Learning on Lie Groups for Invariant Detection and Tracking

Oncel Tuzel[†‡]
[†]Rutgers University, CS
Piscataway, NJ 08854
otuzel@caip.rutgers.edu

Fatih Porikli[‡]
[‡]Mitsubishi Electric Research Labs
Cambridge, MA 02139
fatih@merl.com

Peter Meer[†§]
[§]Rutgers University, ECE
Piscataway, NJ 08854
meer@caip.rutgers.edu

Abstract

This paper presents a novel learning based tracking model combined with object detection. The existing techniques proceed by linearizing the motion, which makes an implicit Euclidean space assumption. Most of the transformations used in computer vision have matrix Lie group structure. We learn the motion model on the Lie algebra and show that the formulation minimizes a first order approximation to the geodesic error. The learning model is extended to train a class specific tracking function, which is then integrated to an existing pose dependent object detector to build a pose invariant object detection algorithm. The proposed model can accurately detect objects in various poses, where the size of the search space is only a fraction compared to the existing object detection methods. The detection rate of the original detector is improved by more than 90% *for large transformations.*

1. Introduction

An extensive literature exists on tracking problem and here we reference only the template alignment methods. In optical flow formulation [9], the sum of squared difference between the template and the image intensities was minimized as an iterative least squares problem. The method requires computation of the image gradient, the Jacobian and the Hessian for each iteration, which makes it slow. Variants of the method were proposed to overcome the difficulty [5, 12]. An overview can be found in [10]. In [2, 7], the motion was estimated using a linear function of the image gradient, which was learned in an off-line process. Later [15], the idea was extended to learn a nonlinear model using relevance vector machine. These methods estimate the additive updates to the motion parameters via linearization.

In this paper we use Lie group theory for motion estimation. A few of the related papers are as follows. In [13], a mode finding algorithm on Euclidean motion group was described for a multiple motion estimation problem. In [4], an addition operation was defined on the Lie algebra for tracking an affine snake. In [1], the additive updates were performed on the Lie algebra for template tracking. However, the approach in [1] fails to account for the noncommutativity of the matrix multiplications and the estimations become valid only around the initial transformation of the target. In a recent study [3], a kernel regression model for manifold valued data is described for analyzing shape changes of the brain on MR images. The approach is computationally expensive and is not well suited for real time applications such as tracking.

We present a novel formulation for motion estimation by learning a regression model on the Lie algebra. We show examples on affine motion group, however the method is applicable to any matrix Lie group structured transformation. The appearance of an object is described with several orientation histograms computed on a regular grid. Using a regression function, we learn the correlation between motions and the observed descriptors.

Majority of the current state of art object detection algorithms are based on sequentially applying a learned classifier of the object model at all the possible subwindows. However, a brute force approach on a high dimensional search space is computationally intractable. The proposed learning model is extended to train a class specific tracking function which can localize the targets with a sparse scan on the motion space. The motion estimator is then integrated to an existing pose dependent object detector and a pose invariant object detection algorithm with respect to the motion model is developed.

2. Tracking as a Learning Problem

The method is demonstrated on affine motions, however, it generalizes to any matrix Lie group transformations. A two-dimensional affine transformation $A(2)$ is given by a 3×3 matrix $\mathbf{M}$

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ 0 & 1 \end{pmatrix} \tag{1}$$

where $\mathbf{A}$ is a nonsingular 2×2 matrix and $\mathbf{b} \in \mathbb{R}^2$. The set of all affine transformations forms a matrix Lie group. Let $\mathbf{M}$ transforms a unit square at the origin to the affine region enclosing the target object

$$[x_{img} \;\; y_{img} \;\; 1]^T = \mathbf{M}[x_{obj} \;\; y_{obj} \;\; 1]^T \tag{2}$$

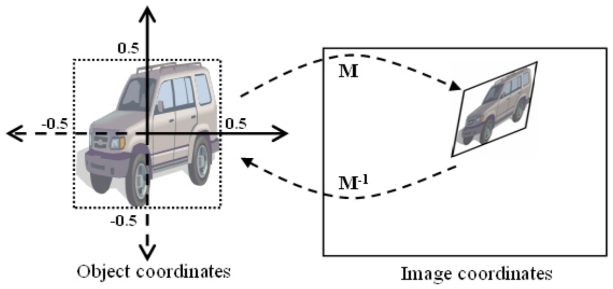


Figure 1. The mapping and its inverse, between the object and image coordinates.

where, the subscripts indicate the object coordinates and image coordinates respectively. The inverse $\mathbf{M}^{-1}$ is also an affine motion matrix and transforms the image coordinates to the object coordinates (Figure 1).

Let I denote the observed images and t be the time index. The aim of tracking is to estimate the transformation matrix $\mathbf{M}_t$, given the observations up to time t, $I_{0\ldots t}$, and the initial transformation $\mathbf{M}_0$. We model the transformations incrementally

$$\mathbf{M}_t = \mathbf{M}_{t-1}.\Delta\mathbf{M}_t \tag{3}$$

and estimate the increments $\Delta\mathbf{M}_t$ at each time frame. The transformation $\Delta\mathbf{M}_t$ corresponds to motion of target from time $t-1$ to t in the object coordinates.

The image in the object coordinates is written as $I(\mathbf{M}^{-1})$. We consider the pixel values inside the unit rectangle and represent the region with a descriptor, such as, orientation histograms. It is denoted by $\mathbf{o}(\mathbf{M}^{-1}) \in \mathbb{R}^m$ where m is the dimension of the descriptor.

We interpret tracking as a matrix valued regression problem. Given the previous location of the object $\mathbf{M}_{t-1}$ and the current observation I_t, we estimate the new transformation $\Delta\mathbf{M}_t$ by the regression function

$$\Delta\mathbf{M}_t = f(\mathbf{o}_t(\mathbf{M}_{t-1}^{-1})). \tag{4}$$

The problem reduces to learning and updating the regression function f, where the details are explained in Section 4.

During initialization, $t = 0$, the observation I_0 and the initial location of the object $\mathbf{M}_0$ are given. We generate a training set of n random affine transformation matrices $\{\Delta\mathbf{M}_i\}_{i=1\ldots n}$ around the identity matrix. The object coordinates are transformed by multiplying on the left with $\Delta\mathbf{M}_i^{-1}$ and the new descriptor is computed by $\mathbf{o}_0^i = \mathbf{o}_0\left(\Delta\mathbf{M}_i^{-1}.\mathbf{M}_0^{-1}\right)$. The transformation $\Delta\mathbf{M}_i$ moves the object back to the unit square. The training set consists of samples $\left\{\mathbf{o}_0^i, \Delta\mathbf{M}_i\right\}_{i=1\ldots n}$ and the process is illustrated in Figure 2. Notice that we use the notation $\Delta\mathbf{M}$ both for the elements of training set with subscript i and the estimated motions during tracking with subscript t.

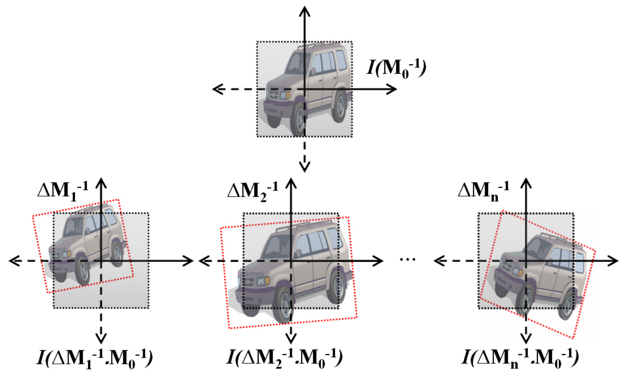


Figure 2. Training samples are generated by applying n affine motions $\Delta\mathbf{M}_{i=1\ldots n}^{-1}$ at the object coordinates. Using (3), the new observations in the object coordinates are $I_0\left((\mathbf{M}_0.\Delta\mathbf{M}_i)^{-1}\right)$, where an equivalent form is used in the image.

The regression function $f : \mathbb{R}^m \mapsto A(2)$ is an affine matrix valued function. The standard approach for motion estimation is through introducing a parametrization of the motion and linearization [2, 7, 15], which in this case is around the identity matrix

$$\Delta\mathbf{M}(\mathbf{p_0} + \Delta\mathbf{p}) \approx \Delta\mathbf{M}(\mathbf{p}_0) + \frac{\partial\Delta\mathbf{M}}{\partial\mathbf{p}}\Delta\mathbf{p}. \tag{5}$$

where $\Delta\mathbf{M}(\mathbf{p_0}) = \mathbf{I}$. The approach proceeds by estimating the increments $\Delta\mathbf{p}$. There are two major drawbacks of the approach. Firstly, the approximation makes a vector space assumption on the parameters. Secondly, the parametrization is arbitrary and do not consider the structure of the motion. We use the Lie group theory [11] to estimate the tracking function.

3. Lie Groups

A Lie group is a group G with the structure of a differentiable manifold such that the group operations, multiplication and inverse, are differentiable maps. The tangent space to the identity element $\mathbf{I}$ of the group forms a Lie algebra $\mathfrak{g}$. In our convention we are referring to points on the group with bold capital letters and vectors on the Lie algebra with small capital letters.

The distances on the manifold are measured by the lengths of the curves connecting the points, and the minimum length curve between two points is called the geodesic. From $\mathbf{I}$ there exists a unique geodesic starting with vector $\mathbf{m} \in \mathfrak{g}$. The exponential map, $\exp : \mathfrak{g} \to G$ maps the vector $\mathbf{m}$ to the point reached by this geodesic. Let $\exp(\mathbf{m}) = \mathbf{M}$, then the length of the geodesic is given by $\rho(\mathbf{I}, \mathbf{M}) = \|\mathbf{m}\|$. The inverse mapping is given by $\log : G \to \mathfrak{g}$. Using the logarithm map and the group operation, the geodesic distance between two group elements is measured by

$$\rho(\mathbf{M}_1, \mathbf{M}_2) = \|\log(\mathbf{M}_1^{-1}\mathbf{M}_2)\|. \tag{6}$$

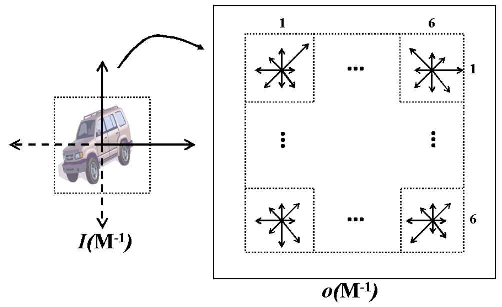


Figure 3. The gradient weighted orientation histograms are utilized as region descriptors.

We focus on matrix Lie groups only. The exponential and logarithm maps of a matrix are given by

$$\exp(\mathbf{m}) = \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{m}^n \quad \log(\mathbf{M}) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n}(\mathbf{M}-\mathbf{I})^n. \tag{7}$$

The structure of affine matrices was given in (1), which is a $d = 6$ dimensional manifold. The associated Lie algebra is the set of matrices

$$\mathbf{m} = \begin{pmatrix} \mathbf{U} & \mathbf{v} \\ 0 & 0 \end{pmatrix} \tag{8}$$

where, $\mathbf{U}$ is a 2×2 matrix and $\mathbf{v} \in \mathbb{R}^2$. The matrix $\mathbf{m}$ is sometimes referred to as a $d = 6$ dimensional vector by selecting each of the entries of $\mathbf{U}$ and $\mathbf{v}$ as an orthonormal basis.

4. Tracking via Regression on Lie Groups

The target region is represented with several orientation histograms computed at a regular grid inside the unit square in object coordinates (Figure 3). Similar to SIFT descriptors [8], the contribution of each pixel to the histogram is proportional to its gradient magnitude. The unit square is divided into $6 \times 6 = 36$ regions and a histogram is computed in each of them. Each histogram is quantized at $\pi/4$ degrees between 0 and 2π. The size of each histogram is eight dimensional and the descriptors, $\mathbf{o}$, are $m = 288$ dimensional. During tracking the peripheral pixels are frequently contaminated by the background, hence we leave a 10% boundary at the outerside of the unit square and construct the descriptor inside the inner rectangle.

4.1. Model Learning

During the model learning, the parameters of the regression function, $f : \mathbb{R}^m \mapsto A(2)$, are estimated. The training set consists of samples $\left\{\mathbf{o}_0^i, \Delta\mathbf{M}_i\right\}_{i=1\ldots n}$. Affine motion matrices lie on a differentiable manifold, and a meaningful error function is the sum of the squared geodesic distances between the estimations $f(\mathbf{o}_0^i)$, and the random transformations $\Delta\mathbf{M}_i$

$$J_g = \sum_{i=1}^{n} \rho^2 \left[f(\mathbf{o}_0^i), \Delta\mathbf{M}_i\right]. \tag{9}$$

Let $\mathbf{M}_1$ and $\mathbf{M}_2$ be two motion matrices, and let $\mathbf{m}_1 = \log(\mathbf{M}_1)$ and $\mathbf{m}_2 = \log(\mathbf{M}_2)$. Using Baker-Campbell-Hausdorff formula [11, p.22-23] which gives the exponential identity for non-commutative Lie groups, a first order approximation to the geodesic distance between the two motion matrices is

$$\begin{aligned} \rho(\mathbf{M}_1, \mathbf{M}_2) &= \left\|\log\left[\mathbf{M}_1^{-1}\mathbf{M}_2\right]\right\| \\ &= \left\|\log\left[\exp(-\mathbf{m}_1)\exp(\mathbf{m}_2)\right]\right\| \\ &= \left\|\log\left[\exp(\mathbf{m}_2 - \mathbf{m}_1 + O(|(\mathbf{m}_1, \mathbf{m}_2)|^2)\right]\right\| \\ &\approx \|\mathbf{m}_2 - \mathbf{m}_1\|. \end{aligned} \tag{10}$$

Selecting d orthonormal bases on the Lie algebra, we can compute the matrix norm as the Euclidean distance between two vectors. Using (10), the function (9) is equivalent to minimizing

$$J_a = \sum_{i=1}^{n} \left\|\log\left(f(\mathbf{o}_0^i)\right) - \log\left(\Delta\mathbf{M}_i\right)\right\|^2 \tag{11}$$

up to first order terms. The approximation is good enough since the transformations are in a small neighborhood of the identity.

We define the regression function as

$$f(\mathbf{o}) = \exp\left(g(\mathbf{o})\right) \tag{12}$$

and learn the function $g : \mathbb{R}^m \mapsto \mathbb{R}^d$ which estimates the tangent vectors, $\log(\Delta\mathbf{M})$, on the Lie algebra. We model the function g as a linear function of the observations $\mathbf{o}$

$$g(\mathbf{o}) = \mathbf{o}^T\mathbf{\Omega} \tag{13}$$

where $\mathbf{\Omega}$ is the $m \times d$ matrix of regression coefficients.

Let $\mathbf{X}$ be the $n \times m$ matrix of initial observations and $\mathbf{Y}$ be the $n \times d$ matrix of mappings of motions to the Lie algebra

$$\mathbf{X} = \begin{pmatrix} \left[\mathbf{o}_0^1\right]^T \\ \vdots \\ \left[\mathbf{o}_0^n\right]^T \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} \left[\log\left(\Delta\mathbf{M}_1\right)\right]^T \\ \vdots \\ \left[\log\left(\Delta\mathbf{M}_n\right)\right]^T \end{pmatrix}. \tag{14}$$

Notice that, $\log(\Delta\mathbf{M}_1)$ is referred here in d-dimensional vector form. Substituting (12) and (13) into (11), we obtain

$$J_a = tr[(\mathbf{X\Omega} - \mathbf{Y})^T(\mathbf{X\Omega} - \mathbf{Y})] \tag{15}$$

where the trace replaces the summation in (11).

```
Input: Location of target at time $t-1$ is $\mathbf{M}_{t-1}$ and the current observation is $I_t$, maximum iteration number is $K$.
• $k = 1$ and $\mathbf{M}_t = \mathbf{M}_{t-1}$
• Repeat
    – $\Delta\mathbf{M}_t = f(\mathbf{o}_t(\mathbf{M}_t^{-1}))$
    – $\mathbf{M}_t = \mathbf{M}_t . \Delta\mathbf{M}_t$
    – $k = k + 1$
• Until $\Delta\mathbf{M}_t = \mathbf{I}$ or $k = K$
```

Figure 4. Tracking algorithm.

For real time tracking we keep the size of the training set relatively small, $n = 200$. Since number of samples is smaller than the dimension of the feature space, $n < m$, the system is underdetermined and the least squares estimate becomes inaccurate. To avoid overfitting, we introduce an additional constraint on the size of the regression coefficients

$$J_r = tr[(\mathbf{X}\mathbf{\Omega} - \mathbf{Y})^T(\mathbf{X}\mathbf{\Omega} - \mathbf{Y})] + \lambda\|\mathbf{\Omega}\|^2 \qquad (16)$$

which is called the *ridge regression* [6, p.59-64]. The minimizer of the error function J_r is given by

$$\mathbf{\Omega} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y} \qquad (17)$$

where $\mathbf{I}$ is an $m \times m$ identity matrix. The regularization coefficient λ determines the degree of shrinkage on the regression coefficients. The details of the parameter λ is explained in Section 6.

4.2. Interframe Correspondence

After learning the regression function f, the tracking problem reduces to estimating the motion via (4) using current observation I_t and updating the target location via (3). To better localize the target, at each frame we repeat the motion estimation using f a maximum of ten times or $\Delta\mathbf{M}_t$ becomes equal to identity (Figure 4).

4.3. Model Update

Since objects can undergo appearance changes in time, it is necessary to adapt to these variations. In our case, the model update reestimates the tracking function f. During tracking the target object, we generate $s = 2$ random observations at each frame with the same method described in Section 2. The observations stored for last $p = 100$ frames constitute the update training set. Let $\mathbf{X}_u$ and $\mathbf{Y}_u$ be the update training set stored in the matrix form as described in (14), and $\mathbf{\Omega}'$ be the previous model parameters. After each p frames of tracking, we update the coefficients of the regression function by minimizing the error

$$J_u = tr[(\mathbf{X}_u\mathbf{\Omega} - \mathbf{Y}_u)^T(\mathbf{X}_u\mathbf{\Omega} - \mathbf{Y}_u)] + \lambda\|\mathbf{\Omega}\|^2 + \gamma\|\mathbf{\Omega} - \mathbf{\Omega}'\|^2. \qquad (18)$$

The error function is similar to (16), but another constraint is introduced on the difference of regression coefficients. The minimum is achieved at

$$\mathbf{\Omega} = (\mathbf{X}_u^T\mathbf{X}_u + (\lambda + \gamma)\mathbf{I})^{-1}(\mathbf{X}_u^T\mathbf{Y}_u + \gamma\mathbf{\Omega}'). \qquad (19)$$

The parameter γ controls how much change on the regression parameters are allowed from the last estimation. More details of the parameter γ is explained in Section 6. To take into account the bias terms all the function estimations are performed using centered data.

5. Invariant Object Detection

In [16], it was argued that scanning of the whole image for detecting anatomic structures in medical images is unnecessary since the problem domain offers strong contextual information for localizing the targets. Utilizing a similar idea, we present a method to build an invariant detection algorithm by integrating a class specific tracking function to an existing pose dependent detector. We demonstrate the approach for affine invariant detection of faces.

We perform a sparse scan of the image, and determine all the possible object locations with a pre-learned class specific tracking function (e.g. tracker for faces). The tracker finds all the locations in the motion space (e.g. affine) which resemble the object model. The object detector is then evaluated only at these locations.

The benefits of the approach is two-fold. Firstly, the size of the search space drastically reduces. For example, we only consider a tracker which can correctly estimate translational motions upto $1/4$ of the object size. Then it is possible to scan the image with jumps equal to $1/2$. The ratio of number of search locations compared to the brute force approach decreases exponentially with the dimensionality of the motion model. Secondly, the proposed method performs continuous estimation of the target pose, whereas the existing techniques perform search on a quantized space, e.g. rotations of $\pi/6$. Utilizing a pose dependent object detection algorithm (e.g., frontal faces in upright position), the method enables to detect objects in arbitrary poses.

Instead of learning a tracking function of the specific target object (4), we train a regression function of the object class. For instance, we consider a face tracker. The learning is performed on the training set generated by applying a total of n random affine transformations to l face images.

The training is an offline process and a more complicated model can be learned compared to tracking applications. However, the learned function should be evaluated fast at runtime, since the tracker is initiated at several locations for each test image. We consider two models for learning. First model is the ridge regression which was explained in Section 4.

As the second model, we consider the regression forest which is a bagged model of tree regressors [6, p.266-270].

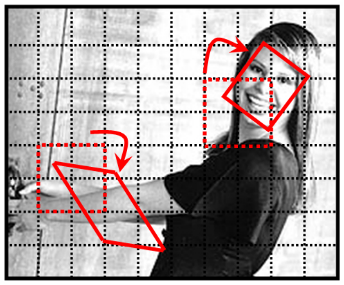


Figure 5. The trackers are initialized at sparse regular grid (dashed boxes). Final locations (solid boxes) are evaluated with face detector and the nonface regions are rejected.

Given a training set, we learn a binary tree model where each leaf node is a motion vector on the Lie algebra. The inner nodes make binary comparisons of two feature dimensions of the descriptor (out of 288), and based on the result split the space into two. During learning, 100 randomly selected features are evaluated at each node and the best pair which minimizes the sum of squared approximation error (11) on the divided spaces is selected. The growing of the tree at a node ends if there is a single sample inside or the depth of the tree reaches 15. The value of the leaf nodes with multiple samples (the nodes with depth 15) are assigned to the mean. In general, a single tree model performs poorly since the variance of the model is very high. To reduce the variance, a bagged model is learned which consists of 100 binary tree regressors. Each of them is trained on a different training set of randomly generated motions. The estimation of the random forest is the average of the 100 motions estimated by the regression trees. The tree model is evaluated fast at runtime since each tree performs an estimation by at most 15 binary comparisons.

To detect faces in a given test image the trackers are initialized at sparse set of locations, on a regular grid with $1/2$ jumps of the window size (minimum 24×24) and scales of factor 2, until the image size. Each tracker is iterated $K = 20$ times and the final locations are evaluated with Viola and Jones face detector [14] (Figure 5).

6. Experiments

We present several experiments both on affine tracking and object detection.

6.1. Affine Tracking

In the *first experiment*, we compare the Lie algebra based parametrization with the linearization (5) around the identity matrix [2, 7, 15] by measuring the estimation errors. We also compare orientation histograms with the intensity difference features used in optical flow estimation and tracking [1, 2, 7].

We generated a training set of $n_{tr} = 200$ samples by random affine transformations of a single object. The motions

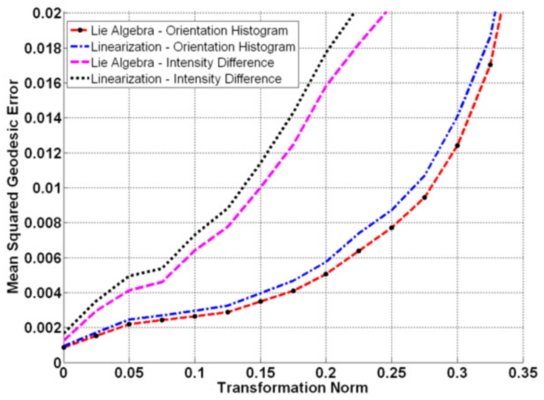


Figure 6. Estimation errors of the Lie algebra and the linearization methods using orientation histograms and intensity features.

are generated on the Lie algebra, by giving random values between -0.2 and 0.2 to each of the six parameters, and mapped to affine matrices via exponentiation. The function f is estimated by ridge regression with $\lambda = 2.10^{-3}$ for orientation histograms and $\lambda = 5.0$ for intensity features, determined by cross validation.

Each test set consists of $n_{te} = 1000$ samples. The samples inside a set have fixed norm. The norms $\|\log(\Delta\mathbf{M})\|$ vary from 0.025 to 0.35. We perform a single tracking iteration by each method, and measure the mean squared geodesic error (MSGE)

$$\frac{1}{n_{te}} \sum_{j=1}^{n_{te}} \rho^2 \left[f(\mathbf{o}_0^j), \Delta\mathbf{M}_j \right] \qquad (20)$$

between the estimations and the true values (Figure 6). The estimation based on the Lie algebra is better than the linearization for transformation of all norms. The ratio is almost constant and on the average the linearization have 12% larger error. The estimations with orientation histograms are significantly better than the intensity based features.

In the *second experiment*, we show tracking examples for several challenging sequences. In *all* the experiments, the parameters of the ridge regression were $\lambda = \gamma = 2.10^{-3}$, which were learned offline via cross validation. The training dataset is generated on the Lie algebra, by giving random values between -0.1 and 0.1 to each of the six parameters. Notice that, although we track the targets with an affine model, almost none of the targets are planar. Therefore, an affine model can not perfectly fit the target but produces the best affine approximation.

In Figure 7, a ball having large motions is tracked. The Lie algebra (first row) based estimation accurately tracks the target, whereas using linearization (second row) the target is lost after a few seconds. In the following sequences, only Lie algebra based estimations are shown.

Since nonplanar objects undergo significant appearance variations due to pose changes, the model update becomes important. In Figure 8, we show the effect of the update

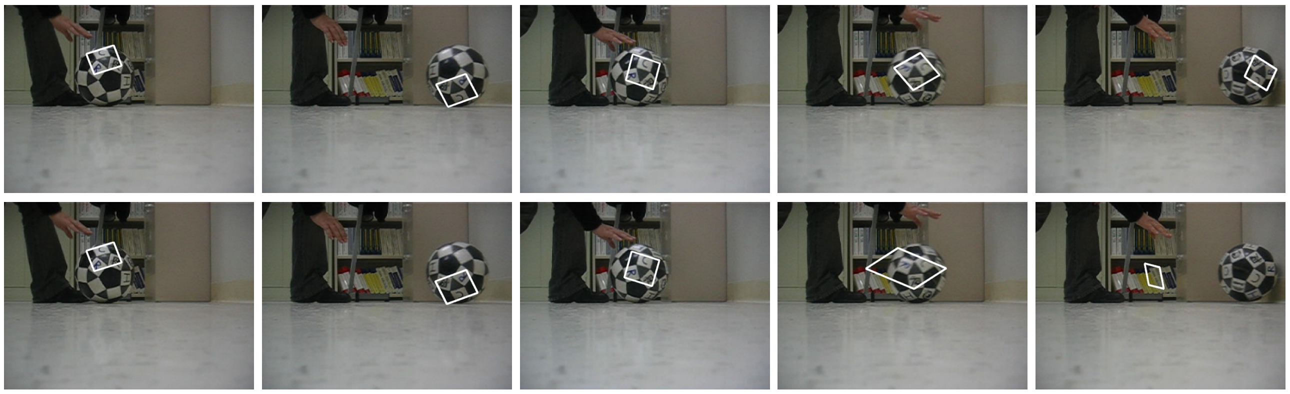


Figure 7. Comparison of the Lie algebra (*first row*) and the linearization (*second row*) based estimations. The target has large motions and the linearization based estimation loses the target after a few seconds. The sequence contains 318 frames.

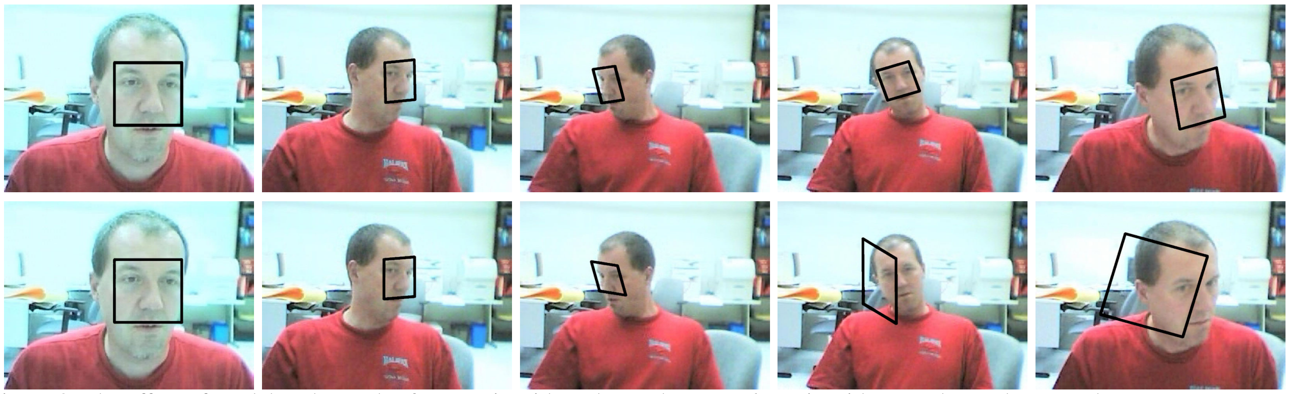


Figure 8. The effect of model update. The *first row* is with update. The *second row* is without update. The target has severe appearance variations due to illumination and pose change throughout the sequence. The sequence contains 324 frames.

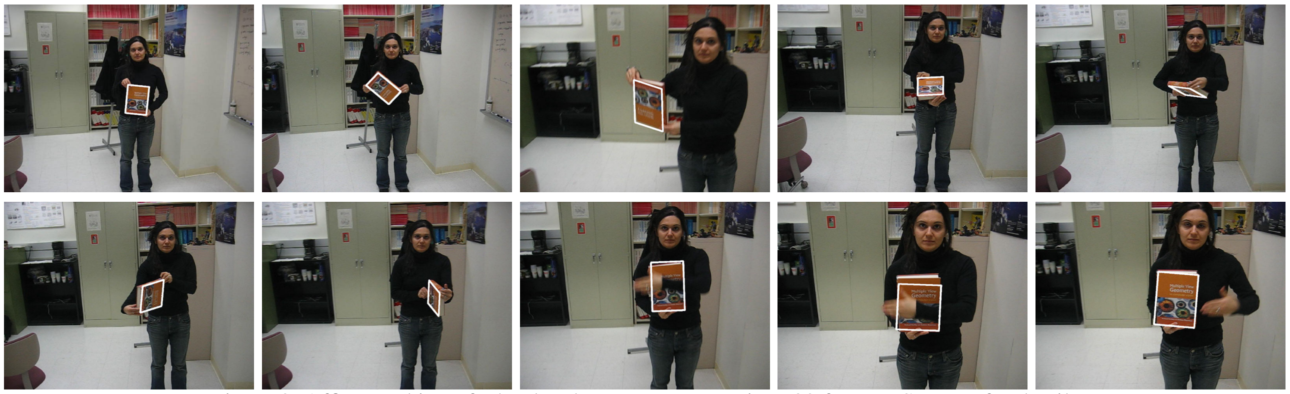


Figure 9. Affine tracking of a book. The sequence contains 739 frames. See text for details.

method on the estimations. The sequence has large appearance variations, but the update method adapts to these changes (first row), whereas without update (second row) the target is lost.

In Figure 9 we show an affine tracking example of a book. The target has large on-plane and off-plane rotations, translations, scale changes and occlusions. The estimations are accurate, which shows the robustness of the proposed approach. Please refer to the supplementary material for videos and more examples.

6.2. Invariant Object Detection

We perform detection experiments on a face dataset which consists of 803 face images from CMU, MIT and MERL datasets. The dataset is divided into 503 images for training and 300 for testing. The training set consists of 25150 samples which are generated by applying 50 transformations having a random norm between 0 to 1.0 to each face image. In contrast to the previous section, each of the six generators on the Lie algebra are weighted. For example, on average a norm 1.0 transformation corresponds to a combined translation of 25% of object size, rotation of $\pi/4$ degrees, and scaling and sheer ratio of 1.7.

In the *first experiment*, we compare the ridge and regression forest models either parameterized on the Lie algebra or based on the linearization. For each test image, we initialize five times the tracking window with a fixed norm random transformation (between 0.0 to 1.2) from the original location. For each initialization we perform 20 iterations of tracking. At the final location, we measure the squared geodesic error (20) from the original location. The mean squared geodesic errors are given in Figure 10a. The Lie algebra based parametrization is significantly better for both regression models, especially for large transformations. The best result is given by the regression forest model.

In the *second experiment*, we compare the detection performances of the models. Using the setting of the first experiment, at the final locations we evaluate the face detector (Figure 10b). The Viola and Jones (VJ) face detector [14] evaluated at the original location of the target could detect 96.7% of the faces, and the detection rate suddenly falls to 5% at locations which are norm 0.5 distant. The Lie algebra based estimations and the regression forest model are significantly superior. For norms between 0.0 to 1.0, the average miss rate for regression forest using Lie algebra is 4.24% which is almost as good as detector applied at the original location. For small transformations, the results are even better than the detector evaluated at the original location, where the small misalignments in the test set are corrected. On average, Lie algebra based parametrization have 50% less miss rate for regression forest and 24% less for ridge regression, compared to linearization. In Figure 11, several examples of initial and final locations found by the tracker are shown for various face images using regression forest model for transformations of norm 1.2.

In Figure 12 we show face detection examples for several challenging images utilizing both the original VJ detector and the proposed method. For an image of size 320×240, the VJ detector evaluates 58367 locations for translation and scale search, whereas the proposed method evaluates face detector at only 642 locations searched on the affine space.

6.3. Computational Requirement

The model is implemented on a Pentium D 2.80Ghz processor with 2.00GB of RAM using C++. The proposed tracking algorithm including model learning and update can process 60 frames per second.

The training of detection models requires 2 minutes for the ridge regression model and around 3 hours for the random forest model. At runtime the most expensive operation is the affine warping of the regions to the object coordinates and computing the object descriptors. For a 320×240 image, the tracker is initialized at 642 locations and 20 tracker iterations are performed which requires 12840 warping operations. Since the warps at each iteration can be performed in parallel we implemented the warping on GPU using NVIDIA GeForce 8800 GTX graphics card and CUDA SDK. The search for an image of size 320×240 takes 0.85 and 2.4 seconds with the linear and the regression forest models respectively.

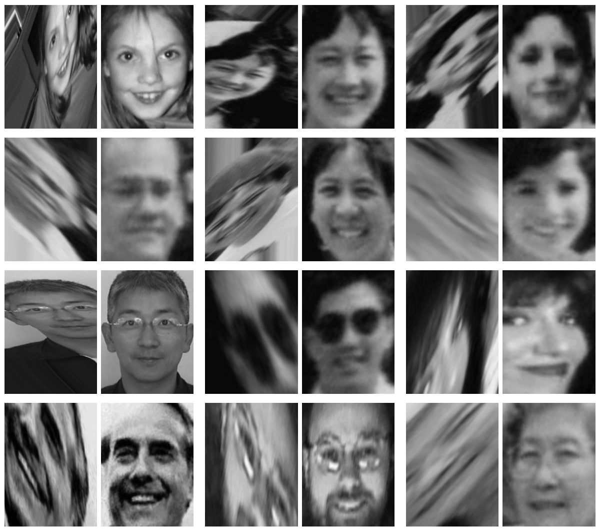


Figure 11. Face tracking. Columns 1, 3 and 5 are the initial windows and 2, 4 and 6 are the recovered locations.

7. Conclusion

We have presented an accurate learning based tracking algorithm which is combined with object detection. A new formulation for learning is derived using the Lie algebra of the motion group which significantly reduces the estimation errors. Several experiments performed on tracking and detection examples demonstrate the superior performance of the approach.

References

[1] E. Bayro-Corrochano and J. Ortegon-Aguilar. Lie algebra template tracking. *Proceedings of the 17th International Conference on Pattern Recognition*, 2:56–59, 2004.

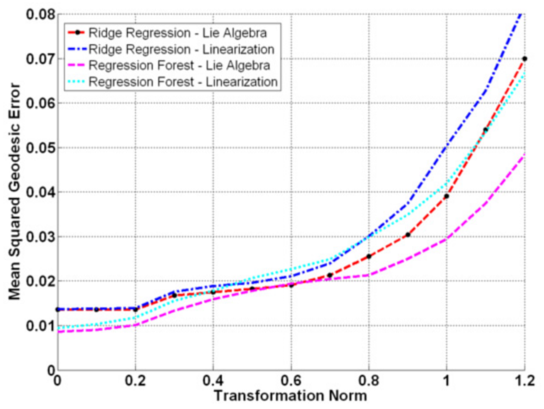
[2] T. Cootes, G. Edwards, and C. Taylor. Active appearance models. In *Proc. European Conf. on Computer Vision,* Freiburg, Germany, pages 484–498, 1998.

[3] B. C. Davis, P. T. Fletcher, E. Bullitt, and S. Joshi. Population shape regression from random design data. In *Proc. 11th Intl. Conf. on Computer Vision,* Rio de Janeiro, Brazil, 2007.

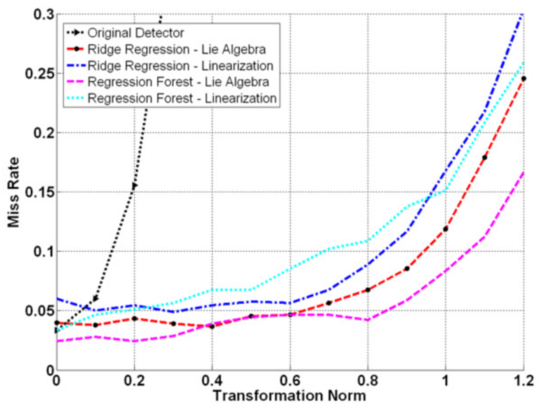
[4] T. Drummond and R. Cipolla. Application of Lie algebras to visual servoing. *Intl. J. of Comp. Vision*, 37:21–41, 2000.

[5] G. Hager and P. Belhumeur. Efficient region tracking with parametricmodels of geometry and illumination. *IEEE Trans. Pattern Anal. Machine Intell.*, 20:1025–1039, 1998.

[6] T. Hastie, R. Tibshirani, and J. Freidman. *The Elements of Statistical Learning*. Springer, 2001.



(a) Estimation errors



(b) Miss rates

Figure 10. Comparison of ridge and regression forest models parameterized on the Lie algebra and the linearization.

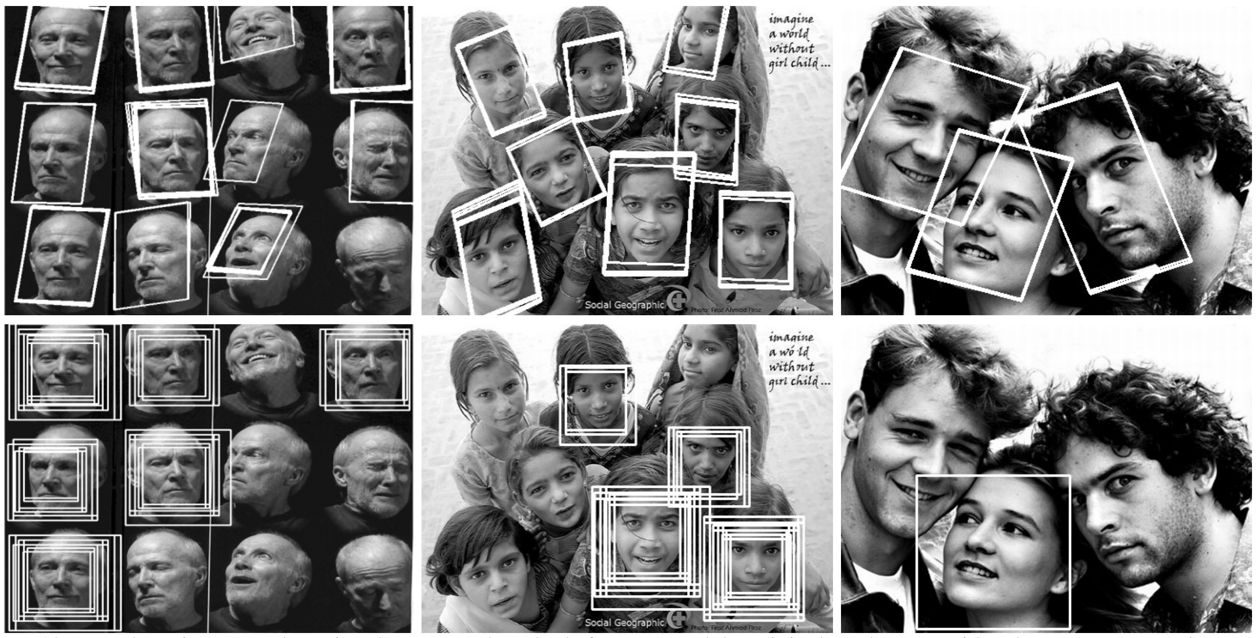


Figure 12. Face detection examples using the proposed method (*first row*) and the original VJ detector with a dense scan (*second row*). The original VJ detector is very sensitive to pose changes.

[7] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *IEEE Trans. Pattern Anal. Machine Intell.*, 24:996–1000, 2002.

[8] D. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. J. of Comp. Vision*, 60(2):91–110, 2004.

[9] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[10] I. Matthews and S. Baker. Active appearance models revisited. *Intl. J. of Comp. Vision*, 60:135–164, 2004.

[11] W. Rossmann. *Lie Groups: An Introduction Through Linear Groups*. Oxford Press, 2002.

[12] H. Shum and R. Szeliski. Construction of panoramic image mosaics with global and local alignment. *Intl. J. of Comp. Vision*, 16:63–84, 2000.

[13] O. Tuzel, R. Subbarao, and P. Meer. Simultaneous multiple 3D motion estimation via mode finding on Lie groups. In *Proc. 10th Intl. Conf. on Computer Vision,* Beijing, China, volume 1, pages 18–25, 2005.

[14] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Kauai, HI, volume 1, pages 511–518, 2001.

[15] O. Williams, A. Blake, and R. Cipolla. Sparse Bayesian learning for efficient visual tracking. *IEEE Trans. Pattern Anal. Machine Intell.*, 27:1292–1304, 2005.

[16] S. Zhou, J. Zhou, and D. Comaniciu. A boosting regression approach to medical anatomy detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition,* Minneapolis, MN, 2007.