An enhanced adaptive coupled-layer LGTracker ++

Jingjing Xiao
School of Electronics, Electrical and Computer Engineering, University of Birmingham, B15 2TT, UK
shine636363@sina.com

Rustam Stolkin
School of Mechanical Engineering, University of Birmingham, B15 2TT, UK
r.stolkin@bham.ac.uk

Aleš Leonardis
School of Computer Science, University of Birmingham, B15 2TT, UK
ales.leonardis@fri.uni-lj.si

Abstract

This paper addresses the problems of tracking targets which undergo rapid and significant appearance changes. Our starting point is a successful, state-of-the-art tracker based on an adaptive coupled-layer visual model [10]. In this paper, we identify four important cases when the original tracker often fails: significant scale changes, environment clutter, and failures due to occlusion and rapid disordered movement. We suggest four new enhancements to solve these problems: we adapt the scale of the patches in addition to adapting the bounding box; marginal patch distributions are used to solve patch drifting in environment clutter; a memory is added and used to assist recovery from occlusion; situations where the tracker may lose the target are automatically detected, and a particle filter is substituted for the Kalman filter to help recover the target. We have evaluated the enhanced tracker on a publicly available dataset of 16 challenging video sequences, using a test toolkit [17]. We demonstrate the advantages of the enhanced tracker over the original tracker, as well as several other state-of-the art trackers from the literature.

1. Introduction

There is a rapidly growing demand for robust automatic tracking from video sequences, due to the proliferation of ubiquitous, low-cost, small-scale camera sensors [18]. However, visual tracking still poses many open challenges: background clutter, occlusion, fast movement, illumination changes, object scale change and deformation, etc. [1]. Over more than 30 years of visual tracking research, numerous approaches have been proposed, most utilizing two primary components: visual features of the object and spatiotemporal prediction [20].

Many different cues can be used for target representation, and making appropriate choices is a non-trivial problem. Popular cues include color, texture, silhouette, motion, e.g. [5]. More detailed representation schemes can be found in [19]. A good tracker should detect and combine target information from multiple features. A global representation reflects the overall statistical characteristics of the target in a simple and computationally efficient model, e.g. [12, 13, 16]. However, global models can fail if the target rapidly changes its structure or appearance. In contrast, a local feature representation typically utilizes interest points or a set of patches to encode target information [6, 7]. It can provide greater robustness for deforming targets, but usually with increased computional cost. For spatiotemporal prediction, deterministic methods, e.g. Mean Shift [9], are fast but susceptible to local minima. In contrast, stochastic methods, e.g. Particle Filter [11], are more robust to local minima but sacrifice computational efficiency. It is increasingly clear that one cannot achieve robust tracking by any single feature or motion model. More recently, approaches integrating multiple schemes have been proposed [2, 7, 10].

In most real applications, the appearance of a target will change, due to deformation, viewpoint, illumination changes and other causes. Therefore, for robust tracking, it is also critical to be able to update the target model adaptively with appearance changes. Martínez-del-Rincón [8] utilized a Rao-Blackwellised particle filter which updated a color target model and target position estimate. However, allowing a single holistic target model to be continuously relearned or updated is inherently unstable, since background features can be erroneously learned into the whole target model. Yang et al [3] built part-based feature sets for tracklets, to learn discriminative appearance models online. However, this algorithm used a fixed number and geometry of patches. Without flexible patches, this method is limited, e.g. it can track upright walking pedestrians, but not humans performing gymnastic movements with large deformations. Kwon and Lee [7] proposed an approach to automatically update the topology of local pose changes. Their method utilized a star model in which all of the parts were connected to the centre of the target. By analyzing the likelihood function landscape of local mode of these patches, this method can move, delete and add patches without any training phase in initialization. While this approach provides a good mechanism for adapting the visual model in a controlled manner, rapid part removal can lead to false structural changes in the geometrical model and result in tracking failure. These problems can be mitigated by having a two-layer model, in which each layer provides robustifying constraints when updating the other layer.

An early attempt at two-layer adaptive models was Stolkin et al. [15]. In this approach, motion and geometric models for a rigid target object were used to impose structure for continuously re-learning an intensity model for target pixels. More recently, Cehovin et al., [10], proposed a coupled-layer visual model that combined a set of parts (local layer) together with global target appearance (global layer). The local layer is a set of small patches that geometrically constrain updates to the target's global appearance models. The global layer models the target's global visual properties of color, motion and shape, and is, in turn, used to help update the local patches. Because both layers interact and provide constraints for each other, it enhances the robustness while adaptively relearning targets which undergo rapid and significant appearance changes. This coupled-layer visual model is considered a sophisticated state-of-the-art technique, and performs well against other methods in comparative evaluations. We use this method as a starting point, since it provides a good mechanism for adaptively updating a visual target model. However, through extensive testing, we have identified four important situations in which this method can fail. In this paper, we describe several extensions and enhancements to the tracker, which substantially alleviate these problems.

In section 2, we introduce the coupled-layer tracking algorithm and identify its vulnerable failure modes in section 3. In section 4, we propose four modifications to the original tracker which enhance its performance in these situations. Section 5 presents the results of testing with challenging public data sets. Section 6 provides a discussion and conclusions. Throughout the paper, we discuss tracking performance with reference to 16 video sequences, drawn from the ICCV 2013 VOT workshop public dataset and evaluation toolkit, [17].

2. A coupled-layer tracking algorithm

The coupled-layer visual model is shown in Fig.1. The local layer is a geometrical constellation of visual parts that describe the target's geometrical properties. At each new image frame, these patches are propagated according to a Kalman Filtered motion model, and a cross-entropy method is used to optimize their locations. When the target's appearance changes, some patches are unable to find highly matching candidate regions and these are gradually removed from the model (representing disappearing target parts). Meanwhile, new patches are selected from regions which are identified (by the global layer) as having a high probability of being the target. Thus, the global layer is a probabilistic region model, encoding the fundamental target characteristics of colour, motion and shape. Once, new local layer patches have been selected, the local layer is used to constrain the adaptive updating of the global layer.

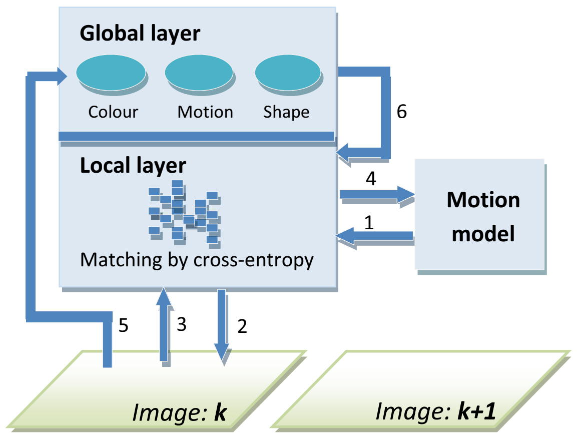


Figure1: A schematic overview of the main steps in the processing of a single frame. 1-spatiotemporal prediction, 2-match the local layer, 3- update patches, 4- update the motion model, 5- update the global layer, 6- add new patches

The global layer provides a probability map for allocating new patches, computed by [10]:

$$p(L|C_k, M_k, F_k) \propto p(L|C_k)p(L|M_k)p(L|F_k) \quad (1)$$

Where C_k, M_k, F_k represent color, motion and shape properties, given a location L in frame k.

The local layer consists of a set of patches, $S_k = \{s_k^{(i)}\}_{i=1:N_k}$, which are modeled as local distributions of pixel measurements, Z_k, with associated weights:

$$\hat{w}_k^{(i)} = p\left(Z_k|s_k^{(i)}\right)p\left(s_k^{(i)}|S_k\right) \quad (2)$$

where $p\left(Z_k|s_k^{(i)}\right)$ is "visual consistency" - a measure of consistency between a patch and a candidate image region, and $p\left(s_k^{(i)}|S_k\right)$ is "drift distance" - a measure of how far a patch has drifted from the target centroid (modeled as a sigmoid function) .

3. Four major causes of failure

By analyzing individual instances of tracker failure, we have identified four common causes of failure in the original adaptive coupled-layer tracker:

- **Significant scale changes**: Once a patch has been initialized, its scale remains fixed. Although the overall bounding box (global layer) adapts to changing target scales, the fixed patch size cannot adequately model targets whose size changes dramatically. For example, when a target becomes very small (Fig. 2), two or three patches can cover the whole target while the remaining patches are forced to occupy nearby background regions. This leads to low precision and tracking failure.

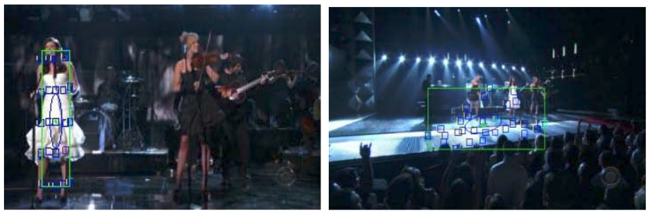

(a) Initialization (b) nearly lost target

Figure 2: Fixed patch scale for a rapidly shrinking target(**singer**)

- **Environment clutter**: Distracting background areas can cause one or more patches to move far away from the main group of patches. The global shape boundary then stretches to include these distant patches, causing a large background region to be mislabeled as target, from which new patches are erroneously sampled. This can lead to instability and tracking failure (Fig.3).

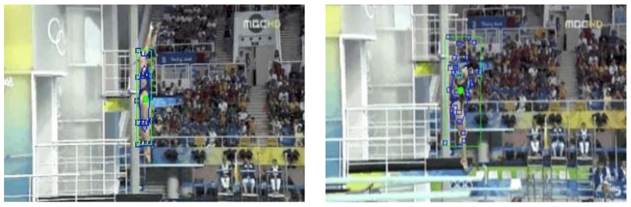

(a) Initialization (b) Patch drifting

Figure 3: Patch drifts from majority due to clutter (**diving**)

- **Occlusion**: The original algorithm did not address the occlusion problem (Fig.4). When part of a target is occluded, patches try to move to unoccluded parts and can shrink to occupy a very small region. Patches without good image matches will be removed. As the bounding box shrinks, the tracker loses the target.

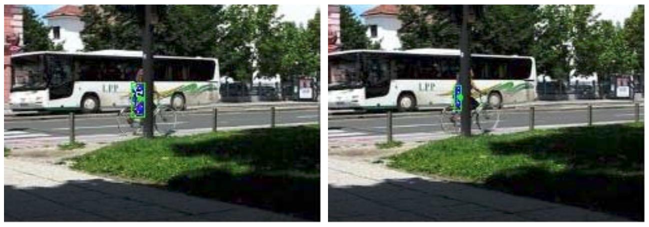

(a) Occulsion ocurs (b) Tracking failure

Figure 4: Full occlusion (**bicycle**)

- **Rapid movement**: Target speed and direction can change rapidly (Fig.5). Even for stationary targets, camera motion or zoom can cause large target motions in image sequences. The original tracker used a Kalman Filter which cannot handle the nonlinear and non-Gaussian problems which this situation often causes. In the *Woman* sequence, the target velocity is around 20 pixels in the X direction in frames 560-565. In frame 566, its speed changes suddenly to -20 pixels. This rapid irregular movement can easily result in tracking failure if Kalman Filtering is used to make a spatiotemporal prediction.

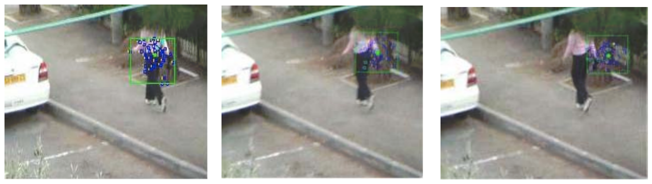

(a) Frame 565 (b) Frame 566 (c) Frame 567

Figure 5: Rapid irregular target movement (**woman**)

4. Four new enhancements

The main contribution of this paper is to add four enhancements to the original coupled-layer tracker, in order to solve the problems highlighted in section 3. Solutions for the patch scale and drifting problems in clutter, decreasing the risk of tracking failure while improving precision. Additionally, we describe a way of detecting occlusion situations, and then recalling a memory of previously seen target features to enable tracking recovery. Furthermore, during periods of low tracking confidence, the Kalman-filter is augmented by a stochastic particle-like approach, to help recover the target.

4.1 Adapting patch scales to the target

The original algorithm uses the spread of local patches to adapt the size of an overall bounding box. However, the size of each patch itself remains fixed. As shown in Fig.2, the tracker can fail when the target shrinks to sizes of similar magnitude to that of individual patches – because some patches are forced off the target and onto background pixels. Therefore, it is essential to enable the scale of patches to adapt.

A naive approach is to scale patches proportionate to size changes in the overall bounding box. However, the bounding box may shrink due to occlusion, and not due to true target scaling, in which case patches should maintain their current size and refrain from shrinking. Additionally, rapid and noisy size changes lead to instability. Allowing patch scale to rapidly respond to noise in the bounding box size engenders rapid, erroneous addition or removal of information which can irrevocably damage the adaptive appearance model. Therefore, an important issue here is how to ensure the stability while adapting the scale.

Local patches are initialized as a proportion, σ, of the bounding box size:

$$A_o^p = A_o^b/\sigma \tag{3}$$

where A_o^p and A_o^b are the areas of the bounding box and each patch, respectively. Scale factor σ cannot be used to change patch scale directly, because noisy variation in bounding box sixe, A_o^b, would cause instability. Therefore, we compute an n-frame moving average bounding box scale:

$$\bar{A}_k^b = \sum_{i=k-n}^{k} A_i^b \tag{4}$$

where A_k^b is the scale of bounding box at the frame k. This moving average scale can be memorized as: $M^b = \bar{A}_k^b$.

During tracking, the moving average $\bar{A}_k^b$ is compared against the last memorized value, M^b, at each frame. If the scale has changed significantly (above the threshold T_A), then the size of patches is adjusted accordingly:

$$A_k^p = \begin{cases} A_{k-1}^p, & \left|\bar{A}_k^b - M^b\right| < T_A \\ \lambda_A A_{k-1}^p + (1-\lambda_A)\bar{A}_k^b/\sigma, & otherwise \end{cases} \quad (5)$$

where λ_d is a persistence factor to control the speed of size adaptation, eliminating sudden, large information changes. Whenever patches are re-scaled, we also need to update the memorized bounding box:

$$M^b = \lambda_A M^b + (1-\lambda_A)\bar{A}_k^b \quad (6)$$

Note that T_A cannot be a fixed threshold, because the significance of any size change must be judged relative to the current absolute size of the bounding box. Therefore, threshold T_d is itself adapted with recent box size:

$$T_A = kM^b \quad (7)$$

We find a value of k = 0.02 to be effective. Following each scale adaptation, the next adaptation is delayed by a minimum time interval to ensure that scale is not changed too frequently.

4.2 Preventing patch drifting in clutter

In cluttered environments, individual patches can easily drift away from the main patch cluster. The original algorithm utilized the median of Euclidean distances between an individual patch and all other patches to evaluate whether or not the patch is drifting. This geometric constraint can adequately prevent a single patch from drifting (Fig.6.a). However, if two or more patches drift at the same time, then the median becomes distorted and can fail (Fig.6.b and Fig.6.c).

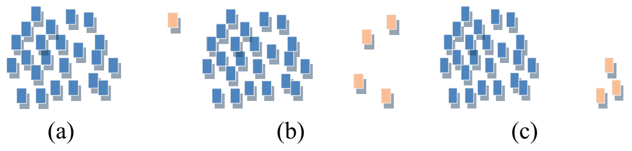


Figure 6: majority (blue) and drifting (orange) patches.

As drifting means small groups of patches distributing differently from the majority patch distribution, one should both consider the number and the density of drifting patches. Here, we employ the marginal distribution to solve this problem. The joint spatial distribution density of patches can be denoted as:

$$\sum\sum_{(x,y)\in I}\rho(x,y) = 1 \quad (8)$$

Where (x,y) represents patch coordinates in image I. We obtain the marginal distribution of patches by:

$$\rho_y(x) = \sum_y \rho(x,y) \text{ and } \rho_x(y) = \sum_x \rho(x,y) \quad (9)$$

Drifting must be checked for in both *x* and *y* directions. Here we illustrate for the *x* direction:

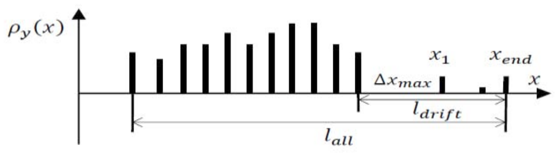


Figure 7: Marginal distribution in *x* direction

In order to identify potential drifting regions, we first search the maximum non-zero interval Δx_{max} in $\rho_y(x)$. The region outside of this interval is regarded as a candidate potential drifting region. Next, the following two criteria are applied to each such candidate region, to confirm whether or not it is really drifting. The weights of confirmed drifting patches are then decreased.

A. search maximum non-zero interval

Drifting causes a large gap in the distribution of patch positions. This can be detected easily in the patches marginal distributions. The algorithm first searches the maximum non-zero interval Δx_{max}, Δy_{max} for each marginal distribution respectively, and drifting is detected when either of these intervals exceed thresholds $T_{\Delta x}$, $T_{\Delta y}$. Noticeably, $T_{\Delta x}$ and $T_{\Delta y}$ are not fixed parameters, but change adaptively according to the scale of the local patch l_{patch} and current distribution length l_{all}. For example, in the X-direction (Fig.7):

$$T_{\Delta x} = \begin{cases} \mu_1\, l_{patch}, & \mu_{all} l_{all} < \mu_1\, l_{patch} \\ \mu_{all} l_{all}, & \mu_1\, l_{patch} \le \mu_{all} l_{all} \le \mu_2 l_{patch} \\ \mu_2\, l_{patch}, & \mu_2 l_{patch} < \mu_{all} l_{all} \end{cases} \quad (10)$$

Where μ_{all} and μ_1, μ_2 are factors which must be chosen by the user according to the tracking situation. We have found that values of $0.2 < \mu_{all} \le 0.4$ and $2 < \mu_1, \mu_2 \le 5$ work well in many tracking situations. These factors define the extent of drifting which is permitted, before being detected and corrected as an error situation. Once Δx_{max} exceeds $T_{\Delta x}$, the region outside the maximum non-zero interval becomes regarded as a potential drifting region and is then chosen as a candidate for additional drift checking according to the following method.

B. detecting drifting from the patch density

Drifting is not only characterised by a small group of patches, which are separated from the majority cluster. Additionally, the distribution density of the drifting patches should be very small. The proportion r of patches in the drifting region can be computed by:

$$r = \sum_{x=x_1}^{x_{end}} \rho_y(x) \quad (11)$$

Where x_1 and x_{end} are coordinates shown in Fig.7, and $\rho_y(x)$ is a normalised distribution which sums to unity.

The user must set a value V_{r1} first (good values lie in the range 0, 40%) to define "***minor***" patch drifting.

A second measure is patch density in the candidate drifting region from the length ratio V_{r2}, denoted by:

$$V_{r2} = l_{drift}/l_{all} \tag{12}$$

where l_{drift} and l_{all} are lengths shown in the Fig.7. If r is smaller than V_{r2}, it indicates that the patch density in the candidate drifting region is less than the average patch density. Considering both constrains, drifting can be automatically detected by satisfaction of the inequality:

$$r < T_V \tag{13}$$

where threshold $T_V = min(V_{r1}, V_{r2})$ (14)

C. automatic recovery from patch drifting

Once a drifting patch has been detected, its weight is re-computed as:

$$p_x\left(s_{d_{(i)}}\middle|S\right) = \begin{cases} e^{-(l_{drift}-l_{d_{(i)}})}, i \in R \\ 1, otherwise \end{cases} \tag{15}$$

where subscript i is used to denote the ith patch in the drifting region, R, and $l_{d_{(i)}}$is the distance between patch i and the closest boundary patch (located in x_{end}) of whole distributed region. l_{drift} is shown in Fig.7.

Taking into account both X-axis and Y-axis marginal distributions and also the sigmoid function constraint from the original tracker, the new weight $p\left(s_{d_{(i)}}\middle|S\right)$ in Eq. 2 is revised as:

$$p\left(s_{d_{(i)}}\middle|S\right) = p_x\left(s_{d_{(i)}}\middle|S\right)p_y\left(s_{d_{(i)}}\middle|S\right)p_o\left(s_{d_{(i)}}\middle|S\right) \tag{16}$$

where $p_o\left(s_{d_{(i)}}\middle|S\right)$ is a sigmoid function that is the same as that in the original tracker. For minor drifting situations, such as Fig.6.b, the weights of patches are allowed to decrease gradually by:

$$w_k^{(i)} = \lambda_w w_{k-1}^{(i)} + (1-\lambda_w)\widehat{w}_k^{(i)} \tag{17}$$

where λ_w is a persistence constant. However, in cases such as Fig.6.c, the drifting should be resolved immediately. It is possible to detect this situation by extremely large values of the maximum non-zero interval, $T_{\Delta x}$. In such cases, λ_w is set to zero so that these drifting patches are immediately deleted.

4.3 Memory recovery in occlusion situations

The original tracker often fails in occlusion situations because information about occluded parts of the target is immediately and permanently deleted during model updating at each frame. An obvious way to overcome this problem is by continuously recording a memory of the visual model in its unoccluded state, and then recalling the memory once an occlusion state has been detected. Here we propose a simple but useful method for detecting such occlusions.

During the updating stage at each frame, local patches can be deleted from the target model for three reasons: low weights due to visual mismatching; low weights due to drifting; and nearby patches being merged. Normally, the first two situations happen frequently while merging patches only occur occasionally. During severe occlusions, most patches will rapidly converge on the unoccluded part of the target (see Fig.4), which rapidly becomes very small. Therefore, we propose two features for detecting occlusion situations: firstly, an extremely small size of bounding box scale; secondly, the rate at which patch mergers occur (reflect the speed of shrinking). This step takes place after local patch matching (step 2 in Fig.1). We can obtain the scale of bounding box $\hat{b}$ and the number of potential merging patches, $\widehat{N}$, from the current set of patch positions. When these values both satisfy the below inequalities, an occlusion situation is confirmed:

$$\hat{b} < T_b \text{ and } \widehat{N} > T_N \tag{18}$$

where T_b is a scale threshold and T_N is a threshold for the number of merged patches. Once an occlusion situation is detected, the model updating stage is stopped, and. a target model, recorded during the most recent unoccluded period, is recalled from memory. Essentially this this is equivalent to forcing the original coupled-layer model to temporarily degenerate into a single holistic model for tracking. In other words, patch positions remain the same, relative to their neighbors, in the patch constellation.

In this situation, we first predict the location of patches from the previous frame (giving all patches the same movement). Then, comparing current observed patches to their templates, one can compute likelihoods for each patch. If the average likelihood is very low, a mixed spatiotemporal prediction mechanism is triggered for tracking which is explained in details in section 4.4.

4.4 Mixed strategies for spatiotemporal prediction

In the original algorithm, a Kalman Filter is utilized for spatiotemporal prediction [14]. It is accurate and highly efficient for tracking a target with a known linear and Gaussian motion model. However, typically only a limited region of the image is searched (usually the same size as the target region) by the Kalman Filter, so that predictions fail when the target motion changes direction or the camera moves significantly.

When prediction is noisy or the tracker cannot find a good match, it is natural to search a larger region so that we can obtain the best estimation of the target state. Therefore, we first compute overall likelihood from the local patches to judge whether or not the tracker has found a good match. In contrast to the usual location optimization of patches, which considers both appearance and geometrical models (Eq.2), we only use the appearance model here to get the overall likelihood of patches by:

$$p_k = \frac{1}{N}\sum_{i=1}^{N} e^{-\rho(h_{ref}^{(i)},h_k^{(i)})} \quad (19)$$

where $\rho(.)$ is the Bhattacharyya distance between histogram [11]. $h_{ref}^{(i)}$ is the histogram of patch i when initialized and $h_k^{(i)}$ is the observed histogram at frame k. N is the number of local patches. A very low value of p_k indicates that Kalman Filter is becoming ineffective. Once, p_k is smaller than a threshold, the Kalman Filter is replaced by a Particle Filter [4] to more robustly estimate candidate target locations. In other words, this tracker will split into several trackers (particles). Each of them can obtain likelihood $p_k^{(j)}$ of particle j by Eq. 19.

After normalization of these likelihoods, the new estimation of target location from the particle state $T_k^{(j)}$ can be denoted by:

$$\hat{T}_k = \sum_{j=1}^{N_p} p_k^{(j)}\, T_k^{(j)} \quad (20)$$

In order to preserve efficiency, we do not perform the original algorithm's patch optimization procedure on every particle, but only on the overall estimated set of patch poses as derived from Eq.19.The particle-based tracking procedure is continued until the overall patch likelihood improves above a minimum threshold, at which point deterministic Kalman Filtering is resumed.

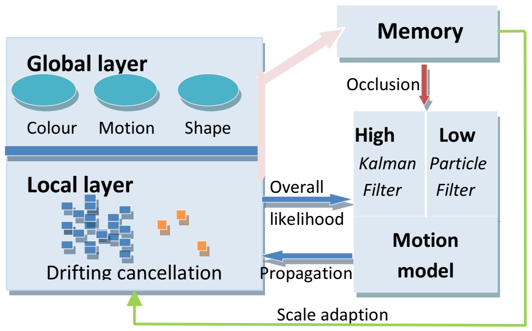


Figure 8: A schematic overview of modified steps

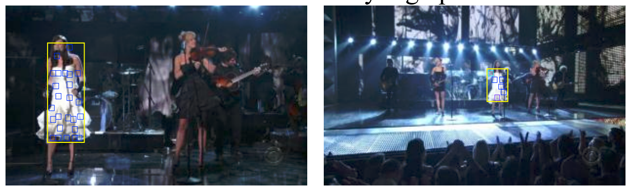
5. Experimental results

We evaluate our algorithm in two stages. First, we compare our enhanced tracker to the original coupled-layer tracker, and demonstrate how the enhancements are advantageous for some specific sequences. Then, the overall performance is evaluated using the public bench-mark toolkit [17].

5.1 Enhanced vs original trackers

Here, we demonstrate that the enhancements can decrease the failure rate and improve the precision for some specific sequences.

Fig. 9 shows an example of a target (*Singer*) which rapidly shrinks to a small size. With the adaptive patch scale enhancement, the bounding box accurately shrinks to match the target, whereas the original tracker with constant patch size fails (Fig.2.). The size of the target rapidly shrinks after frame 70. As shown in figure 10, the precision of the original tracker rapidly deteriorates during this time, whereas the enhanced tracker with adaptive patch scale maintains a consistently high precision.



Frame 10 Frame 250

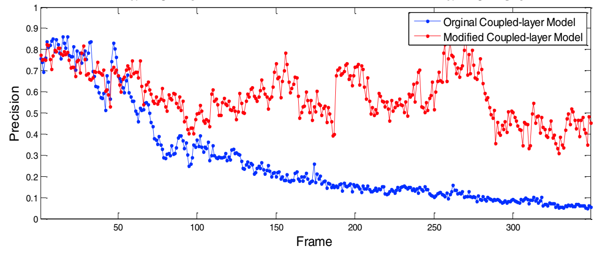
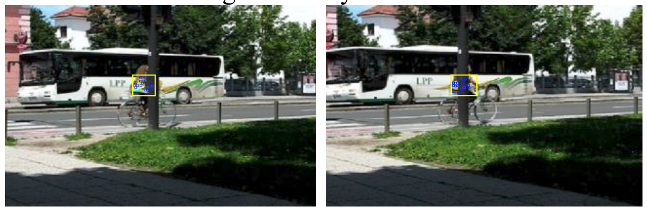


Figure9: Tracking performance: singer

Fig. 10 shows the performance of the enhanced tracker on a difficult occlusion problem, which can be compared the original tracker in Fig.4. The enhanced tracker retains a memory of the target history, automatically detects occlusion conditions, and recalls the memory to recover from occlusion. While the occlusion is occurring, the precision decreases dramatically for both trackers from frame 170. However, while the original algorithm continues to deteriorate, the enhanced algorithm automatically detects the occlusion situation and recovers by using memory. Once the target re-appears, the tracker can continue tracking accurately.



Frame 171 Frame 173

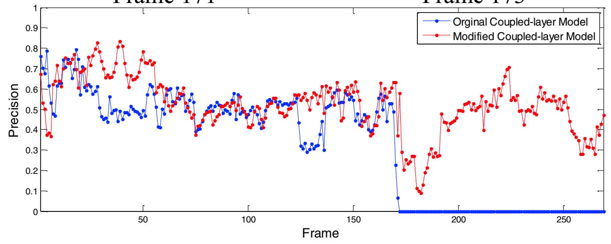


Figure10: tracking performance: bicycle

Fig. 11 shows the *Woman* sequence, where there is a fast movement of the camera view as it zooms in. Near to the end of the sequence, both trackers suffer a sharp decrease in precision due to rapid camera motion (nearly +40 pixels between some successive frames in *Woman* sequence, for a target of width order 40 pixels). Worse still, the velocity predicted by Kalman Filter is actually in the opposite direction of the real target motion. The enhanced tracker detects deteriorating tracking confidence, and triggers the Particle Filter to continue tracking the target successfully, while the original tracker does not recover and fails.

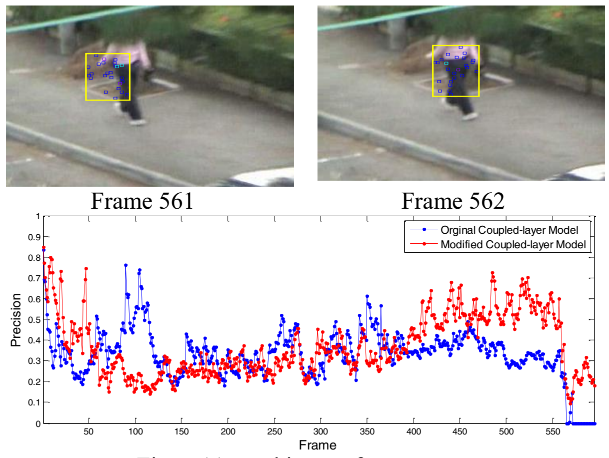


Figure11: tracking performance: woman

It is difficult to demonstrate the efficacy of our drifting solution with examples from specific sequences. However, we demonstrate the effectiveness of this modification though as part of the oveararll results comparison in Section 5.2.

5.2 Objective tracking evaluation

The overall tracking results are obtained by using the publicly available tracking benchmark toolkit of the ICCV 2013 VOT tracking workshop, [17], to evaluate trackers in terms of failure rate and accuracy. Since both LGT and LGT++ trackers are stochastic, each is run 15 times on each video sequence to generate statistically meaningful results. Each tracker is assessed on the criteria of "accuracy" and amount of "failure". Here, we first explain the statistical meaning of the "failure" criterion. For the results of each sequence, 0 means no failure (target was not lost) at all in all 15 repetitions of all 16 video sequences. An increase of 0.067 means the target was lost one more time during the 15 repetitions. The target is defined as "lost" when the tracker output bounding box has no overlap with the human-annotated ground-truth bounding box. Whenever the target is lost, the toolkit records the failure and then automatically re-initializes the bounding box and resumes tracking. Therefore, a bad tracker may "fail" a maximum of 15*(number_of_frames) times for each test video. Values below 1.0 mean the tracker seldom lost the target (less than once per sequence on average). For a total score, averaged over all 16 test videos, an increase of 0.004 means the tracker failed one extra time over the whole experiment. Accuracy φ_k is defined as the overlap between the tracker's predicted bounding boxTT_k and the ground truth bounding boxGT_k:

$$\varphi_k = \frac{TT_k \cap GT_k}{TT_k \cup GT_k} \tag{21}$$

Once the tracker drifts completely to the background, the measure becomes zero, regardless of how far from the target the tracker is currently located. The overlap measure is summarized over an entire sequence by an average overlap over the valid frames.

Experiment 1: runs trackers on all 16 sequences, initialized from ground truth bounding box at first frame.

Table 1 accurate initialization

Name	Failure		Accuracy	
	Original	Modified	Original	Modified
Bicycle	1.000	0.400^+	0.528	0.503^-
Bolt	0.067	0.067	0.414	0.446^+
Car	0.000	0.000	0.484	0.535^+
Cup	0.000	0.000	0.628	0.790^+
David	0.000	0.000	0.568	0.569^+
Diving	1.000	0.200^+	0.435	0.485^+
Face	0.000	0.000	0.598	0.497^-
Gymnastics	0.867	0.267^+	0.475	0.519^+
Hand	0.133	0.133	0.534	0.555^+
Ice-skater	0.000	0.000	0.548	0.556^+
Juice	0.000	0.000	0.695	0.716^+
Jump	0.000	0.000	0.588	0.525^-
Singer	0.000	0.000	0.215	0.474^+
Sunshade	0.333	0.200^+	0.541	0.542^+
Torus	0.000	0.000	0.673	0.750^+
Woman	0.867	0.267^+	0.338	0.352^+

The $(.)^+$ and $(.)^-$ symbols denote improved vs worse performance respectively of the enhanced tracker. The total evaluation scores, averaged over all 16 test videos, are original LGT: accuracy (0.516), failure (0.267); our LGT++ tracker: accuracy (0.551), failure (0.096).

LGT++ enhancements are most noticeable in terms of enhanced robustness (reduced failures). However, significant accuracy gains are also apparent in certain difficult tracking situations. In terms of accuracy, we can find large improvements in the *Cup* and *Singer* sequences but reduced accuracy in the *Face* sequence. Note that it is hard to demonstrate a clear improvement in accuracy, because the drifting solution enhancement tends to hold the bounding box tighter and focus on the main body of the target. In doing so, it may miss some small peripheral parts of the target thus adversely affecting accuracy metrics. However, this enhancement clearly decreases the risk of patches drifting onto background distracters in cluttered environments, thus enhancing robustness. We can see this effect most clearly in the *Diving* and *Gymnastics* sequences (note the large improvements in failure rates). As shown in Section 5.1, the improvement of *Bicycle* comes from the occlusion solution and *Woman* thanks to the mixed spatiotemporal prediction strategy.

Experiment 2: same as Exp. 1, but initialized with randomly perturbed bounding box. Original LGT tracker:

accuracy (0.513), failure (0.221); our LGT++ tracker: accuracy (0.541), failure (0.113).

Experiment 3: same as Exp. 1 but all images converted to grayscale. Original LGT tracker: accuracy (0.480), failure (0.783); Our LGT++: accuracy (0.520), failure (0.463).

6. Conclusion

We have identified four major causes of failure, in a state of the art tracker, and proposed a series of additional techniques to address each of these problems. Detecting and correcting patch drifting enhances performance in heavily cluttered environments. Enabling patches to scale, in addition to the bounding box, solves problems associated with rapid target size changes. Occlusion is automatically detected and resolved by recalling a target memory. Rapid erratic motion is addressed by means of a mixed spatiotemporal prediction strategy. Experimental results demonstrate that the enhanced tracker outperforms the original tracker on a variety of challenging sequences, showing significantly reduced failure rates and improved accuracy in most test sequences. Future work will focus on how to allocate the patches during initialization.

Acknowledgements

The authors would like to thank for the help from authors of [10]. Xiao is supported by the China Scholarship Council and school scholarship in the University of Birmingham. This work was in part supported by ARRS projects J2-2221 and J2-4284 and EU project CogX (FP7-ICT215181-IP).

7. Reference

[1] A. Yao, X. Lin, G. Wang, and S. Yu, "A Compact Association of Particle Filtering and Kernel based Object Tracking," *Pattern Recognition*, vol. 45, pp. 2584-2597, 2012.

[2] B. Stenger, T. Woodley, and R. Cipolla, "Learning to Track with Multiple Observers," *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 2647-2654, 2009.

[3] B. Yang and R. Nevatia, "Online Learned Discriminative Part-based Appearance Models for Multi-human Tracking," *Proc. European Conf. Computer Vision*, pp. 484-498, 2012.

[4] G. Kitagawa, "Non-Gaussian State—Space Modeling of Nonstationary Time Series," *Journal of the American statistical association*, vol. 82, pp. 1032-1041, 1987.

[5] H. Yang, L. Shao, F. Zheng, L. Wang and Z. Song, "Recent Advances and Trends in Visual Tracking: A Review," *Neurocomputing*, vol. 74, pp. 3823-3831, 2011.

[6] J. Fan, Y. Wu, and S. Dai, "Discriminative spatial attention for robust tracking," *Proc. European Conf. Computer Vision*, pp. 480-493, 2010.

[7] J. Kwon and K. M. Lee, "Tracking of a Non-rigid Object via Patch-based Dynamic Appearance Modeling and Adaptive Basin Hopping Monte Carlo Sampling," *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 1208-1215, 2009.

[8] J. Martínez-del-Rincón, C. Orrite, and C. Medrano, "Rao–Blackwellised Particle Filter for Colour-based Tracking," *Pattern Recognition Letters*, vol. 32, pp. 210-220, 2011.

[9] K. Fukunaga and L. Hostetler, "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition," *Information Theory, IEEE Transactions on*, vol. 21, pp. 32-40, 1975.

[10] L. Čehovin, M. Kristan and A. Leonardis, "Robust Visual Tracking Using an Adaptive Coupled-Layer Visual Model," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.35, no.4, pp.941-953, 2013.

[11] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based Probabilistic Tracking," *Proc. European Conf. Computer Vision*, pp. 661-675, 2002.

[12] Q. Zhao, Z. Yang, and H. Tao, "Differential Earth Mover's Distance with its Applications to Visual Tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 274-287, 2010.

[13] R. T. Collins, Y. Liu, and M. Leordeanu, "Online Selection of Discriminative Tracking Features," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, pp. 1631-1643, 2005.

[14] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of basic Engineering*, vol. 82, pp. 35-45, 1960.

[15] R. Stolkin, A. Greig, M. Hodgetts, and J. Gilby, "An EM/E-MRF Algorithm for Adaptive Model based Tracking in Extremely Poor Visibility," *Image and Vision Computing*, vol. 26, pp. 480-495, 2008.

[16] S. Wu, Y. Zhu, and Q. Zhang, "A New Robust Visual Tracking Algorithm based on Transfer Adaptive Boosting," *Mathematical Methods in the Applied Sciences,* vol. 35, pp. 2133-2140, 2012.

[17] The VOT 2013 evaluation kit. http://www.votchallenge.net.

[18] X. Wang, "Intelligent Multi-camera Video Surveillance: A Review, " *Pattern Recognition Letters*, vol. 34, no.1, pp. 3-19, 2013.

[19] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. Hengel," A Survey of Appearance Models in Visual Object Tracking," *TIST*, 2013, in press.

[20] Y.-M. Seong and H. Park, "Multiple Target Tracking Using Cognitive Data Association of Spatiotemporal Prediction and Visual Similarity," *Pattern Recognition*, vol. 45, pp. 3451-3462, 2012.