{1,1}$$

such that $X = TZ + E$ with X defined as before. $\|Z\|_*$ is the trace norm of Z, and is the best method of minimizing the rank of Z, since minimizing the rank of Z directly is not computationally tractable. By representing the function to be minimized different, the Inexact Augmented Lagrange Multiplier method is used to compute a sparse representation of a particle sample. For the details of the optimization method used in this algorithm we refer the reader to [8] due to space constraints.

2.5.1 Shortcomings

The proposed LRST is reported to have a per-frame run time of 340 seconds compared to 3 for the ℓ_1-tracker. This tracker offers performance improvements over the tracker proposed in [9] by improving the efficiency of the method used to determine the sparse representation. However, yet again,

while the major performance bottleneck is targeted, the method used to update the template set remains the same as that proposed in the original ℓ_1-tracker.

2.6 Adaptive Sparse Appearance Model

The algorithm proposed by Jia et al. [2] is the first ℓ_1-tracker to change the method used to update the template set. Previously, the template set is updated by weights assigned to each template and the similarity between the template and estimation of the target candidate. While the past trackers used sets of static templates, the algorithm proposed by Jia et al. [2] uses incremental subspace learning and sparse coding to update templates adaptively.

2.6.1 Template Update

The proposed template update method has two parts. A cumulative probability sequence

$$L = \{0, \frac{1}{2^{n-1}-1}, \frac{3}{2^{n-1}-1}, \ldots, 1\}$$

is generated, where n is the size of the template set. A uniform random number between 0 and 1 is taken to select the template to update. Due to the probability sequence, newer templates are more likely to be updated than older templates.

The estimated target can be modelled as a linear combination of the Principal Component Analysis (PCA) basis vectors and trivial templates in

$$\boldsymbol{p} = [U, I] \begin{bmatrix} q \\ e \end{bmatrix}$$

where $\boldsymbol{p}$ is the observation vector, U is the matrix of eigen base vectors, I is the trivial templates and q is the coefficient vector. A sparse representation is found by solving

$$\min_{c} \|\boldsymbol{p} - H\boldsymbol{c}\|_2^2 + \lambda\|\boldsymbol{c}\|_1$$

where $H = [U, I]$, $c = [q, e]^\top$ and λ is a regularization parameter. The PCA algorithm proposed by Ross et al. [6] for subspace learning is used to obtain the eigen basis matrix.

2.6.2 Shortcomings

The algorithm proposed by Jia et al. [2] uses dynamic templates based on the eigen basis of the template set. Whilst this tracker is the first to focus on

the template update method, it loses many of the performance improvements proposed in the previous trackers. In particular, this tracker is reported to run at 1.5 frames per second, whilst the original ℓ_1-tracker ran at 3 frames per second and the tracker proposed by Zhang et al. [9] able to run at 340 frames per second. Whilst state of the art tracking accuracy is reported, it comes at the cost of run-time performance.

However, it is shown by this tracker that incorporating spacial information into the template update process yields better results. It then follows, that incorporating other information into the template update method will yield better tracking accuracy still.

3 Conclusion

In recent times, there has been a lot of work in the area of visual object tracking, spurred on by the increased availability of computers and video equipment. In particular, the ℓ_1-tracking framework first proposed by Mei and Ling [4] has seen a lot of research.

Often a requirement of an object tracker is to track a target in real-time. The most significant shortcoming of the ℓ_1-tracker first proposed was that it could only run at a very low frame-rate. This was in part addressed in an updated tracker by reducing the number of particles to perform the ℓ_1-minimization calculation on.

The ℓ_1-tracker was for the first time able to be run in real-time by the algorithm proposed by Bao et al. [1]. In this tracker, the method used to solve the ℓ_1-minimization was targeted, and adapted to make use of the much faster APG solver.

At this point, while the speed of the ℓ_1-tracker has received attention, it's tracking accuracy has largely been left ignored. The use of a particle filter has remained constant in the ℓ_1-tracking framework, and this aspect of the framework receives scrutiny in the tracker proposed by Zhang et al. [9], who make use of the spacial relationship between particles to jointly solve the ℓ_1-minimization between particles, thus improving tracking accuracy and improving the performance of the tracker.

Of all the trackers based on the ℓ_1-tracking framework, only the algorithm proposed by Jia et al. [2] improves upon the template update method by updating newer templates more frequently than older templates. Even this very simple intuitive template update method offers improvements in tracking accuracy over the original ℓ_1-tracker. This shows the template update aspect of the ℓ_1-tracking framework to be an important aspect, and one that has been largely ignored.

References

[1] 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.

[2] X. Jia, H. Lu, and M.-H. Yang. Visual tracking via adaptive structural local sparse appearance model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1822–1829. IEEE, 2012.

[3] 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.

[4] 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.

[5] 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.

[6] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008. ISSN 0920-5691. doi: 10.1007/s11263-007-0075-7. URL `http://dx.doi.org/10.1007/s11263-007-0075-7`.

[7] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2008.

[8] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Low-rank sparse learning for robust visual tracking. In *Computer Vision–ECCV 2012*, pages 470–484. Springer, 2012.

[9] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via multi-task sparse learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2042–2049. IEEE, 2012.