# Deep-LK for Efficient Adaptive Object Tracking

Chaoyang Wang, Hamed Kiani Galoogahi, Chen-Hsuan Lin, and Simon Lucey[1]

*Abstract*— **In this paper, we present a new approach for efficient regression-based object tracking. Our approach is closely related to the Generic Object Tracking Using Regression Networks (GOTURN) framework of Held *et al*. [1]. We make the following contributions. First, we demonstrate that there is a theoretical relationship between siamese regression networks like GOTURN and the classical Inverse Compositional Lucas & Kanade (IC-LK) algorithm. Further, we demonstrate that unlike GOTURN IC-LK adapts its regressor to the appearance of the current tracked frame. We argue that the lack of such property in GOTURN attributes to its poor performance on unseen objects and/or viewpoints. Second, we propose a novel framework for object tracking inspired by the IC-LK framework, which we refer to as Deep-LK. Finally, we show impressive results demonstrating that Deep-LK substantially outperforms GOTURN and demonstrate comparable tracking performance against current state-of-the-art deep trackers on high frame-rate sequences whilst being an order of magnitude (100 FPS) computationally efficient.**

## I. Introduction

Regression-based strategies to object tracking have long held appeal from a computational standpoint. Specifically, they apply a computationally efficient regression function to the current source image to predict the geometric transformation between frames. As noted by Held *et al*. [1], most state-of-the-art trackers in vision adopt classification-based approaches, classifying many image patches to find the target object. Notable examples in this regard are correlation filter methods [2], [3], [4], [5], [6], and more recent extensions based on deep learning [7], [8], [9] can be also considered as adhering to this classification-based paradigm to tracking.

Recently, Held *et al*. [1] proposed the Generic Object Tracking Using Regression Networks (GOTURN) framework for object tracking. As it is a regression-based approach, it can be made to operate extremely efficiently – 100 frames per second (FPS) on a high-end GPU. This is in stark contrast to the latest classification-based deep networks for tracking, *e.g*. MDNet [8] and ECO [6] with reported tracking speeds of less than 10 FPS using a GPU. On the other hand, classification-based methods to object tracking that do not rely on deep learning, such as correlation filters, are highly competitive with GOTURN in terms of computational efficiency. However, they often suffer from poor performance as they do not take advantage of the large number of videos that are readily available to improve their performance. Finally, regression-based strategies hold the promise of being able to efficiently track objects with more sophisticated geometric deformations than just translation (*e.g*. affine, homography, thin-plate spline, *etc*.) – something that is not computationally feasible with classification-based approaches.

Although showing much promise, GOTURN has a fundamental drawback. As noted by Held *et al*. [1], the approach performs admirably on objects that have similar appearances to those seen during training. However, when attempting to apply GOTURN to objects or viewpoints that were not presented during training, the approach dramatically fails. This type of poor generalization is not observed in classification-based deep networks such as MDNet [8]. At first glance, this is surprising as GOTURN employs a siamese network which predicts the relative geometric deformation between frames. This network architecture, along with the employment of tracking datasets with large amounts of object variation, should in principle overcome this generalization issue. It is this discrepancy between theory and practice that is central to the discussions of this paper.

We attempt to explain this discrepancy by looking back at a classical algorithm in computer vision literature, namely the Lucas & Kanade (LK) algorithm [10]. Specifically, we demonstrate that Inverse Compositional LK (IC-LK) [11] can be re-interpreted as a siamese regression network that shares many similar properties to GOTURN. We note, however, an important difference that drives the central thesis of our paper – within the IC-LK algorithm, the regression function adapts to the appearance of the previously tracked frame. In GOTURN, their regression function is "frozen" and not adaptable to new images, partially explaining why it performs poorly on previously unseen objects and/or viewpoints.

While IC-LK enjoys the benefit of being adaptable and efficient, it comes at the price of assuming an approximated linear relationship between the image appearance and the geometric displacement. In the concurrent work of Chang *et al*. [12], this assumption is relaxed for planar alignment by optimizing features. However, real-life videos are not limited to predefined planar warp model and come with more complicated variations, *e.g*. occlusion, appearance change, articulated deformation and out-of-plane rotation. In this work, we demonstrate that through end-to-end learning deep features on real sequences, this gap between theory and reality can be alleviated.

Based on the above insights, we propose a new algorithm for efficient object tracking, which we refer to as Deep-LK, that enjoys the same computational advantages of GOTURN without the poor generalization. When evaluated upon standard benchmarks such as VOT14 [13], our Deep-LK method achieves very competitive results to state-of-

[1] All Authors are with Robotics Institute, Carnegie Mellon University, Forbes Avenue 5000, Pittsburgh, PA 15213, USA. chaoyanw@andrew.cmu.edu, hamedkg@gmail.com, chlin@cmu.edu, slucey@cs.cmu.edu.

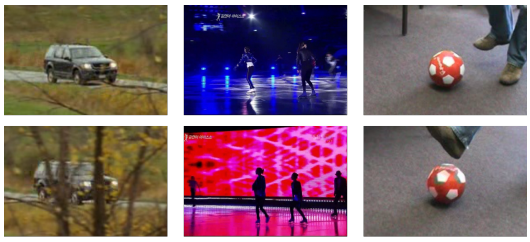


Fig. 1. Three typical types of variations which LK is sensitive to: (a) occlusion; (b) illumination and appearance changes; (c) unmodeled geometric transformation, such as out-of-plane rotation of the ball.

the-art classification-based methods (*e.g.* SiamFC, KCF), and significantly outperforms GOTURN. Furthermore, we demonstrate the superior generalization properites of Deep-LK in comparison to GOTURN. We compare with state-of-the-art trackers on 50 challenging higher frame-rate videos (240 FPS) from the Need for Speed (NfS) [14] tracking dataset. For these higher frame-rate videos, our approach is on par with state-of-the-art classification-based deep trackers while enjoying an order of magnitude higher computational efficiency (100 FPS).

**Notations.** We define our notations throughout as follows: lowercase boldface symbols (*e.g.* $\mathbf{x}$) denote vectors, uppercase boldface symbols (*e.g.* $\mathbf{W}$) denote matrices, and uppercase calligraphic symbols (*e.g.* $\mathcal{I}$) denote images, and we use the notations $\mathcal{I}(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}^K$ to indicate sampling of the $K$-channel image representation at subpixel location $\mathbf{x} = [x, y]^\top$.

## II. The Inverse Compositional LK Algorithm

The core assumption of LK is that the relationship of the image appearance has an approximated linear relationship with the geometric displacements. Given a predefined geometric warp function parametrized by the warp parameters $\mathbf{p}$, this relationship can be written as

$$\mathcal{I}(\Delta\mathbf{p}) \approx \mathcal{I}(\mathbf{0}) + \frac{\partial \mathcal{I}(\mathbf{0})}{\partial \mathbf{p}} \Delta\mathbf{p} \ , \tag{1}$$

which is the first-order Taylor expansion evaluated at the identity warp $\mathbf{p} = \mathbf{0}$.

Given a source image $\mathcal{I}$ and a template image $\mathcal{T}$ to align against, the Inverse Compositional (IC) LK algorithm attempts to find the geometric warp update $\Delta\mathbf{p}$ on $\mathcal{T}$ that could in turn be inversely composed to $\mathcal{I}$ by minimizing the sum of squared differences (SSD) objective

$$\min_{\Delta\mathbf{p}} \|\mathcal{I}(\mathbf{p}) - \mathcal{T}(\Delta\mathbf{p})\|_2^2 \ . \tag{2}$$

IC-LK further utilizes Equation 1 to linearize Equation 2 as

$$\min_{\Delta\mathbf{p}} \|\mathcal{I}(\mathbf{p}) - \mathcal{T}(\mathbf{0}) - \frac{\partial \mathcal{T}(\mathbf{0})}{\partial \mathbf{p}} \Delta\mathbf{p}\|_2^2 \ . \tag{3}$$

Solving for Equation 3, we get the IC-LK update equation:

$$\Delta\mathbf{p} = \mathbf{W}^\dagger [\mathcal{I}(\mathbf{p}) - \mathcal{T}(\mathbf{0})]. \tag{4}$$

where $\dagger$ is the Moore-Penrose pseudo-inverse operator. $\mathbf{W}$ is the so-called template image Jacobian, factorized as

$$\frac{\partial \mathcal{T}(\mathbf{0})}{\partial \mathbf{p}} = \begin{bmatrix} \nabla\mathcal{T}_{\mathbf{x}_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \nabla\mathcal{T}_{\mathbf{x}_D} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{W}(\mathbf{x}_1;\mathbf{0})}{\partial \mathbf{p}^T} \\ \vdots \\ \frac{\partial \mathcal{W}(\mathbf{x}_D;\mathbf{0})}{\partial \mathbf{p}^T} \end{bmatrix}, \tag{5}$$

where $\nabla\mathcal{T}_{\mathbf{x}_i}$ are the image gradients at pixle locations $\mathbf{x}_i$ which are solely a function of $\mathcal{T}(\mathbf{0})$, and $\frac{\partial \mathcal{W}(\mathbf{x}_i;\mathbf{0})}{\partial \mathbf{p}}$ is the predefined warp Jacobian. The warp parameter $\mathbf{p}$ are updated by:

$$\mathbf{p} \leftarrow \mathbf{p} \circ^{-1} \Delta\mathbf{p}, \tag{6}$$

where we use the notation $\circ^{-1}$ to denote the inverse composition of the warp functions parametrized by $\mathbf{p}$, and $\Delta\mathbf{p}$. Since the true relationship between appearance and geometric deformation is not linear in most cases, Equations 4 and 6 need to be applied iteratively until convergence.

The advantage of the IC form of LK over the original lies in its efficiency. Specifically, $\mathbf{W}^\dagger$ is evaluated on the template image $\mathcal{T}$ at the identity warp $\mathbf{p} = \mathbf{0}$; therefore, $\mathbf{W}^\dagger$ remains constant throughout as long as the template image remains static. We refer the readers to [11] for a more detailed treatment.

Recently, there has been a number of works that aims to increase the robustness of the classical LK framework by minimizing feature differences extracted from the images [15], [16], where the objective becomes

$$\min_{\Delta\mathbf{p}} \|\phi(\mathcal{I}(\mathbf{p})) - \phi(\mathcal{T}(\Delta\mathbf{p}))\|_2^2 \ , \tag{7}$$

where $\phi(\cdot)$ is a feature extraction function. Following similar derivations, the geometric warp update $\Delta\mathbf{p}$ in Equation 4 becomes

$$\Delta\mathbf{p} = \mathbf{W}^\dagger [\phi(\mathcal{I}(\mathbf{p})) - \phi(\mathcal{T}(\mathbf{0}))] \ , \tag{8}$$

Similarly, $\mathbf{W} = \frac{\partial \phi(\mathcal{T}(\mathbf{0}))}{\partial \mathbf{p}}$ is also a sole function of $\phi(\mathcal{T}(\mathbf{0}))$; similar to the original IC-LK, $\mathbf{W}^\dagger$ is fixed throughout the iterative update, since $\phi(\mathcal{T}(\mathbf{0}))$ only needs to be evaluated once at the identity warp.

### A. Cascaded Linear Regression

Equation 8 can be written in a more generic form of linear regression as

$$\Delta\mathbf{p} = \mathbf{R} \cdot \phi(\mathcal{I}(\mathbf{p})) + \mathbf{b}, \tag{9}$$

where $\mathbf{R} = \mathbf{W}^\dagger$ and $\mathbf{b} = -\mathbf{W}^\dagger \cdot \phi(\mathcal{T}(\mathbf{0}))$ are the regression matrix and the bias term respectively. Therefore, the IC-LK algorithm belongs to the class of cascaded linear regression methods, as the warp update is iteratively solved for and applied until convergence.

Cascaded regression has been widely used in pose estimation and face alignment. Compared to directly solving the problem with a single sophisticated model (*e.g.* a deep neural network), cascaded regression can usually achieve comparable results with a sequence of much simpler models [17], [18], [19], [20]. This is desirable to visual tracking, as simpler models in most cases are computationally more efficient and thus offer faster speed.

### B. Connection to Siamese Regression Networks

Siamese regression networks for tracking predict the geometric warp from concatenated features of the source and template images together. This can be mathematically expressed as

$$\Delta\mathbf{p} = f(\begin{bmatrix}\phi(\mathcal{I}(\mathbf{p}))\\ \phi(\mathcal{T}(\mathbf{0}))\end{bmatrix}) \ , \tag{10}$$

where $f(\cdot)$ denotes a nonlinear prediction model. In the case of GOTURN [1], $\phi(\cdot)$ is the convolutional features and $f(\cdot)$ is a multi-layer perceptron trained through backpropagation.

We can also write the IC-LK formulation in Equation 8 in the form of Equation 10 as

$$\Delta\mathbf{p} = \begin{bmatrix}\mathbf{W}^{\dagger} & -\mathbf{W}^{\dagger}\end{bmatrix}\begin{bmatrix}\phi(\mathcal{I}(\mathbf{p}))\\ \phi(\mathcal{T}(\mathbf{0}))\end{bmatrix}. \tag{11}$$

This insight elegantly links the IC-LK algorithm with GOTURN: while GOTURN models the non-linear prediction function $f$ with a heavily-parametrized multi-layer perceptron, IC-LK models $f$ through cascaded linear regression. In addition, $\mathbf{W}^{\dagger}$ is a sole function of $\phi(\mathcal{T}(\mathbf{0}))$ and contains no additional learnable parameters.

In comparison to GOTURN, whose weights are learned offline and kept static during testing, the fully-connected layer of IC-LK is formed directly from the template image. In some sense, this is analogous to one-shot learning, since the weights are "learned" from one single example $\mathcal{T}(\mathbf{0})$. This is a very desirable property for tracking, as it allows the tracker to cheaply adapt its parameters in real-time, and as a result, generalize better to unseen objects.

### C. Limitations

Although having these desired properties, prior works on LK have the following limitations (illustrated in Fig. 1): they are (a) sensitive to occlusion, (b) sensitive to appearance variations, (c) intolerant to geometric transformations not modeled by the warp Jacobian (*e.g.* out-of-plane rotation and pose changes of articulated objects). In the scenario of object tracking in a real-world sequence, this cannot be handled simply by learning from synthetically created examples from a single image [12]. All these limitations severely restrict the application of LK framework in real world settings.

For the past three decades, there have been numerous attempts to loosen the above restrictions, most of which focused on dealing with the photometric variation. Those techniques includes: estimating illumination parameters [21], [22], using intrinsically robust similarity measures (*e.g.* mutual information [23], [24] or normalized correlation [25]), preprocessing images [26], [16], [27], or using hand-crafted features (*e.g.* HOG [28], SIFT [29], or LBP [30], [15]) to obtain features more robust to intensity variations. However, none of them is robust against all these variations or shows competitive results in object tracking.

Following the recent trend of deep learning, we propose to improve the robustness of LK by learning a deep feature representation specific to the LK framework. With the learned representation, our Deep-LK tracker is the first in the LK family to achieve competitive results on challenging object tracking datasets and significantly outperforms GOTURN, the state-of-the-art regression-based tracker.

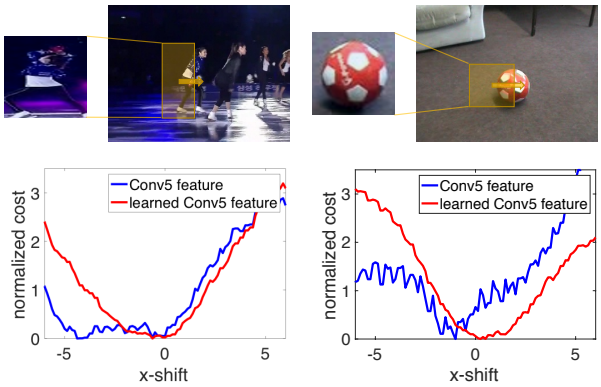


Fig. 2. SSD Cost curves of comparing template images to source images along horizontal shift with vanilla AlexNet's conv5 feature (blue) and Deep-LK's learned feature (red). The curves of learned features not only have correct global minimum (at 0 shift), but also exhibit less local minima, and encourages smoother convergence. This characteristic is desirable in the LK framework.

## III. Deep-LK Regressor

The validity of LK algorithm is based on the consistency of the linear relationship between the pixel intensity/feature value and warp parameter. Thus to improve the robustness of LK against the aforementioned variations in videos, the idea is to learn a deep convolutional network that projects the template and source images to a feature space where this linear relationship is best approximated.

Deep convolutional features pretrained on large-scale datasets (*e.g.* ImageNet [31]) have shown great generalizability to a variety of tasks and demonstrated robustness to a wide range of appearance variations. However, we show in Fig. 2 that vanilla pretrained convolutional features have great room of improvement in a tracking scenario. By integrating LK into the end-to-end learning framework, the cost curves of learned features become smoother with a more apparent local minimum at the identity displacement. These cost curves have a near-quadratic shape, which is also more ideal for Gauss-Newton optimization in the LK framework.

### A. Network Architecture

Fig. 3 summarizes the network architecture of Deep-LK. Given an input source image $\mathcal{I}$, we extract deep convolutional features $\phi(\cdot)$ and apply a single linear regression layer in the feature space to predict the warp update. The linear regression layer is formulated as:

$$\Delta\tilde{\mathbf{p}} = \mathbf{R}_{\phi(\mathcal{T})} \cdot \phi(\mathcal{I}(\mathbf{p})) + \mathbf{b}_{\phi(\mathcal{T})}. \tag{12}$$

We abbreviate $\mathcal{T}(\mathbf{0})$ as $\mathcal{T}$ to simplify notations.

Similar to IC-LK, $\mathbf{R}_{\phi}$ and $\mathbf{b}_{\phi}$ are formed through

$$\mathbf{R}_{\phi(\mathcal{T})} = \mathbf{W}^{\dagger}_{\phi(\mathcal{T})} \tag{13}$$

$$\mathbf{b}_{\phi(\mathcal{T})} = -\mathbf{R}_{\phi(\mathcal{T})} \cdot \phi(\mathcal{T}) \ , \tag{14}$$

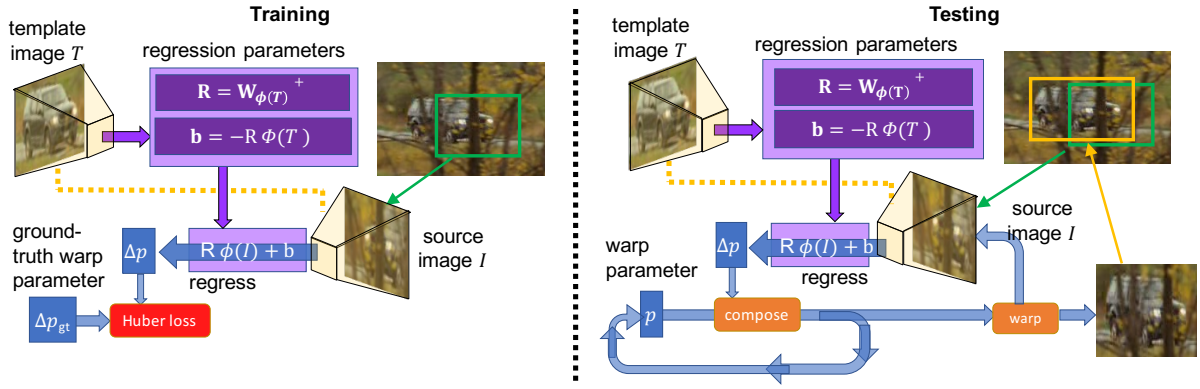


Fig. 3. Left: we train the feature representation $\phi$ such that the warp parameter prediction is close to the ground truth. Right: during test time, given a pair of template image and a source image, Deep-LK precomputes the feature for the template image and forms the regression matrix $\mathbf{R}$ and bias term $\mathbf{b}$. Then Deep-LK repeats the process of computing features for the current source image, estimating the warp by regression, and warping the source image.

where $\mathbf{W}_{\phi(\mathcal{T})} = \frac{\partial \phi(\mathcal{T})}{\partial \mathbf{p}}$ is computed in the same way as IC-LK. In contrast to conventional linear layers that are learnable itself, our regression matrix and bias pair $(\mathbf{R}_{\phi(\mathcal{T})}, \mathbf{b}_{\phi(\mathcal{T})})$ is dependent on the template image $\mathcal{T}$ and the feature extraction function that follows. Unlike IC-LK, on the other hand, the adopted feature $\phi(\cdot)$ is *learnable* instead of being hand-crafted like SIFT or HOG.

### B. The Conditional LK Loss

There are several ways of formulating the learning objective to enforce the consistency of the linear relationship between feature values and warp parameters. One option is to use a *generative loss*, which directly penalizes the difference between the source image and the linear approximation of the template image with a ground truth warp parameter. Another option is to train the feature representation such that the warp parameter prediction is close to the ground truth. We refer to this objective as the *conditional LK loss*.

It has been shown [32] that optimizing in terms of a conditional objective usually leads to better prediction than a generative objective. In our primary experiments, we also confirm that in our case, using conditional LK loss leads to far better tracking accuracy. This finding is in line with the benefits of an end-to-end training paradigm in the deep learning community.

Formally, the conditional LK loss is defined as:

$$\sum_{n \in \mathcal{S}} \mathcal{L}\left(\mathbf{R}_{\phi(\mathcal{T}^{(n)})} \cdot \phi(\mathcal{I}^{(n)}(\mathbf{p})) + \mathbf{b}_{\phi(\mathcal{T}^{(n)})} - \Delta \mathbf{p}_{gt}^{(n)}\right), \tag{15}$$

where $\mathcal{S}$ is the index set of all training data, $\mathcal{L}$ is the loss function, and $\mathbf{p}_{gt}$ is the ground truth warp parameter. This is optimized over the parameters of the feature extraction function $\phi(\cdot)$. We choose to use the Huber loss for optimization as it is less sensitive to outliers compared to SSD.

The conditional LK loss is differentiable and can be incorporated into a deep learning framework trained by back-propagation. Please refer to the supplementary materials for a detailed mathematical derivation.

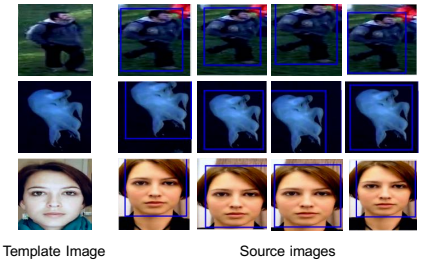


Fig. 4. Source and template image pairs for training.

### C. Training/Online Adaptation

*1) Training:* We train Deep-LK with pairs of source and template images sampled from video sequences. First, a template image is generated for each frame by cropping an image region from an object bounding box. For each template image, we randomly sample a set of source images from the next frame. The source images are sampled by randomly perturbing the bounding box to mimic small motion changes between frames.

We follow the practice in GOTURN, which assumes the motion between frames follows a Laplace distribution. The scale changes are modeled as a Laplace distribution with zero mean and scale $b_s = 1/30$; The translations of bounding box are sampled using $b_x = 0.06$. In addition, we further augment the training set by adding random brightness, contrast and saturation shift to both the template and source images.

Unlike GOTURN, we do not use synthetically perturbed images from ImageNet to augment the training set. While all the assumptions for LK (*e.g.* photometric consistency and the geometric deformation model) hold in such ideal scenarios, it offers little help for improving the feature robustness in real video sequences, where appearance/geometric variations can be more complex.

*2) Online adaptation:* When tested on video, we follow the common practice in correlation filter trackers, which uses a simple template adaptation strategy:

$$\hat{\phi_{\mathcal{T}}}^{t} = (1-\alpha) \cdot \hat{\phi_{\mathcal{T}}}^{t-1} + \alpha \cdot \phi(\mathcal{I}(\mathbf{p}^{t-1})), \tag{16}$$

where $\hat{\phi_{\mathcal{T}}}^{t}$ represents the template feature used to generate linear regression parameters at frame $t$, and $\alpha$ is the learning rate. The computation cost of this online adaptation strategy is cheap, and it effectively improves the robustness to pose, scale and illumination changes.

## IV. Experiment

### A. Implementation Details

In this work, we use the conv5 feature of AlexNet [33] as the deep convolutional feature extraction function $\phi$, which maps an RGB image of size $227 \times 227$ to a feature map of size $13 \times 13$ with 256 filter channels.

We collect our training samples from ALOV300++ [34], which consists of 307 videos (with 7 videos removed because of overlapping with VOT14 [13]). Limited by the number of videos available, we only fine-tune the last convolutional layer to avoid over-fitting.

Although Deep-LK is trained to be able to deal with both translation and scale estimation, in test time we find that it works best in practice to update translation first with scale fixed and then update both together. In addition, we adopted an early stopping mechanism in the LK updates to avoid tracking failure caused by unreliable scale estimation: in each iteration, we check if the squared distance between the source image and template image is decreasing, otherwise, we stop the update and report the resulting warp parameter.

We empirically set the learning rates $\alpha = 0.03$ for 30-FPS videos; for 240-FPS videos, we adjust $\alpha$ to be inversely proportional to the video frame rate [14], which is $0.0037$.

### B. Experiment Setup

We evaluated the proposed Deep-LK tracker over four sets of sequences, including the VOT14/VOT16 dataset [13], [35], a set of 50 higher frame-rate videos borrowed from the NfS dataset [14], and a small set of 15 videos captured by a higher frame-rate (240 FPS) unmanned aerial vehicle (UAV) camera from ground objects.

The sequences from the NfS dataset include generic objects in real-world scenarios captured by 240-FPS smartphone cameras (*e.g.* iPhone 6). The UAV videos allow us to evaluate the generalization capacity of our tracker on unseen viewpoints and objects. The reason we tested the Deep-LK on videos with different capture rates (30-FPS in VOT14, 240-FPS in NfS and UAV videos) is to explore the robustness of our method for both lower and higher frame-rate videos. We compared our proposed algorithm against several state-of-the-art object trackers including deep trackers (SiamFC [7], GOTURN [1], MDNet [8] and FCNT [9]), CF trackers with hand-crafted features (SRDCF [2], Staple [5], DSST [36], SAMF [37], LCT [38], KCF [3] and CFLB [4]), and CF trackers with deep features (HCF [39] and HDT [40]). We do not report results on other public datasets, like OTB [41] because our training set from ALOV++ overlaps with their test videos.

*Evaluation Methodology:* We use the success metric to evaluate all trackers [41]. Success measures the intersection over union (IoU) of predicted and ground truth bounding boxes. The success plot shows the percentage of bounding boxes whose IoU is larger than a given threshold. We use the area under curve (AUC) of success plots to rank the trackers. We also compare all the trackers by the success rate at the conventional threshold of 0.50 (IoU $> 0.50$) [41].

### C. Feature Evaluation

The first experiment evaluates the effect of learned features on Deep-LK's tracking accuracy. To do so, we ran the Deep-LK tracker using two different sets of deep features, including features we learn over the conditional LK loss and those directly borrowed from AlexNet (pretrained on ImageNet). We also explored the effect of utilizing features from different layers (*e.g.* conv3, conv4 and conv5).

The result on VOT14 sequences is presented in Fig. 5. First, compared to AlexNet features, features optimized from the conditional LK loss offer much higher accuracy and robustness. Second, conv5 outperforms conv3 and conv4. We also provide attribute-based comparison of such features in Fig. 6. Similarly, this evaluation shows that features learned from the conditional LK loss outperform those from AlexNet for all attributes. Moreover, among different feature layers learned from the conditional LK loss, conv5 features in general are more robust in challenging situations such as large camera motion, occlusion and scale changes. Based on this observation, we choose to use the conv5 learned features in our Deep-LK framework.

### D. Evaluation on VOT14 and VOT16

Fig. 8 illustrates the evaluation of Deep-LK on 25 challenging videos of the VOT14 dataset and 60 videos of VOT16(Fig. 8). Following the VOT14 protocol [13], results are reported by means of accuracy and robustness. The comparison shows the superior accuracy and robustness of our method to the leading methods in the VOT14 challenge, such as DSST, KCF, SAMF and GOTURN.

Moreover, our method outperformed the correlation filter tracker KCF and the deep tracker GOTURN, showing the advantage of our method from three different perspectives. First, unlike KCF which only relies on online adaptation of hand-crafted features, Deep-LK adapts the regressor using discriminative deep features. This offers a well-generalized tracker which is more robust against challenging circumstances such as deformation and scaling. Second, compared to SiamFC [7] which also uses offline trained deep features, Deep-LK achieves comparable performance without the need to do exhaustive search through space and scale. Finally, compared to GOTURN, which is the only compared deep regression-based tracker (see Fig. 7 for a finer-grained comparison), Deep-LK delivers much more robust tracking due to its crucial ability to update the regressor with new frames. This not only improves the generalization of this tracker to unseen objects, but also increases its robustness to geometric and photometric variations between two consecutive frames.

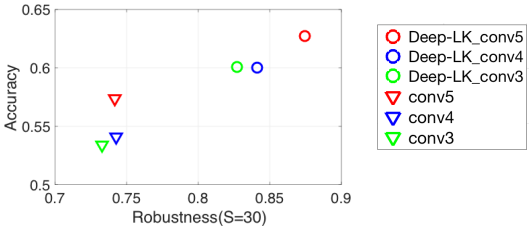


Fig. 5. Accuracy and robustness plot on VOT14 dataset, evaluating Deep-LK tracking performance using different layers of learned and AlexNet's features.

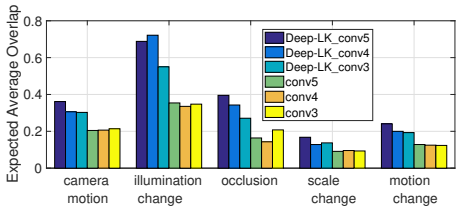


Fig. 6. Comparing the effect of AlexNet features with those learned over the conditional LK loss on tracking performance of Deep-LK. Results are reported by the means of average overlap for conv3, conv4 and conv5 layers, on 5 attributes of VOT14.

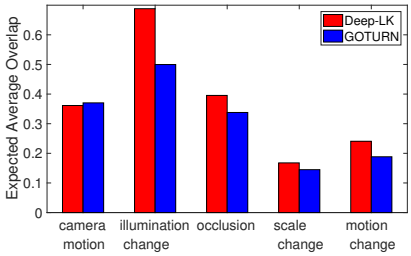


Fig. 7. Expected Average Overlap scores for each attribute on VOT14. Comparing our Deep-LK with GOTURN.

We note that compared to Deep-LK, MDNet and CCOT achieve better robustness and accuracy rank; however, these trackers suffer from very expensive computation (less than 10 FPS on a high-end GPU), while Deep-LK runs almost an order of magnitude faster on lower frame-rate sequences of VOT (up to 75 FPS).

### *E. Evaluation on NfS Sequences*

The results of this experiment are shown in Table I and Fig. 9, comparing Deep-LK with the state-of-the-art CF and deep trackers on 50 NfS videos. These sequences are captured by 240-FPS cameras and thus exhibit less appearance changes between two consecutive frames [14]. This is desirable characteristic for Deep-LK which directly regresses from current frame to the next one with much less appearance changes.

As summarized in Table I, our method achieves the highest AUC (51.2), closely followed by MDNet (50.9) and SRDCF (50.7). This result shows that on higher frame-rate videos, our regression-based tracker can perform as accurate as such classification/detection-based methods. Deep-LK also demonstrates a significant improvement over GOTURN ($\sim$10 %). This is mainly because of Deep-LK's ability to adapt its regressor to the appearance of the currently tracker frame, serving as a crucial key of outperforming GOTURN in adapting to unseen scenarios.

In terms of tracking speed on CPUs, Deep-LK is faster than all other deep trackers as well as several CF trackers (SRDCF, LCT, DSST, HCF and HDT). On GPUs, however, GOTURN showed a tracking speed of 155 FPS, which is 50% faster than Deep-LK on the same GPU. During this experiment, we observed that Deep-LK performs much faster on higher frame-rate videos. The reason is that since inter-frame difference in higher frame-rate videos is much less than lower frame-rate ones, Deep-LK requires less iterations to converge. This offers a tracking speed of 100 FPS on GPUs, which is 25% faster than tracking lower frame rate sequences of VOT14 (75 FPS).

Fig. 11 (a) visualizes tracking performance of Deep-LK with SRDCF, Staple, MDNet and GOTURN on three different scenarios. This qualitative result demonstrates the robustness of our method against severe scale changes, out-plane-rotation, occlusion and non-rigid deformation.

### *F. Robustness to Unseen Objects and Viewpoints*

The goal of this experiment is to evaluate the robustness of Deep-LK to unseen objects and viewpoints, compared to GOTURN (as the baseline). For this purpose, we captured 15 challenging higher frame-rate videos (240 FPS) using an UAV (drone) camera. These videos contain $\sim$100K frames showing real-world scenarios of different objects such as drones, cars, trucks, boats and humans (biking, running, walking, or playing basketball).

The results in Fig. 10 demonstrate the notable accuracy of our method (60.38%) compared to GOTURN (48.13%). As mentioned earlier, the robustness of Deep-LK comes from its ability to adapt its regressor to the appearance of current frame. This, however, is not the case in GOTURN. Its regression function is frozen and thus performs poorly on unseen targets and viewpoints. Fig. 11 (b) visualizes the poor generalization of GOTURN to the unseen aerial viewpoint (of a truck) and object (drone). It also shows that Deep-LK generalizes well to unseen viewpoint/object with challenging appearance variation.

### *G. Speed Analysis of Deep-LK*

As detailed in Sec. IV-A, Deep-LK updates iteratively and stops when either it converges or meets other stopping criteria. Therefore, its tracking speed is highly dependent on 1) the capture frame rate of the video, and 2) the motion of the target object. To analyze the speed of Deep-LK, we break down the computational cost of each operation within the Deep-LK framework.

On GPUs, it takes 1.4ms to form the regression parameters from a template image and approximately 3ms to run an LK iteration (0.5ms for image cropping and resizing, 2.2 $\sim$ 2.5ms to compute conv5 features and around 0.15ms

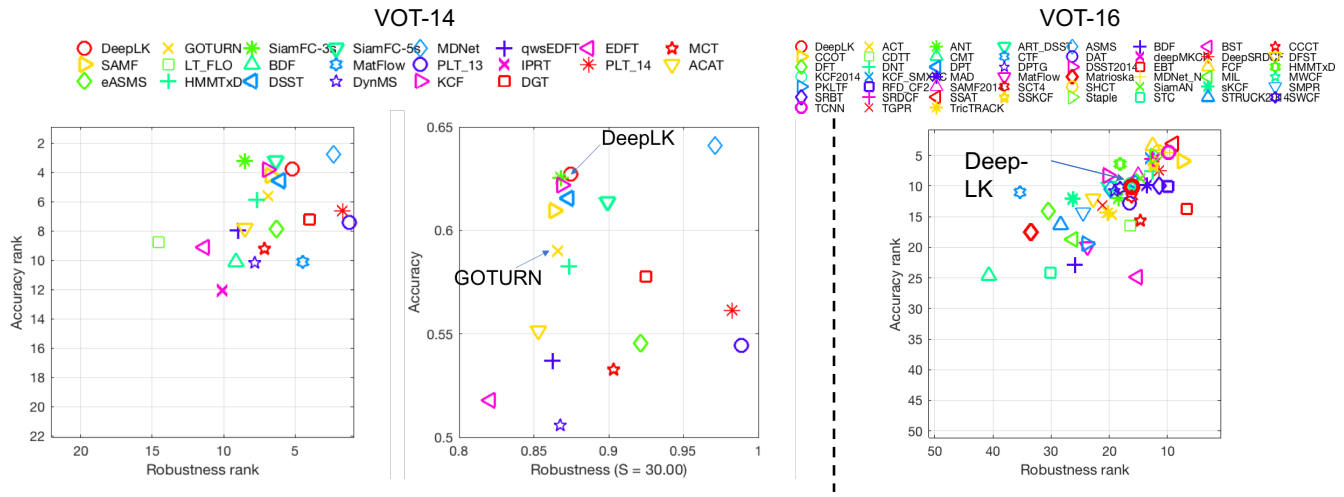


Fig. 8. Results on VOT14(left) and VOT16(right), comparing Deep-LK with the leading methods in the VOT challenges. Best trackers are closer to the top right corner .

TABLE I

COMPARING DEEP-LK WITH THE RECENT CF TRACKERS ON 50 HIGHER FRAME-RATE SEQUENCES- FROM NFS. RESULTS ARE REPORTED AS AUCS OF THE SUCCESS PLOT. TRACKING SPEED IS REPORTED IN FRAMES PER SECOND (FPS) ON THE CPU AND/OR GPU IF APPLICABLE.

| | Deep-LK | SRDCF | Staple | LCT | DSST | SAMF | KCF | CFLB | HCF | HDT | MDNet | SiamFC | FCNT | GOTURN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AUC | 51.2 | 50.7 | 45.4 | 37.9 | 48.1 | 47.8 | 36.9 | 22.5 | 41.3 | 50.4 | 50.9 | 49.2 | 50.0 | 40.8 |
| Speed (CPU) | 20.7 | 3.8 | 50.8 | 10 | 12.5 | 16.6 | 170.4 | 85.5 | 10.8 | 9.7 | 0.7 | 2.5 | 3.2 | 3.9 |
| Speed (GPU) | 100 | - | - | - | - | - | - | - | - | 43.1 | 2.6 | 48.2 | 51.8 | 155.3 |

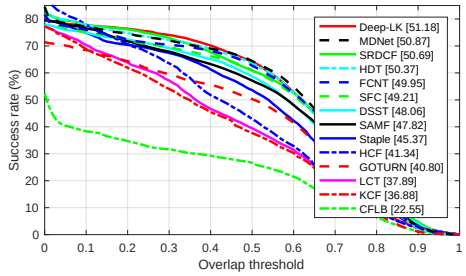


Fig. 9. Comparing Deep-LK with the state of the art CF and deep trackers on 50 NfS videos. AUCs are reported in the legend.

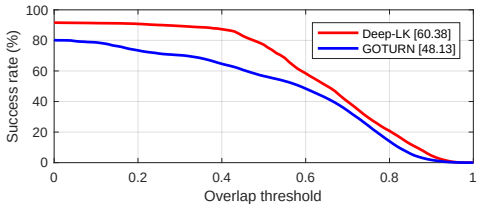


Fig. 10. Comparison Deep-LK with GOTURN on 15 UAV sequences. AUCs are reported in the legend.

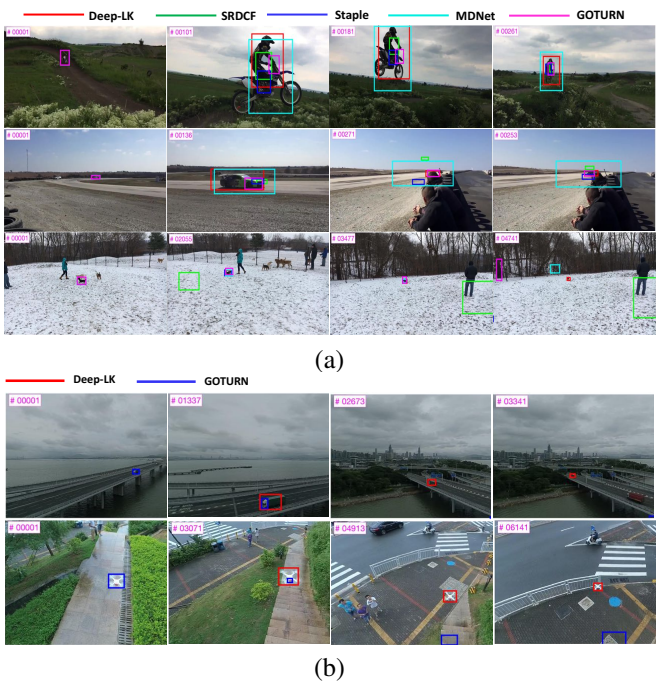


Fig. 11. Qualitative results. (a) shows tracking performance of Deep-LK, SRDCF, Staple, MDNet and GOTURN on 4 higher frame-rate videos selected from NfS. Our method is robust against scaling, non-rigid deformation, out-of-plane rotation and occlusion. (b) comparing the robustness of our method to unseen object and viewpoint compared to GOTURN.

to perform regression). On VOT14 with lower frame rate sequences, Deep-LK on average converges within 4 iterations per frame, hence our average speed is about 75 FPS on GPU. On NfS (and UAV) higher frame-rate sequences, since there is much less inter-frame appearance changes, Deep-LK converges much faster over 2 $\sim$ 3 iterations per frame. This

offers a tracking speed of 100 FPS on the same GPU.

On CPUs, on the other hand, Deep-LK requires relatively longer time to compute conv5 features and update the regressor. As a result, it performs $\sim$15 FPS on lower frame rate videos of VOT14, and $\sim$20 FPS on higher frame-rate videos of NfS and UAV data.

## V. Conclusion

We propose a novel regression-based object tracking framework, which successfully incorporates Lucas & Kanade algorithm into an end-to-end deep learning paradigm. We conclude that the combination of offline trained feature representation and the efficient online adaptation of regression parameters respect to template images, are crucial advantages of our method to generalize well against real-world siturations such as severe scaling, deformation and unseen objects/viewpoints. Compared to the state-of-the-art regression-based deep tracker, GOTURN, our Deep-LK shows impressive robustness to unseen objects and viewpoints. Moreover, we demonstrate that on higher frame-rate sequences, Deep-LK offers comparable tracking performance to current state-of-the-art trackers including both correlation filter and deep framework trackers.

## References


[1] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *ECCV*, 2016.

[2] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Learning spatially regularized correlation filters for visual tracking," in *ICCV*, 2015, pp. 4310–4318.

[3] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *PAMI*, vol. 37, no. 3, pp. 583–596, 2015.

[4] H. Kiani Galoogahi, T. Sim, and S. Lucey, "Correlation filters with limited boundaries," in *CVPR*, 2015, pp. 4630–4638.

[5] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, "Staple: Complementary learners for real-time tracking," in *CVPR*, June 2016.

[6] M. Danelljan, G. Bhat, F. Shahbaz Khan, and M. Felsberg, "Eco: Efficient convolution operators for tracking," in *CVPR*, 2017.

[7] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *ECCV Workshop*, 2016, pp. 850–865.

[8] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *CVPR*, June 2016.

[9] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *ICCV*, 2015, pp. 3119–3127.

[10] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (darpa)," in *Proceedings of the 1981 DARPA Image Understanding Workshop*, April 1981, pp. 121–130.

[11] S. Baker and I. A. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[12] C.-H. Chang, C.-N. Chou, and E. Y. Chang, "Clkn: Cascaded lucas-kanade networks for image alignment," in *CVPR*, July 2017.

[13] K. Matej, M. Kristan, R. Pftugfelder, and A. Leonardis, "The visual object tracking vot 2014 challenge results," 2014.

[14] H. K. Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," *ICCV*, 2017.

[15] H. Alismail, B. Browning, and S. Lucey, "Bit-planes: Dense subpixel alignment of binary descriptors," *arXiv preprint arXiv:1602.00307*, 2016.

[16] E. Antonakos, J. Alabort-i Medina, G. Tzimiropoulos, and S. P. Zafeiriou, "Feature-based lucas–kanade and active appearance models," *IEEE Transactions on Image Processing*, vol. 24, no. 9, pp. 2617–2632, 2015.

[17] P. Dollár, P. Welinder, and P. Perona, "Cascaded pose regression," in *CVPR*. IEEE, 2010, pp. 1078–1085.

[18] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *CVPR*, June 2013.

[19] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 824–832.

[20] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *IJCV*, vol. 107, no. 2, pp. 177–190, 2014.

[21] A. Bartoli, "Groupwise geometric and photometric direct image registration," *PAMI*, vol. 30, no. 12, pp. 2098–2108, 2008.

[22] G. Tzimiropoulos, J. Alabort-i-Medina, S. Zafeiriou, and M. Pantic, "Generic active appearance models revisited," in *ACCV*, 2012, pp. 650–663.

[23] N. Dowson and R. Bowden, "Mutual information for lucas-kanade tracking (milk): An inverse compositional formulation," *PAMI*, vol. 30, no. 1, pp. 180–185, 2008.

[24] A. Dame and E. Marchand, "Accurate real-time tracking using mutual information," in *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*. IEEE, 2010, pp. 47–56.

[25] M. Irani and P. Anandan, "Robust multi-sensor image alignment," in *ICCV*, Jan 1998, pp. 959–966.

[26] S. Lucey, R. Navarathna, A. B. Ashraf, and S. Sridharan, "Fourier lucas-kanade algorithm," *PAMI*, vol. 35, no. 6, pp. 1383–1396, 2013.

[27] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, "An improved algorithm for tv-l 1 optical flow," in *Statistical and Geometrical Approaches to Visual Motion Analysis*. Springer, 2009, pp. 23–45.

[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1, 2005, pp. 886–893.

[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[30] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[32] C.-H. Lin, R. Zhu, and S. Lucey, "The conditional lucas & kanade algorithm," in *ECCV*. Springer, 2016, pp. 793–808.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[34] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *PAMI*, vol. 36, no. 7, pp. 1442–1468, 2014.

[35] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernández, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder, "The visual object tracking vot2016 challenge results," in *ICCV Workshop*, 2016.

[36] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *BMVC*, 2014.

[37] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *ECCV*, 2014, pp. 254–265.

[38] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *CVPR*, 2015, pp. 5388–5396.

[39] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *CVPR*, 2015, pp. 3074–3082.

[40] Y. Qi, S. Zhang, L. Qin, H. Yao, Q. Huang, J. Lim, and M.-H. Yang, "Hedged deep tracking," in *CVPR*, 2016, pp. 4303–4311.

[41] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *CVPR*, 2013, pp. 2411–2418.