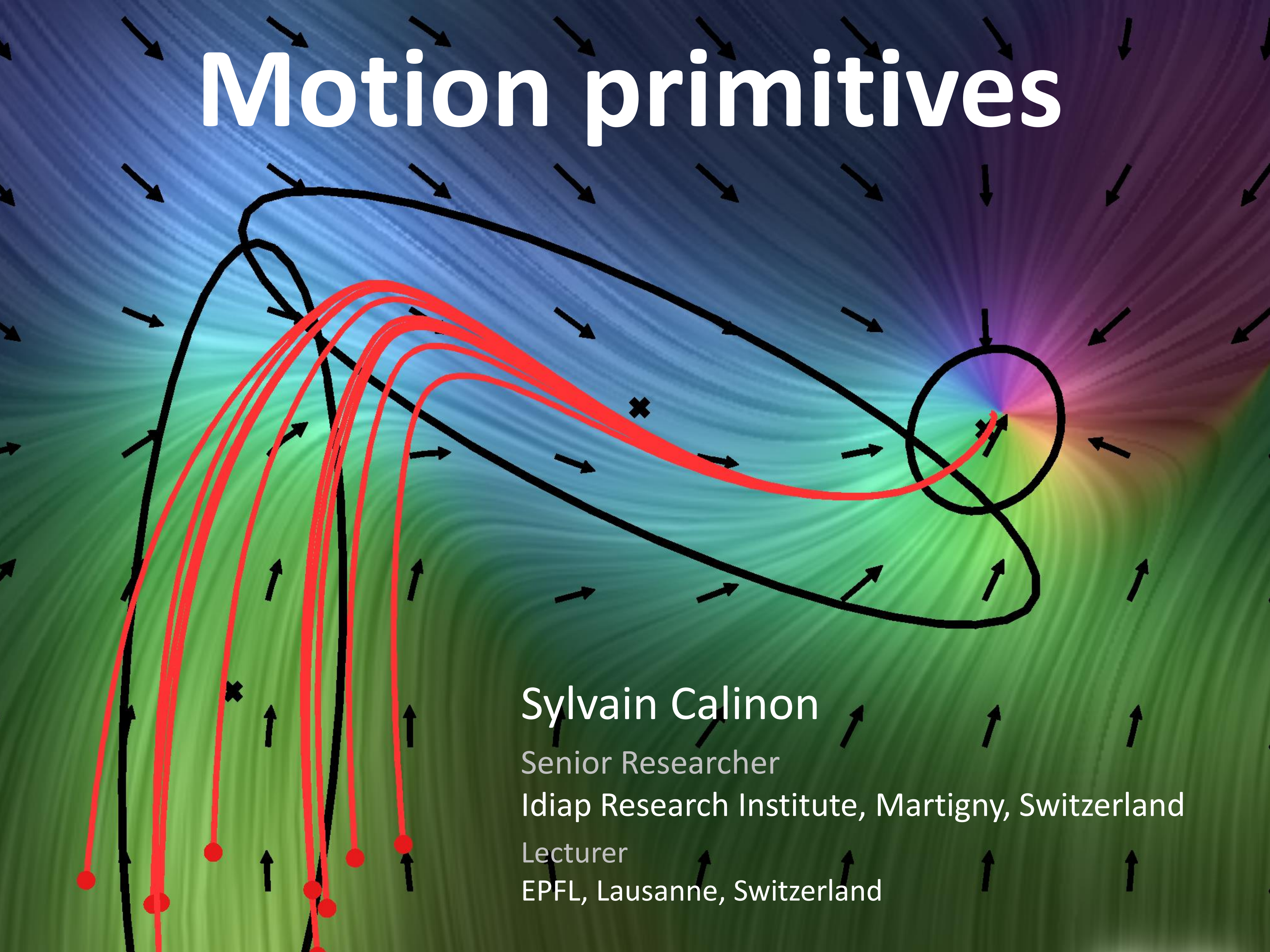
Motion primitives



Sylvain Calinon

Senior Researcher

Idiap Research Institute, Martigny, Switzerland

Lecturer

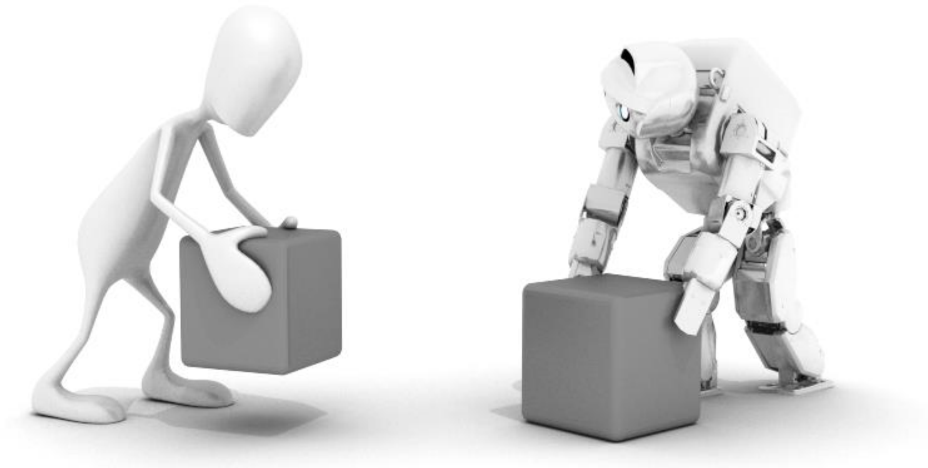
EPFL, Lausanne, Switzerland

Artificial Intelligence for Society

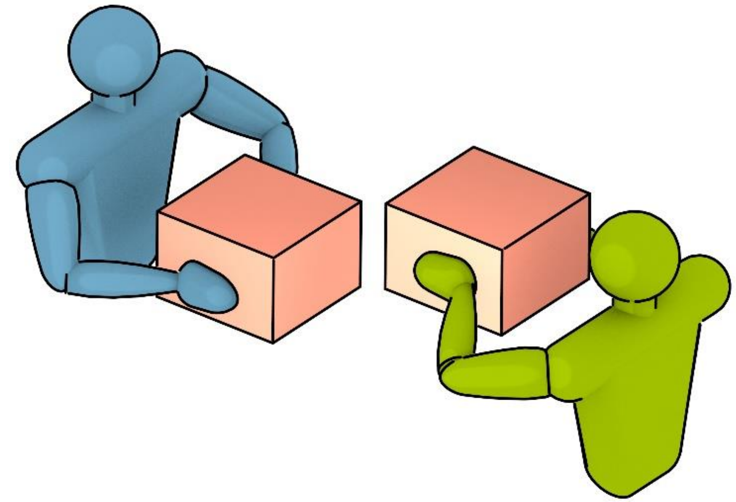
Research Groups:

- Speech & Audio Processing
- Natural Language Understanding
- Perception & Activity Understanding
- Machine Learning
- Social Computing
- Biometrics Security and Privacy
- Biosignal Processing
- Computational Bioimaging
- Energy Informatics
- Uncertainty Quantification and Optimal Design
- **Robot Learning & Interaction**

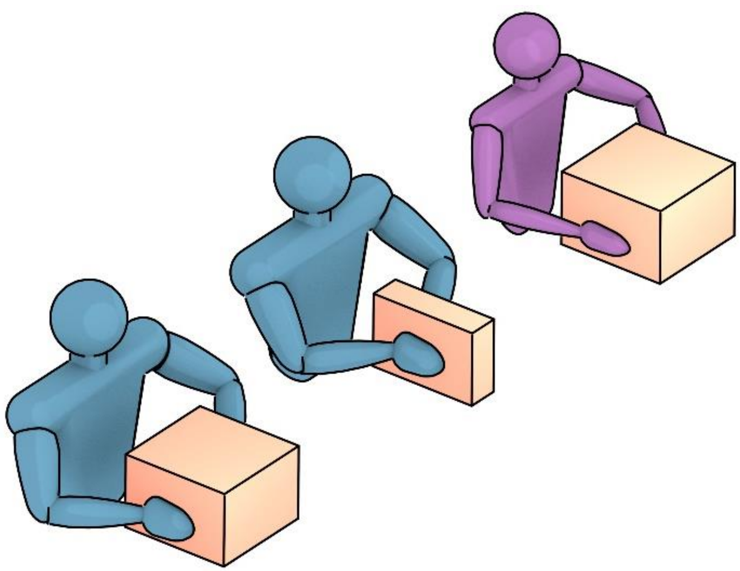
Research
Education
Technology transfer



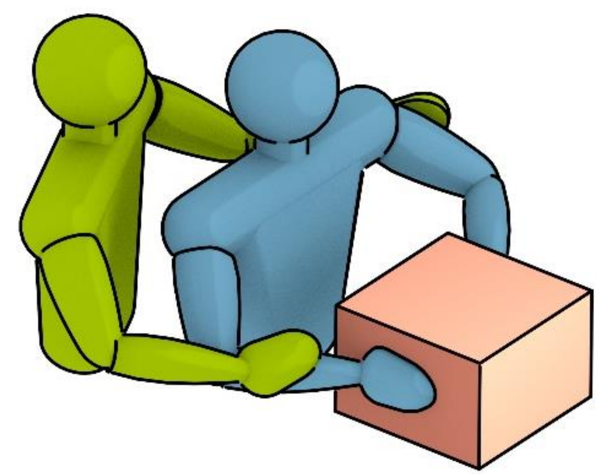
Learning from demonstration



Observational learning

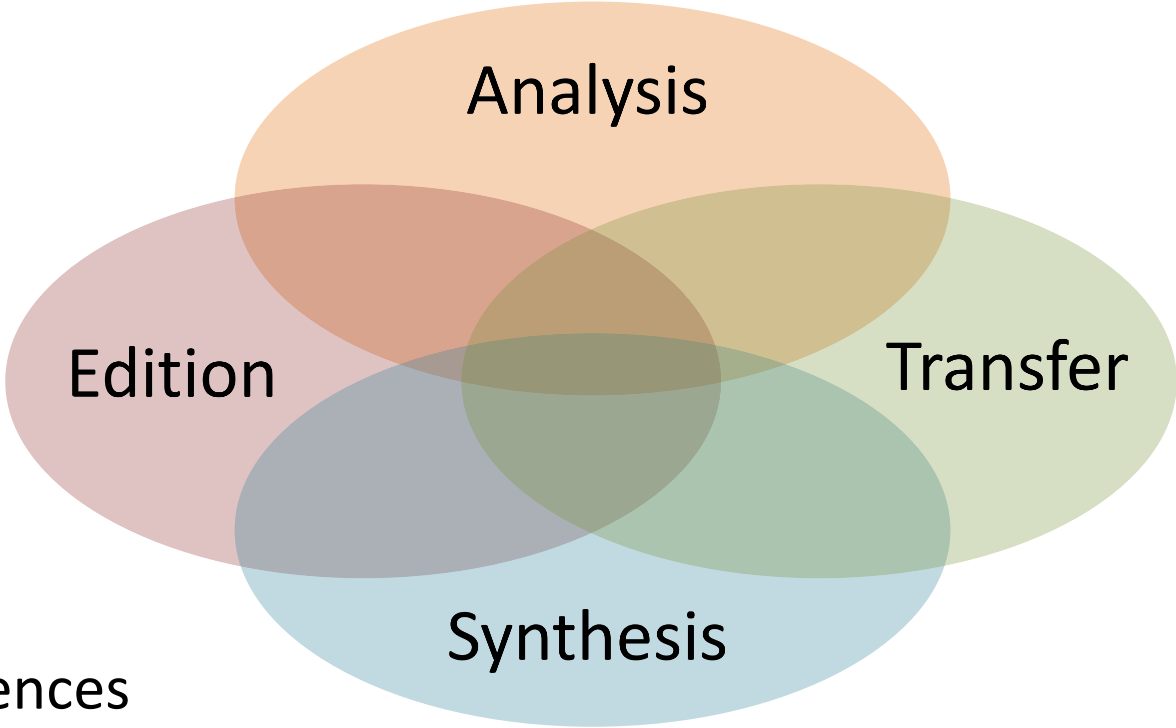


Correspondence problems



Kinesthetic teaching

Statistical, geometrical and dynamical representations of movements



- Robotics
- Human movement sciences
- Sensorimotor control
- Neurosciences
- Biomechanics
- Computer graphics
- Sport sciences
- …

Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**
  Product of Gaussians

- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

- **Differential geometry**
  Riemannian manifolds

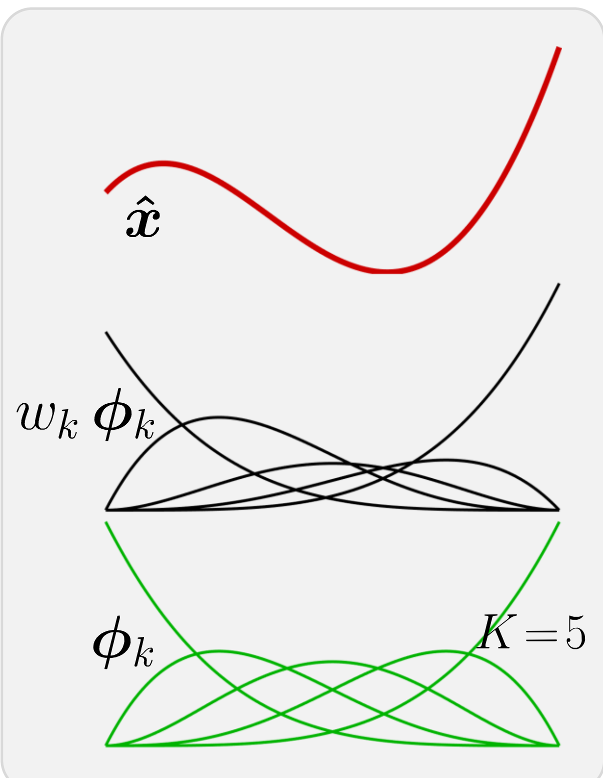
Superposition with basis functions

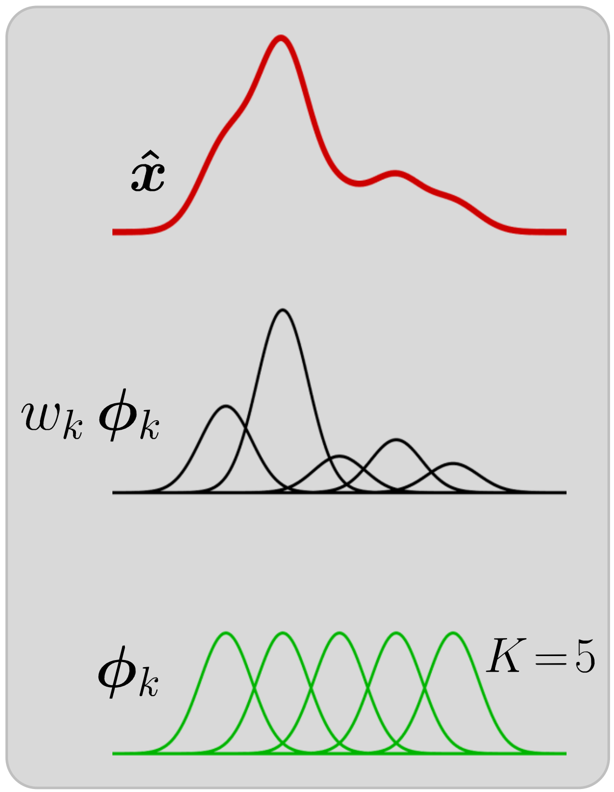
$$\hat{\boldsymbol{x}} = \sum_{k=1}^{K} w_k\, \boldsymbol{\phi}_k$$

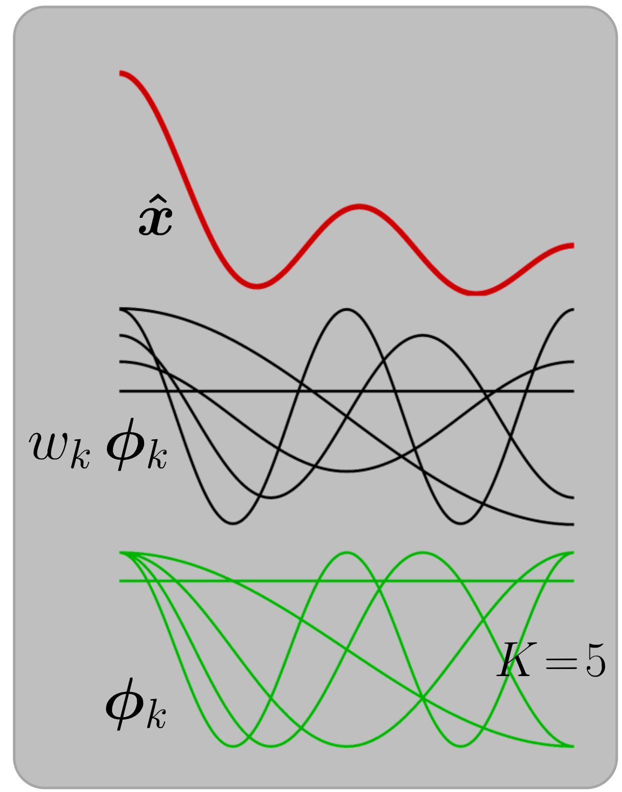
Bézier curves (Bernstein bases)

Locally weighted regression (radial bases)

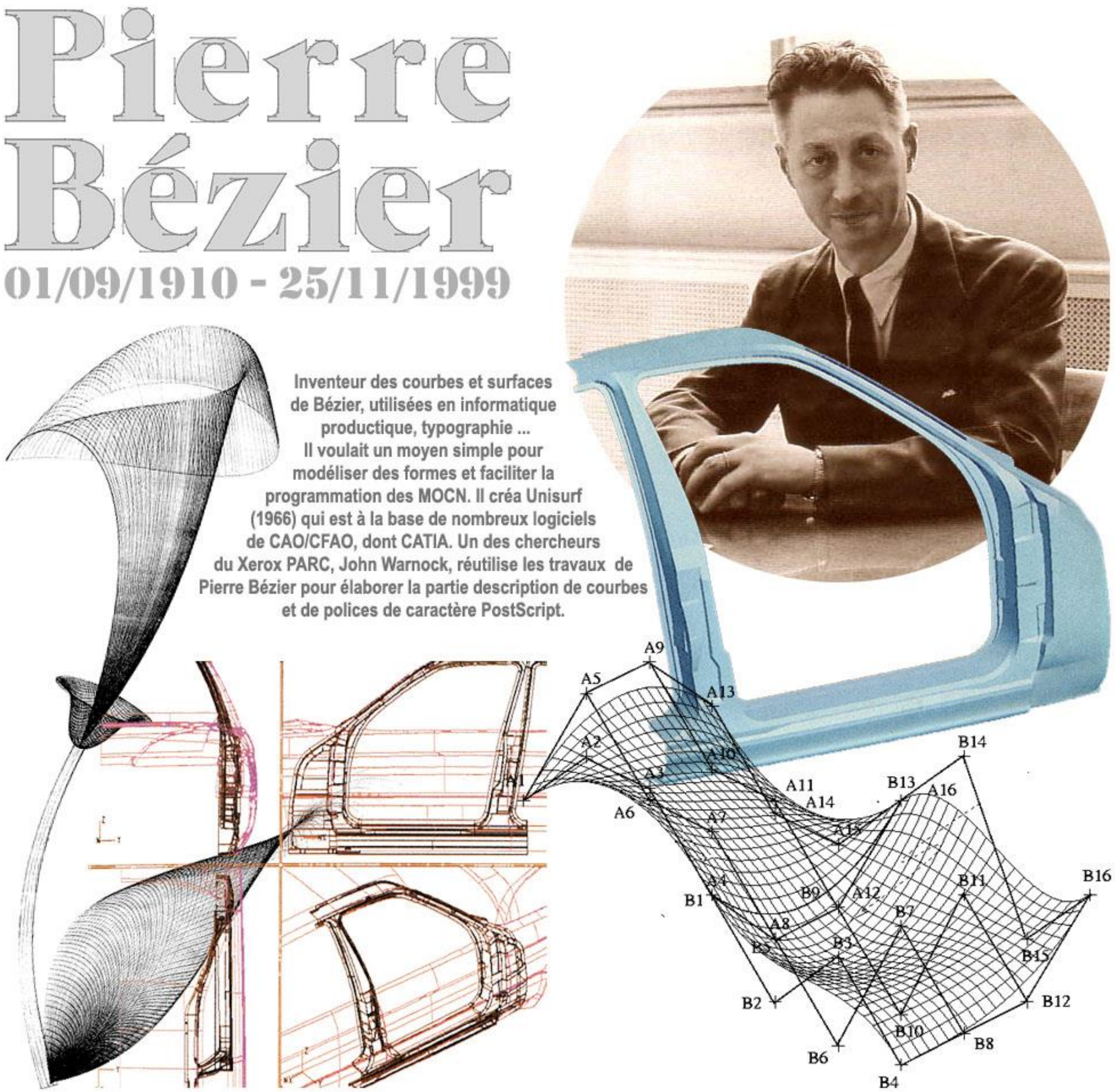
Fourier series (cosine bases)





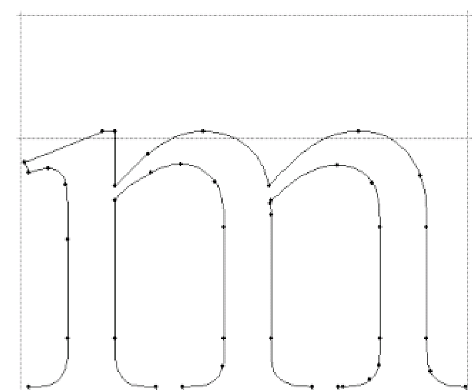


Bezier curves



Sergei Bernstein

Paul de Casteljau

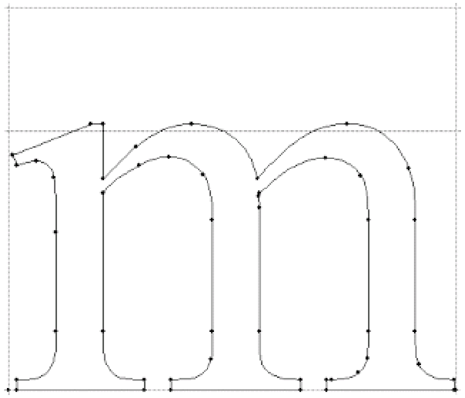


Bezier curves

- Mathematical basis for Bézier curves: **Bernstein polynomials**, known since 1912 but applied to graphics only 50 years later.
- **Pierre Bézier** patented and popularized, but did not invent the Bézier curves and Bézier surfaces that are now used in most computer-aided design and computer graphics systems.
- The study of these curves was first developed in 1959 by **Paul de Casteljau** at Citroën, where he was initially not permitted to publish his work. Pierre Bézier used them to design cars at Renault.
- Bézier curves are often used to define **3D paths** as well as 2D curves for **keyframe interpolation**.
- Bézier curves are now very frequently used to edit movements and animations.
- **TrueType fonts** use quadratic Bézier curves, PostScript and SVG use cubic Bézier curves.

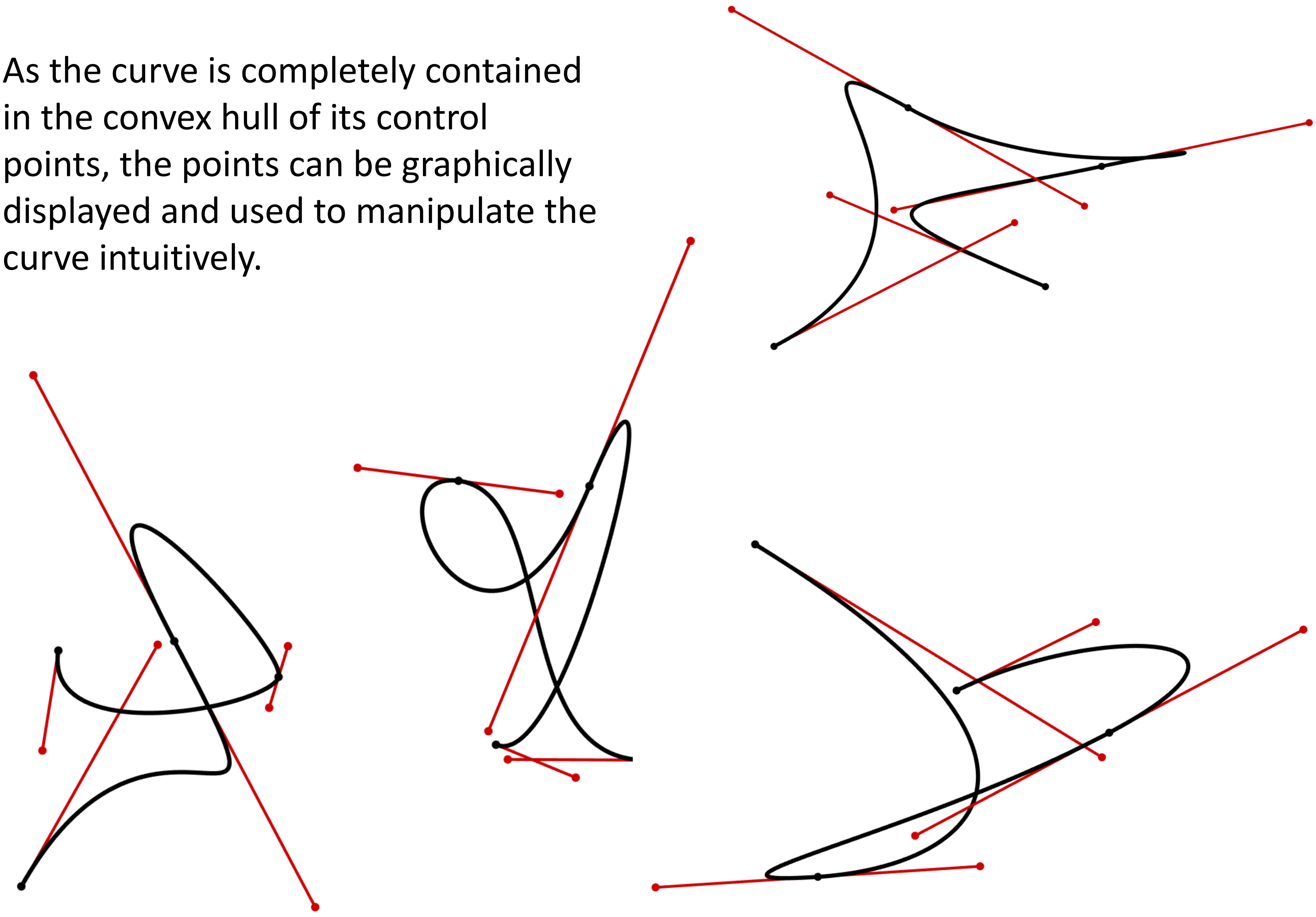
Sergei Bernstein

Paul de Casteljau



Bezier curves

As the curve is completely contained in the convex hull of its control points, the points can be graphically displayed and used to manipulate the curve intuitively.



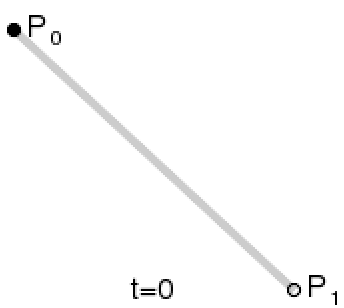
Quadratic Bezier curves

For $0 \leqslant t \leqslant 1$, a **linear Bézier curve** is the line traced by

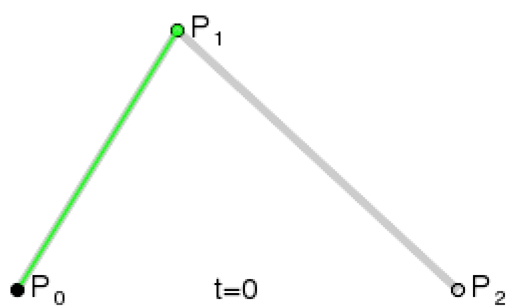
$$\boldsymbol{B}_{\boldsymbol{P}_0,\boldsymbol{P}_1}(t) = (1-t)\boldsymbol{P}_0 + t\,\boldsymbol{P}_1$$

For $0 \leqslant t \leqslant 1$, a **quadratic Bézier curve** is the path traced by

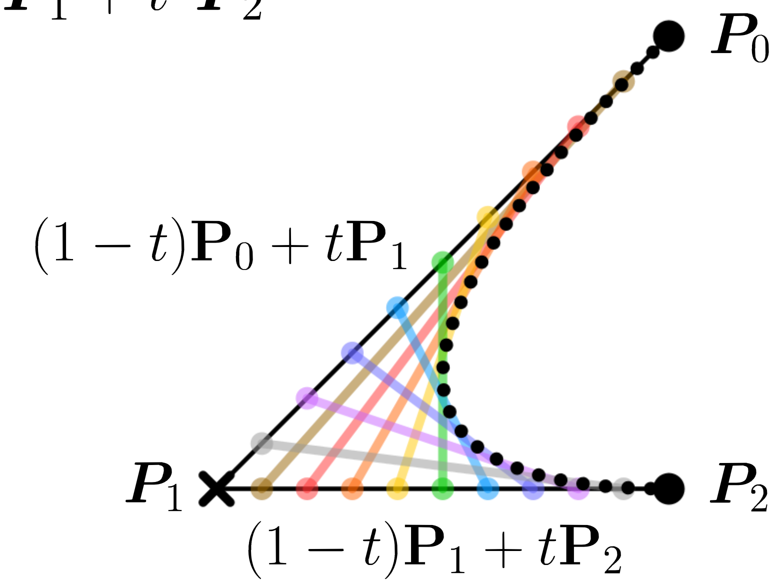
$$\begin{aligned}
\boldsymbol{B}_{\boldsymbol{P}_0,\boldsymbol{P}_1,\boldsymbol{P}_2}(t) &= (1-t)\ \boldsymbol{B}_{\boldsymbol{P}_0,\boldsymbol{P}_1}(t) + t\ \boldsymbol{B}_{\boldsymbol{P}_1,\boldsymbol{P}_2}(t) \\
&= (1-t)\Big((1-t)\boldsymbol{P}_0 + t\boldsymbol{P}_1\Big) + t\Big((1-t)\boldsymbol{P}_1 + t\boldsymbol{P}_2\Big) \\
&= (1-t)^2\boldsymbol{P}_0 + 2(1-t)t\,\boldsymbol{P}_1 + t^2\boldsymbol{P}_2
\end{aligned}$$



Linear Bézier curve



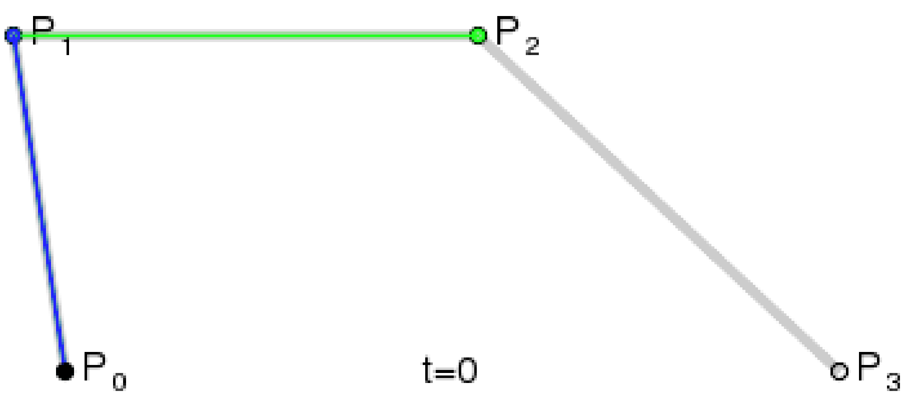
Quadratic Bézier curve



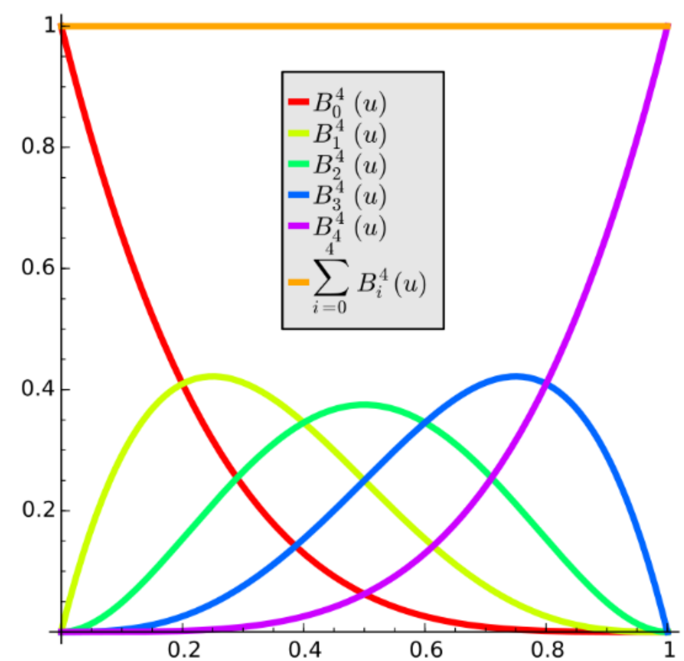
Cubic Bezier curves

For $0 \leqslant t \leqslant 1$, a **cubic Bézier curve** is the path traced by

$$\begin{aligned}\boldsymbol{B}_{\boldsymbol{P}_0,\boldsymbol{P}_1,\boldsymbol{P}_2,\boldsymbol{P}_3}(t) &= (1-t)\ \boldsymbol{B}_{\boldsymbol{P}_0,\boldsymbol{P}_1,\boldsymbol{P}_2}(t) + t\ \boldsymbol{B}_{\boldsymbol{P}_1,\boldsymbol{P}_2,\boldsymbol{P}_3}(t) \\ &= (1-t)^3\boldsymbol{P}_0 + 3(1-t)^2 t\boldsymbol{P}_1 + 3(1-t)t^2\boldsymbol{P}_2 + t^3\boldsymbol{P}_3\end{aligned}$$



Cubic Bézier curve



Basis functions (Bernstein polynomials) of cubic Bézier curves

Bezier curves of degree n

$$\hat{\boldsymbol{x}} = \sum_{k=1}^{K} w_k \, \boldsymbol{\phi}_k$$

For $0 \leqslant t \leqslant 1$, a recursive definition for the **Bézier curve of degree** n can be expressed as a linear interpolation of a pair of corresponding points in two Bézier curves of degree $n-1$, namely

$$\boldsymbol{B}(t) = \sum_{i=0}^{n} b_{i,n}(t) \boldsymbol{P}_i$$

with

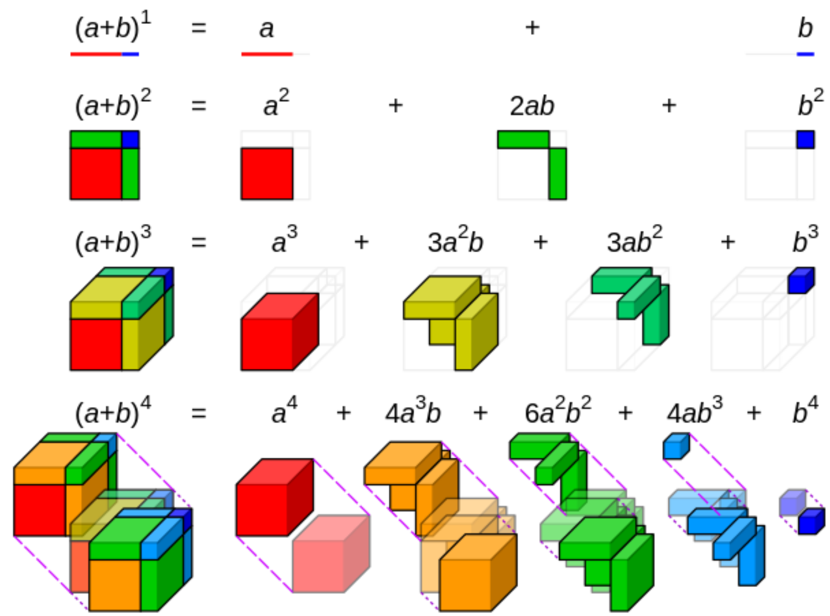
$$b_{i,n}(t) = \binom{n}{i} (1-t)^{n-i} \, t^i$$

the **Bernstein polynomials** of degree n,

where $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ are **binomial coefficients**.

Binomial coefficients

$$\boldsymbol{B}(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} (1-t)^{n-i}\, t^i \boldsymbol{P}_i$$



1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

$$\begin{aligned}
\mathbf{B}(t) &= \ (1-t)^4\ \mathbf{P}_0 \\
&+ 4t\ (1-t)^3\ \mathbf{P}_1 \\
&+ 6t^2\ (1-t)^2\ \mathbf{P}_2 \\
&+ 4t^3\ (1-t)\ \mathbf{P}_3 \\
&+ t^4\ \mathbf{P}_4
\end{aligned}$$

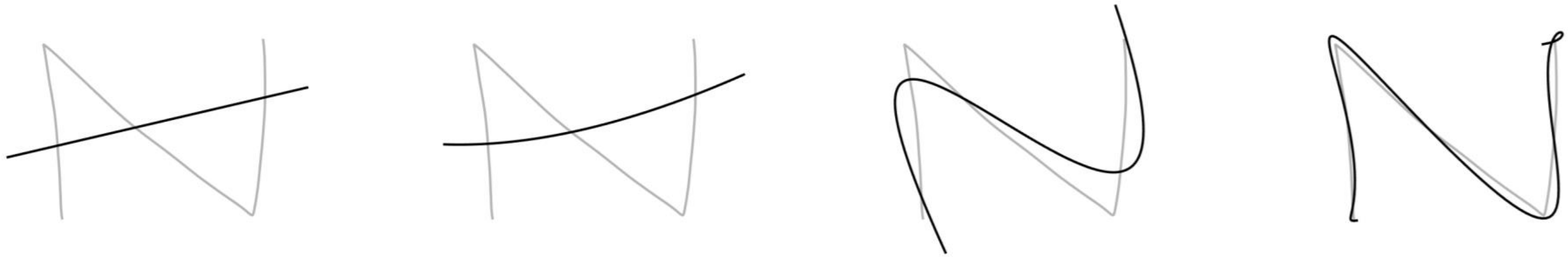
$$\begin{aligned}
\mathbf{B}(t) &= \ (1-t)^5\ \mathbf{P}_0 \\
&+ 5t\ (1-t)^4\ \mathbf{P}_1 \\
&+ 10t^2\ (1-t)^3\ \mathbf{P}_2 \\
&+ 10t^3\ (1-t)^2\ \mathbf{P}_3 \\
&+ 5t^4\ (1-t)\ \mathbf{P}_4 \\
&+ t^5\ \mathbf{P}_5
\end{aligned}$$

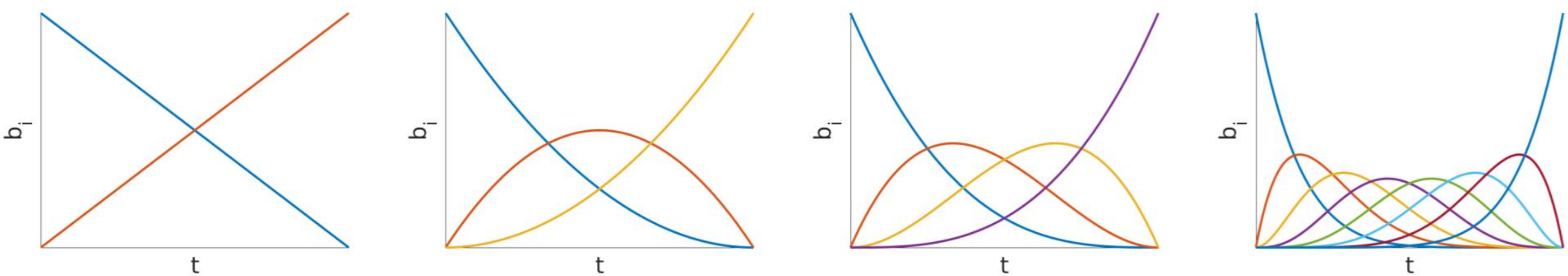
$t^0 = 1$

$(1-t)^0 = 1$

Bezier curves - Summary

$$\hat{\boldsymbol{x}} = \sum_{k=1}^{K} w_k \, \boldsymbol{\phi}_k$$





$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} \, (1-t)^{n-i} \, t^i \, \mathbf{P}_i$$

Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**
  Product of Gaussians

- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

- **Differential geometry**
  Riemannian manifolds

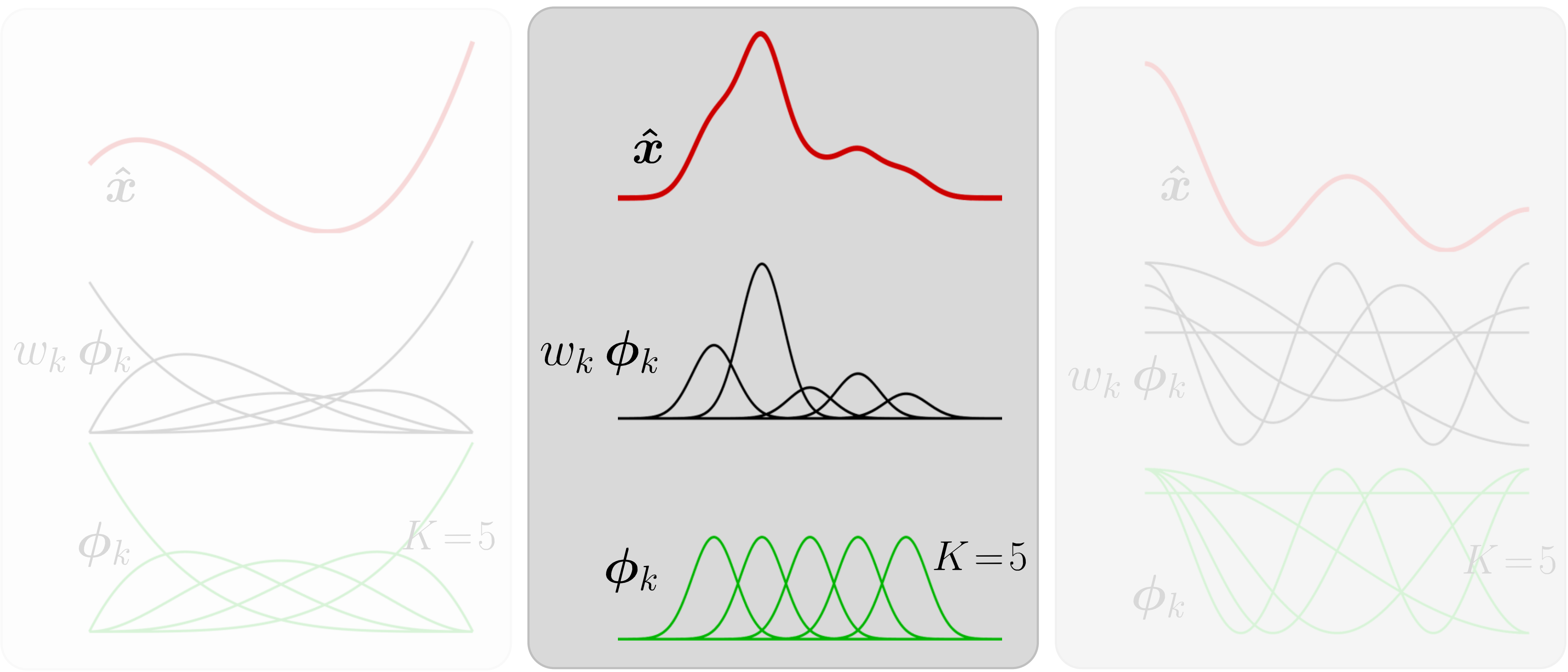
Superposition with basis functions

$$\hat{\boldsymbol{x}} = \sum_{k=1}^{K} w_k\, \boldsymbol{\phi}_k$$

Bézier curves (Bernstein bases)

Locally weighted regression (radial bases)

Fourier series (cosine bases)



Multivariate ordinary least squares

By describing the input data as $\boldsymbol{X} \in \mathbb{R}^{N \times D^{\mathcal{I}}}$ and the output data as $\boldsymbol{Y} \in \mathbb{R}^{N \times D^{\mathcal{O}}}$, we want to find $\boldsymbol{A} \in \mathbb{R}^{D^{\mathcal{I}} \times D^{\mathcal{O}}}$ such that $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{A}$.

A solution can be found by minimizing the Frobenius norm

$$\begin{aligned}\hat{\boldsymbol{A}} &= \arg\min_{\boldsymbol{A}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A}\|_F^2 \\ &= \arg\min_{\boldsymbol{A}} \operatorname{tr}\Big((\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A})^\top(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A})\Big) \\ &= \arg\min_{\boldsymbol{A}} \operatorname{tr}(\boldsymbol{Y}^\top\boldsymbol{Y} - 2\boldsymbol{A}^\top\boldsymbol{X}^\top\boldsymbol{Y} + \boldsymbol{A}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{A})\end{aligned}$$

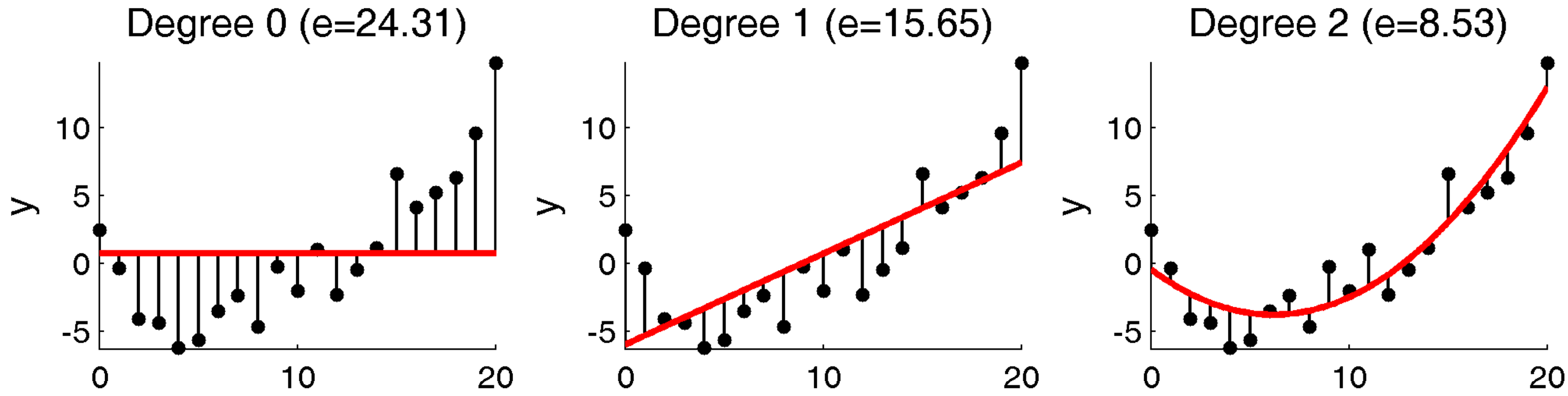
By differentiating with respect to $\boldsymbol{A}$ and equating to zero

$$-2\boldsymbol{X}^\top\boldsymbol{Y} + 2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{A} = \boldsymbol{0} \quad \Longleftrightarrow \quad \hat{\boldsymbol{A}} = \boxed{(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top}\boldsymbol{Y}$$

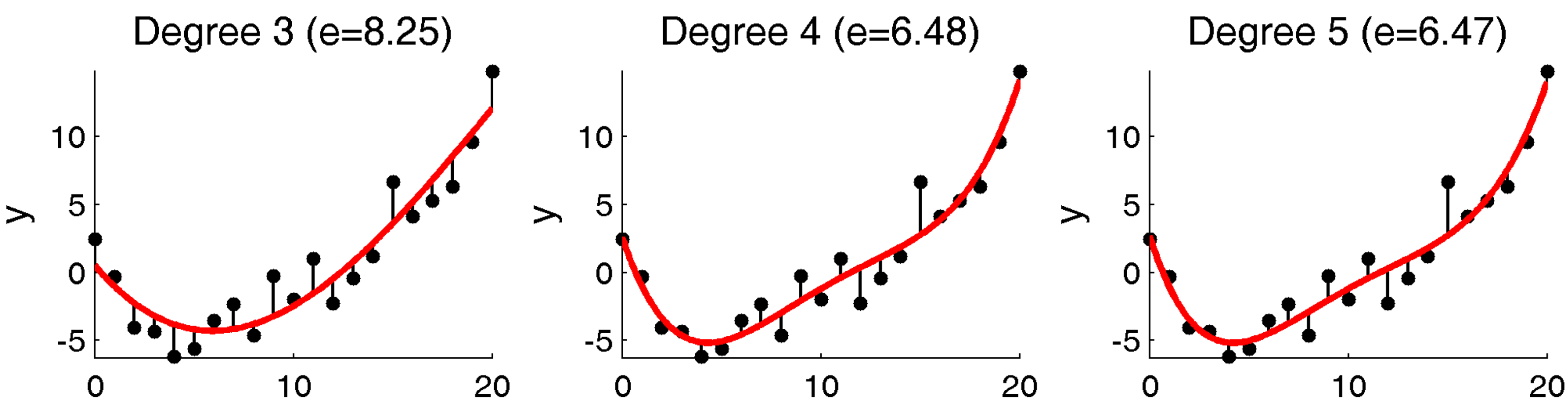
Moore-Penrose pseudoinverse $\boldsymbol{X}^\dagger$

Polynomial fitting with least squares

$\hat{A} = X^{\dagger}\, Y$



$$X = 1 \qquad X = [x, 1] \qquad X = [x^2, x, 1]$$



$$X = [x^3, x^2, x, 1] \qquad X = [x^4, x^3, \ldots, 1] \qquad X = [x^5, x^4, \ldots, 1]$$

Weighted least squares

By describing the input data as $\boldsymbol{X} \in \mathbb{R}^{N \times D^{\mathcal{I}}}$ and the output data as $\boldsymbol{Y} \in \mathbb{R}^{N \times D^{\mathcal{O}}}$, we want to minimize

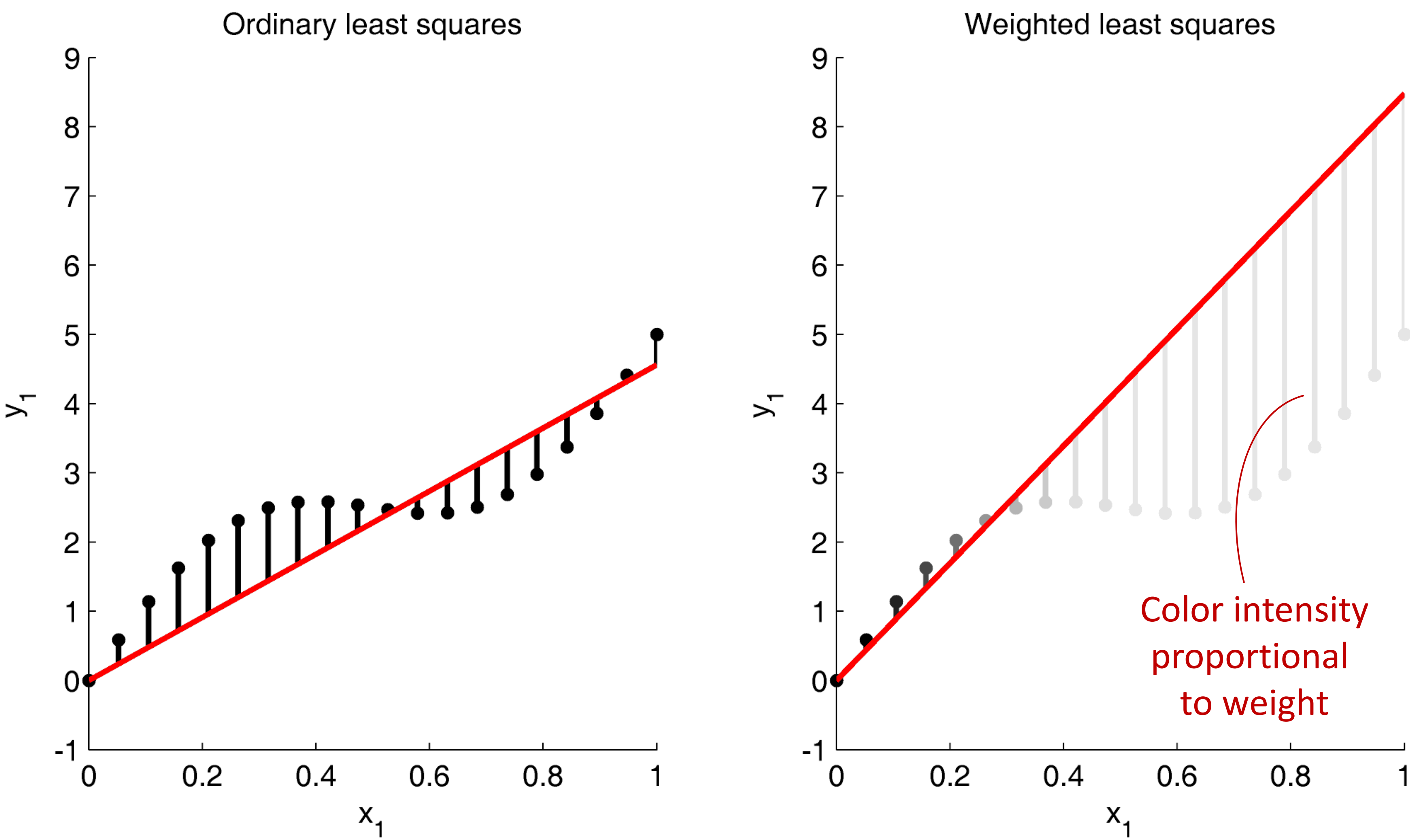
$$\begin{aligned}\hat{\boldsymbol{A}} &= \arg\min_{\boldsymbol{A}} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A}\|^2_{F,\boldsymbol{W}} \\ &= \arg\min_{\boldsymbol{A}} \operatorname{tr}\Big((\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A})^{\top}\boldsymbol{W}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A})\Big) \\ &= \arg\min_{\boldsymbol{A}} \operatorname{tr}(\boldsymbol{Y}^{\top}\boldsymbol{W}\boldsymbol{Y} - 2\boldsymbol{A}^{\top}\boldsymbol{X}^{\top}\boldsymbol{W}\boldsymbol{Y} + \boldsymbol{A}^{\top}\boldsymbol{X}^{\top}\boldsymbol{W}\boldsymbol{X}\boldsymbol{A})\end{aligned}$$

By differentiating with respect to $\boldsymbol{A}$ and equating to zero

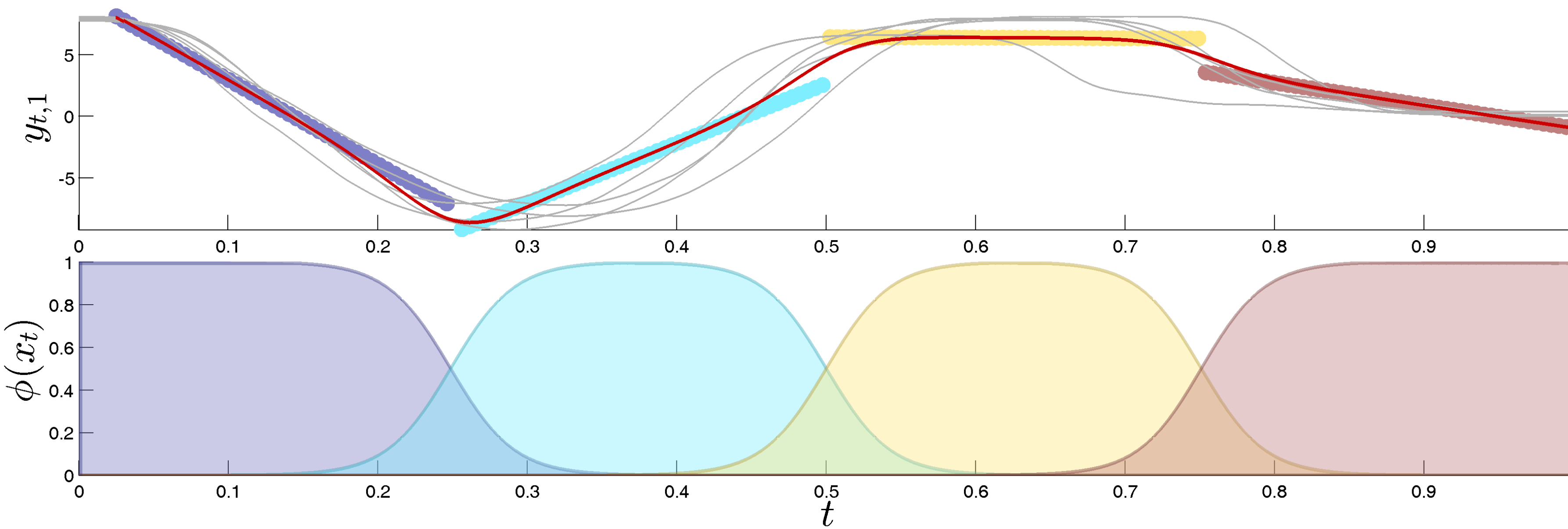
$$-2\boldsymbol{X}^{\top}\boldsymbol{W}\boldsymbol{Y} + 2\boldsymbol{X}^{\top}\boldsymbol{W}\boldsymbol{X}\boldsymbol{A} = \boldsymbol{0} \iff \hat{\boldsymbol{A}} = \underbrace{\boxed{(\boldsymbol{X}^{\top}\boldsymbol{W}\boldsymbol{X})^{-1}\boldsymbol{X}^{\top}\boldsymbol{W}}}_{\boldsymbol{X}^{\dagger}_{\boldsymbol{W}}}\boldsymbol{Y}$$

Weighted least squares

$$\hat{A} = (X^\top W X)^{-1} X^\top W Y$$



Locally weighted regression (LWR)

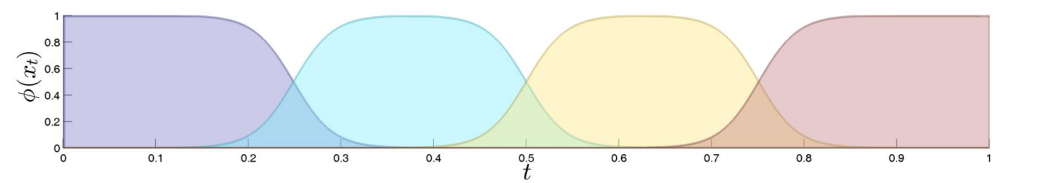


Locally weighted regression (LWR)

LWR computes K estimates $\hat{\boldsymbol{A}}_k$, each with a different weighting function $\phi_k(\boldsymbol{x}_n^{\mathcal{I}})$, often defined as the **radial basis functions** (RBF)

$$\tilde{\phi}_k(\boldsymbol{x}_n^{\mathcal{I}}) = \exp\Big(-\frac{1}{2}(\boldsymbol{x}_n^{\mathcal{I}} - \boldsymbol{\mu}_k^{\mathcal{I}})^\top \boldsymbol{\Sigma}_k^{\mathcal{I}\ -1}(\boldsymbol{x}_n^{\mathcal{I}} - \boldsymbol{\mu}_k^{\mathcal{I}})\Big),$$

or in its rescaled form as



$$\phi_k(\boldsymbol{x}_n^{\mathcal{I}}) = \frac{\tilde{\phi}_k(\boldsymbol{x}_n^{\mathcal{I}})}{\sum_{i=1}^K \tilde{\phi}_i(\boldsymbol{x}_n^{\mathcal{I}})},$$

where $\boldsymbol{\mu}_k^{\mathcal{I}}$ and $\boldsymbol{\Sigma}_k^{\mathcal{I}}$ are the parameters of the k-th RBF.

$\rightarrow$ K **weighted regressions on the same dataset** $\{\boldsymbol{X}^{\mathcal{I}}, \boldsymbol{X}^{\mathcal{O}}\}$

$\rightarrow$ **Nonlinear problem solved locally by linear regression**

Locally weighted regression (LWR)

Often, the centroids $\boldsymbol{\mu}_k^{\mathcal{I}}$ are set to uniformly cover the input space, and $\boldsymbol{\Sigma}_k^{\mathcal{I}} = \boldsymbol{I}\sigma^2$ is used as a common bandwidth shared by all basis functions.

$$\boldsymbol{X}^{\mathcal{I}} = [t_1, t_2, \ldots, t_N]^\top$$

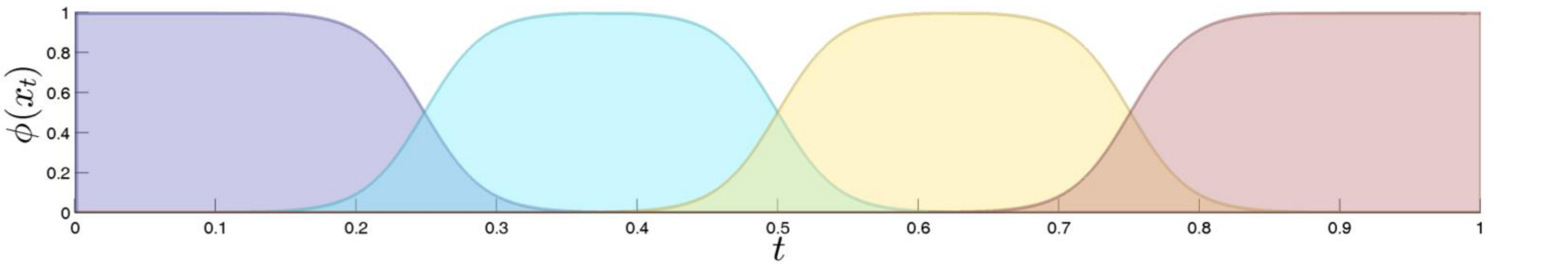
$$\hat{\boldsymbol{A}}_k = (\boldsymbol{X}^{\mathcal{I}\top} \boldsymbol{W}_k \boldsymbol{X}^{\mathcal{I}})^{-1} \boldsymbol{X}^{\mathcal{I}\top} \boldsymbol{W}_k \, \boldsymbol{X}^{\mathcal{O}}$$

An associated diagonal matrix

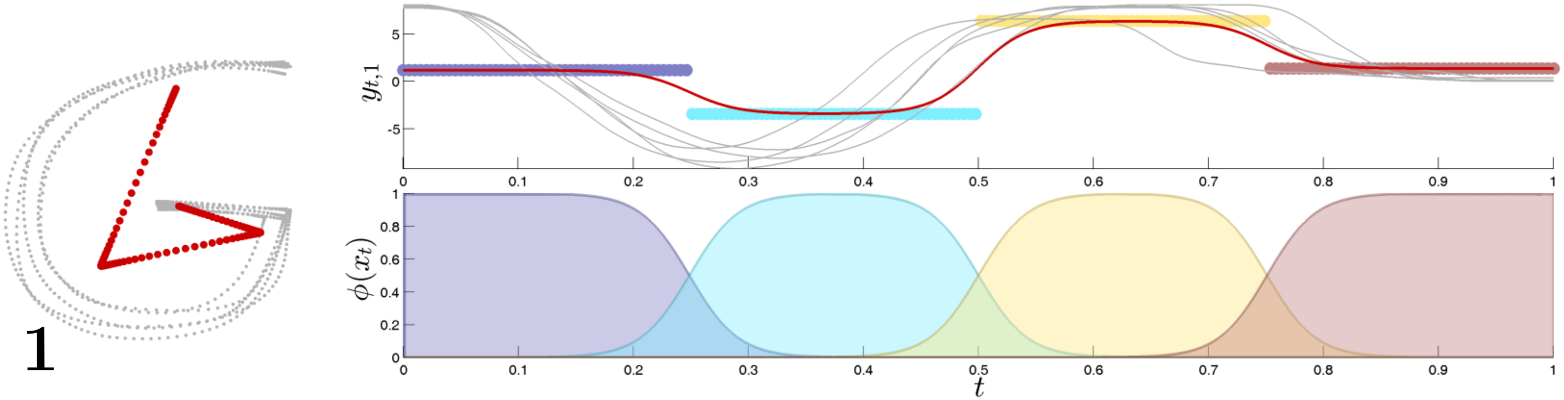
$$\boldsymbol{W}_k = \mathrm{diag}\Big(\phi_k(\boldsymbol{x}_1^{\mathcal{I}}), \phi_k(\boldsymbol{x}_2^{\mathcal{I}}), \ldots, \phi_k(\boldsymbol{x}_N^{\mathcal{I}})\Big)$$

can be used to evaluate $\hat{\boldsymbol{A}}_k$. The result can then be used to compute

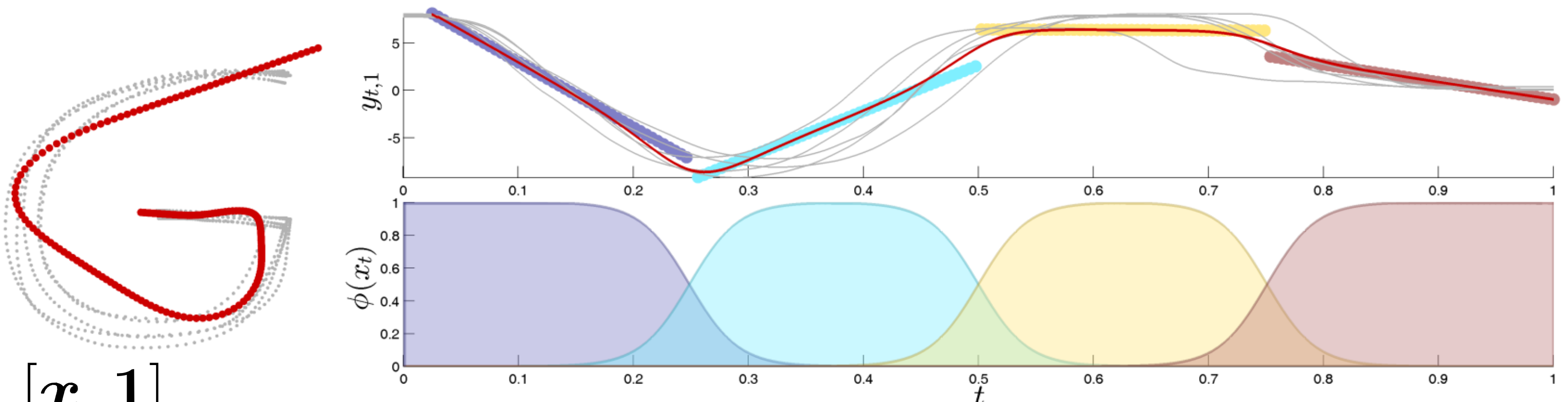
$$\boldsymbol{X}^{\mathcal{O}} = \sum_{k=1}^{K} \boldsymbol{W}_k \boldsymbol{X}^{\mathcal{I}} \hat{\boldsymbol{A}}_k$$



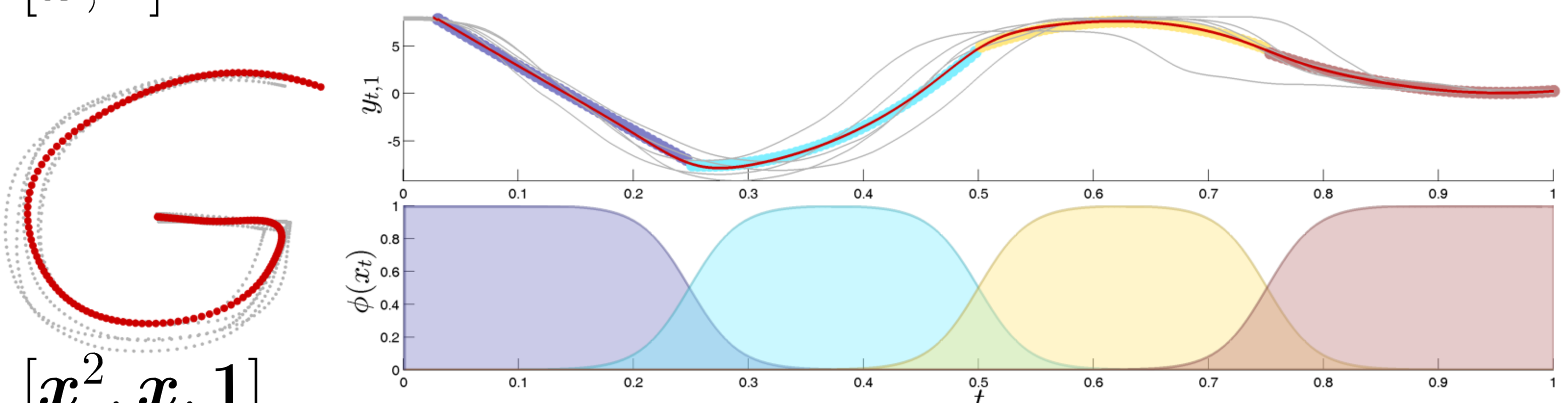
Locally weighted regression (LWR)



$\boldsymbol{X} = \mathbf{1}$

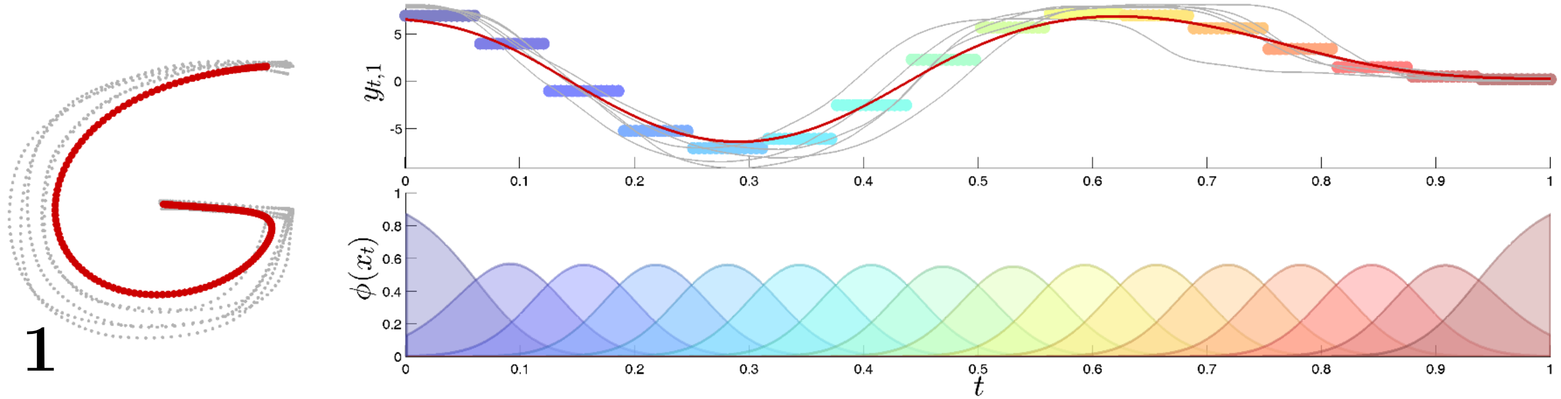


$\boldsymbol{X} = [\boldsymbol{x}, \mathbf{1}]$

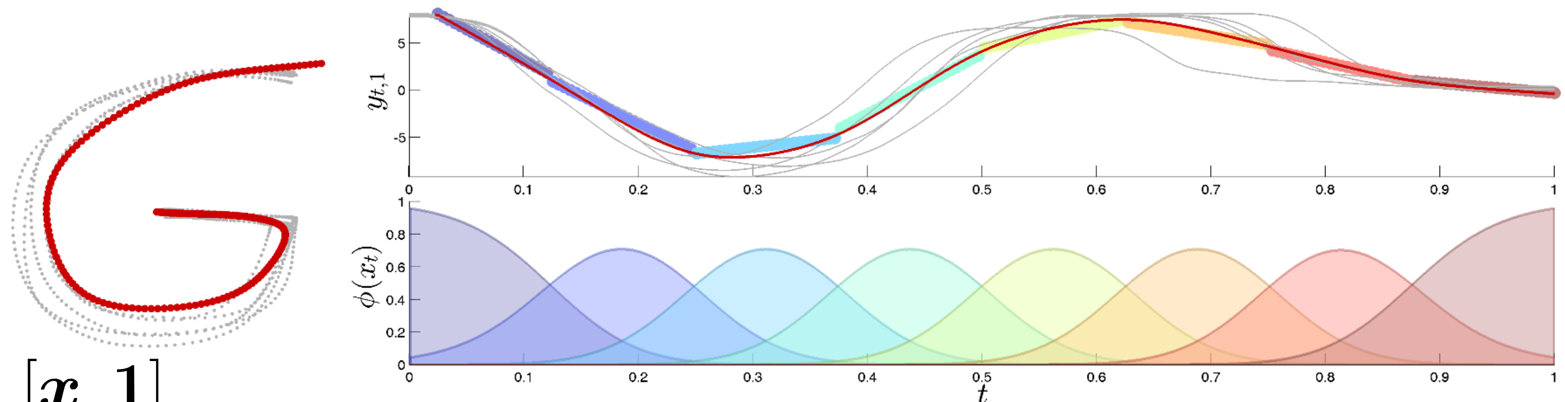


$\boldsymbol{X} = [\boldsymbol{x}^2, \boldsymbol{x}, \mathbf{1}]$

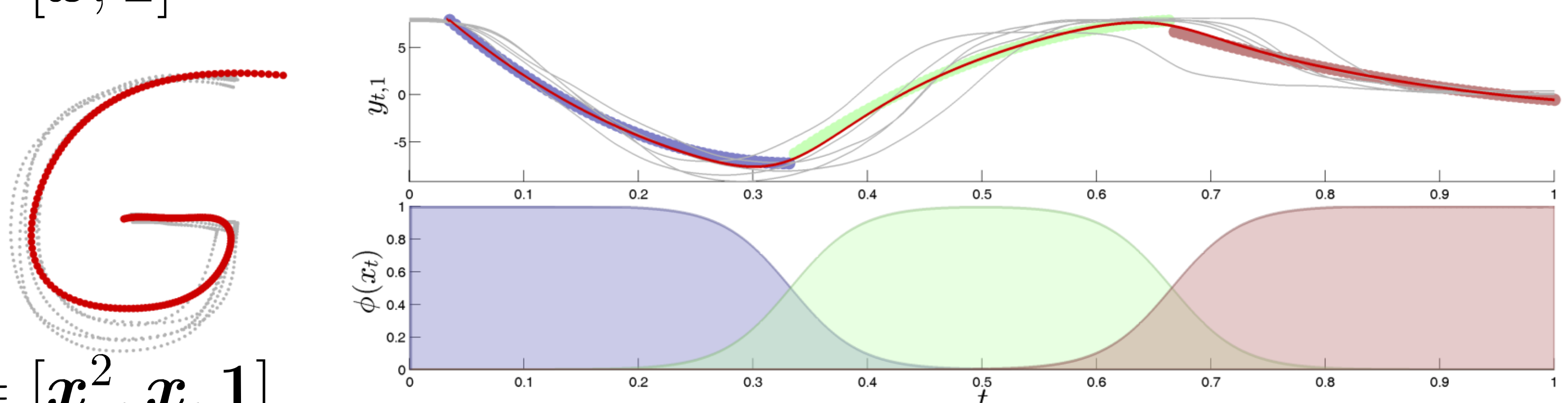
Locally weighted regression (LWR)



$$\boldsymbol{X} = \mathbf{1}$$



$$\boldsymbol{X} = [\boldsymbol{x}, \mathbf{1}]$$



$$\boldsymbol{X} = [\boldsymbol{x}^2, \boldsymbol{x}, \mathbf{1}]$$

Locally weighted regression (LWR) - Resources

Softwares

http://www.idiap.ch/software/pbdlib/

Matlab codes: demo_LWR01.m

C++ codes: demo_LWR_batch01.cpp

References

[Atkeson, Moore and Schaal, *"Locally weighted learning for control"*, Artificial Intelligence Review, 11(1-5), 1997]

[Cleveland, *"Robust locally weighted regression and smoothing scatterplots"*, American Statistical Association 74(368), 1979]

[Calinon and Lee, *"Learning Control"*, Humanoid Robotics: a Reference (Springer), 2019]

Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**
  Product of Gaussians

- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

- **Differential geometry**
  Riemannian manifolds

Gaussian Mixture Model (GMM)

K Gaussians
N datapoints of dimension D

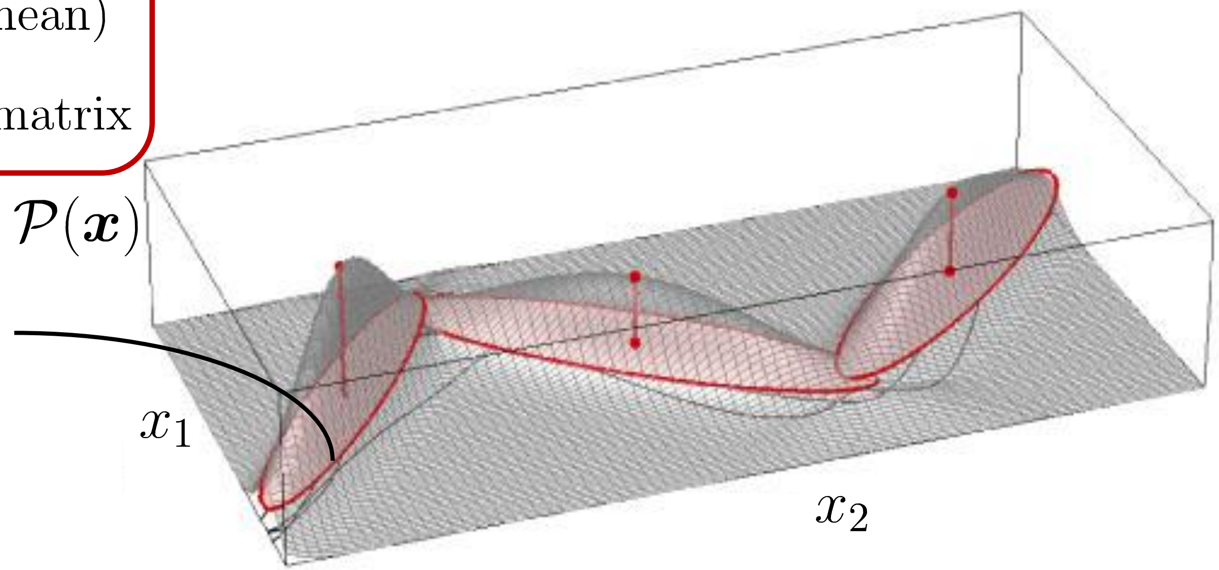
$$\mathcal{P}(\boldsymbol{x}_t) = \sum_{i=1}^{K} \pi_i \, \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

$$\mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (\boldsymbol{x}_t - \boldsymbol{\mu}_i)^{\top} \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x}_t - \boldsymbol{\mu}_i) \right)$$

$\boldsymbol{x} \in \mathbb{R}^{D \times N}$ Observations ($N = \sum_{m=1}^{M} T_m$, the m-th trajectory has T_m datapoints)

$\pi_i \in \mathbb{R}$	Mixing coefficient
$\boldsymbol{\mu}_i \in \mathbb{R}^D$	Center (or mean)
$\boldsymbol{\Sigma}_i \in \mathbb{R}^{D \times D}$	Covariance matrix

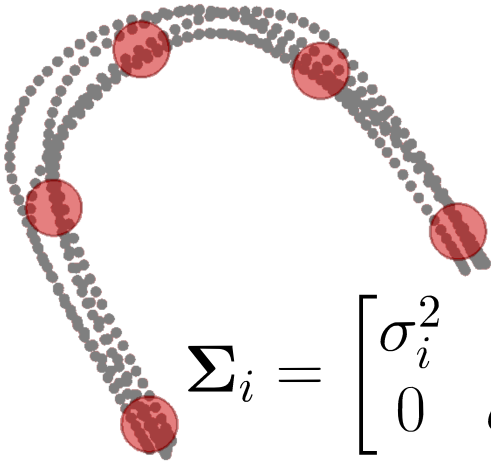
Parameters $\boldsymbol{\Theta}^{\text{GMM}} = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$



Equidensity contour of one standard deviation

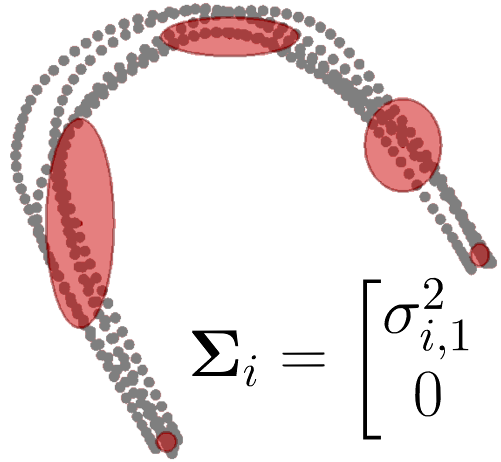
Covariance structures in GMM

Isotropic



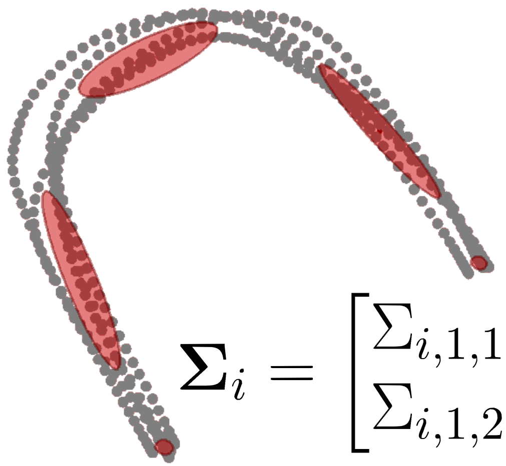
$$\boldsymbol{\Sigma}_i = \begin{bmatrix} \sigma_i^2 & 0 \\ 0 & \sigma_i^2 \end{bmatrix}$$

Diagonal



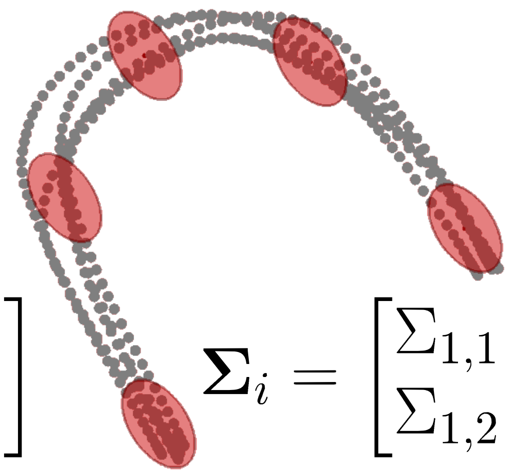
$$\boldsymbol{\Sigma}_i = \begin{bmatrix} \sigma_{i,1}^2 & 0 \\ 0 & \sigma_{i,2}^2 \end{bmatrix}$$

Full



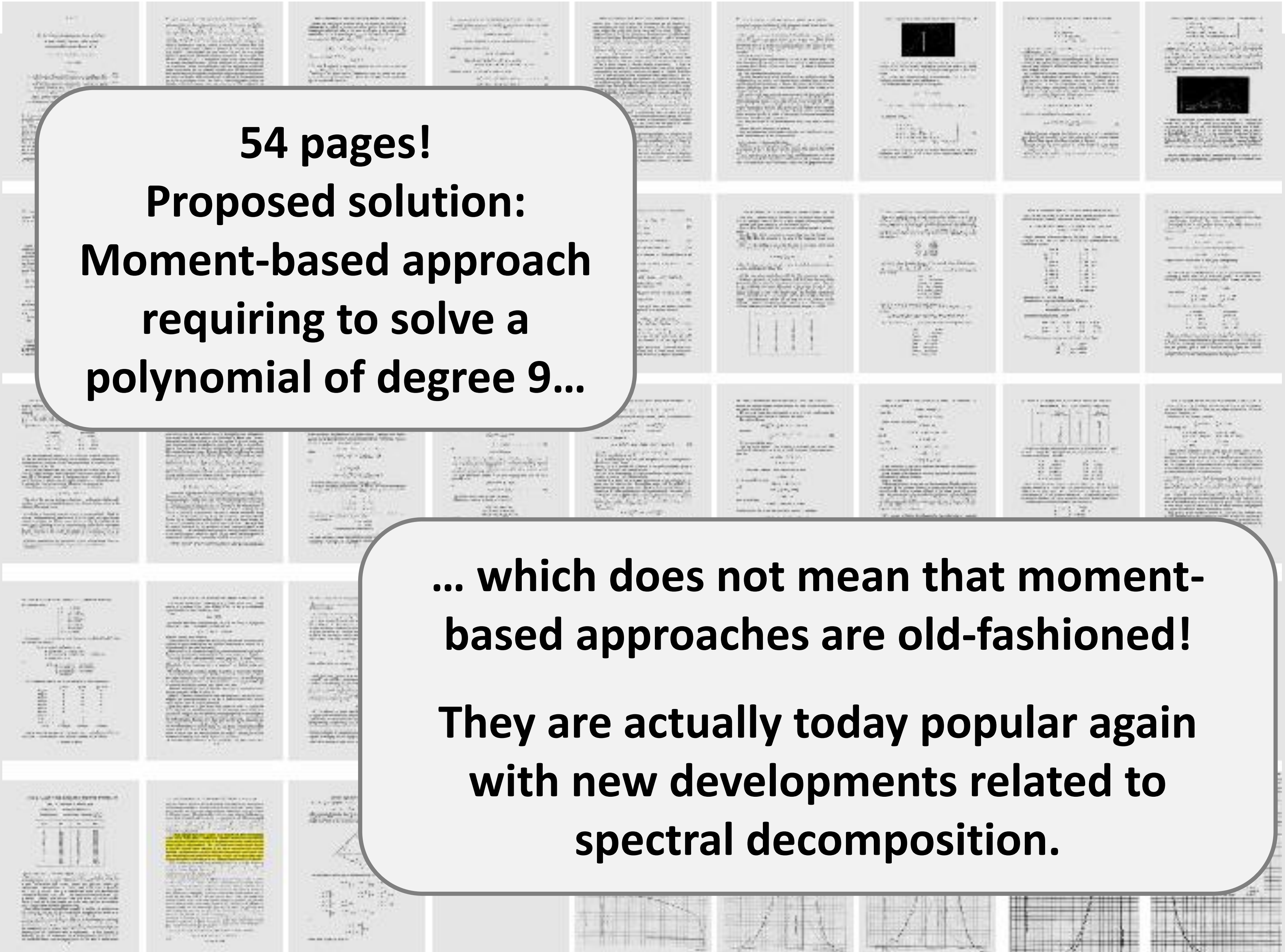
$$\boldsymbol{\Sigma}_i = \begin{bmatrix} \Sigma_{i,1,1} & \Sigma_{i,1,2} \\ \Sigma_{i,1,2} & \Sigma_{i,2,2} \end{bmatrix}$$

Tied



$$\boldsymbol{\Sigma}_i = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{1,2} & \Sigma_{2,2} \end{bmatrix}$$

Parameters estimation in GMM… in 1893

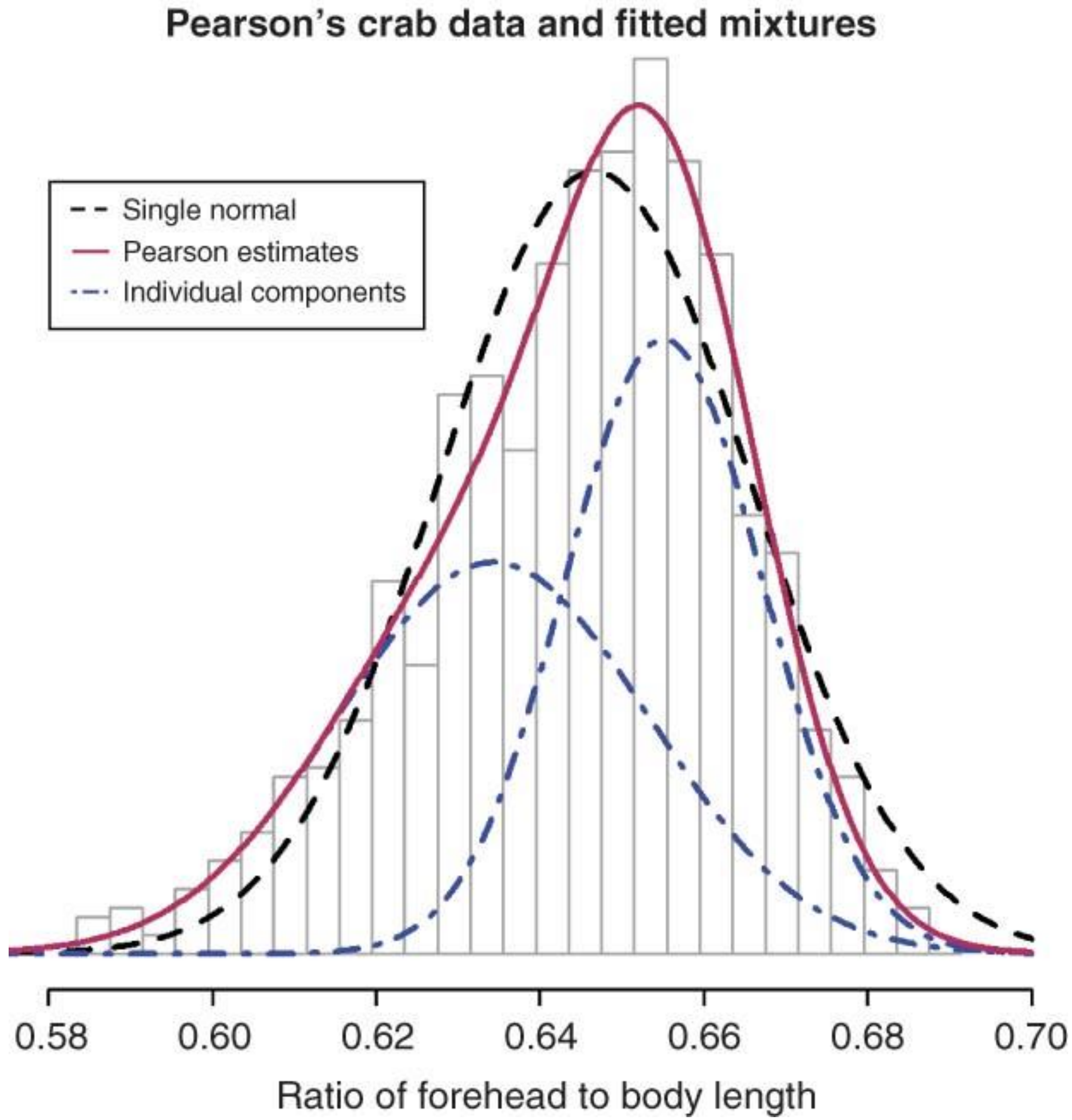


54 pages!
Proposed solution:
Moment-based approach requiring to solve a polynomial of degree 9…

… which does not mean that moment-based approaches are old-fashioned!

They are actually today popular again with new developments related to spectral decomposition.

Parameters estimation in GMM... in 1893



EM for GMM: Resulting procedure

K *Gaussians*
N *datapoints*

E-step:

$$h_{t,i} = \frac{\pi_i \, \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{k=1}^{K} \pi_k \, \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

M-step:

$$\pi_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i}}{N},$$

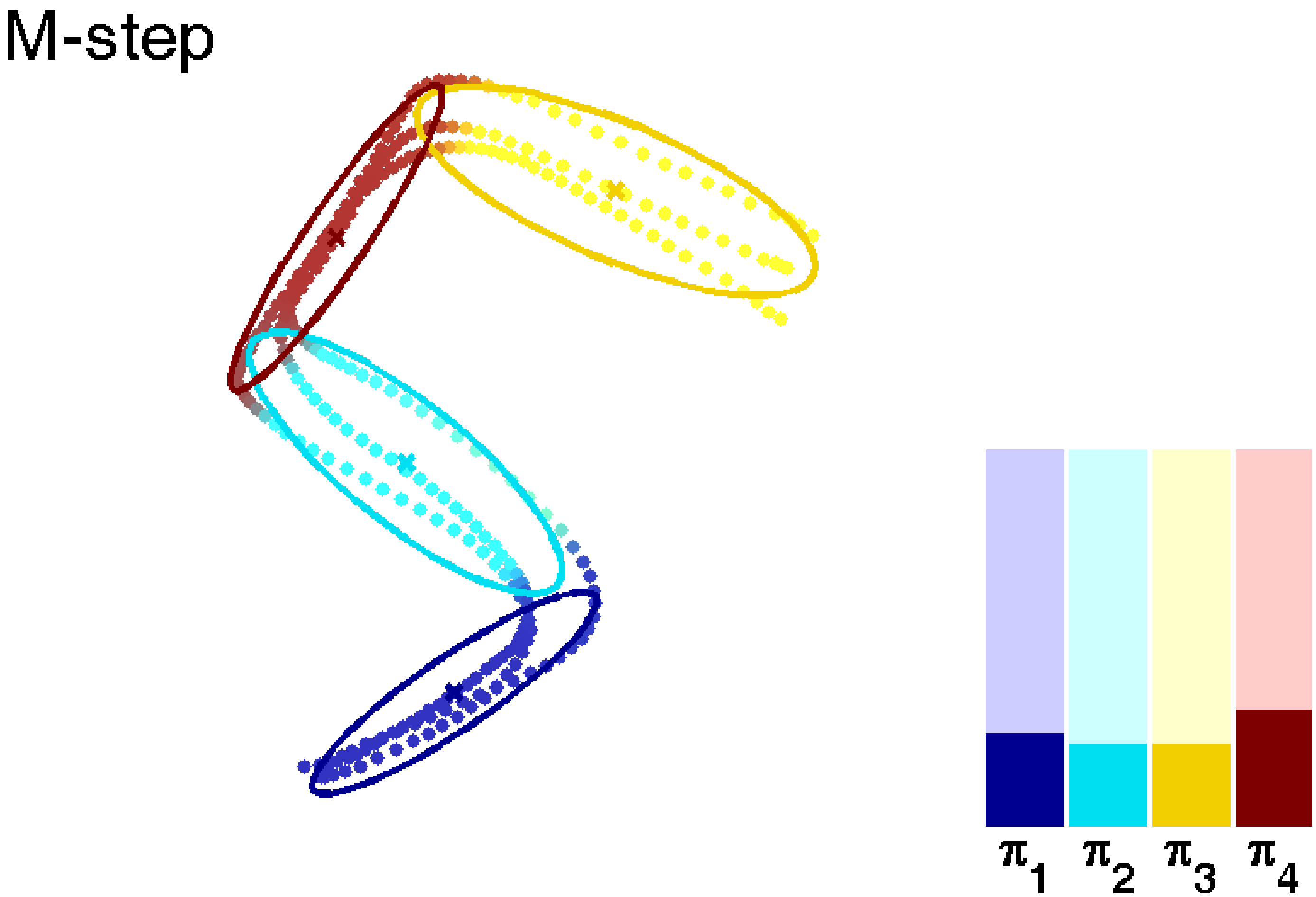
$$\boldsymbol{\mu}_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i} \, \boldsymbol{x}_t}{\sum_{t=1}^{N} h_{t,i}},$$

$$\boldsymbol{\Sigma}_i \leftarrow \frac{\sum_{t=1}^{N} h_{t,i} \, (\boldsymbol{x}_t - \boldsymbol{\mu}_i)(\boldsymbol{x}_t - \boldsymbol{\mu}_i)^{\top}}{\sum_{t=1}^{N} h_{t,i}}$$

These results can be intuitively interpreted in terms of normalized counts.
EM provides a systematic approach to derive such procedure.

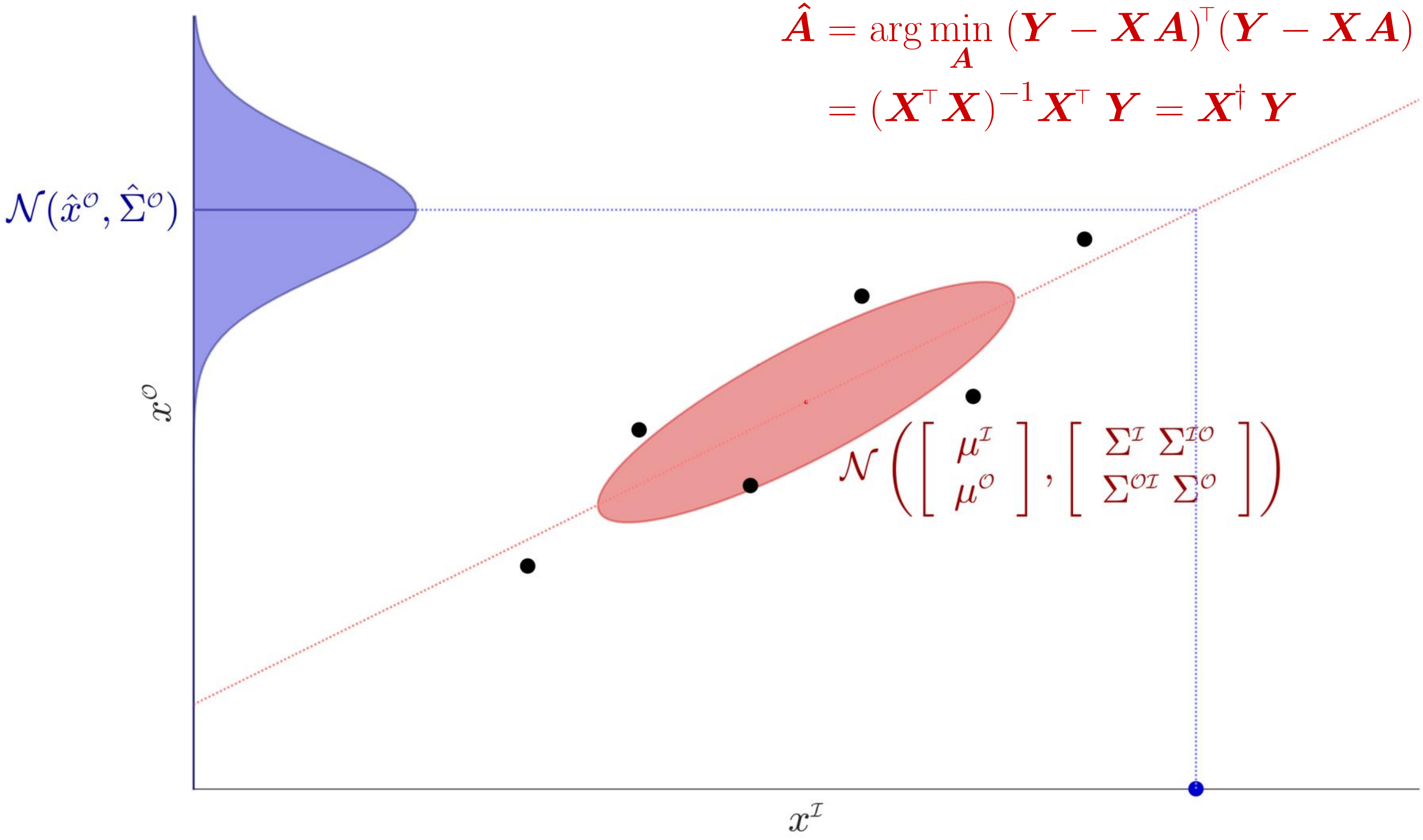
→ Weighted averages taking into account the responsibility of each datapoint in each cluster.

EM for GMM



Gaussian conditioning
(regression from joint distribution)

$$\hat{\boldsymbol{A}} = \arg\min_{\boldsymbol{A}} (\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A})^{\top}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{A})$$
$$= (\boldsymbol{X}^{\top}\boldsymbol{X})^{-1}\boldsymbol{X}^{\top}\,\boldsymbol{Y} = \boldsymbol{X}^{\dagger}\,\boldsymbol{Y}$$



→ Linear regression from joint distribution

Gaussian conditioning

We consider multivariate datapoints $\boldsymbol{x}$ and multivariate Gaussian distributions characterized by centers $\boldsymbol{\mu}$ and covariances $\boldsymbol{\Sigma}$, that can be partitioned as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{\mathcal{I}} \\ \boldsymbol{x}^{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}^{\mathcal{I}} \\ \boldsymbol{\mu}^{\mathcal{O}} \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}^{\mathcal{I}} & \boldsymbol{\Sigma}^{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Sigma}^{\mathcal{O}\mathcal{I}} & \boldsymbol{\Sigma}^{\mathcal{O}} \end{bmatrix}.$$

If $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we have that $\boldsymbol{x}^{\mathcal{O}}|\boldsymbol{x}^{\mathcal{I}} \sim \mathcal{N}\big(\hat{\boldsymbol{x}}^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}^{\mathcal{O}}\big)$, with parameters

$$\begin{aligned} \hat{\boldsymbol{x}}^{\mathcal{O}} &= \boldsymbol{\mu}^{\mathcal{O}} + \boldsymbol{\Sigma}^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}^{\mathcal{I}-1} (\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}^{\mathcal{I}}), \\ \hat{\boldsymbol{\Sigma}}^{\mathcal{O}} &= \boldsymbol{\Sigma}^{\mathcal{O}} - \boldsymbol{\Sigma}^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}^{\mathcal{I}-1} \boldsymbol{\Sigma}^{\mathcal{I}\mathcal{O}}. \end{aligned}$$

We can see that $\hat{\boldsymbol{x}}^{\mathcal{O}}$ is linearly dependent on $\boldsymbol{x}^{\mathcal{I}}$, and that $\hat{\boldsymbol{\Sigma}}^{\mathcal{O}}$ is independent of $\boldsymbol{x}^{\mathcal{I}}$.

We can also notice that for full joint covariance, the conditional covariance $\hat{\boldsymbol{\Sigma}}^{\mathcal{O}}$ will typically be smaller than the marginal $\boldsymbol{\Sigma}^{\mathcal{O}}$.

Gaussian mixture regression (GMR)

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{\mathcal{I}} \\ \boldsymbol{x}^{\mathcal{O}} \end{bmatrix} \quad \boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^{\mathcal{I}} \\ \boldsymbol{\mu}_i^{\mathcal{O}} \end{bmatrix} \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{O}} \end{bmatrix}$$

$\mathcal{P}(\boldsymbol{x}^{\mathcal{O}}|\boldsymbol{x}^{\mathcal{I}})$ can be computed as the multimodal conditional distribution

$$\begin{aligned} \mathcal{P}(\boldsymbol{x}^{\mathcal{O}}|\boldsymbol{x}^{\mathcal{I}}) &= \sum_{i=1}^{K} h_i \, \mathcal{N}\Big(\boldsymbol{x}^{\mathcal{O}} \Big| \hat{\boldsymbol{\mu}}_i^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}}\Big), \\ \text{with} \quad \hat{\boldsymbol{\mu}}_i^{\mathcal{O}} &= \boldsymbol{\mu}_i^{\mathcal{O}} + \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} (\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}}), \\ \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} &= \boldsymbol{\Sigma}_i^{\mathcal{O}} - \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}} \\ \text{and} \quad h_i &= \frac{\pi_i \, \mathcal{N}(\boldsymbol{x}^{\mathcal{I}} |\, \boldsymbol{\mu}_i^{\mathcal{I}}, \boldsymbol{\Sigma}_i^{\mathcal{I}})}{\sum_k^K \pi_k \, \mathcal{N}(\boldsymbol{x}^{\mathcal{I}} |\, \boldsymbol{\mu}_k^{\mathcal{I}}, \boldsymbol{\Sigma}_k^{\mathcal{I}})}, \end{aligned}$$

computed with the marginal

$$\mathcal{N}(\boldsymbol{x}^{\mathcal{I}} |\, \boldsymbol{\mu}_i^{\mathcal{I}}, \boldsymbol{\Sigma}_i^{\mathcal{I}}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}_i^{\mathcal{I}}|^{-\frac{1}{2}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}})^{\top} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} (\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}})\Big).$$

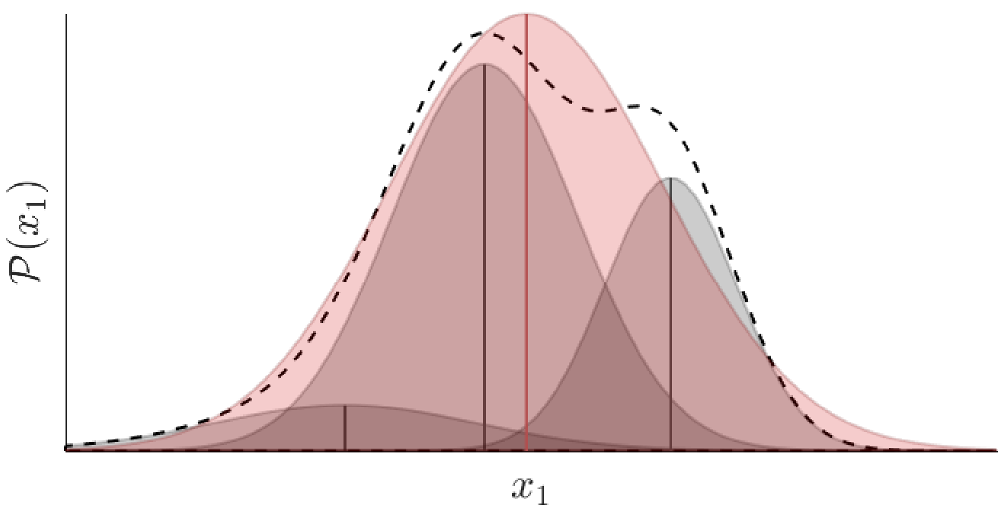
Gaussian estimate of a mixture of Gaussians

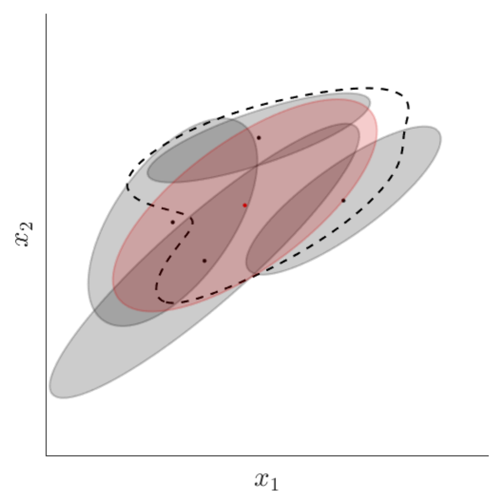
We can approximate a mixture of Gaussians $\sum_{i=1}^{K} h_i \, \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with a single Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, by **moment matching of the means (first moments) and covariances (second moments)** with

$$\boldsymbol{\mu} = \sum_{i=1}^{K} h_i \; \boldsymbol{\mu}_i,$$

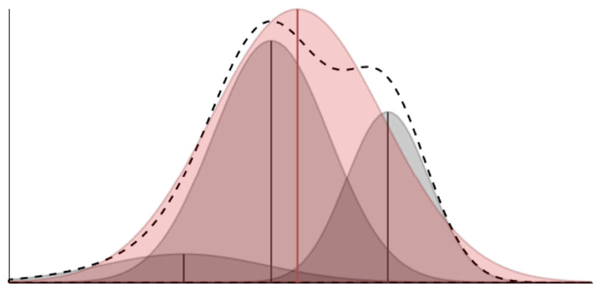
$$\boldsymbol{\Sigma} = \sum_{i=1}^{K} h_i \Big(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i \boldsymbol{\mu}_i^{\top} \Big) - \boldsymbol{\mu} \boldsymbol{\mu}^{\top},$$

also referred to as the **law of total mean and (co)variance**.





Gaussian mixture regression (GMR)



$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{\mathcal{I}} \\ \boldsymbol{x}^{\mathcal{O}} \end{bmatrix} \quad \boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^{\mathcal{I}} \\ \boldsymbol{\mu}_i^{\mathcal{O}} \end{bmatrix} \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}} \\ \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} & \boldsymbol{\Sigma}_i^{\mathcal{O}} \end{bmatrix}$$

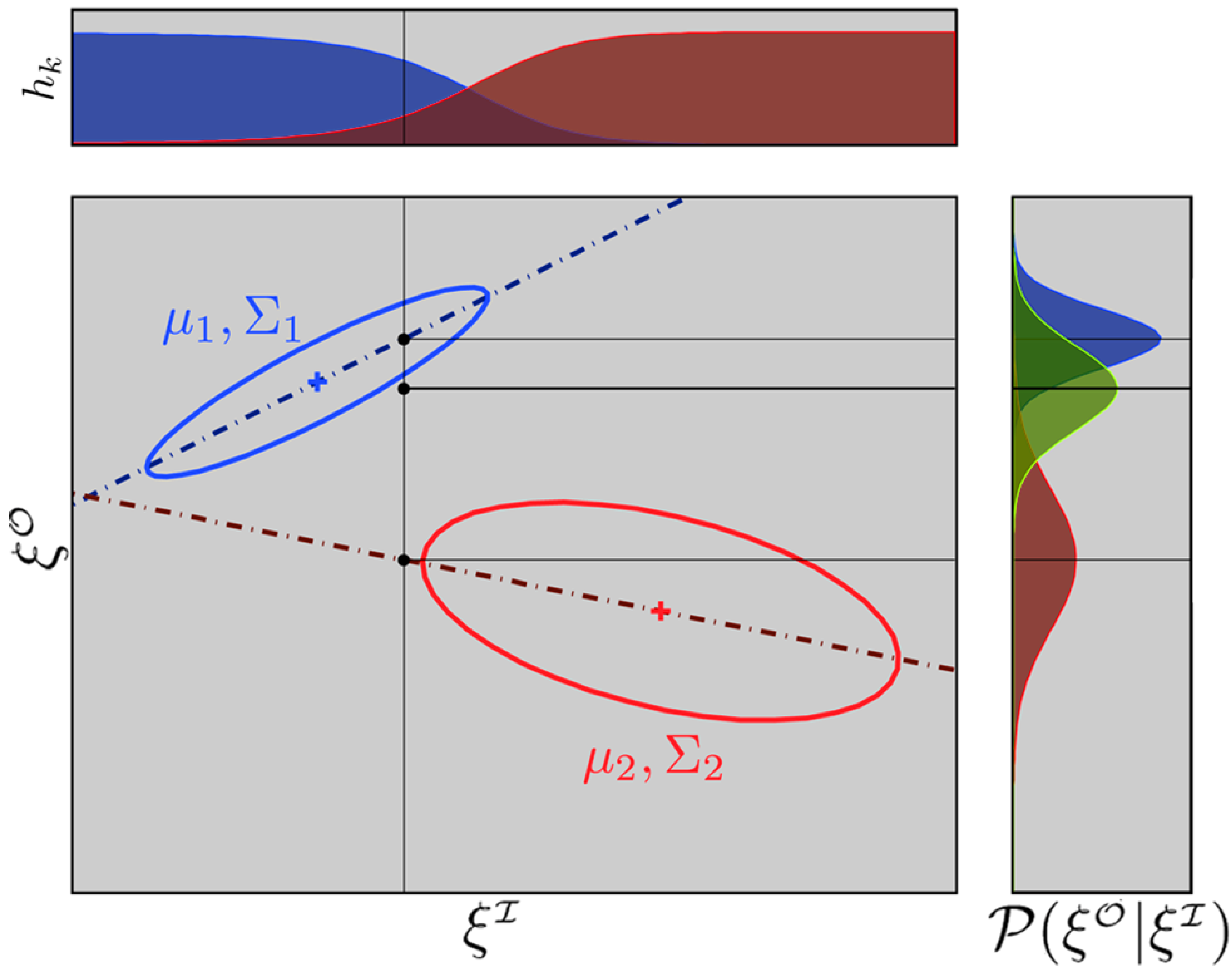
$$\begin{aligned} \hat{\boldsymbol{\mu}}_i^{\mathcal{O}} &= \boldsymbol{\mu}_i^{\mathcal{O}} + \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} (\boldsymbol{x}^{\mathcal{I}} - \boldsymbol{\mu}_i^{\mathcal{I}}) \\ \hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} &= \boldsymbol{\Sigma}_i^{\mathcal{O}} - \boldsymbol{\Sigma}_i^{\mathcal{O}\mathcal{I}} \boldsymbol{\Sigma}_i^{\mathcal{I}-1} \boldsymbol{\Sigma}_i^{\mathcal{I}\mathcal{O}} \end{aligned}$$

In GMR, an output distribution as a single multivariate Gaussian can be evaluated by moment matching of the means and covariances. The resulting Gaussian distribution $\mathcal{N}(\hat{\boldsymbol{\mu}}^{\mathcal{O}}, \hat{\boldsymbol{\Sigma}}^{\mathcal{O}})$ has parameters

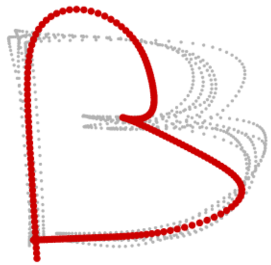
$$\hat{\boldsymbol{\mu}}^{\mathcal{O}} = \sum_{i=1}^{K} h_i \; \hat{\boldsymbol{\mu}}_i^{\mathcal{O}},$$

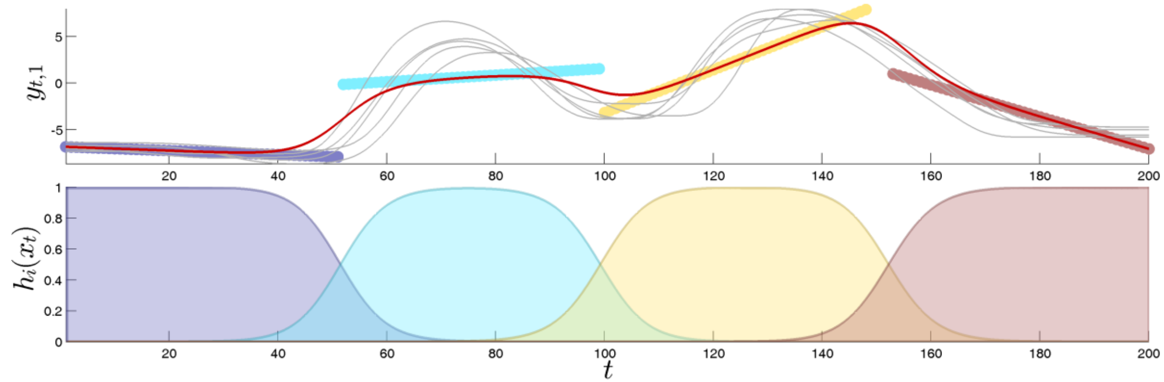
$$\hat{\boldsymbol{\Sigma}}^{\mathcal{O}} = \sum_{i=1}^{K} h_i \Big(\hat{\boldsymbol{\Sigma}}_i^{\mathcal{O}} + \hat{\boldsymbol{\mu}}_i^{\mathcal{O}} \; \hat{\boldsymbol{\mu}}_i^{\mathcal{O}\top} \Big) - \hat{\boldsymbol{\mu}}^{\mathcal{O}} \hat{\boldsymbol{\mu}}^{\mathcal{O}\top}.$$

Gaussian mixture regression (GMR)

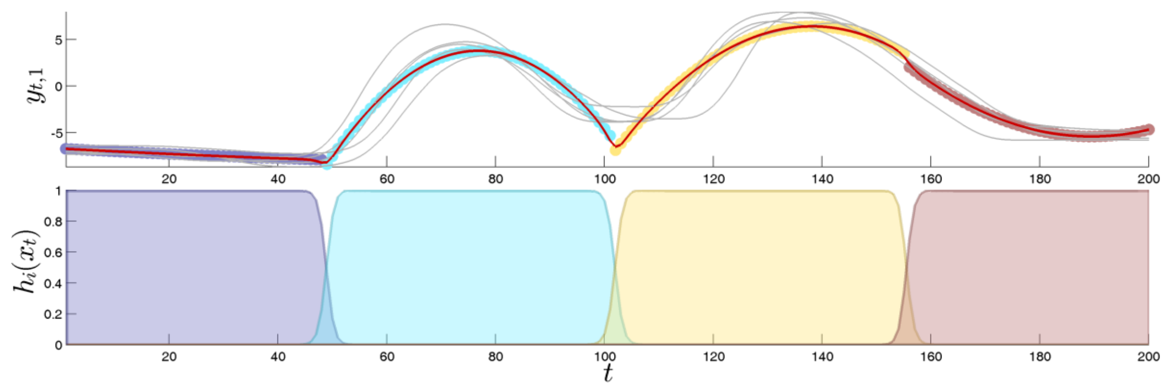


GMR for smooth piecewise polynomial fitting

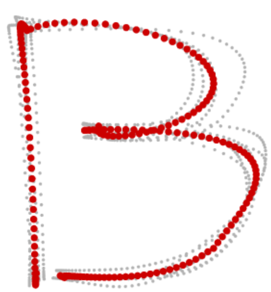


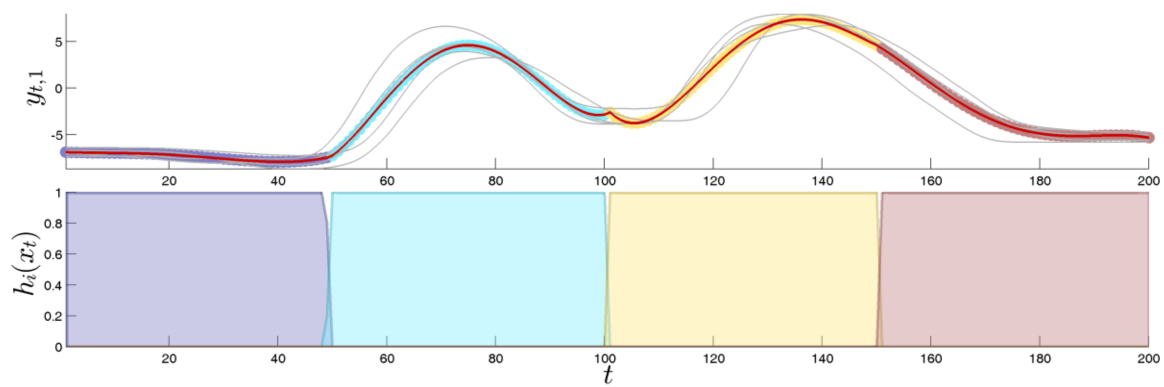


$$\boldsymbol{X} = \boldsymbol{x}$$



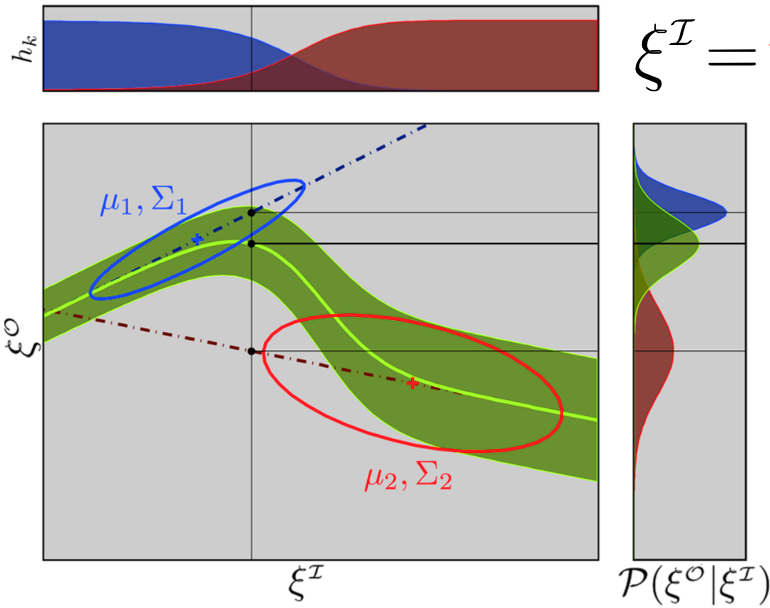
$$\boldsymbol{X} = [\boldsymbol{x}^2, \boldsymbol{x}]$$





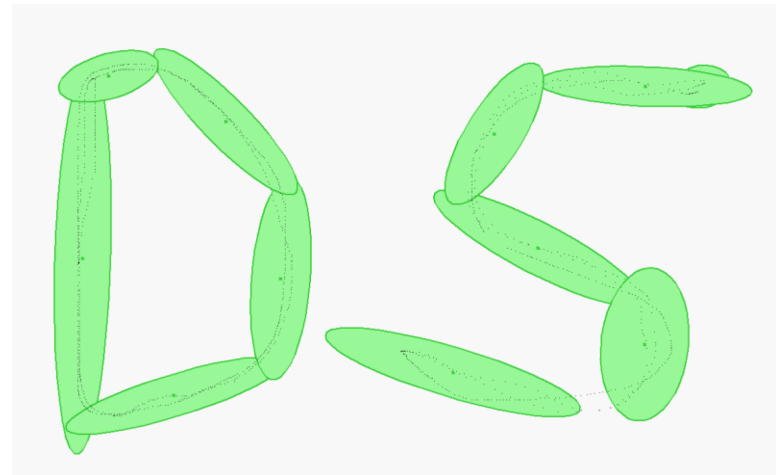
$$\boldsymbol{X} = [\boldsymbol{x}^3, \boldsymbol{x}^2, \boldsymbol{x}]$$

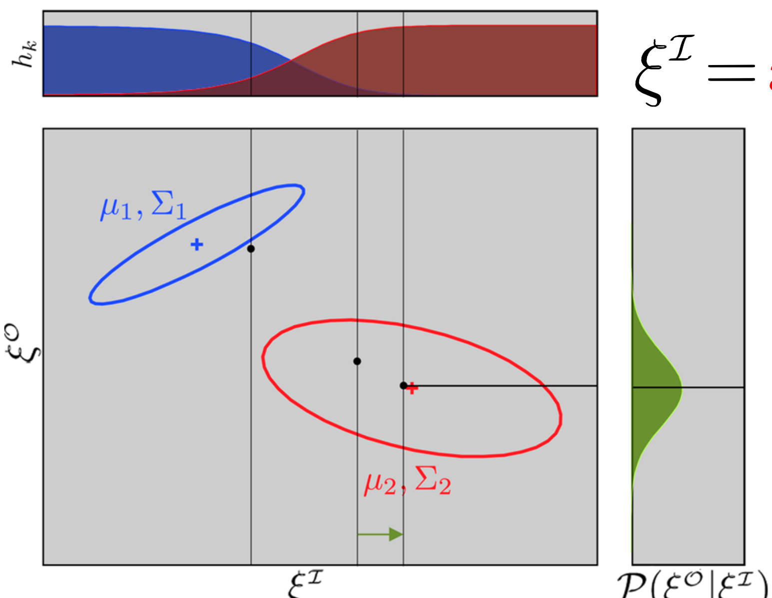
Gaussian mixture regression - Examples



$\xi^{\mathcal{I}} = t, \;\; \xi^{\mathcal{O}} = x$

[Calinon, Guenter and Billard,
IEEE Trans. on SMC-B 37(2), 2007]





$\xi^{\mathcal{I}} = x, \;\; \xi^{\mathcal{O}} = \dot{x}$

With expectation-maximization (EM):
(maximizing log-likelihood)

[Hersch, Guenter, Calinon and Billard,
IEEE Trans. on Robotics 24(6), 2008]

With quadratic programming solver:
(maximizing log-likelihood s.t. stability constraints)

[Khansari-Zadeh and Billard,
IEEE Trans. on Robotics 27(5), 2011]

Gaussian mixture regression (GMR) - Resources

Softwares

http://www.idiap.ch/software/pbdlib/

Matlab codes: demo_GMR01.m

C++ codes: demo_GMR01.cpp

References

[Ghahramani and Jordan, *"Supervised learning from incomplete data via an EM approach"*, NIPS'1994]

[Calinon, *"A tutorial on task-parameterized movement learning and retrieval"*, Intelligent Service Robotics 9(1), 2016]

[Calinon and Lee, *"Learning Control"*, Humanoid Robotics: a Reference (Springer), 2019]

Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**
  Product of Gaussians

- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

- **Differential geometry**
  Riemannian manifolds

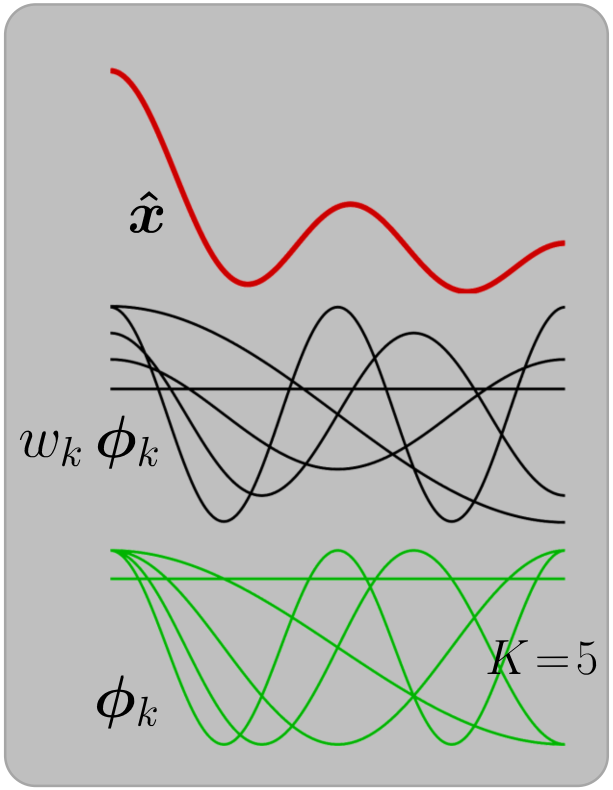
Superposition with basis functions

$$\hat{\boldsymbol{x}} = \sum_{k=1}^{K} w_k \, \boldsymbol{\phi}_k$$

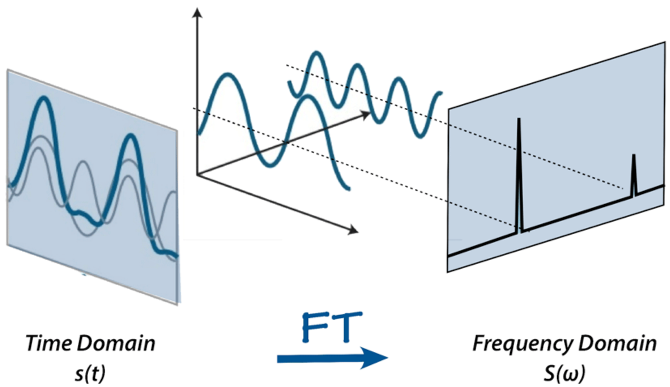
Bézier curves
(Bernstein bases)

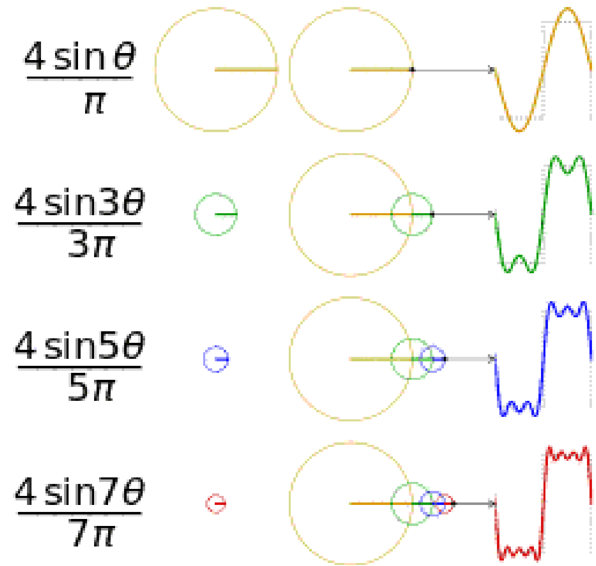
Locally weighted regression (radial bases)

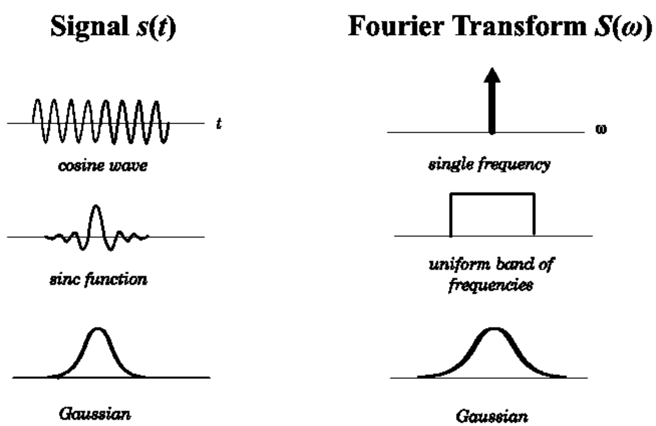
Fourier series
(cosine bases)

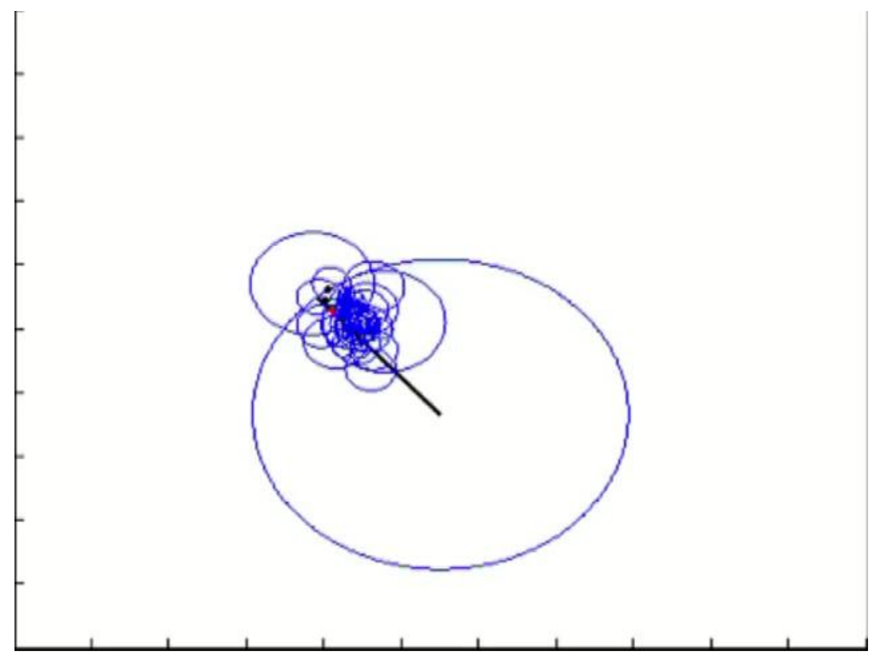


Fourier series



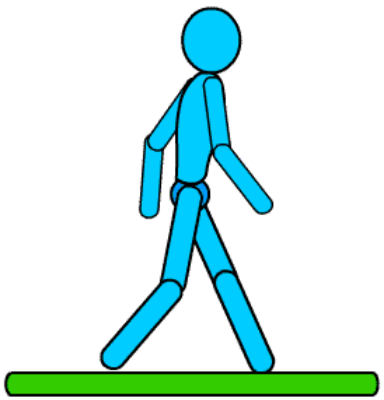


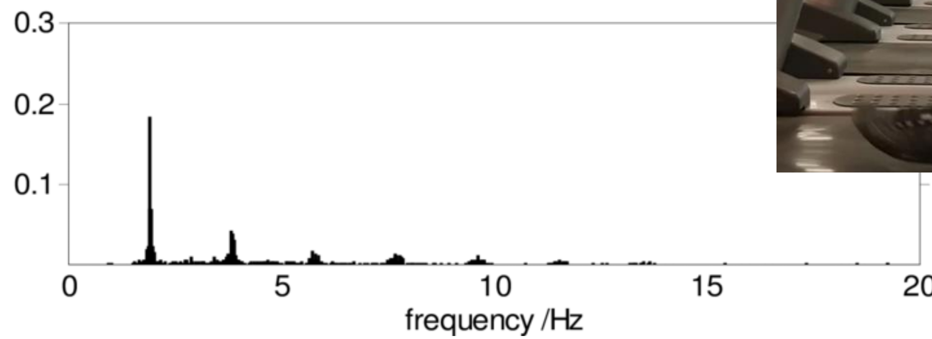


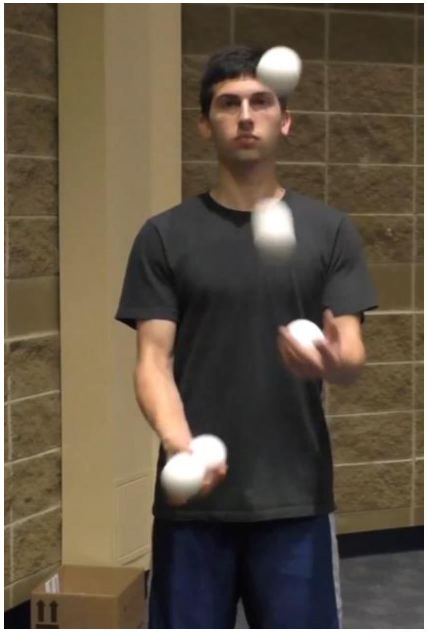


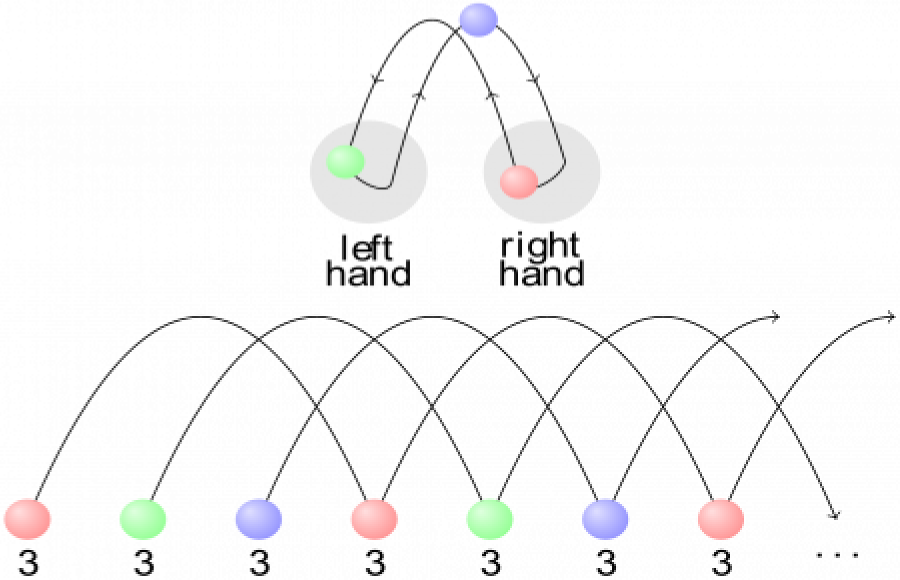
$$
\begin{aligned}
f_k(x) &= \exp\left(i\frac{2\pi kx}{L}\right) \\
&= \cos\left(\frac{2\pi kx}{L}\right) + i\sin\left(\frac{2\pi kx}{L}\right)
\end{aligned}
$$

Periodic movements









Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

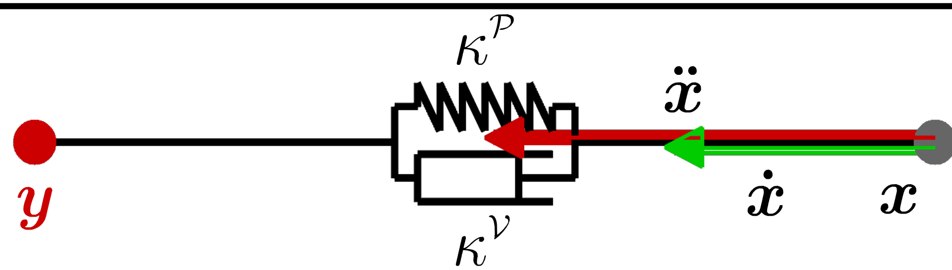
- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**
  Product of Gaussians

- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

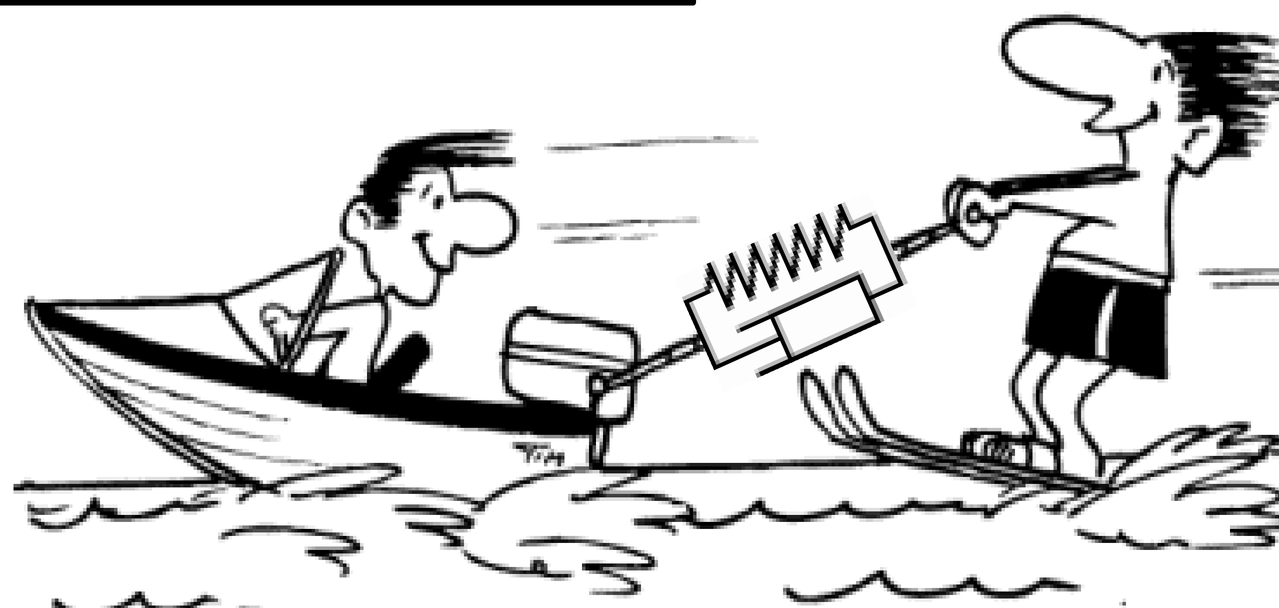
- **Differential geometry**
  Riemannian manifolds

Spring-damper system



$$\ddot{\boldsymbol{x}} = \kappa^{\mathcal{P}}[\boldsymbol{y} - \boldsymbol{x}] - \kappa^{\mathcal{V}}\dot{\boldsymbol{x}}$$

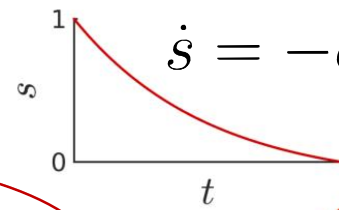
$$\Rightarrow \boldsymbol{y} = \frac{1}{\kappa^{\mathcal{P}}}\ddot{\boldsymbol{x}} + \frac{\kappa^{\mathcal{V}}}{\kappa^{\mathcal{P}}}\dot{\boldsymbol{x}} + \boldsymbol{x}$$



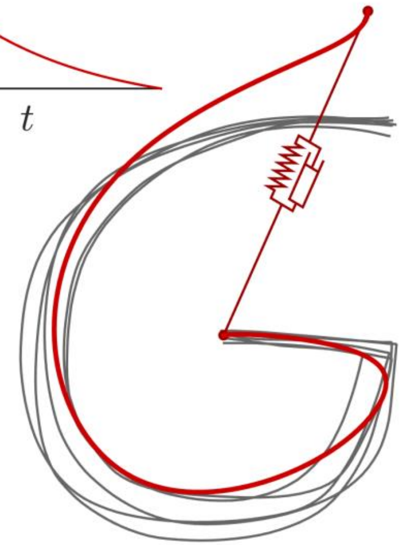
Dynamical movement primitives (DMP)

$$\ddot{\boldsymbol{x}} = k^{\mathcal{P}}(\boldsymbol{\mu}_T - \boldsymbol{x}) - k^{\mathcal{V}}\dot{\boldsymbol{x}} \; + \; \boldsymbol{f}(s)$$

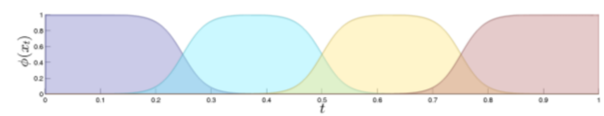
$$\dot{s} = -\alpha s$$



$$\boldsymbol{f}(s) = \; s \sum_{k=1}^{K} \phi_k(s)\, \boldsymbol{F}_k$$



$$\boldsymbol{X}^{\mathcal{O}} = \begin{bmatrix} \ddot{\boldsymbol{x}}_1 - k^{\mathcal{P}}(\boldsymbol{\mu}_T - \boldsymbol{x}_1) + k^{\mathcal{V}}\dot{\boldsymbol{x}}_1 \\ \ddot{\boldsymbol{x}}_2 - k^{\mathcal{P}}(\boldsymbol{\mu}_T - \boldsymbol{x}_2) + k^{\mathcal{V}}\dot{\boldsymbol{x}}_2 \\ \vdots \\ \ddot{\boldsymbol{x}}_T - k^{\mathcal{P}}(\boldsymbol{\mu}_T - \boldsymbol{x}_T) + k^{\mathcal{V}}\dot{\boldsymbol{x}}_T \end{bmatrix} \quad \boldsymbol{X}^{\mathcal{I}} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_T \end{bmatrix}$$

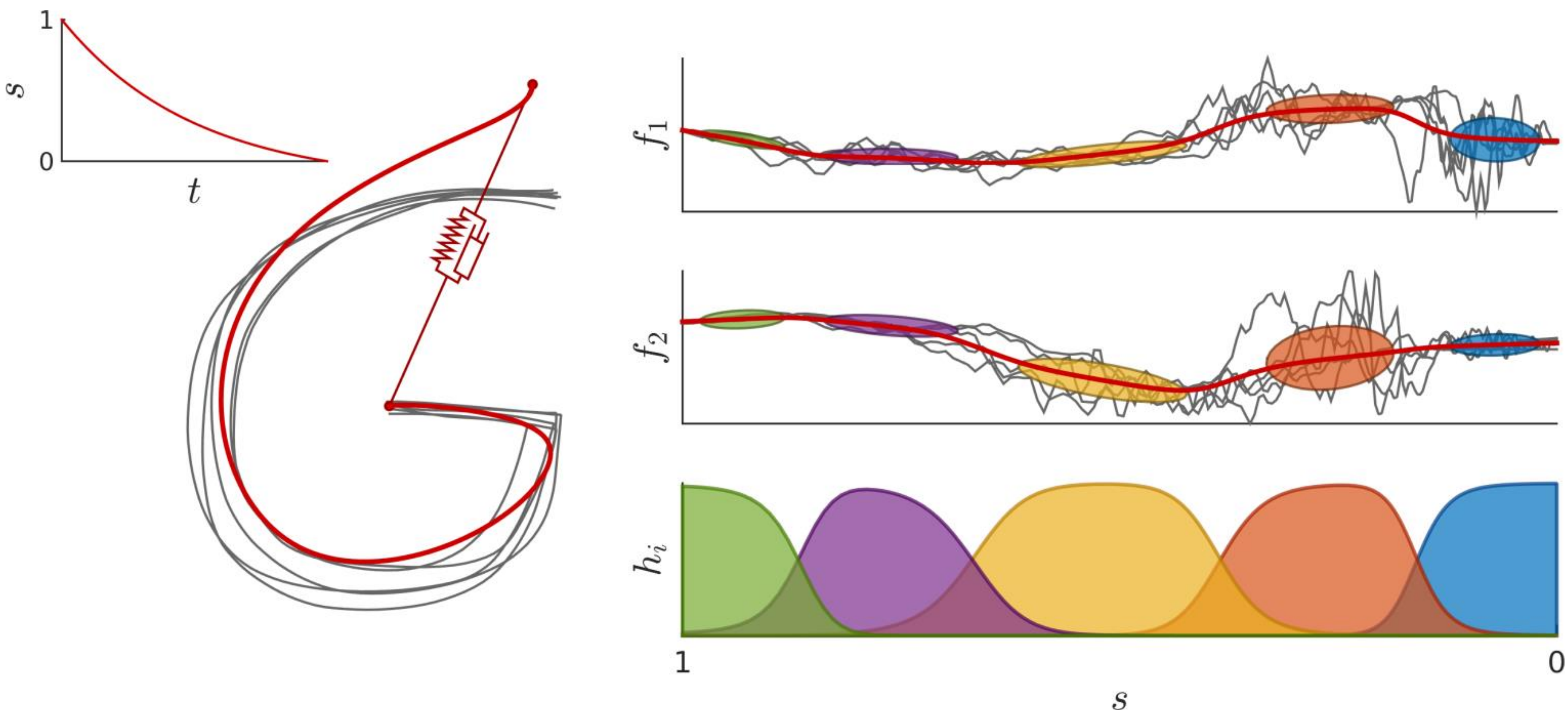


$$\boldsymbol{W}_k = \mathrm{diag}\Big(\phi_k(s_1), \phi_k(s_2), \dots, \phi_k(s_T)\Big)$$

$$\hat{\boldsymbol{F}}_k = ({\boldsymbol{X}^{\mathcal{I}}}^{\top} \boldsymbol{W}_k \boldsymbol{X}^{\mathcal{I}})^{-1} {\boldsymbol{X}^{\mathcal{I}}}^{\top} \boldsymbol{W}_k\, \boldsymbol{X}^{\mathcal{O}}$$

Dynamical movement primitives with GMR

Learning of $\mathcal{P}(s, \boldsymbol{x})$ and retrieval of $\mathcal{P}(\boldsymbol{x}|s)$



Dynamical movement primitives (DMP) - Resources

Softwares

http://www.idiap.ch/software/pbdlib/

Matlab codes: demo_DMP01.m

References

[Ijspeert, Nakanishi and Schaal, *"Learning Control Policies For Movement Imitation and Movement recognition"*, NIPS'2003]

[Ijspeert, Nakanishi, Pastor, Hoffmann and Schaal, *"Dynamical movement primitives: Learning attractor models for motor behaviors"*, Neural Computation 25(2), 2013]

[Calinon and Lee, *"Learning Control"*, Humanoid Robotics: a Reference (Springer), 2019]

Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

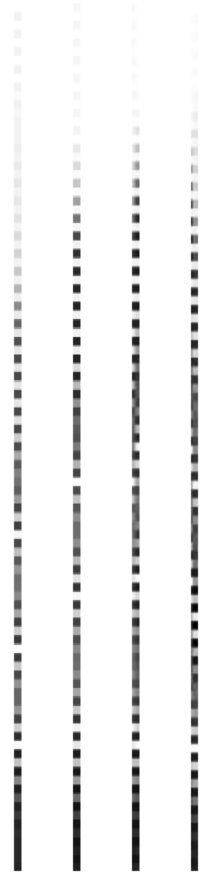
- **Superposition Vs fusion**
  Product of Gaussians

- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

- **Differential geometry**
  Riemannian manifolds

Trajectory distribution

$$\boldsymbol{x}_m = \begin{bmatrix} \boldsymbol{x}_{m,1} \\ \boldsymbol{x}_{m,2} \\ \vdots \\ \boldsymbol{x}_{m,T} \end{bmatrix} \in \mathbb{R}^{DT}$$



M samples

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\frac{1}{M}\sum_{m=1}^{M} \boldsymbol{x}_m$$

$$\frac{1}{M}\sum_{m=1}^{M} (\boldsymbol{x}_m - \boldsymbol{\mu})(\boldsymbol{x}_m - \boldsymbol{\mu})^{\top}$$

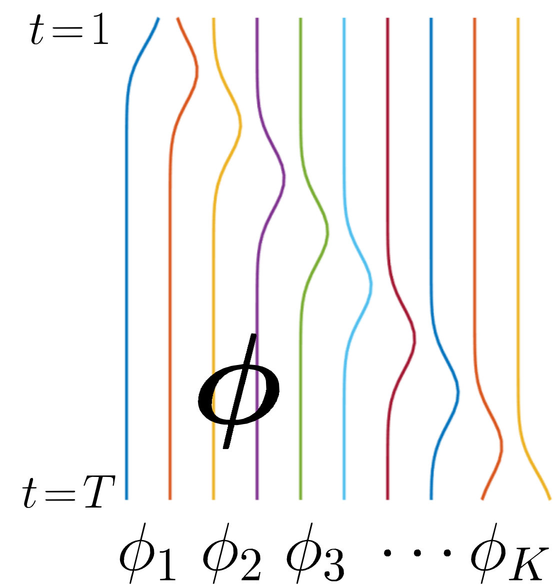
Probabilistic movement primitives (ProMP)

uni-dimensional trajectory:

$$\hat{\boldsymbol{x}} = \sum_{k=1}^{K} w_k\, \boldsymbol{\phi}_k$$

$$\hat{\boldsymbol{x}} = \boldsymbol{\phi}\, \boldsymbol{w}$$

$$\boxed{\boldsymbol{\phi} \in \mathbb{R}^{T\times K}} \qquad \boxed{\boldsymbol{w} \in \mathbb{R}^{K}}$$

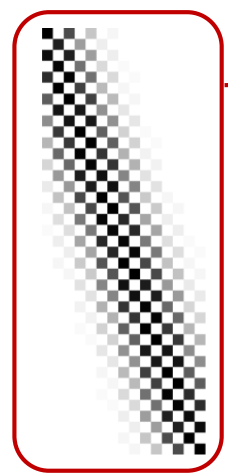


d-dimensional trajectory:

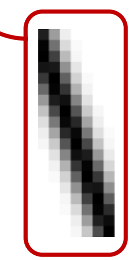
$$\hat{\boldsymbol{x}} = \boldsymbol{\Psi}\, \boldsymbol{w}$$

$$\boxed{\boldsymbol{\Psi} \in \mathbb{R}^{DT\times DK}} \qquad \boxed{\boldsymbol{w} \in \mathbb{R}^{DK}}$$

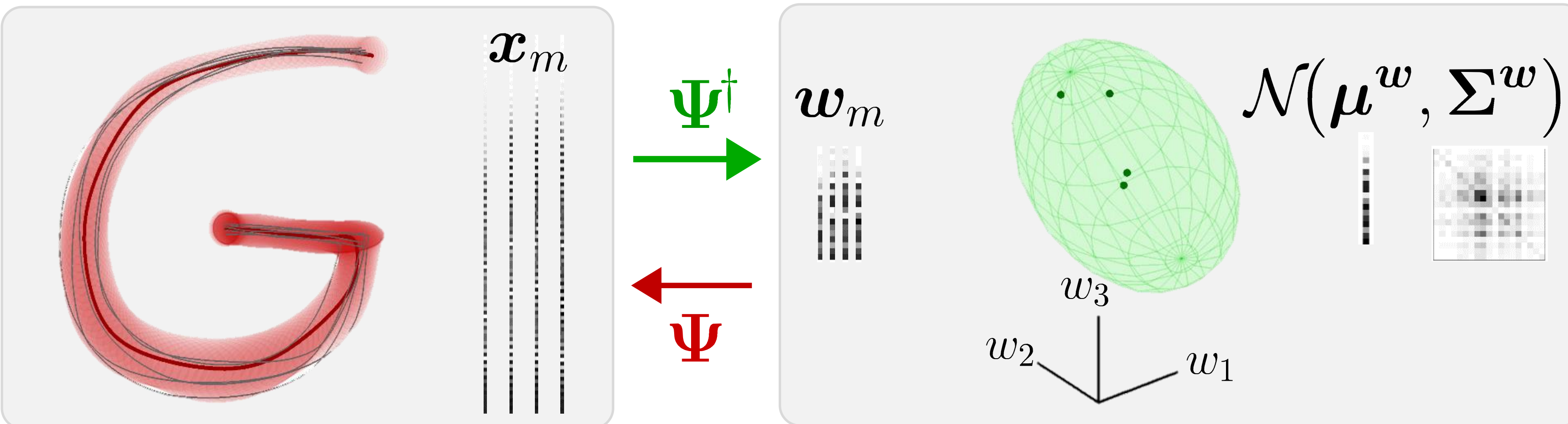
$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{I}\phi_1(t_1) & \boldsymbol{I}\phi_2(t_1) & \cdots & \boldsymbol{I}\phi_K(t_1) \\ \boldsymbol{I}\phi_1(t_2) & \boldsymbol{I}\phi_2(t_2) & \cdots & \boldsymbol{I}\phi_K(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I}\phi_1(t_T) & \boldsymbol{I}\phi_2(t_T) & \cdots & \boldsymbol{I}\phi_K(t_T) \end{bmatrix}$$

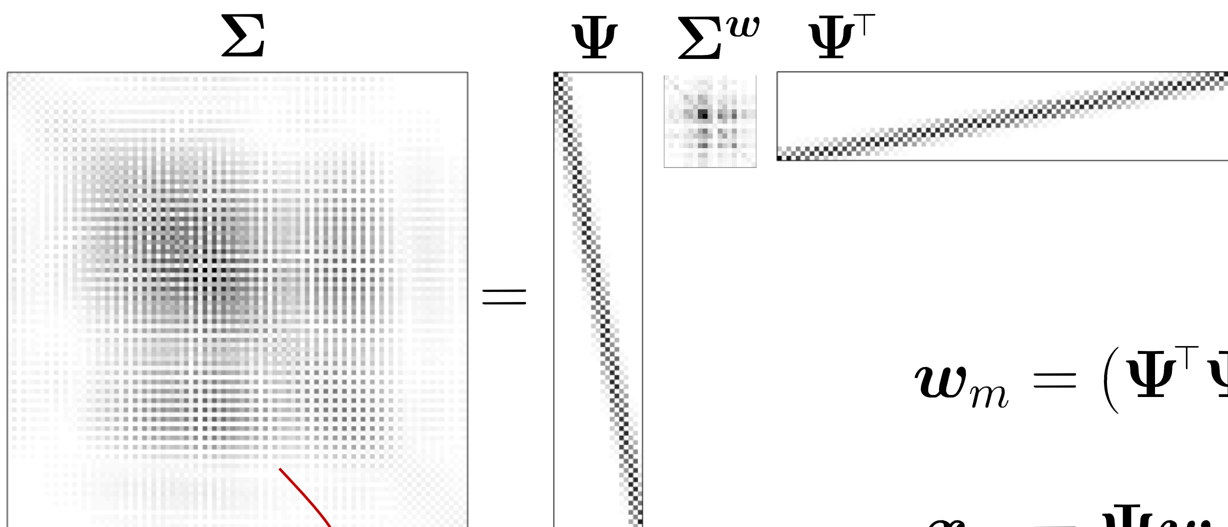


$$\boldsymbol{\Psi} = \boldsymbol{\phi} \otimes \boldsymbol{I}$$



Probabilistic movement primitives (ProMP)





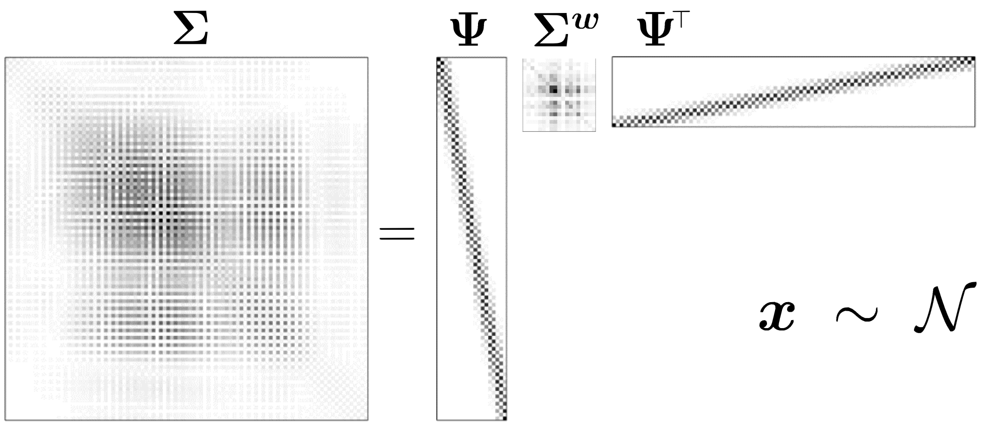
$$\boldsymbol{w}_m = \left(\boldsymbol{\Psi}^\top \boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^\top \boldsymbol{x}_m$$

$$\boldsymbol{x}_m = \boldsymbol{\Psi} \boldsymbol{w}_m + \boldsymbol{\epsilon}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \lambda \boldsymbol{I})$$

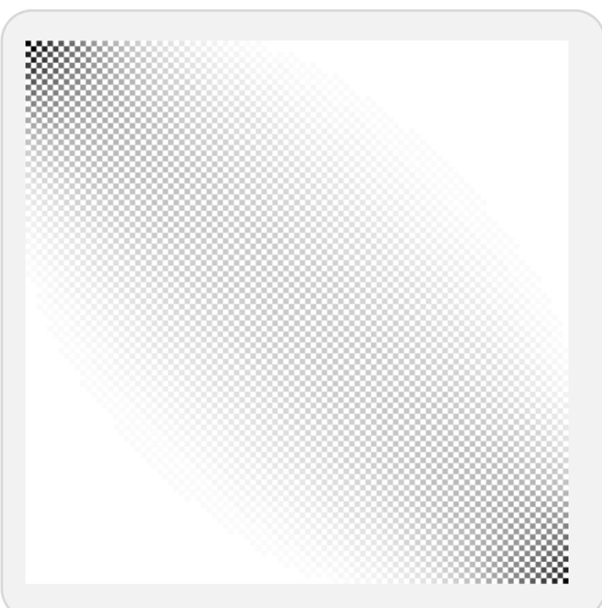
$$\boldsymbol{x} \sim \mathcal{N}\Big(\boldsymbol{\Psi}\boldsymbol{\mu}^w, \boldsymbol{\Psi}\boldsymbol{\Sigma}^w\boldsymbol{\Psi}^\top + \lambda \boldsymbol{I}\Big)$$

Probabilistic movement primitives (ProMP)

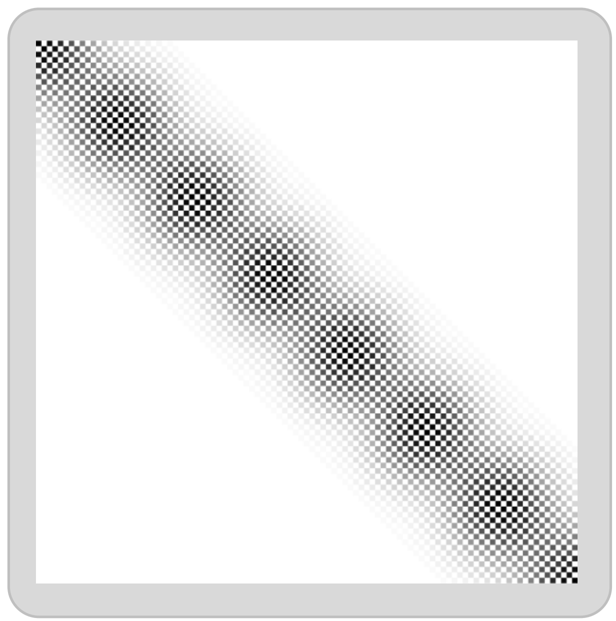


$$x \sim \mathcal{N}\left(\Psi \mu^w , \; \Psi \Sigma^w \Psi^\top + \lambda I\right)$$

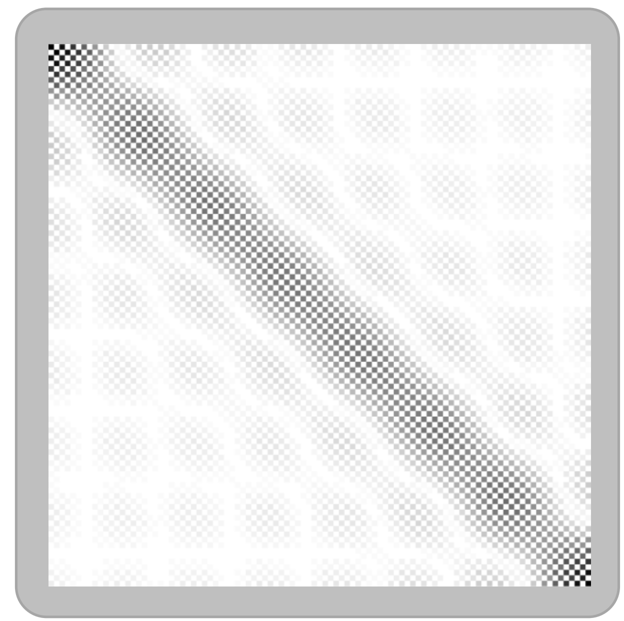
Bézier curves (Bernstein bases)



Locally weighted regression (radial bases)



Fourier series (cosine bases)



Probabilistic movement primitives - Resources

Softwares

http://www.idiap.ch/software/pbdlib/

C++ codes: demo_proMP01.cpp

References

[Paraschos, Daniel, Peters and Neumann, *"Probabilistic Movement Primitives"*, NIPS'2013]

[Calinon and Lee, *"Learning Control"*, Humanoid Robotics: a Reference (Springer), 2019]

Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**
  Product of Gaussians

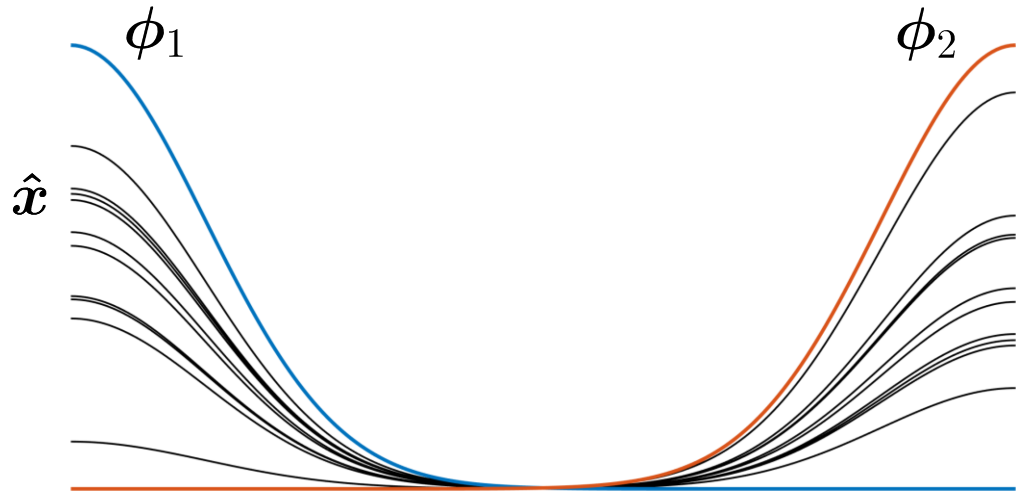
- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

- **Differential geometry**
  Riemannian manifolds

Combination of primitives as a fusion problem

Superposition

$$\hat{\boldsymbol{x}} = \frac{\sum_{k=1}^{K} w_k \, \boldsymbol{\phi}_k}{\sum_{k=1}^{K} w_k}$$

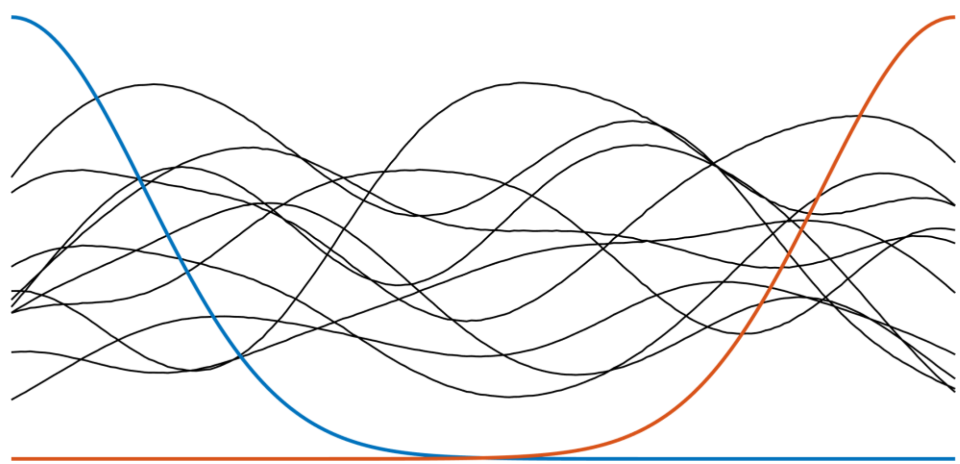


$K = 2$

Choosing between using scalar weights or full weight matrices does not look like a detail!

Fusion

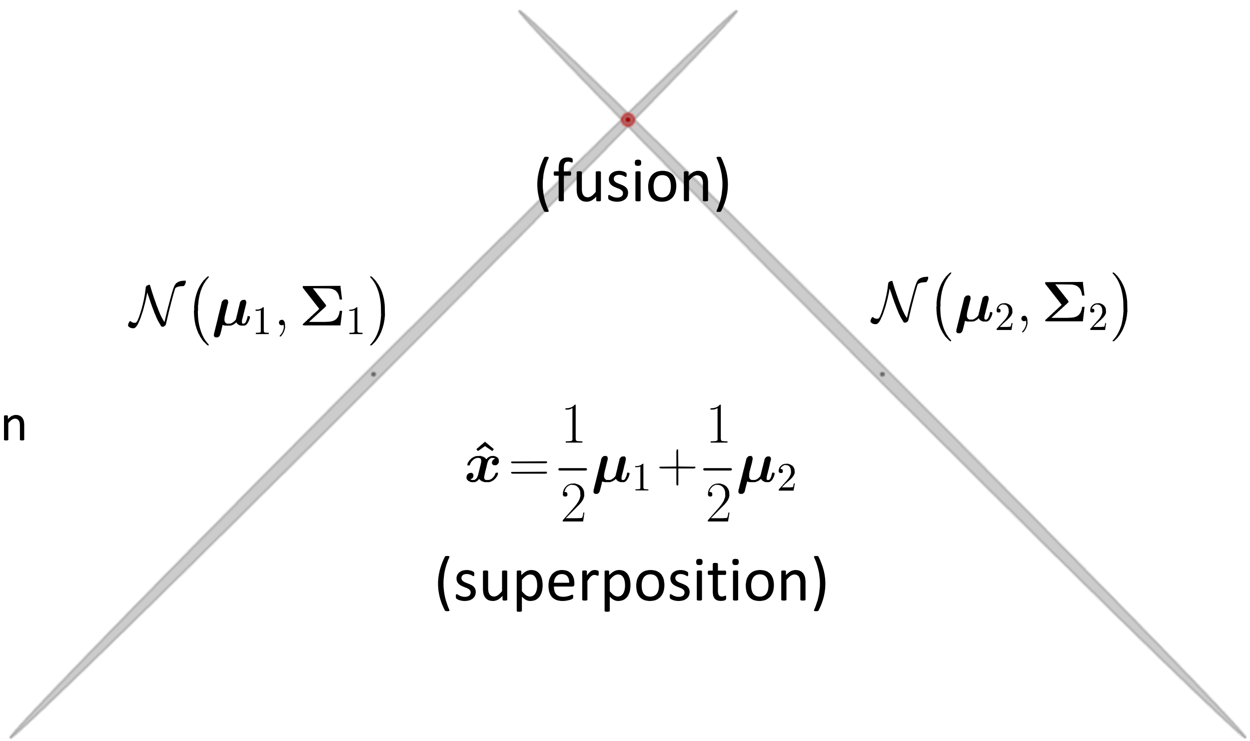
$$\hat{\boldsymbol{x}} = \left(\sum_{k=1}^{K} \boldsymbol{W}_k \right)^{-1} \sum_{k=1}^{K} \boldsymbol{W}_k \, \boldsymbol{\phi}_k$$



Motivating example:
A probabilistic view on segment crossing!

$$
\begin{aligned}
\hat{\boldsymbol{x}} &= \arg\min_{\boldsymbol{x}} \left\|\boldsymbol{\mu}_1 - \boldsymbol{x}\right\|^2_{\boldsymbol{W}_1} + \left\|\boldsymbol{\mu}_2 - \boldsymbol{x}\right\|^2_{\boldsymbol{W}_2} \\
&= (\boldsymbol{W}_1 + \boldsymbol{W}_2)^{-1} (\boldsymbol{W}_1 \boldsymbol{\mu}_1 + \boldsymbol{W}_2 \boldsymbol{\mu}_2) \\
&= \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1} \left(\boldsymbol{\Sigma}_1^{-1} \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_2^{-1} \boldsymbol{\mu}_2\right)
\end{aligned}
$$

Product of Gaussians



$\boldsymbol{\mu}_i$ center of the Gaussian

$\boldsymbol{\Sigma}_i$ covariance matrix

$\boldsymbol{W}_i$ precision matrix

$(\boldsymbol{W}_i = \boldsymbol{\Sigma}_i^{-1})$

Kalman filter

Kalman filter with feedback gains

$$\boldsymbol{\Sigma}_t = \left(\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{C}\right) \boldsymbol{\Sigma}_t^{(1)}$$

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_t^{(1)} + \boldsymbol{K}_t(\boldsymbol{y}_t - \boldsymbol{C}\boldsymbol{\mu}_t^{(1)})$$

$$\boldsymbol{K}_t = \boldsymbol{\Sigma}_t^{(1)} \boldsymbol{C}^{\top} \left(\boldsymbol{\Sigma}_{\boldsymbol{y}} + \boldsymbol{C}\boldsymbol{\Sigma}_t^{(1)} \boldsymbol{C}^{\top}\right)^{-1}$$

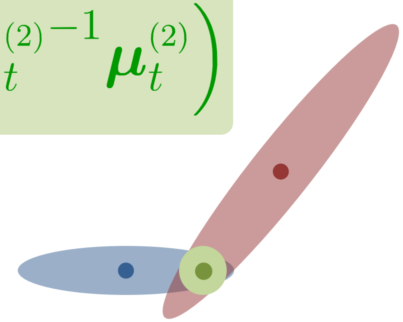
$$\boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{e}_{\boldsymbol{y}}$$

$$\boldsymbol{e}_{\boldsymbol{y}} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{y}}\right)$$

Kalman filter as product of Gaussians

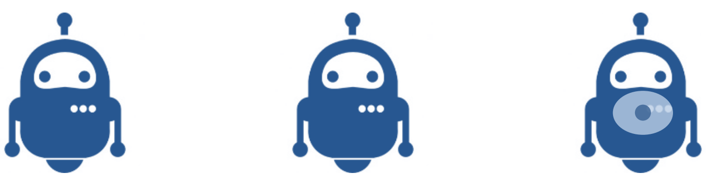
$$\boldsymbol{\Sigma}_t = \left({\boldsymbol{\Sigma}_t^{(1)}}^{-1} + {\boldsymbol{\Sigma}_t^{(2)}}^{-1}\right)^{-1}$$

$$\boldsymbol{\mu}_t = \boldsymbol{\Sigma}_t \left({\boldsymbol{\Sigma}_t^{(1)}}^{-1} \boldsymbol{\mu}_t^{(1)} + {\boldsymbol{\Sigma}_t^{(2)}}^{-1} \boldsymbol{\mu}_t^{(2)}\right)$$



$$\boldsymbol{\mu}_t^{(2)} \triangleq \boldsymbol{C}^{\dagger} \boldsymbol{y}_t$$

$$\boldsymbol{\Sigma}_t^{(2)} \triangleq \boldsymbol{C}^{\dagger} \boldsymbol{\Sigma}_{\boldsymbol{y}} {\boldsymbol{C}^{\dagger}}^{\top}$$



t=0 t=1 t=2

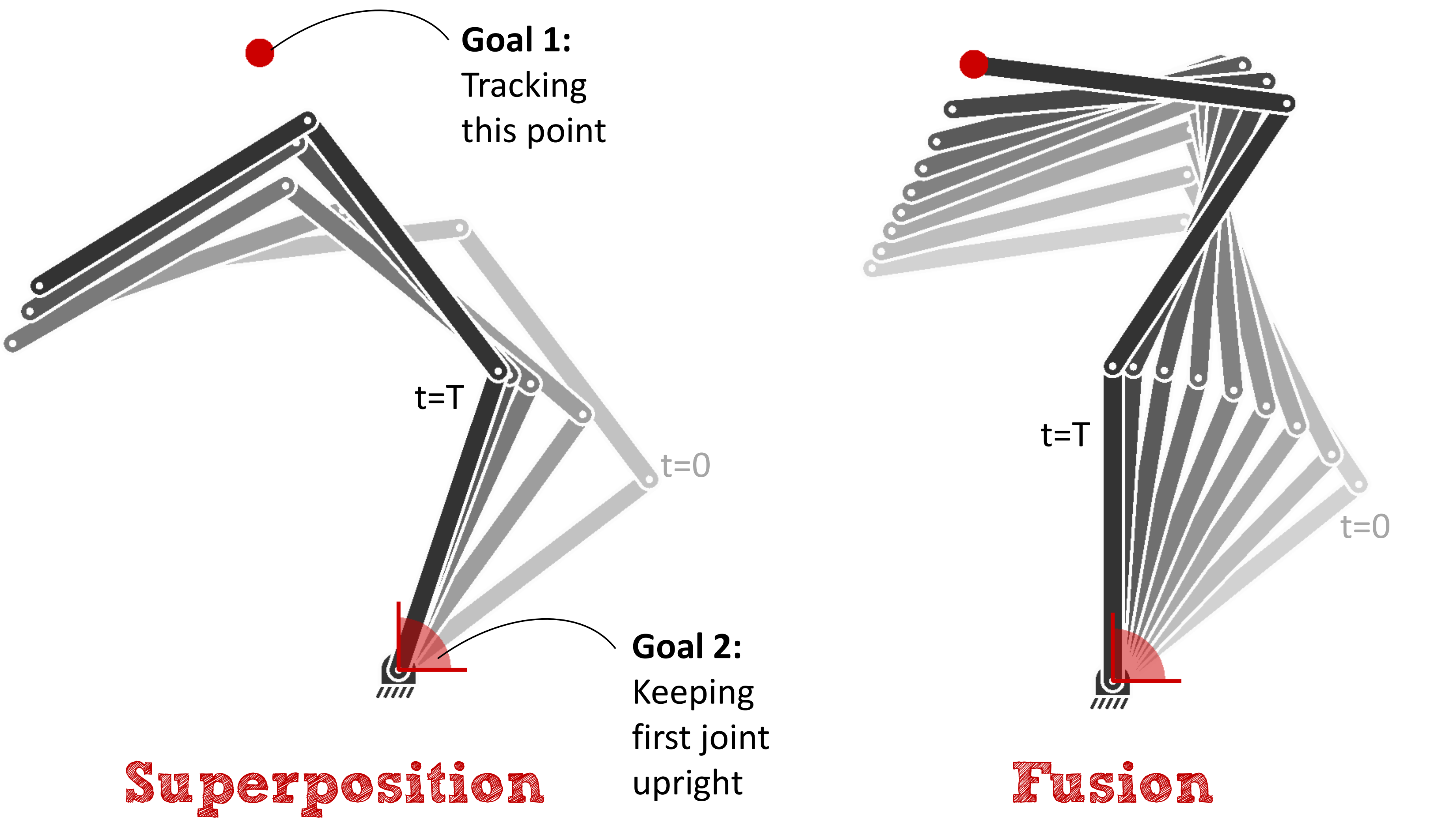
$$\boldsymbol{x}_t = \boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{B}\boldsymbol{u}_t + \boldsymbol{e}_{\boldsymbol{x}}$$

$$\boldsymbol{e}_{\boldsymbol{x}} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Sigma}_{\boldsymbol{x}}\right)$$

$$\boldsymbol{\mu}_t^{(1)} \triangleq \boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{B}\boldsymbol{u}_t$$

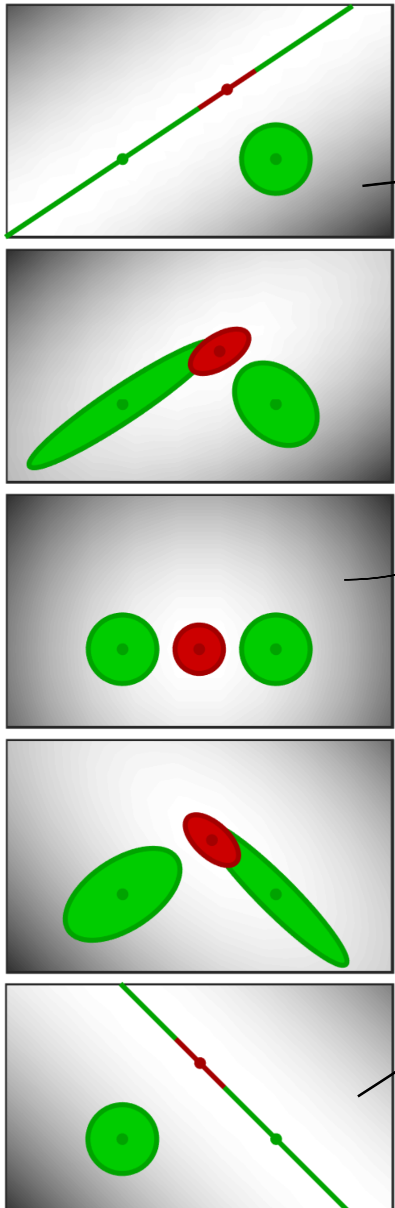
$$\boldsymbol{\Sigma}_t^{(1)} \triangleq \boldsymbol{A}\boldsymbol{\Sigma}_{t-1}\boldsymbol{A}^{\top} + \boldsymbol{\Sigma}_{\boldsymbol{x}}$$

Motivating example:
Fusion of IK and joint angle controllers



Combination of primitives as a fusion problem

$$\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right) \propto \mathcal{N}\left(\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}\right) \mathcal{N}\left(\boldsymbol{\mu}^{(2)}, \boldsymbol{\Sigma}^{(2)}\right)$$



Null space projection
(hierarchy constraints)

Scalar superposition

The full weight matrices approach covers both **scalar weights** (with isotropic diagonal matrix) and **null space projection** operations!

Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**
  Product of Gaussians

- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

- **Differential geometry**
  Riemannian manifolds

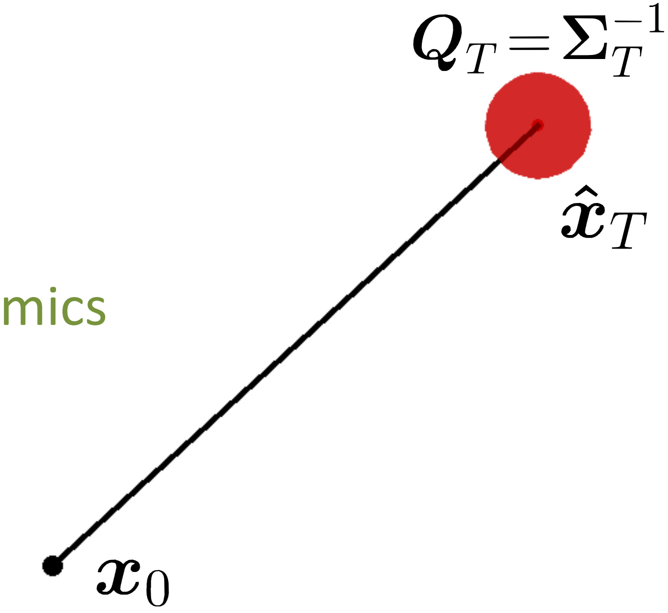
Linear quadratic tracking (LQT)

Track path! Use low control commands!

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \left\|\boldsymbol{\mu}_t - \boldsymbol{x}_t\right\|^2_{\boldsymbol{Q}_t} + \left\|\boldsymbol{u}_t\right\|^2_{\boldsymbol{R}_t}$$

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t$$

System dynamics



Model predictive control (MPC):

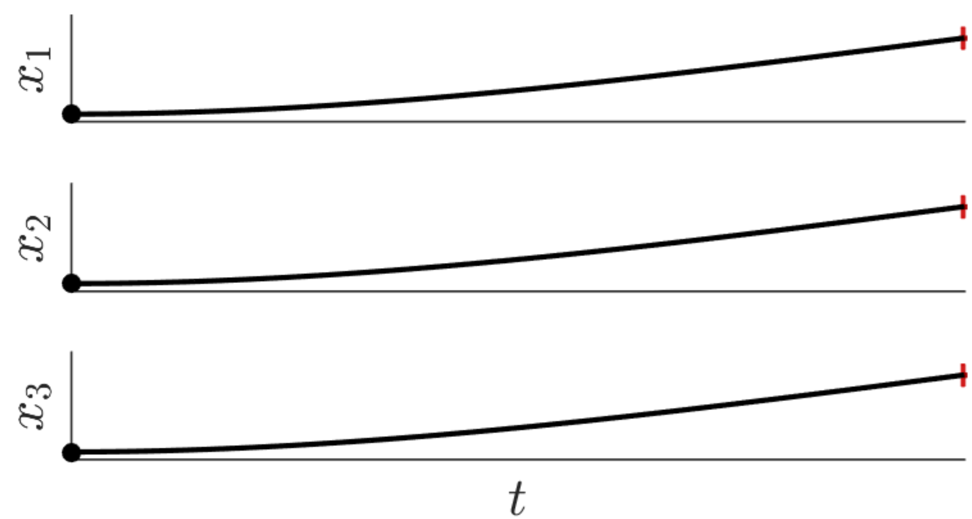
$\boldsymbol{x}_t$ state variable (position+velocity)

$\boldsymbol{\mu}_t$ desired state

$\boldsymbol{u}_t$ control command (acceleration)

$\boldsymbol{Q}_t$ precision matrix

$\boldsymbol{R}_t$ control weight matrix



How to solve this objective function?

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \overbrace{\left\|\boldsymbol{\mu}_t - \boldsymbol{x}_t\right\|^2_{\boldsymbol{Q}_t}}^{\text{Track path!}} + \overbrace{\left\|\boldsymbol{u}_t\right\|^2_{\boldsymbol{R}_t}}^{\text{Use low control commands!}}$$

$$\text{s.t.} \quad \underbrace{\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t}_{\text{System dynamics}}$$

Pontryagin's max. principle, Riccati equation, Hamilton-Jacobi-Bellman

(the Physicist perspective)

Differential dynamic programming

(the Computer Scientist perspective)

Linear algebra

(the Algebraist perspective)

Let's first re-organize the objective function...

$$c = \sum_{t=1}^{T} \left((\boldsymbol{\mu}_t - \boldsymbol{x}_t)^\top \boldsymbol{Q}_t (\boldsymbol{\mu}_t - \boldsymbol{x}_t) \;+\; \boldsymbol{u}_t^\top \boldsymbol{R}_t \, \boldsymbol{u}_t \right)$$

$$= (\boldsymbol{\mu} - \boldsymbol{x})^\top \boldsymbol{Q} (\boldsymbol{\mu} - \boldsymbol{x}) \;+\; \boldsymbol{u}^\top \boldsymbol{R} \boldsymbol{u}$$

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{Q}_T \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{R}_T \end{bmatrix}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_T \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix} \qquad \boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_T \end{bmatrix}$$

Let's then re-organize the constraint...

$$\boxed{\boldsymbol{x}_{t+1} = \boldsymbol{A}\,\boldsymbol{x}_t + \boldsymbol{B}\,\boldsymbol{u}_t}$$

$$\begin{aligned}
\boldsymbol{x}_2 &= \boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{u}_1 \\
\boldsymbol{x}_3 &= \boldsymbol{A}\boldsymbol{x}_2 + \boldsymbol{B}\boldsymbol{u}_2 = \boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{u}_1) + \boldsymbol{B}\boldsymbol{u}_2 \\
&\vdots \\
\boldsymbol{x}_T &= \boldsymbol{A}^{T-1}\boldsymbol{x}_1 + \boldsymbol{A}^{T-2}\boldsymbol{B}\boldsymbol{u}_1 + \boldsymbol{A}^{T-3}\boldsymbol{B}\boldsymbol{u}_2 + \cdots + \boldsymbol{B}_{T-1}\boldsymbol{u}_{T-1}
\end{aligned}$$

$$\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^{T-1} \end{bmatrix}}_{\boldsymbol{S}^{\boldsymbol{x}}} \boldsymbol{x}_1 + \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ \boldsymbol{B} & 0 & \cdots & 0 & 0 \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{A}^{T-2}\boldsymbol{B} & \boldsymbol{A}^{T-3}\boldsymbol{B} & \cdots & \boldsymbol{B} & 0 \end{bmatrix}}_{\boldsymbol{S}^{\boldsymbol{u}}} \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_T \end{bmatrix}$$

$$\boxed{\boldsymbol{x} = \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1 + \boldsymbol{S}^{\boldsymbol{u}}\boldsymbol{u}}$$

Linear quadratic tracking: Analytic solution

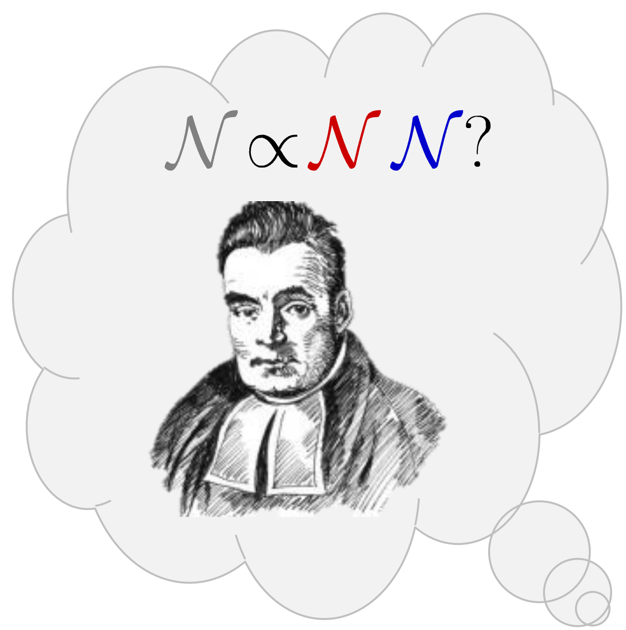
The constraint can then be put into the objective function:

$$\boldsymbol{x} = \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1 + \boldsymbol{S}^{\boldsymbol{u}}\boldsymbol{u}$$

$$\begin{aligned} c &= (\boldsymbol{\mu} - \boldsymbol{x})^{\top}\boldsymbol{Q}(\boldsymbol{\mu} - \boldsymbol{x}) \; + \; \boldsymbol{u}^{\top}\boldsymbol{R}\boldsymbol{u} \\ &= (\boldsymbol{\mu} - \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1 - \boldsymbol{S}^{\boldsymbol{u}}\boldsymbol{u})^{\top}\boldsymbol{Q}\,(\boldsymbol{\mu} - \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1 - \boldsymbol{S}^{\boldsymbol{u}}\boldsymbol{u}) + \boldsymbol{u}^{\top}\boldsymbol{R}\boldsymbol{u} \end{aligned}$$

Solving for *u* results in the analytic solution:

$$\hat{\boldsymbol{u}} = \left(\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\boldsymbol{S}^{\boldsymbol{u}} + \boldsymbol{R}\right)^{-1}\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\,\left(\boldsymbol{\mu} - \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1\right)$$



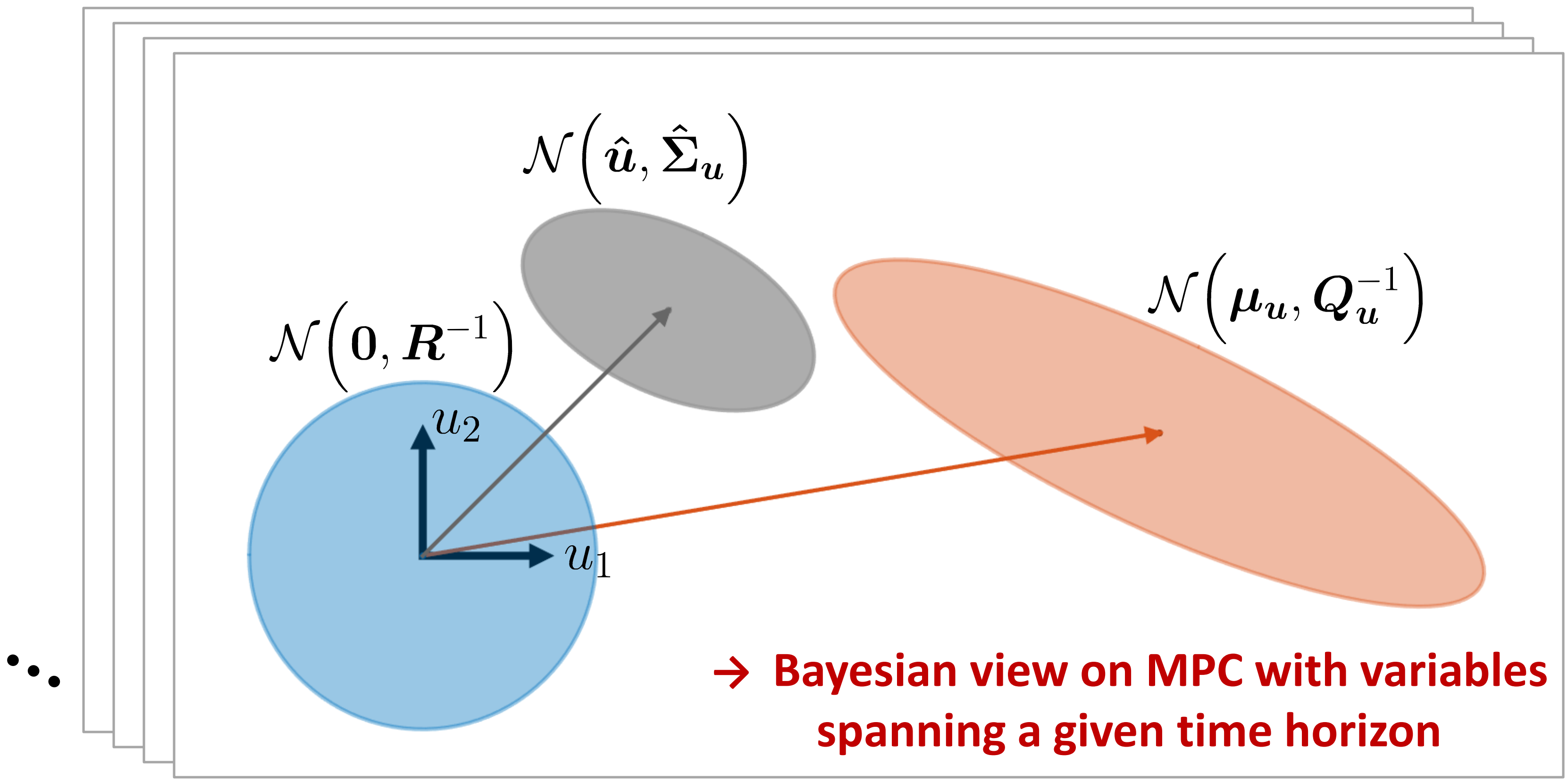
MPC/LQT as a product of Gaussians

Track path! Use low control commands!

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \left\| \boldsymbol{\mu}_t - \boldsymbol{x}_t \right\|^2_{\boldsymbol{Q}_t} + \left\| \boldsymbol{u}_t \right\|^2_{\boldsymbol{R}_t}$$

$$\mathcal{N}\Big(\hat{\boldsymbol{u}}, \hat{\boldsymbol{\Sigma}}^{\boldsymbol{u}}\Big) \propto \mathcal{N}\Big(\boldsymbol{\mu}_{\boldsymbol{u}}, \boldsymbol{Q}_{\boldsymbol{u}}^{-1}\Big)\, \mathcal{N}\Big(\boldsymbol{0}, \boldsymbol{R}^{-1}\Big)$$

$$\boldsymbol{\mu}_{\boldsymbol{u}} \triangleq \boldsymbol{S}^{\boldsymbol{u}\dagger}(\boldsymbol{\mu} - \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1)$$
$$\boldsymbol{Q}_{\boldsymbol{u}} \triangleq \boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\boldsymbol{S}^{\boldsymbol{u}}$$



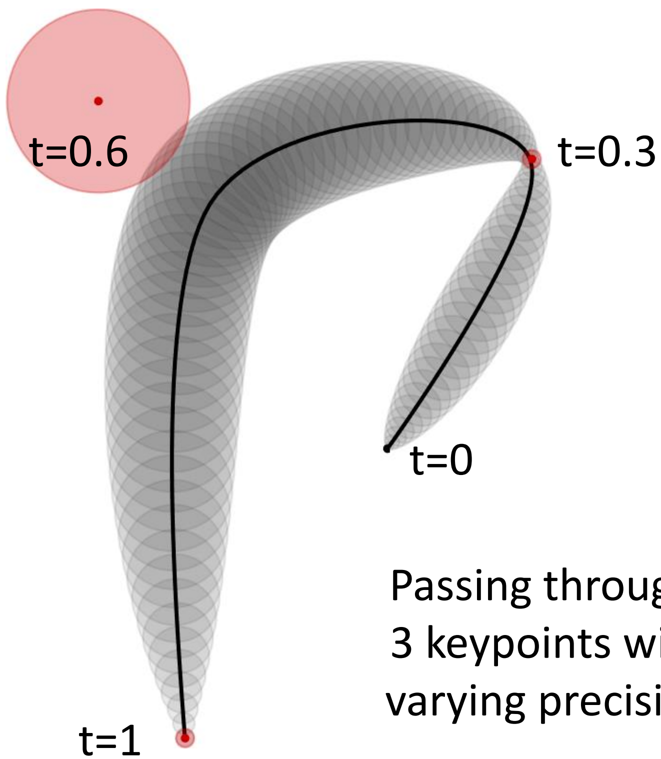
→ Bayesian view on MPC with variables spanning a given time horizon

Probabilistic representation of MPC/LQT

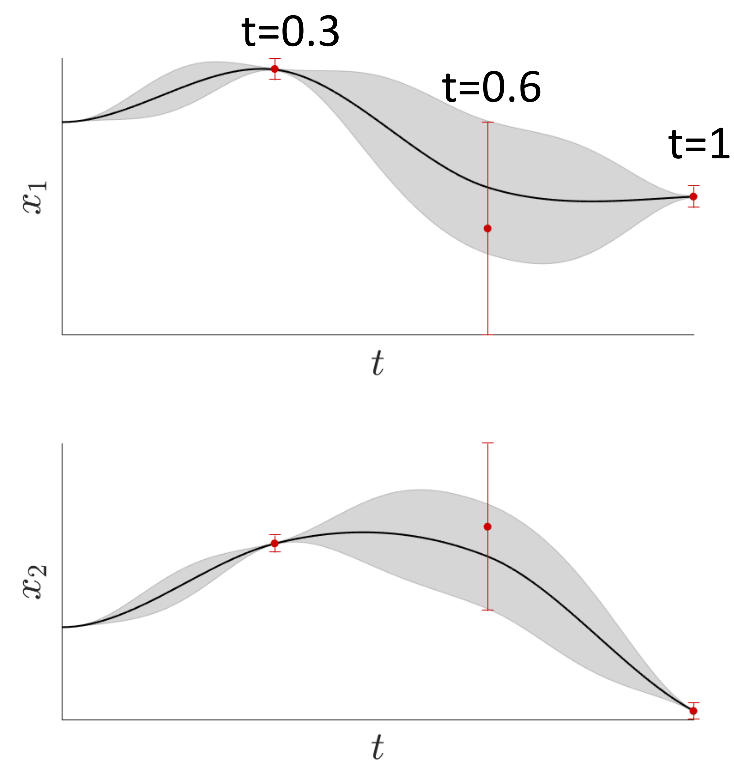
$$\hat{\boldsymbol{u}} = \left(\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\boldsymbol{S}^{\boldsymbol{u}} + \boldsymbol{R}\right)^{-1}\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\left(\boldsymbol{\mu} - \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1\right)$$
$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{u}} = \left(\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\boldsymbol{S}^{\boldsymbol{u}} + \boldsymbol{R}\right)^{-1}$$

$$\hat{\boldsymbol{x}} = \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1 + \boldsymbol{S}^{\boldsymbol{u}}\hat{\boldsymbol{u}}$$
$$\hat{\boldsymbol{\Sigma}}^{\boldsymbol{x}} = \boldsymbol{S}^{\boldsymbol{u}}\left(\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\boldsymbol{S}^{\boldsymbol{u}} + \boldsymbol{R}\right)^{-1}\boldsymbol{S}^{\boldsymbol{u}\top}$$

The distribution in control space can be projected back to the state space



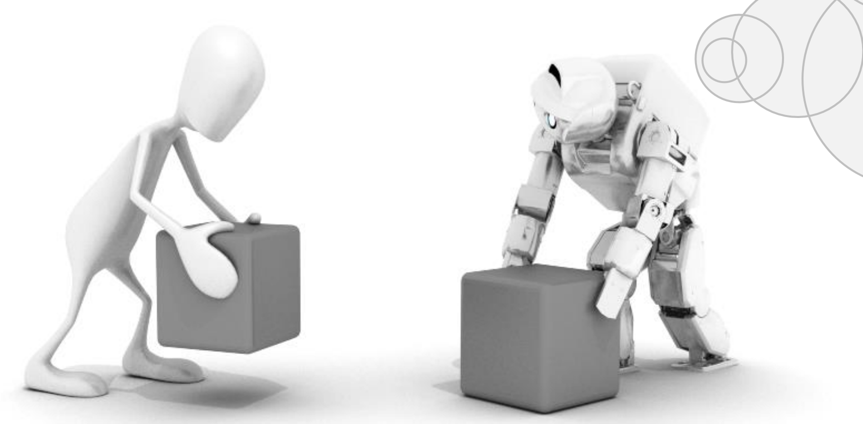
Passing through 3 keypoints with varying precision



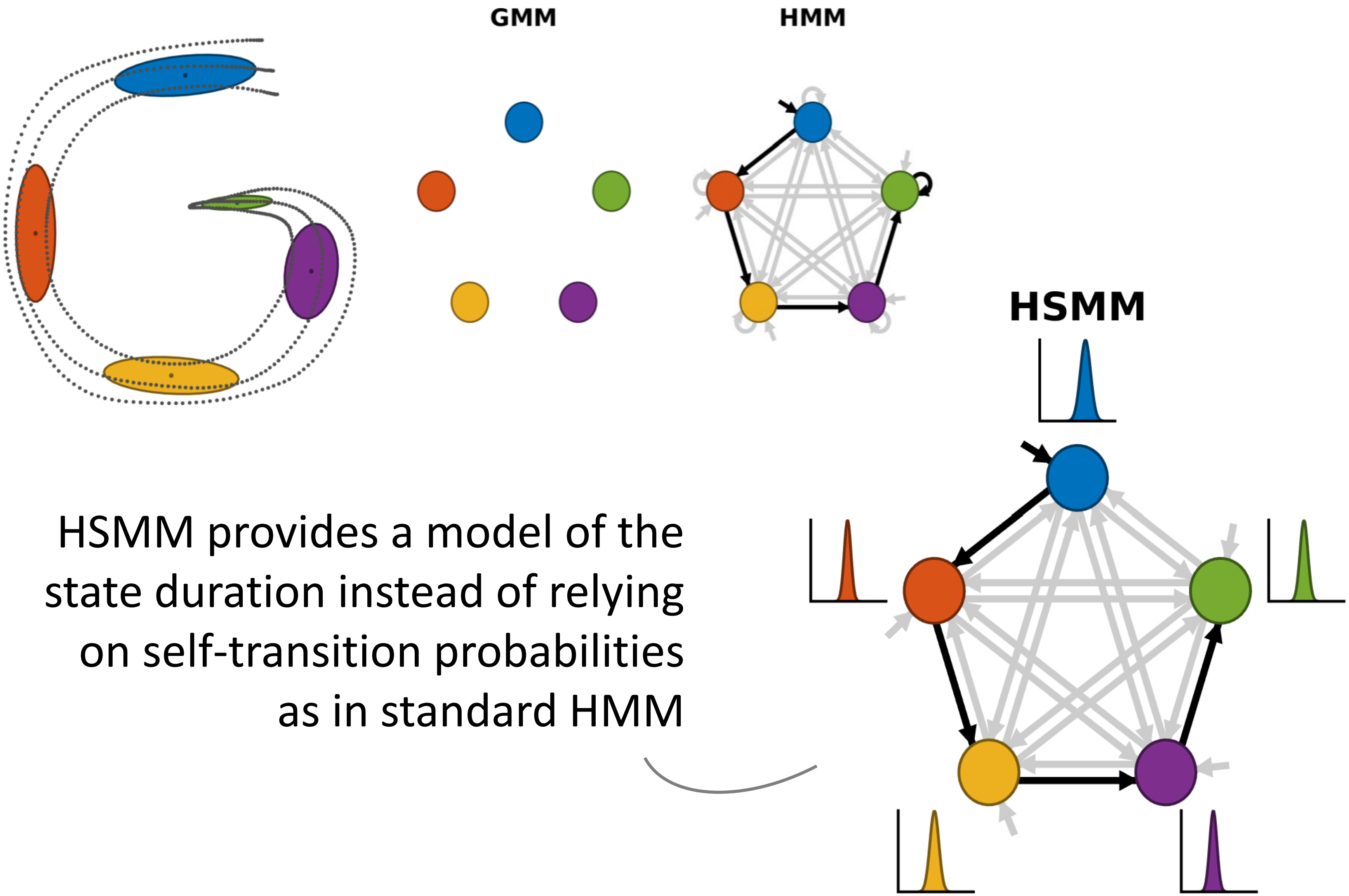
Model Predictive Control (MPC) combined with probabilistic representation of movement primitives

$$c = (\boldsymbol{\mu} - \boldsymbol{x})^{\top} \boldsymbol{Q} (\boldsymbol{\mu} - \boldsymbol{x}) \; + \; \boldsymbol{u}^{\top} \boldsymbol{R} \boldsymbol{u}$$

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_T \end{bmatrix} \qquad \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{Q}_T \end{bmatrix}$$



Hidden semi-Markov model (HSMM)



HSMM provides a model of the state duration instead of relying on self-transition probabilities as in standard HMM

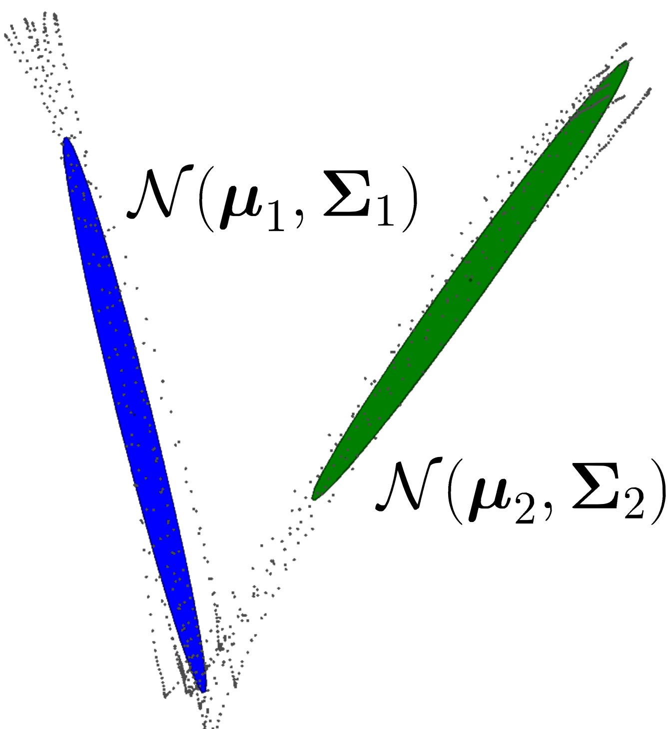
Hidden semi-Markov model (HSMM)

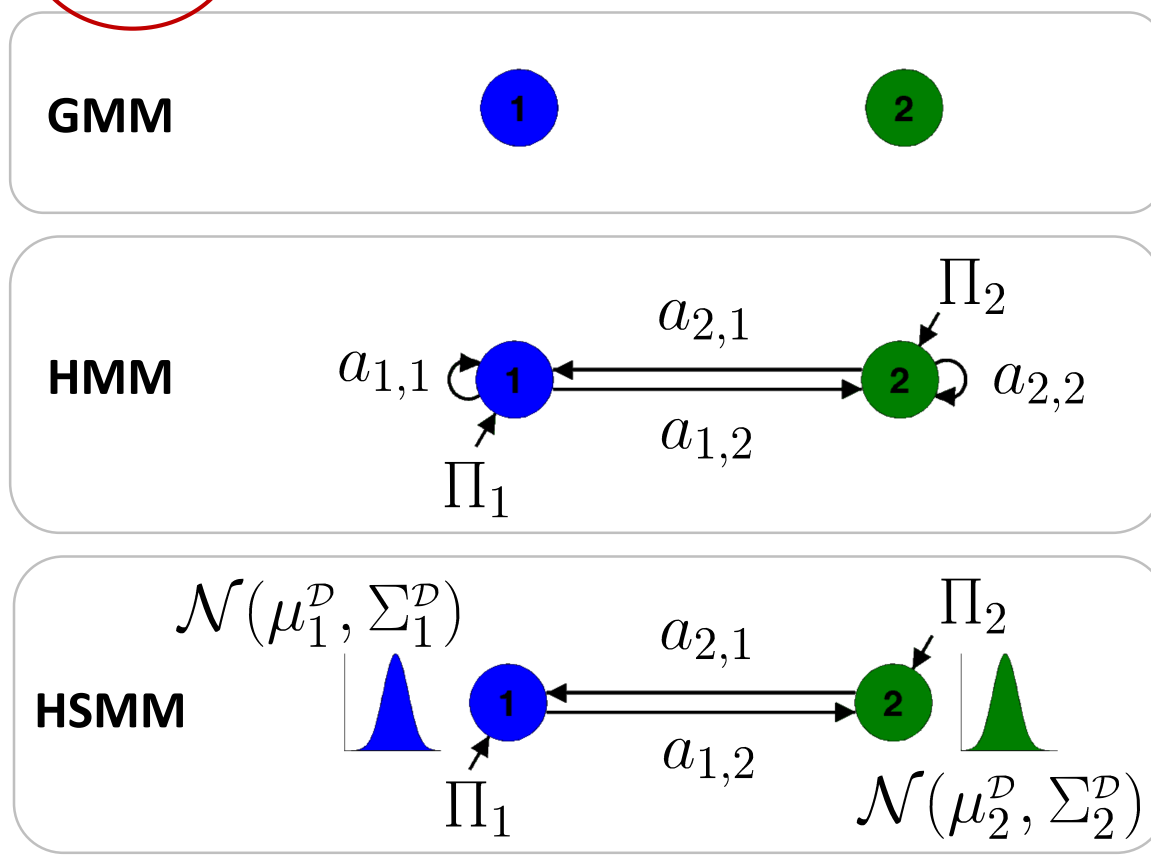
$$\boldsymbol{\Theta}^{\text{GMM}} = \{\pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$$

$$\boldsymbol{\Theta}^{\text{HMM}} = \{\{a_{i,j}\}_{j=1}^{K}, \Pi_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$$

Parametric duration distribution

$$\boldsymbol{\Theta}^{\text{HSMM}} = \{\{a_{i,j}\}_{j=1, j\neq i}^{K}, \Pi_i, \mu_i^{\mathcal{D}}, \Sigma_i^{\mathcal{D}}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{K}$$

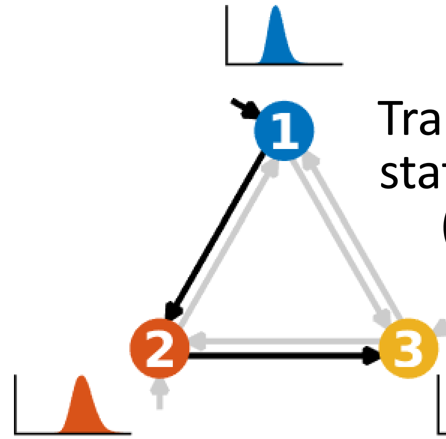




Learning minimal intervention controllers

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\|_{\boldsymbol{Q}_t}^2 + \|\boldsymbol{u}_t\|_{\boldsymbol{R}_t}^2$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}_t = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t$$

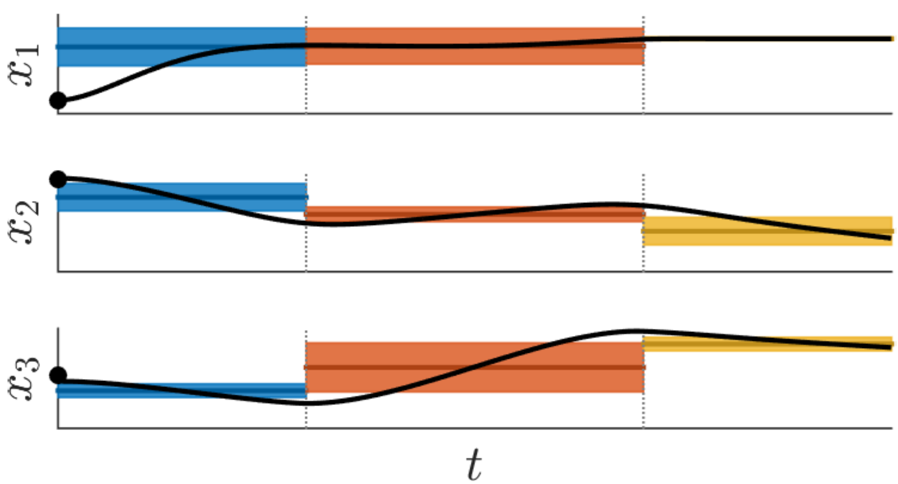


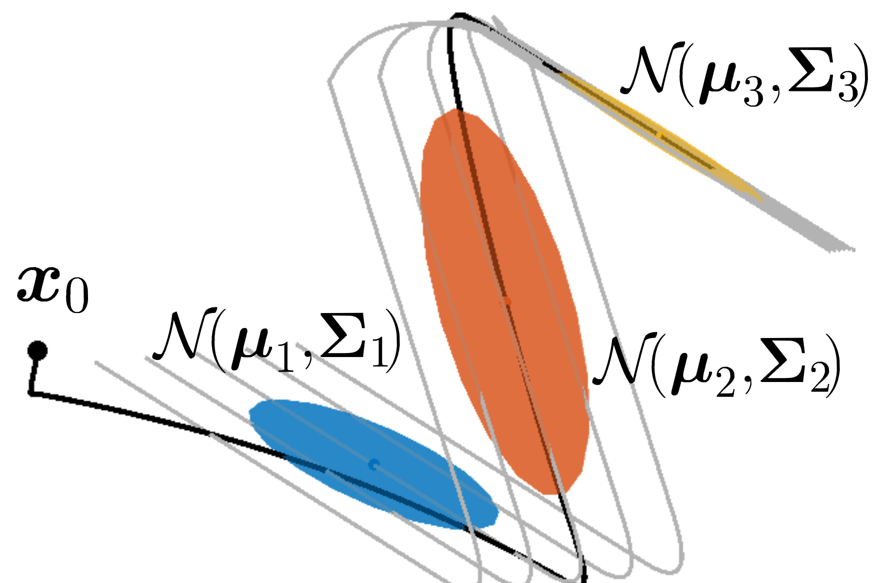
Transition and state duration (HSMM)

Stepwise reference path given by:

$$\hat{\boldsymbol{x}}_t = \boldsymbol{\mu}_{s_t} \quad \boldsymbol{Q}_t = \boldsymbol{\Sigma}_{s_t}^{-1}$$

s_t **11111111222222222222333333333**



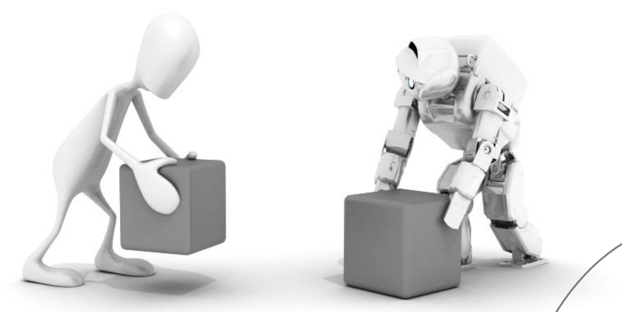


$\boldsymbol{\mu}_i$ center of the Gaussian

$\boldsymbol{\Sigma}_i$ covariance matrix

Learning minimal intervention controllers

$$\boldsymbol{s} = \{1, 1, 1, 2, \ldots, 4\}$$



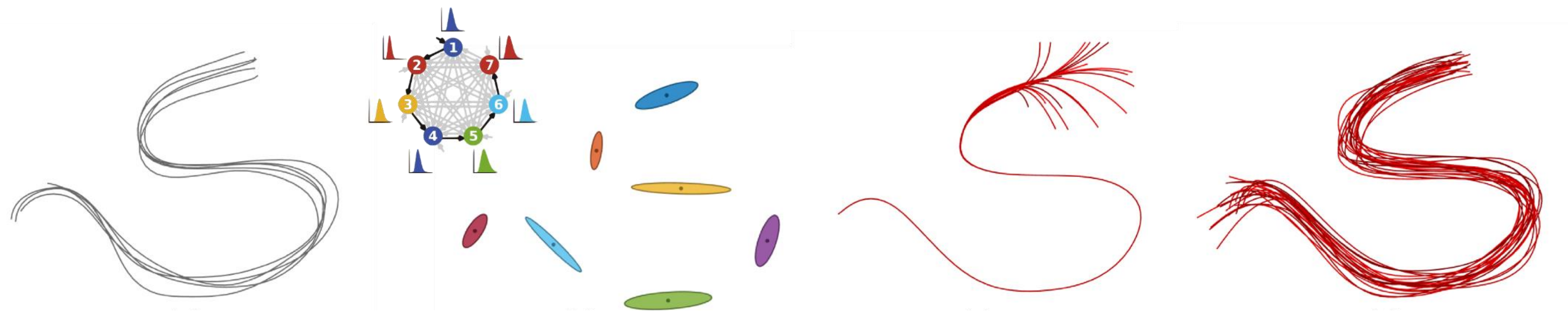
$$\begin{bmatrix} \boldsymbol{\Sigma}_1^{-1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_1^{-1} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\Sigma}_1^{-1} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\Sigma}_2^{-1} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{\Sigma}_K^{-1} \end{bmatrix} \qquad \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \\ \vdots \\ \boldsymbol{\mu}_K \end{bmatrix}$$

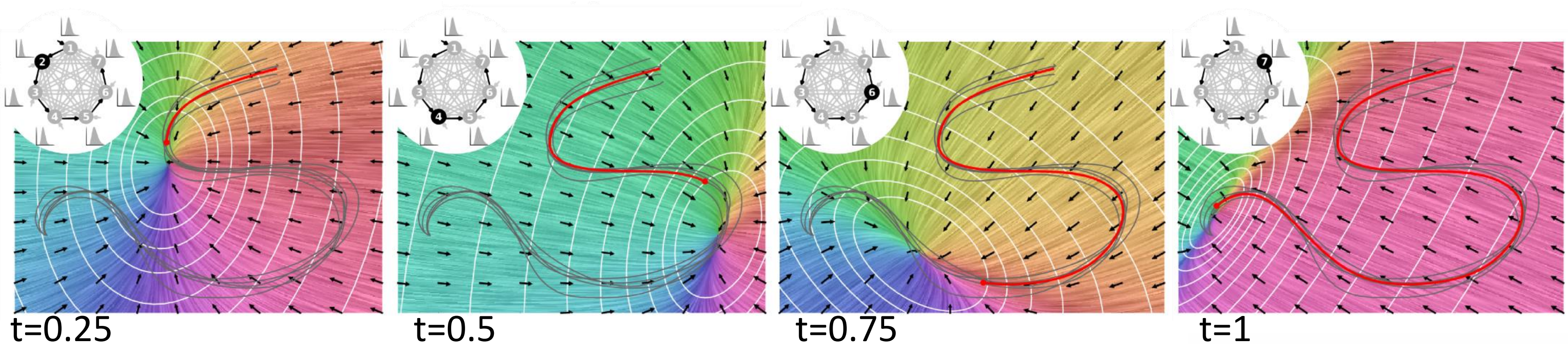
$$\hat{\boldsymbol{u}} = \left(\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\boldsymbol{S}^{\boldsymbol{u}} + \boldsymbol{R}\right)^{-1}\boldsymbol{S}^{\boldsymbol{u}\top}\boldsymbol{Q}\left(\boldsymbol{\mu} - \boldsymbol{S}^{\boldsymbol{x}}\boldsymbol{x}_1\right)$$

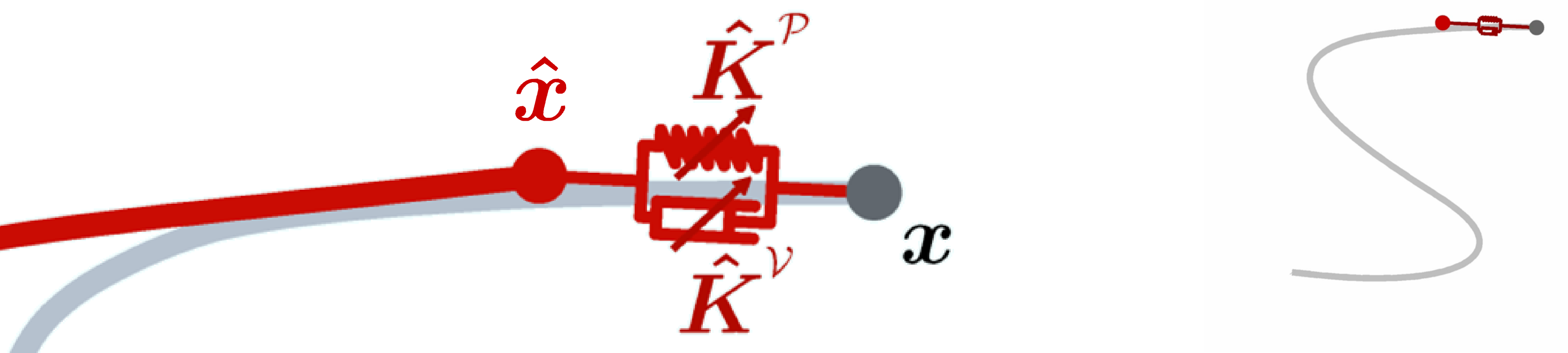
$$\begin{bmatrix} \hat{\boldsymbol{u}}_1 \\ \hat{\boldsymbol{u}}_2 \\ \hat{\boldsymbol{u}}_3 \\ \vdots \\ \hat{\boldsymbol{u}}_{T-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{T-2}\boldsymbol{B} & \boldsymbol{A}^{T-3}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \qquad \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^{T-1} \end{bmatrix}$$

Learning controllers instead of trajectories







Outline

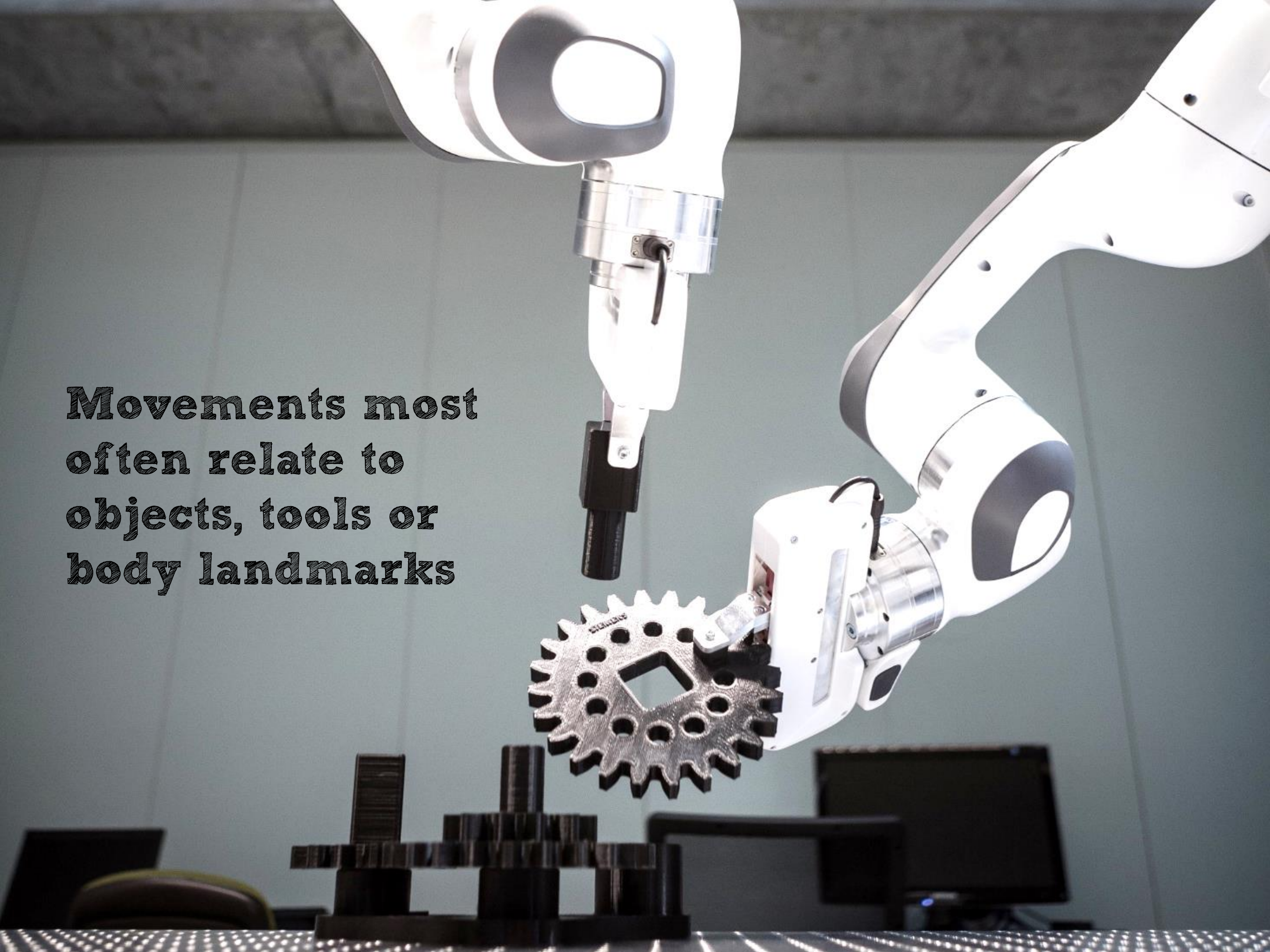
- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

- **Superposition Vs fusion**
  Product of Gaussians

- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

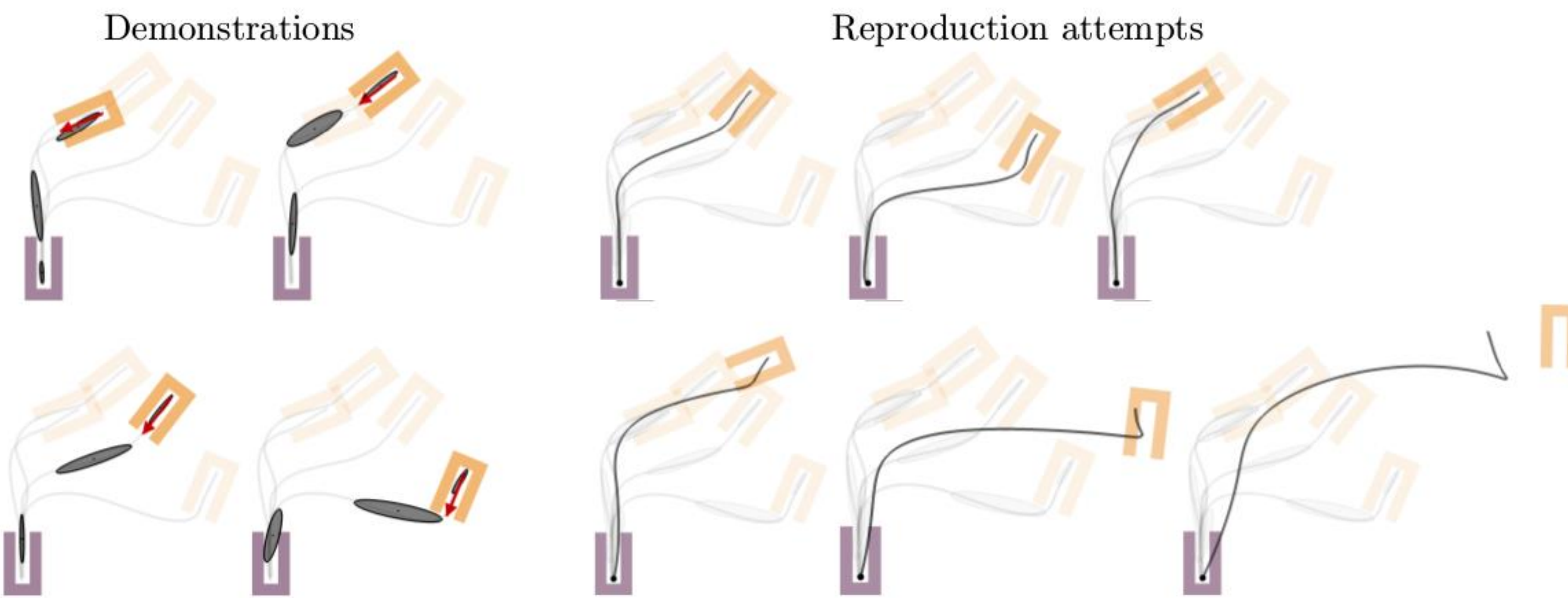
- **Differential geometry**
  Riemannian manifolds


Movements most often relate to objects, tools or body landmarks

Conditioning-based approach

Regression with a context variable *c*:

→ Learning of $\mathcal{P}(\boldsymbol{x}|\boldsymbol{c})$



→ Generic approach, but limited generalization capability

MPC/LQT in multiple coordinate systems

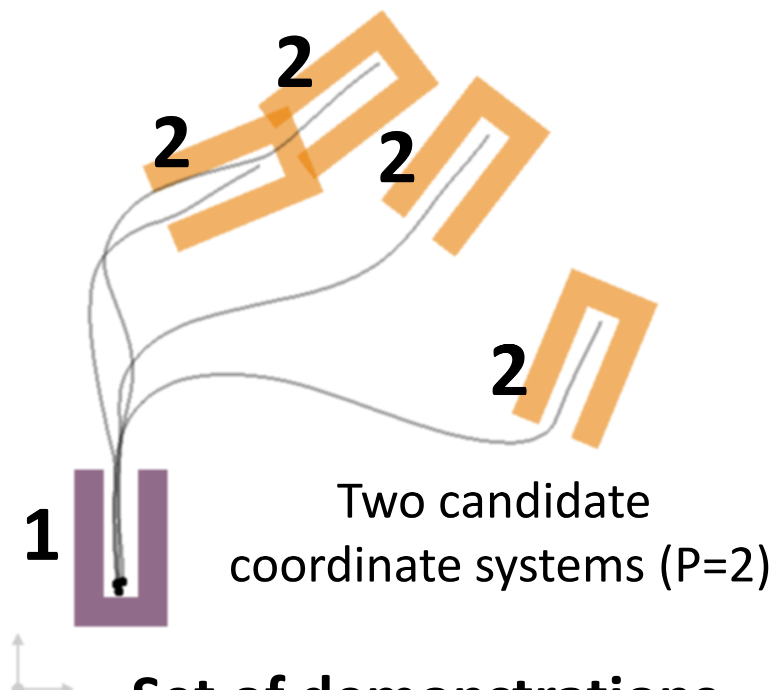
Track path in coordinate system j

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \sum_{j=1}^{P} \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|_{\boldsymbol{Q}_t^{(j)}}^2 + \left\| \boldsymbol{u}_t \right\|_{\boldsymbol{R}_t}^2$$

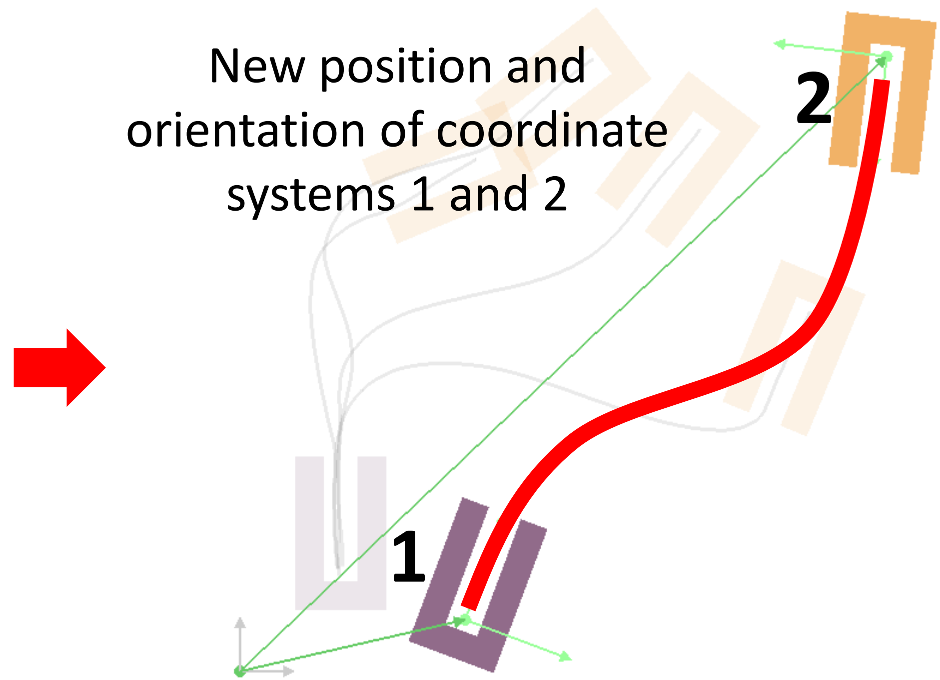
Use low control commands

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t$$

System dynamics



Set of demonstrations



Reproduction in new situation

MPC/LQT in multiple coordinate systems

Track path in coordinate system j

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \sum_{j=1}^{P} \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|^2_{\boldsymbol{Q}_t^{(j)}} + \left\| \boldsymbol{u}_t \right\|^2_{\boldsymbol{R}_t}$$

Use low control commands

$$\text{s.t.} \quad \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t$$

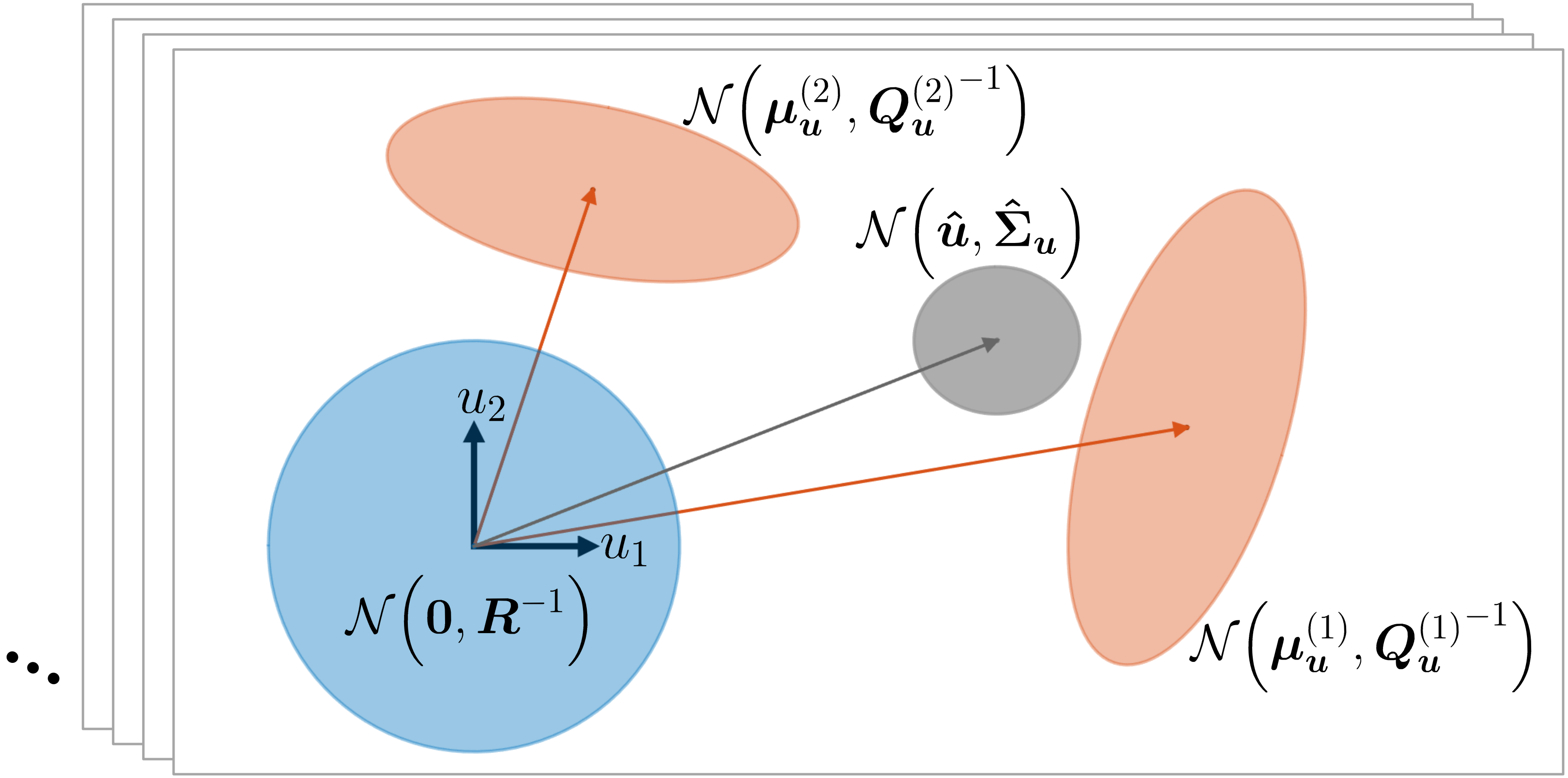
System dynamics

MPC/LQT in multiple coordinate systems

Track path in coordinate system j

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \sum_{j=1}^{P} \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|^2_{\boldsymbol{Q}_t^{(j)}} + \left\| \boldsymbol{u}_t \right\|^2_{\boldsymbol{R}_t}$$

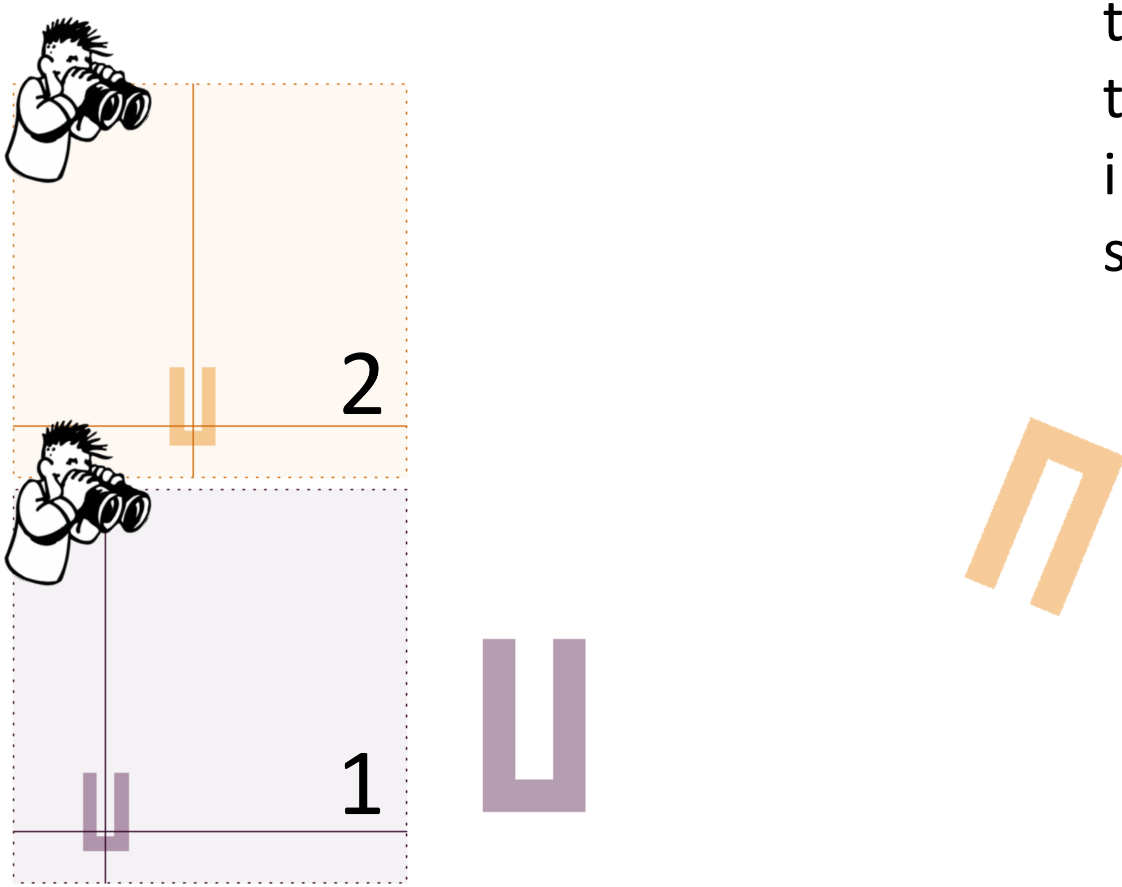
Use low control commands



MPC/LQT in multiple coordinate systems

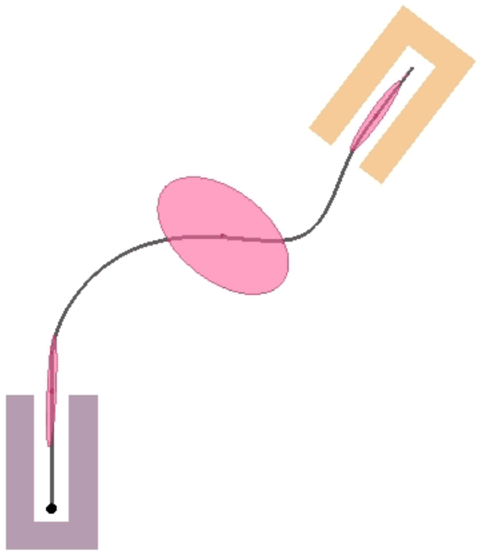
$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \sum_{j=1}^{P} \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|^2_{\boldsymbol{Q}_t^{(j)}} + \left\| \boldsymbol{u}_t \right\|^2_{\boldsymbol{R}_t}$$

In many robotics problems, the parameters describing the task or situation can be interpreted as coordinate systems

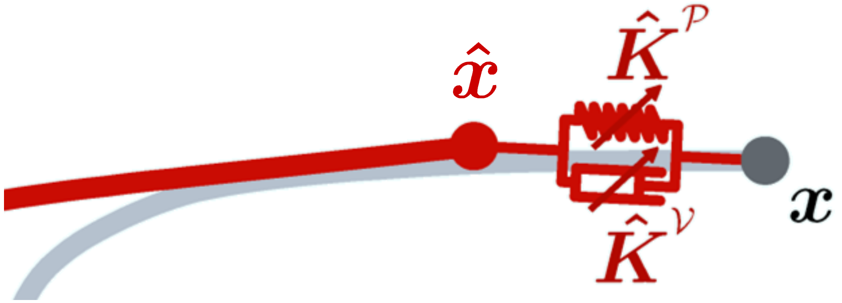


MPC/LQT in multiple coordinate systems

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \sum_{j=1}^{P} \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|^2_{\boldsymbol{Q}_t^{(j)}} + \left\| \boldsymbol{u}_t \right\|^2_{\boldsymbol{R}_t}$$



➡ **Learning of a controller**
(instead of learning a trajectory)
that adapts to new situations
while regulating the gains
according to the precision and
coordination required by the task

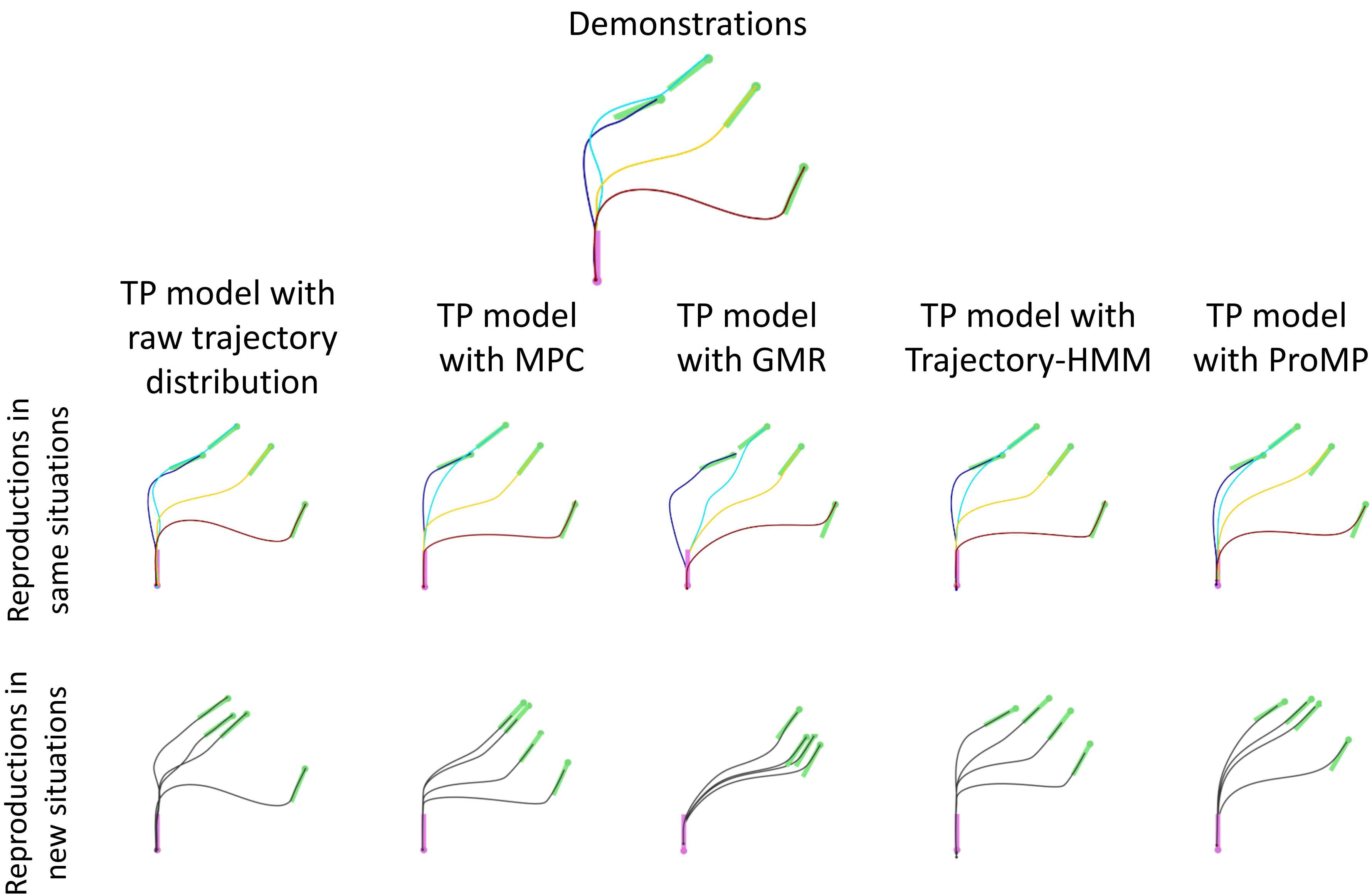


MPC/LQT in multiple coordinate systems

$$\min_{\boldsymbol{u}} \sum_{t=1}^{T} \sum_{j=1}^{P} \left\| \boldsymbol{\mu}_t^{(j)} - \boldsymbol{x}_t \right\|^2_{\boldsymbol{Q}_t^{(j)}} + \left\| \boldsymbol{u}_t \right\|^2_{\boldsymbol{R}_t}$$

➡ Retrieval of control commands in the form of trajectory distributions, facilitating exploration and adaptation (in either control or state space)

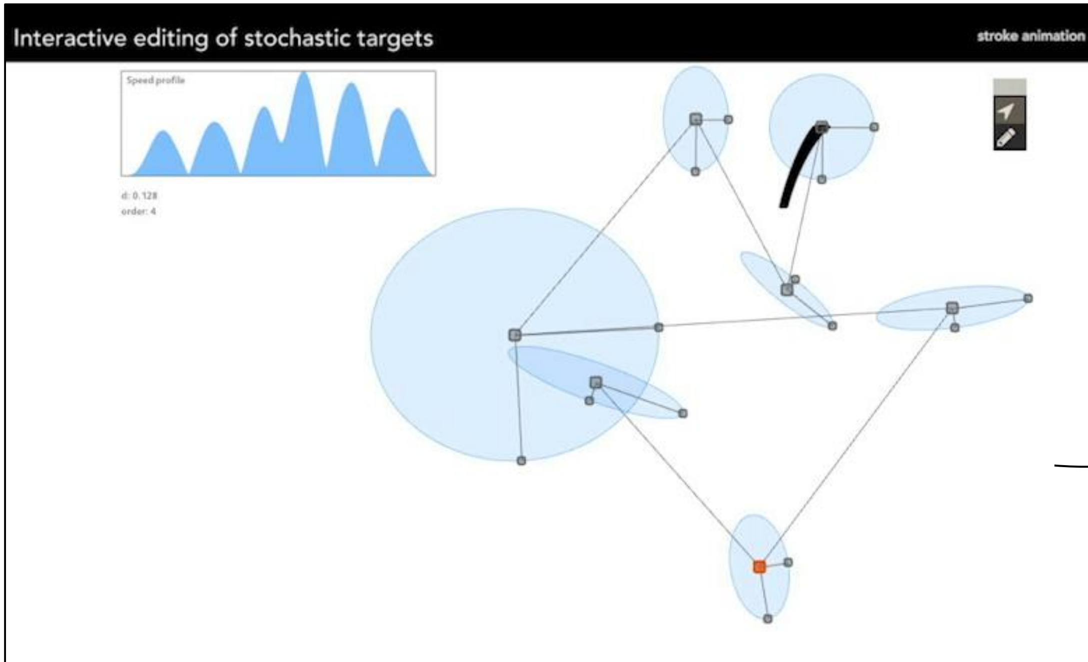
Exploitation in other probabilistic models



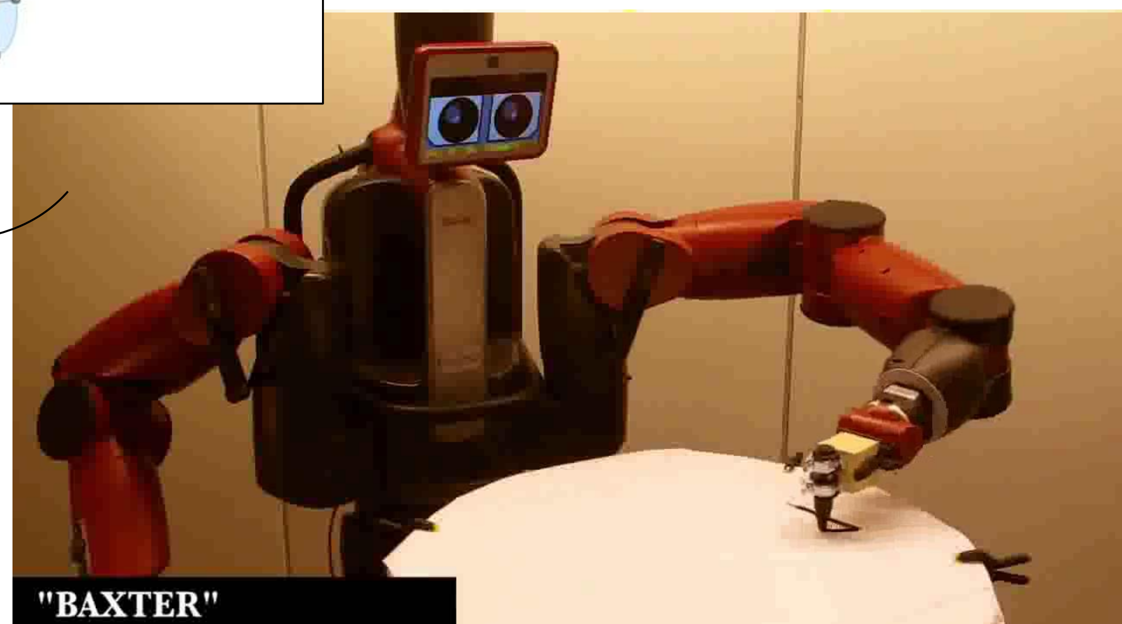
http://www.idiap.ch/software/pbdlib/

Application examples

Application: Editing movements with variations



User interface to edit and generate natural and dynamic motions by considering variation and coordination



Compliant controller to retrieve safe and human-like motions

Daniel Berio

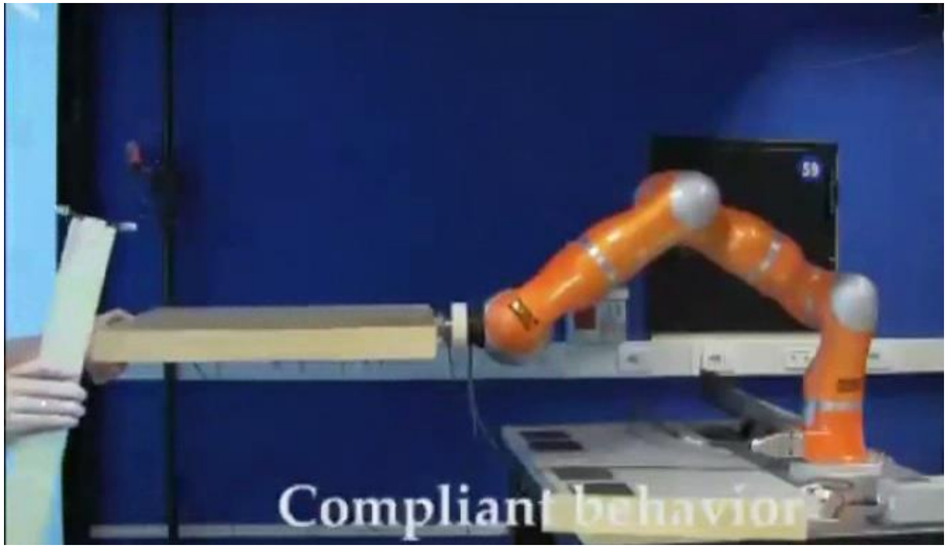
Frederic Fol Leymarie

[Berio, Calinon and Leymarie, IROS'2016] [Berio, Calinon and Leymarie , MOCO'2017]

Coordination and co-manipulation

[Silvério et al., IROS'2015]

[Rozo et al., IROS'2015]



[Rozo et al., IEEE T-RO 32(3), 2016]

ISTITUTO
ITALIANO DI
TECNOLOGIA

Assistive dressing

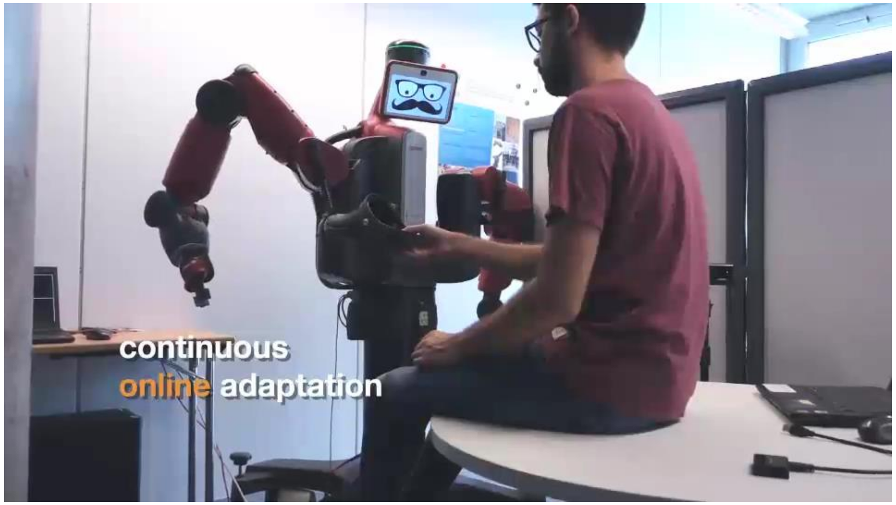
I-DRESS
Assistive interactive robotic system for support in dressing

SNSF, CHIST-ERA (2015-2018)
https://i-dress-project.eu

Institut de Robòtica i Informàtica Industrial

UWE Bristol University of the West of England

idiap RESEARCH INSTITUTE



[Pignat and Calinon, RAS 93, 2017] [Canal, Pignat, Alenya, Calinon and Torras, ICRA’2018]

Shared control

DexROV

spaceapplications SERVICES

GRAALtech

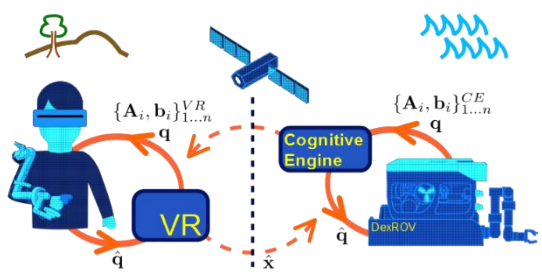
JACOBS UNIVERSITY

comex

ISME Integrated Systems for Marine Environment

EJR-QUARTZ

idiap RESEARCH INSTITUTE

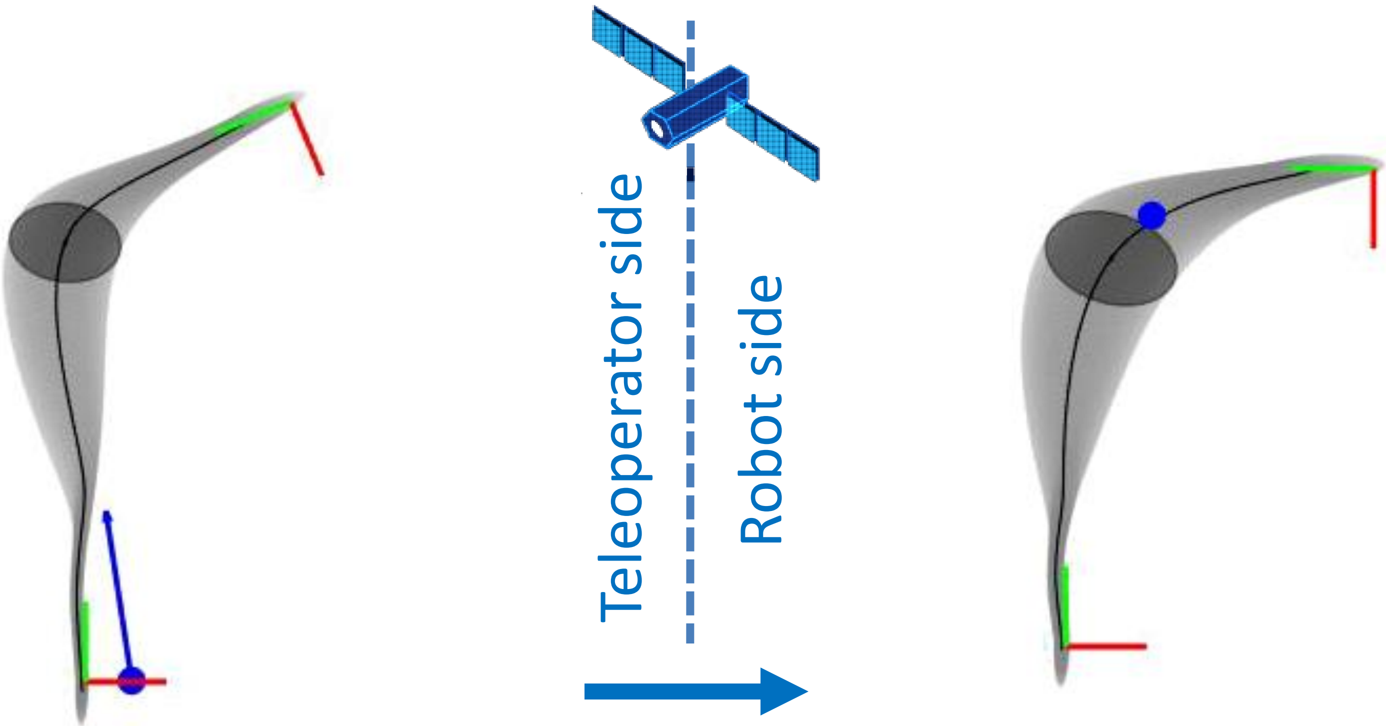


http://dexrov.eu

EC, H2020 (2015-2018)

Shared control

http://dexrov.eu
EC, H2020 (2015-2018)

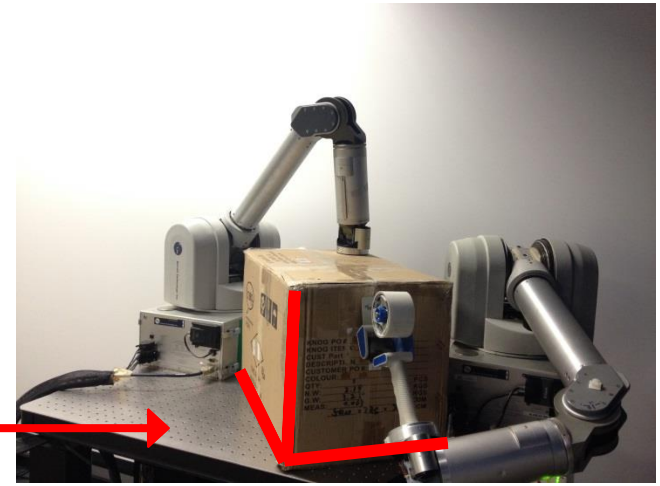


only Gaussian ID
is transmitted

[Birk *et al.*, IEEE Robotics and Autom. Magazine, 2018 (in press)]   [Havoutis & Calinon, Autonomous Robots, 2018]

Adaptation to different object shapes

Coordinate system as task parameter



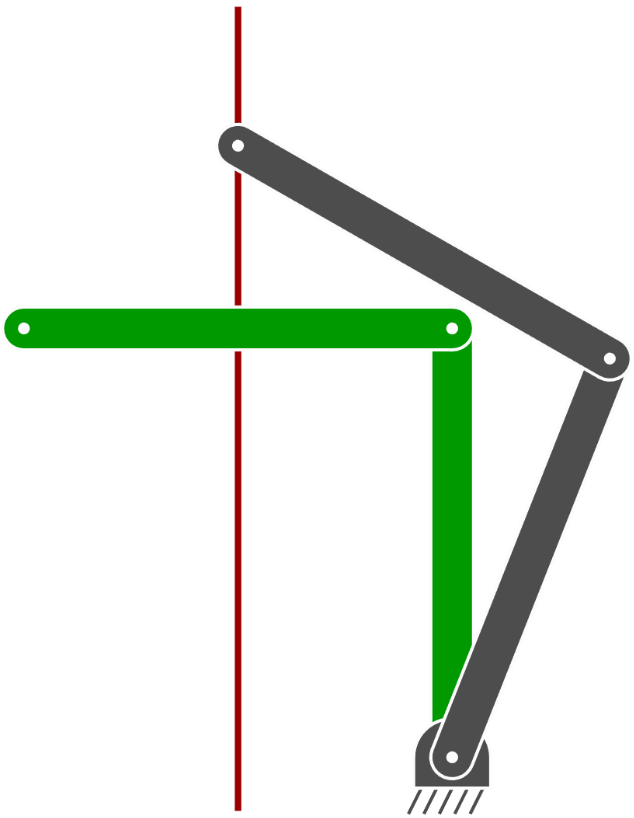
[Calinon, Alizadeh and Caldwell, IROS'2013]

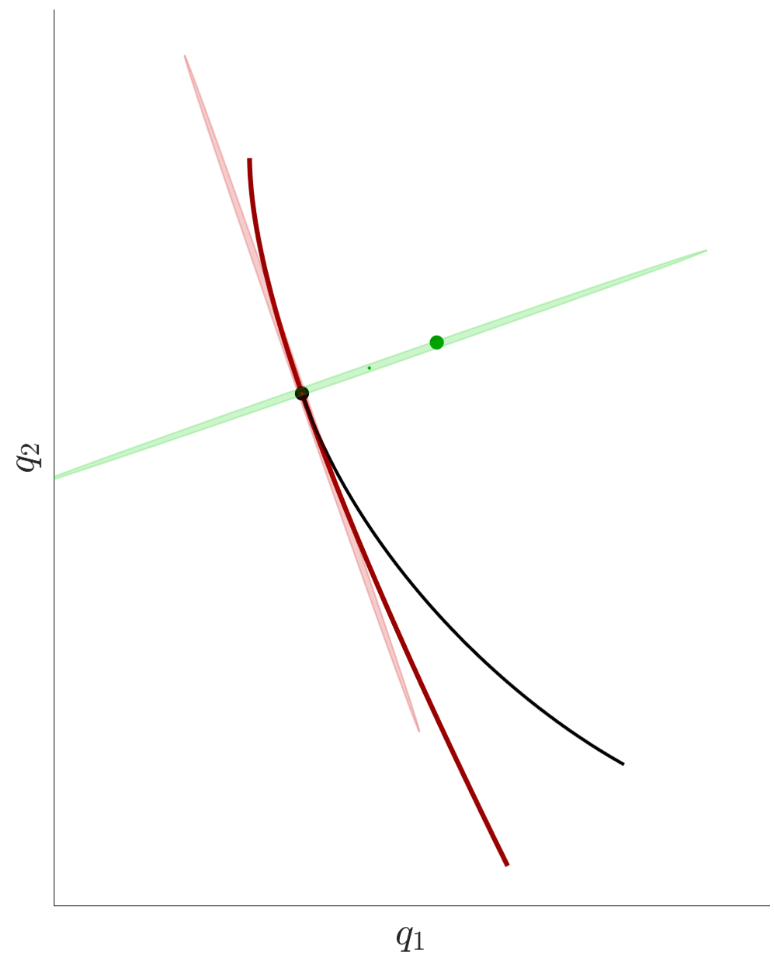
Learning tasks prioritization

$$\hat{\boldsymbol{u}} = \arg\min_{\boldsymbol{u}} \left\|\boldsymbol{u} - \boldsymbol{J}^{\dagger}\dot{\boldsymbol{x}}\right\|^2_{\boldsymbol{J}^{\dagger}\boldsymbol{J}} + \left\|\boldsymbol{u} - \dot{\boldsymbol{q}}\right\|^2_{\boldsymbol{I}-\boldsymbol{J}^{\dagger}\boldsymbol{J}}$$
$$= \boldsymbol{J}^{\dagger}\dot{\boldsymbol{x}} + (\boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J})\dot{\boldsymbol{q}}$$

Principal task:
track horizontal reference

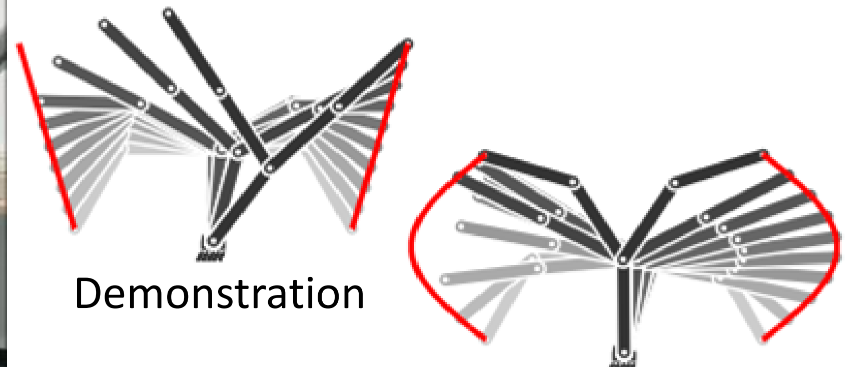
Secondary task:
track desired posture





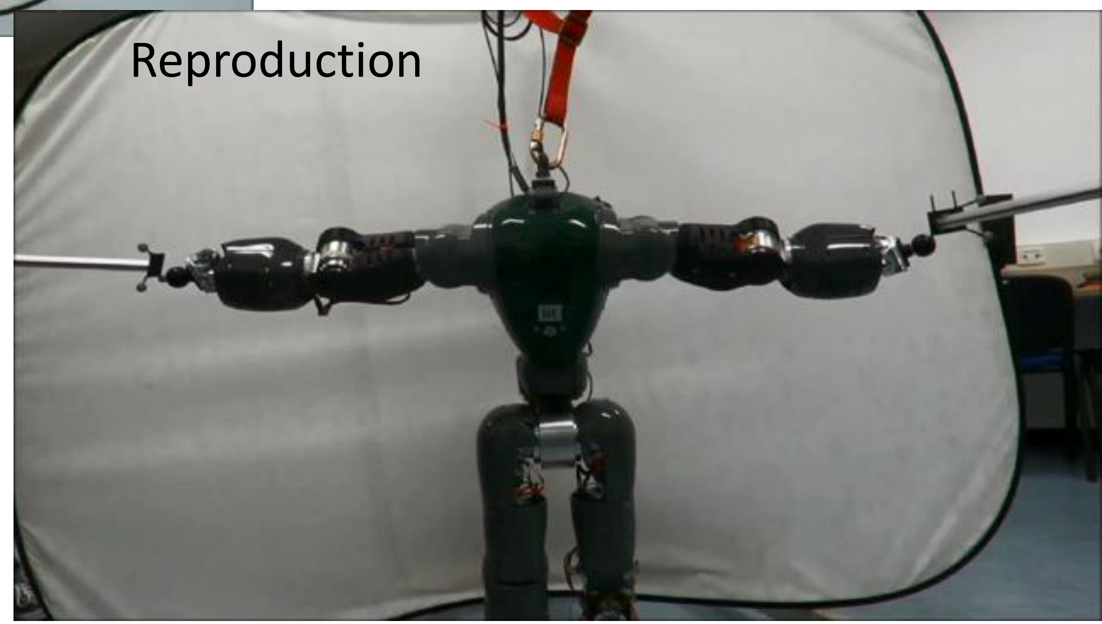
Learning tasks prioritization

Priority on left hand



Demonstration

Reproduction



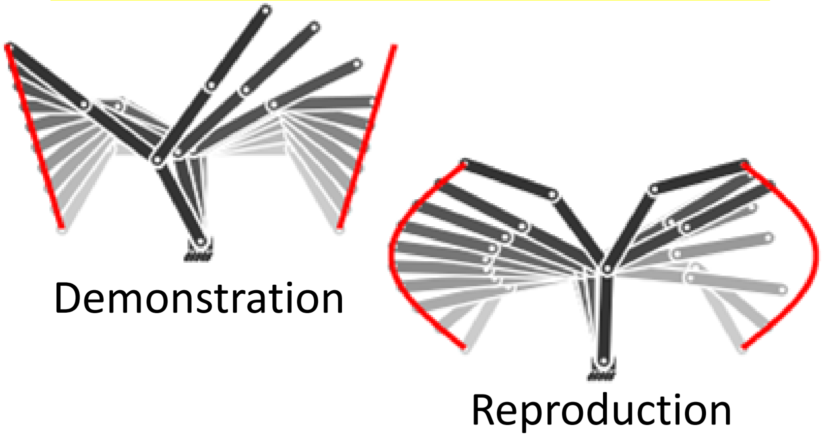
$$\hat{\dot{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} \boldsymbol{J}_1^\dagger & \boldsymbol{N}_1 \boldsymbol{J}_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } \boldsymbol{A}_1} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}$$

$$\hat{\dot{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} \boldsymbol{N}_2 \boldsymbol{J}_1^\dagger & \boldsymbol{J}_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } \boldsymbol{A}_2} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}$$

[Silvério, Calinon, Rozo and Caldwell, IEEE T-RO 2019] [Calinon, ISRR’15]

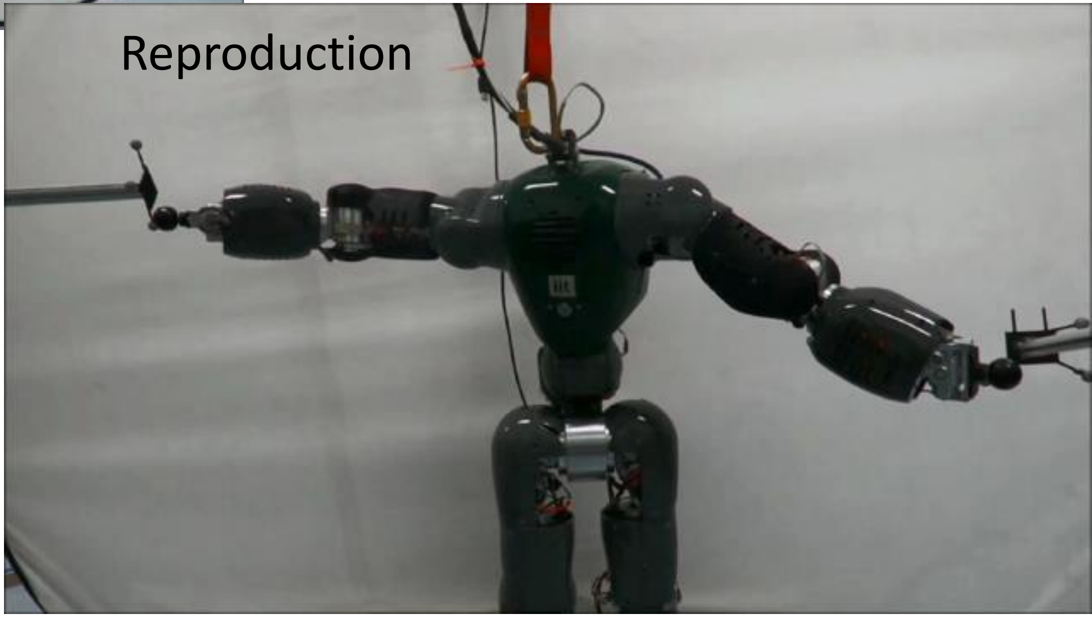
Learning tasks prioritization

Priority on right hand



$$\dot{\hat{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} \boldsymbol{J}_1^\dagger & \boldsymbol{N}_1 \boldsymbol{J}_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } \boldsymbol{A}_1} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}$$

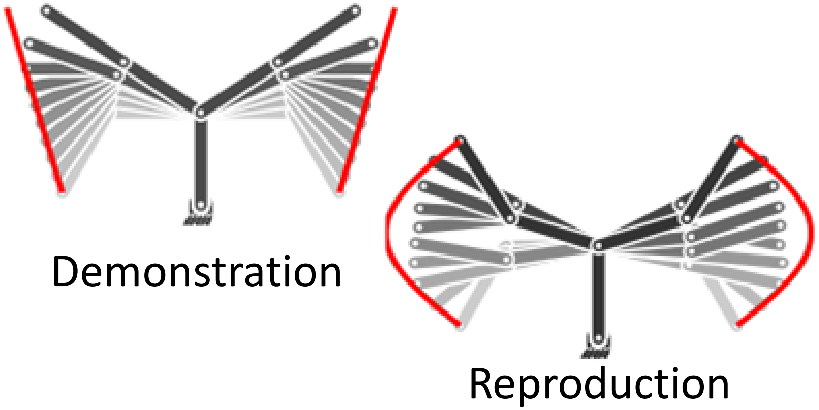
$$\dot{\hat{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} \boldsymbol{N}_2 \boldsymbol{J}_1^\dagger & \boldsymbol{J}_2^\dagger \end{bmatrix}}_{\text{Candidate hierarchy } \boldsymbol{A}_2} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}$$



[Silvério, Calinon, Rozo and Caldwell, IEEE T-RO 2019] [Calinon, ISRR'15]

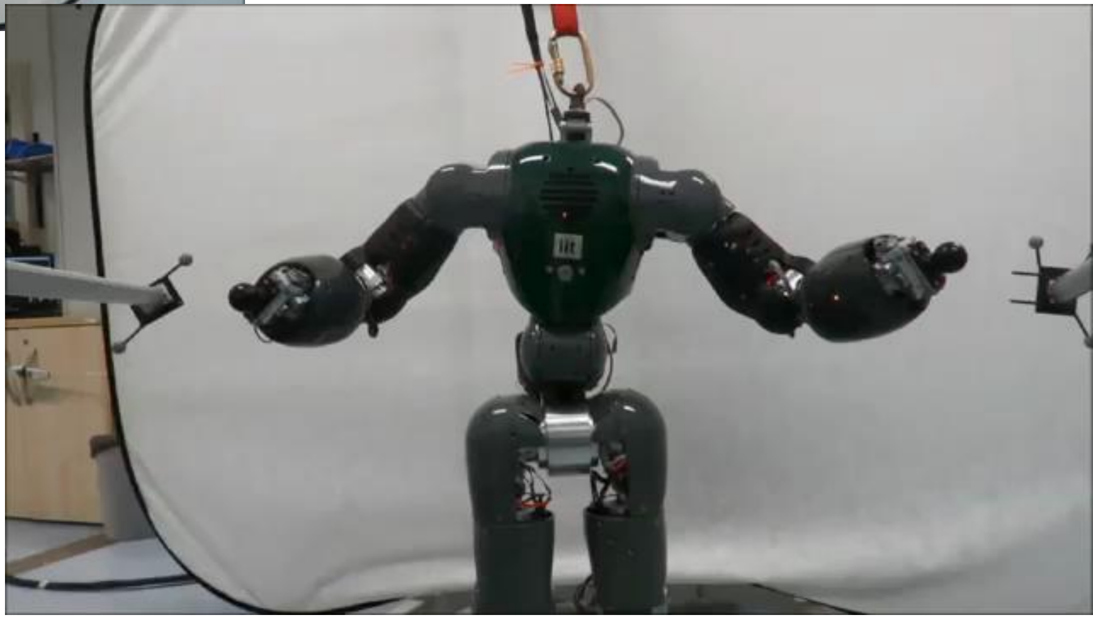
Learning tasks prioritization

Equal priority



$$\hat{\dot{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} \boldsymbol{J}_1^{\dagger} & \boldsymbol{N}_1 \boldsymbol{J}_2^{\dagger} \end{bmatrix}}_{\text{Candidate hierarchy } \boldsymbol{A}_1} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}$$

$$\hat{\dot{\boldsymbol{q}}} = \underbrace{\begin{bmatrix} \boldsymbol{N}_2 \boldsymbol{J}_1^{\dagger} & \boldsymbol{J}_2^{\dagger} \end{bmatrix}}_{\text{Candidate hierarchy } \boldsymbol{A}_2} \begin{bmatrix} \dot{\boldsymbol{x}}_1 \\ \dot{\boldsymbol{x}}_2 \end{bmatrix}$$



[Silvério, Calinon, Rozo and Caldwell, IEEE T-RO 2019] [Calinon, ISRR'15]

Outline

- **Superposition with basis functions**
  Bezier curves
  Locally weighted regression (LWR)
  Gaussian mixture regression (GMR)
  Fourier series for periodic motion and ergodic control

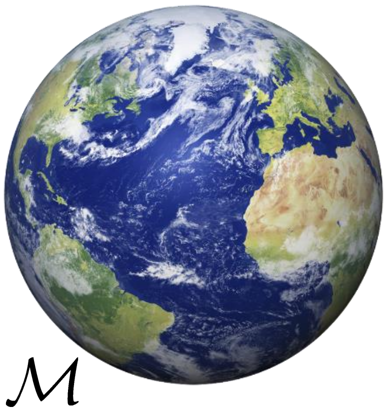
- **Dynamical movement primitives (DMP)**
  Probabilistic movement primitives (ProMP)

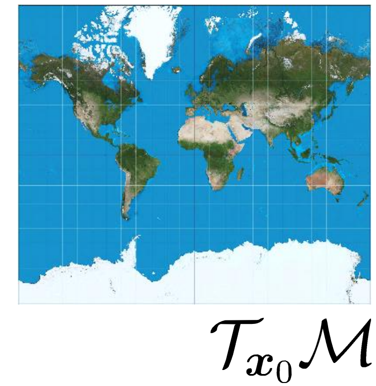
- **Superposition Vs fusion**
  Product of Gaussians

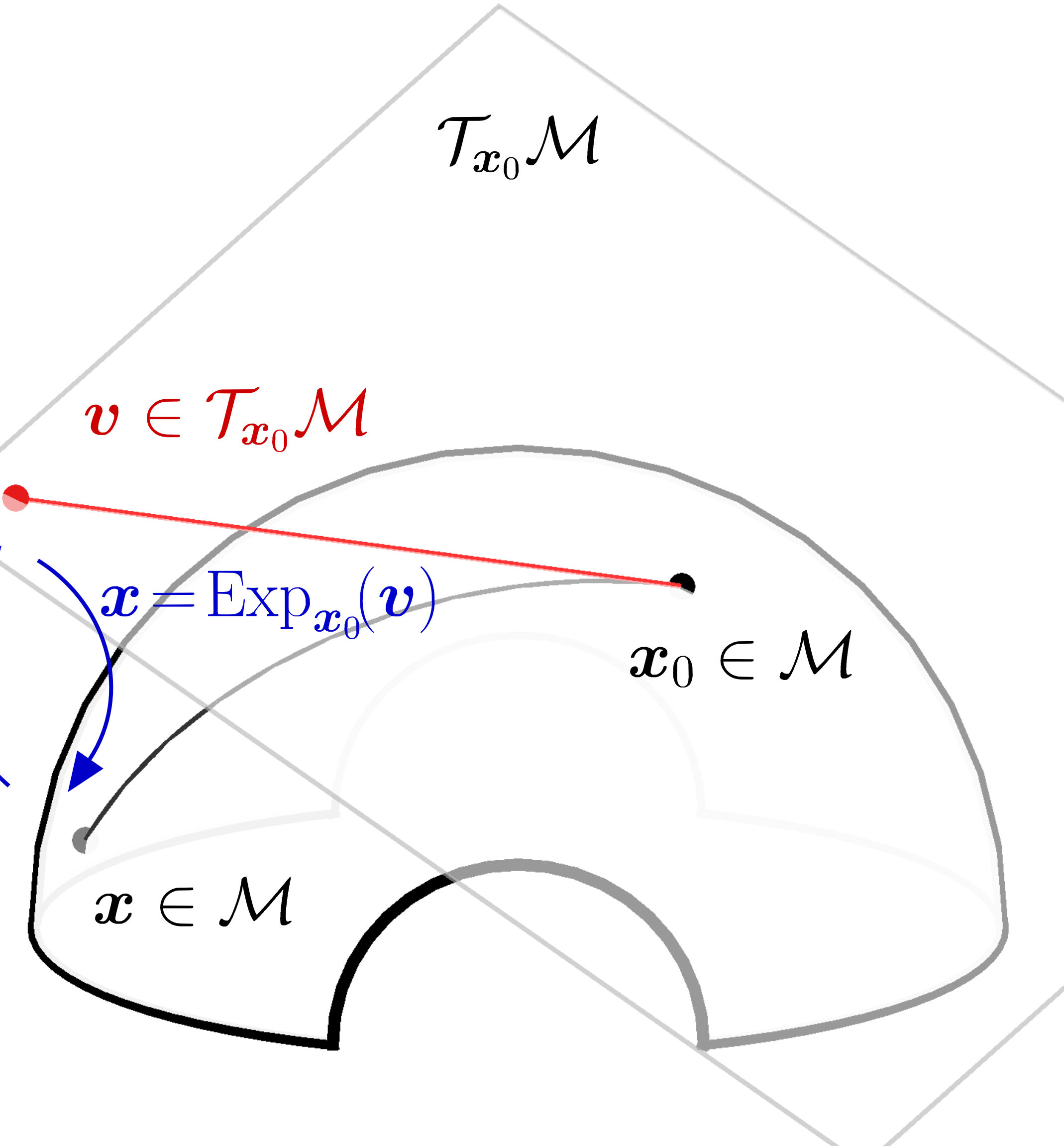
- **Model predictive control (MPC)**
  Linear quadratic tracking (LQT)
  Task-parameterized movement models

- **Differential geometry**
  Riemannian manifolds

Riemannian geometry: use of tangent spaces

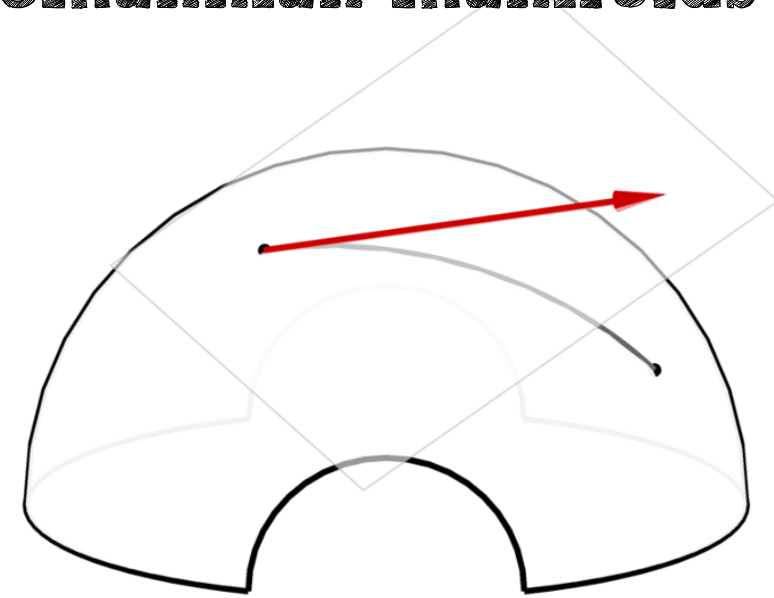




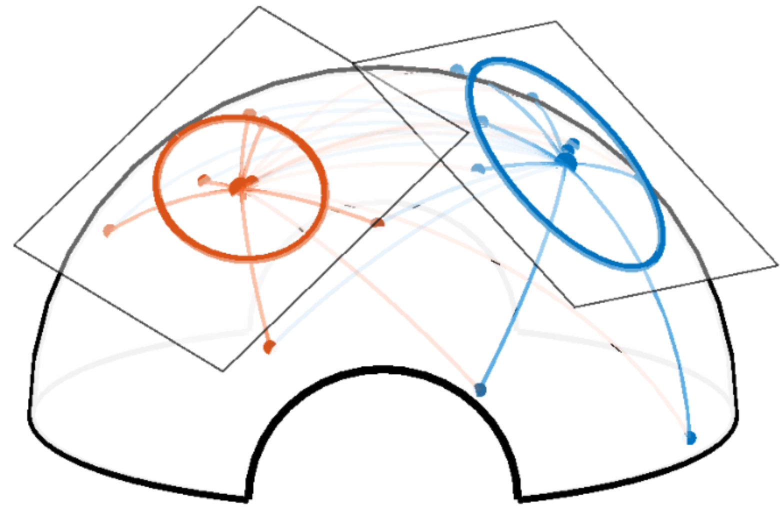


Sphere and orientation (unit quaternion) manifolds

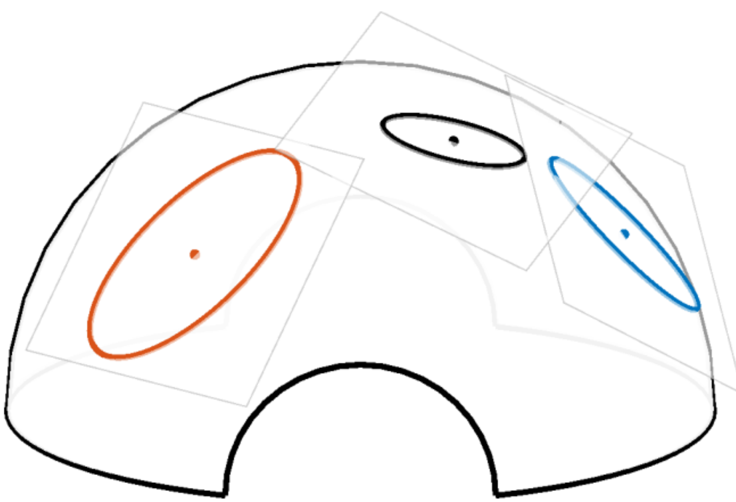
Riemannian manifolds



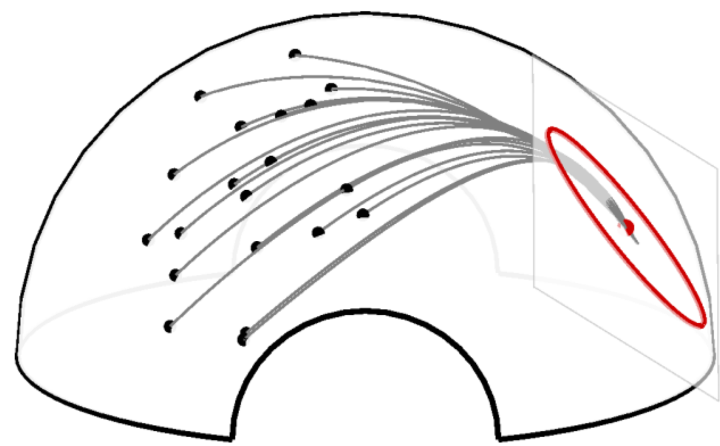
Interpolation and extrapolation



Clustering and distribution



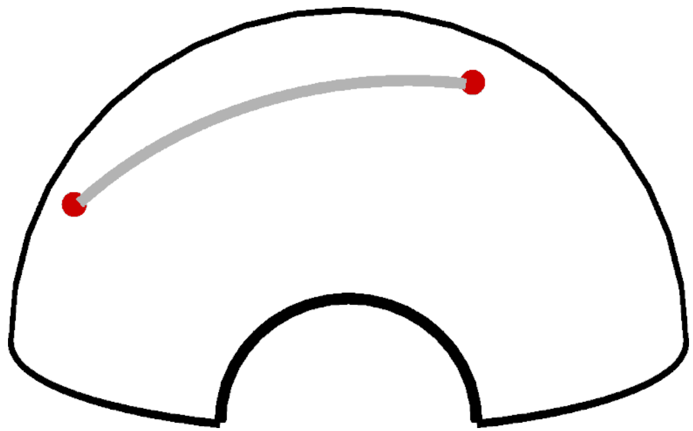
Fusion of sensing/control information

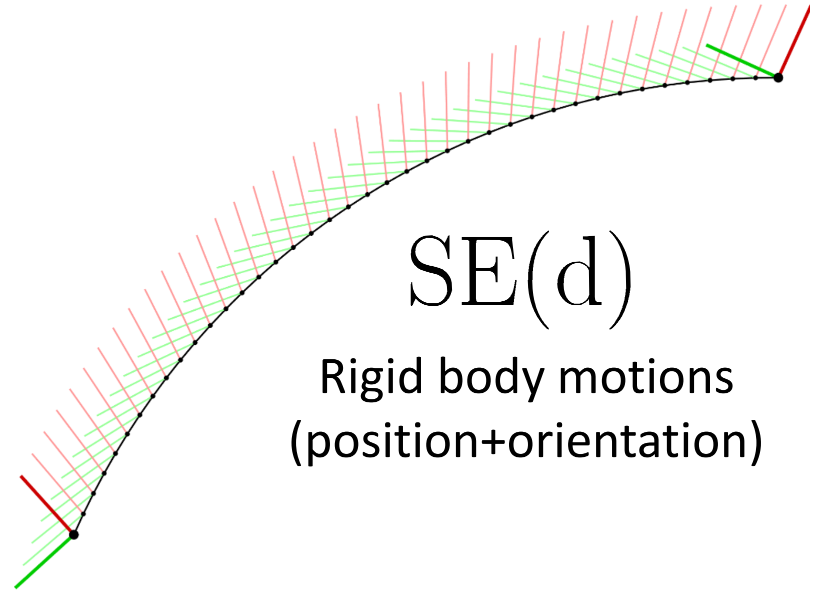


Linear quadratic tracking

Interpolation on Riemannian manifolds

Orientation
(unit quaternions) $\mathcal{S}^d$



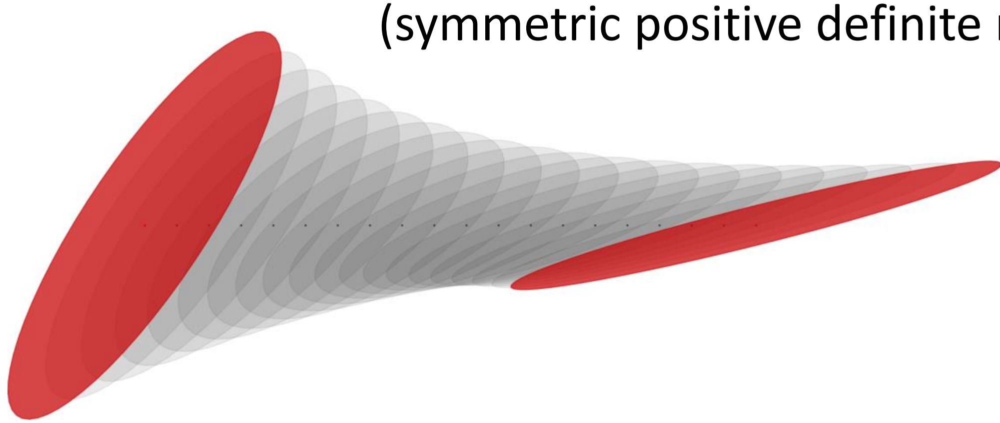


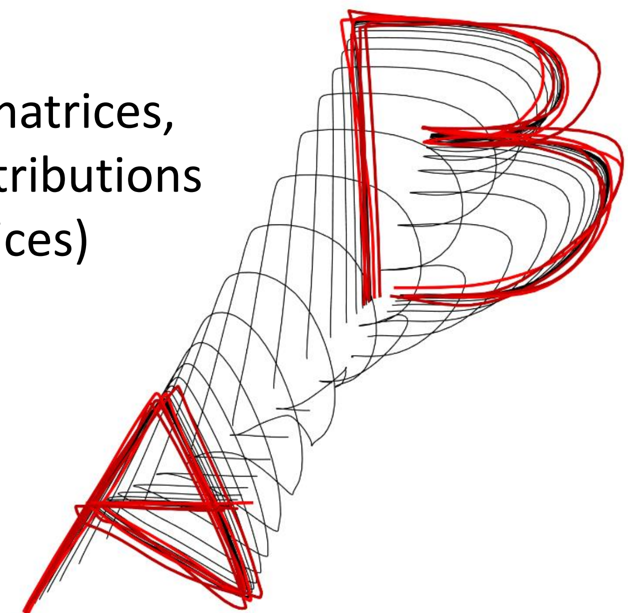
$\mathrm{SE(d)}$

Rigid body motions
(position+orientation)

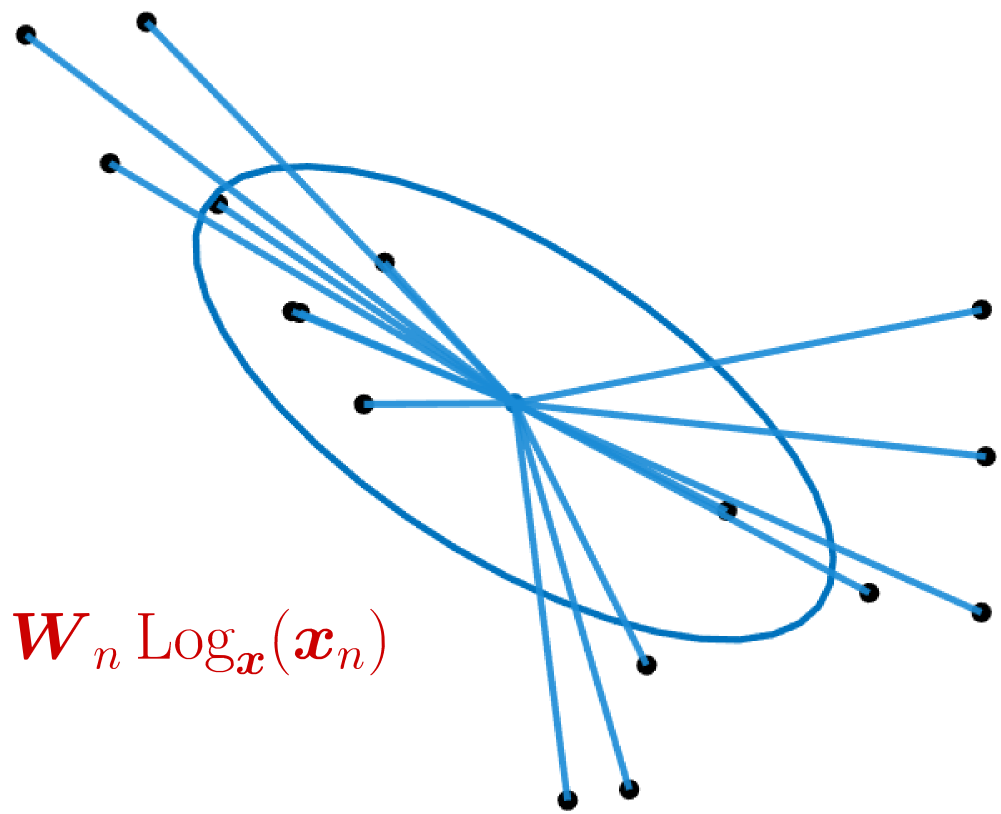
$\mathcal{S}^d_{++}$

Covariance features, inertia and gain matrices,
manipulability ellipsoids, trajectory distributions
(symmetric positive definite matrices)





Statistics on Riemannian manifolds



$$\boldsymbol{\mu} = \arg\min_{\boldsymbol{x}} \sum_{n=1}^{N} \frac{1}{2} \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{x}_n)^{\top} \boldsymbol{W}_n \, \mathrm{Log}_{\boldsymbol{x}}(\boldsymbol{x}_n)$$

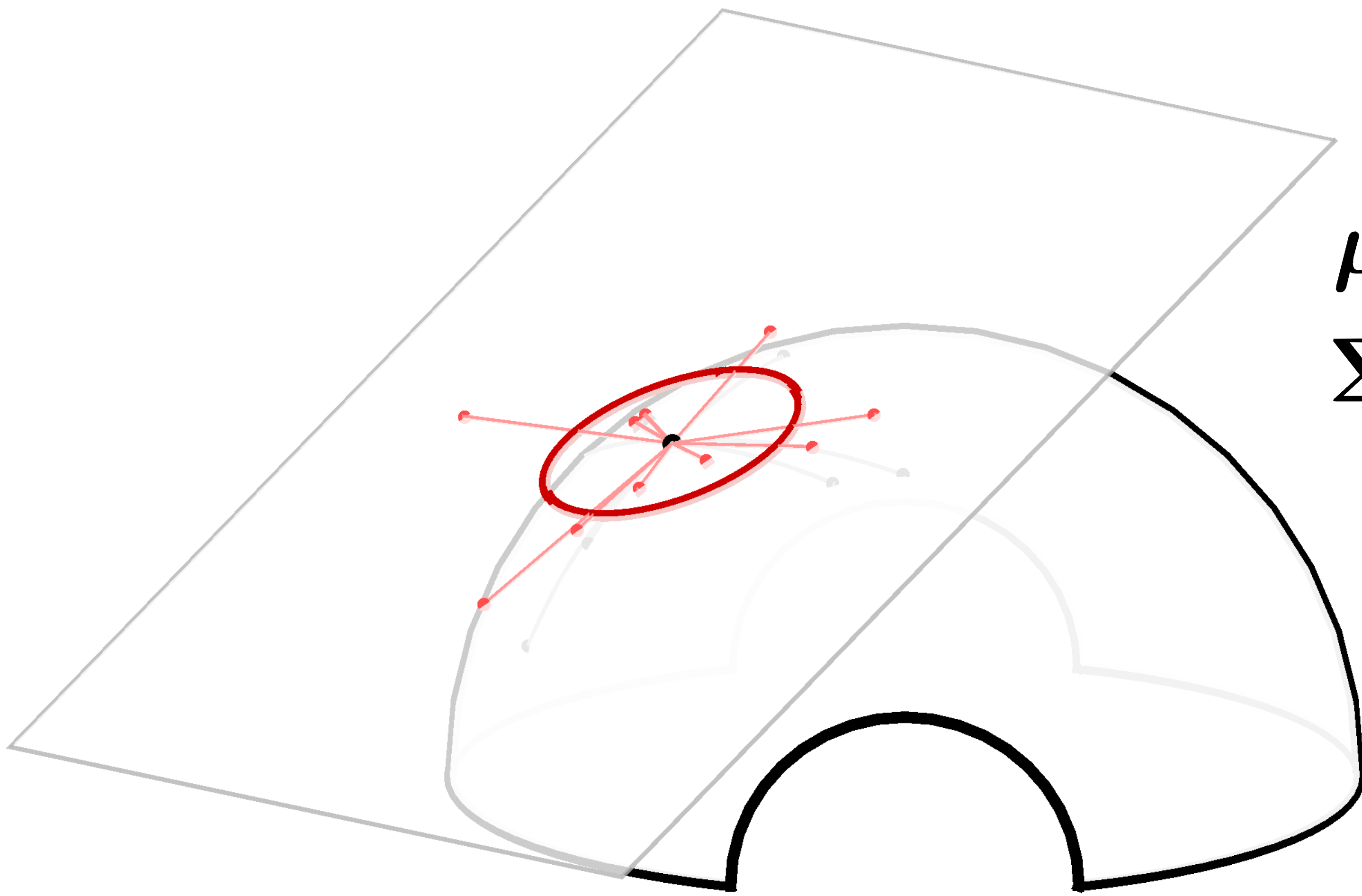
$$\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^{\top}$$

N datapoints

Riemannian geometry: Gaussian distribution

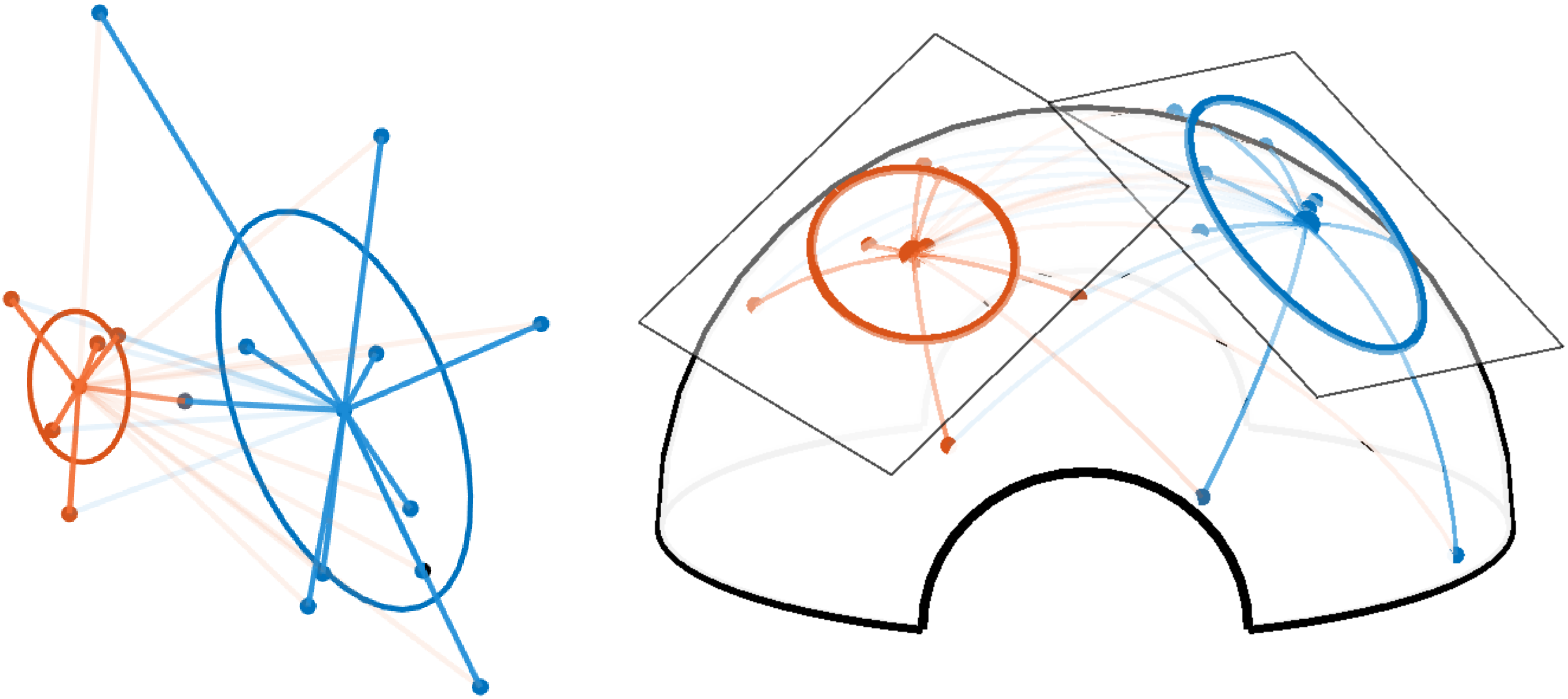
$$\mathcal{N}_{\mathcal{M}}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}\ \exp\Big(-\frac{1}{2}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})^{\top}\boldsymbol{\Sigma}^{-1}\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{x})\Big)$$



$\boldsymbol{\mu} \in \mathcal{M}$

$\boldsymbol{\Sigma} \in \mathcal{T}_{\boldsymbol{\mu}}\mathcal{M}$

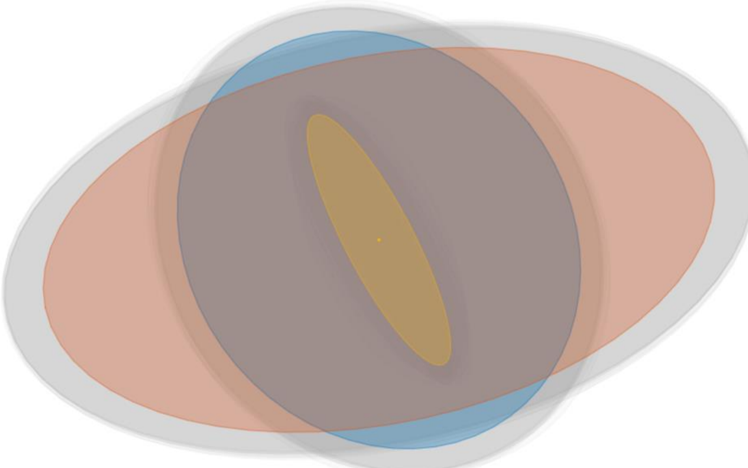
Clustering on Riemannian manifolds



$$\mathcal{P}(\boldsymbol{x}_n) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

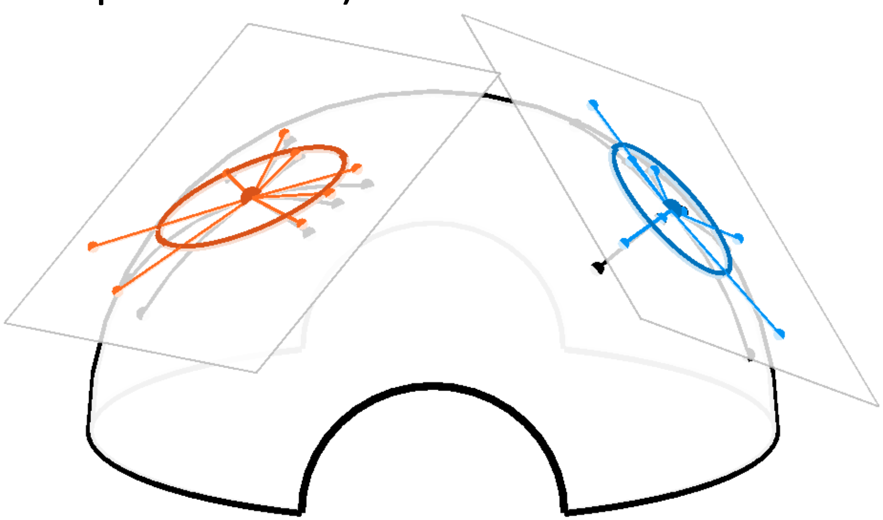
K Gaussians
N datapoints

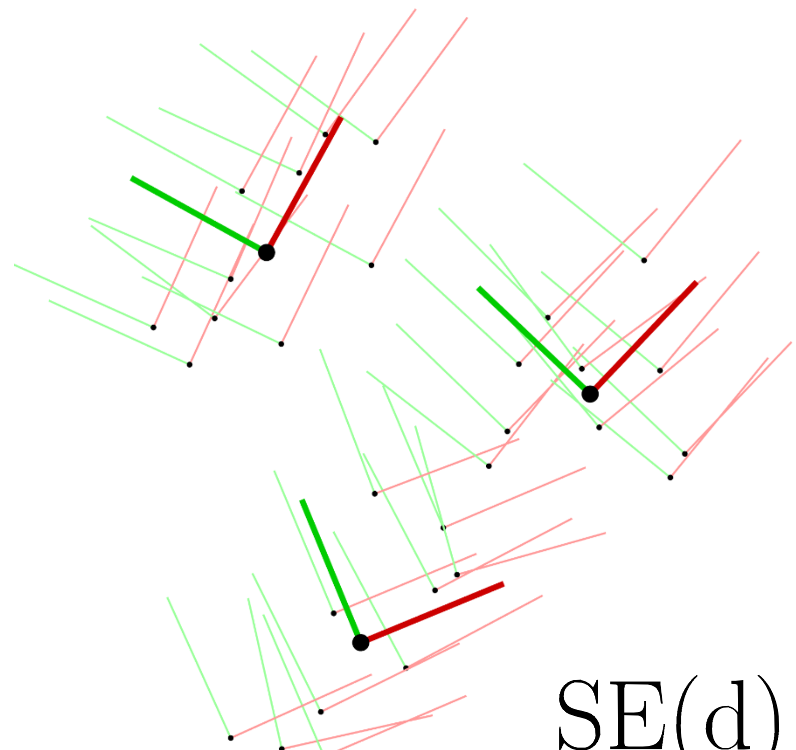
Clustering on Riemannian manifolds



$\mathcal{S}^d_{++}$ Covariance features, inertia and gain matrices, manipulability ellipsoids, trajectory distributions (symmetric positive definite matrices)

Orientation (unit quaternions) $\mathcal{S}^d$





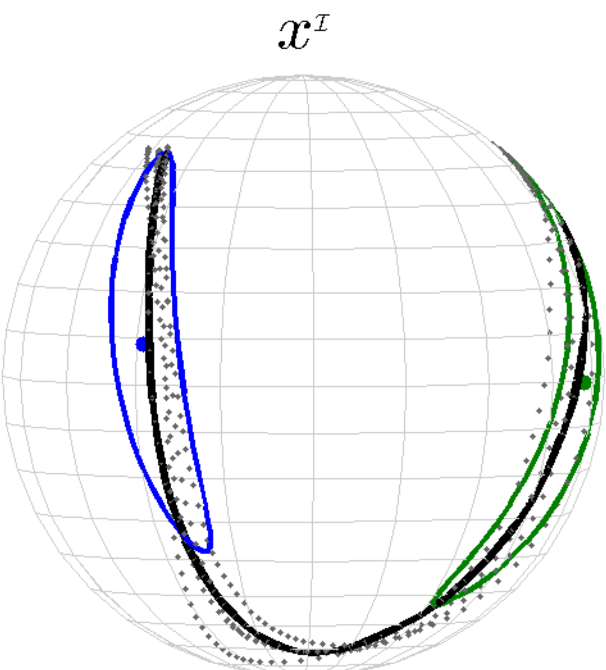
$\mathrm{SE(d)}$

Rigid body motions (position+orientation)

Regression on Riemannian manifolds

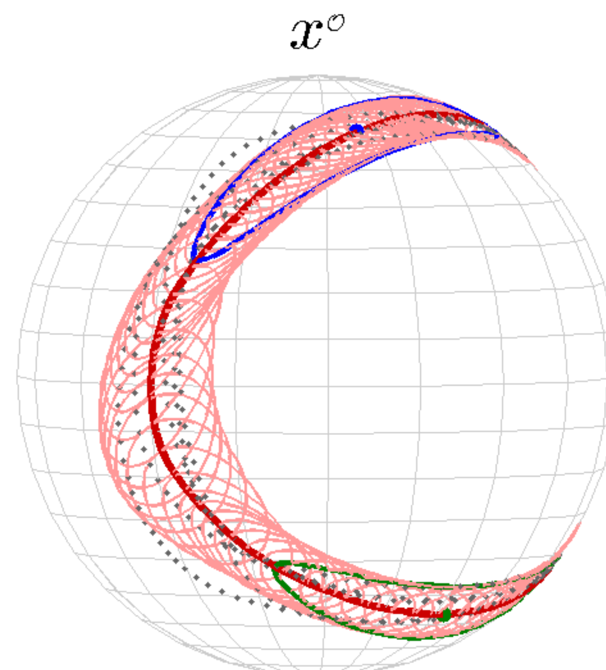
Gaussian mixture regression (GMR) to compute $\mathcal{P}(\boldsymbol{x}^{\mathcal{O}}|\boldsymbol{x}^{\mathcal{I}})$

from the joint distribution $\mathcal{P}(\boldsymbol{x}^{\mathcal{I}}, \boldsymbol{x}^{\mathcal{O}})$ encoded as a GMM



$$\boldsymbol{x}^{\mathcal{I}} \in \mathcal{S}^2$$

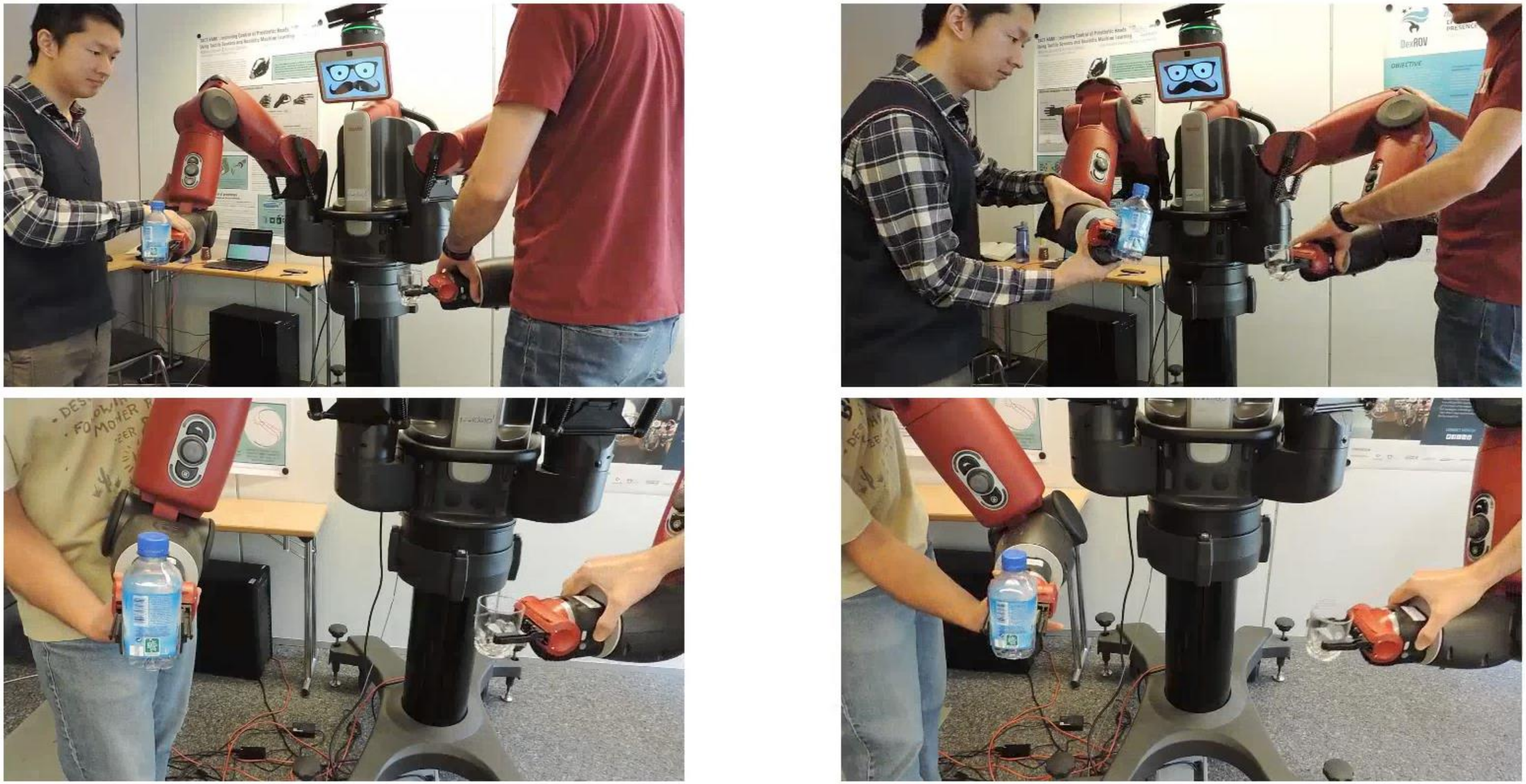
$$\boldsymbol{x}^{\mathcal{O}} \in \mathcal{S}^2$$



→ **Regression for orientation data (unit quaternions on $\mathcal{S}^3$)**

Regression with orientation and position data

Four demonstrations of coordinated bimanual movement



[Zeestraten, Havoutis, Silverio, Calinon and Caldwell, IEEE RA-L 2(3), 2017]

Regression with orientation and position data

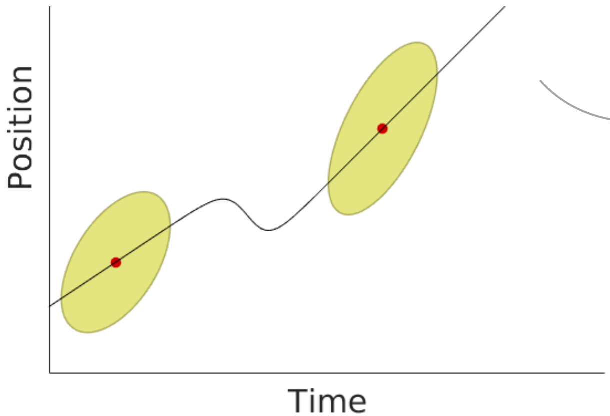
Four reproductions with perturbations by the user

[Zeestraten, Havoutis, Silverio, Calinon and Caldwell, IEEE RA-L 2(3), 2017]

Regression with covariance features

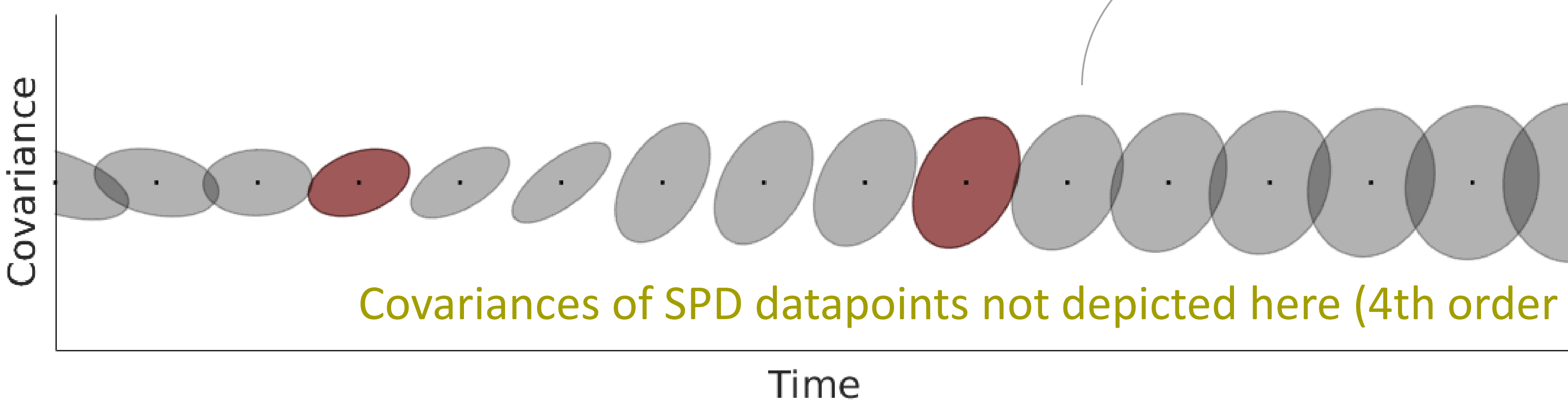
Gaussian mixture regression (GMR) with SPD datapoints

- centers
- covariances



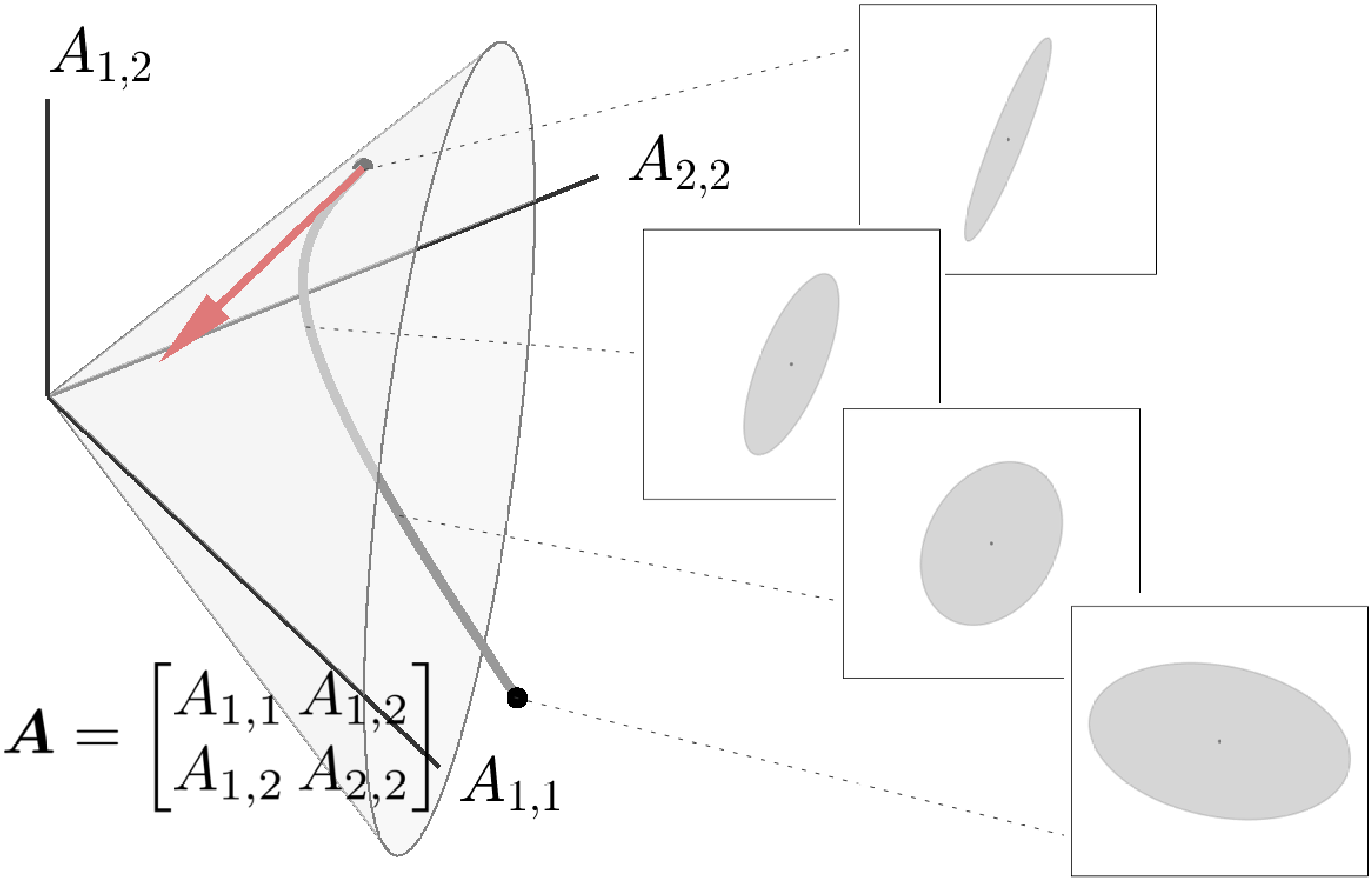
$$\boldsymbol{x}^{\mathcal{I}} \in \mathbb{R}$$
$$\boldsymbol{x}^{\mathcal{O}} \in \mathbb{R}$$

$$\boldsymbol{x}^{\mathcal{I}} \in \mathbb{R}$$
$$\boldsymbol{x}^{\mathcal{O}} \in \mathcal{S}^2_{++}$$



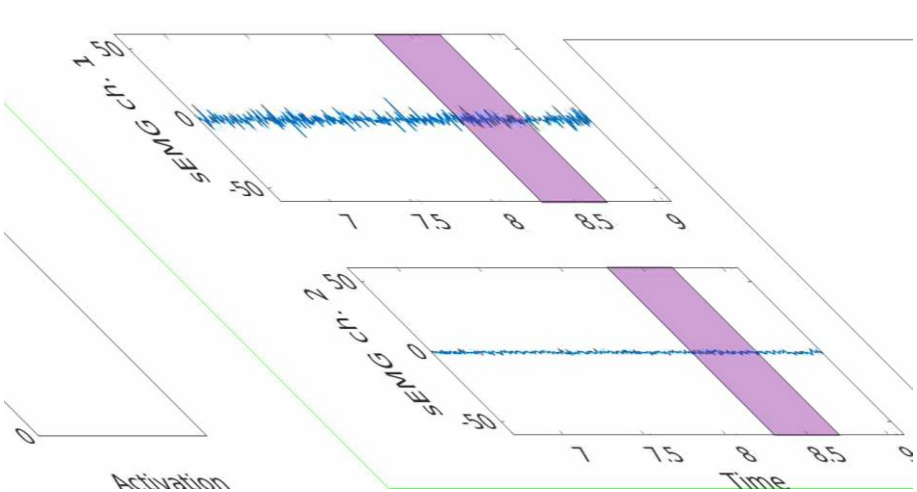
→ **Predicting the temporal change of covariance features**

Regression with covariance features



Symmetric positive definite (SPD) matrix manifold

Regression with electromyography data

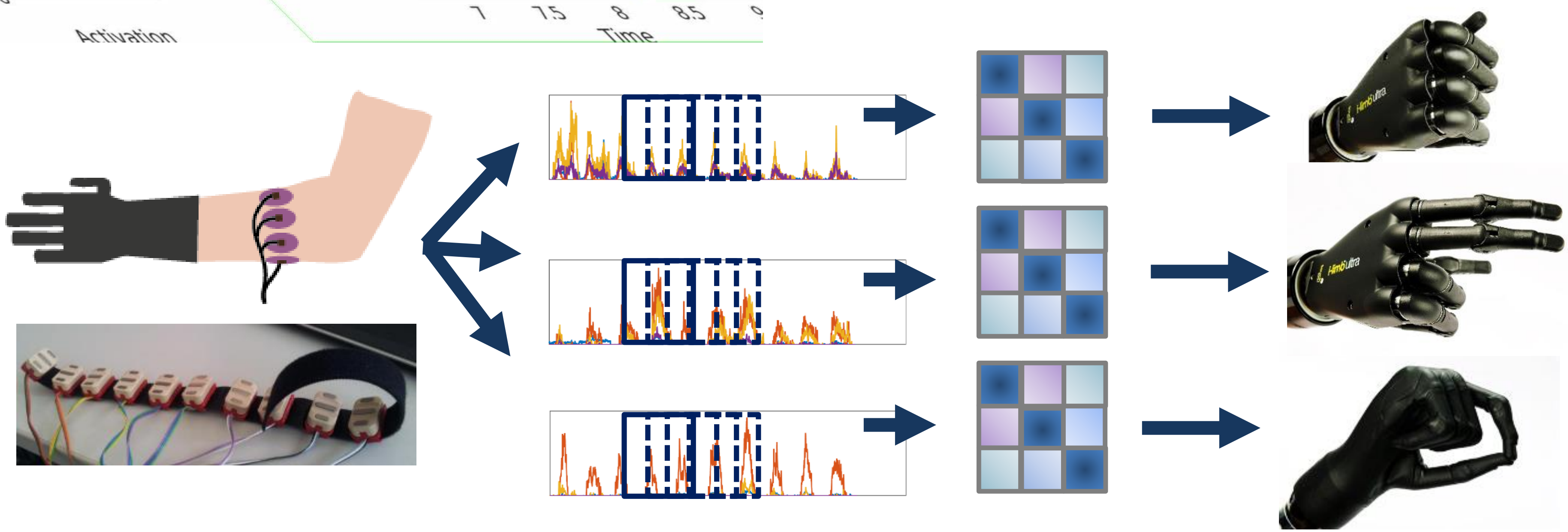


Deutsches Zentrum für Luft- und Raumfahrt

German Aerospace Center

TACT-HAND

SNSF, D-A-CH (2016-2019)



Surface electromyography (**sEMG**) measurements

Transformation in **spatial covariances** (SPD matrices)

Control of the corresponding hand pose

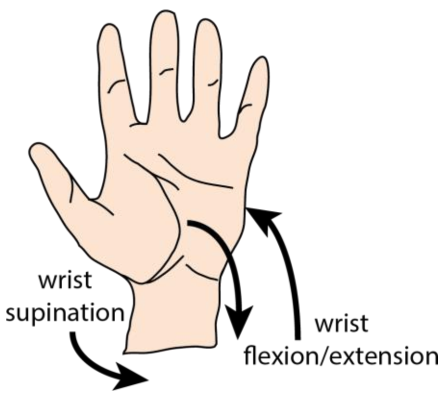
[Jaquier and Calinon, IROS 2017]

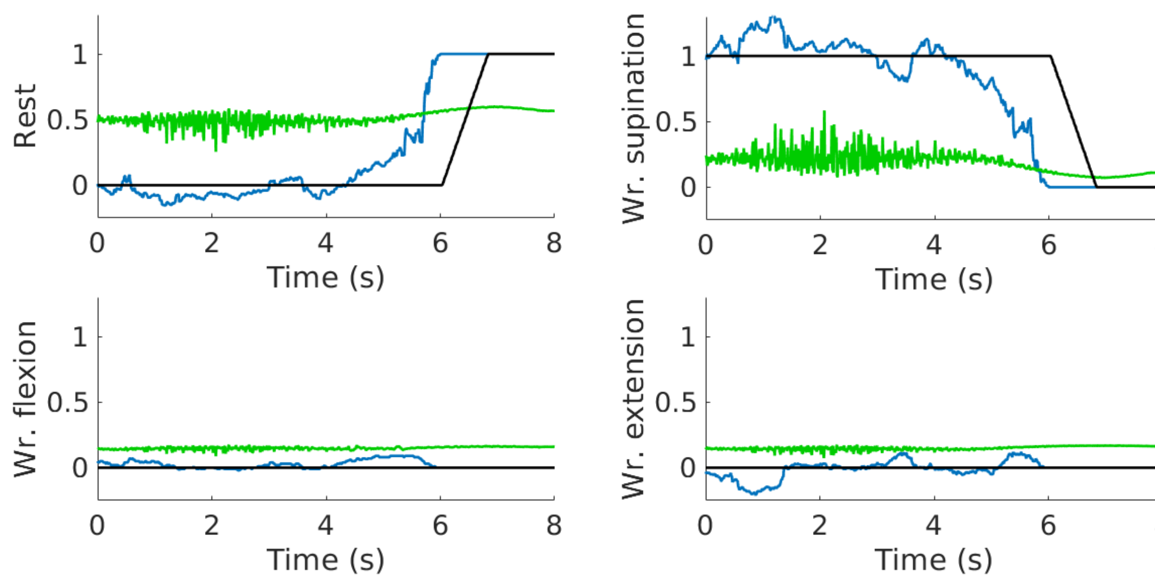
Regression with electromyography data

sEMG data from Ninapro database processed as spatial covariances:

$$\text{Input} \in \mathcal{S}_{++}^{12}$$

$$\text{Output} \in \mathbb{R}^4$$

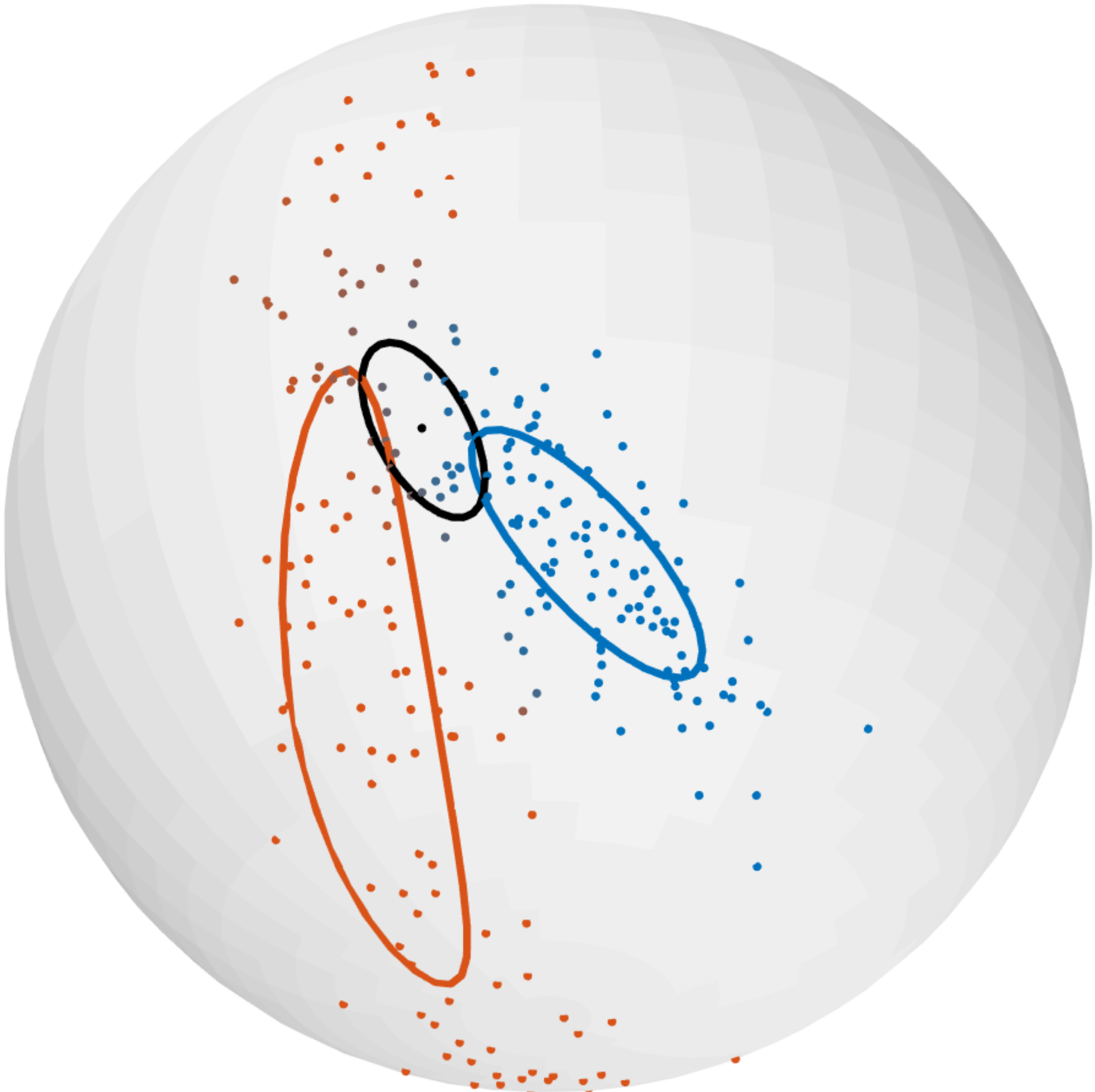




- Reference
- Standard regression
- Regression on SPD manifold

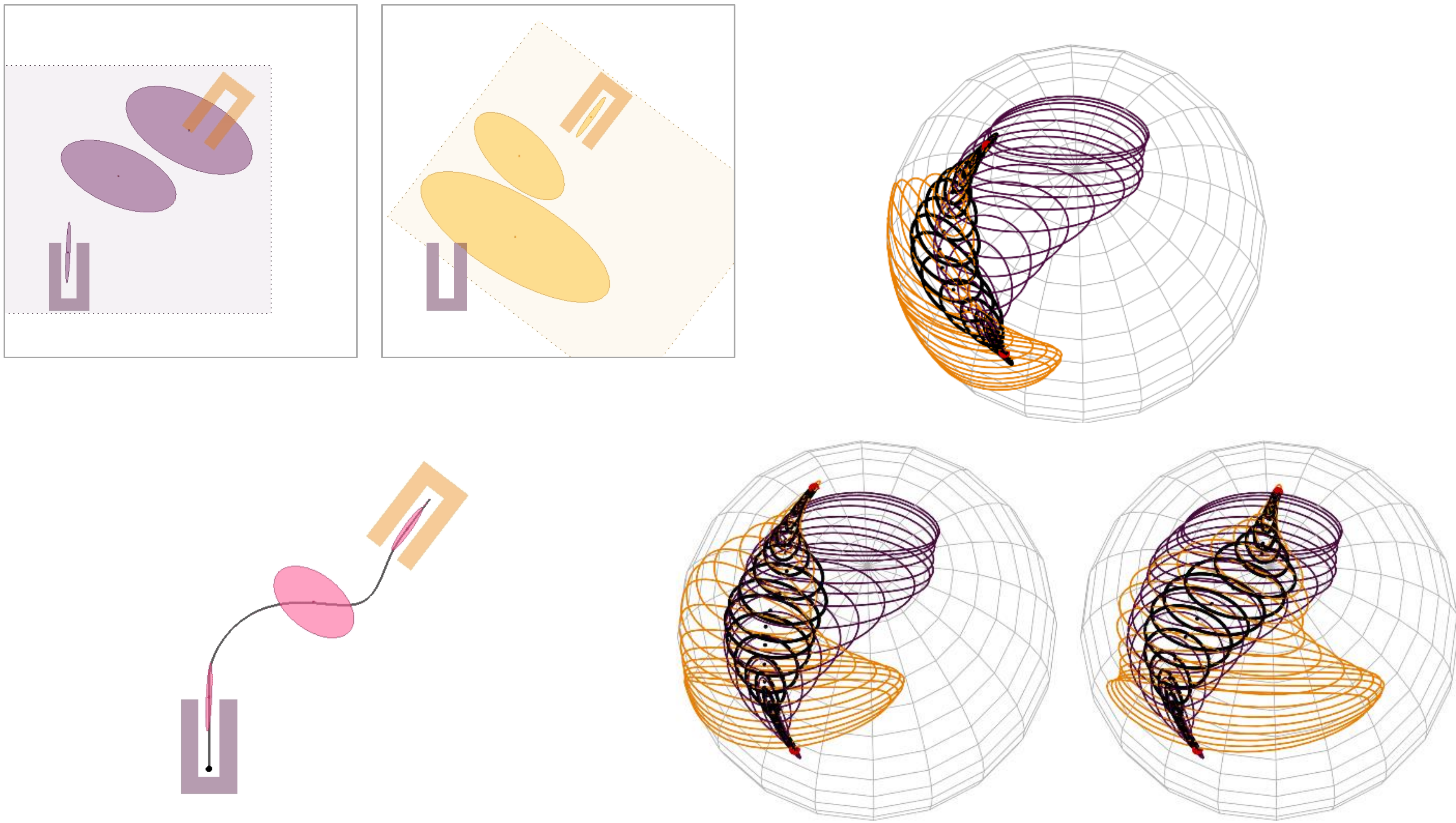
[Jaquier and Calinon, IROS 2017]

Fusion (product of Gaussians) on manifolds



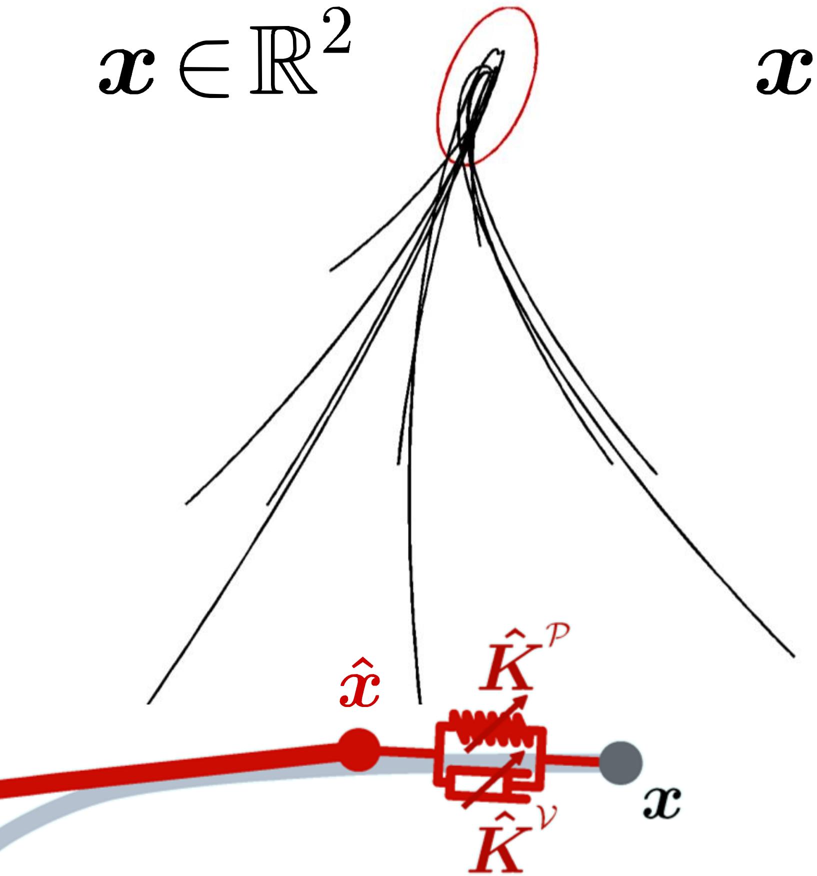
K Gaussians

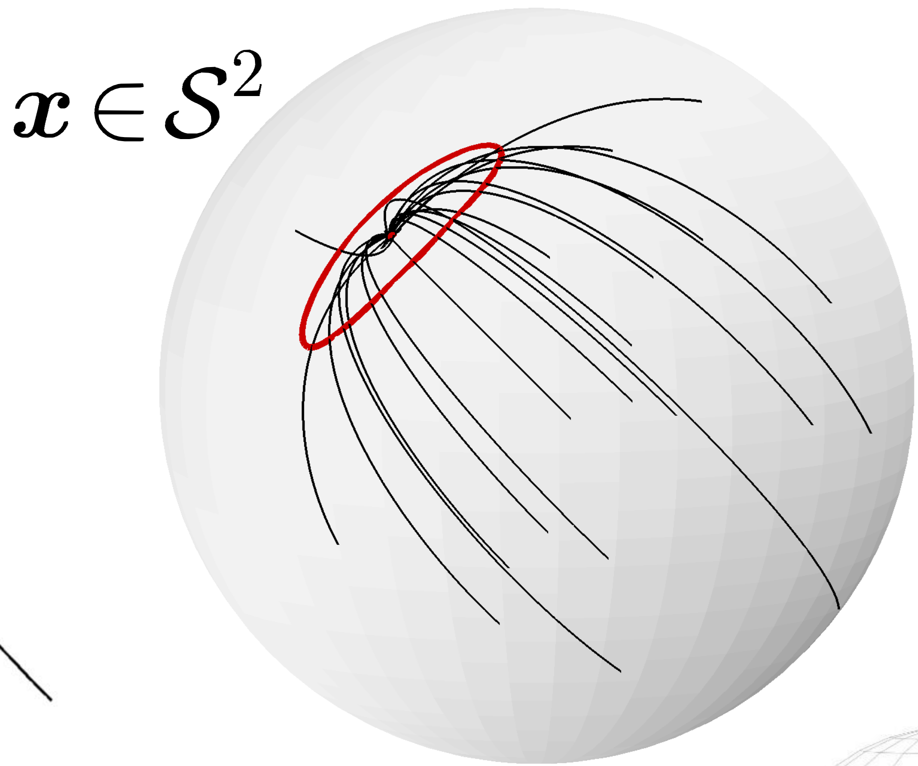
Fusion (product of Gaussians) on manifolds

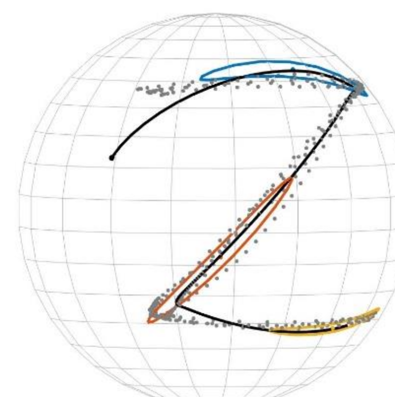


→ **Control the orientation of the robot hand in accordance to the orientation of objects, tools or virtual landmarks**

Control on Riemannian manifolds







Example: Model predictive control of orientation with unit quaternions by using Riemannian geometry

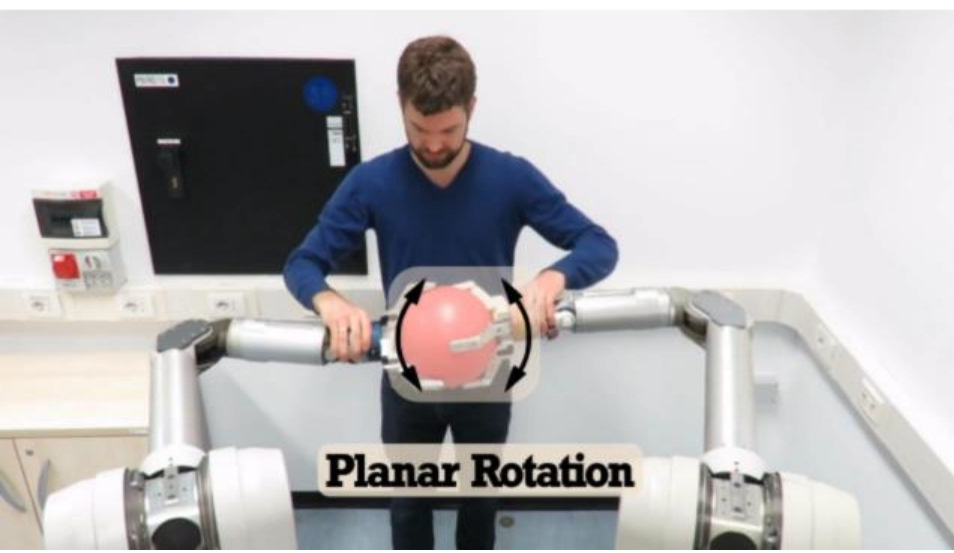
Control on Riemannian manifolds

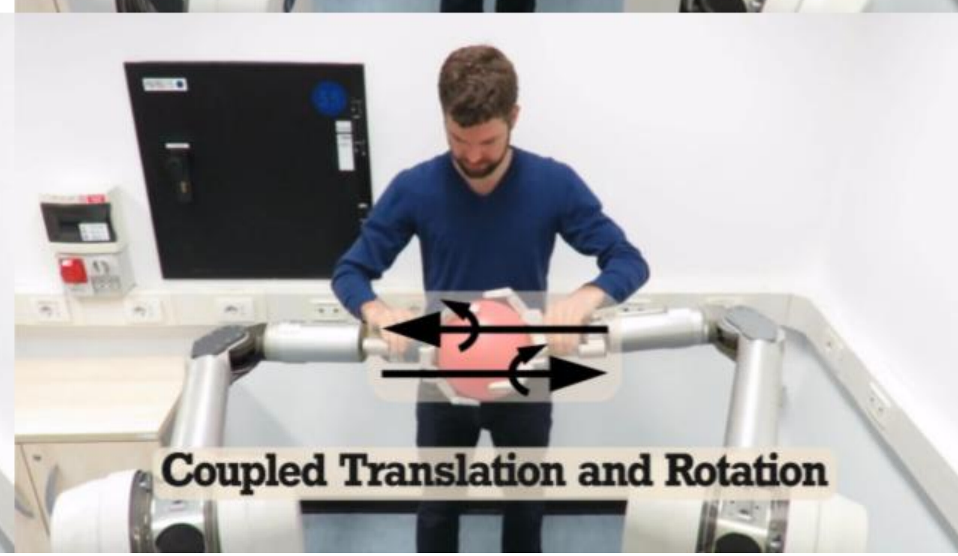
We demonstrate three different tasks, each requiring a different synergy between the end-effectors.

[Zeestraten, Havoutis, Calinon and Caldwell, IROS'17]

Control on Riemannian manifolds

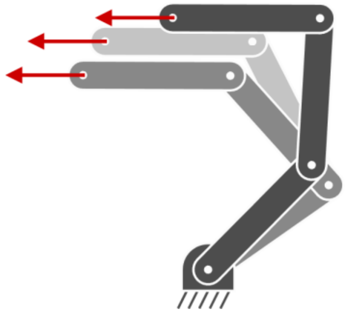
We demonstrate three different tasks, each requiring a different synergy between the end-effectors.

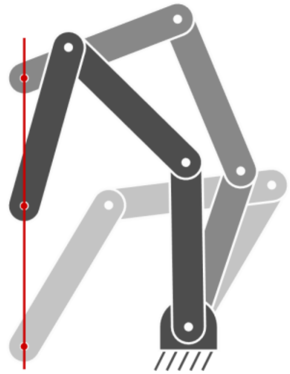


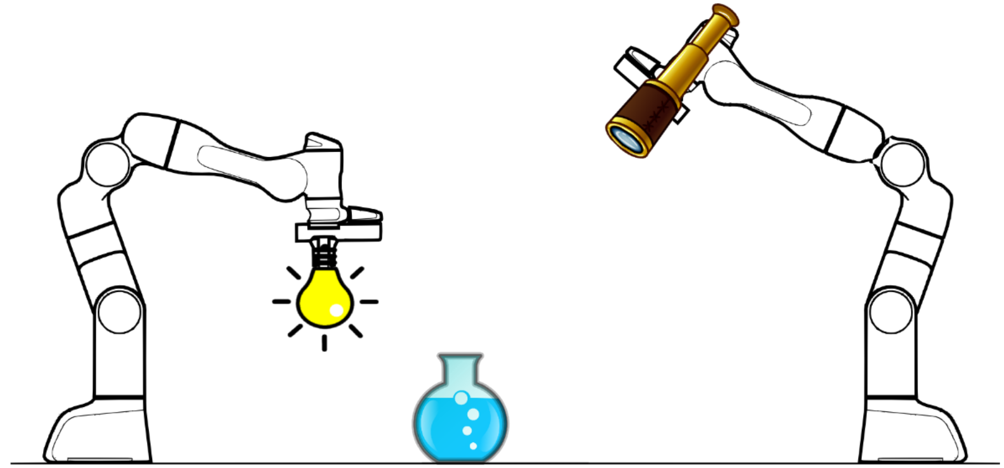


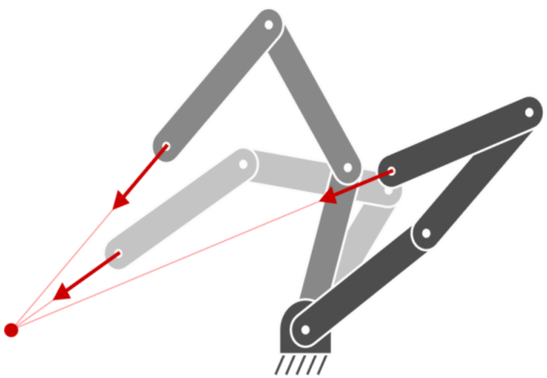
[Zeestraten, Havoutis, Calinon and Caldwell, IROS'17]

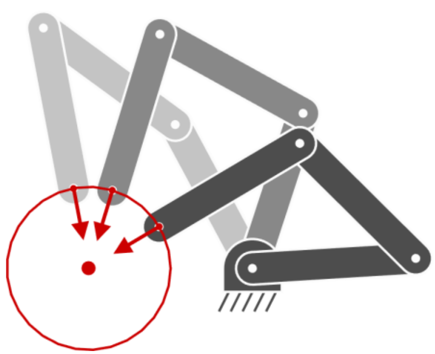
Platform for reproducible research in science

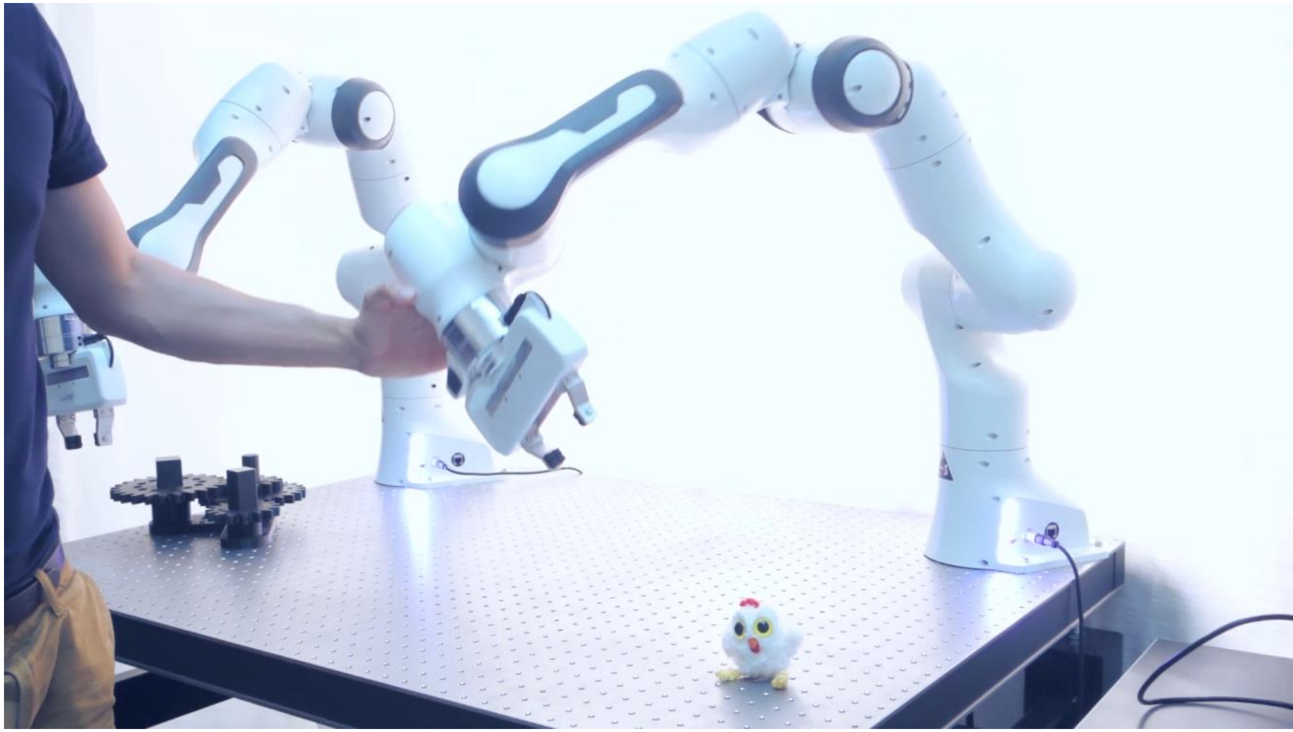












Riemannian geometry - Resources

Softwares

http://www.idiap.ch/software/pbdlib/

Matlab codes: demo_Riemannian_sphere_GMM01.m

C++ codes: demo_Riemannian_sphere_GMM01.cpp

References

[Zeestraten, Havoutis, Silvério, Calinon and Caldwell, *"An Approach for Imitation Learning on Riemannian Manifolds"*, IEEE Robotics and Automation Letters 2(3), 2017]

[Jaquier and Calinon, *"Gaussian Mixture Regression on Symmetric Positive Definite Matrices Manifolds: Application to Wrist Motion Estimation with sEMG"*, IROS'2017]

Source codes

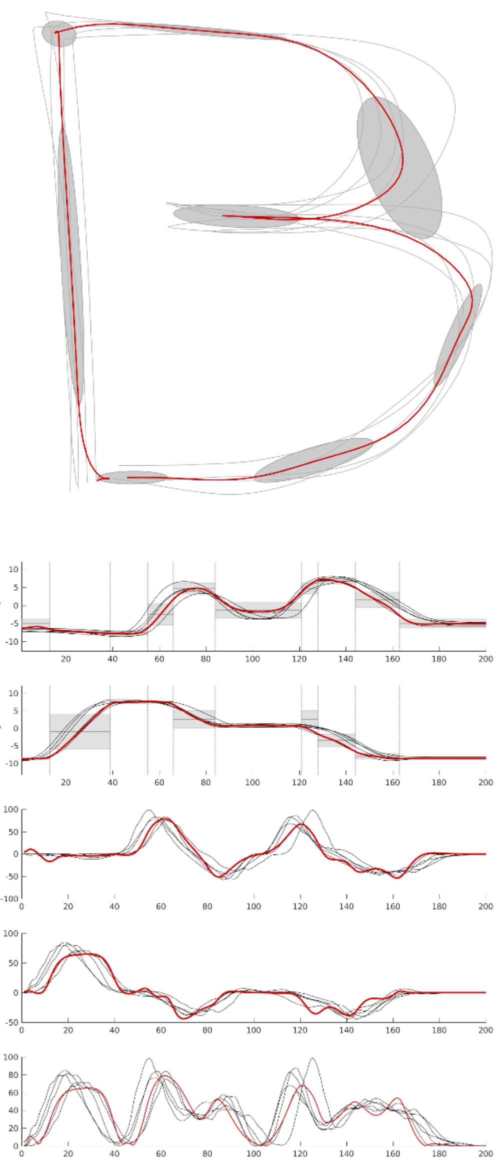
http://www.idiap.ch/software/pbdlib/

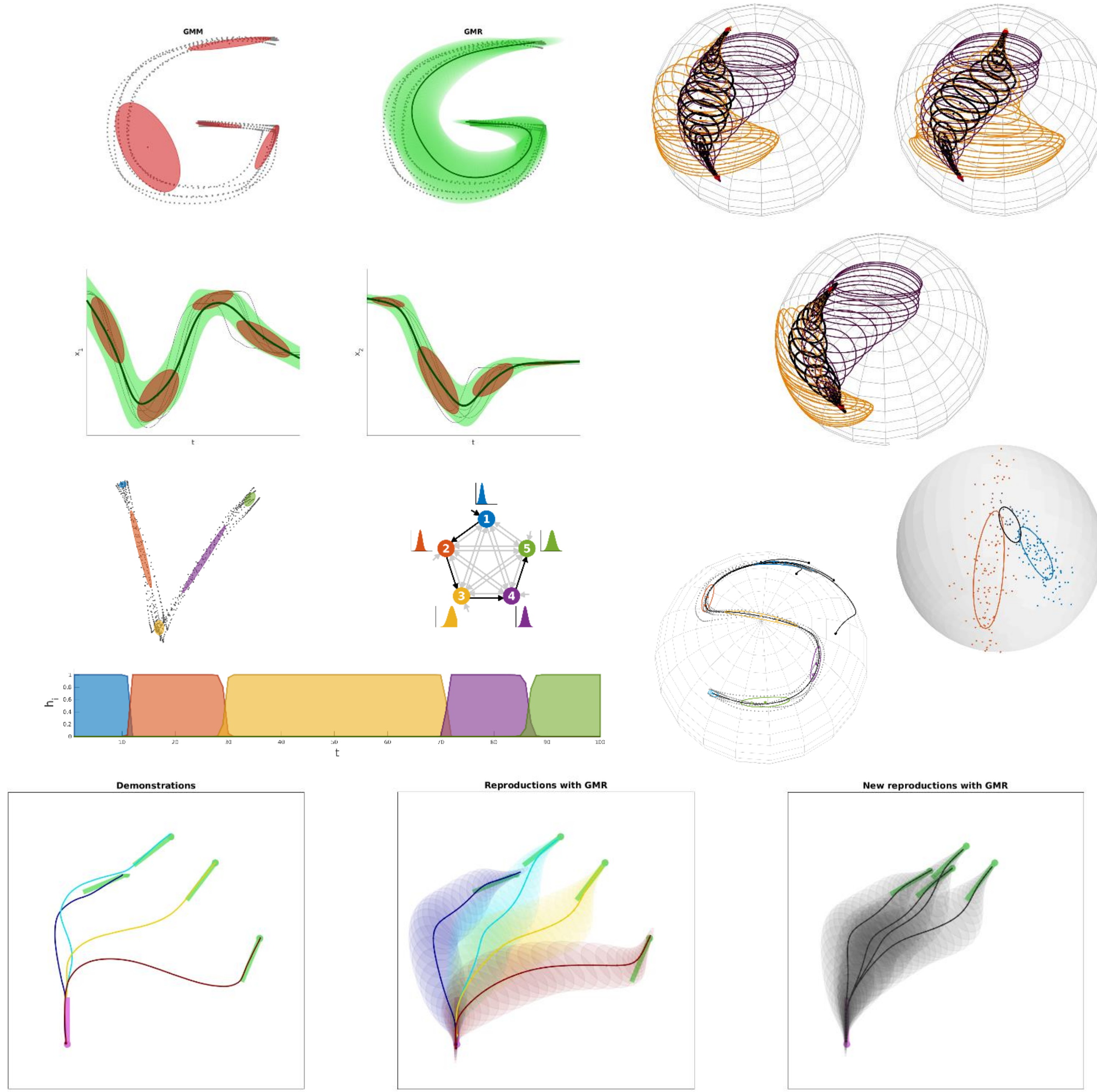
Matlab / GNU Octave

C++

Python

Pbdlib





http://www.idiap.ch/software/pbdlib/

PbDlib - Mozilla Firefox 13:48

PbDlib

idiap.ch/software/pbdlib/

PbDlib

PbDlib is a collection of source codes for robot programming by demonstration (learning from demonstration). It includes a varied set of functionalities at the crossroad of statistical learning, dynamical systems, optimal control and differential geometry. It is available in the following languages:

- Matlab / GNU Octave
- C++
- Python

PbDlib can be used in applications requiring task adaptation, human-robot skill transfer, safe controllers based on minimal intervention principle, as well as for probabilistic motion analysis and synthesis in multiple coordinate systems.

Terminal

14:10

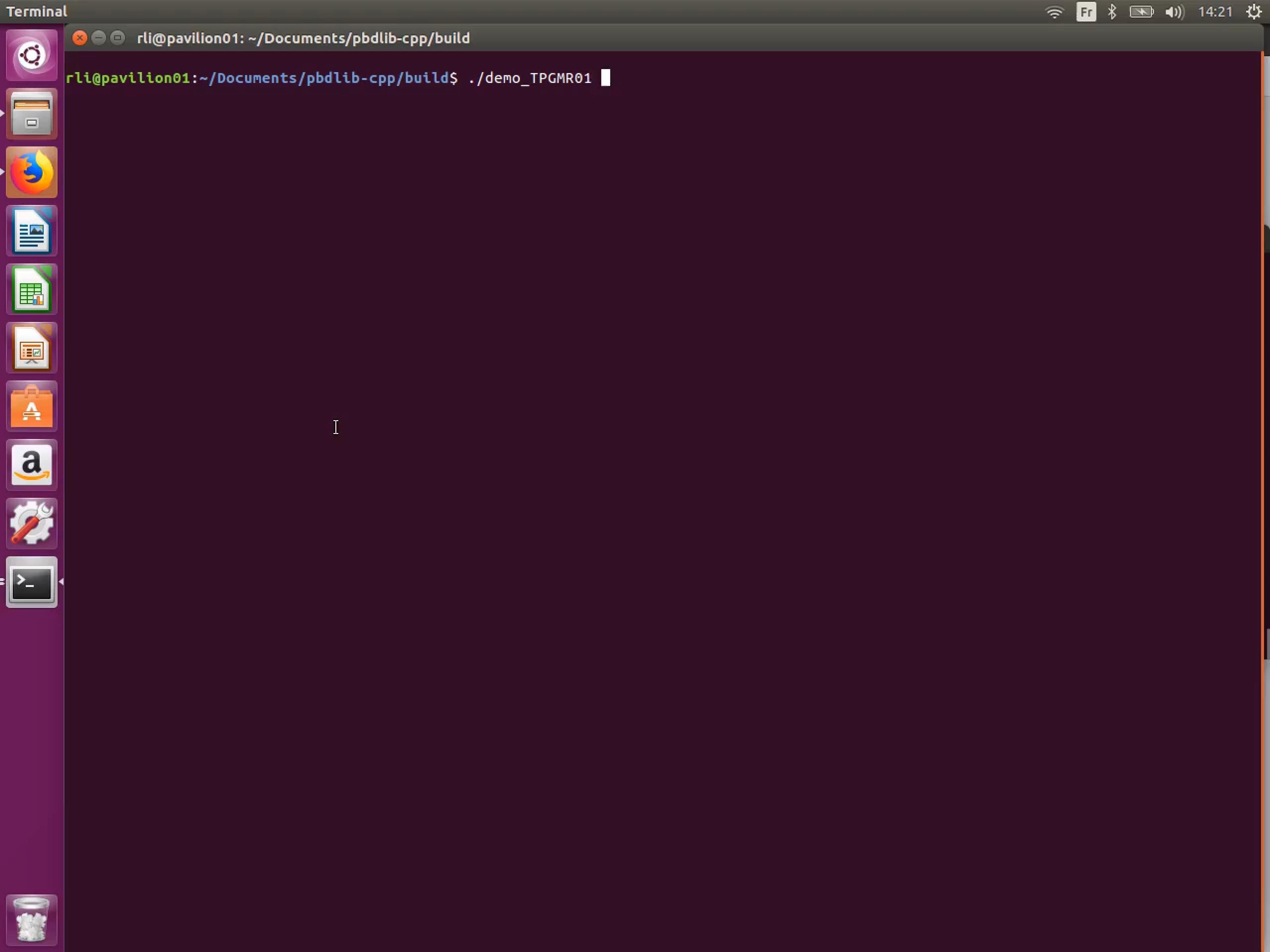
rli@pavilion01: ~/Documents/pbdlib-cpp/build

```
rli@pavilion01:~/Documents/pbdlib-cpp/build$ ./demo_HSMM_batchLQR01
```

Terminal 14:13

rli@pavilion01: ~/Documents/pbdlib-cpp/build

```
rli@pavilion01:~/Documents/pbdlib-cpp/build$ ./demo_TPGMMProduct01
```


Terminal
14:21
rli@pavilion01: ~/Documents/pbdlib-cpp/build
rli@pavilion01:~/Documents/pbdlib-cpp/build$ ./demo_TPGMR01

Terminal

14:16

rli@pavilion01: ~/Documents/pbdlib-cpp/build

```
rli@pavilion01:~/Documents/pbdlib-cpp/build$ ./demo_TPbatchLQR01
```

Source codes (Matlab/Octave, C++ and Python):

http://www.idiap.ch/software/pbdlib/

Contact:

sylvain.calinon@idiap.ch
http://calinon.ch

idiap

Photo: Basilio Noris