# Regressed Importance Sampling on Manifolds for Efficient Object Tracking

Fatih Porikli
Mitsubishi Electric Research Laboratories
Cambridge, MA
fatih@merl.com

Pan Pan
University of Illinois at Chicago
Chicago, IL
ppan3@uic.edu

## Abstract

*In this paper, a new integrated particle filter is proposed for video object tracking. After particles are generated by importance sampling, each particle is regressed on the transformation space where the mapping function is learned offline by regression on pose manifold using Lie algebra, leading to a more effective allocation of particles. Experimental results on synthetic and real sequences clearly demonstrate the improved pose (affine) tracking performance of the proposed method compared with the original regression tracker and particle filters.*

## 1. Introduction

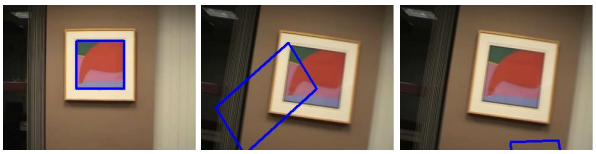
Two decades of diligent effort shows that object and pose tracking is still one of the most challenging tasks in computer vision. It faces with many difficulties. For instance, imaging projections cause loss of essential 3D information, thus, estimating 3D pose from 2D correspondences becomes an ill-posed problem. Objects frequently encounter deformations and significant appearance changes in real world scenarios. They partially or fully occlude each other for extended periods of time. They exhibit complex and erratic motion patterns, which invalidates common inertial assumptions. To make everything more complicated, the scene illumination varies perpetually, noise corrupts images irrecoverably, and cameras move and vibrate inexpediently. Different tracking methods are proposed to try to these problems, e.g. particle filter [6], kernel methods [3, 7], regression tracker [11].

Particle filter [1, 6, 12, 9] is a Monte Carlo (MC) method known also as bootstrap filtering, survival of the fittest, and the condensation algorithm. The key idea is to represent the posterior density function (of the pose object for instance) by a set of random samples with associated weights and to compute estimates based on these samples and weights. It can be shown that according to the Bayesian theory, the weighted average of these particles converges to the true state when the number of samples is large. Yet, this is computationally infeasible. In theory, the particle filter can track any parametric variation including the pose as in [8] where the affine motion is imposed as the state and particle filtering is applied on affine group. However, the intrinsic dependency to random sampling tends to degenerate and debilitate the estimated likelihoods especially for higher dimensional pose representations. Moreover, the computational requirements exponentially grow by the number of state variables, which makes the direct application of particle filter unsuitable for tracking of complex pose changes.

On the other hand, regression methods attempts to estimate the state by learning a transformation from feature space to state space. There were attempts using the Lie algebra of transformations for tracking problems. In [2], the additive updates were performed on the Lie algebra for template tracking. However, the approach in [2] 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 [4], a kernel regression model for manifold valued data is described for analyzing shape changes of the brain on MR images. This approach is computationally expensive and is not well suited for real time applications. More recently, a learning based tracking on Lie algebra is presented in [11]. This method minimizes a first order approximation to geodesic error by fitting a regression function, and reports satisfactory pose tracking results especially when the object motion (particularly translation) is not large. Yet, the existing regression approaches are limited to small variations where the object kernels in successive frames overlap significantly.

Figure 1 shows typical failure cases of the regression tracker introduced in [11] and a standard particle filter as in [1] where both methods use the same low level shape based appearance features. It is evident that the persevering large affine motion of the `Wall` sequence causes the regression tracker to fail since the regression kernels would not overlap closely (Fig. 1-b). In the meanwhile, the particle filter cannot track the input region basically due to the insufficient number of particles (200) for the 6-dimensional affine state space (Fig. 1-c). Note that, increasing the num-

978-0-7695-3718-4/09 $25.00 © 2009 IEEE
DOI 10.1109/AVSS.2009.95



(a) (b) (c)

Figure 1. Typical results demonstrating tracking performance for large motion: (a) Frame 0. (b) Estimated region by the regression tracker [11] at the frame 1. (c) Result of the particle filter [1] with limited number of (200) particles.

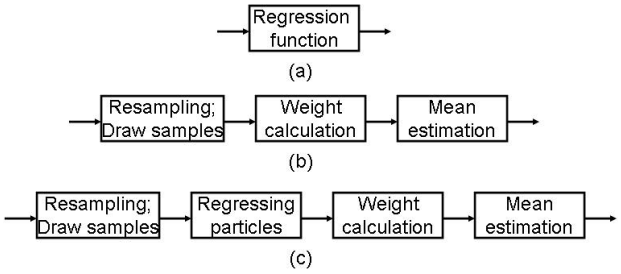



Figure 2. Schematics of three tracking algorithms: (a) Regression tracker. (b) Particle filter. (c) Our proposed method.

ber of particles cripples the computational speed.

To overcome the shortcomings of the above techniques, we propose a novel method that inherits the individual advantages of both particle filtering and regression tracking. Since the affine motion imposes a manifold, specifically a Lie group structure, our formulation employs geodesic distance and mean computation on Lie algebra. Our contributions are threefold: 1) Unlike the regression only approach, our combined method robustly estimates large parametric variations thanks to the importance sampling in the particle filter. 2) Without inflating the number of particles, our method accurately computes the particle likelihoods thanks to the additional refinement provided by the regression. This keeps the computational requirements at minimum (real-time), which makes our method suitable for tracking of complex pose changes in high dimensional state spaces. 3) The adopted low-level features (histograms of oriented gradients) make pose tracking in monocular sequences possible.

## 2. Regressing Particles on Lie Algebra

Schematics of the regression tracker, particle filter, and our combined method are presented in Fig. 2.

At the initialization of the object, the regression tracker estimates a function that maps the region feature vectors to the hypothesized affine motion vectors by first hypothesizing a set of random motion vectors within the given bounds, determining the transformed regions for these motions, and then computing the corresponding features within each warped region. In the tracking time, it extracts the feature vector only for the previous object region and applies the learned regression function, which is a matrix multiplication in case of linear regression is used.

Particle filter first draws new particles by importance sampling in state space, computes likelihood values at each particle, and finally calculates the mean as the estimate.

Our method, on the other hand, refines the particle states by the learned regression function before the mean computation, which helps aligning the corresponding image regions better in case any of the particles partially overlaps with the tracked object, and iteratively determines the mean in the tangent space [10] instead of the Euclidean coordinates as the affine motion constitutes a manifold.

### 2.1. Object Model

The object state in this paper is demonstrated on affine motions [5], however, it generalizes to any matrix Lie group transformations including 3D pose estimation. A two-dimensional affine transformation $A(2)$ is given by a $3 \times 3$ matrix $\mathbf{M}$

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

where $\mathbf{M}$ is determined by six parameters: $\mathbf{A}$ is a nonsingular $2 \times 2$ matrix of rotation, scale, and skewness, and $\mathbf{T} \in \mathbb{R}^2$ is translation.

$\mathbf{M}$ maps a unit square at the origin to the affine region enclosing the target object

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

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.

### 2.2. Particle Filtering on Affine Motions

For particle filtering on affine motions, the state of a sample is its transformation matrix, where six parameters describe scale, orientation, skewness, and translation.

We denote the $i^{th}$ sample at time $t$ as $\mathbf{M}_t^i$, and its weight as $w_t^i$. The observation $I_t$ is the given image at time $t$. The samples $\{\mathbf{M}_t^i, i = 1, 2, \ldots, n\}$ are generated from a proposal density $q(\cdot)$. The weights at time $t$ are updated by

$$w_t^i \propto w_{t-1}^i \frac{p(I_t|\mathbf{M}_t^i)p(v_t^i|\mathbf{M}_{t-1}^i)}{q(\mathbf{M}_t^i|\mathbf{M}_{t-1}^i, I_t)}, \tag{3}$$

where $p(I_t|\mathbf{M}_t^i)$ is the likelihood of the $i^{th}$ particle; $q(\mathbf{M}_t^i|\mathbf{M}_{t-1}^i, I_t)$ is the proposal density from which the particles $\mathbf{M}_t^i$ have been generated; $p(\mathbf{M}_t^i|\mathbf{M}_{t-1}^i)$ is the transition probability, and is determined by the dynamics (motion history) of the object. This can be learned through the data

even though it is usually considered as a random walk. The normalized weights $\pi^i$ are given by

$$\pi_t^i = \frac{w_t^i}{\sum_{j=1}^n w_t^j}.$$

Since the state matrix $\mathbf{M}$ do not conform to Euclidean geometry, the state estimate $\hat{\mathbf{M}}_t$ is approximated by the weighted *intrinsic mean* as follows [10]

- initialize $\hat{\mathbf{M}}_t = \mathbf{M}_t^1$
- **repeat**
  - for $i = 1$ to $n$
    compute $\mathbf{m}_t^i = log(\hat{\mathbf{M}}_t^{-1}\mathbf{M}_t^i)$
  - compute $\Gamma\hat{\mathbf{M}}_t = exp(\sum_{i=1}^n \pi_t^i \mathbf{m}_t^i)$
  - assign $\hat{\mathbf{M}}_t = \hat{\mathbf{M}}_t \Gamma\hat{\mathbf{M}}_t$
- **until** $\|log(\Gamma\hat{\mathbf{M}}_t)\| < \epsilon$

The disadvantage of particle filtering for large affine motions is that the importance sampling is required to have relatively larger variance values to compensate for large motions, yet, an estimation with small number of particles performs inefficiently in 6-dimensional state space. Figure 1-b shows the performance of the particle filter, which is inferior since a set of 200 particles is rarely enough to populate the affine probability density. The number of particles grows exponentially along with the increase of the dimensions for more complex pose changes.

### 2.3. Regression Tracker

The regression tracker was proposed in [11]. It estimates 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

$$\mathbf{M}_t = \mathbf{M}_{t-1}.\Delta\mathbf{M}_t \tag{4}$$

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, that is, an orientation histogram. It is denoted by $\mathbf{h}(\mathbf{M}^{-1}) \in \mathbb{R}^m$ where $m$ is the dimension of the descriptor. Given the previous location of the object $\mathbf{M}_{t-1}$ and the current observation $I_t$, the new transformation $\Delta\mathbf{M}_t$ is estimated by the regression function

$$\Delta\mathbf{M}_t = f(\mathbf{h}_t(\mathbf{M}_{t-1}^{-1})). \tag{5}$$

The tracking problem reduces to learning and updating the regression function $f$ to estimate the pose of the object.

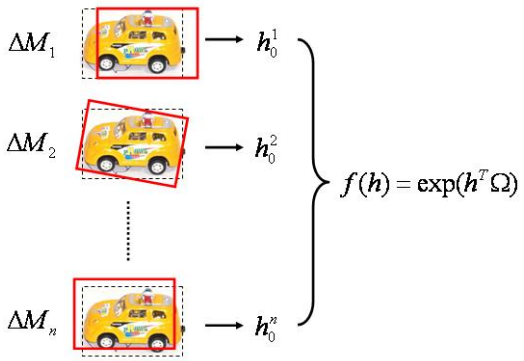



Figure 3. Learning regression function at initialization.

Figure 3 illustrates the way we learn the regression function before tracking. During initialization, $t = 0$, the observation $I_0$ and the initial location of the object $\mathbf{M}_0$ are given. A training set of $n$ random affine transformation matrices $\{\Delta\mathbf{M}_i\}_{i=1\ldots n}$ is generated 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 $\mathbf{h}_0^i = \mathbf{h}_0\left(\Delta\mathbf{M}_i^{-1}.\mathbf{M}_0^{-1}\right)$. The transformation $\Delta\mathbf{M}_i$ moves the object back to the unit square. The training set consists of samples $\left\{\mathbf{h}_0^i, \Delta\mathbf{M}_i\right\}_{i=1\ldots n}$. 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 $t$.

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

$$\mathbf{X} = \begin{pmatrix} \left[\mathbf{h}_0^1\right]^T \\ \vdots \\ \left[\mathbf{h}_0^n\right]^T \end{pmatrix} \quad \mathbf{Y} = \begin{pmatrix} \left[\log\left(\Delta\mathbf{M}_1\right)\right]^T \\ \vdots \\ \left[\log\left(\Delta\mathbf{M}_n\right)\right]^T \end{pmatrix}. \tag{6}$$

The regression function can be found as

$$f(\mathbf{h}) = \exp\left(\mathbf{h}^T\mathbf{\Omega}\right), \tag{7}$$

where $\mathbf{\Omega} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$ and $\lambda$ prevents from feature coefficients to become over dominant to others (refer to [11] for details on the ridge regression).

After learning the model by randomly generating motion parameters within given scale, rotation, and translation bounds in the first frame, the update process is very fast at the consecutive frames as it requires only simple matrix multiplications. Learning the model does not need to be repeated at each frame as long as the object undergoes a change is inside the bounds, e.g. human face does not turn all the way around.

Even though the regression can estimate the affine transformations in a computationally efficient way by itself, it

needs significant overlap of the object window (thus, recoverable states) between the adjacent frames. As we mentioned before, it fails in case of large motion changes where such an overlap could not be satisfied, e.g. Fig. 1-a.

### 2.4. Regressing Particles

Our method first draw samples from a normal probability density function,

$$\mathbf{M}_{t,m}^{i} = \mathbf{M}_{t-1}^{i} + \mathcal{N}(0, \mathcal{C}). \tag{8}$$

where $\mathbf{M}_{t,m}^{i}$ stands for the intermediate samples of $\mathbf{M}_{t}^{i}$; $\mathcal{N}(0, \mathcal{C})$, i.e. a Gaussian distribution with zero mean and covariance matrix $\mathcal{C}$.

To refine the position of the particle in the space such that the underlying domain fits better to the object model, we apply the regression function to each intermediate particles as follows

$$\mathbf{M}_{t}^{i} = \mathbf{M}_{t,m}^{i} \Delta\mathbf{M}_{t}^{i} = \mathbf{M}_{t,m}^{i} f(\mathbf{h}_{t}(\mathbf{M}_{t,m}^{-1})). \tag{9}$$

In order to capture pose changes, we utilize the histogram of gradient as the cue to calculate the likelihood of each particles. Therefore, $p(I_t|\mathbf{M}_t^i)$ in 3 is defined as

$$p(I_t|\mathbf{M}_t^i) = e^{-\frac{D^2}{2\sigma^2}}, \tag{10}$$

where $D$ is the dissimilarity, e.g. by Bhattacharya distance, of the gradient weighted orientation histograms between the 2D image region corresponding to $\mathbf{M}_t^i$ and the template.

Ideally we like to get a closed form expression of the proposal density $q(\mathbf{M}_t^i|\mathbf{M}_{t-1}^i, I_t)$ based on the way we generate the samples, i.e. (8) and (9). However, the regression tracking is a mapping from the feature density to transformation space. The low-level feature we utilized, the gradient histogram $\mathbf{h}_t(\mathbf{M}_{t,m}^{-1})$ in (9), prevents deriving a closed form expression. Instead, we assume that the sampling of the particles matches the transformation dynamics of the object, i.e. $p(\mathbf{M}_t^i|\mathbf{M}_{t-1}^i) = q(\mathbf{M}_t^i|\mathbf{M}_{t-1}^i, I_t)$. Then (3) reduces to

$$w_t^i \propto w_{t-1}^i p(I_t|\mathbf{M}_t^i) \tag{11}$$

A pseudo-code of the proposed algorithm is given in Fig. 4.

In principle, there is direct correlation between the way we draw samples from the proposal density function in particle filtering and generate the random motions for the regression function. The particle filter is designed to compensate for rather large translations than rotations and scale assuming the recoverable translation in regression part is constrained by the object size, but not in particle filter.

## 3. Experimental Results

We performed extensive tests on both synthetic sequences as well as real-world videos. In each test, we compared our approach with the regression tracker and the particle filter (different versions). All three algorithms were implemented in MATLAB 8.0 on a 3.0GHz Intel Duo processor. The summary of algorithm performance comparisons is given in Table 1.

**Step 1: Resampling and Generate particles**
∗For i=1:n

- Resampling
- Draw particles from a Gaussian distribution (8) $\mathbf{M}_{t,m}^{i} = \mathbf{M}_{t-1}^{i} + \mathcal{N}(0, \mathcal{C})$.

∗End
**Step 2: Regression for each particle**
∗For i=1:n

- $k = 1$ and $\mathbf{M}_t^i = \mathbf{M}_{t,m}^i$
- Repeat
  - $\mathbf{M}_t^i = \mathbf{M}_t^i.\Delta\mathbf{M}_t^i$
  - $k = k + 1$
- Until $\Delta\mathbf{M}_t = \mathbf{I}$ or $k = K$

∗End
**Step 3: Estimate the state**
∗For i=1:n

- weight calculation (11)
- mean estimate

∗End

Figure 4. pseudocode of the proposed algorithm.

Table 1. Overall evaluation.

| Tracker type | Pose estimation | Large motion |
|---|---|---|
| Regression tracker | Good | Poor |
| Particle filter | Poor | Good |
| Our method | Good | Good |

In order to perform a fair comparison, we kept the proposal density functions in both the conventional particle filter and our method same. In addition, the number of particles in the conventional particle filter is empirically selected such that the CPU time of our method is less and almost equal to particle filter. Gradient information is used as the only cue to calculate the likelihood in the particle filter and the region descriptors in the regression tracking. We applied the histogram of oriented gradients (HOG) descriptors with 288 coefficients. Similar to SIFT descriptors, 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 $2\pi$. The size of each histogram is eight dimensional and the descriptors, $\mathbf{h}$, 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

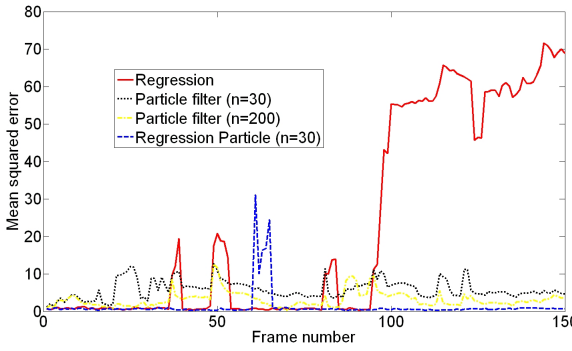



Figure 5. MSE of tracking algorithms on `Logo` sequence.

the unit square and construct the descriptor inside the inner rectangle.

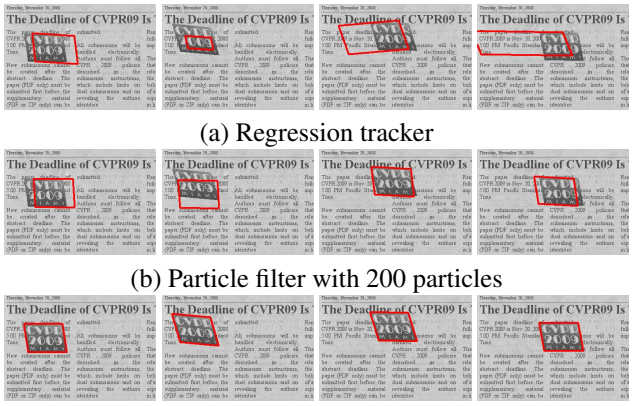
For synthetic sequences where the ground truth affine parameters are available we performed a single tracking iteration by each method, and simply measured the mean squared error (MSE) on all six parameters instead of the geodesic distance between the estimations and the true values. Notice that, although we track the targets with an affine model, some targets are not planar. Even though an affine model cannot perfectly fit the non-planar targets, we observed it still produces the best affine approximations.

**Cluttered Scene**: In the synthetic sequence `Logo`, the logo of CVPR09 wander erratically on a text background. Note that, this sequence is a challenging not only because of the motion but also due to the fact that we do not use the intensity features but only the gradients in a this highly cluttered image. As shown in Fig. 6, the regression tracker fails to compensate for the pose when the logo jumps in the image (MSE=22.93) after frame 97. The results for the particle filter with 30 particles are not accurate either with MSE=5.89. Even though the particle filter with 200 particles managed to approximate the translations it gives worse MSE score (3.21) and visual results than the proposed method (1.25). The proposed method tracked the logo robustly using only 30 particles and it was able to recover from the small error in frame 62. In our advantage, the average CPU time is lower (2.29 to 2.55) than the 200 particle implementation. We presented the average CPU times and errors in Table 2 and the corresponding frame-wise results in Fig. 5.

**Large Motion and Scale Changes**: For the real-world sequences given in Figures 7, 10, 8 we visually observed even better results when we applied the proposed method. All sequences contain large and erratic motion, scale changes, even tracking the picture in `Wall` is not straightforward as it shows two spatially close concentric rectangular regions moves together, which makes scale estimate much harder. Note that, we do not need to make use of the color information available in this sequence.

Table 2. Statics of tracking algorithms on `Logo` sequence. CPU times are for MATLAB implementation.

| Algorithm | CPU time per frame (sec) | Average MSE |
|---|---|---|
| Regression tracker | 0.25 | 22.93 |
| Particle Filter with 30 particles | 0.47 | 5.89 |
| Particle Filter with 200 particles | 2.55 | 3.21 |
| Our method using 30 particles | 2.29 | 1.25 |

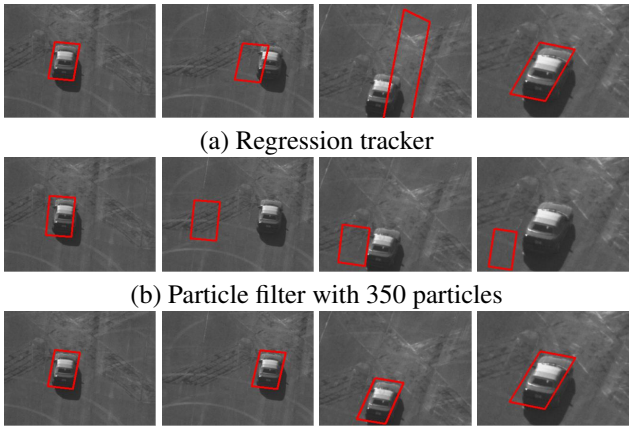


(a) Regression tracker

(b) Particle filter with 200 particles

(c) Our method using 30 particles

Figure 6. An example of tracking in cluttered challenging environment. Results of (a) Regression tracker (b) Particle filter with 200 particles (c) Our method using 30 particles on `logo` sequence.



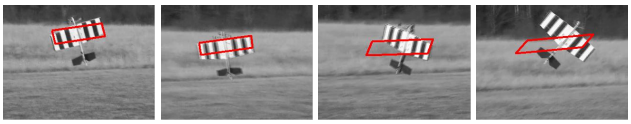
(a) Regression tracker

(b) Particle filter with 350 particles

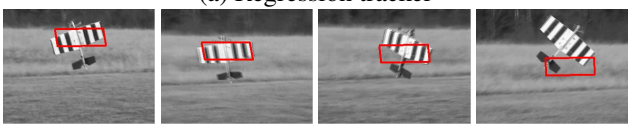
(c) Our method using 50 particles

Figure 7. An example of tracking with big size change of the object. Results of (a) Regression tracker (b) Particle filter with 350 particles (c) Our method using 50 particles on `car` sequence.

Our experiments prove that the integration of the regression tracker into the particle filter significantly improves the performance.



(a) Regression tracker



(b) Particle filter with 220 particles

(c) Our method using 30 particles

Figure 8. Results of (a) Regression tracker (b) Particle filter with 220 particles (c) Our method using 30 particles on `toy plane` sequence.

(a) Regression tracker

(b) Particle filter with 350 particles

(c) Our method using 50 particles

Figure 9. Results of (a) Regression tracker (b) Particle filter with 350 particles (c) Our method using 50 particles on `face` sequence. Appearance of the face changes as well as its pose.

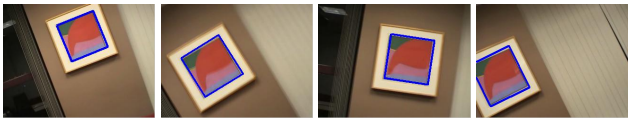


Figure 10. Results of our method using 30 particles on `wall` sequence. This sequence **is not** very simple; two spatially close concentric rectangular regions moves together in the sequence, which makes scale estimate much harder. Also note that, we do not make use of any color information.

## 4. Conclusions

We presented a simple but elegant method that integrates the regression pose estimate on Lie algebra into the sequential importance sampling particle filter. Our method provides more accurate results than the conventional particle filter with tenfold particles, and has the ability of recovering large translational motion unlike the conventional regression tracker on Lie algebra. Our method is not restricted to the affine motion and can be easily extended to more complex parametric motions.

As a future study, we will apply it to 3D pose estimation problems.

## References


[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174C188, 2002.

[2] E. Bayro-Corrochano and J. Ortegon-Aguilar. Lie algebra template tracking. In *ICPR*, 2004.

[3] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000.

[4] B. C. Davis, P. T. Fletcher, E. Bullitt, and S. Joshi. Population shape regression from random design data. In *ICCV*, 2007.

[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] M. Isard and I. Blake. Condensation – conditional density propagation for visual tracking. In *Intl. J. of Comp. Vision*, volume 29, pages 5–28, 1998.

[7] A. Jepson, D. Fleet, and T. Elmaraghi. Robust online appearance models for visual tracking. *IEEE Trans. Pattern Anal. Machine Intell.*, 25:1296–1311, 2003.

[8] J. Kwon and F. C. Park. Visual tracking via particle filtering on the affine group. In *ICIA*.

[9] P. Pan and D. Schonfeld. Dynamic proposal variance and optimal particle allocation in particle filtering for video tracking. *IEEE Trans. Circuits and Systems for Video Technology*, 18, 2008.

[10] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on Lie algebra. In *CVPR06*, 2006.

[11] O. Tuzel, F. Porikli, and P. Meer. Learning on lie groups for invariant detection and tracking. In *CVPR*, 2008.

[12] S. Zhou, R. Chellappa, and B. Moghaddam. Visual tracking and recognition using appearance-based modeling in particle filters. In *ICME*, 2003.