Online Tracking using Saliency

Mohammed A. Yousefhussien[1] N. Andrew Browning[2] Christopher Kanan[1*]
[1]Rochester Institute of Technology [2]Scientific Systems Company, Inc.
mxy7332@rit.edu, andrew.browning@ssci.com, kanan@rit.edu

Abstract

When tracking small moving objects, primates use smooth pursuit eye movements to keep a target in the center of the field of view. In this paper, we propose the Smooth Pursuit tracking algorithm, which uses three kinds of saliency maps to perform online target tracking: appearance, location, and motion. In addition to tracking single targets, our method can track multiple targets with little additional overhead. The appearance saliency map uses deep convolutional neural network features along with gnostic fields, a brain-inspired model for object recognition. The location saliency map predicts where the object will move next. Finally, the motion saliency map indicates which objects are moving in the scene. We combine all three saliency maps into a smooth pursuit map, which is used to generate bounding boxes for tracked objects. We evaluate our algorithm and others from the literature on a vehicle tracking task. Our approach achieves the best overall performance, including being the only method we tested capable of handling long-term occlusions.

1. Introduction

When tracking small moving objects, primates use two types of eye movements to keep the object foveated: smooth pursuit and saccades. Saccades are ballistic eye movements in which objects in the retinal periphery are brought to the center of the field of view. Smooth pursuit eye movements, in contrast, are continuous eye movements that attempt to counteract the object's movement to keep it continually in the center of the field of view. While saccades can be elicited in a wide variety of situations, smooth pursuit eye movements are only possible when an object is in motion [29]. Motivated by the neural circuits that underlie smooth pursuit, we created the Smooth Pursuit Tracking (SPT) algorithm. In primates, smooth pursuit is disabled by complete occlusions because the object is no longer seen moving. To cope with this, we introduce a crude fixation-like model when SPT is not confident in the target's position. We also show that SPT can do online tracking of multiple targets, which requires little additional overhead compared to single-target tracking.

[*]Corresponding author.

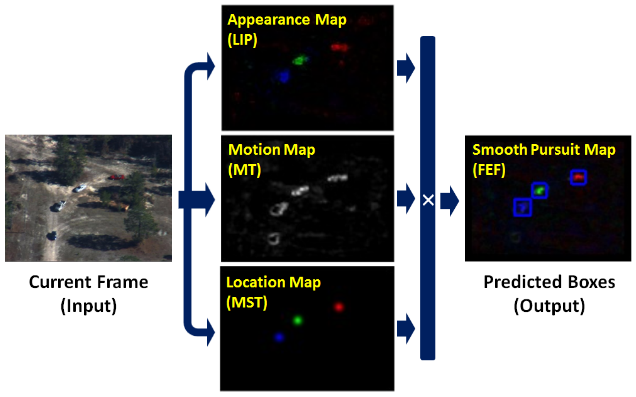

Figure 1. Inspired by how the primate brain tracks objects, in this paper we develop a new algorithm for online tracking, which combines three kinds of saliency maps: appearance, motion, and location. The algorithm can track multiple objects simultaneously with little additional overhead. We evaluate the algorithm on its ability to track ground vehicles in aerial video footage.

Saliency maps are topologically organized maps that represent the salient information in a visual scene [17]. The brains of humans and other animals are thought to contain saliency maps that allow them to determine what things in the visual field deserve attention [16, 17, 34]. Algorithms for saliency maps have been widely used to model human attention and to predict eye movements. There are two main kinds of saliency maps: bottom-up (or stimulus driven) and top-down (or task-driven). For static images, bottom-up maps assign higher saliency to rare features in the image, where the definition of rarity differs per algorithm. For videos, either feature rarity or motion are assigned high saliency. Bottom-up maps have been used to predict eye movements during free-viewing in static scenes and videos, when human subjects are not given a specific task [8, 45, 24]. Top-down maps assign high saliency to image regions that are likely relevant to the current task.

When trying to look for particular targets, the process is often modeled by looking for locations where the object is likely to be and regions that resemble the target's appearance [5, 21, 35, 42]. Here, we use one kind of bottom-up saliency map, object motion, and two kinds of top-down saliency, target appearance and future target location.

We evaluate our tracking algorithm on its ability to track moving vehicles in footage acquired by an aerial platform. This kind of footage is a great test for our algorithm, since smooth-pursuit in primates is used for tracking small moving objects. Moreover, tracking in aerial footage is becoming increasingly important. Unmanned aerial vehicles (UAVs) continue to become more popular every year, and they are now being used in film production, mining, construction, real estate, news media, and agriculture. Moreover, the world's security agencies are gathering enormous amounts of video data in which they are looking for events of interest, such as suspicious vehicles. When these vehicles of interest are found, in many cases they would like a UAV to follow the vehicle over time, which requires online visual tracking. Videos of these events can then be analyzed by a human analyst. Tracking ground vehicles in UAV video streams is especially difficult because we usually have a relatively small number of pixels on the target compared to other tracking problems, and targets can change drastically in appearance due to changes in lighting conditions, UAV altitude, and perspective.

2. Background

2.1. The Neural Circuits for Smooth Pursuit

Many regions of the brain are involved in smooth pursuit. Among the most important are the middle temporal visual area (MT or V5) [15, 32], medial superior temporal area (MST) [15], lateral intraparietal sulcus (LIP) [23], and frontal eye fields (FEF) [23]. MT receives information from earlier visual areas (V1 and V2), and nearly every cell responds to visual motion in a direction sensitive manner [15]. Pursuit cells in MT stop firing when blinking [28]. MST receives input directly from MT. During smooth pursuit, MST neurons respond to retinal image motion as well as head and eye movements [15], and many neurons continue to be active during blinking [28]. We interpret these brain regions as computing two kinds of saliency maps: MT is a map for target motion, and MST is a map that predicts the tracked object's future location.

LIP is a topologically organized brain region that has been implicated as one of the most important saliency maps in the brain for visual attention [11]. LIP neurons indicate which regions of the visual field are relevant to the current task. LIP is involved in both smooth pursuit and saccades, and it receives projections from many visual areas involved in feature extraction and object recognition, including V2, V4, and inferotemporal cortex (IT) [9]. Stimulation of LIP neurons can evoke smooth-pursuit [23], but little is known of its exact role in smooth pursuit. However, given its role in task-driven attention and because it receives information from V2, V4, and IT, we hypothesize that it is the source of the object appearance saliency map during tracking.

FEF is a topologically organized brain region that produces the motor commands responsible for eye movements, including smooth pursuit. It is often considered the location of the brain's final saliency map for eye movements [16, 34]. It receives information from many brain regions, including MT, MST, and LIP [33, 30]. V2, V4, and IT are all involved extracting visual features and understanding object appearance in an invariant manner, with IT being considered the location where high-level invariant object recognition occurs in the brain [4]. When engaged in smooth pursuit, we consider these regions as generating a map for the target object's appearance.

While we employ some of the same mechanisms as the neural circuits that underlie the primate smooth pursuit system, there are important distinctions between the biological system and our model. Primates physically move their eyes during smooth pursuit to keep the target centered on the fovea, the region of the retina with the highest resolution, but we cannot do this with pre-recorded videos. However, videos are of uniform resolution, so foveating regions is simply simulated by the predicted bounding box. While the brain's saliency maps for smooth pursuit control eye movements, for our model they are interpreted as controlling the target's predicted location. This is similar to the distinction between models of overt and covert attention.

2.2. Comparison Algorithms for Online Tracking

There have been a large number of tracking algorithms developed (for a review, see [41] and [43]). We compare our method to: L1 tracker [27], CVT [3], ASLA [18], MIL [2], KCF [13], OAB [12], and SPOT [46]. There were no publicly available results using our evaluation metrics on the dataset we evaluated on, so we ran code for each of these algorithms ourselves. We briefly summarize how each of these online trackers works. With the exception of SPOT [46], each of these trackers is only designed to follow one object at a time.

The L1 tracker [27] poses the visual tracking problem in a particle filter framework. It tracks objects by finding a sparse approximation of the target in a template subspace, where the sparsity is achieved by solving an L1-regularized least squares problem. The L1-tracker incorporates a mechanism to handle occlusions. ASLA [18] uses a set of target templates to perform tracking. The ground-truth target is divided into a set of overlapping local image patches, which are then sampled to build a sparse dictionary representing the target's appearance. The target is then detected in subsequent frames using a dynamical Bayesian framework with an alignment pooling operator. MIL tracker [2] uses an online variant of boosting that incorporates Multiple Instance Learning. Instead of using a single positive patch to update its classifier, it uses a collection of positive patches. KCF Tracker [13] uses a dense sampling strategy around the given bounding box to produce a circulant matrix structure when kernels are applied. The algorithm uses a regularized least square classifier by labeling the training samples with continuous Gaussian values. Using the circulant structure of the matrix, the tracker uses the filter theorem to perform tracking in the frequency domain with high frame-rate. CVT [3] integrates the color attribute model of [37] into the KCF [13]. CVT constructs a multi-dimensional color attribute vector by assigning observations in the RGB color space to linguistic color labels. OAB tracker [12] uses an online version of AdaBoost to update a feature ensemble used for tracking the target. OAB uses the initial bounding box and the surrounding background as positive and negative examples. The weak classifiers use hand-crafted features. While most multiple-target tracking algorithms are used offline, SPOT [46] is one of the few online multi-target tracking algorithms. Using a structured SVM trained on features from the first frame, it jointly learns object appearance and structural constraints among the targets. The published implementation of SPOT requires multiple targets to be tracked, so it is only evaluated on our multi-target tracking video.

2.3. Related Deep ConvNet and Saliency Trackers

In computer vision, there is currently a great interest in deep convolutional neural networks (ConvNets) due to their efficacy at object recognition, object detection, semantic segmentation, and other areas. It is also possible to simply use ConvNets as a kind of feature representation, treating the output of each convolutional layer as a collection of topologically organized and highly discriminative descriptors. We adopted this approach.

While we use ConvNets to extract image descriptors that are fed into gnostic fields, a few recent papers have used ConvNets or other deep learning algorithms to track objects in videos. In [40], the authors used a stacked denoising auto-encoder for tracking. During tracking, the algorithm draws particles as candidate locations from the new frame. The last layer of the auto-encoder was used to represent the features of the target and background, and a logistic regression classifier was trained to detect the target. Unlike our approach, they did not use motion information and their features were not trained to be discriminative for recognizing objects. In closely related work, [14] performed target tracking by creating a target specific saliency map found by back-propagating SVM weights. The map was then used to generate an appearance model distribution in a Bayesian filtering framework. In [25], the authors proposed the discriminant saliency model. They first filter an image with DCT filters, and then each filter response is used to generate a saliency map. The target is detected at the location of the maximum value in the sum of the saliency maps. Unlike us, the authors did not use motion or location saliency. Instead of DCT features, we used ConvNet features, along with Gnostic Fields to estimate an appearance saliency map.

In [44], the authors developed a bottom-up saliency tracker that tracked any salient target in the scene, which was done in an entirely unsupervised manner. This is in contrast to the typical online tracking problem, in which a particular target is to be followed throughout a video. The algorithm uses color features and sparse optical flow to generate a bottom-up saliency map. The salient region in the frame is tracked using a particle filter.

In [24], the authors used a ConvNet that is trained online to discriminate and track a target from the surrounding background. The algorithm proposes a truncated structural loss function to avoid the constraint of requiring large training data. In subsequent frames, the target is detected by classifying candidates using sliding window approach. The method did not use motion or location saliency. In [7], the authors used a ConvNet to track humans in videos. They trained the ConvNet to learn both spatial and temporal features together.

3. The Smooth Pursuit Tracking Algorithm

Our SPT algorithm multiplicatively combines appearance, location, and motion saliency maps to generate a smooth pursuit map. We assume that each of the sub-saliency maps contains only a few salient regions, so the multiplicative combination of them should produce a sparse map with the most salient region containing the target. Formally, to track a target k, SPT defines the smooth pursuit saliency map s as

$$s\left(k,t,z\right) = a\left(k,t,z\right)\ell\left(k,t,z\right)m\left(t,z\right), \tag{1}$$

where z is the (x,y) Cartesian pixel location, t represents the frame number, a is a function that computes the appearance saliency map, ℓ is a function that computes the location saliency map, and m is a function that computes the motion saliency map. This equation is depicted in Fig. 1.

We explain how each of these saliency maps are computed in the following subsections, and example saliency maps are shown in Fig. 2. As explained in Section 3.5, our algorithm uses an alternative technique when its confidence is low, enabling it to handle occlusions.

3.1. Appearance Saliency

Top-down saliency maps use information about the algorithm's goal to generate the saliency map. When a person

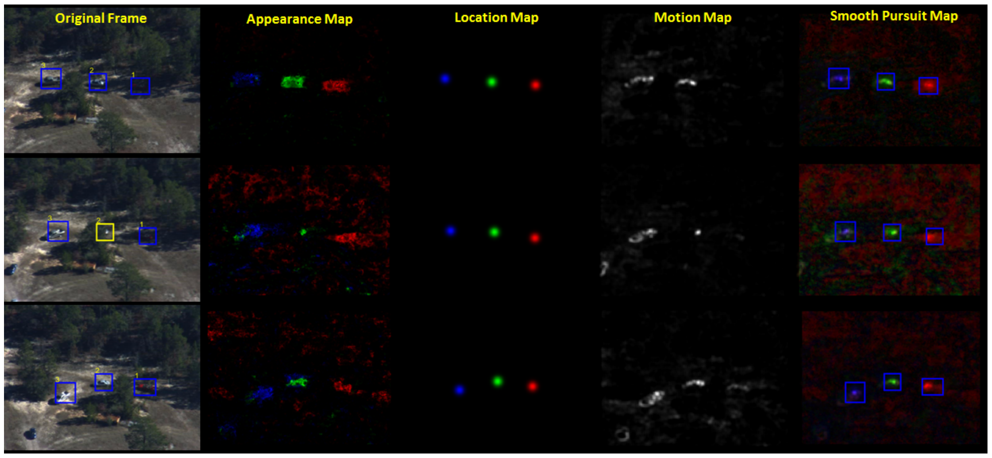

Figure 2. The appearance, location, motion, and smooth pursuit saliency maps for three frames of a video, where we are overlaying saliency maps for each of the three vehicle targets using different colors (blue, green, and red) along with the current bounding box prediction.

is trying to look for a particular kind of object, top-down saliency maps driven by object appearance are far more predictive of their eye movements than bottom-up models [21]. These saliency maps have high values in regions that correspond to areas of the image that resemble the target.

Here, we created a new appearance-based saliency algorithm using an online variant of gnostic fields. The neuroscientist Jerzy Konorski proposed gnostic fields in 1967 as a theoretical model for how the brain performs object recognition [22]. A gnostic field consists of competing gnostic sets, with each set containing a population of gnostic neurons that act as template matchers for a particular class. More recently, gnostic fields were turned into an algorithm and achieved good results on various object recognition benchmarks [19, 20]. Gnostic fields generally operate on dense topologically organized image descriptors, and each gnostic set compares each descriptor to the learned templates for the class. We have adapted this object recognition model to serve as our appearance saliency map for each tracked target.

We use one gnostic field for each feature channel c (*i.e.*, feature type), which can serve as a classifier for all K targets of interest. A gnostic field first applies a whitening matrix $\mathbf{W}_c$ to the features $\mathbf{g}_{c,t,z}$, which decorrelates and reduces the dimensionality of the features, followed by making these features unit length, *i.e.*,

$$\mathbf{f}_{c,t,z} = \frac{\mathbf{W}_c \mathbf{g}_{c,t,z}}{\parallel \mathbf{W}_c \mathbf{g}_{c,t,z} \parallel}. \tag{2}$$

For each channel, $\mathbf{W}_c$ is computed using whitened principal components analysis applied to the descriptors extracted in the first video frame, and only the first 100 principal components are retained.

To track K objects, we create a gnostic field with $K+1$ gnostic sets, with one set per object plus one set representing the background. The initial activity of each gnostic set is given by the most active unit in the set, *i.e.*,

$$a_{c,k,t,z} = \max_j (\hat{\mathbf{v}}_{c,k,j} \cdot \mathbf{f}_{c,t,z}), \tag{3}$$

where each $\hat{\mathbf{v}}_{c,k,j}$ represents gnostic unit j for set k that has been normalized to unit length, *i.e.*, $\hat{\mathbf{v}}_{c,k,j} = \frac{\mathbf{v}_{c,k,j}}{\|\mathbf{v}_{c,k,j}\|}$. Subsequently, we competitively normalize the responses of the gnostic sets,

$$q_{c,k,t,z} = \max\left(a_{c,k,t,z} - \theta_{c,t,z}, 0\right), \tag{4}$$

where $\theta_{c,t,z} = \frac{1}{K+1} \sum_{k'} a_{c,k',t,z}$. The non-zero responses are then normalized using

$$\beta_{c,k,t,z} = q_{c,k,t,z} \frac{\sum_{k'} q_{c,k',t,z}}{(r + \sum_{k'} q^2_{c,k',t,z})^{3/2}}, \tag{5}$$

where $r = .01$ regularizes the strength of the normalization. This competitive normalization step was found to be crucial for ensuring good object recognition accuracy in [19]. We then bilinearly interpolate $\beta_{c,k,t,z}$ to align it with and make it the same size as the input frame at time t, which yields $\beta'_{c,k,t,z}$. To create the final appearance map for the target, these are summed across all channels and then the map is normalized across spatial locations to sum to one, *i.e.*,

$$a\left(k,t,z\right) = \frac{\sum_c \beta'_{c,k,t,z}}{\sum_z \sum_c \beta'_{c,k,t,z}}. \tag{6}$$

Earlier papers on gnostic fields were trained offline and used spherical k-means to learn the gnostic units. However, that approach is not suitable for online tracking, so we created an online version of spherical k-means, in which the number of units is allowed to increase. To incorporate a descriptor $\mathbf{f}_{c,h}$ into the gnostic set corresponding to channel c

and tracked object k, we do the following. If the gnostic set contains one or more units, then the descriptor is compared to all of the set's units to find the most active unit j', *i.e.*,

$$j' = \arg\max_{j} \hat{\mathbf{v}}_{c,k,j} \cdot \mathbf{f}_{c,h}. \tag{7}$$

If $\hat{\mathbf{v}}_{c,k,j'} \cdot \mathbf{f}_{c,h} > \rho$, where ρ acts as a similarity threshold, or if $\psi_{c,k} = \lambda$, where $\psi_{c,k}$ represents the current number of units in the gnostic set and λ represents the maximum number of units allowed, then we update unit j' using

$$\mathbf{v}_{c,k,j'} = \alpha \mathbf{f}_{c,h} + (1-\alpha)\,\mathbf{v}_{c,k,j'}, \tag{8}$$

where α controls the balance between the old and new representations. This update rule is based on the exponential moving average, so it could potentially forget object appearances in the distant past. Otherwise, the descriptor is simply copied into the set as an additional unit. In our experiments, we set $\lambda = 1000$, $\alpha = 0.5$, $\rho = 0.98$ for the tracked targets, and $\rho = 0.9$ for the background set. No attempt was made to tune these parameter values.

In the first frame, the gnostic sets for target k are initialized using all of the descriptors within the ground truth bounding box. All descriptors not assigned to targets are used to initialize the background gnostic set. In subsequent frames, we run a Harris corner detector and we only update the target gnostic sets with descriptors at the corners found within the predicted target bounding boxes, and we use descriptors that are not located within any predicted target boxes to update the background gnostic set.

For image descriptors, we used multiple layers of a ConvNet that was pre-trained on ImageNet. Specifically, we used the 16-layer ConvNet known as VGG16 [31], which has 13 convolutional layers and 3 fully-connected layers. The fully-connected layers were not used. We convolved the network with each video frame, and used the output of convolutional layers 4, 7, 10, and 13 as distinct channels that were input to four gnostic fields. Respectively, these descriptors were 128–, 256–, 512–, and 512–dimensional, and for 640×480 images (as in our dataset), each respective layer produced 16384, 4096, 1024, and 256 topologically organized descriptors. For the descriptors, all negative values were set to zero prior to feeding them to the gnostic fields (*i.e.*, applying the rectified linear unit non-linearity). We used the MatConvNet toolbox [38] to run the network.

3.2. Location Saliency

For each tracked object k in a video, we estimate the location saliency map $\ell(k,t,z)$ using the Kalman filter prediction stage. The Kalman filter is applied to generate the location map, and it is not used as the main tracking framework. The filter predicts the future location of the target k using a constant velocity motion model, based on online updates of its past locations. In the first frame, the center location of the target's bounding box is used to initialize the Kalman filter. In subsequent frames, if the confidence in the target's position is sufficiently high (see Sec. 3.5), the Kalman filter is updated using the new location, and then the prediction stage is used again to estimate the location in the next frame. However, if the confidence is low, *e.g.*, due to an occlusion, Kalman filter will not be updated and the prediction stage of the last updated model will be used. It should be noted that We only used the Kalman filter to predict the target's central location and not the size of the target. Instead, we used a spherical 2-dimensional Gaussian centered at that location with a fixed variance of $\sigma^2 = 300$, which was chosen in preliminary experiments.

3.3. Motion Saliency

One of the key characteristics of smooth pursuit is that it can only be used to track moving objects. We capture object movement in frame t using a motion (or spatiotemporal) saliency map $m(t,z)$, which uses a simple background subtraction algorithm. More sophisticated spatiotemporal saliency map algorithms are possible (e.g., [26]), but we have chosen to adopt this simple approach because of its speed and ability to compensate for camera motion.

The motion saliency map is created by performing background subtraction using an average model of the previous n frames as the background, *i.e.*,

$$m'(t,z) = \left| I_{t,z} - \frac{1}{n}\sum_{j=1}^{n} I'_{t-j,z} \right|, \tag{9}$$

where $I_{t,z}$ is the current frame and $I'_{j,z}$ is a version of the frame at time j that has been aligned to the frame at time t. This representation is normalized by subtracting the mean across locations and doing half-wave rectification, *i.e.*,

$$m(t,z) = \max\left(m'(t,z) - \theta(t,z), 0\right) + \epsilon, \tag{10}$$

where $\theta = \frac{1}{n}\sum_{z'} m'(t,z')$ and $\epsilon = .01$ to ensure that if the tracked objects have stopped moving that the smooth pursuit map will not be null. Subsequently, $m(t,z)$ is normalized to sum to one. Aligning the previous frames to the current frame is necessary to counteract camera motion. We did this alignment using the MATLAB Image Alignment Toolbox [6]. We configured the toolbox to find corresponding points between images using SURF descriptors and a RANSAC fitting was used to estimate the transformation between the correspondences. In our experiments, we set $n = 3$ in general, but if less than n frames were available then all of the frames were used. The versions of the images used in the background subtraction are in an opponent color-space [10], similar to that used by the retina and lateral geniculate nucleus in primates.

3.4. Getting Boxes from the Smooth Pursuit Map

To generate bounding boxes from the smooth pursuit map for a target k in frame t, we find the location of the largest value in the saliency map, *i.e.*, $z' = \arg\max_z s(k,t,z)$. Then we threshold the map to set all values less than $\frac{1}{3}s(k,z',t)$ to zero and setting all other points to one. Finally, we find the connected components about point z' and use that to generate the predicted bounding box $B_{k,t}$.

3.5. Handling Occlusion

In primates, smooth pursuit only works when an object being tracked is moving and visible, meaning other mechanisms need to be used to recover a track that has been lost due to occlusion. Primates use saccades to recover the object's location in this situation. Here, we use an object detection algorithm to try to do the same.

To determine if a track k has been lost in frame t we use the appearance saliency map prior to its final normalization by taking the mean around $B_{k,t}$, which is the bounding box predicted by the smooth pursuit map, *i.e.*,

$$\phi_{k,t}(B_{k,t}) = \frac{\sum_{z\in B_{k,t}} a(k,t,z)}{|B_{k,t}|\max_{z'} a(k,t,z')}, \tag{11}$$

where $|B_{k,t}|$ contains the number of points in the bounding box. The greater $\phi_{k,t}$ is for $B_{k,t}$, the more confident the system that the box contains the target. If $\phi_{k,t}(B_{k,t}) < 0.4$ then the system may not use $B_{k,t}$. Instead, we run the Selective Search algorithm [36] on the appearance and temporal saliency maps to generate bounding box hypotheses. Selective Search hierarchically segments the maps and generates a bounding box hypotheses. We prune the box hypotheses to constrain their centers to be within $10p$ pixels, where p is the number of frames in which confidence was low, *i.e.*, $\phi_{k,t} < 0.4$ has been true. This allows the algorithm to search for boxes farther away as the length of time the track has been lost increases. For each box hypothesis we evaluate equation 11. Let the box hypothesis with the greatest confidence be $B'_{k,t}$. If the confidence in $B'_{k,t}$ is greater than 0.4, then we use it instead of $B_{k,t}$.

4. Results

Here we present the results of our algorithm on the VIVID dataset. VIVID is a large video dataset captured by an aerial platform [1]. Each video contains multiple vehicles moving on dirt or paved roads. Many videos have occlusions, such as tree canopy, and others have identical vehicles moving in formation, making it especially challenging for methods that rely on appearance alone to succeed because the tracks can easily be confused. Only five RGB VIVID videos have been annotated by others, and these annotations only have ground-truth for a single vehicle in each video. We first show results on these single-vehicle tracking results. To study our approach's ability to track multiple vehicles simultaneously, we also hand-annotated three vehicles in a single VIVID video. We quantitatively compare our algorithm to the trackers mentioned in Section 2.2. All algorithms, including our own, had all of their meta-parameters fixed across all videos. All of our qualitative results are presented in videos[1].

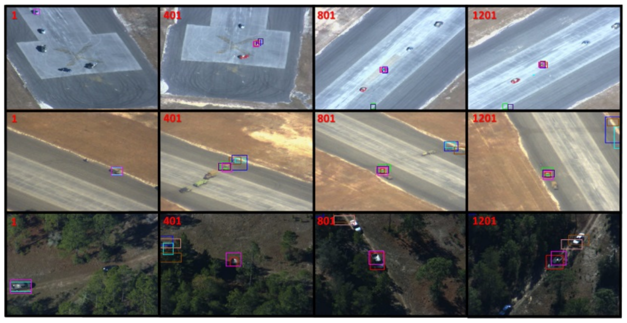

Figure 3. Frames from three single-target tracking videos with predicted tracks for each algorithm. Each row is for Egtest01, Egtest03, and Egtest05 respectively, while the columns represent the selected frames. The bounding boxes are colored as follows: ground-truth, CVT, ASLA, L1, MIL, KCF, OAB, and Ours.

We measure the performance of the algorithms using recent standard metrics for online tracking [41]: precision plots, success plots, and center location error (CLE) plots. For success plots, the Area Under the Curve (AUC) is reported in the figure, and we also report the precision at a threshold of 20 in precision plots. The mean CLE for the whole video per tracker is also reported.

4.1. Tracking Individual Objects

Fig. 3 shows four example frames for three of the five videos in each row, Egtest01, Egtest03, and Egtest05, as well as predicted bounding boxes for each method. See supplementary materials for additional figures. For video Egtest01, all of the trackers were able to follow the target for the first 400 frames. By frame 420, other trackers start to drift, and only our method, ASLA and OAB continued tracking the target throughout the video. In contrast, for video Egtest03 (second row), ASLA performs poorly, while ours and OAB track well. The third row, video Egtest05, contains strong changes in illumination and has long occlusions, but our method tracked well throughout the video compared to all trackers. Table 1 shows the mean CLE for each video, as well as the average CLE across videos. Overall, our method performed significantly better than the other algorithms, and our overall error was the lowest by a margin

[1]A list of links can be found in the supplementary materials, and a YouTube playlist with these videos can be found here: `https://goo.gl/7XVR9v`

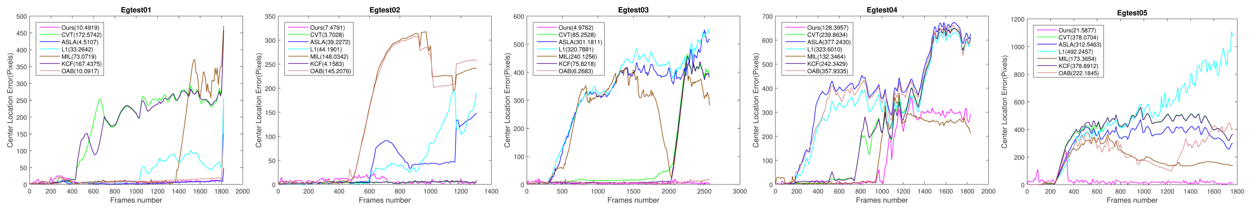

Figure 4. Center location error on the five VIVID videos in which only a single-target was tracked.

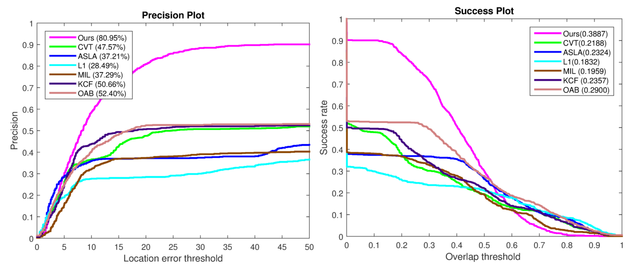

Figure 5. Precision and success plots for the single-target tracking results, averaged across all five videos.

of 117 pixels. While our method did not score the best on two of the videos, the differences from the best and second best methods were minor (see Table 1).

Fig. 4 shows CLE for each frame in the individual videos. In video Egtest03, our method outperformed most of the trackers by a large margin, scoring the mean CLE of only 4.9 pixels. In comparison, all other trackers, except for OAB, had a mean CLE of over 60 pixels. Our method outperformed most of the others by a margin of over 100 pixels for Egtest04, except for MIL. In Egtest05, our algorithm outperforms all the trackers by at least 150 pixels. Interestingly, some of the algorithms produced similar error patterns. For example, in Egtest04 video, ASLA, OAB and L1 have similar error curves, which hints that neither algorithm can handle the series of occlusions presented in the video. Also, in Egtest05, which contains changes in illumination and long occlusions, ASLA, KCF and CVT have similar error curves with only a height offset. Fig. 5 shows the precision and success plots averaged across all five videos. Our method performs better than the comparison algorithms by a large margin. For precision, our method scored 81% at a threshold of 20 pixels, and the second closest is the OAB tracker with 52%. This indicates that our method's predicted bounding boxes were consistently close to the ground truth bounding box centers. Similarly, in the success plot, our method achieves very good results, with an AUC of 0.39. OAB again was the second best method, achieving an AUC of 0.29.

4.2. Simultaneously Tracking Multiple Objects

We used VATIC [39] to carefully annotate three vehicles in a challenging VIVID video that contains numerous long duration occlusions. While our method and SPOT [46] can track multiple objects simultaneously, for the other methods we simply re-run their algorithm three times, *i.e.*, once per vehicle. Our algorithm did especially well on the multi-target tracking video compared to the other methods[2]. As shown in Fig. 6, the mean CLE across the three vehicles was 13 pixels for our method, while the second best was 179 pixels for SPOT. Fig. 7 shows the precision and success plots for each of the trackers averaged across the three vehicles, and our method performed very well by these metrics.

4.3. Appearance vs. Location vs. Motion

How important is each component of SPT? To study this, we created all sensible sub-models of SPT: 1) $a\left(k,t,z\right)\ell\left(k,t,z\right)m\left(t,z\right)$, 2) $a\left(k,t,z\right)\ell\left(k,t,z\right)$, 3) $a\left(k,t,z\right)m\left(t,z\right)$, 4) $\ell\left(k,t,z\right)m\left(t,z\right)$, and 5) $a\left(k,t,z\right)$. We evaluated each of these models on all of the single-target videos. Results with center location error are shown in Table 2. Overall, the entire model works better than any of the sub-models. Location and motion alone is the second-best model, and the three top models all use motion, which is consistent with its central role in our model of smooth pursuit and in the primate visual system. Additional results are shown in Supplementary Materials.

4.4. Timing Benchmarks

A comparison of the speed for all of the methods is shown in Table 3. We did our timing benchmarks on a desktop with an Intel Core i7-5820K CPU and an NVIDIA Titan X, which was used to speed up ConvNet feature computation. SPT is written in MATLAB, and it is not optimized for speed. While our implementation is slower than most of the methods, it was faster than OAB. We further analyzed how much time each component of SPT requires. Feature computation took 0.07s, computing the motion map took 0.16s, computing the appearance map took 0.08s, and computing the location map took 0.01s. One of the most expensive

[2]A YouTube multi-target video showing the results for all trackers can be found here: `https://goo.gl/ptIoHA`

Table 1. Mean center location error for each of the VIVID videos in the single-object tracking results and the mean error across all of the videos. For each video, the algorithm with the lowest error is denoted in red and the second lowest error is denoted in blue.

	Ours	**CVT**	**ASLA**	**L1**	**MIL**	**KCF**	**OAB**
Egtest01	10.49	172.57	4.51	33.26	73.07	167.43	10.09
Egtest02	7.47	3.70	39.22	44.19	148.03	4.15	145.20
Egtest03	4.97	85.25	301.18	320.78	240.12	75.82	6.26
Egtest04	128.39	239.86	377.24	323.60	132.34	242.34	357.93
Egtest05	21.58	378.07	312.54	492.24	173.36	378.89	222.18
Mean	34.57	170.37	206.93	242.81	153.38	173.72	151.93

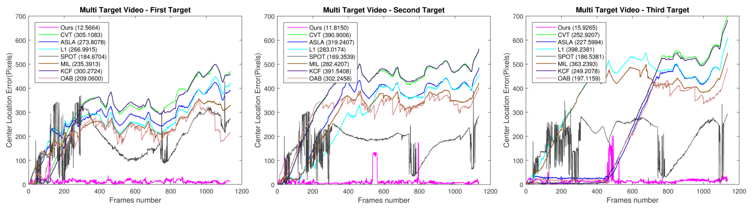

Figure 6. Center location error for each target in the multi-target video.

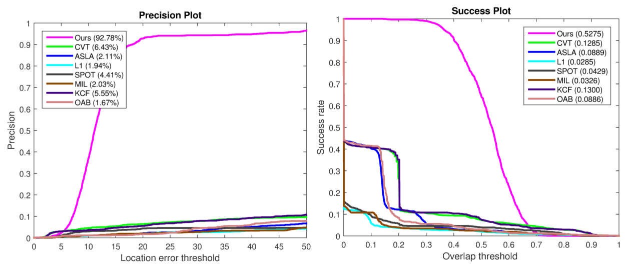

Figure 7. Precision and success plots for multi-target tracking, computed by averaging across the three targets in the video.

Table 2. Mean CLE values for SPT sub-models

	$a\,\ell\,m$	$\ell\,m$	$a\,m$	$a\,\ell$	a
Egtest01	10.49	105.37	99.17	229.49	108.59
Egtest02	7.47	92.50	110.64	134.01	306.89
Egtest03	4.97	117.69	100.75	174.49	203.43
Egtest04	128.39	81.97	179.54	312.97	167.25
Egtest05	21.58	59.55	170.16	87.39	166.47
Mean	34.57	91.41	132.05	187.67	190.52

Table 3. The speed, in seconds per frame, and implementation language for each tracker.

Method	**Overall Speed (s)**	**Language**
Ours	0.65	MATLAB
CVT	0.02	MATLAB
ASLA	0.52	MATLAB
L1	0.08	MATLAB
MIL	0.28	C++
KCF	0.07	C++
OAB	0.71	C++
SPOT	0.27	MATLAB

operations was Selective Search, which required 0.93s each time it was invoked. The frame alignment required to compensate for camera motion in the computation of the motion map is expensive, and most of that time could be eliminated if a stationary camera was used.

5. Conclusions

In this paper, we proposed an algorithm for online visual tracking of small moving objects. SPT is motivated by how the primate smooth pursuit system is thought to work, and the algorithm elegantly combines saliency maps for appearance, future location, and motion. Although primates are limited to tracking one object at a time using smooth pursuit, SPT extends easily to tracking multiple objects simultaneously with little computational overhead. We showed that the algorithm surpassed comparison methods from the literature on tracking moving vehicles in videos acquired by an aerial platform.

Acknowledgments

Research reported in this article was supported in part by NAVAIR under contract #N68335-14-C-0331. The content is solely the responsibility of the authors and does not necessarily represent the official views of the sponsoring agencies.

References

[1] VIVID tracking evaluation web site. url: http://vision.cse.psu.edu/data/vivideval/datasets/datasets.html.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[3] M. Danelljan, F. Khan, M. Felsberg, and J. van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proc IEEE Computer Vision and Pattern Recognition*, 2014.

[4] J. DiCarlo, D. Zoccolan, and N. Rust. How does the brain solve visual object recognition? *Neuron*, 73:415–434, 2012.

[5] K. Ehinger, B. Hidalgo-Sotelo, A. Torralba, and A. Oliva. Modeling search for people in 900 scenes. *Visual Cognition*, 17:945–978, 2009.

[6] G. Evangelidis. IAT: A matlab toolbox for image alignment, 2013.

[7] J. Fan, W. Xu, Y. Wu, and Y. Gong. Human tracking using convolutional neural networks. *IEEE Transactions on Neural Networks*, 21:1610–1623, 2010.

[8] D. Gao, V. Mahadevan, and N. Vasconcelos. On the plausibility of the discriminant center-surround hypothesis for visual saliency. *Journal of Vision*, 8(7):1–18, 2008.

[9] R. Gattass, S. Nascimento-Silva, J. Soares, B. Lima, A. Jansen, A. Diogo, M. Farias, M. Botelho, O. Mariani, J. Azzi, and M. Fioani. Cortical visual areas in monkeys: location, topography, connections, columns, plasticity and cortical dynamics. *Philos Trans R Soc Lond B*, 360:709–731, 2005.

[10] J.-M. Geusebroek, R. Van den Boomgaard, A. W. Smeulders, and H. Geerts. Color invariance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(12):1338–1350, 2001.

[11] M. Goldberg, J. Bisley, K. Powell, and J. Gottlieb. Saccades, saliency and attention: the role of the lateral intraparietal area in visual behavior. *Progress in Brain Research*, 155:157–175, 2006.

[12] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proc. BMVC*, 2006.

[13] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *proceedings of the European Conference on Computer Vision*, 2012.

[14] S. Hong, T. You, S. Kwak, and B. Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015.

[15] U. Ilg. The role of areas mt and mst in coding of visual motion underlying the execution of smooth pursuit. *Vision Research*, 48:2062–2069, 2008.

[16] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40:1489–1506, 2000.

[17] L. Itti and C. Koch. Computational Modeling of Visual Attention. *Nature Reviews Neuroscience*, 2(3):194–203, 2001.

[18] X. Jia, H. Lu, and M. Yang. Visual tracking via adaptive structural local sparse appearance model. In *Proc IEEE Computer Vision and Pattern Recognition*, 2012.

[19] C. Kanan. Recognizing sights, smells, and sounds with gnostic fields. *PLOS ONE*, e54088, 2013.

[20] C. Kanan. Fine-grained object recognition with gnostic fields. In *Proceedings of the IEEE Winter Applications of Computer Vision Conference*, 2014.

[21] C. Kanan, M. Tong, L. Zhang, and G. Cottrell. SUN: Top-down saliency using natural statistics. *Visual Cognition*, 17:979–1003, 2009.

[22] J. Konorski. *Integrative Activity of the Brain*. Univ. Chicago Press, Chicago, 1967.

[23] R. Krauzlis. Recasting the smooth pursuit eye movement system. *Journal of Neurophysiology*, 91(2):591–603, 2004.

[24] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *Proceedings of the British Machine Vision Conference*, 2014.

[25] V. Mahadevan and N. Vasconcelos. Saliency-based discriminant tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[26] V. Mahadevan and N. Vasconcelos. Spatiotemporal saliency in dynamic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:171–177, 2010.

[27] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 33:2259–2272, 2011.

[28] W. Newsome, R. Wurtz, and H. Komatsu. Relation of cortical areas MT and MST to pursuit eye movements. II. Differentiation of retinal from extraretinal inputs. *Journal of Neurophysiology*, 60(2):604–620, 1988.

[29] C. Rashbass. The relationship between saccadic and smooth tracking eye movements. *Journal of Physiology*, 159:326–338, 1961.

[30] J. Schall, A. Morel, D. King, and J. Bullier. Topography of visual cortex connections with frontal eye field in macaque: convergence and segregation of processing streams. *Journal of Neuroscience*, 15:4464–4487, 1995.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[32] M. Spering and A. Montagnini. Do we track what we see? Common versus independent processing for motion perception and smooth pursuit eye movements: A review. *Vision Research*, 51:836–852, 2011.

[33] G. Stanton, C. Bruce, and M. Goldberg. Topography of projections to posterior cortical areas from the macaque frontal eye fields. *Journal of Comparative Neurology*, 353:291–305, 1995.

[34] B. Thompson. A visual salience map in the primate frontal eye field. *Progress in brain research*, 147:249–262, 2005.

[35] A. Torralba, A. Oliva, M. Castelhano, and J. Henderson. Contextual guidance of eye movements and attention in real-world scenes: The role of global features on object search. *Psychology Review*, 113(4):766–786, 2006.

[36] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 2013.

[37] J. Van De Weijer, C. Schmid, J. Verbeek, and D. Larlus. Learning Color Names for Real-World Applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, July 2009.

[38] A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014.

[39] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, pages 1–21. 10.1007/s11263-012-0564-1.

[40] N. Wang and D.-Y. Yeung. Learning a deep compact image representation for visual tracking. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 809–817. 2013.

[41] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Proc IEEE Computer Vision and Pattern Recognition*, 2013.

[42] J. Yang and M. Yang. Top-down visual saliency via joint CRF and dictionary learning. In *Proc IEEE Computer Vision and Pattern Recognition*, 2012.

[43] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45, 2006.

[44] G. Zhang, Z. Yuan, N. Zheng, X. Sheng, and T. Liu. Visual saliency based object tracking. In *Computer Vision ACCV 2009*. 2010.

[45] L. Zhang, M. Tong, T. Marks, H. Shan, and G. Cottrell. SUN: A Bayesian framework for saliency using natural statistics. *Journal of Vision*, 8:1–20, 2008.

[46] L. Zhang and L. van der Maaten. Structure preserving object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.