Fast and accurate motion segmentation using Linear Combination of Views

Vasileios Zografos and Klas Nordberg

Computer Vision Laboratory

Linköping University, Sweden

Garnics

British Machine Vision Conference 2011, Dundee

SEVENTH FRAMEWORK PROGRAMME

What is motion segmentation

"The task of separating a sequence of images into different regions, each corresponding to a distinct, consistent 3d motion"

More specifically

- 3d **rigid** motions
- **Sparse** features in **correspondence** across **all** frames

- Simplifications:
  - Weak perspective effects
  - Number of motions assumed known

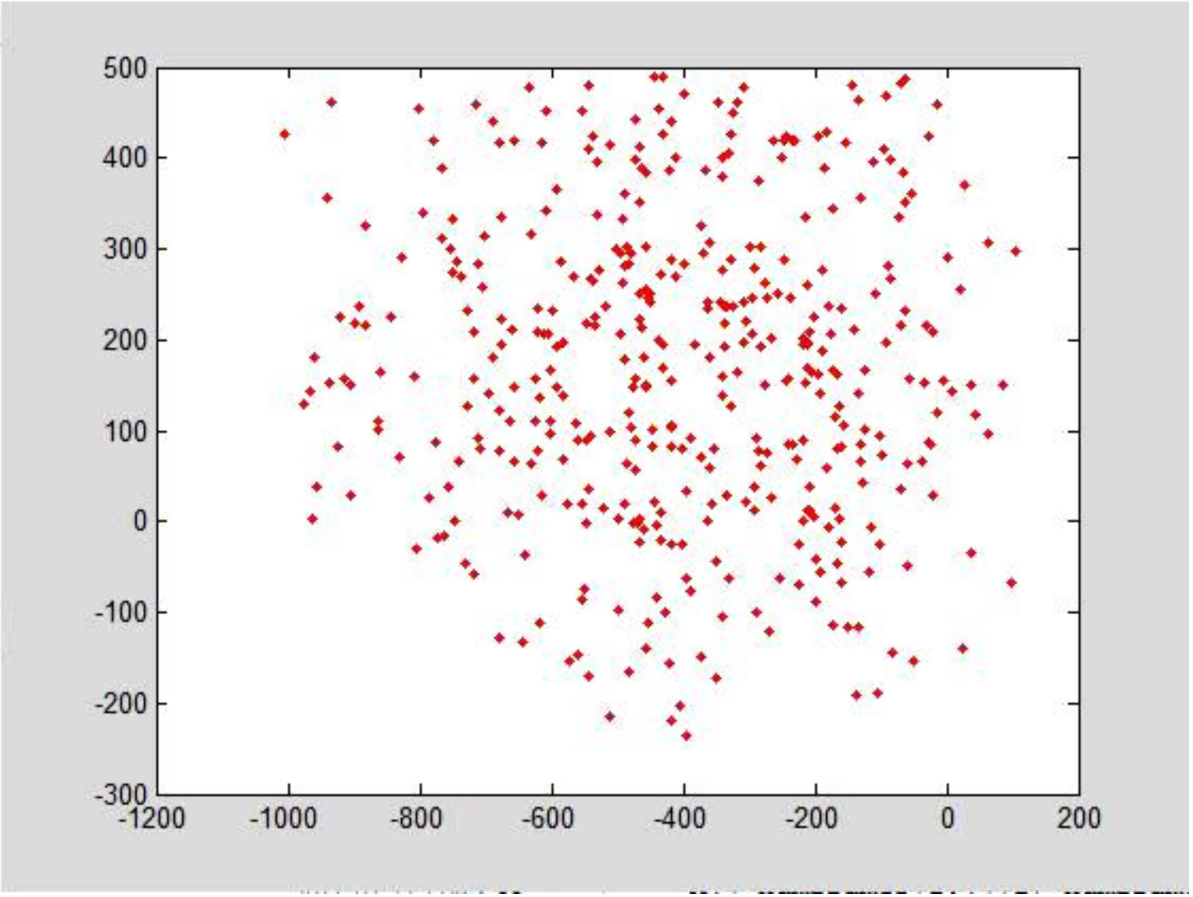
Applications of motion segmentation

- Tracking
- Multi-body structure from motion
- Navigation
- Activity / Anomaly detection
- Image semantic analysis

Why is it a difficult problem?

- Relying only on geometry (motion trajectories) can be difficult
- Humans rely also on secondary features (spatial, colour, texture)
- Such secondary features can aid segmentation
- But the core problem has to be solved in geometry

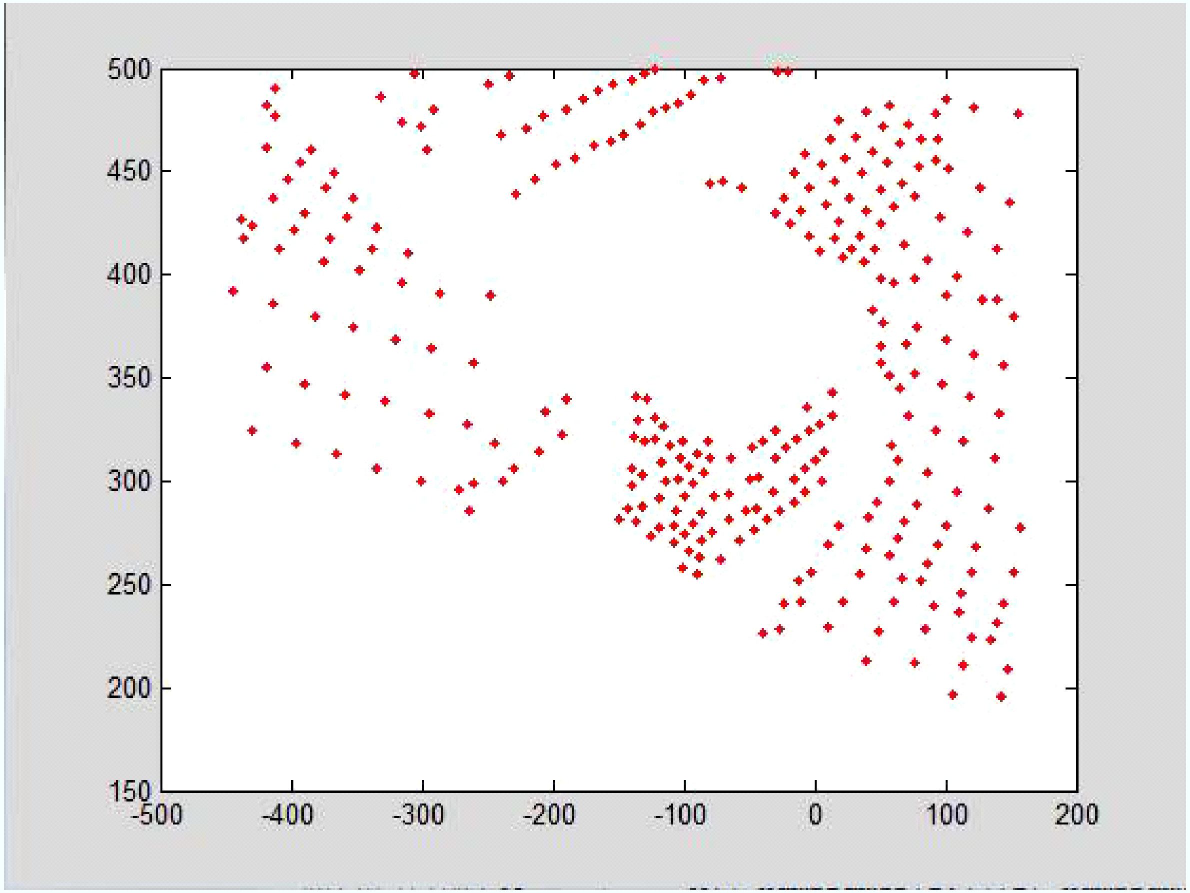
Example (motion trajectories only)



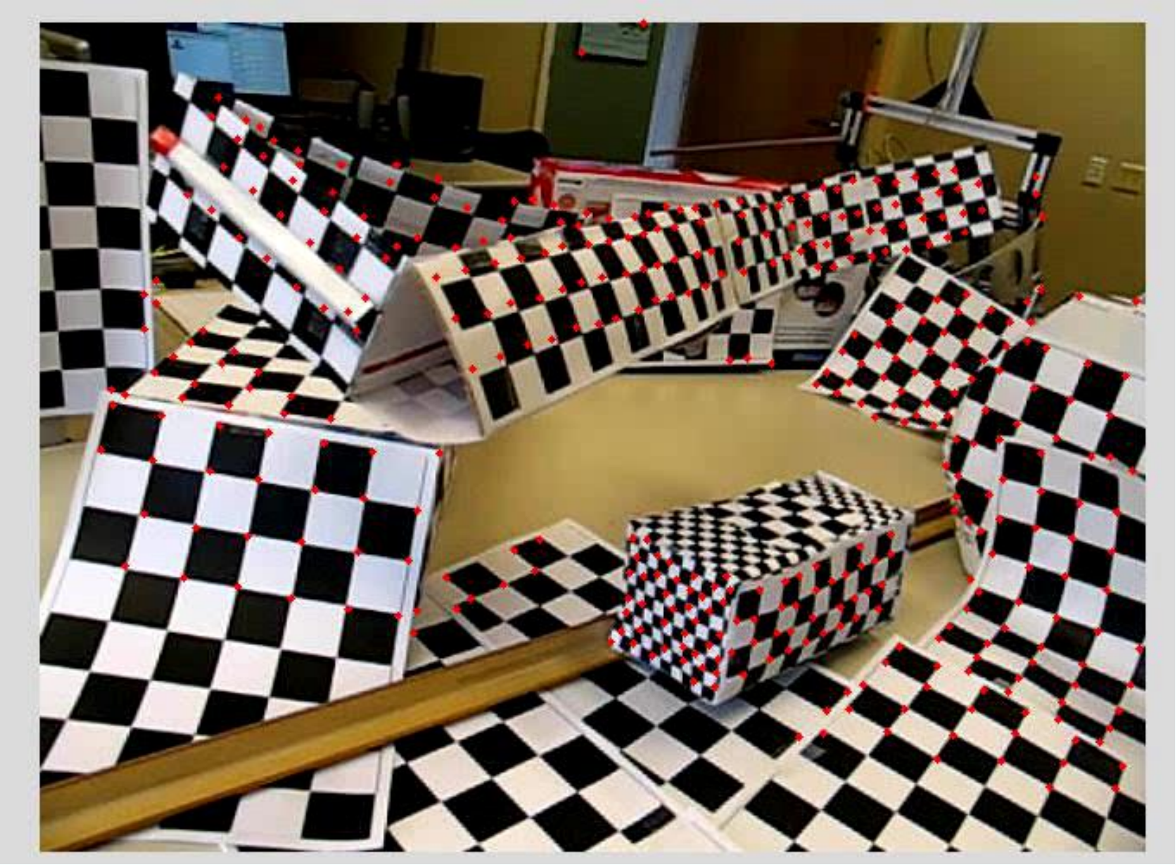
Garnics

SEVENTH FRAMEWORK PROGRAMME

Example (including spatial configuration)



Example (All features)



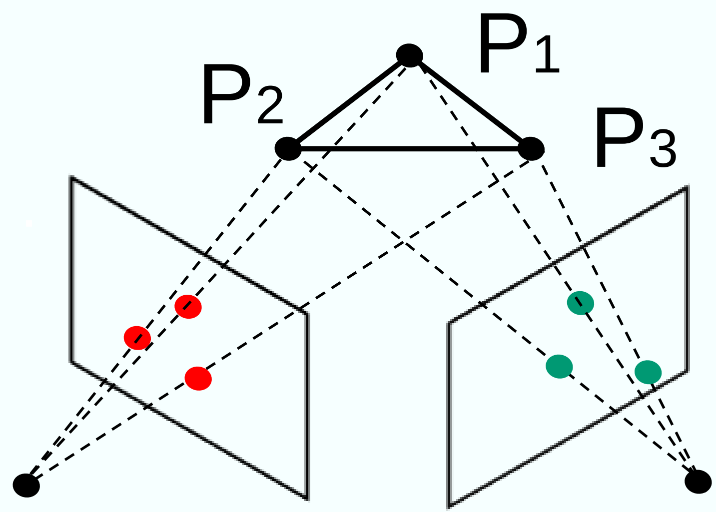
Current methods

- Most are subspace clustering methods that work in trajectory space
- Some employ spatial relations
- Differ on their definition of affinity
- Very slow or too complex

Multi-view geometry Affine camera

- 3d scene with N points on a single object

$$P_j = [X_j, Y_j, Z_j, 1]^T \quad j = 1, ..., N$$



- F images $i = 1, ..., F$
- Projected to the i[th] image $p_{ij} = m_i \; P_j$
- m_i is the 2x4 affine camera projection matrix

Multi-view geometry Affine camera: **The View**

- The set of N 2d points, in image i, define a ***view*** of the object:

$$v_i = m_i \ [P_1 \ P_2 \ ... \ P_N] = m_i S, \quad i = 1, ..., F$$

- The 4xN matrix S, is the 3d ***shape*** matrix of the object.

Multi-view geometry Affine camera: **The trajectory**

- Alternatively, a single 3d point projected onto all the images defines a trajectory:

$$t_j = \begin{bmatrix} m_1 \\ \vdots \\ m_F \end{bmatrix} P_j = M P_j, \quad j = 1, ..., N$$

- The 2Fx4 matrix M is the ***motion*** matrix

Multi-view geometry
Affine camera

- Combine everything into the 2FxN ***measurement*** matrix W:

$$W = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_N^1 \\ y_1^1 & y_2^1 & \dots & y_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^F & x_2^F & \dots & x_N^F \\ y_1^F & y_2^F & \dots & y_N^F \end{bmatrix} = M\,S$$

- Also $\mathrm{rank}(W) \leq \min\,(\mathrm{rank}(M), \mathrm{rank}(S)) \leq 4$

The Tomasi-Kanade factorisation

- Use SVD to obtain a similar factorisation of W into **shape** x **motion**
- Provides basis vectors that span the R^{2F} space of W called the ***Joint Image Space*** (JIS)
- The JIS represents a connection between 2d and 3d where the camera parameters have been eliminated

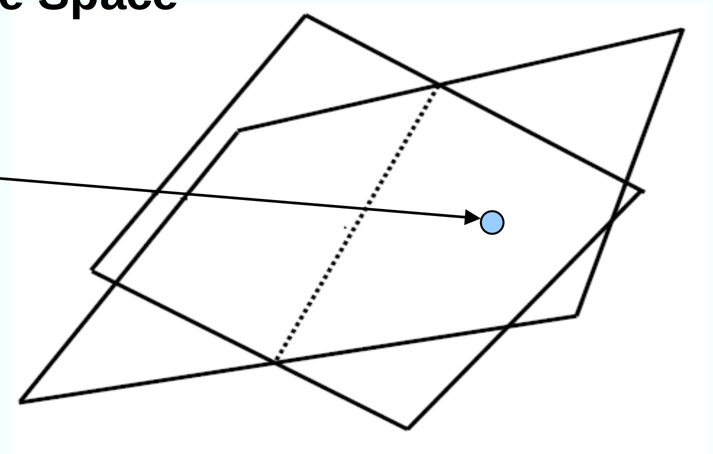
Garnics

SEVENTH FRAMEWORK PROGRAMME

The T-K factorisation for motion segmentation

$$\begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ y_1^1 & y_2^1 & \cdots & y_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^F & x_2^F & \cdots & x_N^F \\ y_1^F & y_2^F & \cdots & y_N^F \end{bmatrix}$$

Joint Image Space



- Rank 4 means a 4d linear subspace
- For k rigidly moving objects, their trajectories will lie in a union of k linear subspaces in R^{2F}
- Motion segmentation equates to subspace clustering

Ullman-Basri
Linear combination of views

- We can instead look at the row-space of W.
- Each row is a vector in R^N space called the ***Joint-Point-Space*** (JPS)

$$\begin{bmatrix} x_1^1 & x_2^1 & \dots & x_N^1 \\ y_1^1 & y_2^1 & \dots & y_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^F & x_2^F & \dots & x_N^F \\ y_1^F & y_2^F & \dots & y_N^F \end{bmatrix}$$

Half-view

The JPS represents a connection between 2d and camera parameters where the 3d shape has been eliminated

Linear combination of views

- Given any 2 basis views v_1, v_2 we can reconstruct the 3d shape S by:

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} S = M_{12}\, S \;\Rightarrow\; S = M_{12}^{-1} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

- We are not interested in S explicitly but a third view can be synthesised by:

$$v_3 = m_3\, S = m_3\, M_{12}^{-1} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Linear combination of views

- The inversion of M_{12} is valid if it has full rank. For the general case:

$$v_3 = Q \begin{bmatrix} 1 \\ v_1 \\ v_2 \end{bmatrix}, \quad Q = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ b_0 & b_1 & b_2 & b_3 & b_4 \end{bmatrix}$$

- This leads to the familiar equations:

$$x_j^3 = a_0 + a_1 x_j^1 + a_2 y_j^1 + a_3 x_j^2 + a_4 y_j^2$$
$$y_j^3 = b_0 + b_1 x_j^1 + b_2 y_j^1 + b_3 x_j^2 + b_4 y_j^2,$$

Linear combination of views

- This expression is valid for all the points on an object in the third view
- Q is not known but can be found from 5 or more corresponding points, visible in 3 views

$$\begin{bmatrix} x_1^3, \ldots, x_r^3 \\ y_1^3, \ldots, y_r^3 \end{bmatrix} = \underbrace{\begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ b_0 & b_1 & b_2 & b_3 & b_4 \end{bmatrix}}_{Q} \begin{bmatrix} 1, \ldots, 1 \\ x_1^1, \ldots, x_r^1 \\ y_1^1, \ldots, y_r^1 \\ x_1^2, \ldots, x_r^2 \\ x_1^2, \ldots, x_r^2 \end{bmatrix}$$

LCV in summary

- Given **2 basis views** of an object
- And a set of LCV **coefficients** Q
- We can synthesise a **novel view** of the object
- Applications: Object recognition and view synthesis

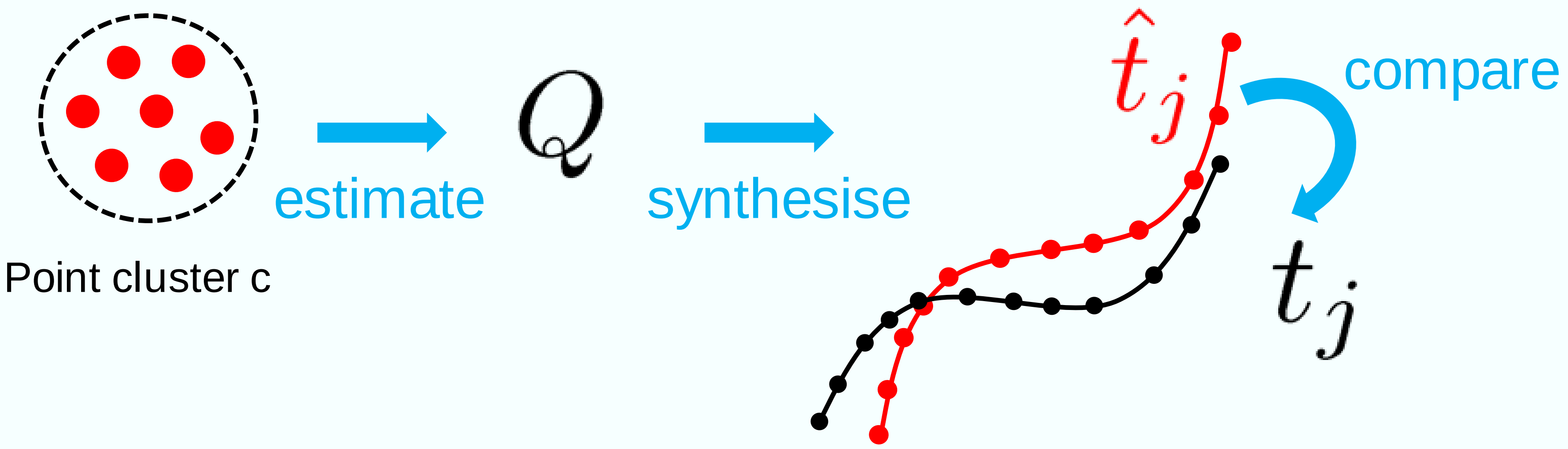
LCV for motion segmentation: What is required?

- Motion-based affinity (n-tuple or pairwise)
  - If pairwise then we can use standard clustering approaches (e.g. spectral clustering)
- Some clustering algorithm

LCV for motion segmentation: Simple concept

- Given a set Q of LCV coefficients we can synthesise a trajectory $\hat{t}_j$ of a point p_j and compare it with its real trajectory t_j
- We may then say something about p_j and the points used to estimate Q.

Illustration of concept

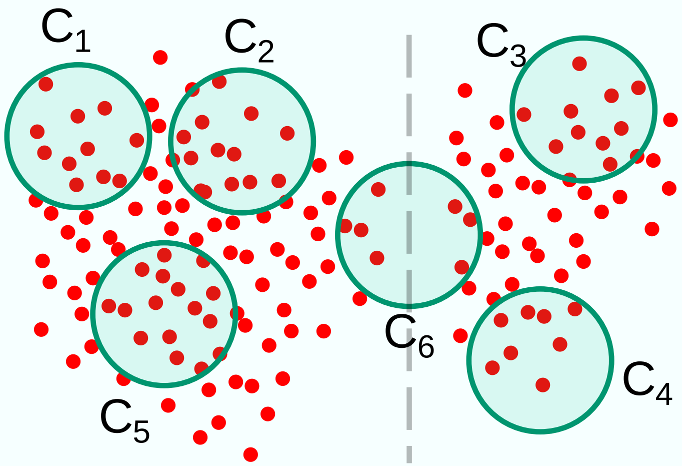


- If then $\hat{t}_j \approx t_j$ then Q describes well the motion of the point p_j
- Also if Q was estimated from points c, where $p_j \notin c$ then p_j and c probably lie on the same object

LCV for motion segmentation: 3-step algorithm outline

- Step 1: Motion hypotheses sampling
- Step 2: Trajectory synthesis + affinity generation
- Step 3: Clustering

Algorithm: Step 1 – Motion hypotheses sampling



Sample in image space. Any single frame will do.

Estimate by inverting a 5xn matrix for each

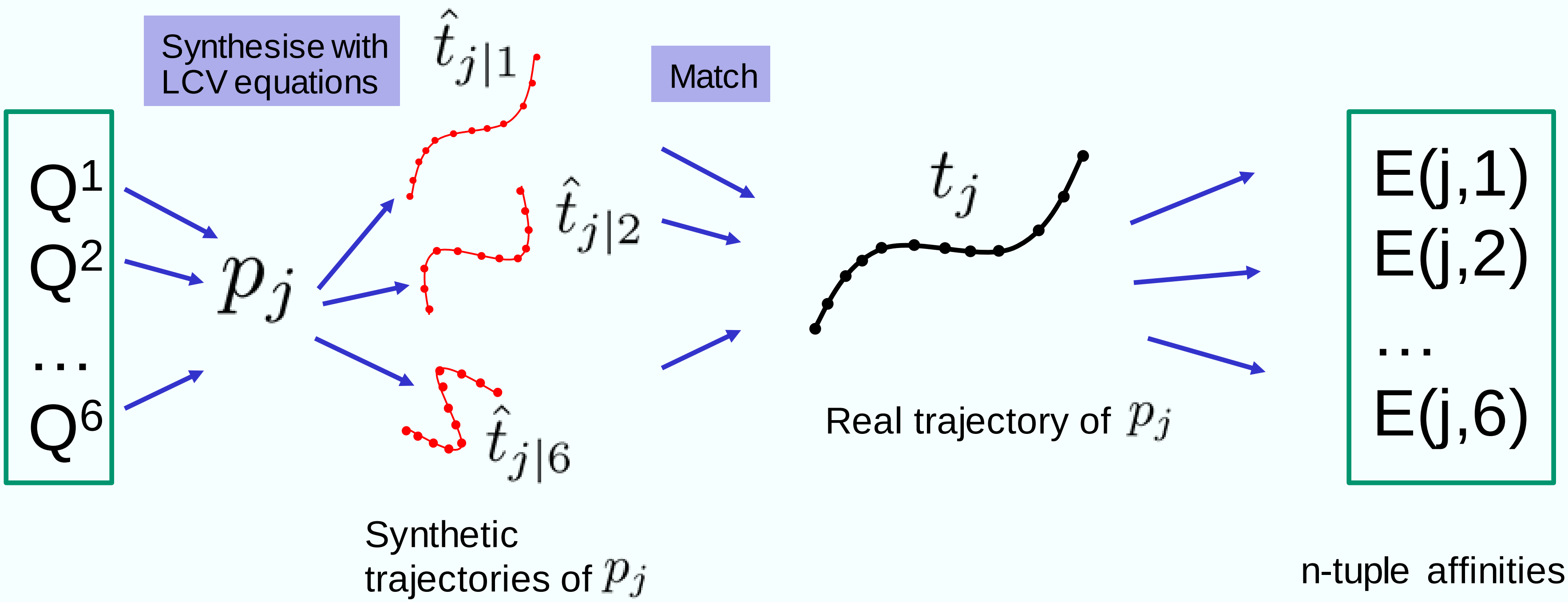
Q^1
Q^2
Q^3
$\dots$
Q^6

Each LCV coefficient matrix represents a "*motion hypothesis*".

They are not necessarily unique

- Using as basis views first and last frame of the sequence
- Sample C n-point clusters in any frame (e.g. n=7)
- n-closest point in Euclidean distance (spatial configuration implicit)
- Estimate the LCV coefficients Q^c

Algorithm: Step 2 – Trajectory synthesis and affinity matrix



Algorithm: Step 2 – Trajectory synthesis and affinity matrix

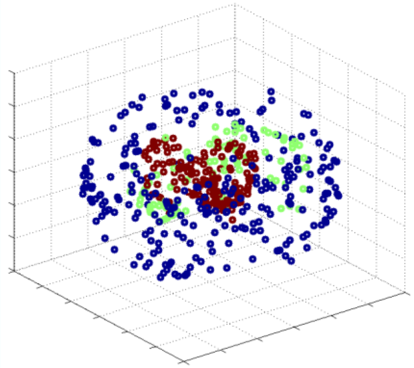
- The n-tuple affinity between the point p_j and the n-points c is defined as:

$$E(j,c) = K\left(\left\|t_j - \hat{t}_{j|c}\right\|_H / F\right)$$

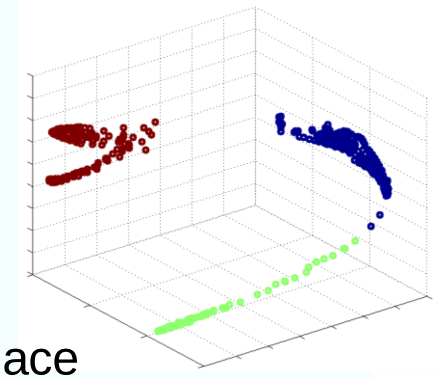
- K is a kernel function $K(x,\sigma) = \left(x^2 + \sigma^2\right)^{-1/2}$
- $\left\|.\right\|_H$ is the robust Huber norm
- Affinity is defined in image space, in pixels
- Pairwise affinity is then A $\approx$ EE^T

Algorithm: Step 3 - Spectral clustering

- Uses the eigen-structure of the affinity matrix A for dimensionality reduction and clustering
- Usually one parameter (kernel width)
- We search for the optimal kernel width



Data space



Kernel eigen-space

Garnics

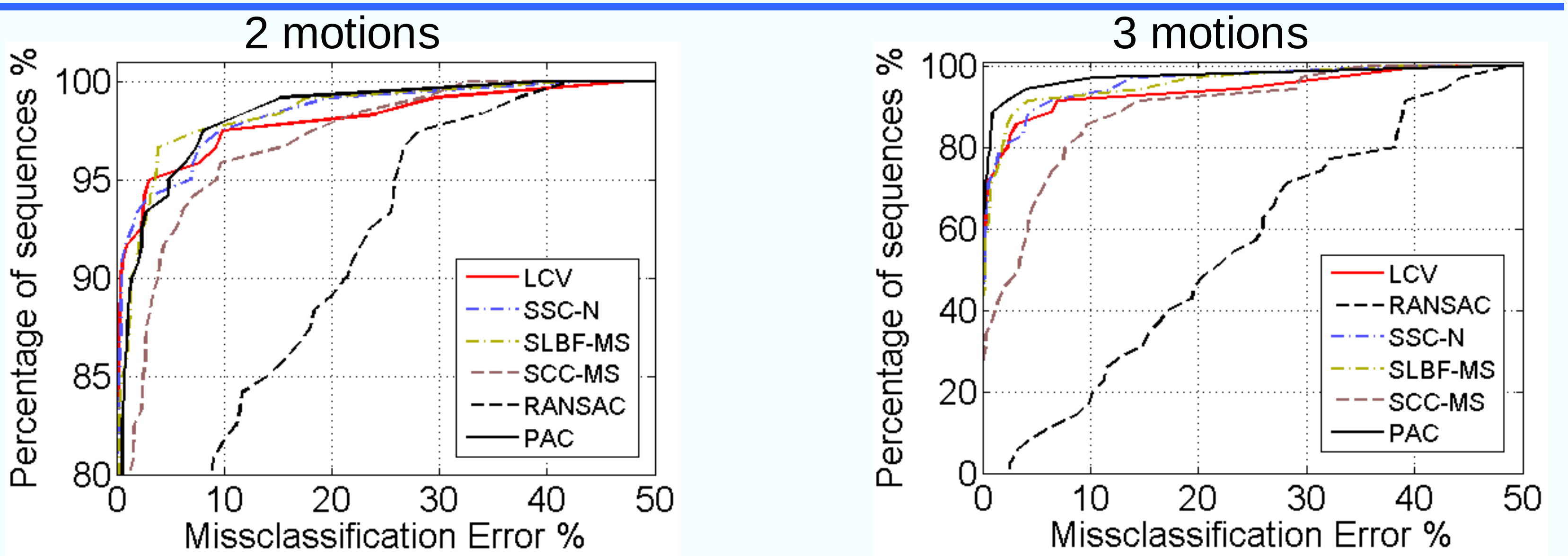
SEVENTH FRAMEWORK PROGRAMME

Experiments and comparison to other methods

- State-of-the-art methods of the last few years (SSC, SCC, SLBF, PAC). All of them use the T-K factorisation

- Hopkins155
  - Standard dataset for sparse motion segmentation methods
  - Manually refined, complete trajectories (20-30 frames, 100-500 points)
  - 155 sequences of 2 and 3 motions.
  - Varying difficulty (general, degenerate, articulated)
- No per sequence tuning. Either fixed parameters or determined automatically.
- The number of motions are assumed known

Accuracy and speed



Method	RANSAC	SCC-MS	SLBF-MS	SSC-N	PAC	LCV
Average time (sec)	0.387	1.264	10.83	165	952.25	0.93
Total time (sec)	60	196	1680	25620	147600	145
Average error (%)	9.48	2.70	1.35	1.36	1.24	1.86

The average and total runtime on the full Hokpins155 dataset

Algorithm complexity overview

N: number of scene points, F number of frames, C: number of 7-point clusters

- Step1 (motion sampling):
  - K-means
  - C*F matrix inversions of size 5x7
- Step2 (synthesis+affinity):
  - N*C*F LCV equations
  - N*C*Affinity
- Step3 (spectral clustering with kernel width search):
  - 10*eigen-decompositions of a NxN matrix
  - 10*K-means

Conclusion

- Motion segmentation method based on LCV theory for affine camera model
- Good combined accuracy and speed.
- Easy to implement and can be used in practice (fast and parameters are auto tuned)
- Outstanding issues: missing trajectories, number of objects, degeneracies