Augmenting the L1 Tracker with appearance-based tracking improvements

Charles Bradshaw
Department of Image Processing
University of Cape Town
Cape Town, South Africa
Email: brdcha003@myuct.ac.za

Fred Nicolls
Department of Image Processing
University of Cape Town
Cape Town, South Africa
Email: fred.nicolls@uct.ac.za

Gerhard de Jager
Department of Image Processing
University of Cape Town
Cape Town, South Africa
Email: Gerhard.DeJager@uct.ac.za

***Abstract*—The L1 Tracker is an existing algorithm for adaptive tracking by detection which uses a particle filter and tracks by attempting to express proposed matches as sparse linear sums of known templates. The sparse representation of the object gives information of the current appearance of the target which, for certain objects where appearance is linked to anticipated motion, can help performance on subsequent frames. Currently the L1 Tracker neither collects nor uses this information. The L1 Tracker also only maintains 10 templates, and has a naïve approach to updating the list when a new template is added. This paper describes the application of the L1 tracker on a particular sequence where the object motion is strongly linked to appearance, and the adjustments made to improve performance. The adjustments include: 1) learning the motion associated with each template, and using it in the particle propagation; 2) improved template selection for the 10 templates used for each frame; 3) using backtracking to improve performance when the object disappears and reappears and 4) including negative templates to create a discriminative tracker.**

I. Introduction

Visual object tracking is a common task in image processing. The objective is to follow a certain target through a series of frames. This has many applications, such as surveillance, human-computer-interfacing, visual servoing and traffic monitoring. Sometimes the target is known in advance, and an a-priori model can be built describing it. Often there is no a-priori knowledge, and all that is available is an initial frame annotated with the location of the object.

An approach that has achieved good results recently is tracking by detection. In tracking by detection, the tracker only models the 2-D appearances of the targets, discarding any concept of a 3-D object in a 3-D world. One algorithm that uses this approach is the L1 Tracker.

The L1 Tracker uses a particle filter to model the probability density function (PDF) of the target location. It evaluates each particle by attempting to express it as a linear sum of templates. The better a particle is modelled by the templates, the more likely the state is to represent the target. Ignoring the obvious problems with taking weighted sums of different views of the target, this approach has shown some success. However, the implementation produced by the authors only maintains 10 templates. Every time a new template is recorded, an old one needs to be removed. This is potentially useful information that is being discarded. There is also useful information from motion observed for each template that is discarded. The authors' implementation also does not handle the target leaving the scene. The L1 tracker also does not model the background; modelling the background is an effective strategy. We address these issues and demonstrate improved tracking performance, specifically on a sequence where object appearance is strongly linked to object motion in the frame.

The rest of this paper proceeds as follows: in Section II we review the relevant literature. In Section III we discuss the L1 tracker in depth. In Section IV we motivate and explain our improvements. In Section V we describe our experiments and present our results, and in Section VI we make conclusions.

II. Related literature

Tracking-by-detection trackers can be grouped into generative trackers and discriminative trackers. In this section we will briefly discuss several trackers in each group.

A. Generative Trackers

A generative tracker builds a model of the target's appearance and then tries to find the location in each frame that is most similar to the model. Examples include the fragments-based tracker [1], visual tracking decomposition [4], and the L1 tracker [5].

The fragments-based tracker [1] takes the initial target image and divides it into several sub-patches, or fragments. It then takes pixel intensity histograms of these fragments. The object model is the combination of each fragment's position relative to the target frame, and the histogram expected for that fragment. When a new frame is presented, each fragment compares its expected histogram to the observed histogram at each possible location, and votes for appropriate target locations. All the fragments' votes are aggregated and the tracker outputs the best combined target location. The visual tracking decomposition tracker [4] separates the model into a observation model and a motion model. It then builds multiple trackers with varying object and motion models. It runs these multiple trackers in parallel, allowing them to influence each other. The L1 tracker will be discussed in more detail in the next section.

B. Discriminative Trackers

Discriminative trackers model the foreground and the background, and then classify patches of each presented frame

into these classes. The patch that the tracker has the most confidence in is classified as foreground, and is selected as the output. Examples of discriminative trackers include the multiple instance learning tracker [3] and the ensemble tracker [2].

The multiple instance learning tracker [3] draws a selection of patches from each frame. Those far from the tracking result are labelled negative, while those near the tracker result are all put into a bag marked positive. The model assumes one of the patches in each positive bag is the target but does not decide which. This reduces model drift. The ensemble classifier [2] uses an ensemble of weak classifiers to build a strong classifier that discriminates foreground pixels versus background pixels.

III. The L1 tracker

This section describes the L1 tracker [5] as obtained from www.dabi.temple.edu/∼hbling/code_data.htm. First the tracking algorithm is explained, then we discuss aspects of the algorithm and implementation that are relevant to the changes we have made to the tracker.

A. Overview of algorithm

The L1 tracker uses a particle filter framework to model the PDF of a Bayesian posterior probability: $p(x_t|y_{1:t})$ where x_t is the state at frame t as represented by an affine transformation, and $y_{1:t}$ are the observations from frames 1 to t represented as raw pixel values for the resized image patches. At each time instant t a prediction is made:

$$p(x_t|y_{1:t-1}) = \int p(x_{t-1}|y_{1:t-1}) \times p(x_t|x_{t-1}). \qquad (1)$$

Here the first factor is the Bayesian prior and the second is the motion model. The prediction is calculated as an equally-weighted discretised set of states s_t by transforming S_{t-1} (the previous frame's discretised posterior probability, used as the current frame's prior probability) randomly according to a zero-mean Gaussian distribution. The assumption is that because states far from S_{t-1} are unlikely to be drawn from the distribution they will be poorly represented in s_t. This is evidently equivalent to a low $p(x_t|y_{1:t-1})$, and as such the weights are not stored (s_t is assumed to be equally weighted).

Next the algorithm crops the image patch associated with each of the proposed states, and calculates an observation model $p(y_t|x_t)$. It attempts to express each patch as a linear sum of the 10 stored templates. The better the match is, the higher the observation probability will be. It does this by augmenting the 10 templates with a set of trivial templates, where the n^{th} trivial template is non-zero only at pixel n. Any noise in the image can therefore be explained by adding weight to the corresponding trivial template. The tracker then expresses each particle as a sparse sum of the templates and the trivial templates through L1-minimisation using an accelerated proximal gradient descent.

The updated posterior probability is

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t) \times p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}. \qquad (2)$$

The denominator is independent of the state, and since we are only interested in maximising this posterior as a function of state, we ignore it. The state with the maximum posterior is output as the tracker result for that frame. If the image patch associated with this state passes certain criteria, it is added to the set of 10 templates, and the template that contributed the least to the current frame is deleted. Finally s_t is resampled according to Eq. 2, the updated posterior, to form a set of equally-weighted states S_t for use in the next frame.

For its initial template set, the L1 tracker crops 10 near-identical subimages from the initial frame.

B. Comments

Modelling the motion (i.e. the prediction probability) implicitly through particle density (but with each particle having the same weight) is counterintuitive. It means that in the optimisation stage a particle that was far away from the mean of the Gaussian (i.e. the motion model predicts that it is possible but very unlikely) will outperform one that is near the mean (i.e. likely) but with a slightly lower observation model value. This is undesirable, especially considering the small computational effort required to update the weights to include the motion model value explicitly.

The L1 paper [5] refers to a state transition model (in Section 3.1), where the prediction step takes the target velocity into account. There was little evidence of this in the code. This is pertinent because our motion model is based on object appearance, not on recent object movement.

IV. Proposed Improvements

In this section we describe and justify our improvements on the L1 tracker. Our improvements fall into 4 main categories: 1) particle guiding, 2) template selection, 3) backtracking and 4) negative templates. Particle guiding and template selection both stem from our observation that the appearance of the target in a frame carries information that is useful in tracking during subsequent frames. As such we add a 7^{th} state variable to each particle: the template most used in its sparse representation. We call the space described by the first 6 state variables (i.e. those describing the affine transformation) the state space, and the space described by the 7^{th} state variable the template space. In particle guiding we look at how the template-space state variable can be used to help guide the particles in state space. In template selection we guide the particles through template space, by controlling the 10 templates in the template set used for sparse representation. In backtracking, we address issues relating to the target leaving the frame and returning. In negative templates, we adapt the L1 tracker to be a discriminative tracker.

We now address each of the improvement categories in detail.

A. Particle Guiding

At the end of a pair of consecutive successfully tracked frames, the following information is available:

1) The template that best described the first frame: the template that contributed the most in the weighted sum used to match the image patch.

2) The motion between the two states: the difference in state-space between the two outputs.

This is useful information. We can learn what state-space motion to expect for each template, and cluster the particles more densely in that location in state space. This will mean that the search resolution is usually highest where it is most needed. We can do this prediction in a particle-by-particle manner: each particle moves according to its template-space state variable. This is in line with the particle filter being able to handle multi-modal data.

In testing we found that having the tracker learn all 6 affine states' transitions led to dramatic changes in aspect ratio, and so we set the tracker to learn only the 2 state transitions corresponding to translation.

It is useful to note that this motion prediction is based on appearance and hence is independent to the usual velocity-based movement model.

B. Template Selection

The L1 tracker maintains a small pool of 10 templates. When a tracked result meets the criteria for inclusion into the set of templates, one of the old templates needs to be discarded. The original L1 tracker discards the template which contributed the least to the current frame. We propose that a better approach would be to discard the template with the least unique information. The L1 tracker already calculates the inter-template correlation for use in the accelerated proximal descent, so at no extra computational cost we can discard the template with the highest mean correlation to the other templates.

It is also intuitive that discarding information should detract from performance. If there is enough processing power and memory, then keeping a larger pool of templates from which to select 10 for each frame should improve performance. In a similar manner to which we plan to learn what state space movement is associated with each template, we will learn which templates tend to follow each other. In this way we model the template space to describe the distance between each pair of templates. On each tracking frame, the new set of 10 templates is selected by considering which templates are highly represented among the particles. Each template m contributes the n closest templates to itself, where n is the the floor of 10 times the fraction of particles whose template-space state variable is m. The remaining templates (whose space has been created by the floor operation) are selected randomly from the large set of templates.

C. Backtracking

Our third modification is to improve the tracker's handling of the target leaving the scene. The original code did not provide any mechanism for handling this. We implemented a simple threshold: if the winning particle's sparse representation has a combined weight for the trivial templates above a predefined threshold, then the frame is considered empty. On an empty frame, no track result is returned and all templates are free to continue propagating (i.e. $S_t = s_t$).

We found that, while the target was out of frame, the tracker could get caught on background objects and ignore the target when it reappeared. To address this, when we rediscover the target (i.e. a match after an empty frame), we make a recursive call to our algorithm, backwards in time, initialised on the rediscovered target, until the last tracked frame (we did not set a maximum number of frames for this recursive call, although in practice this would not be feasible). This backtrack can have three possible outcomes:

1) Backtrack loses track before last known frame: in this case it appears that the target left and returned. The backtrack is successful. Allow it as a positive track.

2) Backtrack tracks to the last known frame, but final frame does not overlap with the target in the last known frame: in this case we have found a background object. The backtrack was unsuccessful. Do not allow this to be a positive track. If the tracker was a discriminative tracker, this would be included as a negative template.

3) Backtrack tracks to last frame, and final frame does overlap with the target in the last known frame: in this case we have experienced several frames of false negatives. The backtrack was successful. Allow it as a positive track. As an aside, these could be a good positive templates. We did not use these to update the template set.

By backtracking whenever we rediscover the target and reacting appropriately, we should be able to reduce false positives during empty frames.

In practice we found that the particle filter's multi-modal PDF description could lead to problems. We observed cases where the tracker correctly rejected background objects via backtracking. However, when the object returned and was correctly identified, the particles still near the background object managed to overpower the target several frames later. The backtracker was not triggered, as the previous frame was not empty. In response to this we culled the particles on target rediscovery, setting them all to the track result. This zeros the PDF everywhere other than at the target location.

D. Negative Templates

Due to repeated problems of background objects being being awarded high observation model values, we decided to implement a discriminative version of the L1 tracker. We trained a negative tracker with negative templates, cropped from around the target in the first frame. Whenever good candidates were found for negative templates (e.g. after an unsuccessful backtrack) they replaced the negative template with the least unique informations.

During tracking, if the winning particle was better described by the negative templates than the positive templates, it was discarded.

V. Results

In this section we describe the testing procedure. Then we look at the effect of our different improvements. Because our improvements affect each other's performance, and there are too many permutations to look meaningfully at them all, we look at the each adjustment's effect on the final tracker. For each adjustment tested, we first describe the trackers, then we present qualitative evidence that the adjustments are

Fig. 1. Composite made up of image regions from selected frames of the jet ski sequence

improving results. Finally, we present quantitative results for all the trackers tested.

A. Testing Procedure

We tested the trackers on a jetski sequence consisting of 1154 frames. In the sequence a jetski zigzags its way towards the camera, in a dock environment, leaving the scene several times. Figure 1 shows an overview of the jetski's motion throughout the sequence. As ground truth we used a successful track result verified by a human. This leads to a slight inaccuracy of results, but since most tracking failures consisted of latching onto background objects, we felt this was a reasonable approach. We assigned each frame a label (True Positive, True Negative, False Positive, Missed Positive, False Negative), where a True Positive had a bounding-box overlap with the ground truth of at least 50%. We then used the well-known F-score to quantify the tracker's performance:

$$F\text{-}score = \frac{2PR}{P+R} \tag{3}$$

where

$$P = \frac{True\ Positives}{True\ Positives + False\ Positives + Missed\ Positives} \tag{4}$$

$$R = \frac{True\ Positives}{True\ Positives + False\ Negatives + Missed\ Positives}. \tag{5}$$

Because the L1 tracker is a probabilistic algorithm we averaged over 50 test results.

B. Tracker performance

1) Initial Tracker: As a baseline, we ran the L1 tracker with no improvements on the sequence. this tracked well until the target's first disappearance. Because the tracker had no mechanism for handling object disappearance, it continued to track background objects, and did not latch onto the target when it reappeared. Figure 2a shows the tracker succeeding while the object is in frame, Figure 2b shows the tracker failing when the target disappears, and Figure 2c shows the tracker failing to rediscover the target on reappearance.

Figure 3 shows the progress of the tests through the sequence, with each column representing a frame. If the target

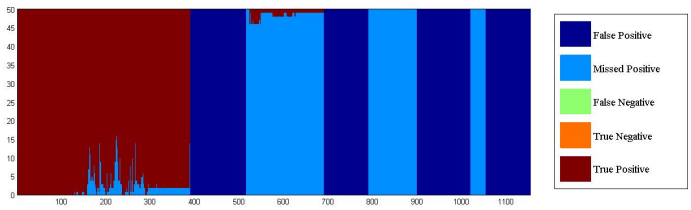


Fig. 3. L1 Tracker test state versus time showing the fraction of tests in each state for each frame

is in frame during that time instance, the column will be part red, part light blue and part green. The red part represents the fraction of tests in which the tracker correctly identified the target for the frame (i.e. a true positive). The light blue part represents tests in which the tracker's output was not deemed correct for the frame (i.e. a missed positive). The green represents tests in which the tracker falsely identified the frame as empty (i.e. a false negative). Periods for which the target was out of frame are part orange, part dark blue. The orange corresponds to tests where the tracker correctly identified the frame as empty (i.e. a true negative). The dark blue corresponds to tests where the tracker gave an output for an empty frame (i.e. a false positive). As can be seen, the tracker performed well up to the first disappearance (i.e. the first block of red,light blue and green is mostly red). In no test cases did the tracker identify that the object had left the scene, and when it returned very few tests rediscovered it (i.e. the rest of the plot is either light or dark blue, depending on whether the target was in frame or not).

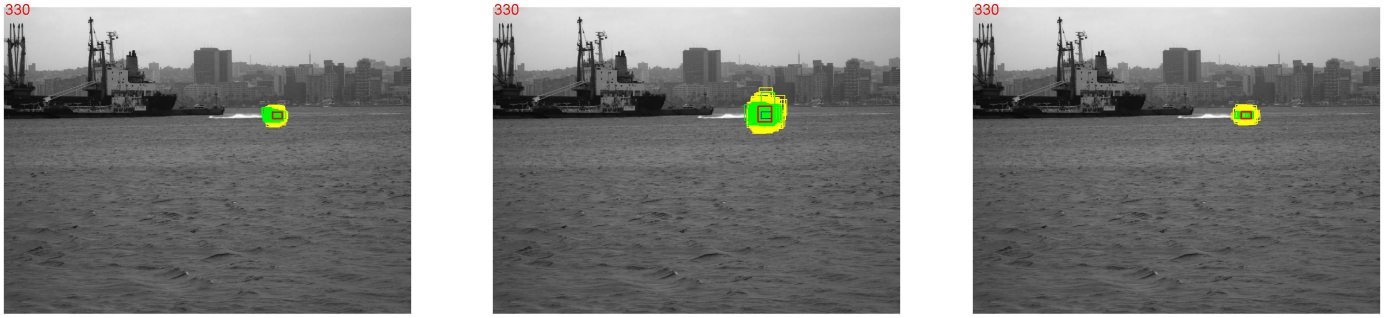
2) Final Tracker adjusting Particle Guiding: To test the effectiveness of particle guiding, we enabled the full template selection (i.e. maintaining a large pool of templates and selecting the 10 most useful for each time instance), full backtracking (i.e. backtrack on target rediscovery, and zero the PDF on target location after a successful rediscovery) and negative templates. We then tested three cases of particle guiding: no guiding; full particle guiding (i.e. all 6 state-space variables are guided), and modified guiding (i.e. only the two state-space variables corresponding to translation are guided).

Figure 4 shows a sample frame that characterises the differences in tracking for the different particle guiding variations. In Figure 4c, we see the modified guiding tracker's result is in the centre of the particle cloud, whereas in Figure 4a we see the no guiding tracker's result is on the leading edge of the cloud. This is because the modified guiding tracker predicted the location better, and was able to concentrate the particles so that the greatest density of particles (i.e. a better search resolution) was around the target's location. It is pertinent to point out that this prediction is made not on the target's motion over previous frames, but according to the appearance on the previous frame. In Figure 4b we see that the full-guiding tracker's response is very different to the ground truth. The tracker was too confident on the aspect ratio change during the turn. When the target completed the turn, the particles continued their aspect ratio change, which led to inferior tracking.

3) Final Tracker adjusting Template Selection: To test the effectiveness of template selection, we enabled the modified particle guiding (i.e. learning the two translation state vari-

(a) Frame 250 (b) Frame 470 (c) Frame 600

Fig. 2. The L1 tracker at frames 250 (2a), 470 (2b) and 600 (2c). The rectangles represent all the particles. Yellow rectangles represent particles that were in s_t but did not survive into S_t. Green rectangles represent particles that survived to S_t. The red rectangle is the tracker's output, and the blue rectangle is the ground truth.



(a) No guiding (b) Full guiding (c) Modified guiding

Fig. 4. The final tracker adjusting particle guiding for frame 330 with no particle guiding (4a), full particle guiding (4b) and modified particle guiding (4c). The rectangles represent all the particles. Yellow rectangles represent particles that were in s_t but did not survive into S_t. Green rectangles represent particles that survived to S_t. The Red rectangle is the trackers output, and the blue rectangle is the ground truth.

ables for each template), the full backtracking and negative templates. We then tested three cases of template selection: naïve template selection (i.e. when a template needs to be discarded, the template which contributed the least to the current frame will be discarded); improved template selection (i.e. when a template needs to be discarded, the template which contributes the least unique information will be discarded), and full template selection (i.e. maintaining a large pool of templates and selecting the 10 most useful for each time instant).

Figure 5 shows the different performance per frame for the three test cases. The occurrences of false negatives (as shown by green sections) in the naïve template selection tracker after each reappearance are noticeably larger than in the improved and full template selection tracker. It is evident that for each frame the improved template selection tracker has more correct test cases (as shown by more red or orange per column) than the naïve template selection tracker. The full template selection tracker in turn outperforms the improved template selection tracker in every frame.

4) Final Tracker adjusting Backtracking: To test the effectiveness of backtracking, we enabled the modified particle guiding, full template selection, and negative templates. We

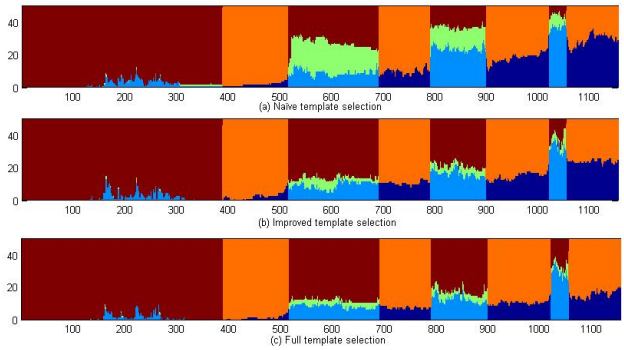


Fig. 5. Final tracker adjusting template selection test state versus time showing the fraction of tests in each state for each frame (same colour scheme as Figure 3). The results for a naïve template selection ar shown in 5a, 5b shows the results for the improved template selection and 5c shows the results for the full template selection.

then tested three cases of backtracking: a simple threshold; a threshold with backtracking (i.e. the PDF is not focused on the target if backtracking is successful), and full backtracking (if a backtrack is successful then the particles are all set to the

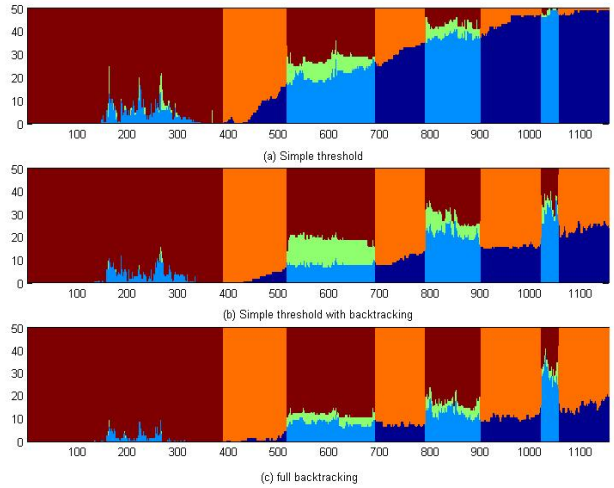


Fig. 6. Final tracker adjusting backtracking test state versus time showing the fraction of tests in each state for each frame (same colour scheme as Figure 3). The results for a simple threshold are shown in 6a, 6b shows the results for a simple threshold with backtracking, and 6c shows results full backtracking (i.e. with PDF zeroing after rediscovery of target).

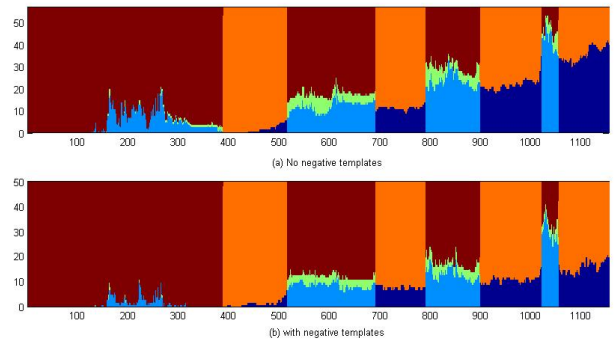


Fig. 7. Final tracker adjusting negative templates test state versus time showing the fraction of tests in each state for each frame (same colour scheme as Figure 3). The results without negative templates are shown in 7a and 7b shows the results with negative templates.

target location, zeroing the PDF elsewhere).

Figure 6 shows the different performance per frame for the three test cases. The backtracking only takes effect on particle rediscovery, so it is not surprising that the most noticeable difference is a reduced gradient during the empty frames: fewer true negatives become false positives (as shown by the orange and dark blue rectangles, which have fewer orange pixels becoming dark blue pixels).

5) Final Tracker adjusting negative templates: To test the effectiveness of negative templates, we enabled the modified particle guiding, full template selection, and full backtracking. We then tested two cases: with negative templates, and without.

Figure 7 shows the different performance per frame for the two test cases. It is evident that for most frames there were more instances of correct tracking for the tracker with negative templates.

TABLE I. AVERAGE F-SCORE ACROSS 50 TRIALS

Tracker	F-score
L1 tracker	0.4860
Final tracker with no particle guiding	0.8205
Final tracker with full particle guiding	0.7165
Final tracker with modified particle guiding	0.8485
Final tracker with naïve template selection	0.7178
Final tracker with improved template selection	0.8104
Final tracker with full template selection	0.8485
Final tracker with a simple threshold	0.6129
Final tracker with a threshold and simple backtracking	0.7717
Final tracker with full backtracking	0.8485
Final tracker with no negative templates	0.7597
Final tracker with negative templates	0.8485
Final tracker	0.8485

C. Summary of results

Table I shows the average F-score for all the trackers tested. The results have been grouped according to the adjustment being tested, with the last member of each group being the final tracker. The L1's performance is noticeably lower than the other trackers tested. This is largely because it has no mechanism for handling the target leaving the scene. The particle guiding shows a small improvement for the modified particle guiding, and a deterioration for full particle guiding, whereas the improvements for template selection, backtracking and negative templates are all substantial.

VI. CONCLUSION

In this paper we modified the L1 tracker to test the effectiveness of 4 different strategies: 1) particle guiding, 2) template selection, 3) backtracking and 4) negative templates. The first two strategies were in response to our observation that the target appearance is another state parameter to consider while tracking. The third was to improve false positive rejection while the target was out of frame, and the fourth was to convert the L1 tracker into a discriminative tracker. We tested on a sequence where target appearance was strongly linked to target motion and which included the target leaving the scene and returning. We found that the particle guiding led to small improvements, and the template selection, backtracking and negative templates all led to large improvements on the L1 tracker.

ACKNOWLEDGMENTS

We thank the authors of the L1 Tracker [5], for releasing their source code. We also thank the National Research Foundation of South Africa for their funding of this project.

REFERENCES

[1] A. Adam, E. Rivlin and I. Shimshoni, "Robust Fragments-based Tracking using the Integral Histogram", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06), pp.798–805, 2006.

[2] S. Avidan, "Ensemble Tracking", Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, pp. 494-501, 2005.

[3] B. Babenko, M. H. Yang, & S. Belongie, "Visual tracking with online multiple instance learning", IEEE conference on Computer Vision and Pattern Recognition pp. 983–990, 2009.

[4] J. Kwon, K. M. Lee, "Visual tracking decomposition", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10); pp. 1269-1276. 2010

[5] X. Mei, H. Ling, "Robust Visual Tracking and Vehicle Classification via Sparse Representation", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 33(11):2259–2272, 2011.