Lie-Struck: Affine Tracking on Lie Groups using Structured SVM

Gao Zhu[1], Fatih Porikli[1,2], Yansheng Ming[1], Hongdong Li[1,3]
Australian National University[1] and NICTA[2]
ARC Centre of Excellence for Robotic Vision[3]
{gao.zhu,fatih.porikli,yansheng.ming,hongdong.li}@anu.edu.au

Abstract

This paper presents a novel and reliable tracking-by-detection method for image regions that undergo affine transformations such as translation, rotation, scale, dilatation and shear deformations, which span the six degrees of freedom of motion. Our method takes advantage of the intrinsic Lie group structure of the 2D affine motion matrices and imposes this motion structure on a kernelized structured output SVM classifier that provides an appearance based prediction function to directly estimate the object transformation between frames using geodesic distances on manifolds unlike the existing methods proceeding by linearizing the motion. We demonstrate that these combined motion and appearance model structures greatly improve the tracking performance while an incorporated particle filter on the motion hypothesis space keeps the computational load feasible. Experimentally, we show that our algorithm is able to outperform state-of-the-art affine trackers in various scenarios.

1. Introduction

Tracking affine transformations of image regions is essential in many applications from pose estimation to object recognition, and still one of the most challenging tasks in computer vision. In addition to critical problems such as appearance changes, lighting variations, indiscriminate backgrounds and occlusions that arise in tracking translational motion of an image window, tracking affine motion confronts with a higher dimensional parameter space that blows up the computational complexity and non-Euclidean manifold structure of motion matrices that leads into inaccurate distance computations when they are flattened.

Existing methods often attempt to solve the affine motion tracking problem in vector space and can be roughly categorized into state-space estimation [20, 31, 18, 11, 2], template alignment [12, 5, 6, 22] and feature correspondence [23, 32, 17] approaches. State-space estimators assume affine tracking as a Markovian process and construct a probability density function of object parameters, which is a normal distribution in case of Kalman filtering and a multi-modal distribution for particle filtering. In theory, particle filter can track any parametric variation including affine motion. However, its dependency to random sampling induces degenerate likelihood estimations especially for the higher dimensional parameter spaces. In template alignment, parametrized motion models are estimated using appearance and shape models that are usually fitted by nonlinear optimization. One shortcoming of these algorithms is that they require computation of partial derivatives, Jacobian, and Hessian for each iteration, which makes them impractical. Feature point based methods mainly differ in the type of features and descriptors used for matching the object model to the current frame. Their shortcoming is that in many cases only little texture is present on the object.

It is worthwhile to mention that tracking-by-detection, which allows an online trained classifier [3, 14, 27] as an object model to distinguish the object from its surrounding background, has recently become particularly popular. Most tracking-by-detection update the classifier by a set of binary labeled training samples that are obtained using heuristics such as the distance of a sample from the estimated object location. One implication of this is that slight inaccuracy during tracking can lead to poorly labeled samples, thus, tracking failure. Rather than explicitly coupling to the accurate estimation of object position, [4, 21, 28] limit their focus on increasing the robustness to poorly labeled samples. As a remedy, [16] proposed directly predicting the change in object location between frames by an online structured output SVM. Even though [16] produces comparably accurate tracking for translational motion, for affine motion it has two major drawbacks. Since it strictly depends on a bounding box overlap based loss function in its compatibility function, it can not distinguish rotations and complex affine deformations. Besides, it uniformly samples the state space to generate positive and negative support vectors. Such a brute force approach on a high dimensional search space is computationally intractable.

Unlike the prevalent practice, the set of 2D affine transformations do not constitute a vector space, but rather an analytical manifold $\mathcal{M}$ that has the structure of a Lie group $\mathrm{Aff}(\mathbb{R}^2)$. Existing methods for the most part disregard this manifold structure and flatten the topology in a vector space. Vector forms cannot globally parameterize the intrinsic topology on $\mathcal{M}$ in a homogeneous fashion, thus fail to accurately evaluate the distance between affine motion matrices causing unreliable tracking performance. There are only a few relevant work for parameter estimation on Lie groups, e.g. [13] for tracking an affine snake and [7, 30, 19] for tracking a template. However, [7] fails to account for the noncommutativity of the matrix multiplications thus the estimations are valid only around the initial transformation. [30] learned the correlation between affine motions and the observed descriptors using a regression model on Lie algebra. Inherent topology is considered by [19] where a conventional particle filter based tracker where the state dynamics are defined on $\mathcal{M}$ using a log-Euclidean metric. However, none of these methods incorporate an efficient mechanism to incorporate object appearance changes.

To overcome the shortcomings of the existing methods, here we propose a novel affine tracking-by-detection method, Lie-Struck, that takes advantage of the intrinsic topology using a geodesic distance when it compares two motion matrices. Our method incorporates Lie group structure $\mathrm{Aff}(\mathbb{R}^2)$ presented in [30] into a structured output SVM classifier introduced in [16] to directly estimate 2D affine transformation using an appearance based prediction function. Unlike [30], Lie-Struck can efficiently learn object's temporal appearance changes. Unlike [16], our method can accurately track affine transformations. We demonstrate that these combined motion and appearance model structures significantly improve the tracking performance while an incorporated particle filter mechanism keeps the computational complexity minimal. Experimentally, we show that our method consistently outperforms the state-of-the-art trackers on various scenarios even when the object undergoes challenging aspects such as occlusion and motion blur.

We also introduce a manually annotated affine tracking dataset since most existing datasets have only rectangle ground truth regions, thus are not suitable for performance evaluation of affine trackers.

2. Lie-Struck Formulation

2D affine motion matrices constitute Lie group $\mathrm{Aff}(\mathbb{R}^2)$ with the structure of a differentiable manifold $\mathcal{M}$ such that the group operations, multiplication and inverse, are differentiable maps. The structure of $\mathrm{Aff}(\mathbb{R}^2)$ is a 6 dimensional manifold with the 3×3 affine transformation matrix as:

$$\mathsf{M} = \begin{pmatrix} \mathsf{A} & v \\ 0 & 1 \end{pmatrix} \tag{1}$$

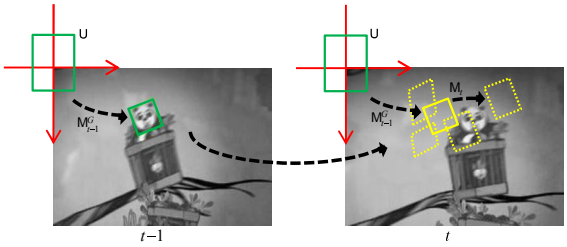


Figure 1. Predict the location of object region in frame t based on the location obtained in the previous frame $t-1$.

where, A is a 2×2 matrix (for rotation, scale, sheer), and $v \in \mathbb{R}^2$ (for translation). Here, the tangent space $\mathcal{T}_{\mathsf{I}}\mathcal{M}$ to the identity element I of the group forms a Lie algebra. The distances on the manifold $\mathcal{M}$ are measured by the lengths of the curves connecting the points, and the minimum length curve between two points is called the geodesic. From I there exists a unique geodesic starting with point m. The exponential map, $\exp : \mathcal{T}_{\mathsf{I}}\mathcal{M} \rightarrow \mathcal{M}$ maps the point m on the tangent space to the point reached by this geodesic. Let $\exp(\mathsf{m}) = \mathsf{M}$, then the length of the geodesic is given by $\rho(\mathsf{I}, \mathsf{M}) = \|\mathsf{m}\|$. The inverse mapping is given by $\log : \mathcal{M} \rightarrow \mathcal{T}_{\mathsf{I}}\mathcal{M}$. Using the logarithm map and the group operation, the geodesic distance between two group elements is measured by

$$\rho(\mathsf{M}_1, \mathsf{M}_2) = \|\log(\mathsf{M}_1^{-1}\mathsf{M}_2)\|. \tag{2}$$

The exponential and logarithm maps of a motion matrix are given by

$$\exp(\mathsf{m}) = \sum_{n=0}^{\infty} \frac{1}{n!}\mathsf{m}^n \quad \log(\mathsf{M}) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n}(\mathsf{M}-\mathsf{I})^n. \tag{3}$$

In case M^G_{t-1} represents the affine transformation from the unit rectangle (for normalization purposes, we map back onto unit rectangle U when we compute image features) to the object region in frame $t-1$, the *incremental* motion M_t is defined as

$$\mathsf{M}^G_t = \mathsf{M}_t\mathsf{M}^G_{t-1} \tag{4}$$

where M^G_t is the object box parallelogram in the frame t.

Inspired by [16], we treat the affine tracking-by-detection problem as learning a prediction function $f : x \rightarrow \mathcal{M}$ where x is the feature vector extracted from the object region. The prediction function f is determined in a structured output SVM framework [8]. Let $F : x \times \mathcal{M} \rightarrow \mathbb{R}$ be a discriminant function that maps both an affine motion matrix and the feature corresponding to its region in the image to a scalar label. Here, we assign the discriminant function as it as

$$\mathsf{M}_t = f(x_t(\mathsf{M}^G_{t-1})) = \arg\max_{\mathsf{M}\in\mathcal{M}} F(x_t(\mathsf{M}^G_{t-1}), \mathsf{M}), \tag{5}$$

where M_{t-1}^G is the location of the object region in frame $t-1$ as we stated above. Here, M_{t-1}^G transforms the unit rectangle U to the parallelogram that bounds the target region in frame $t-1$.

The discriminant function F measures the compatibility between the feature x and incremental affine motion M pairs $(x_t(\mathsf{M}_{t-1}^G), \mathsf{M})$. In other words, F has a higher score when the affine transformation M leads into a more accurate location of object region in frame t. By exhaustively searching over all possible transformations $\mathsf{M} \in \mathcal{M}$ near the object region in the previous frame, the target M_t can be obtained as the maximizer of $F(x_t(\mathsf{M}_{t-1}^G), \mathsf{M})$.

As the structured output SVM formulations [8], we express the discriminant function in the form of

$$F(x_t(\mathsf{M}_{t-1}^G), \mathsf{M}) = \langle \boldsymbol{w}, \phi(x_t(\mathsf{M}_{t-1}^G), \mathsf{M}) \rangle, \tag{6}$$

where $\phi(x_t(\mathsf{M}_{t-1}^G), \mathsf{M})$ is a raising function from the joint (feature, motion) space to a transform space. The specific form of ϕ is not necessarily to be defined explicitly by taking advantage of the kernel-based method [29] (Section 4). The linear coefficient vector $\boldsymbol{w}$ can be learned through an incrementally obtained set of example pairs

$$\mathcal{S}_t = \{(x_1(\mathsf{M}_0^G), \mathsf{M}_1), \ldots, (x_t(\mathsf{M}_{t-1}^G), \mathsf{M}_t)\}, \tag{7}$$

Learning procedure is then minimizing the following convex objective function:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + c\sum_{i=1}^{t}\xi_i, \\ \text{s.t.} \quad & \xi_i \geq 0, \forall i \\ & \langle \boldsymbol{w}, \delta\phi(x_i(\mathsf{M}_{i-1}^G), \mathsf{M}) \rangle \geq \mathcal{L}(\mathsf{M}_i, \mathsf{M}) - \xi_i, \forall i, \forall \mathsf{M} \neq \mathsf{M}_i, \end{aligned} \tag{8}$$

where $\delta\phi(x_i(\mathsf{M}_{i-1}^G), \mathsf{M}) = \phi(x_i(\mathsf{M}_{i-1}^G), \mathsf{M}_i) - \phi(x_i(\mathsf{M}_{i-1}^G), \mathsf{M})$ and c is the blending weight for the (soft-margin) errors. Thus optimization of (8) finds such $\boldsymbol{w}$ that enables discriminant function (6) to produce lower values for $\mathsf{M} \neq \mathsf{M}_i$, by a margin depends on a loss function $\mathcal{L}(\mathsf{M}_i, \mathsf{M})$. This loss function should satisfy $\mathcal{L}(\mathsf{M}_i, \mathsf{M}) = 0$ iff $\mathsf{M} = \mathsf{M}_i$ and decrease towards 0 as M and M_i become more similar.

3. Loss Function

Loss function $\mathcal{L}(\mathsf{M}_i, \mathsf{M})$ plays an important role in optimizing (8), as it quantifies the loss associated with a prediction M, if the true output value is M_i [29]. It allows to address the issue raised in the previous works that all negative samples being treated equally [16]. Thus, the standard zero-one loss function typically used in classification is not appropriate for this problem and we introduce three loss function forms in this paper.

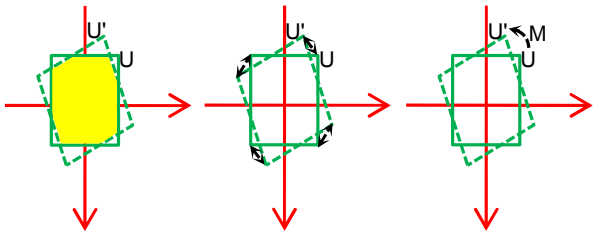


Figure 2. An illustration of three ways to design the loss function. From left to right: overlap rate based, average vertex distance based and geodesic distance based.

3.1. Conventional: Based on Overlap Rate

For affine tracking, one can use the loss function based on the bounding box overlap rate as in [16]:

$$\mathcal{L}_o(\mathsf{M}_i, \mathsf{M}) = 1 - \mathcal{O}_{\mathsf{M}_{i-1}^G}(\mathsf{M}_i, \mathsf{M}), \tag{9}$$

$$\mathcal{O}_{\mathsf{M}_{i-1}^G}(\mathsf{M}_i, \mathsf{M}) = \frac{(\mathsf{M}_i\mathsf{M}_{i-1}^G) \cap (\mathsf{M}\mathsf{M}_{i-1}^G)}{(\mathsf{M}_i\mathsf{M}_{i-1}^G) \cup (\mathsf{M}\mathsf{M}_{i-1}^G)}. \tag{10}$$

Function (10) measures the degree of overlap between two parallelograms in frame i: $\mathsf{M}_i\mathsf{M}_{i-1}^G$ and $\mathsf{M}\mathsf{M}_{i-1}^G$, as illustrated on the left of Figure 2.

The overlap rate based loss function is used in [16] for the translational motion. This treatment is, however, more of a coarse heuristic approach than a rigorous mathematical method for our problem, as the overlap rate measurement (10) can be very ambiguous when representing the similarity between two affine transformations M_i and M. Take a unit rectangle for example, rotating it by 90 degrees in any direction produces 100% overlap rate, which indicates the loss function of [16] based on overlap rate is not suitable for affine tracking.

3.2. Average Vertex Distance Based Loss Function

Instead of using the overlap rate measurement (10), the following average vertex distance based loss function can be applied:

$$\mathcal{L}_v(\mathsf{M}_i, \mathsf{M}) = 1 - \exp(-\tau_v\rho_v^2(\mathsf{M}_i, \mathsf{M})), \tag{11}$$

where τ_v is a constant and $\rho_v(\mathsf{M}_i, \mathsf{M})$ is the mean distance of four pairs of corresponding vertices as shown on the middle of Figure 2:

$$\rho_v(\mathsf{M}_i, \mathsf{M}) = \frac{1}{4}\sum_{j=1}^{4}\|\mathsf{v}_i^j - \mathsf{v}^j\|^2, \tag{12}$$

where $\{\mathsf{v}_i^1, \mathsf{v}_i^2, \mathsf{v}_i^3, \mathsf{v}_i^4\}$ and $\{\mathsf{v}^1, \mathsf{v}^2, \mathsf{v}^3, \mathsf{v}^4\}$ represent the four corresponding vertices of parallelograms $\mathsf{M}_i\mathsf{M}_{i-1}^G$ and $\mathsf{M}\mathsf{M}_{i-1}^G$ respectively.

Compared to the overlap rate based loss function (9), the average vertex distance based loss function gives a more reliable loss measurement between two transformations: M_i and M. The spatial order of the four corresponding vertices is taken into account and there is no more obvious rotation ambiguity, though it is still in the image space and not handled principally.

3.3. Geodesic Distance Based Loss Function

In order to obtain a loss function which correctly measures the difference between two affine transformations, we propose a geodesic distance based function:

$$\mathcal{L}_g(\mathsf{M}_i, \mathsf{M}) = 1 - \exp(-\tau_g \rho_g^2(\mathsf{M}_i, \mathsf{M})), \qquad (13)$$

where τ_g is a constant and $\rho_g(\mathsf{M}_i, \mathsf{M})$ is the geodesic distance of two affine transformations. For loss function (13), any transformation $\mathsf{M} \neq \mathsf{M}_i$ can thus be correctly assigned with a mathematically well-defined loss. Here, the geodesic distance distance $\rho_g(\mathsf{M}_i, \mathsf{M})$ is $\| \log(\mathsf{M}_i^{-1}\mathsf{M}) \|$. Following [26], we use a first order approximation:

$$\rho_g(\mathsf{M}_i, \mathsf{M}) \approx \| \log(\mathsf{M}) - \log(\mathsf{M}_i) \|. \qquad (14)$$

To visually demonstrate the differences among the overlap rate based, average vertex distance based and geodesic distance based loss functions, we apply a pure rotation transformation on a unit square as shown on the left of Figure 3. The corresponding loss measurements are calculated then we illustrate them on the right of Figure 3. It clearly shows that the principled geodesic distance is linear to the rotation angle while other two measurements are not.

4. Tracking Procedure

We summarize the basic tracking steps of the proposed algorithm below:

Given the bounding box of object in the first frame as M_0^G.
Set the training example pair set as $\mathcal{S}_0 = \{\emptyset\}$.
For each frame t in the sequence, do the following steps.

---

1. Add the current example pair into the training set as $\mathcal{S}_t = \{(x_1(\mathsf{M}_0^G), \mathsf{I}), \ldots, (x_t(\mathsf{M}_{t-1}^G), \mathsf{M}_t)\}$.
2. Learn discriminant function F by optimizing (8) with training set $\mathcal{S}_t$.
3. Estimate the location of object in frame $t+1$: $\mathsf{M}_{t+1} = \arg\max_{\mathsf{M} \in \mathcal{M}} F(x_{t+1}(\mathsf{M}_t^G), \mathsf{M})$.

During tracking, two major problems need to be carefully considered. Training stage: minimization of the convex learning function (8). As the training set $\mathcal{S}_t$ is incrementally obtained, re-optimizing function (8) independently every time after obtaining a new sample pair will be time consuming. In other words, training with all possible motion hypotheses in the affine motion manifold $\mathcal{M}$ would make a brute-force optimization intractable.

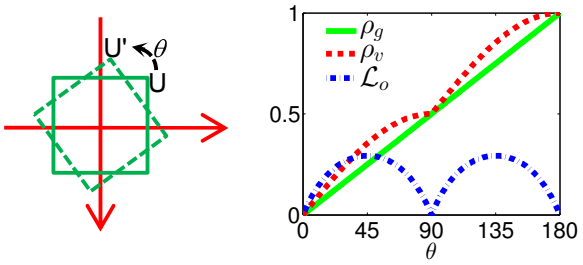


Figure 3. The overlap rate based, average vertex distance based and geodesic distance based loss measurements under pure rotation transformation.

Testing stage: optimizing the objective function (5). As $\mathcal{M}$ denotes a matrix Lie group the set of all affine transformations, which is 6 degrees of freedom, an efficient searching approach over $\mathcal{M}$ needs to be employed in order to achieve a practical processing speed. In the following section, we explain how to optimize (8) given a set of example pairs $\mathcal{S}_t$.

4.1. Minimizing the Objective Function

The approach proposed in [10] is used for optimizing the function (8) as suggested in [16]. Here we provide the version on the manifold following the equivalent dual form problem:

$$\begin{aligned} \max_{\boldsymbol{\beta}} \quad & -\sum_{i,\mathsf{M}} \mathcal{L}(\mathsf{M}_i, \mathsf{M}) \beta_i^{\mathsf{M}} \\ & -\frac{1}{2} \sum_{i,\mathsf{M},j,\widetilde{\mathsf{M}}} \beta_i^{\mathsf{M}} \beta_j^{\widetilde{\mathsf{M}}} \langle \phi(x_i(\mathsf{M}_{i-1}^G), \mathsf{M}), \phi(x_j(\mathsf{M}_{j-1}^G), \widetilde{\mathsf{M}}) \rangle, \\ \text{s.t.} \quad & \beta_i^{\mathsf{M}} \leq c\Delta(\mathsf{M}_i, \mathsf{M}), \forall i, \forall \mathsf{M} \\ & \sum_{\mathsf{M}} \beta_i^{\mathsf{M}} = 0, \forall i \end{aligned} \qquad (15)$$

where $\Delta(\mathsf{M}_i, \mathsf{M}) = 1$ if $\mathsf{M}_i = \mathsf{M}$ and 0 otherwise, c is the same as in (8). The discriminant function $F(x_i(\mathsf{M}_{i-1}^G), \mathsf{M})$ can also be represented in the dual form as

$$\sum_{j,\widetilde{\mathsf{M}}} \beta_j^{\widetilde{\mathsf{M}}} \langle \phi(x_i(\mathsf{M}_{i-1}^G), \mathsf{M}), \phi(x_j(\mathsf{M}_{j-1}^G), \widetilde{\mathsf{M}}) \rangle. \qquad (16)$$

We refer those pairs $(x_i(\mathsf{M}_{i-1}^G), \mathsf{M})$ for which $\beta_i^{\mathsf{M}} \neq 0$ as support vectors as [9, 10]. Only the support vector $(x_i(\mathsf{M}_{i-1}^G), \mathsf{M}_i)$ will have $\beta_i^{\mathsf{M}} > 0$, while any other support vector will have $\beta_i^{\mathsf{M}} < 0$, $\mathsf{M} \neq \mathsf{M}_i$. They are referred as positive and negative support vectors respectively. Function

(16) is computed using a joint kernel function as:

$$\begin{aligned}&\langle\phi(x_i(\mathsf{M}_{i-1}^G),\mathsf{M}),\phi(x_j(\mathsf{M}_{j-1}^G),\widetilde{\mathsf{M}})\rangle\\&=K(\hat{x}_i(\mathsf{MM}_{i-1}^G),\hat{x}_j(\widetilde{\mathsf{M}}\mathsf{M}_{j-1}^G)).\end{aligned}\tag{17}$$

Here, $\hat{x}_i(\mathsf{MM}_{i-1}^G)$ is the feature vector (HOG, Haar) extracted from the parallelogram MM_{i-1}^G in frame i (Figure 1). The kernel K can be any kernel such as Gaussian.

Optimizing function (15) is then composed of mainly two basic operations: select a triplet $\{i,\mathsf{M}_+,\mathsf{M}_-\}$ and optimize its corresponding coefficients $\beta_i^{\mathsf{M}_+}$ and $\beta_i^{\mathsf{M}_-}$ using an SMO step [24]. The parameter i in the triplet is randomly selected, and for a given i, M_+ and M_- are chosen with respect to the gradient of the function (15):

$$\partial_i(\mathsf{M})=-\mathcal{L}(\mathsf{M}_i,\mathsf{M})-F(x_i(\mathsf{M}_{i-1}^G),\mathsf{M}).\tag{18}$$

For example, M_- can be chosen by $\mathsf{M}_-=\arg\min_{\mathsf{M}\in\mathcal{M}}\partial_i(\mathsf{M})$, i.e., finding the most important negative sample in frame i: the one has high compatibility value of F while possesses a big difference with M_i.

4.2. Online Update

As we mentioned at the beginning of Section 4, training example set $\mathcal{S}_t$ is incrementally obtained and re-optimizing function (15) for every frame will be time-consuming. Thus, we propose the following approach.

Given support vectors $\mathcal{V}=\{(x_i(M_{i-1}^G),\mathsf{M})\mid\beta_i^{\mathsf{M}}\neq 0,i=1,\ldots,t-1\}$ and a new example pair $(x_t(\mathsf{M}_{t-1}^G),\mathsf{M}_t)$ at frame t.

---

1. Select a triplet $\{t,\mathsf{M}_+,\mathsf{M}_-\}$, where $\mathsf{M}_+=\mathsf{M}_t$ and $\mathsf{M}_-=\arg\min_{\mathsf{M}\in\mathcal{M}}\partial_t(\mathsf{M})$.
2. Optimize the triplet $\{t,\mathsf{M}_+,\mathsf{M}_-\}$ obtained in step 1 using SMO, if the resulted coefficients $\beta_t^{\mathsf{M}_+}$ and $\beta_t^{\mathsf{M}_-}$ are not zero, add them into $\mathcal{V}$.
3. Select the triplet $\{i,\mathsf{M}_+,\mathsf{M}_-\}$ (for random i): $\mathsf{M}_+=\arg\max_{\mathsf{M}\in\mathcal{M}_i}\partial_i(\mathsf{M})$ and $\mathsf{M}_-=\arg\min_{\mathsf{M}\in\mathcal{M}_i}\partial_i(\mathsf{M})$, where $\mathcal{M}_i=\{\mathsf{M}\in\mathcal{M}|\beta_i^{\mathsf{M}}\neq 0\}$.
4. Optimize the triplet $\{i,\mathsf{M}_+,\mathsf{M}_-\}$ obtained in step 3 using SMO, if $\beta_t^{\mathsf{M}_+}$ or $\beta_t^{\mathsf{M}_-}$ is zero, remove them from $\mathcal{V}$, else update the corresponding coefficient in $\mathcal{V}$.
5. Repeat step 3 to step 4 N_o times.
6. Repeat step 1 to step 5 N_a times.

The biggest difference with the method proposed in [16] is that we do not revisit previous frames for adding new samples as negative support vectors any more but only in the current frame (corresponds to step 1). This strategy is widely used in sparsity-based tracking methods [34, 35, 25], as they argue that recent observations will be more indicative. Note that step 3 will not add new support vectors, but can remove existing support vectors depending on the result of the SMO optimization [24]. In our paper, $N_o=10$ and $N_a=10$.

4.3. Efficient Tracking

Another issue we mentioned at the beginning of Section 4 is how to efficiently optimize the objective function (5) in the testing stage. Similar difficulty exits when to add new support vector in Section 4.2 solving $\arg\min_{\mathsf{M}\in\mathcal{M}}\partial_t(\mathsf{M})$.

Two sampling strategies are proposed: uniform sampling and particle filter sampling. The former uniformly samples points on the manifold $\mathcal{M}$. For particle filter sampling, advanced methods such as [19] can be employed. Here, we treat 6 affine parameters independently and model them with 6 Gaussian distributions as in [35, 25, 34].

The number of support vectors has to be limited, as the computational and memory costs increase with the number of support vectors and the number of training examples can be huge in the tracking procedure. Employing the method proposed in [33], we remove support vector $(x_r(\mathsf{M}_{r-1}^G),\mathsf{M})$ that results in the smallest change to the coefficient vector $\boldsymbol{w}$, as measured by $\|\Delta\boldsymbol{w}\|^2$. when the budget is exceeded.

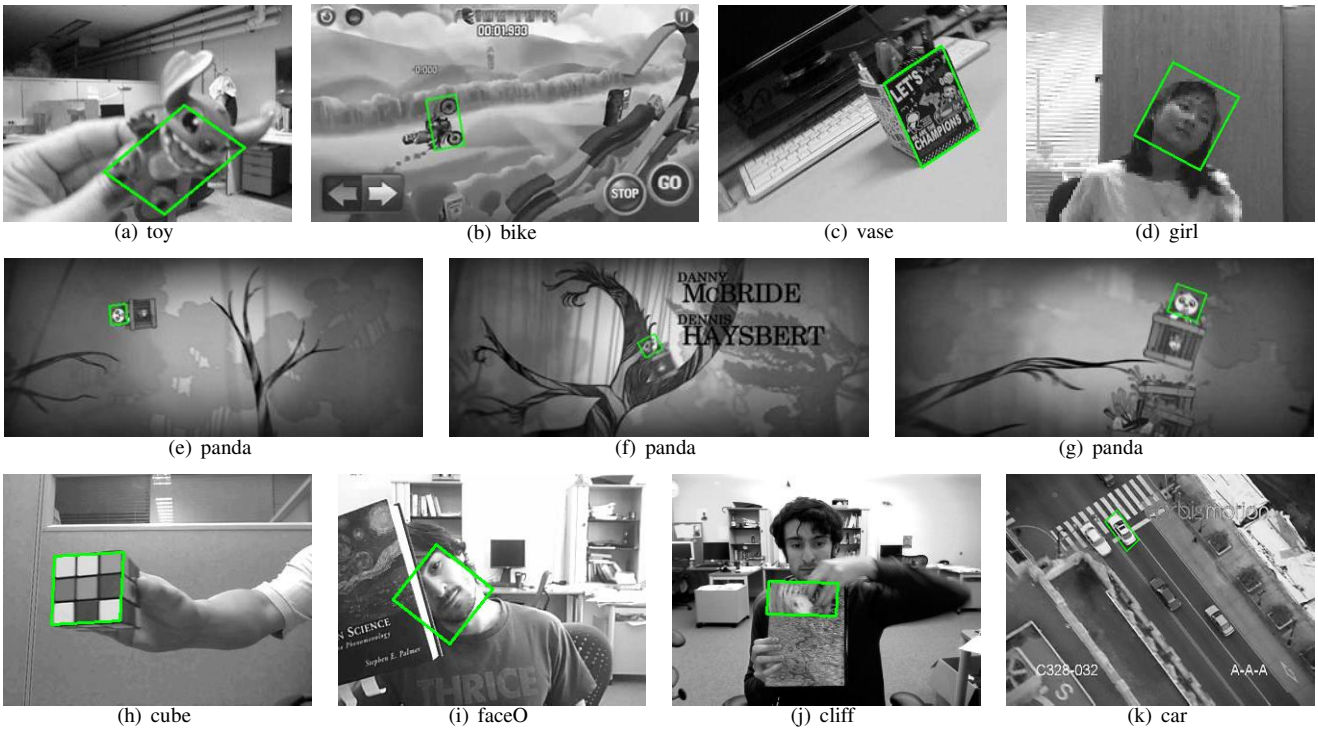
5. Experiments

5.1. Datasets and Evaluation Method

Since there are no existing annotated datasets specified for affine tracking, we collect nine image sequences and manually annotate them frame by frame. Most sequences in the proposed dataset are subjected to some challenging aspects such as motion blur and occlusion. This dataset will be publicly available and we summarize those aspects as follows: • toy: out-of-plane rotation. • bike: out-of-plane rotation, background clutters. • girl, faceO: occlusion. • panda: motion blur, occlusion. • cliff: background clutters, motion blur. • vase, cube, car: no challenging aspects involved.

Additionally, sequences from VOT2014 Challenge [1] are employed to evaluate our proposed method. They provided per-frame labeled rotated bounding box as ground truth and are not specifically designed for affine tracking purpose. We evaluate the proposed tracker on this dataset to demonstrate its performance for general tracking task.

To evaluate, traditional overlap rate (10) and center location error are used. We also propose the average vertex distance error ρ_v (12) and geodesic distance error ρ_g (14) to measure the error of the affine tracking results for an objective evaluation. Computational speed is evaluated as frames per second (FPS).



(a) toy (b) bike (c) vase (d) girl

(e) panda (f) panda (g) panda

(h) cube (i) faceO (j) cliff (k) car

Figure 4. Sample frames from our newly-constructed affine tracking dataset .

	toy	bike	vase	girl	panda	cube	faceO	cliff	car	**MeanV**	**MeanC**	**MeanO**	**MeanG**	**FPS**
Lie-Struck$_u$	**8.6**	**12.5**	9.7	**7.6**	**3.4**	6.5	7.7	**5.7**	5.5	**7.5**	**5.3**	**0.79**	**37.2**	0.4
O-Struck$_u$	9.1	81.4	9.8	28.3	85.3	11.1	9.7	6.6	5.7	27.4	21.3	0.68	55.1	0.4
V-Struck$_u$	9.6	74.7	9.8	15.4	56.1	5.1	8.9	6.1	6.1	20.4	14.9	0.74	61.2	0.4
Struck	23.1	95.5	47.1	19.3	104.3	46.3	29.9	67.9	31.6	51.7	36.5	0.39	91.5	2.7
SCM	32.7	35.8	7.4	25.3	110.1	**4.9**	161.3	65.4	66.5	56.6	47.0	0.51	95.8	0.2
Lie-Tracker	20.0	90.4	**4.2**	17.6	266.2	8.9	15.1	86.8	**4.5**	57.1	25.5	0.62	77.8	**4.3**
Lie-Struck$_{20}$	8.7	39.0	10.4	20.6	15.9	5.1	**6.2**	35.4	5.5	16.4	10.8	0.73	50.3	1.4
Lie-Struck$_{10}$	8.8	57.4	10.6	37.5	48.2	7.8	7.1	34.6	5.7	24.1	17.4	0.69	51.2	1.7
Lie-Struck$_6$	10.7	55.1	12.0	26.1	89.0	6.3	8.3	40.8	7.1	28.4	21.3	0.67	54.7	1.9

Table 1. The average vertex distance error is reported for each sequence on the proposed affine tracking dataset. Processing speed (FPS) and means of center location error (MeanC), overlap rate (MeanO) and geodesic distance error (MeanG) over all sequences are reported as well. Best results are marked in bold.

5.2. Implementation

Lie-Struck is implemented in C++ and experiments are carried out on an Intel Core i7-2600 3.40GHz PC with 4 GB memory. The joint kernel function (17) is implemented using the same Harr feature and Gaussian kernel as [16]:

$$\exp(-\sigma\|\hat{x}_i(\mathsf{MM}_{i-1}^G) - \hat{x}_j(\widetilde{\mathsf{MM}}_{j-1}^G)\|^2),$$

where σ is 0.2. The budget size of support vectors is set to 100. For the geodesic distance based loss function (13), $\tau_g = 2$ and the matrix logarithm (3) is implemented using the Eigen C++ template library [15].

The same affine motion parameters are used for all tests in our experimental evaluation. For uniform sampling, we use 81 samples for the two translation parameters as same as [16], 5 samples with 4 degrees and 0.03 interval for the rotation angle and scaling parameters, 3 samples with 4 degrees and 0.03 interval for the skew angle and aspect ratio parameters. For particle filter sampling, the Gaussian parameters are $\{8, 8, 5, 0.04, 5, 0.04\}$ respectively. The proposed approach using uniform sampling is denoted as Lie-Struck$_u$, while the one uses 2000 particles as Lie-Struck$_{20}$, 1000 particles as Lie-Struck$_{10}$, and 600 particles as Lie-Struck$_6$.

Lie-Struck are compared against 2 state-of-the-art trackers that are able to perform affine tracking : SCM [35] and Lie-Tracker [30]. For SCM tracker, particle filter number is set to 1000 and same parameters are used. For Lie-Tracker, default setting is used. To demonstrate the effectiveness of the geodesic distance based loss function (13), we also implement the approaches using the overlap rate based loss function (9) and the average vertex distance based ($\tau_v = 5$) loss function (11), which are denoted as O-Struck and V-Struck respectively. Struck [16] is evaluated as well.

	ball	bicy.	car	david	drunk	jog.	moto.	polar.	skat.	sphe.	suns.	surf.	tun.	**MeanV**	**MeanO**
Lie-Struck$_u$	38.8	51.6	27.8	**25.7**	44.6	**9.6**	**107.3**	44.6	93.0	21.1	7.3	6.9	13.9	**37.9**	**0.52**
O-Struck$_u$	42.1	28.5	29.1	43.0	108.1	60.2	316.7	71.6	101.5	26.9	6.7	25.3	21.9	67.8	0.43
V-Struck$_u$	40.9	51.7	28.7	57.3	60.2	105.1	479.5	43.7	117.1	24.7	6.2	13.1	21.3	80.7	0.44
Struck	**32.8**	11.6	43.4	57.2	69.1	11.1	139.1	15.6	76.1	**19.2**	**4.2**	3.7	22.3	38.9	0.50
SCM	103.3	9.9	**19.4**	26.2	72.4	153.0	203.0	17.4	79.5	26.5	6.6	3.7	43.1	58.8	0.47
Lie-Tracker	138.7	92.8	94.4	181.9	**34.9**	10.0	377.9	**8.8**	**46.4**	664.4	88.5	4.6	80.3	140.2	0.38
Lie-Struck$_{20}$	102.0	**8.3**	26.4	27.2	61.0	122.3	112.3	35.8	89.3	36.7	7.0	**3.6**	**10.0**	49.4	0.49
Lie-Struck$_{10}$	126.3	8.5	26.8	45.6	62.8	125.4	114.8	36.2	91.7	40.9	7.9	3.8	12.2	54.1	0.48
Lie-Struck$_6$	207.1	8.6	30.8	69.2	63.6	149.3	153.2	42.1	93.6	60.2	7.9	5.3	13.0	69.5	0.46

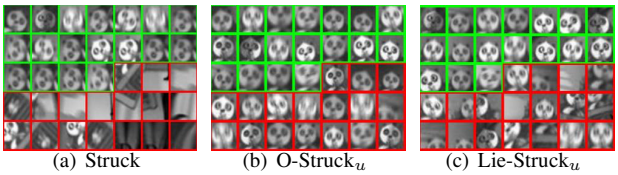
Table 2. The average vertex distance errors are reported for sequences from VOT2014 Challenge dataset (more general tracking scenarios). The mean of overlap rate (MeanO) over all sequences is reported as well. Best results are marked in bold.

5.3. Performance Evaluation

The results of evaluation are summarized in Table 1 for sequences on the proposed affine tracking dataset and Table 2 for sequences from VOT2014 Challenge dataset. Every sequence is repeated 10 times for those stochastic methods, especially ones using particle filter sampling, then the average result is reported.

Affine Tracking: Comparing the performance between Lie-Struck$_u$ and Struck on the affine tracking dataset, it convincingly demonstrated that the combined motion and appearance model structures greatly improve the tracking accuracy. This can be further validated using Figure 5, in which we visualize the support vectors maintained in the respective trackers. Those positive support vectors maintained in Lie-Struck$_u$ have consistent appearance, while Struck treats every rotated object region as a new positive support vector since it is not aware of rotations. This difference is significant and it causes a big performance gap as the novelly incorporated motion model simplifies the classification task for structure SVM.

The performance gaps among Lie-Struck$_u$, O-Struck$_u$ and V-Struck$_u$, especially for sequences such as "bike", "girl" and "panda", are huge because Lie-Struck$_u$ keeps tracking the objects while O-Struck$_u$ and V-Struck$_u$ lost the objects in the middle of sequences. The differences among them can also be demonstrated using Figure 5, as the negative support vectors in O-Struck$_u$ are not as distinctive from the positive support vectors as Lie-Struck$_u$, which indicates a better SVM classification boundary and more robustness from drifting away from the object region.



(a) Struck (b) O-Struck$_u$ (c) Lie-Struck$_u$

Figure 5. Visualization of the support vectors maintained by trackers at the frame $t = 125$ in sequence "panda". Positive and negative support vectors have green and red borders respectively. Notice that Struck treats every rotated object region as a positive support vector, while Lie-Struck$_u$ generates consistent positive support vectors and distinctive negative support vectors, which indicates an simplified classification boundary.

The incorporated particle filter mechanism has resulted comparable performance as the uniform sampling one, while the processing speed is improved. It even produced better results ("cube", "faceO") as the Gaussian distribution well captures the smooth motion of objects. Overall, it clearly demonstrated that the proposed methods outperform existing state-of-the-arts: SCM and Lie-Tracker, with comparable computational speeds.

General Tracking: Based on the experimental results (Table 2) for sequences from VOT2014 Challenge dataset, we can see that the proposed methods achieve competitive performance, though this dataset is not specifically designed for affine tracking purpose. Especially on sequences such as "jog." and "moto.", where affine transformations present, our methods showed clear advantages over the state-of-the-arts. In all, the resulted performance differences among Lie-Struck$_u$, O-Struck$_u$ and V-Struck$_u$ demonstrated that the principled geodesic distance based loss function outperforms other image space based loss functions in both affine and general tracking scenarios.

6. Conclusions

We proposed a novel affine tracking-by-detection method which took advantage of the intrinsic topology of manifold. It incorporated the Lie group structure into a structured SVM classifier to directly estimate 2D affine transformation using an appearance based prediction function. We demonstrated that these combined motion and appearance model structures significantly improved the tracking performance on a newly-constructed affine tracking dataset and a challenging general tracking dataset, while an incorporated particle filter mechanism kept the processing speed fast.

Acknowledgements

The research is funded in part by ARC-DP120103896, DP130104567, LP100100588, CE140100016 and NICTA.

References

[1] VOT2014 Challenge. http://www.votchallenge.net/vot2014/index.html.

[2] T. Albrecht, M. Luthi, and T. Vetter. A statistical deformation prior for non-rigid image and shape registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[3] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.

[4] 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.

[5] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[6] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[7] E. Bayro-Corrochano and J. Ortegon-Aguilar. Lie algebra template tracking. In *International Conference on Pattern Recognition (ICPR)*, 2004.

[8] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *European Conference on Computer Vision (ECCV)*, 2008.

[9] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with LaRank. In *International Conference on Machine Learning (ICML)*, 2007.

[10] A. Bordes, N. Usunier, and L. Bottou. Sequence labelling SVMs trained in one pass. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2008.

[11] Y. Boykov and D. Huttenlocher. Adaptive bayesian recognition in tracking rigid objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[12] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. In *European Conference on Computer Vision (ECCV)*, 1998.

[13] T. Drummond and R. Cipolla. Application of Lie algebras to visual servoing. *International Journal on Computer Vision*, 37:21–41, 2000.

[14] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *British Machine Vision Conference (BMVC)*, 2006.

[15] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[16] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[17] W. He, T. Yamashita, H. Lu, and S. Lao. SURF tracking. In *International Conference on Computer Vision (ICCV)*, 2009.

[18] M. Isard and I. Blake. Condensation – conditional density propagation for visual tracking. *International Journal on Computer Vision*, 29:5–28, 1998.

[19] J. Kwon, K. M. Lee, and F. Park. Visual tracking via geometric particle filtering on the affine group with optimal importance functions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[20] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1981.

[21] H. Masnadi-Shirazi, V. Mahadevan, and N. Vasconcelos. On the design of robust classifiers for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[22] I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60:135–164, 2004.

[23] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[24] J. C. Platt. Advances in kernel methods. chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pages 185–208. MIT Press, 1999.

[25] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal on Computer Vision*, 77(1-3):125–141, 2008.

[26] W. Rossmann. *Lie groups: A introduction through linear groups*. Oxford University Press, 2002.

[27] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof. Online multi-class LPBoost. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2010.

[28] A. Saffari, C. Leistner, M. Godec, and H. Bischof. Robust multi-view boosting with priors. In *European Conference on Computer Vision Conference on Computer Vision (ECCV)*, 2010.

[29] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[30] O. Tuzel, F. Porikli, and P. Meer. Learning on Lie groups for invariant detection and tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[31] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19:733–742, 1997.

[32] D. Wagner, T. Langlotz, and D. Schmalstieg. Robust and unobtrusive marker tracking on mobile phones. In *ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2008.

[33] Z. Wang, K. Crammer, and S. Vucetic. Multi-class Pegasos on a budget. In *International Conference on Machine Learning (ICML)*, 2010.

[34] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan. Robust object tracking with online multi-lifespan dictionary learning. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[35] W. Zhong, H. Lu, and M.-H. Yang. Robust object tracking via sparsity-based collaborative model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.