Object Tracking and Person Re-Identification on Manifolds

Conrad Sanderson

NICTA, Australia

- e-mail: conrad.sanderson [at] nicta.com.au
- web: http://conradsanderson.id.au

Presented at:

Fatih Porikli, Mehrtash Harandi, Conrad Sanderson.
Tutorial on Riemannian Geometry in Computer Vision.
Asian Conference on Computer Vision (ACCV), 2014.
Part 1: Object Tracking on Manifolds

Published in:

  **Bags of Affine Subspaces for Robust Object Tracking.**

Object tracking is hard:

- occlusions
- deformations
- variations in pose
- variations in scale
- variations in illumination
- imposters / similar objects
Tracking algorithms can be categorised into:

1. **generative tracking**
   - represent object through a particular appearance model
   - search for image area with most similar appearance
   - examples: mean shift tracker \[1\] and FragTrack \[2\]

2. **discriminative tracking**
   - treat tracking as binary classification task
   - discriminative classifier trained to explicitly separate object from non-object areas
   - example: Multiple Instance Learning (MILTrack) \[3\]
   - example: Tracking-Learning-Detection (TLD) \[4\]
   - requires larger training dataset than generative tracking

---

Promising approach for generative tracking:

→ model object appearance via **subspaces**

- originated with the work of Black and Jepson \[5\]
- apply eigen decomposition on a set of object images
- resulting eigen vectors define a linear subspace
- subspaces able to capture perturbations of object appearance

Many developments to address limitations:

- sequentially update the subspace \(^6\)[7]
- more robust update of the subspace \(^8\)[9][10]
- online updates using distances to subspaces on Grassmann manifolds \(^11\)

But still not competitive with discriminative methods!

\(^7\) Yongmin Li: On incremental and robust subspace learning. In: Pattern Recognition 37.7 (2004).
Two major **shortcomings** in all subspace based trackers:

1. **mean** of the image set is not used
   - the mean can hold useful discriminatory information!

2. search for object location is typically done using **point-to-subspace distance**
   - compare a candidate image area from ONE frame against the model (multiple frames)
   - easily affected by drastic appearance changes (e.g., occlusions)
Minimum point to subspace distance
Proposed Tracking Approach

Comprised of 4 intertwined components:

1. particle filtering framework (for efficient search)

2. model appearance of each particle as an **affine subspace**
   - takes into account tracking history (longer memory)
   - takes into account the mean

3. object model: **bag of affine subspaces**
   - continuously updated set of affine subspaces
   - longer memory
   - handles drastic appearance changes

4. likelihood of each particle according to object model:
   (i) distance between means
   (ii) distance between bases: **subspace-to-subspace distance**
1. Particle Filtering Framework

- Using standard particle filtering framework \cite{12}
- History of object’s location is parameterised as a distribution
  - set of particles represents the distribution
  - each particle represents a location and scale:
    \[ z_i^{(t)} = [x_i^{(t)}, y_i^{(t)}, s_i^{(t)}] \]
- Use distribution to create a set of candidate object locations in a new frame
- Obtain appearance of each particle: \( A_i^{(t)} \)
- Choose new location of object as the particle with highest likelihood according to object model \( B \):
  \[ z_{*}^{(t)} = z_j^{(t)}, \quad \text{where} \quad j = \arg\max_i p \left( A_i^{(t)} | B \right) \]

2. Model Appearance of Each Particle as an Affine Subspace

- Affine subspace represented as a 2-tuple:
  \[ A_i(t) = \{ \mu_i(t), U_i(t) \} \]
  \( \mu \): mean
  \( U \): subspace basis

- Appearance includes:
  1. appearance of the \( i \)-th candidate location
  2. appearance of tracked object in several preceding frames
Previously Tracked Frames

Frame 1
Frame 2
Frame 3
Frame t-2
Frame t-1
Frame t

(Candidate #1)
(Candidate #2)
(Candidate #3)

Object Model

Minimum distance

Candidates

3. Object Model: Bag of Affine Subspaces

- Drastic appearance changes (e.g. occlusions) adversely affect subspaces.
- Instead of modelling the object using only one subspace, use a **bag of subspaces**:

\[ \mathcal{B} = \{ A_1, \cdots, A_K \} \]

- Simple **model update**: the bag is updated every \( W \) frames by replacing the oldest affine subspace with the newest.
4. Likelihood of Each Particle According to Object Model

- Particle filtering framework requires: \( p \left( A_i^{(t)} | B \right) \)
- Appearance of each candidate area: \( A_i^{(t)} = \{ \mu_i^{(t)}, U_i^{(t)} \} \)
- Object model: \( B = \{ A_1, \cdots, A_K \} \)
- Our definition: \( p \left( A_i^{(t)} | B \right) = \sum_{k=1}^{K} \hat{p} \left( A_i^{(t)} | B[k] \right) \)
  - \( B[k] \) is the \( k \)-th affine subspace in bag \( B \)
  - \( \hat{p} \left( A_i^{(t)} | B[k] \right) = \frac{p \left( A_i^{(t)} | B[k] \right)}{\sum_{j=1}^{N} p \left( A_j^{(t)} | B[k] \right)} \), where \( N = \text{num. of particles} \)
  - \( p \left( A_i^{(t)} | B[k] \right) \approx \exp \left\{ - \text{dist}(A_i^{(t)}, B[k]) \right\} \) distance between affine subspaces
Define the **distance** between two affine subspaces as:

$$\text{dist}(\mathcal{A}_i, \mathcal{A}_j) = \alpha \hat{d}_o (\mu_i, \mu_j) + (1 - \alpha) \hat{d}_g (U_i, U_j)$$

- $\hat{d}_o (\mu_i, \mu_j) =$ normalised Euclidean distance between means
- $\hat{d}_g (U_i, U_j) =$ normalised geodesic distance between bases

**Grassmann manifolds:**
- space of all $n$-dimensional linear subspaces of $\mathbb{R}^D$ for $0 < n < D$
- a point on Grassmann manifold $\mathcal{G}_{D,n}$ in a $D \times n$ matrix

Geodesic distance between subspaces $U_i$ and $U_j$ is:

$$d_g (U_i, U_j) = \| [\theta_1, \theta_2, \cdots, \theta_n] \|$$

- $[\theta_1, \theta_2, \cdots, \theta_n] =$ vector of principal angles
- $\theta_1 =$ smallest angle btwn. all pairs of unit vectors in $U_i$ and $U_j$
- principal angles are computed via SVD of $U_i^T U_j$
Computational Complexity

- Generation of new affine subspace:
  - patch size: $H_1 \times H_2$
  - represent patch as vector: $D = H_1 \times H_2$
  - use patches from $P$ frames
  - $\therefore$ SVD of $D \times P$ matrix
  - $D >> P$
  - using optimised thin SVD$^{[13]}$: $O(Dn^2)$ operations
  - $n =$ number of basis vectors

- To keep computational requirements relatively low:
  - patch size: $32 \times 32$
  - number of frames: 5
  - number of basis vectors: 3

---

Comparative Evaluation

- Evaluation on 8 commonly used videos in the literature
- Compared against recent tracking algorithms:
  - Tracking-Learning-Detection (TLD)\(^{14}\)
  - Multiple Instance Learning (MILTrack) \(^{15}\)
  - Sparse Collaborative Model (SCM) \(^{16}\)
- Qualitative and quantitative evaluation

<table>
<thead>
<tr>
<th>proposed method</th>
<th>TLD (PAMI 2012)</th>
<th>MILTrack (PAMI 2011)</th>
<th>SCM (CVPR 2012)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td><img src="image9" alt="Image" /></td>
<td><img src="image10" alt="Image" /></td>
<td><img src="image11" alt="Image" /></td>
<td><img src="image12" alt="Image" /></td>
</tr>
<tr>
<td><img src="image13" alt="Image" /></td>
<td><img src="image14" alt="Image" /></td>
<td><img src="image15" alt="Image" /></td>
<td><img src="image16" alt="Image" /></td>
</tr>
<tr>
<td><img src="image17" alt="Image" /></td>
<td><img src="image18" alt="Image" /></td>
<td><img src="image19" alt="Image" /></td>
<td><img src="image20" alt="Image" /></td>
</tr>
<tr>
<td><img src="image21" alt="Image" /></td>
<td><img src="image22" alt="Image" /></td>
<td><img src="image23" alt="Image" /></td>
<td><img src="image24" alt="Image" /></td>
</tr>
<tr>
<td><img src="image25" alt="Image" /></td>
<td><img src="image26" alt="Image" /></td>
<td><img src="image27" alt="Image" /></td>
<td><img src="image28" alt="Image" /></td>
</tr>
<tr>
<td><img src="image29" alt="Image" /></td>
<td><img src="image30" alt="Image" /></td>
<td><img src="image31" alt="Image" /></td>
<td><img src="image32" alt="Image" /></td>
</tr>
<tr>
<td><img src="image33" alt="Image" /></td>
<td><img src="image34" alt="Image" /></td>
<td><img src="image35" alt="Image" /></td>
<td><img src="image36" alt="Image" /></td>
</tr>
<tr>
<td><img src="image37" alt="Image" /></td>
<td><img src="image38" alt="Image" /></td>
<td><img src="image39" alt="Image" /></td>
<td><img src="image40" alt="Image" /></td>
</tr>
<tr>
<td><img src="image41" alt="Image" /></td>
<td><img src="image42" alt="Image" /></td>
<td><img src="image43" alt="Image" /></td>
<td><img src="image44" alt="Image" /></td>
</tr>
</tbody>
</table>
Quantitative Results

- Used two measures:
  1. **centre location error**: distance between the centre of the bounding box and the ground truth object position
  2. **precision**: percentage of frames where the estimated object location is within a pre-defined distance to ground truth

![Graphs showing mean error and mean precision for different methods: proposed, TLD, MILTrack, SCM, OAB, IVT. The y-axis represents the error or precision values, and the x-axis represents the methods. The proposed method has the lowest mean error, indicating better performance.](image)

- **average error** (lower = better)
- **average precision** (higher = better)
**Future Work**

- Affected by motion blurring (rapid motion or pose variations)
- Better update scheme by measuring the effectiveness of new affine subspace before adding it to the bag
- Allow bag size and update rate to be dynamic, possibly dependent on tracking difficulty
Part 2: Person Re-Identification on Manifolds

Published in:

- Full paper: [http://dx.doi.org/10.1109/ICIP.2013.6738731](http://dx.doi.org/10.1109/ICIP.2013.6738731)
Given images of a person from camera view 1,
find matching person from camera view 2

Difficult:

- imperfect person detection / localisation
- large pose changes
- occlusions
- illumination changes
- low resolution
Popular Previous Approaches

Partial Least Squares (PLS) based [17]

- decompose an image into overlapping blocks
- extracts features from each block: textures, edges, colours
- concatenated into one feature vector (high dimensional)
- learn discriminative dimensionality reduction for each person
- classification: projection to each model + Euclidean distance

**downsides:**

- concatenation = fixed spatial relations between blocks
- \[\therefore \text{ does not allow for movement of blocks!}\]
- \[\therefore \text{ easily affected} \] by imperfect localisation and pose variations

---

Symmetry-Driven Accumulation of Local Features (SDALF)\textsuperscript{[18]}

- foreground detection
- two horizontal axes of asymmetry to isolate: head, torso, legs
- use vertical axes of appearance symmetry for torso and legs
- extract: HSV histogram, stable colour regions, textures
- estimation of symmetry affected by deformations & pose variations:
  - \textit{noisy features}

\textsuperscript{18}M. Farenzena et al.: \textit{Person re-identification by symmetry-driven accumulation of local features}. In: CVPR (2010).
Proposed Method

- Aim to obtain a compact & robust representation of an image:
  - allow for imprecise person detection
  - allow for deformations
  - do not use rigid spatial relations
  - do not use brittle feature extraction based on symmetry

Steps:

1. foreground estimation
2. for each foreground pixel, extract feature vector containing colour and local texture information
3. represent the set of feature vectors as a covariance matrix
4. covariance matrix is a point on a Riemannian manifold
5. map matrix from R. manifold to vector in Euclidean space, while taking into account curvature of the manifold!
6. use standard machine learning for classification
Feature Extraction

- For each foreground pixel, extract feature vector:

\[ f = [x, y, HSV_{xy}, \Lambda_{xy}, \Theta_{xy}]^T \]

where

- \( HSV_{xy} = [H_{xy}, S_{xy}, \hat{V}_{xy}] \) = colour values of the HSV channels
- \( \Lambda_{xy} = [\lambda^R_{xy}, \lambda^G_{xy}, \lambda^B_{xy}] \) = gradient magnitudes
- \( \Theta_{xy} = [\theta^R_{xy}, \theta^G_{xy}, \theta^B_{xy}] \) = gradient orientations

- (not limited to above, can certainly use other features)

- Given set \( F = \{f_i\}_{i=1}^N \), calculate covariance matrix:

\[
C = \frac{1}{N-1} \sum_{i=1}^{N} (f_i - \mu)(f_i - \mu)^T
\]

- low dimensional representation, independent of image size
How to Compare Covariance Matrices?

- Naive method:
  - brute-force vectorisation of matrix
  - use Euclidean distance between resultant vectors

- Naive method kind-of works, BUT:
  - covariance matrix = symmetric positive definite (SPD) matrix
  - space of SPD matrices = interior of a convex cone in $\mathbb{R}^{D^2}$
  - space of SPD matrices = Riemannian manifold\(^{[19]}\)
  - $\therefore$ covariance matrix = point on a Riemannian manifold
  - naive method **disregards** curvature of manifold!
  - geodesic distance: shortest path along the manifold (eg. on a sphere)

How to Measure Distances on Riemannian Manifolds?

- Use Affine Invariant Riemannian Metric (AIRM) \(^{[20]}\):
  \[
  \delta_R (A, B) = \left\| \log \left( B^{-\frac{1}{2}} A B^{-\frac{1}{2}} \right) \right\|_F
  \]
  - intensive use of matrix inverses, square roots, logarithms \(^{[21]}\)
  - \(\therefore\) computationally demanding!

- Choose a tangent pole, and map all points to tangent space

  - tangent space is Euclidean space
  - faster, but less precise
  - **true geodesic distances are only to the tangent pole!**

---


Stein Divergence

- Related to AIRM, but much faster \cite{Sra2012}

\[
\delta_S(A, B) = \log(\det(\frac{A+B}{2})) - \frac{1}{2} \log(\det(AB))
\]

- divergence, not a true distance!

---

Proposed: Relational Divergence Classification

- Obtain a set of training covariance matrices \(\{T\}_{i=1}^{N}\)
- For matrix \(C\), calculate its Stein divergence to each training covariance matrix:
  \[
  \begin{bmatrix}
  \delta_S(C, T_1) & \delta_S(C, T_2) & \cdots & \delta_S(C, T_N)
  \end{bmatrix} \in \mathbb{R}^N
  \]

- In effect, we have mapped matrix \(C\) from manifold space to Euclidean space, while taking into account manifold curvature
- Can now use standard machine learning methods

---

Comparative Evaluation

- After mapping from manifold space to Euclidean space, use LDA based classifier

- Use ETHZ dataset \[23\]
  - captured from a moving camera
  - occlusions and wide variations in appearance

- Compare with:
  - directly using the Stein divergence
  - Histogram Plus Epitome (HPE) \[24\]
  - Partial Least Squares (PLS) \[25\]
  - Symmetry-Driven Accumulation of Local Features (SDALF) \[26\]

RDC = Relational Divergence Classification (proposed method)
Stein = direct use of Stein divergence (no mapping)
HPE = Histogram Plus Epitome
PLS = Partial Least Squares
SDALF = Symmetry-Driven Accumulation of Local Features
Questions?
e-mail: conrad.sanderson [at] nicta.com.au

More papers on machine learning & computer vision using manifolds:
http://conradsanderson.id.au/papers.html