Higher accuracy and fast convergence in learning the kinematics thanks to effective representations. **Advantageous representations** are mandatory even for low dimensional **machine learning** application!

On the Merits of Joint Space and Orientation Representations in Learning the Forward Kinematics in SE(3)

Reinhard M. Grassmann and Jessica Burgner-Kahrs

Motivation

- Regression is more sensitive w.r.t. input than classification
- Learning methods and architecture are investigated so far
- What are the right representations?
- Are common representations applicable?

Approach

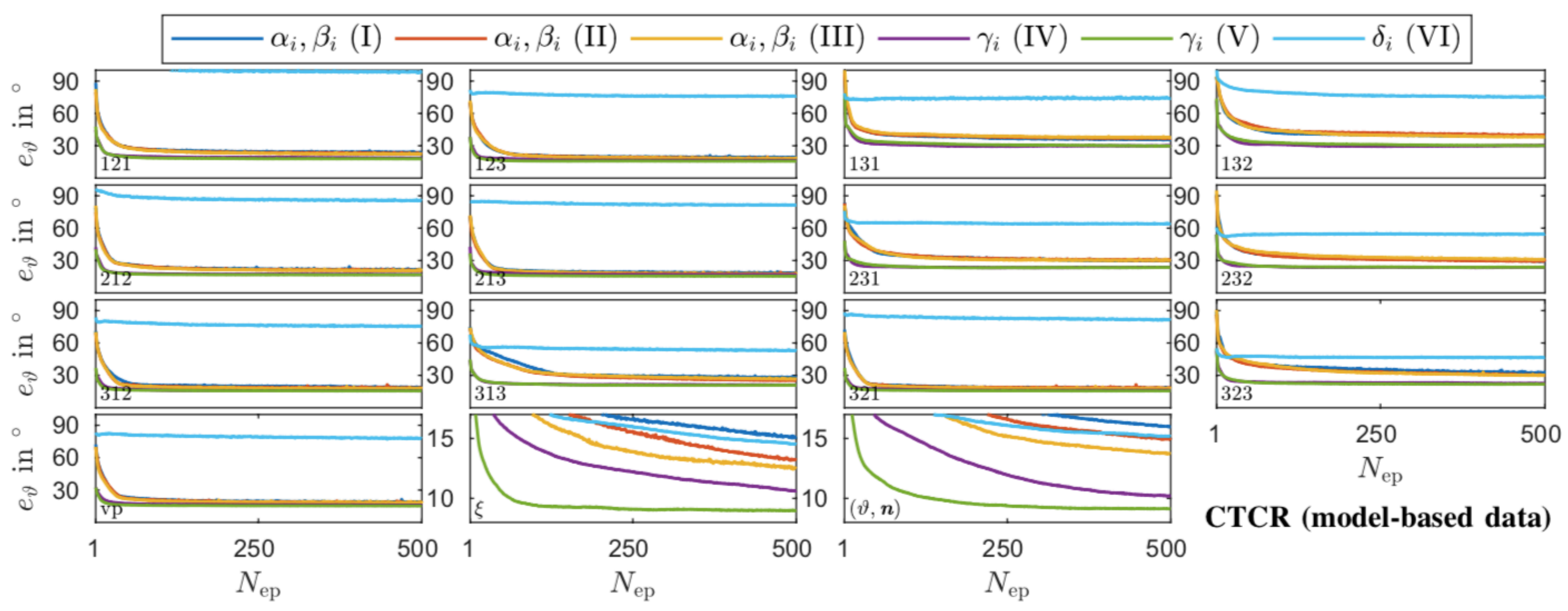
- Shallow artificial neural network
- Empirical study with different robot types
- 6 different joint space representations
- 15 different orientation representations
- Normalization w.r.t. number of neurons
- Affine transformation for disentanglement

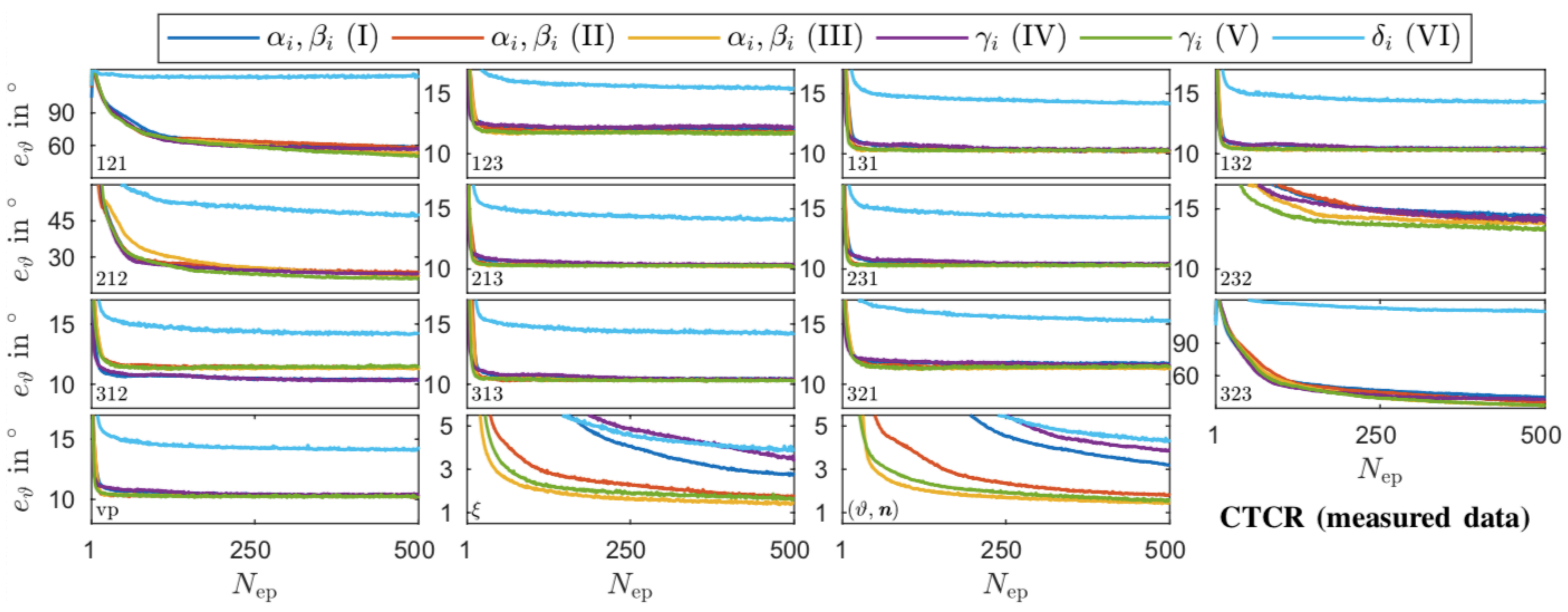
Results

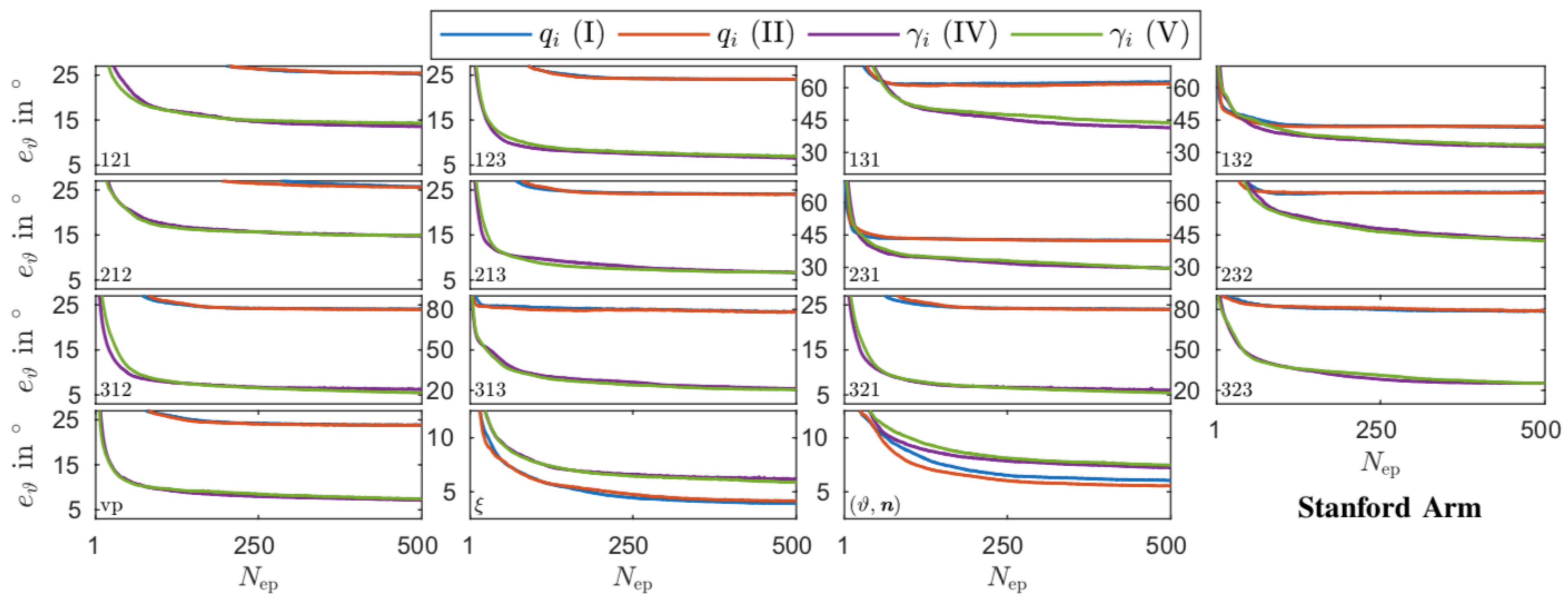
- Better with quaternions
- Better with affine transformation

Conclusion

- Use quaternion/vector-pair
- Transform your inputs and outpus
- Mandatory even for low dimensional problems







APPLIED SCALING FOR THE JOINT SPACE REPRESENTATION.

Legend[1]	Description of the scaling
q_i (I)	No scaling is applied to the joints of the Stanford Arm.
q_i (II)	Each joint is scaled via the maximum value, cf. Table I.
γ_i (IV)	Rotary joints of the Stanford Arm are transformed by means of trigonometric functions similar to (17).
γ_i (V)	Rotary joints are transformed by means of trigonometric functions, whereas the prismatic joint is scaled by 0.75 m.
α_i, β_i (I)	No scaling is applied to the joints of the CTCR.
α_i, β_i (II)	Each joint of the CTCR is scaled by the absolute value, i.e. $\alpha_{i,\max}$ and $\beta_{i,\min}$, cf. Table II.
α_i, β_i (III)	α_i is scaled by $\alpha_{i,\max}$ whereas β_i is transformed into an unit cube utilizing the inverse of (22).
γ_i (IV)	The cylindrical form γ_i is applied, see (17).
γ_i (V)	Rotary joints of the CTCR are transformed by means of trigonometric function similar to (17) while β_i is transformed by the inverse of (22).
δ_i (VI)	The polar form δ_i is used, which is given by (19).

[1] Notation in the legend used in Fig. 6, Fig. 7, and Fig. 8.

TABLE V
FEEDFORWARD NETWORKS FOR FORWARD KINEMATIC APPROXIMATION.

Architecture with ReLU			Training with Adam optimizer[2]			Weights & bias[3]
N_{ip}	N_h	N_{op}	N_{bs}	N_{ep}	λ	$\mathcal{C}$
6	100	6	128	500	3×10^{-5}	1306
6	93	7	128	500	1×10^{-5}	1309
9	81	6	128	500	3×10^{-5}	1302
9	77	7	128	500	1×10^{-5}	1316
11	68	7	128	500	1×10^{-5}	1299
11	72	6	128	500	1×10^{-5}	1302

[2] Beside of learning rate λ and following the notation in [18], it is further parametrized with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$.
[3] The complexity $\mathcal{C}$ is the sum of all weights and bias in the neural network. For this shallow neural network it can be computed by $\mathcal{C} = (1 + N_{ip})N_h + (1 + N_h)N_{op}$.

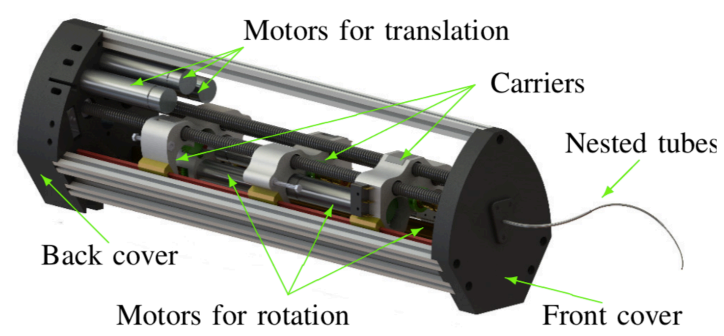
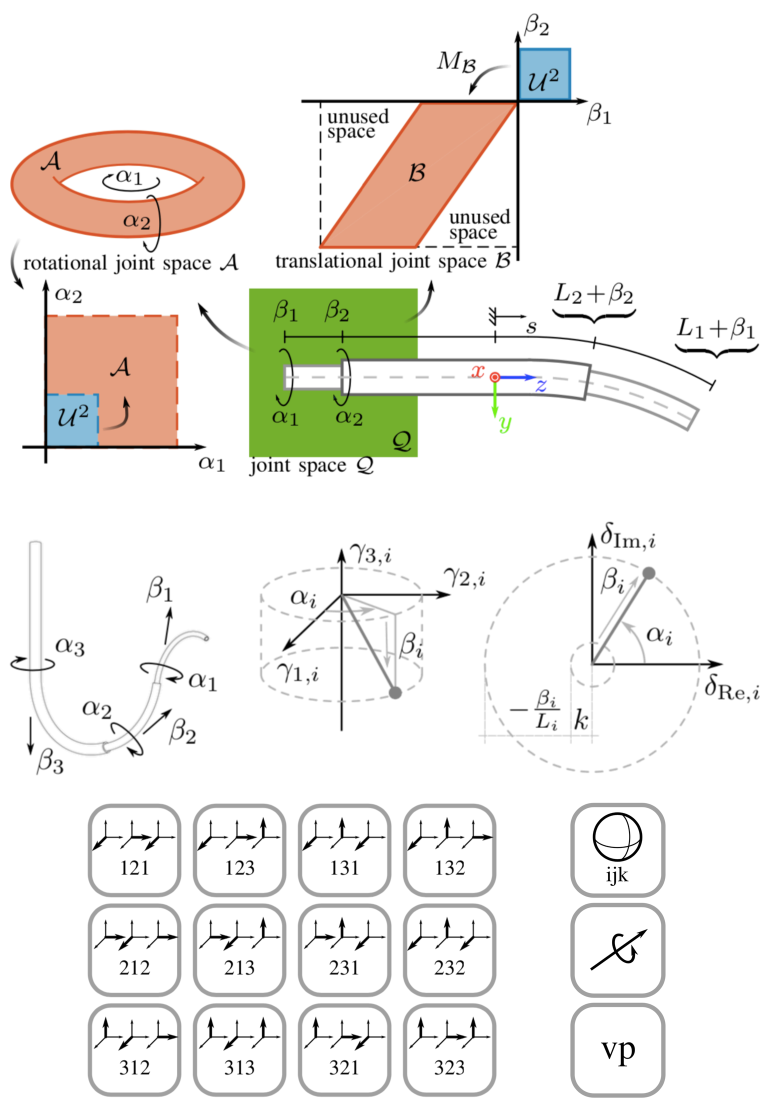


Fig. 4. CTCR prototype has three carriers each consisting a tube and a motor for rotation. Motors for translation are attached at the back cover. Six equal motors (DCX 16 L, Maxon Motor AG, OW, Switzerland) are controlled with a motion control board (DCM4163, Galil Motion Control, CA, USA).



Paper-ID 80

CRL website

UNIVERSITY OF TORONTO

Continuum Robotics Laboratory