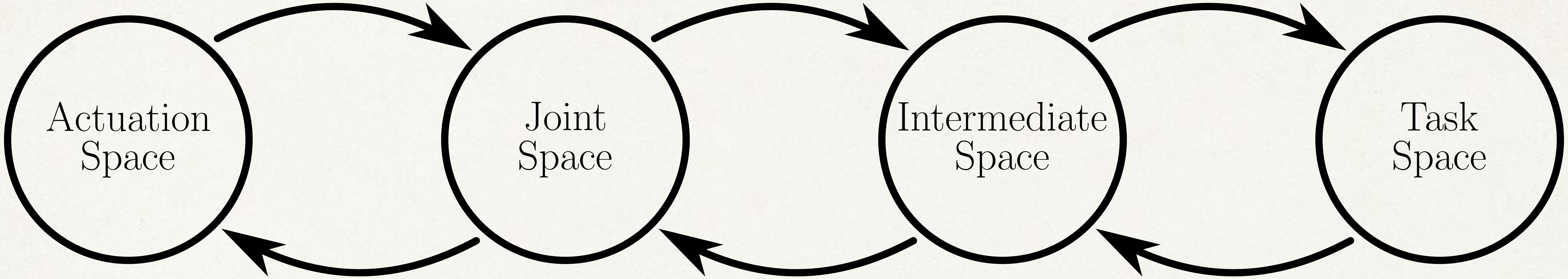
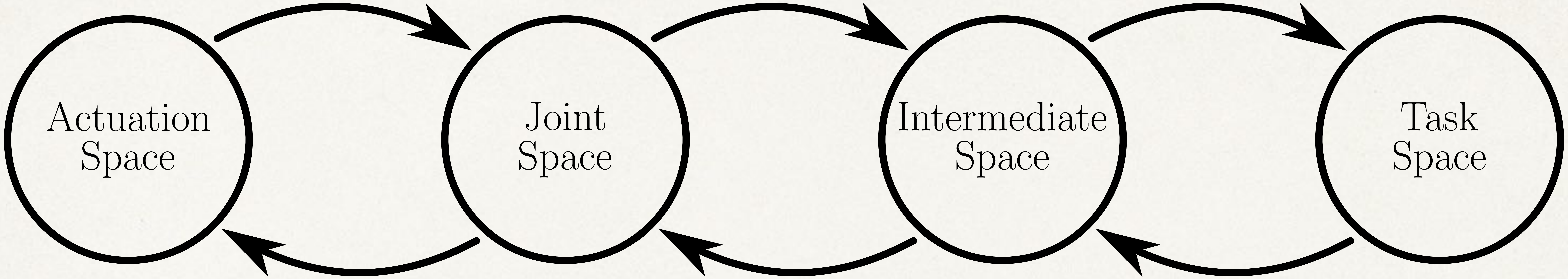
Clarke Transform: A Lingua Franca for Continuum Robotics

Benchmarking in Soft Robotics: Standardizing Data Collection and Evaluation for Actuation, Sensing, and Control

Reinhard M. Grassmann

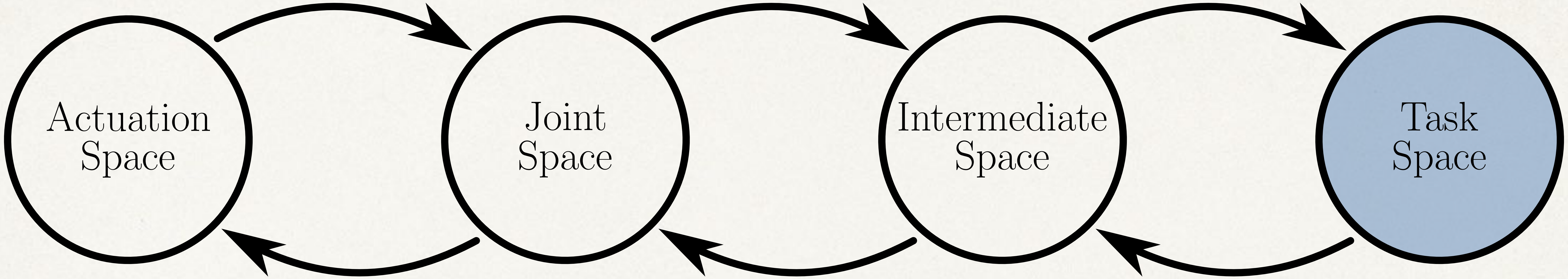


Robotics is the art of transformations between spaces

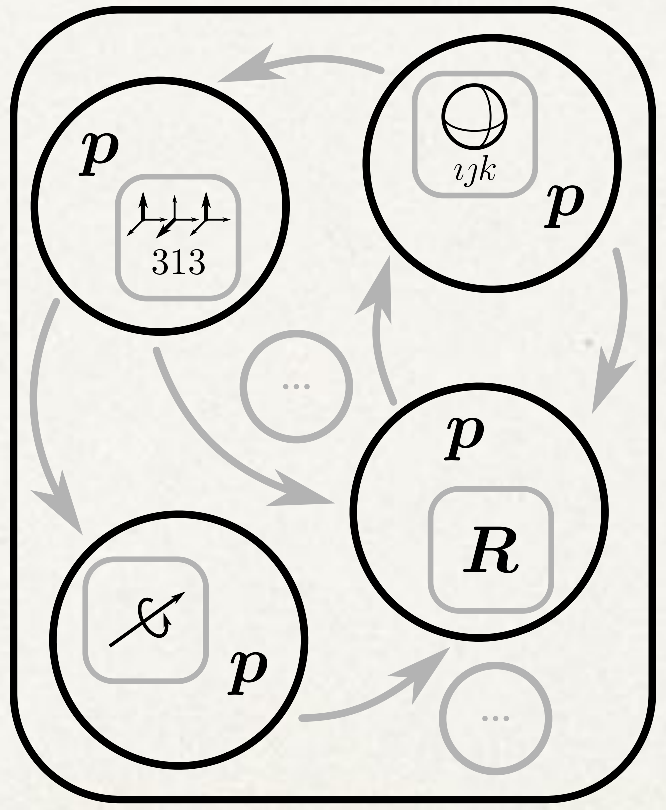


Robotics is the art of transformations between spaces

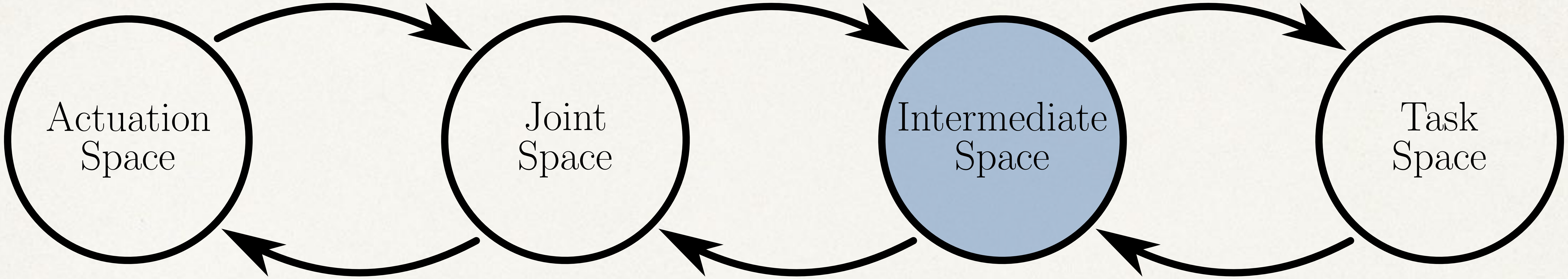
At the same time, each of the spaces has its own pitfalls.



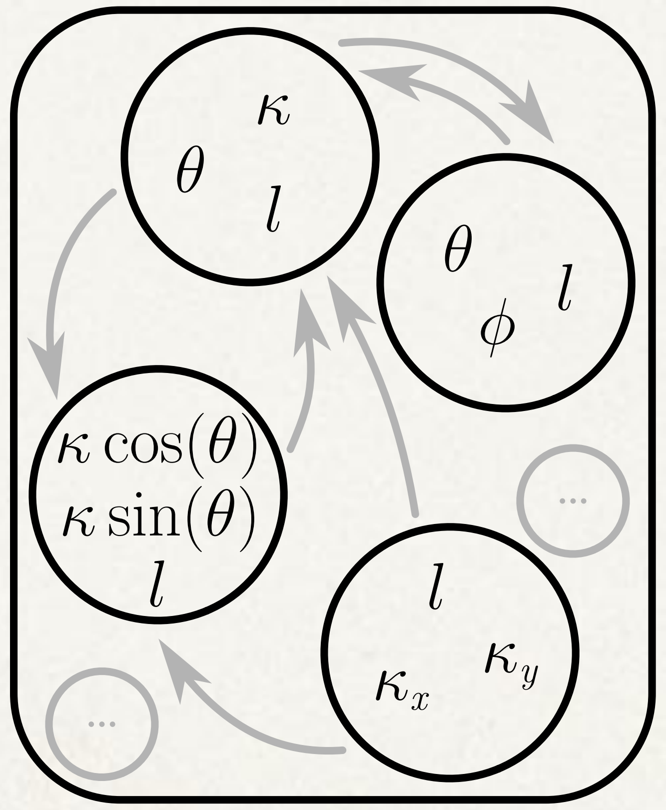
- Euler angles
  - 12 sets, singularities, …
- unit quaternions
  - double coverage, unit length
- rotation matrix
  - 9 values, 6 constrains



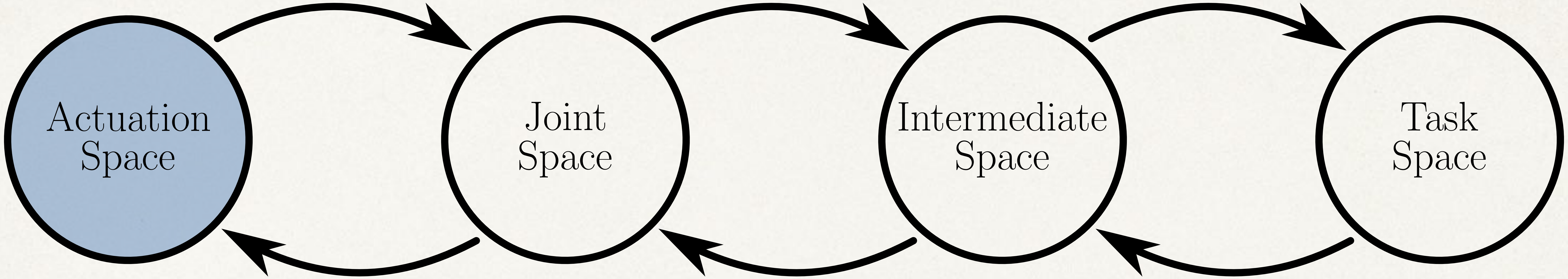
Task Space

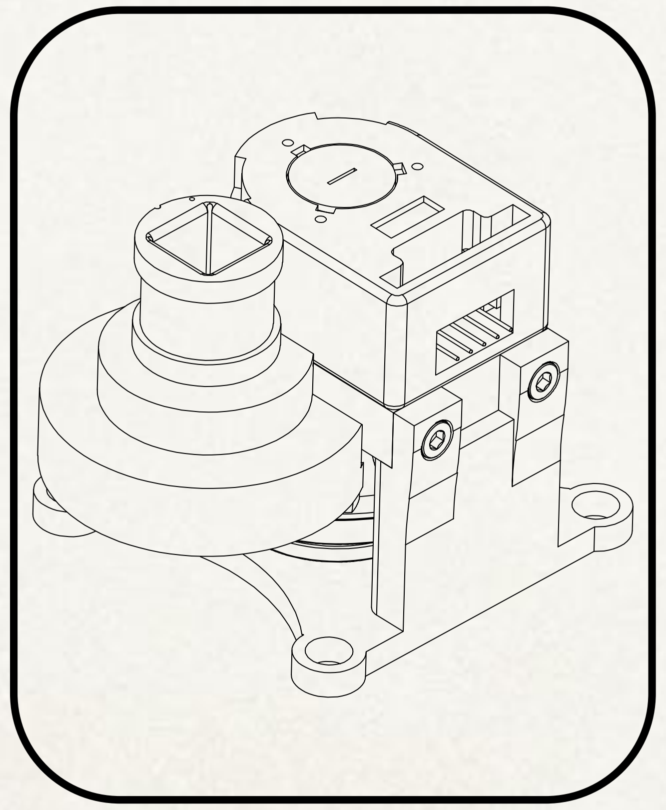


- curvature
  - coordinate singularities, …
- arc parameters
  - linear combination
  - non-linear combination
- intermediate space



Arc Space

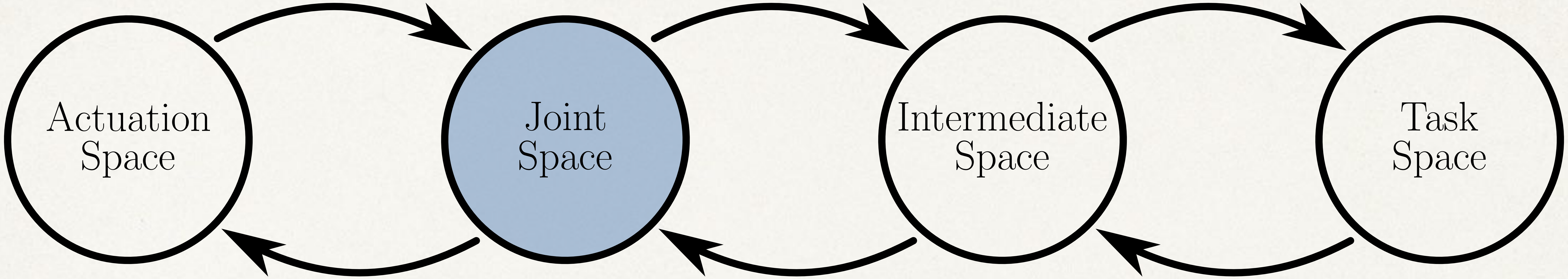


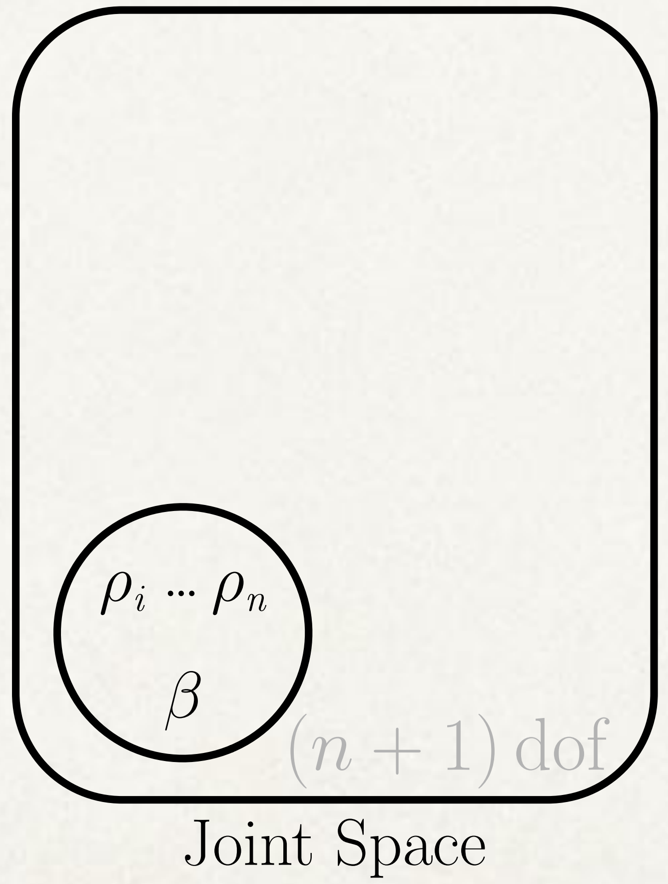


Actuation Space

- flow and effort of the system
  - mostly position controlled
  - a few torque controlled

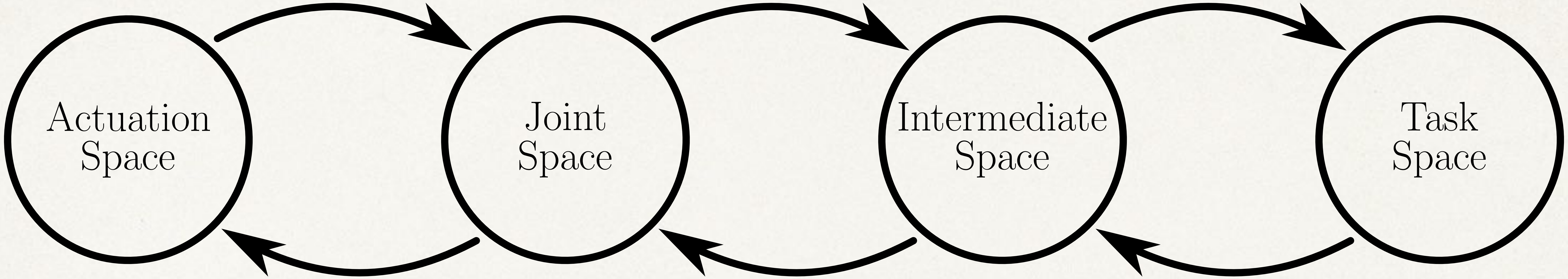
[**Grassmann** et al., Frontiers 2024] "Open Continuum Robotics — One Actuation Module to Create them All"

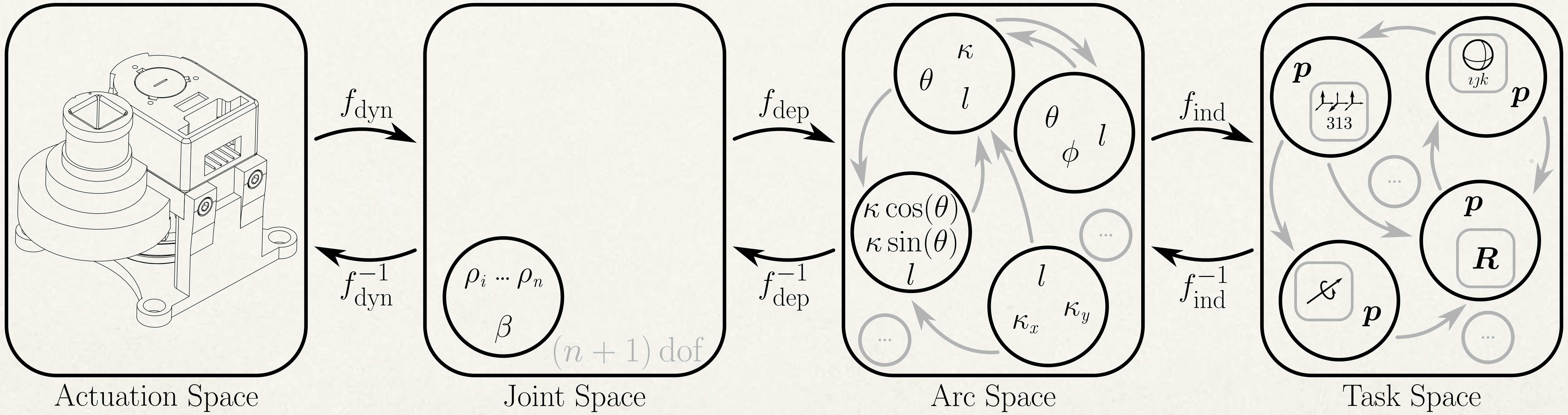




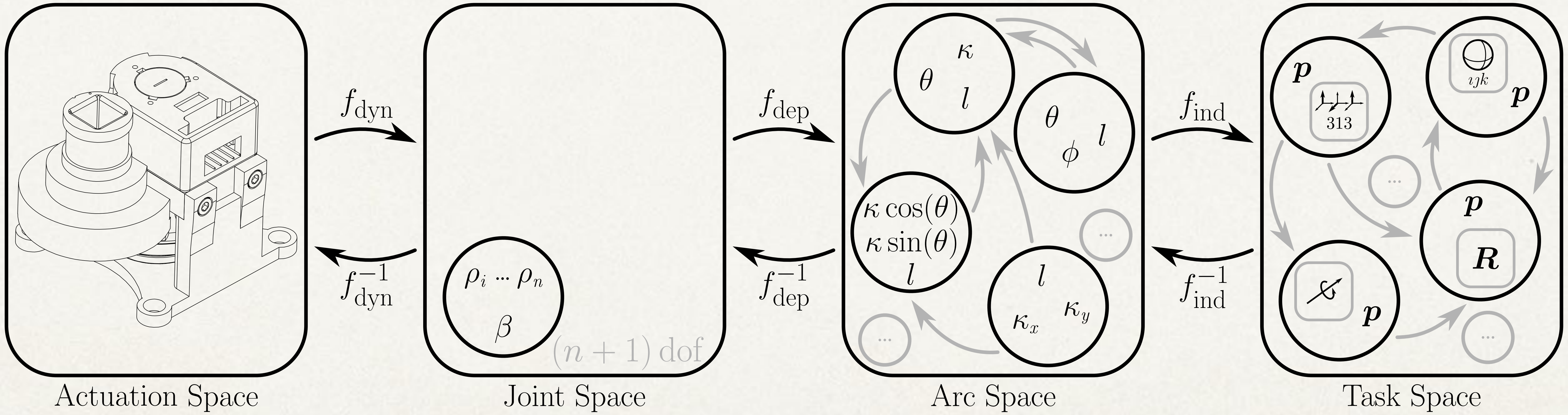
- improved state representations
  - limited to 3 and 4 joints
- missing manifold
  - overlooked constrains

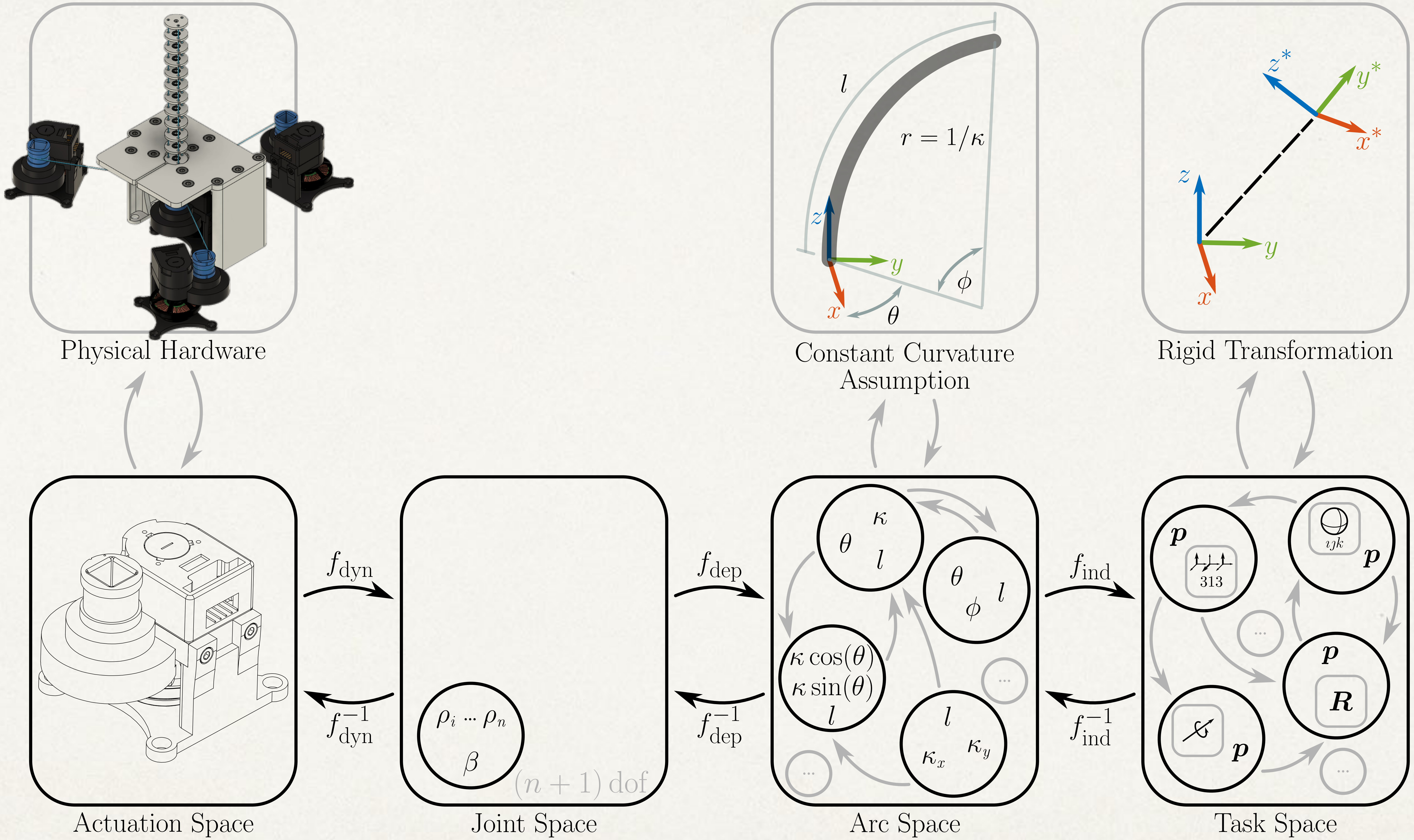
[Della Santina et al., RA-L 2020] "On an Improved State Parametrization for Soft Robots with Piecewise Constant Curvature and Its Use in Model-Based Control"
[Allen et al., RoboSoft 2020] "Closed-Form Non-Singular Constant-Curvature Continuum Manipulator Kinematics"
[Dian et al., Access 2022] "A Novel Disturbance-Rejection Control Framework for Cable-Driven Continuum Robots With Improved State Parameterization"

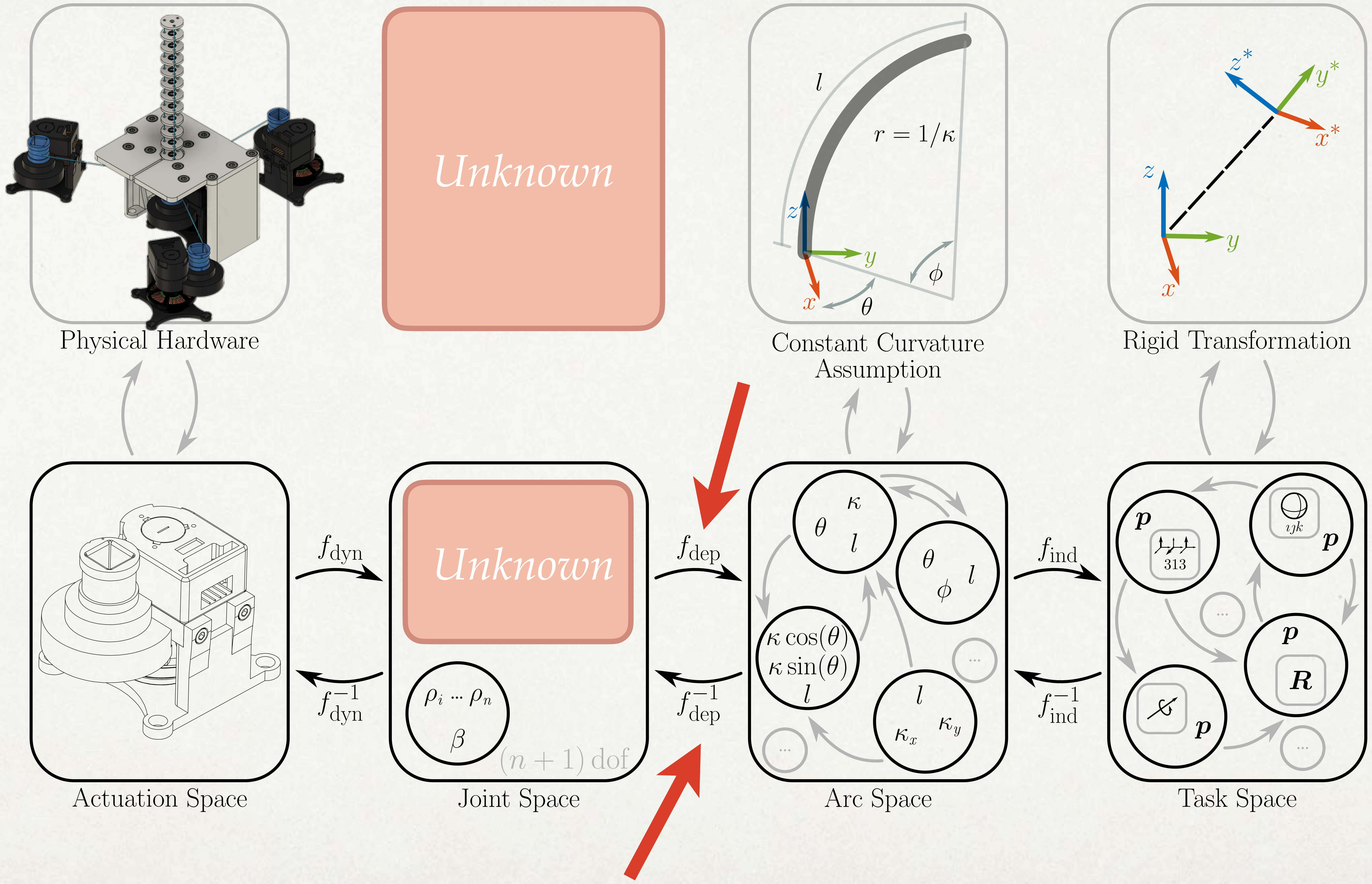

Actuation Space
Joint Space
Intermediate Space
Task Space


f_{dyn}
f_{dyn}^{-1}
$\rho_i \cdots \rho_n$
β
$(n+1)$ dof
f_{dep}
f_{dep}^{-1}
θ κ l
θ ϕ l
$\kappa\cos(\theta)$ $\kappa\sin(\theta)$ l
l κ_x κ_y
…
f_{ind}
f_{ind}^{-1}
$\boldsymbol{p}$
313
ijk
$\boldsymbol{R}$
Actuation Space
Joint Space
Arc Space
Task Space

We need to master this




Physical Hardware
Constant Curvature Assumption
Rigid Transformation
Actuation Space
Joint Space
Arc Space
Task Space
f_{dyn}
f_{dyn}^{-1}
f_{dep}
f_{dep}^{-1}
f_{ind}
f_{ind}^{-1}
$r = 1/\kappa$
$(n+1)$ dof
$\rho_i \cdots \rho_n$
β
$\kappa \cos(\theta)$
$\kappa \sin(\theta)$

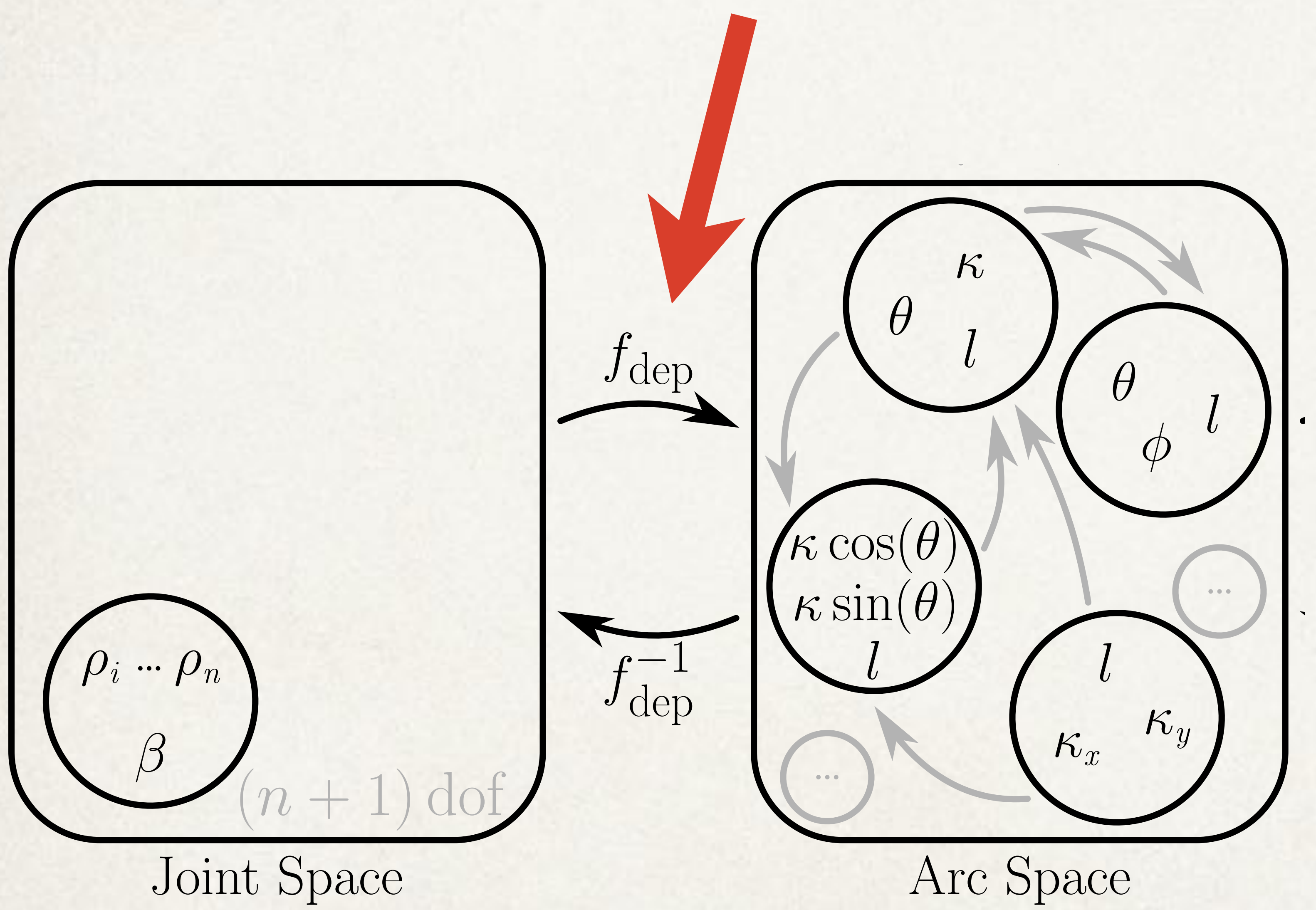


Physical Hardware
Unknown
Constant Curvature Assumption
Rigid Transformation
l
$r = 1/\kappa$
ϕ
θ
f_{dyn}
f_{dyn}^{-1}
Unknown
$\rho_i \dots \rho_n$
β
$(n+1)$ dof
f_{dep}
f_{dep}^{-1}
$\kappa\ \theta\ l$
$\theta\ \phi\ l$
$\kappa\cos(\theta)$
$\kappa\sin(\theta)$
l
$l\ \kappa_x\ \kappa_y$
f_{ind}
f_{ind}^{-1}
$\boldsymbol{p}$
313
$\boldsymbol{R}$
Actuation Space
Joint Space
Arc Space
Task Space

Ripple Effects

known for n=3 and n=4 actuators

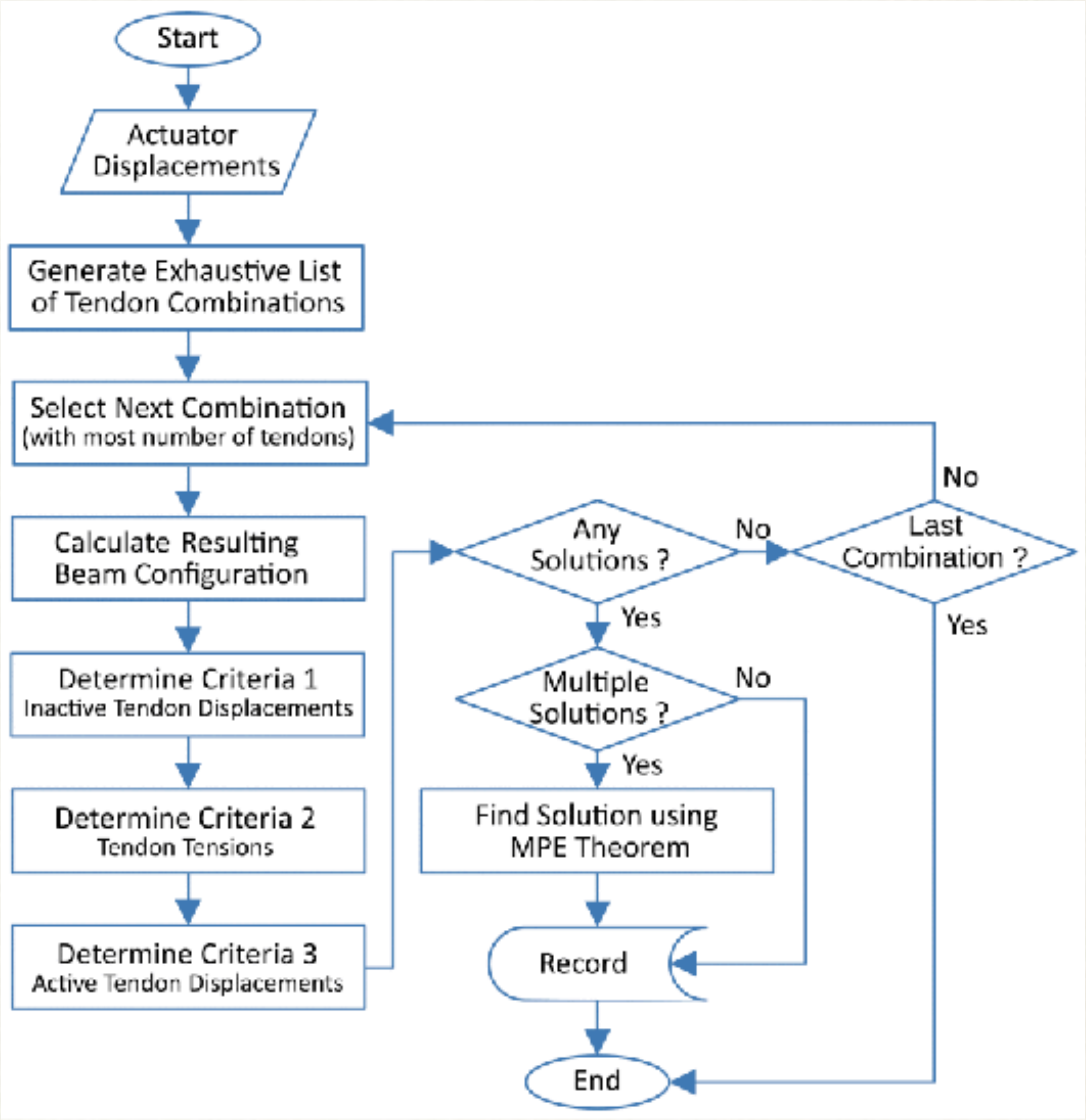
robot-dependent mapping for n actuators ?



the problem

```
input : Actuator displacements ([ΔL_{a_all}]), tendon locations ([α_all], [R_all]) and geometry and mechanical properties of
        beam and tendons
output: Active/Slack set of tendons and the resulting beam configuration ([θ, φ, L_c])
Algorithm SolveForwardKinematics ([ΔL_{a_all}], [α_all], [R_all], [properties])
    n ⟵ size([α_all], 1);
    Generate exhaustive list of n- to 1-tendon combinations;
    foreach j ∈ n do
        foreach i ∈ number of j-tendon combinations do
            θ_0 ⟵ 0;
            φ_0 ⟵ 0;
            L_{c_0} ⟵ L_0;
            [comb] ⟵ i-th j-tendon combinations;
            [θ, φ, L_c] = FindBeamConfig ([comb], [θ_0, φ_0, L_{c_0}], [ΔL_{a_all}], [α_all], [R_all]);
            Find resulting tendon displacements [ΔL_i]_res using Equation (2);
            S_1 ⟵ Any([ΔL_i]_res out of [comb] larger than the corresponding [ΔL_i]);
            Find resulting [comb] tendon tensions [F_i]_res using Equation (3);
            S_2 ⟵ Not(Any([F_i]_res < 0));
            Find resulting [comb] tendon stretches [δ_i]_res using Equation (4);
            Find resulting [comb] actuater displacements using [ΔL_{a_i}]_res = [ΔL_i]_res + [δ_i]_res;
            S_3 ⟵ Any([ΔL_{a_i}]_res of [comb] differs from the corresponding [ΔL_{a_i}]);
            Compute the beam potential energy using Equation (5);
            Record status parameters of S_1, S_2, and S_3, as well as the potential energy of this combination;
        [sol_index], [sol_comb] ⟵ Find(combinations with all S_1, S_2, and S_3 equal to 1);
        if size([sol_index], 1) = 1 then
            solution ⟵ [sol_comb];
            return solution and [θ, φ, L_c];
        else if size([sol_index], 1) > 1 then
            solution ⟵ Find([sol_comb] element with the least amount of potential energy);
            return solution and [θ, φ, L_c];
        else
            Proceed with the next list of (j − 1)-tendon combinations;
    return
Procedure FindBeamConfig ([comb], [θ_0, φ_0, L_{c_0}], [ΔL_{a_all}], [α_all])
    Set [ΔL_{a_i}] based on [comb] and [ΔL_{a_all}];
    Set [α_i] based on [comb] and [α_all];
    Set [R_i] based on [comb] and [R_all];
    Derive tendon displacements [ΔL_i] using Equation (2) ;
    Derive tendon tensions [F_i] using Equation (3);
    Derive tendon stretches [δ_i] using Equation (4);
    Define equations [Eq_i] = [ΔL_{a_i}] − [ΔL_i] − [δ_i];
    Find beam configuration [θ, φ, L_c] by numerically solving the set of Equations [Eq_i] = 0 based on the initial
      condition [θ_0, φ_0, L_{c_0}];
    return [θ, φ, L_c];
```

[Dalvand et al., Access 2022]

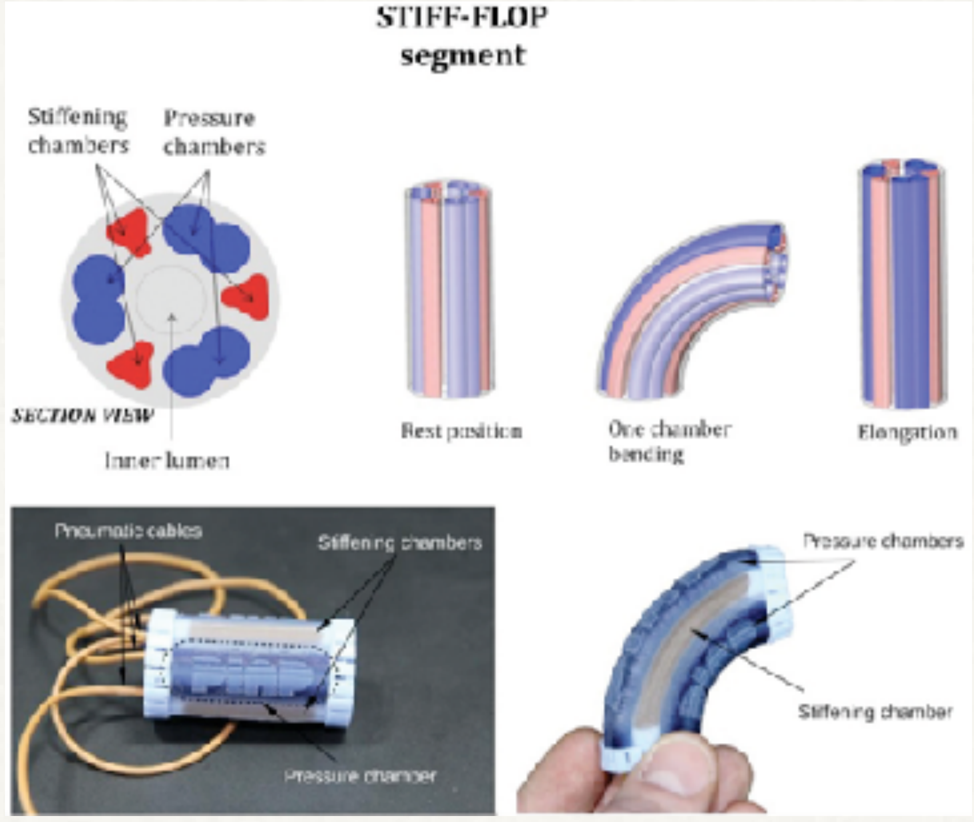


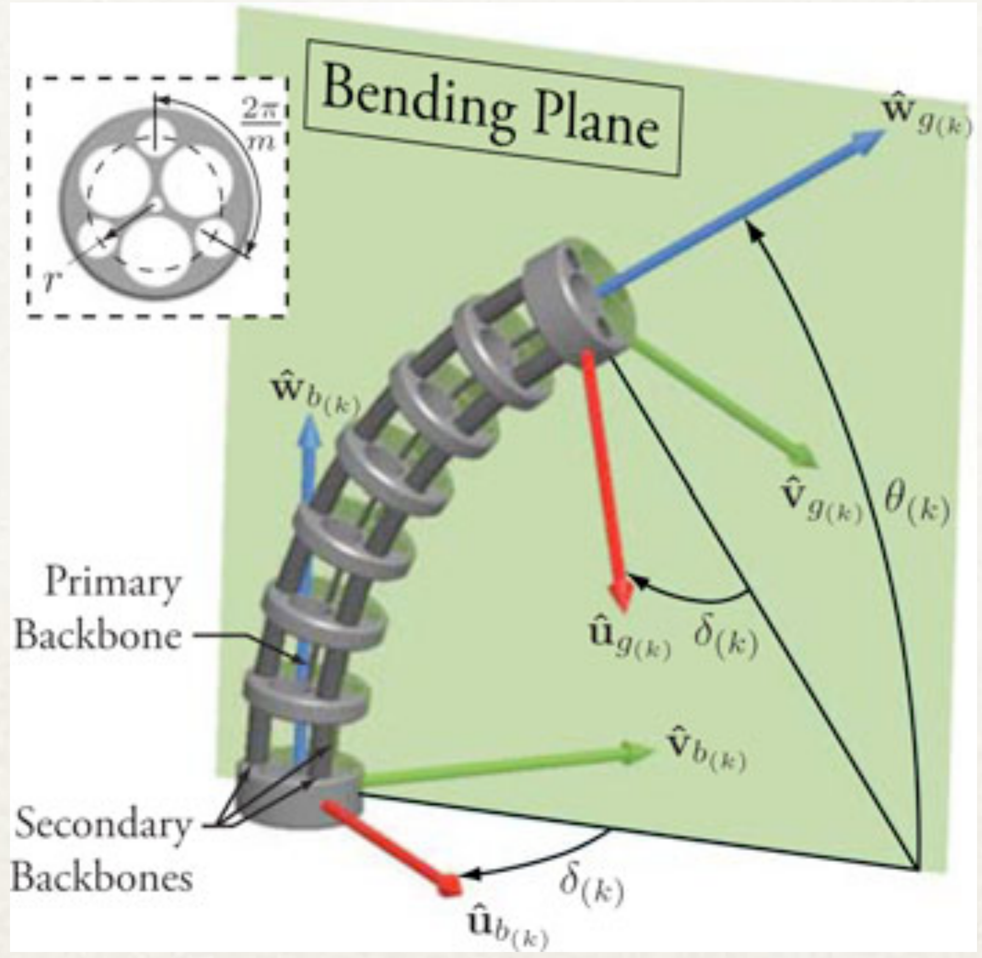
[Dalvand et al., Access 2022]

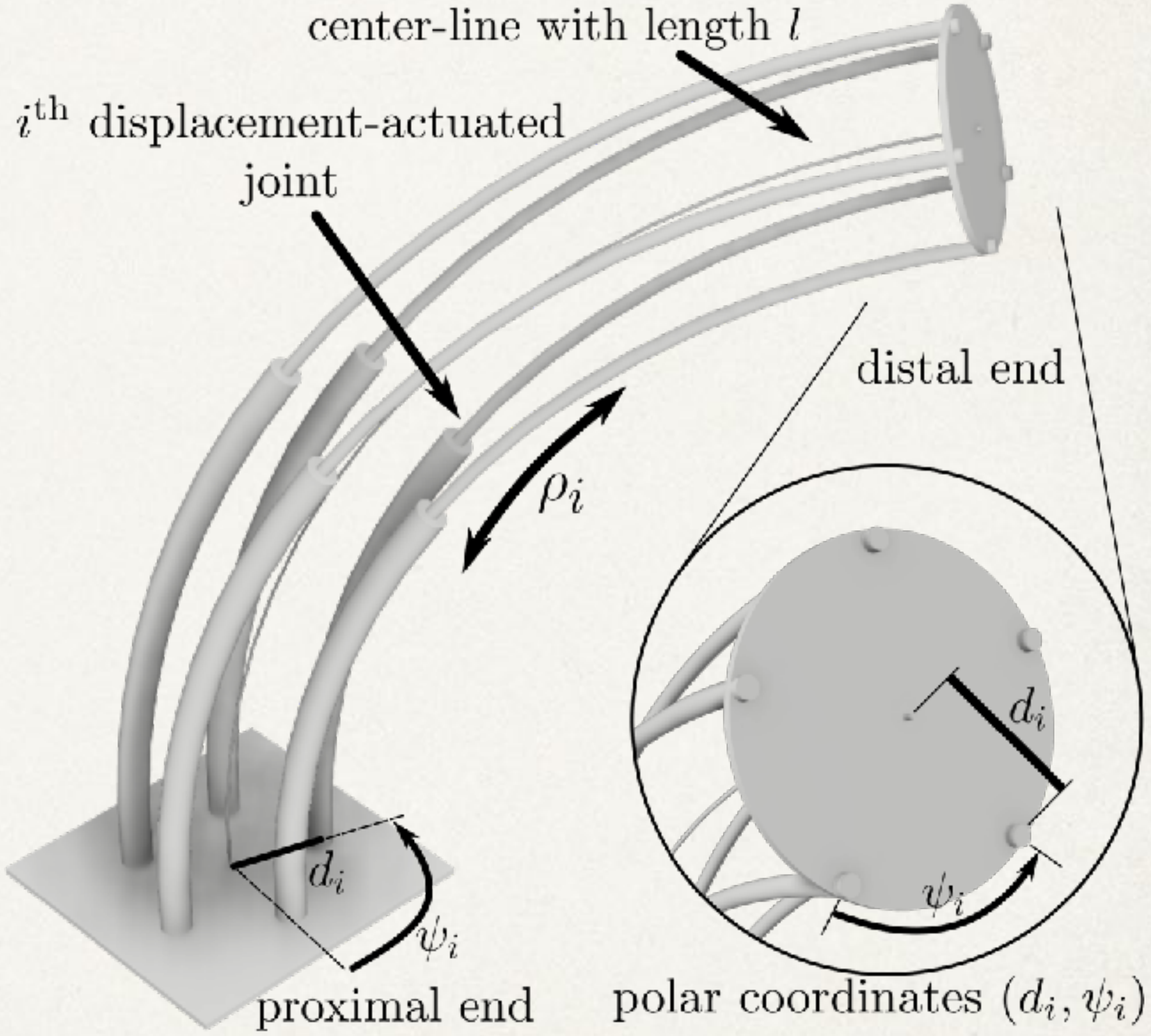
current state-of-the-art approach

[Dalvand et al., Access 2022] "General Forward Kinematics for Tendon-Driven Continuum Robots"

Abstraction — DACR







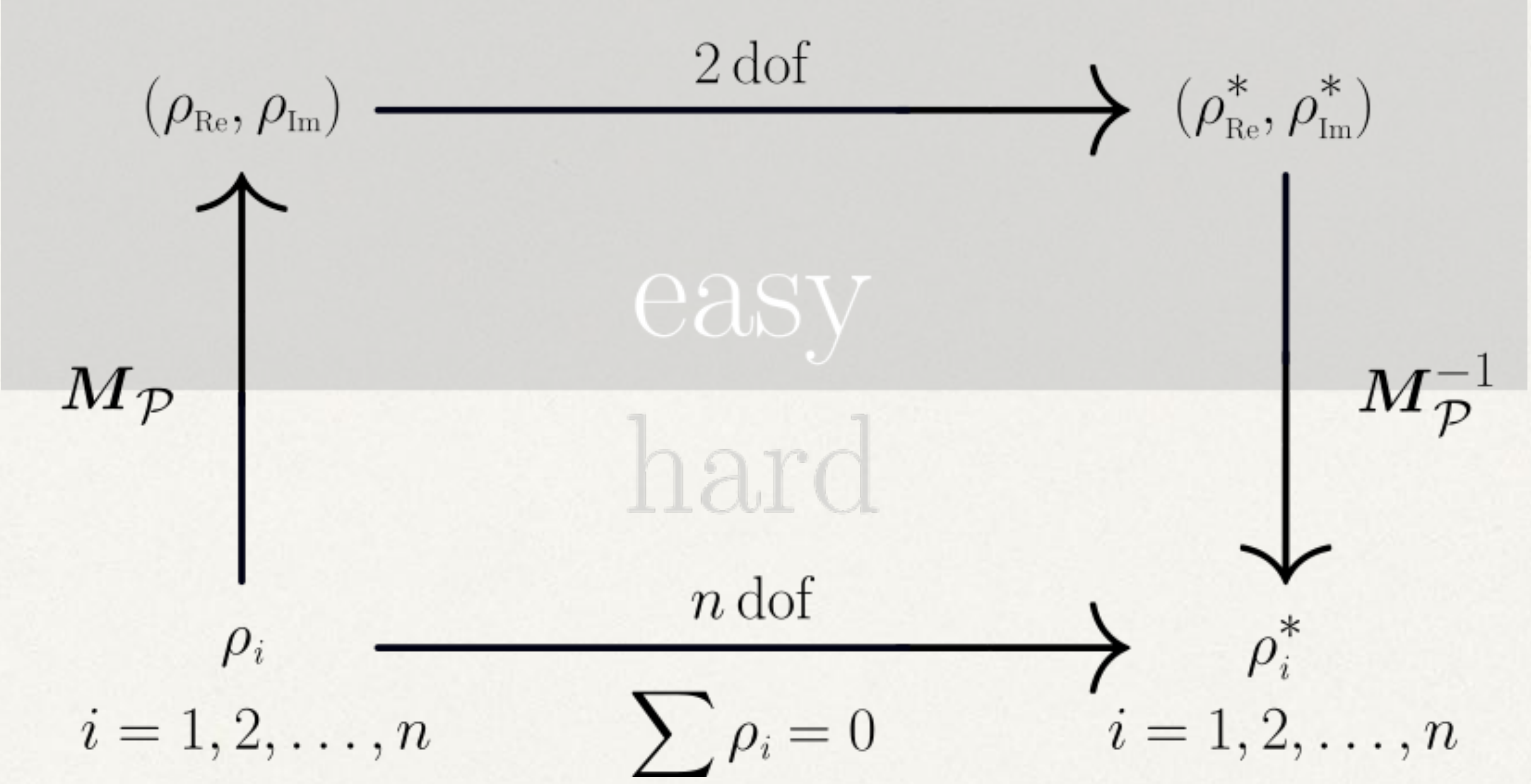
Displacement-Actuated Continuum Robot

[**Grassmann** et al., arXiv (under review)] "Clarke Transform — A Fundamental Tool for Continuum Robotics"
[**Grassmann** & Burgner-Kahrs, arXiv (under review)] "Displacement-Actuated Continuum Robot: A Joint Space Abstraction"

Clarke Transform and Clarke Coordinates

Forward

$$\overline{\boldsymbol{\rho}} = \boldsymbol{M}_{\mathcal{P}} \boldsymbol{\rho} \in \mathbb{R}^2$$



Backward

$$\boldsymbol{\rho} = \boldsymbol{M}_{\mathcal{P}}^{-1} \overline{\boldsymbol{\rho}} \in \mathbb{R}^n$$

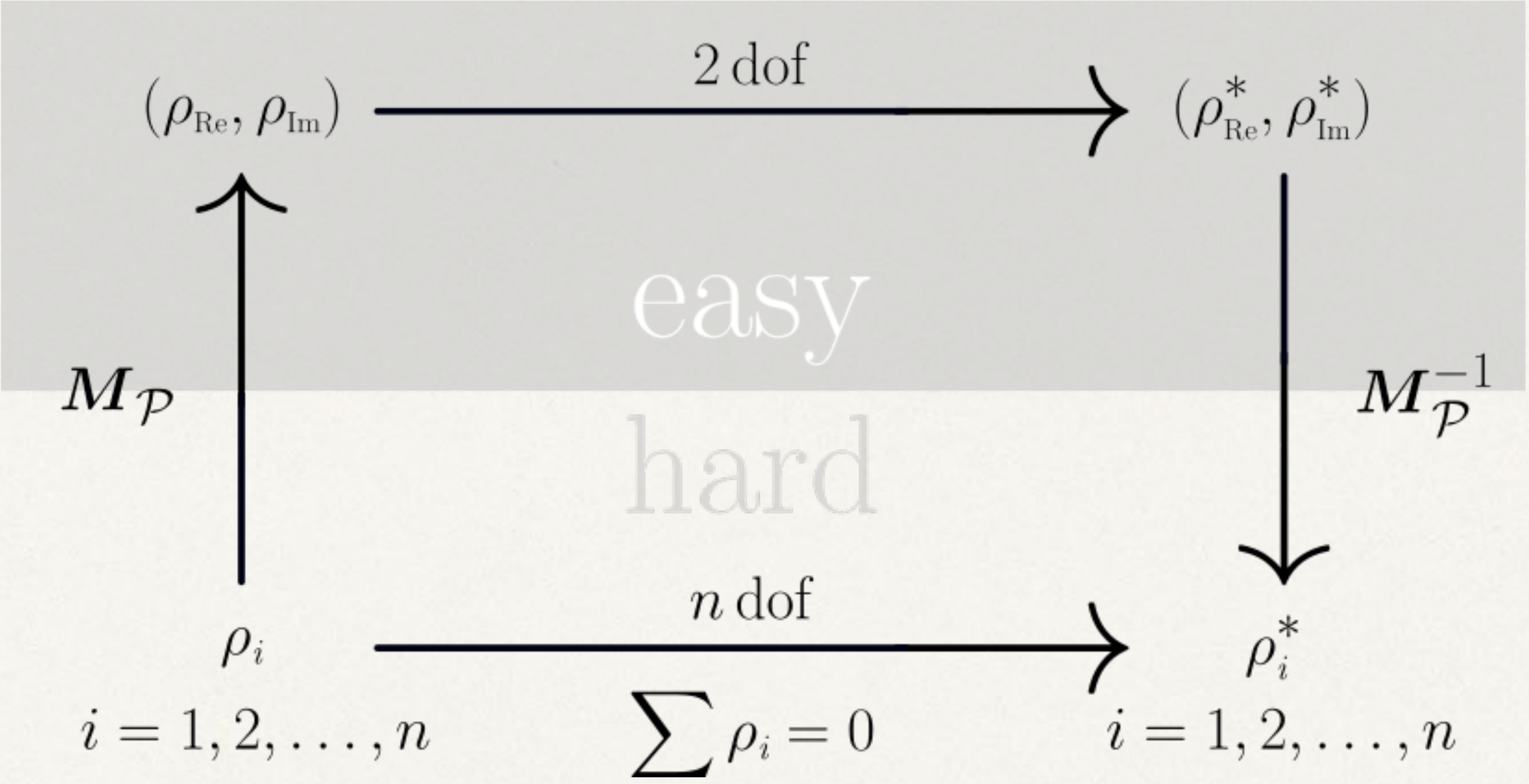
Clarke Transform
(manuscript on arXiv)

[**Grassmann** et al., arXiv (under review)] "Clarke Transform — A Fundamental Tool for Continuum Robotics"

Clarke Transform and Clarke Coordinates

Forward

$$\overline{\boldsymbol{\rho}} = \boldsymbol{M}_{\mathcal{P}} \boldsymbol{\rho} \in \mathbb{R}^2$$



Backward

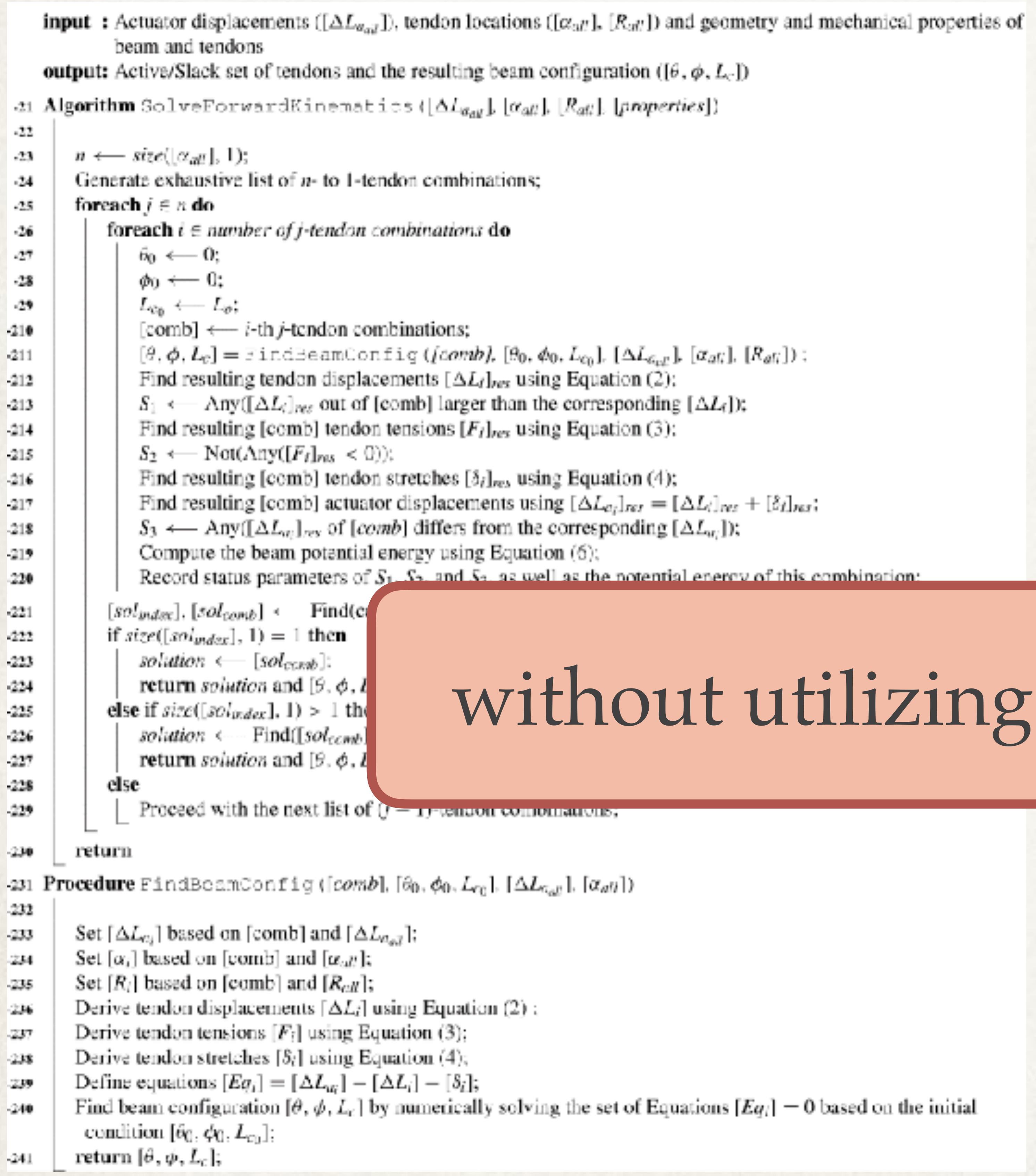
$$\boldsymbol{\rho} = \boldsymbol{M}_{\mathcal{P}}^{-1} \overline{\boldsymbol{\rho}} \in \mathbb{R}^n$$

Clarke Transform provide solution that are exact, closed-formed, and interpretable.

Clarke Transform
(manuscript on arXiv)

[**Grassmann** et al., arXiv (under review)] "Clarke Transform — A Fundamental Tool for Continuum Robotics"

Robot-Dependent Mapping

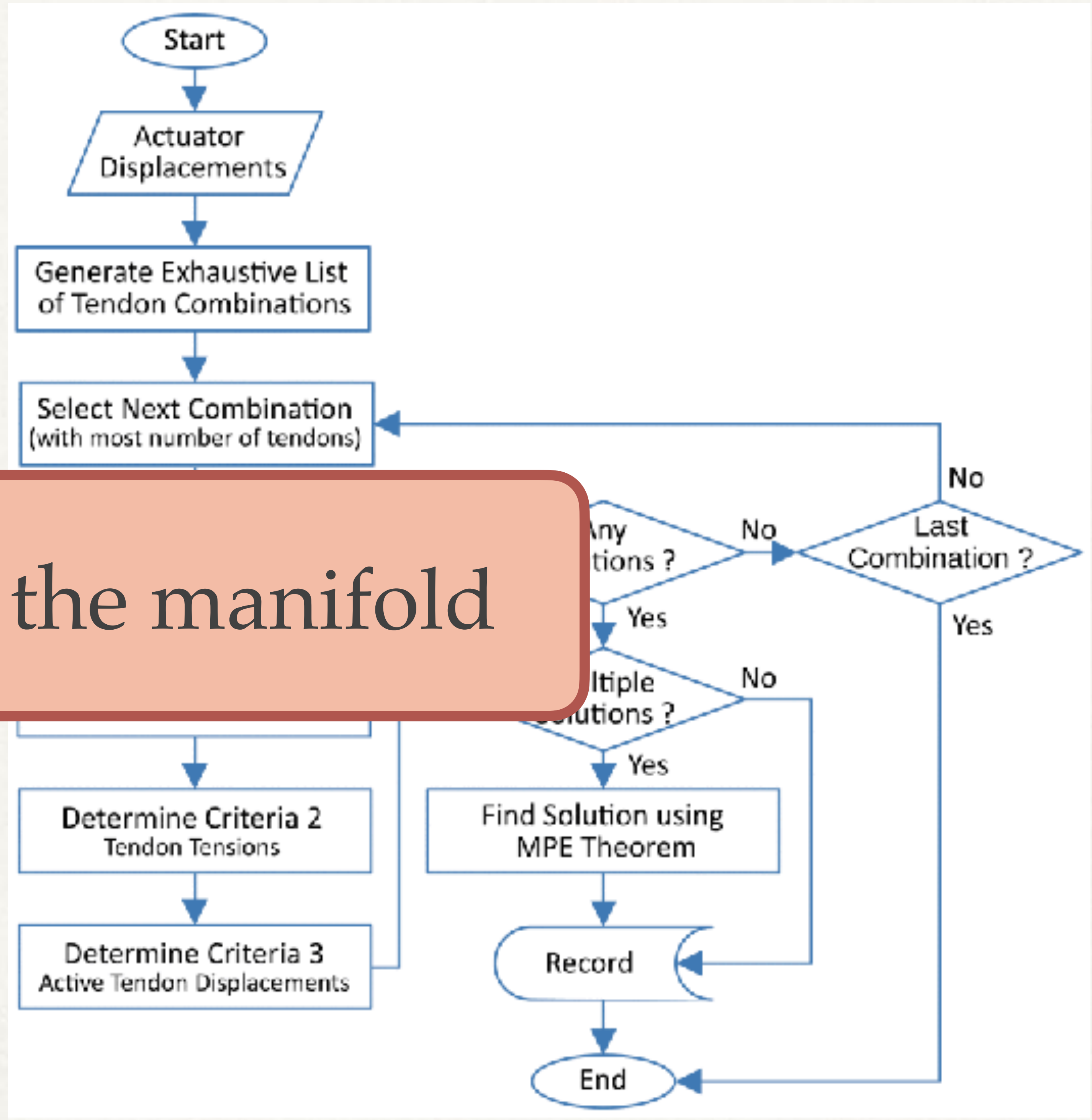


input : Actuator displacements ($[\Delta L_{a_{all}}]$), tendon locations ($[\alpha_{all}]$, $[R_{all}]$) and geometry and mechanical properties of beam and tendons
output: Active/Slack set of tendons and the resulting beam configuration ($[\theta, \phi, L_c]$)

Algorithm SolveForwardKinematics ($[\Delta L_{a_{all}}]$, $[\alpha_{all}]$, $[R_{all}]$, $[properties]$)
$n \longleftarrow size([\alpha_{all}], 1)$;
Generate exhaustive list of n- to 1-tendon combinations;
foreach $j \in n$ **do**
foreach $i \in$ *number of j-tendon combinations* **do**
$\theta_0 \longleftarrow 0$;
$\phi_0 \longleftarrow 0$;
$L_{c_0} \longleftarrow L_0$;
[comb] $\longleftarrow$ i-th j-tendon combinations;
$[\theta, \phi, L_c]$ = FindBeamConfig ($[comb]$, $[\theta_0, \phi_0, L_{c_0}]$, $[\Delta L_{a_{all}}]$, $[\alpha_{all}]$, $[R_{all}]$) ;
Find resulting tendon displacements $[\Delta L_t]_{res}$ using Equation (2);
$S_1 \longleftarrow$ Any($[\Delta L_t]_{res}$ out of [comb] larger than the corresponding $[\Delta L_t]$);
Find resulting [comb] tendon tensions $[F_t]_{res}$ using Equation (3);
$S_2 \longleftarrow$ Not(Any($[F_t]_{res} < 0$));
Find resulting [comb] tendon stretches $[\delta_t]_{res}$ using Equation (4);
Find resulting [comb] actuator displacements using $[\Delta L_{a_i}]_{res} = [\Delta L_t]_{res} + [\delta_t]_{res}$;
$S_3 \longleftarrow$ Any($[\Delta L_{a_i}]_{res}$ of $[comb]$ differs from the corresponding $[\Delta L_{a_i}]$);
Compute the beam potential energy using Equation (6);
Record status parameters of S_1, S_2, and S_3, as well as the potential energy of this combination;
$[sol_{index}]$, $[sol_{comb}]$ $\longleftarrow$ Find(c
if $size([sol_{index}], 1) = 1$ **then**
solution $\longleftarrow$ $[sol_{comb}]$;
return *solution* and $[\theta, \phi,$
else if $size([sol_{index}], 1) > 1$ the
solution $\longleftarrow$ Find($[sol_{comb}$
return *solution* and $[\theta, \phi,$
else
Proceed with the next list of
return

Procedure FindBeamConfig ($[comb]$, $[\theta_0, \phi_0, L_{c_0}]$, $[\Delta L_{a_{all}}]$, $[\alpha_{all}]$)
Set $[\Delta L_{a_i}]$ based on [comb] and $[\Delta L_{a_{all}}]$;
Set $[\alpha_i]$ based on [comb] and $[\alpha_{all}]$;
Set $[R_i]$ based on [comb] and $[R_{all}]$;
Derive tendon displacements $[\Delta L_i]$ using Equation (2) ;
Derive tendon tensions $[F_i]$ using Equation (3);
Derive tendon stretches $[\delta_i]$ using Equation (4);
Define equations $[Eq_i] = [\Delta L_{a_i}] - [\Delta L_i] - [\delta_i]$;
Find beam configuration $[\theta, \phi, L_c]$ by numerically solving the set of Equations $[Eq_i] = 0$ based on the initial condition $[\theta_0, \phi_0, L_{c_0}]$;
return $[\theta, \phi, L_c]$;

[Dalvand et al., Access 2022]

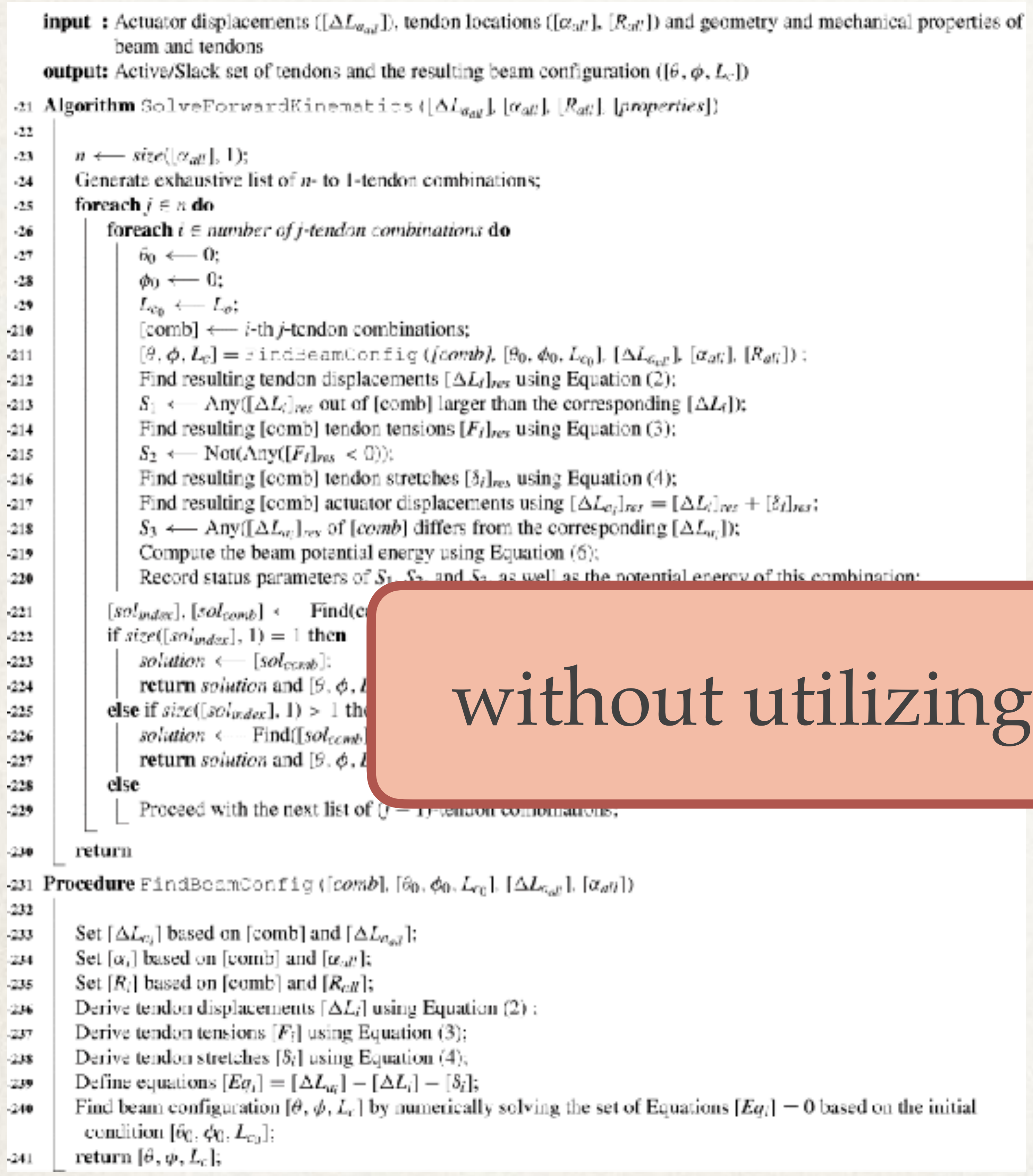


[Dalvand et al., Access 2022]

without utilizing the manifold

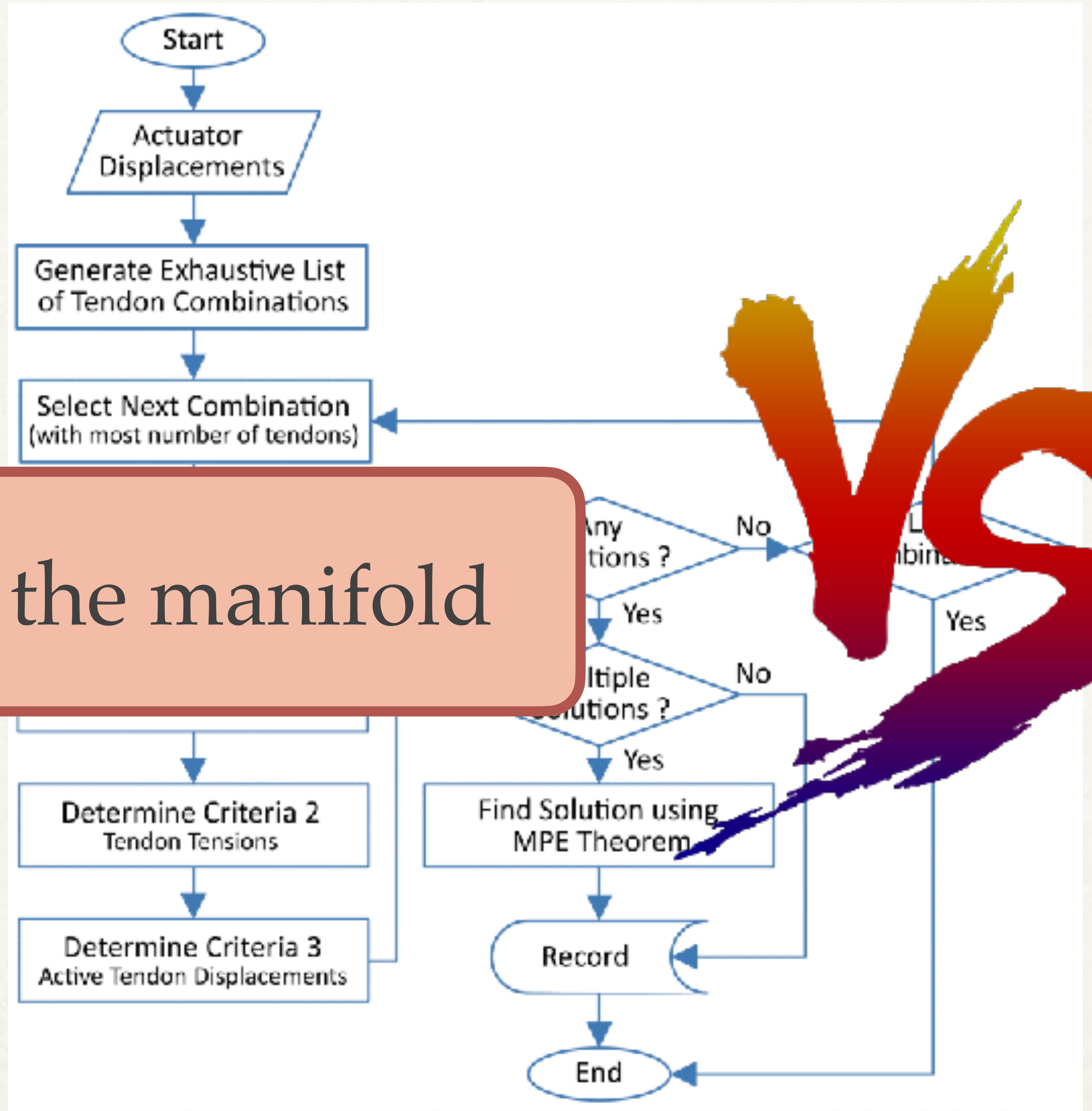
[Dalvand et al., Access 2022] “General Forward Kinematics for Tendon-Driven Continuum Robots”
[**Grassmann** et al., arXiv (under review)] “Clarke Transform — A Fundamental Tool for Continuum Robotics”

Robot-Dependent Mapping


input : Actuator displacements ($[\Delta L_{a_{all}}]$), tendon locations ($[\alpha_{all}]$, $[R_{all}]$) and geometry and mechanical properties of beam and tendons
output: Active/Slack set of tendons and the resulting beam configuration ($[\theta, \phi, L_c]$)

```
Algorithm SolveForwardKinematics([ΔL_a_all], [α_all], [R_all], [properties])
    n ⟵ size([α_all], 1);
    Generate exhaustive list of n- to 1-tendon combinations;
    foreach j ∈ n do
        foreach i ∈ number of j-tendon combinations do
            θ_0 ⟵ 0;
            φ_0 ⟵ 0;
            L_c0 ⟵ L_o;
            [comb] ⟵ i-th j-tendon combinations;
            [θ, φ, L_c] = FindBeamConfig([comb], [θ_0, φ_0, L_c0], [ΔL_a_all], [α_all], [R_all]);
            Find resulting tendon displacements [ΔL_t]_res using Equation (2);
            S_1 ⟵ Any([ΔL_t]_res out of [comb] larger than the corresponding [ΔL_t]);
            Find resulting [comb] tendon tensions [F_t]_res using Equation (3);
            S_2 ⟵ Not(Any([F_t]_res < 0));
            Find resulting [comb] tendon stretches [δ_t]_res using Equation (4);
            Find resulting [comb] actuator displacements using [ΔL_a_t]_res = [ΔL_t]_res + [δ_t]_res;
            S_3 ⟵ Any([ΔL_a_t]_res of [comb] differs from the corresponding [ΔL_a_t]);
            Compute the beam potential energy using Equation (6);
            Record status parameters of S_1, S_2, and S_3, as well as the potential energy of this combination;
        [sol_index], [sol_comb] ⟵ Find(c
        if size([sol_index], 1) = 1 then
            solution ⟵ [sol_comb];
            return solution and [θ, φ, L
        else if size([sol_index], 1) > 1 the
            solution ⟵ Find([sol_comb]
            return solution and [θ, φ, L
        else
            Proceed with the next list of (j − 1)-tendon combinations;
    return
Procedure FindBeamConfig([comb], [θ_0, φ_0, L_c0], [ΔL_a_all], [α_all])
    Set [ΔL_a_t] based on [comb] and [ΔL_a_all];
    Set [α_t] based on [comb] and [α_all];
    Set [R_t] based on [comb] and [R_all];
    Derive tendon displacements [ΔL_t] using Equation (2);
    Derive tendon tensions [F_t] using Equation (3);
    Derive tendon stretches [δ_t] using Equation (4);
    Define equations [Eq_t] = [ΔL_a_t] − [ΔL_t] − [δ_t];
    Find beam configuration [θ, φ, L_c] by numerically solving the set of Equations [Eq_t] = 0 based on the initial
     condition [θ_0, φ_0, L_c0];
    return [θ, φ, L_c];
```

without utilizing the manifold



VS

$$\begin{bmatrix} \kappa \cos(\theta) \\ \kappa \sin(\theta) \end{bmatrix} = \underbrace{1/l}_{\text{removes } l} \overbrace{\boldsymbol{M}_{\mathcal{P}}}^{\text{removes } \psi_i} \underbrace{\operatorname{diag}(1/d_i)}_{\text{removes } d_i} \boldsymbol{\rho}$$

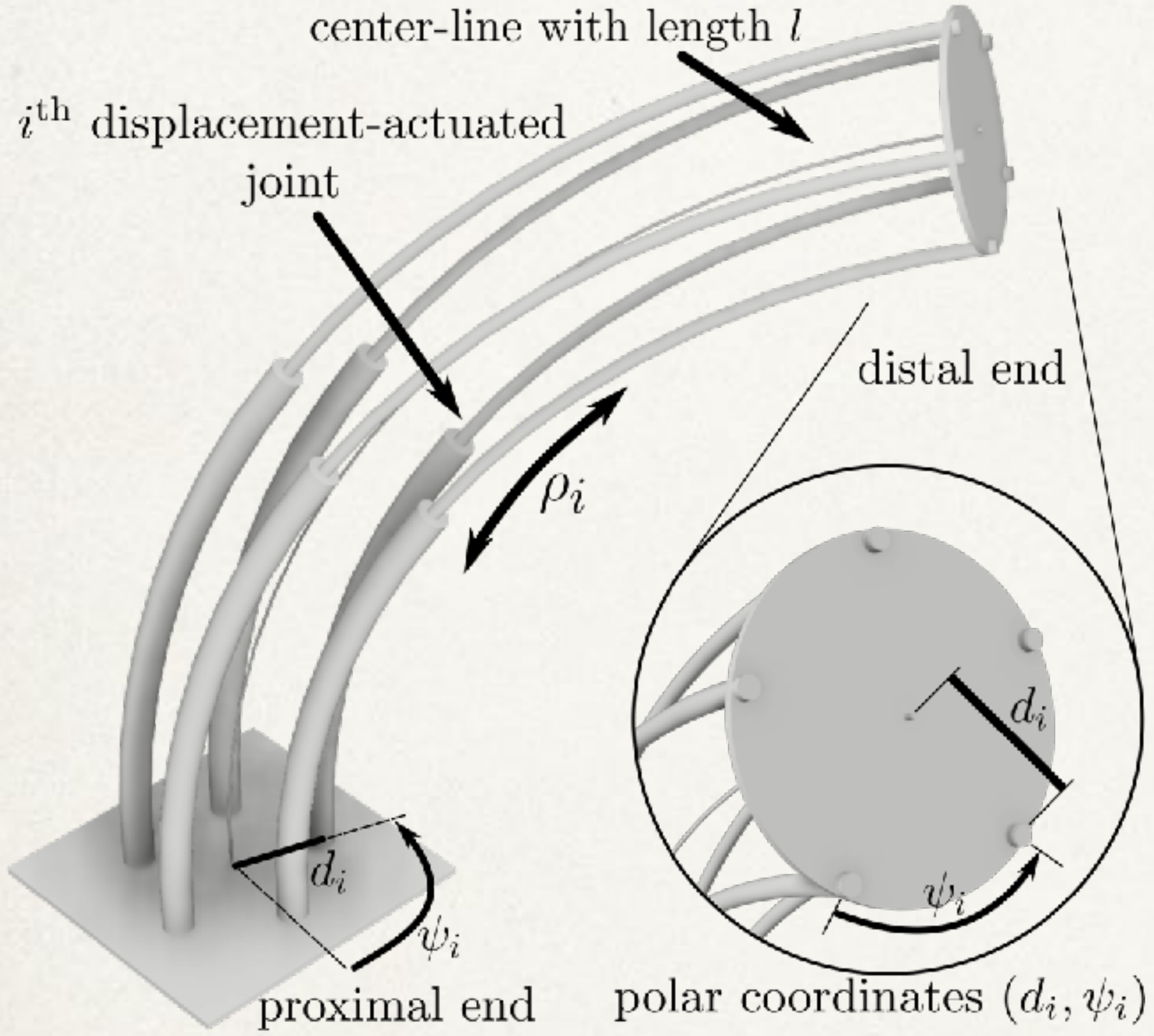
utilizing the manifold

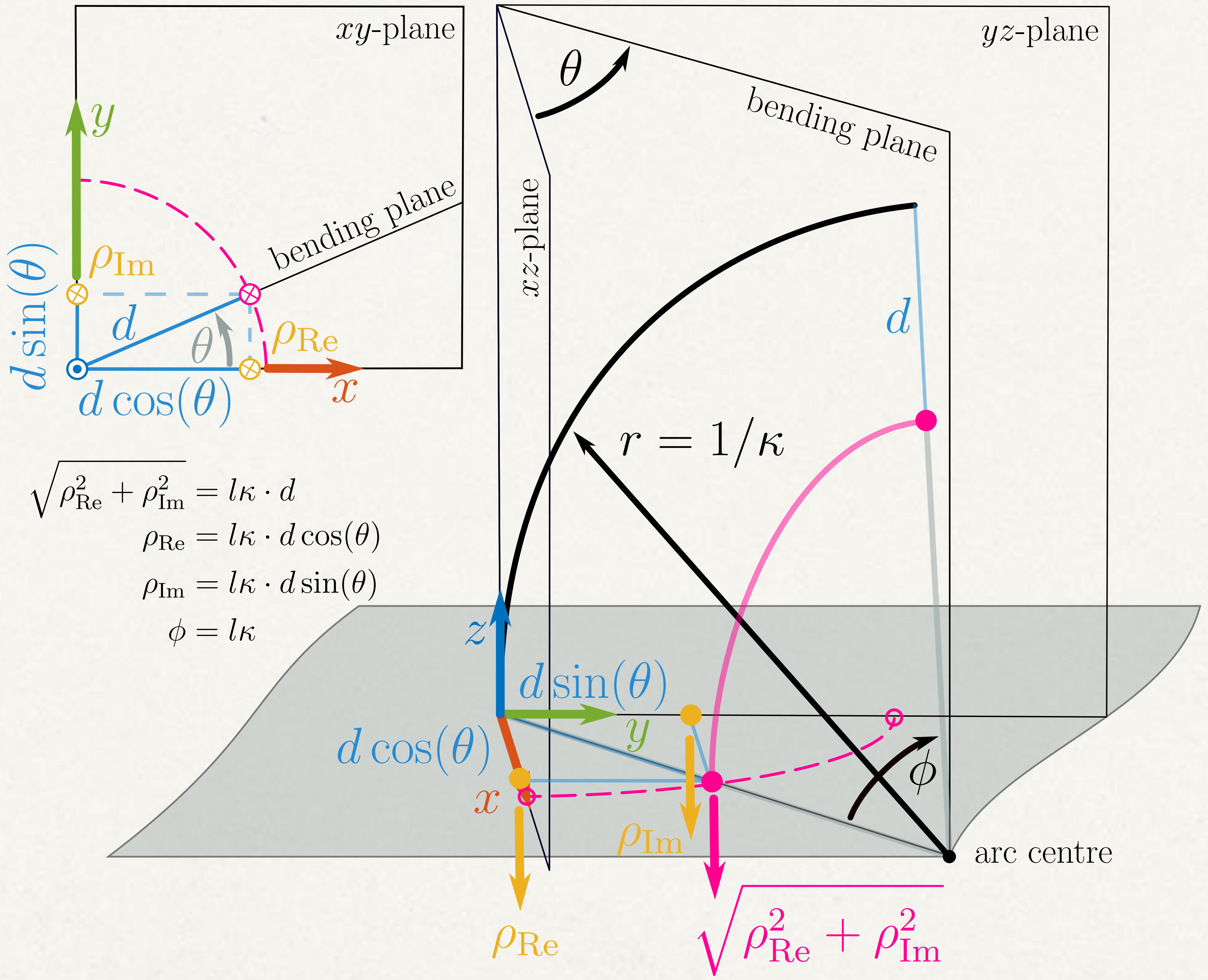
[Dalvand et al., Access 2022]

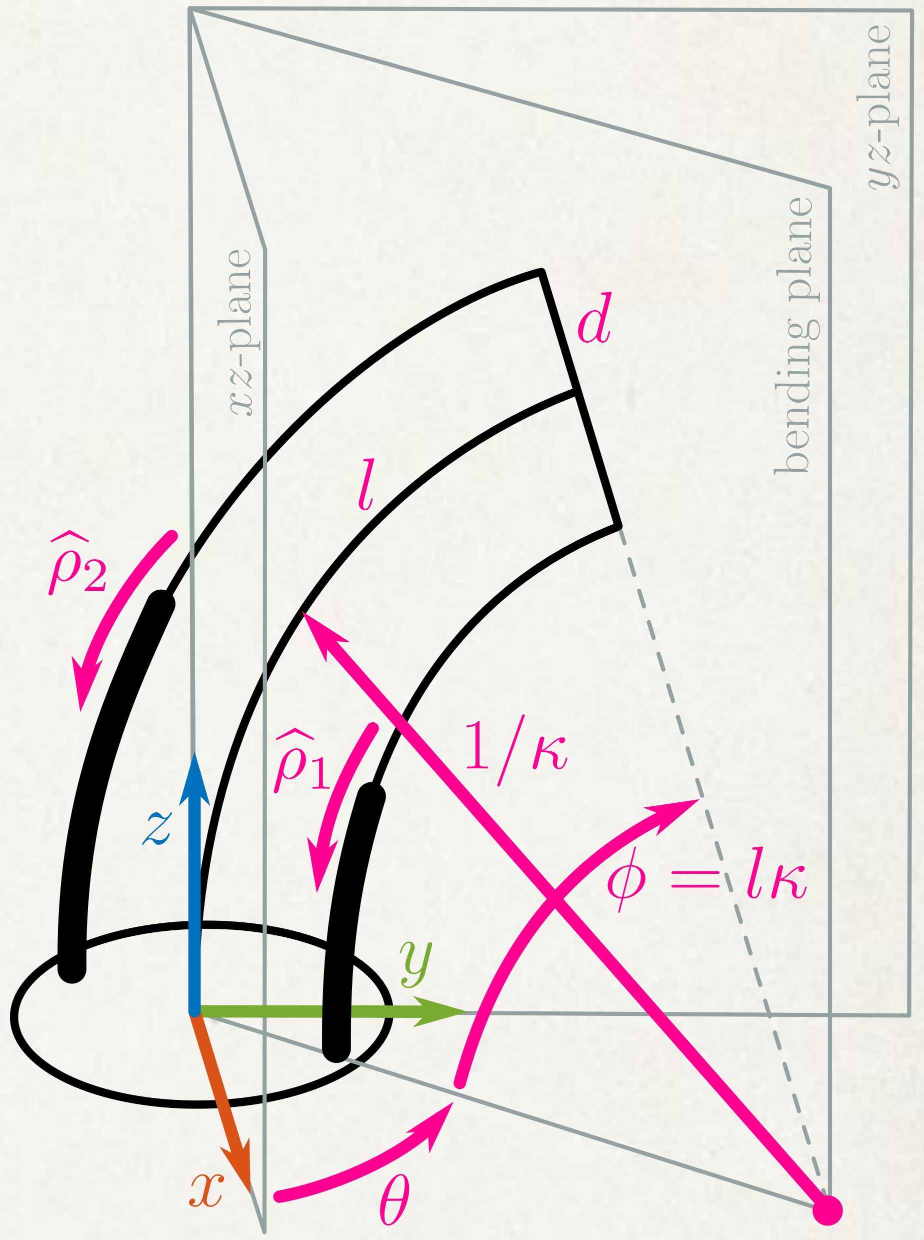
[Dalvand et al., Access 2022]

[Dalvand et al., Access 2022] “General Forward Kinematics for Tendon-Driven Continuum Robots”
[**Grassmann** et al., arXiv (under review)] “Clarke Transform — A Fundamental Tool for Continuum Robotics”

Interpretable

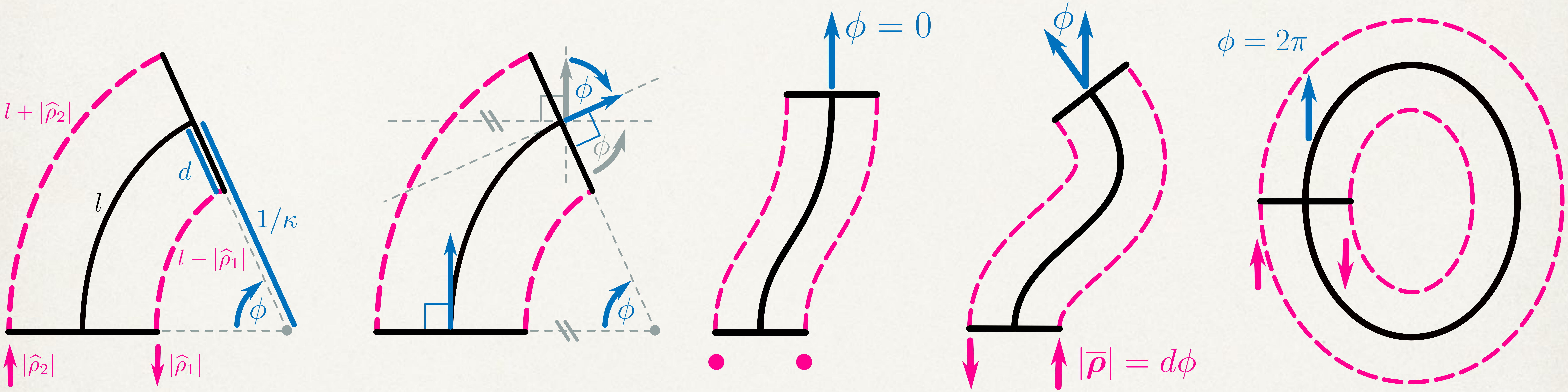






[**Grassmann** et al., arXiv (under review)] "Clarke Transform — A Fundamental Tool for Continuum Robotics"
[**Grassmann** & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"
[**Grassmann** & Burgner-Kahrs, arXiv (under review)] "Clarke Coordinates Are Generalized Improved State Parametrization for Continuum Robots"
[**Grassmann** & Burgner-Kahrs, arXiv (under review)] "Displacement-Actuated Continuum Robots: A Joint Space Abstraction"

Applicable Beyond Constant-Curvature



$$\begin{bmatrix} d\phi\cos(\theta) \\ d\phi\sin(\theta) \end{bmatrix} = \overline{\boldsymbol{\rho}} = \boldsymbol{M}_{\mathcal{P}}\boldsymbol{\rho}$$

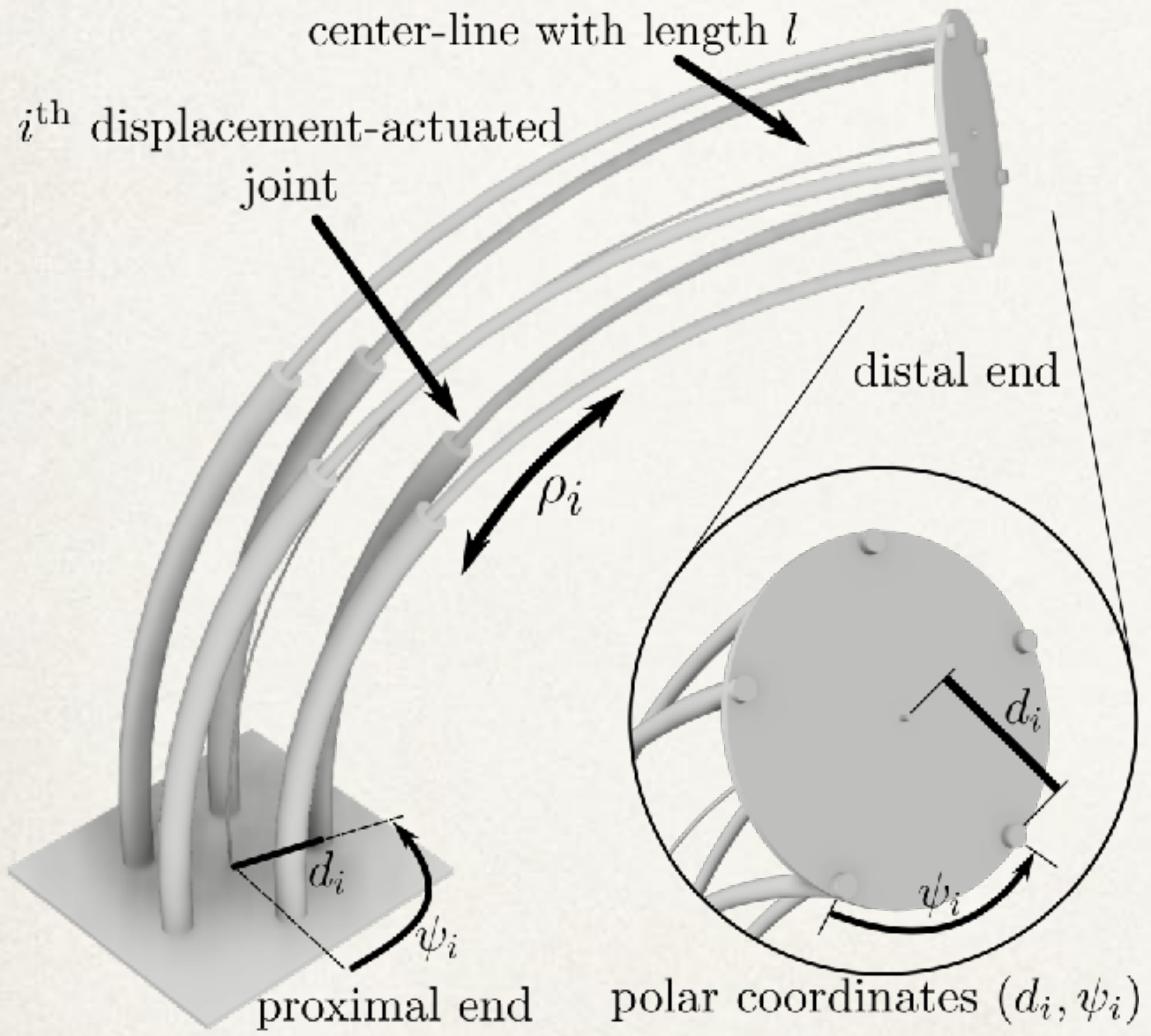
[**Grassmann** & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"
[**Grassmann** & Burgner-Kahrs, arXiv (under review)] "Displacement-Actuated Continuum Robots: A Joint Space Abstraction"

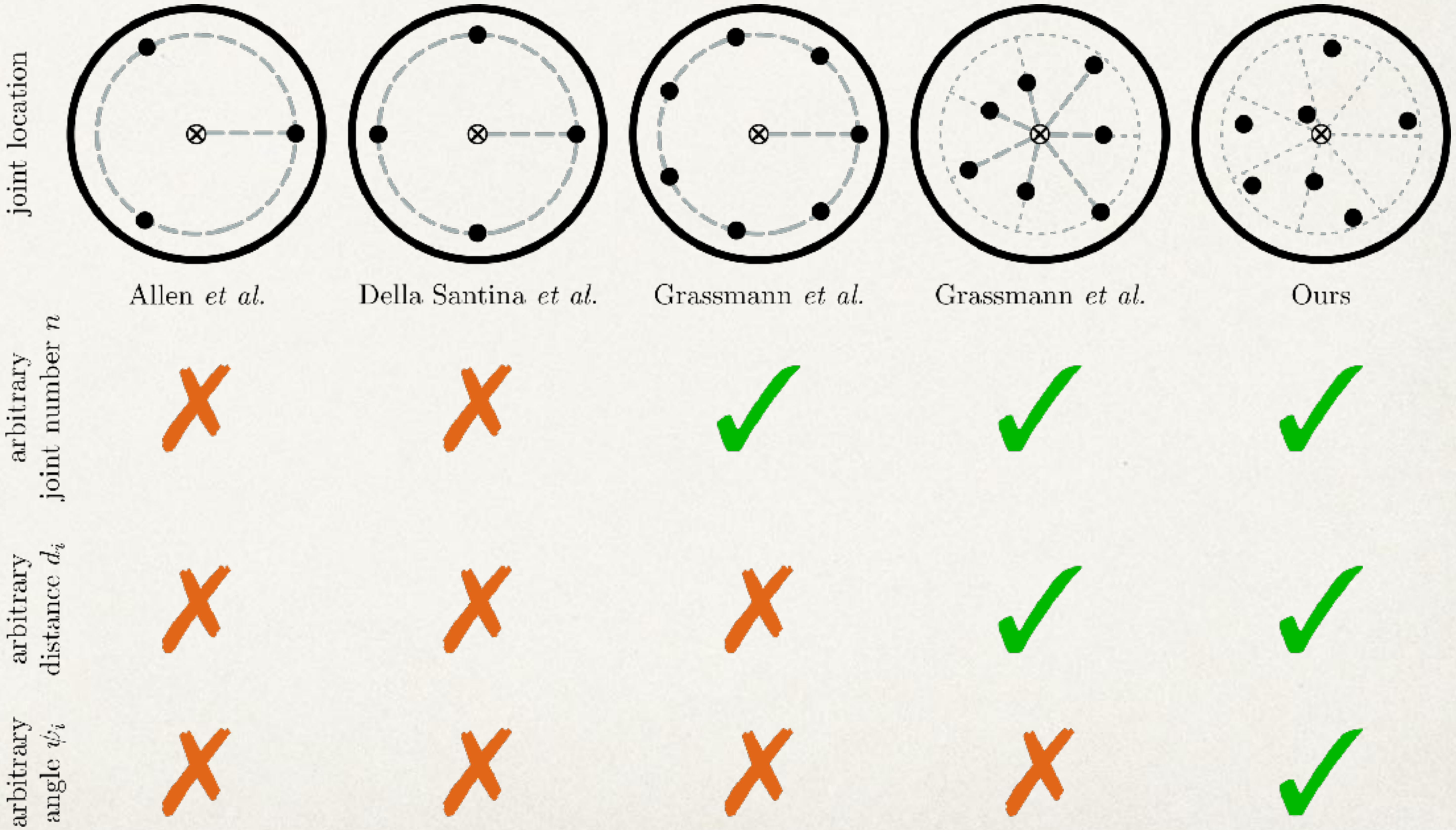
Improved State Parameterizations

Reference	n	Parameterization *w.r.t.* joint values		Parameterization *w.r.t.* Clarke Coordinates	
Della Santina *et al.*	4	$\Delta_x = \frac{l_3 - l_1}{2}$	$\Delta_y = \frac{l_4 - l_2}{2}$	$\Delta_x = \rho_{\mathrm{Re}}$	$\Delta_y = \rho_{\mathrm{Im}}$
Allen *et al.*	3	$u = \frac{l_2 - l_3}{\sqrt{3}d}$	$v = \frac{(l_1 + l_2 + l_3)/3 - l_1}{d}$	$v = (1/d)\,\rho_{\mathrm{Re}}$	$u = (1/d)\,\rho_{\mathrm{Im}}$
Allen *et al.*	4	$u = \frac{l_2 - l_4}{d}$	$v = \frac{l_3 - l_1}{d}$	$v = (2/d)\,\rho_{\mathrm{Re}}$	$u = (2/d)\,\rho_{\mathrm{Im}}$
Dian *et al.*	3	$\Delta x = \frac{l_2 + l_3 - 2l_1}{3}$	$\Delta y = \frac{l_3 - l_2}{\sqrt{3}}$	$\Delta x = \rho_{\mathrm{Re}}$	$\Delta y = \rho_{\mathrm{Im}}$
Grassmann *et al.*	n	$[\rho_{\mathrm{Re}}, \rho_{\mathrm{Im}}]^\top = \boldsymbol{M}_{\mathcal{P}}\boldsymbol{\rho}$		ρ_{Re}	ρ_{Im}

[Della Santina et al., RA-L 2020] “On an Improved State Parametrization for Soft Robots with Piecewise Constant Curvature and Its Use in Model-Based Control”
[Allen et al., RoboSoft 2020] “Closed-Form Non-Singular Constant-Curvature Continuum Manipulator Kinematics”
[Dian et al., Access 2022] “A Novel Disturbance-Rejection Control Framework for Cable-Driven Continuum Robots With Improved State Parameterization”
[**Grassmann** & Burgner-Kahrs, arXiv (under review)] “Clarke Coordinates Are Generalized Improved State Parametrization for Continuum Robots”

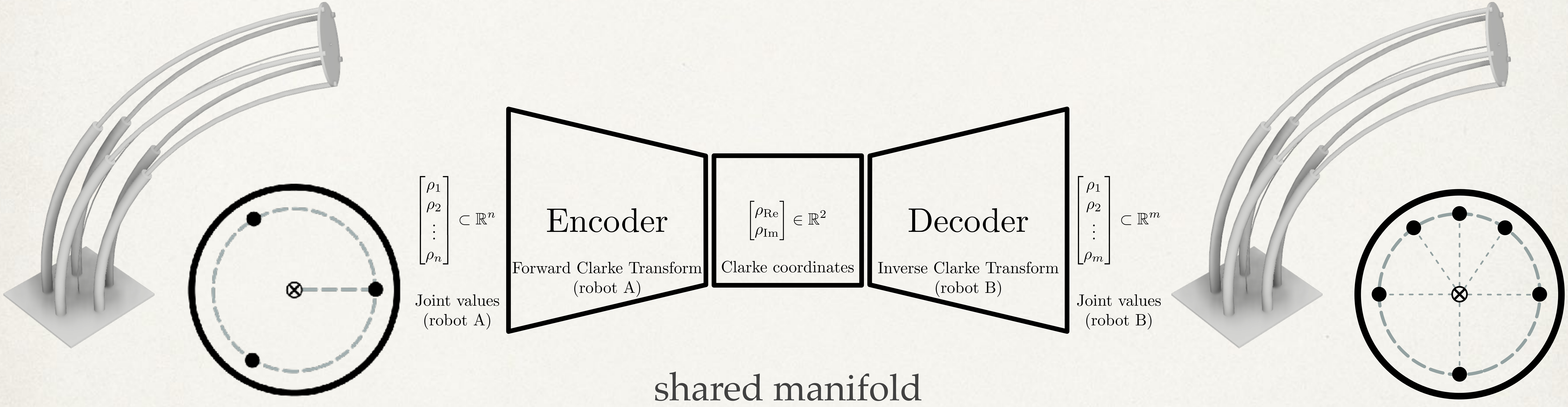
Arbitrary Joint Locations





[**Grassmann** & Burgner-Kahrs, RoboSoft 2025] “Clarke Transform and Encoder-Decoder Architecture for Arbitrary Joint Location in Displacement-Actuated Continuum Robots”

Re-using Previous Approaches

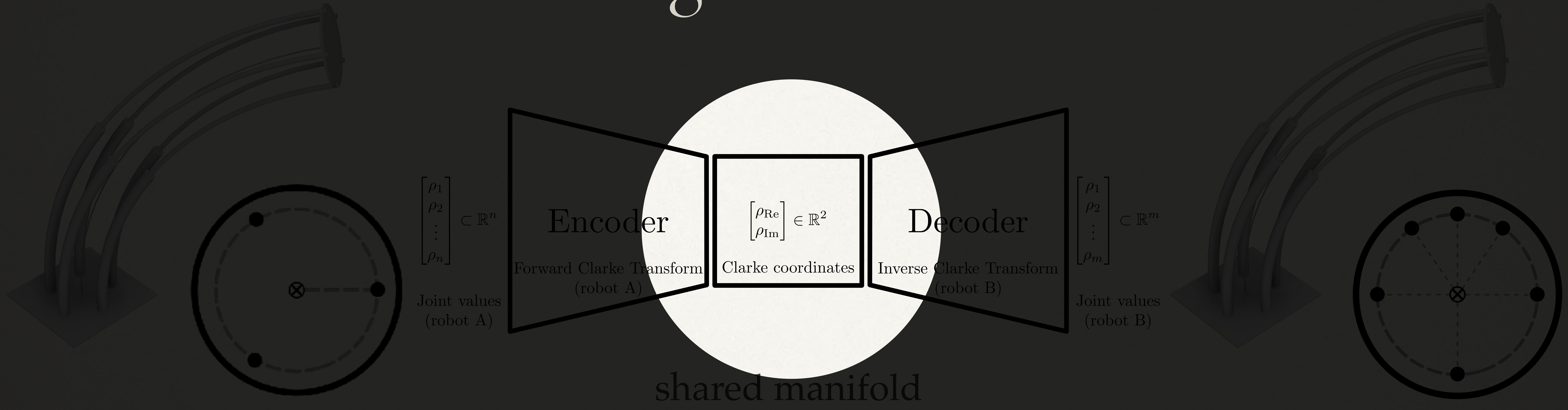


$$\boldsymbol{\rho}_{(\text{robot B})} = \overbrace{l_{(\text{robot B})}\,\mathrm{diag}\left(d_{i,(\text{robot B})}\right)\boldsymbol{M}_{\mathcal{P}}^{-1}{}_{(\text{robot B})}}^{\text{adds design parameters of robot B}} \cdot \underbrace{\frac{1}{l_{(\text{robot A})}}\boldsymbol{M}_{\mathcal{P}\,(\text{robot A})}\,\mathrm{diag}\left(\frac{1}{d_{i,(\text{robot A})}}\right)}_{\text{removes design parameters of robot A}}\boldsymbol{\rho}_{(\text{robot A})}$$

[**Grassmann** & Burgner-Kahrs, RoboSoft 2025] "Clarke Transform and Encoder-Decoder Architecture for Arbitrary Joint Location in Displacement-Actuated Continuum Robots"

Re-using Previous Approaches

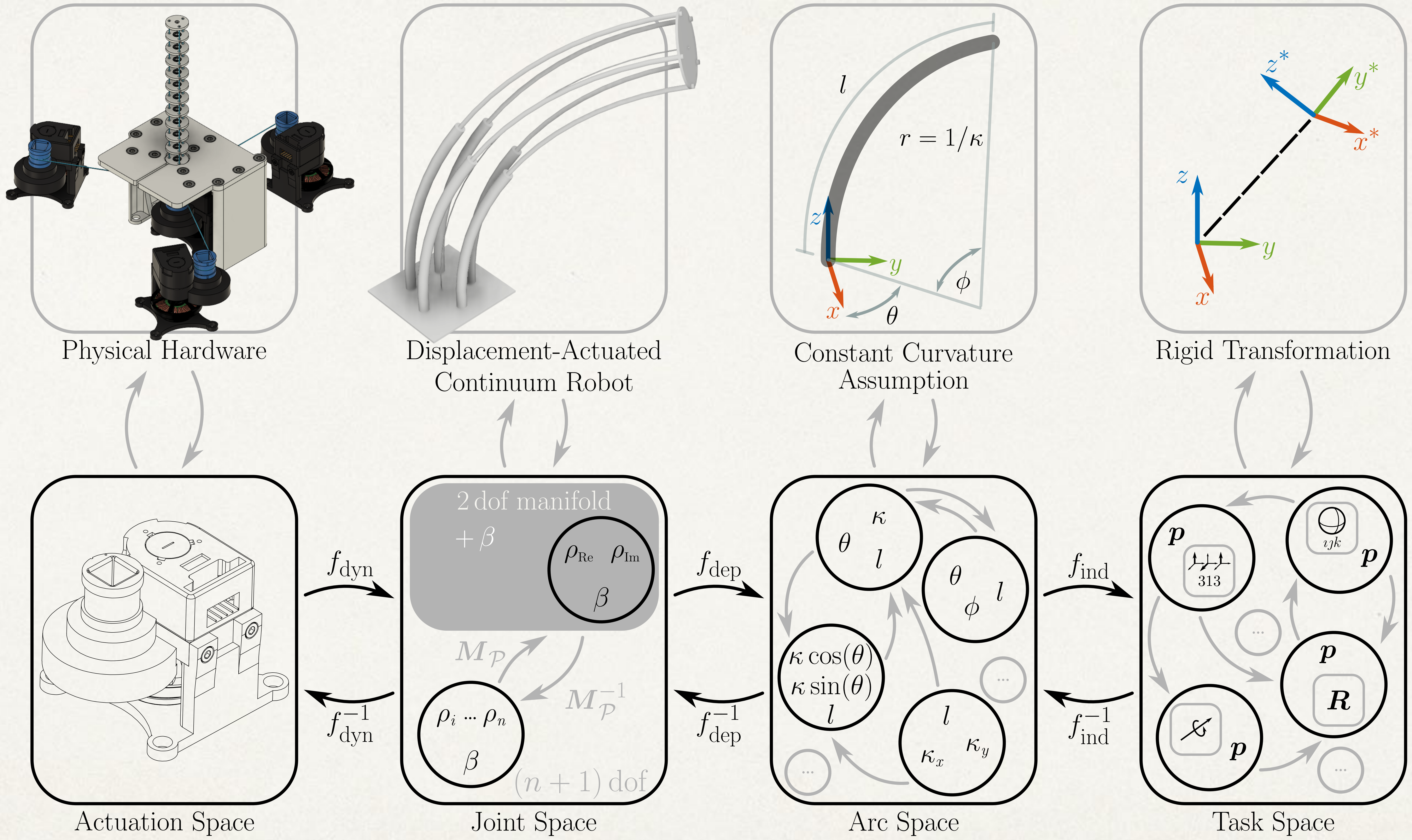
Lingua Franca



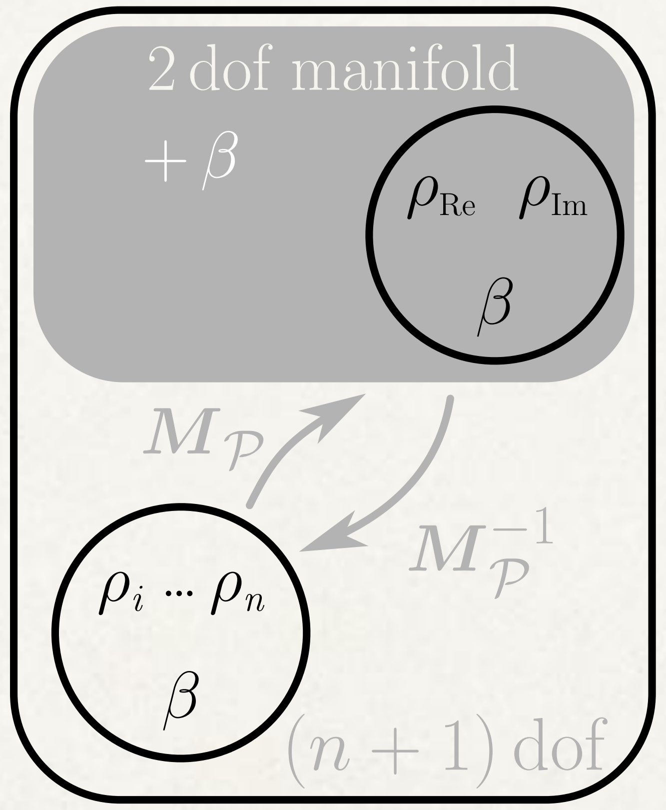
Clarke Transform & Clarke Coordinates

$$\boldsymbol{\rho}_{(\text{robot B})} = \overbrace{l_{(\text{robot B})} \operatorname{diag}\left(d_{i,(\text{robot B})}\right) \boldsymbol{M}_{\mathcal{P}}^{-1}}^{\text{adds design parameters of robot B}}{}_{(\text{robot B})} \underbrace{\frac{1}{l_{(\text{robot A})}} \boldsymbol{M}_{\mathcal{P}\,(\text{robot A})} \operatorname{diag}\left(\frac{1}{d_{i,(\text{robot A})}}\right)}_{\text{removes design parameters of robot A}} \boldsymbol{\rho}_{(\text{robot A})}$$

[**Grassmann** & Burgner-Kahrs, RoboSoft 2025] “Clarke Transform and Encoder-Decoder Architecture for Arbitrary Joint Location in Displacement-Actuated Continuum Robots”

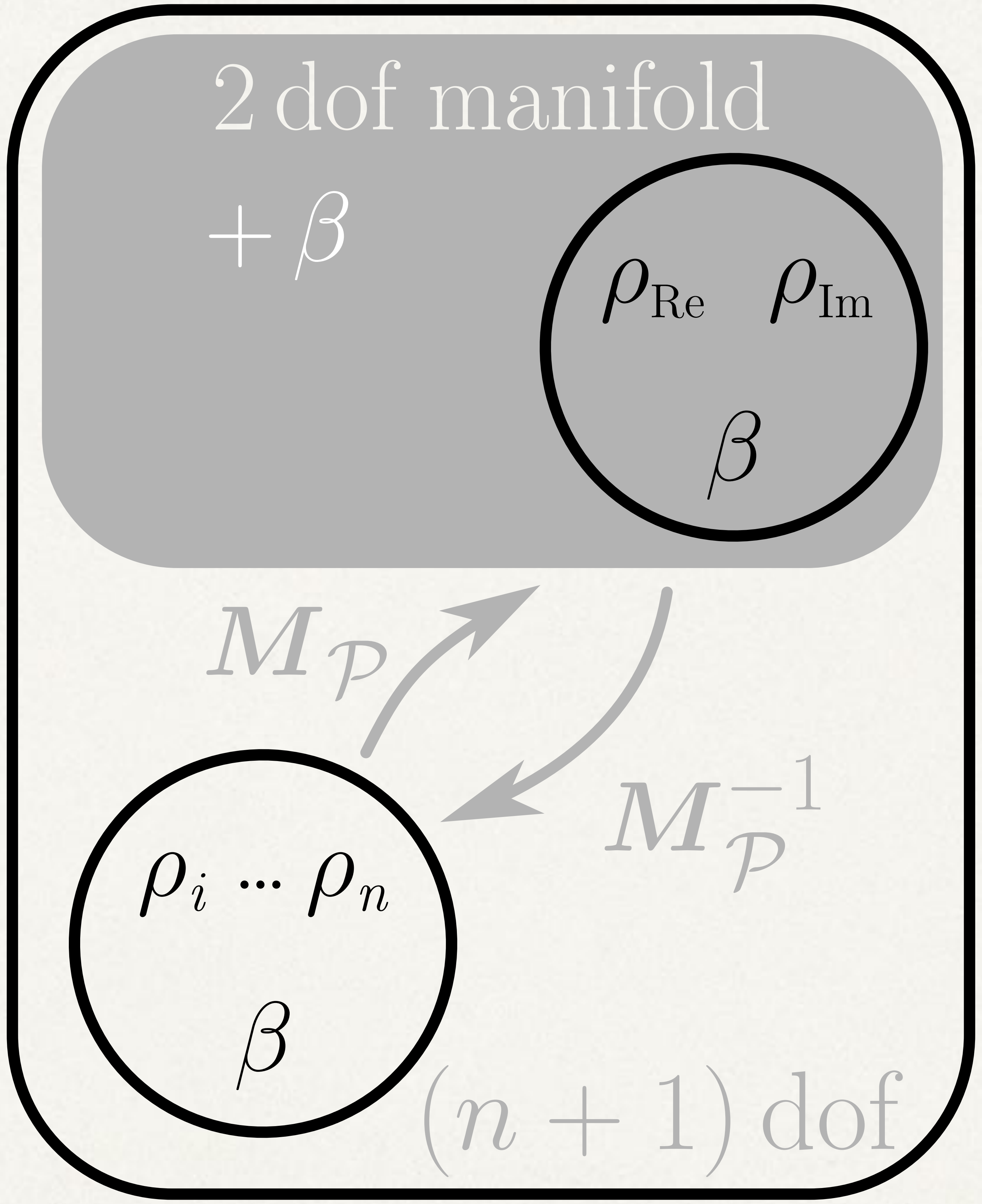


Physical Hardware
Displacement-Actuated Continuum Robot
Constant Curvature Assumption
Rigid Transformation
Actuation Space
Joint Space
Arc Space
Task Space
2 dof manifold
$+\beta$
$(n+1)$ dof
f_{dyn}
f_{dyn}^{-1}
f_{dep}
f_{dep}^{-1}
f_{ind}
f_{ind}^{-1}
$r = 1/\kappa$



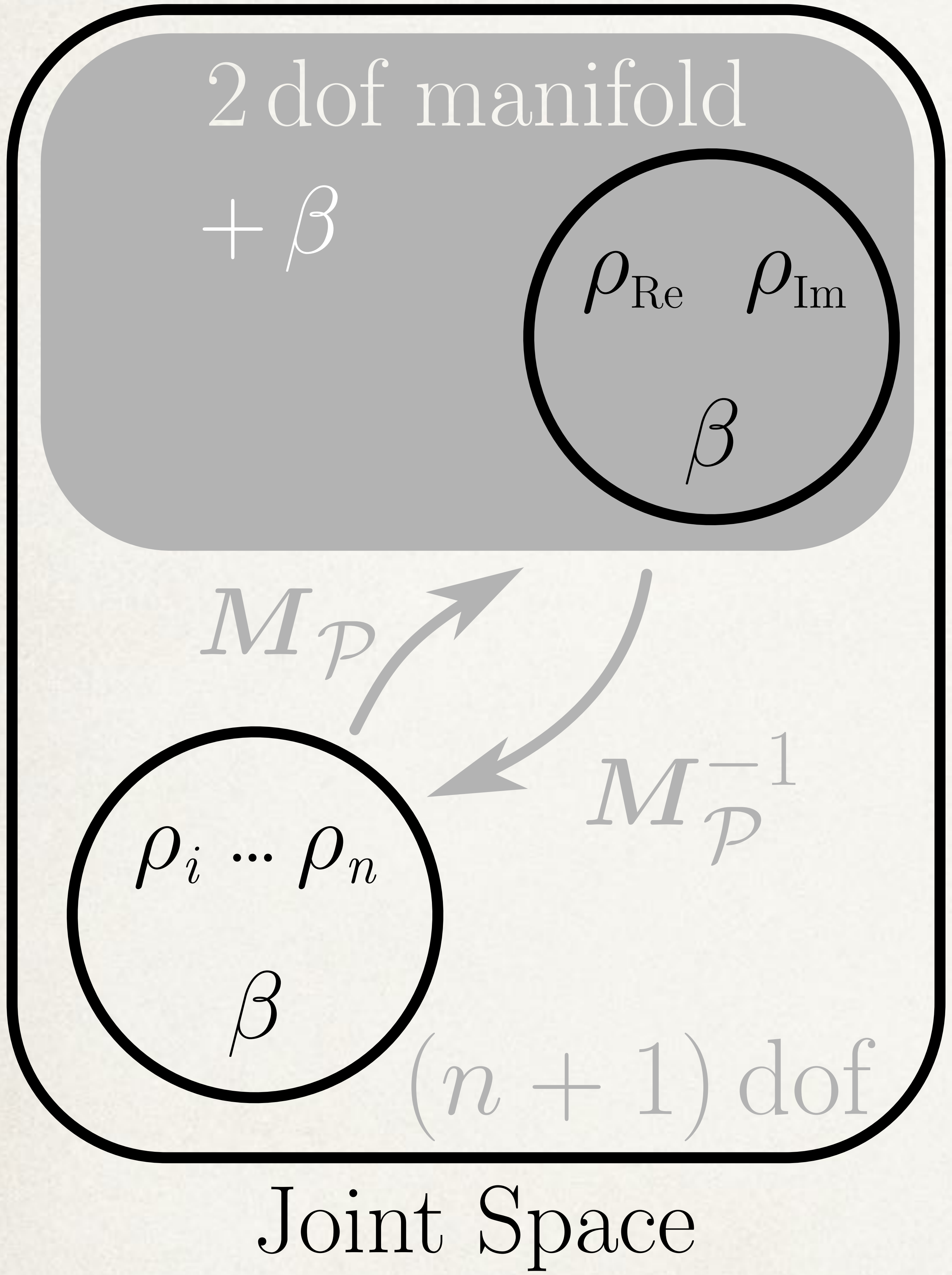
2 dof manifold
$+\beta$
ρ_{Re} ρ_{Im}
β
$\boldsymbol{M}_{\mathcal{P}}$
$\boldsymbol{M}_{\mathcal{P}}^{-1}$
$\rho_i \cdots \rho_n$
β
$(n+1)$ dof

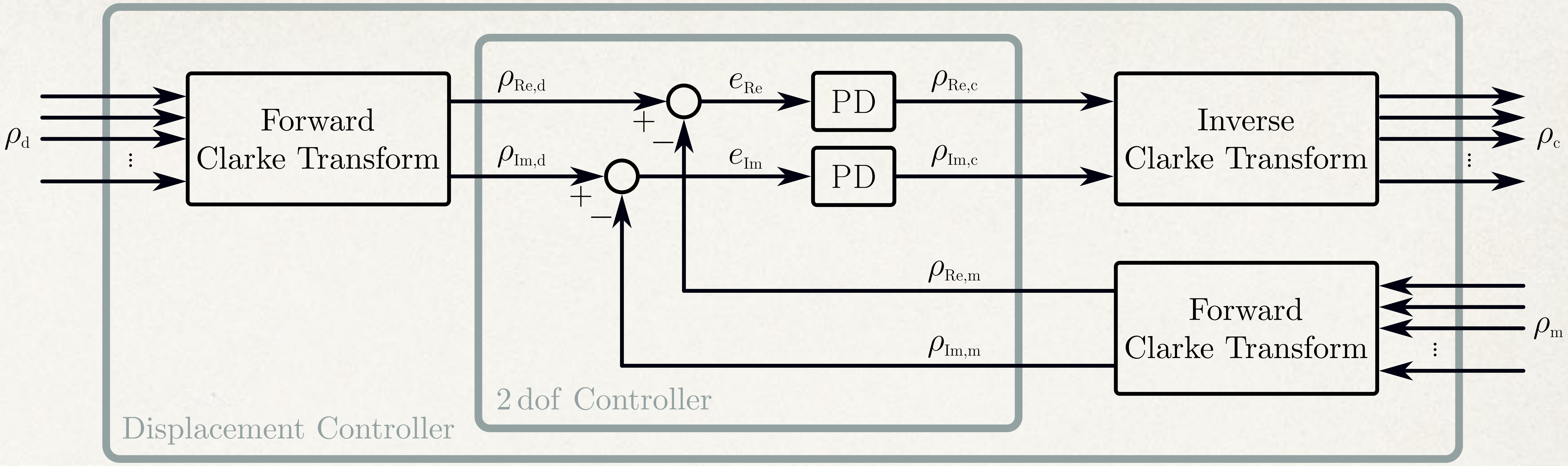
Joint Space

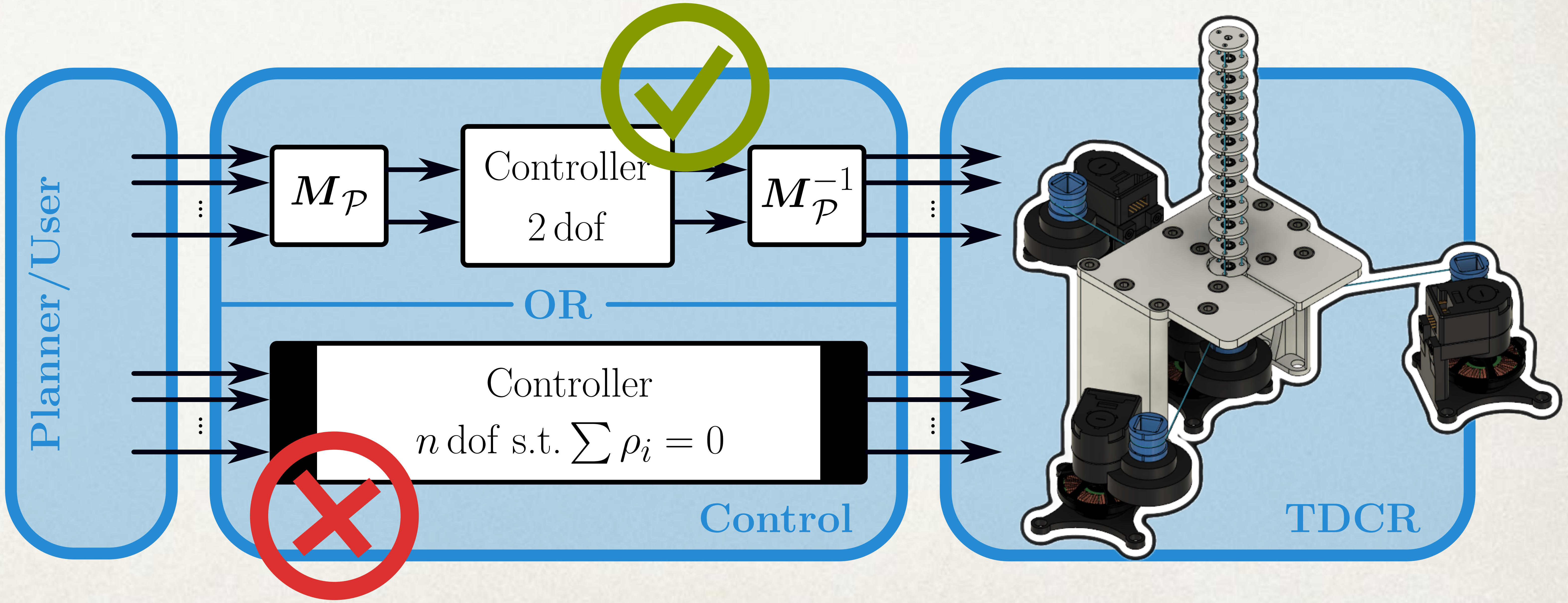


2 dof manifold
$+\beta$
ρ_{Re} ρ_{Im}
β
$M_{\mathcal{P}}$
$M_{\mathcal{P}}^{-1}$
$\rho_i \dots \rho_n$
β
$(n+1)$ dof

Joint Space

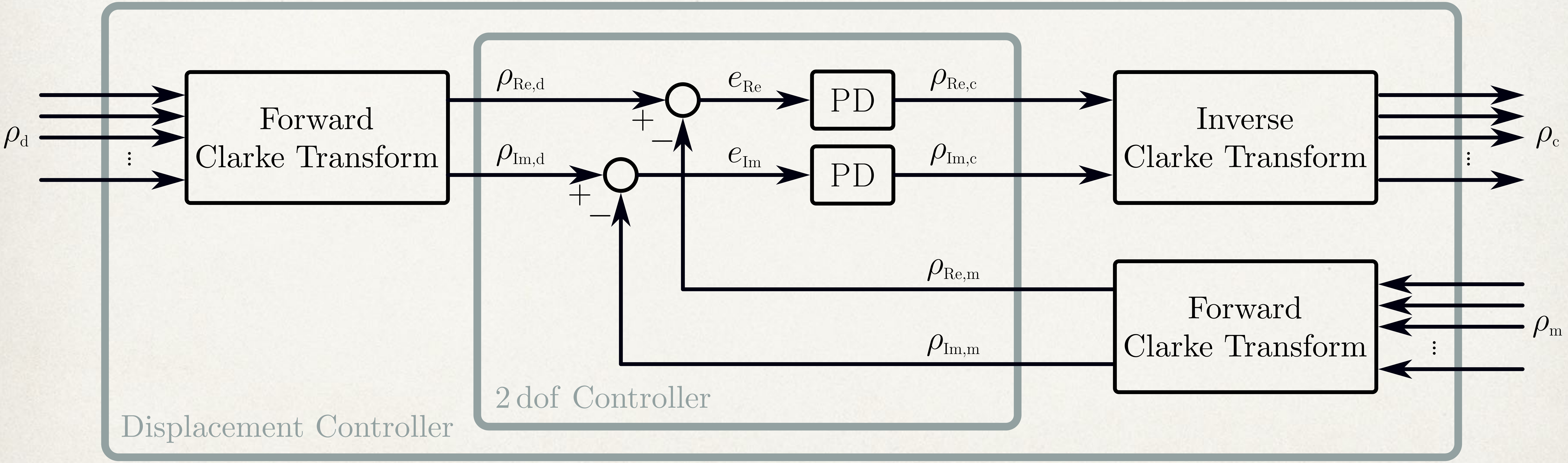






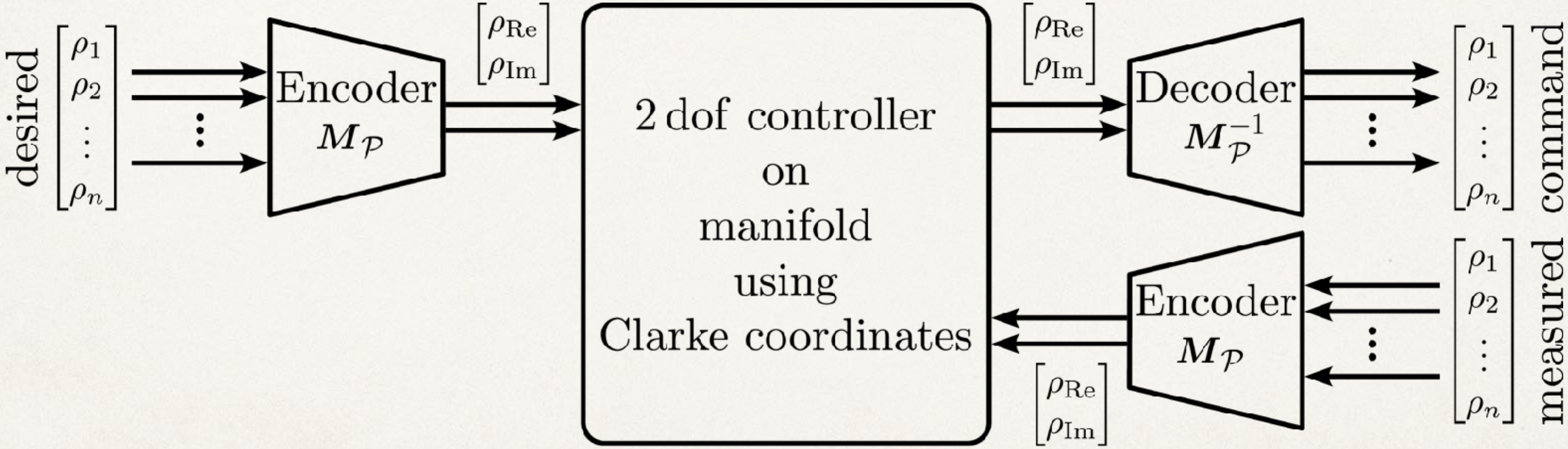
[**Grassmann** et al., arXiv (under review)] "Clarke Transform — A Fundamental Tool for Continuum Robotics"
[**Grassmann** & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Simple Units



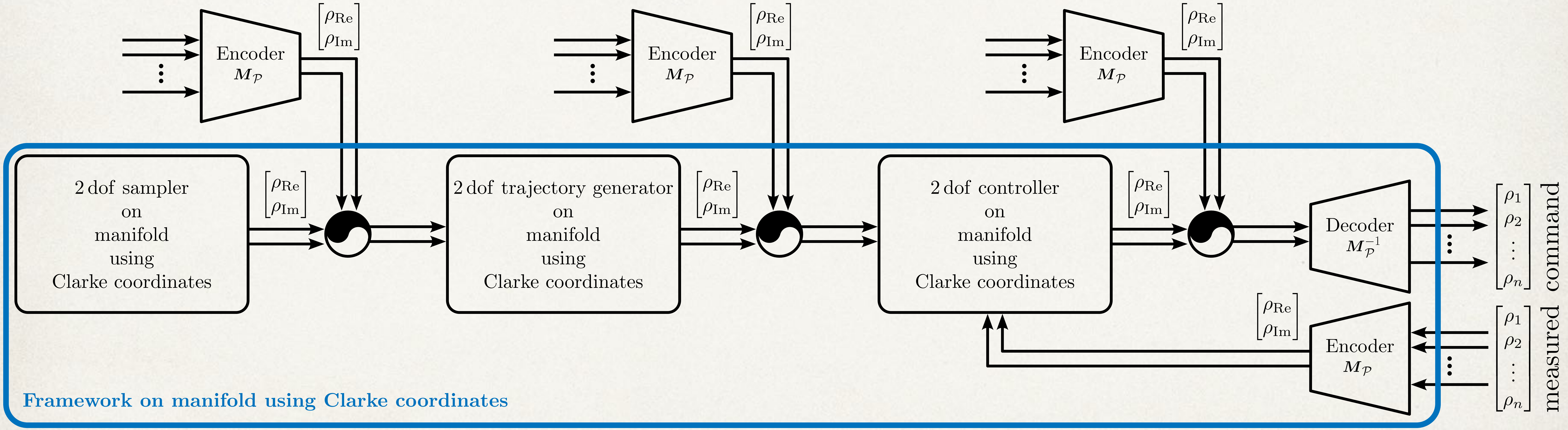
[Grassmann & Burgner-Kahrs, RoboSoft 2025] "Clarke Transform and Encoder-Decoder Architecture for Arbitrary Joint Location in Displacement-Actuated Continuum Robots"
[Grassmann & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Simple Units



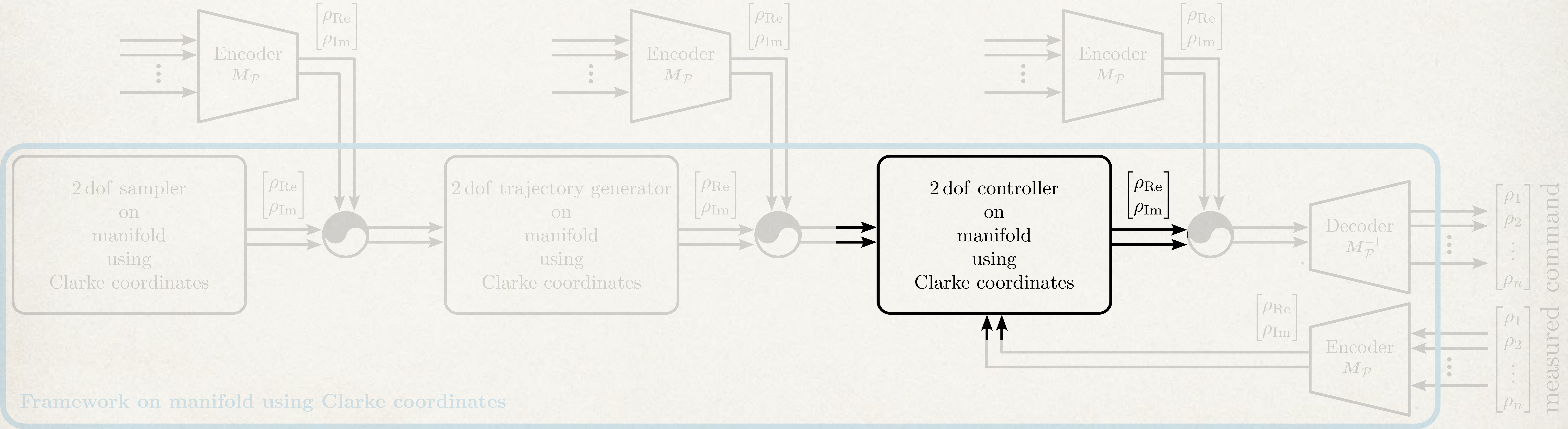
[Grassmann & Burgner-Kahrs, RoboSoft 2025] "Clarke Transform and Encoder-Decoder Architecture for Arbitrary Joint Location in Displacement-Actuated Continuum Robots"
[Grassmann & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Concatenation of Units



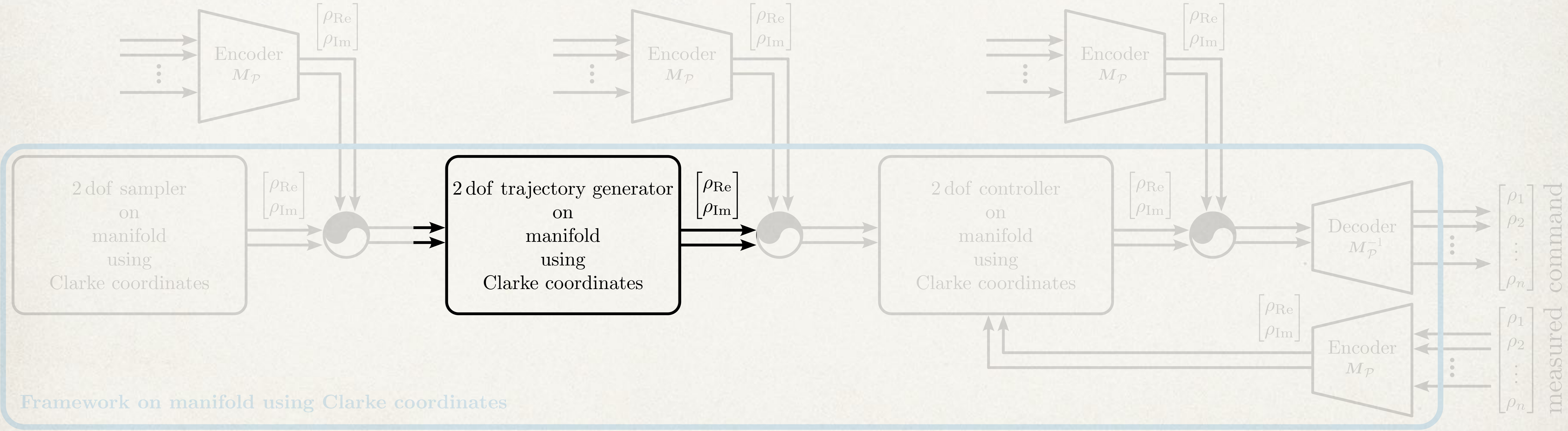
[Grassmann & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Concatenation of Units



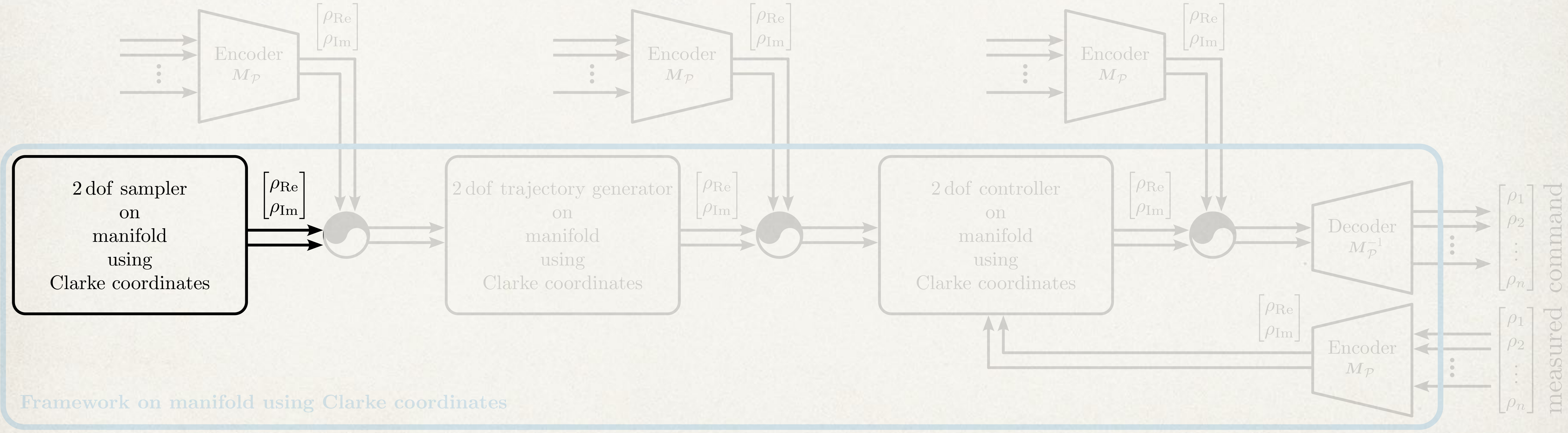
[Grassmann & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Concatenation of Units



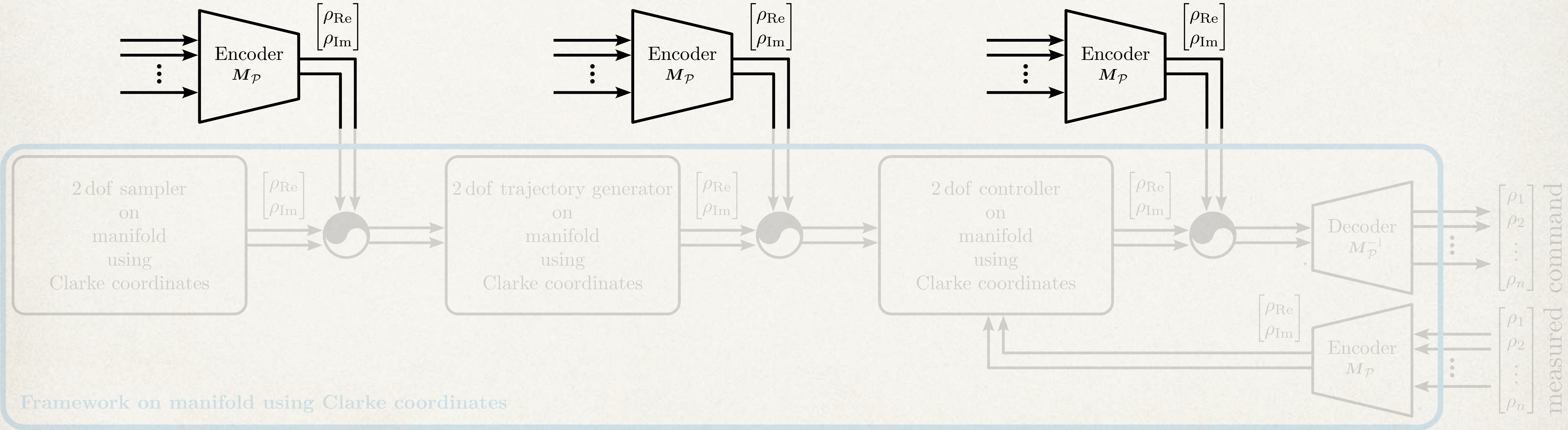
[Grassmann & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Concatenation of Units



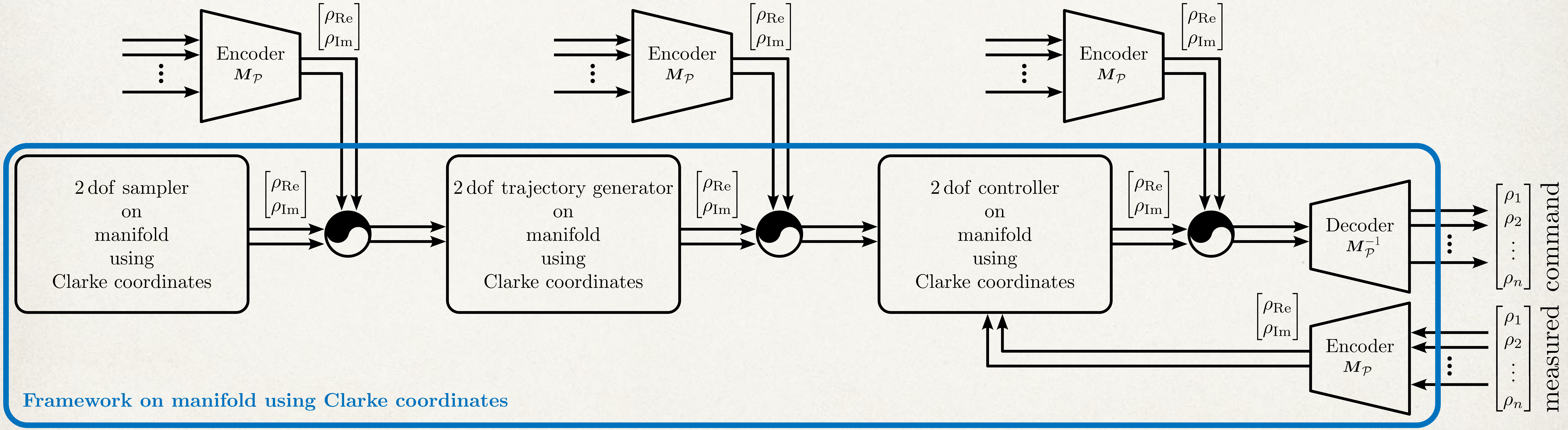
[Grassmann & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Concatenation of Units



[Grassmann & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Concatenation of Units

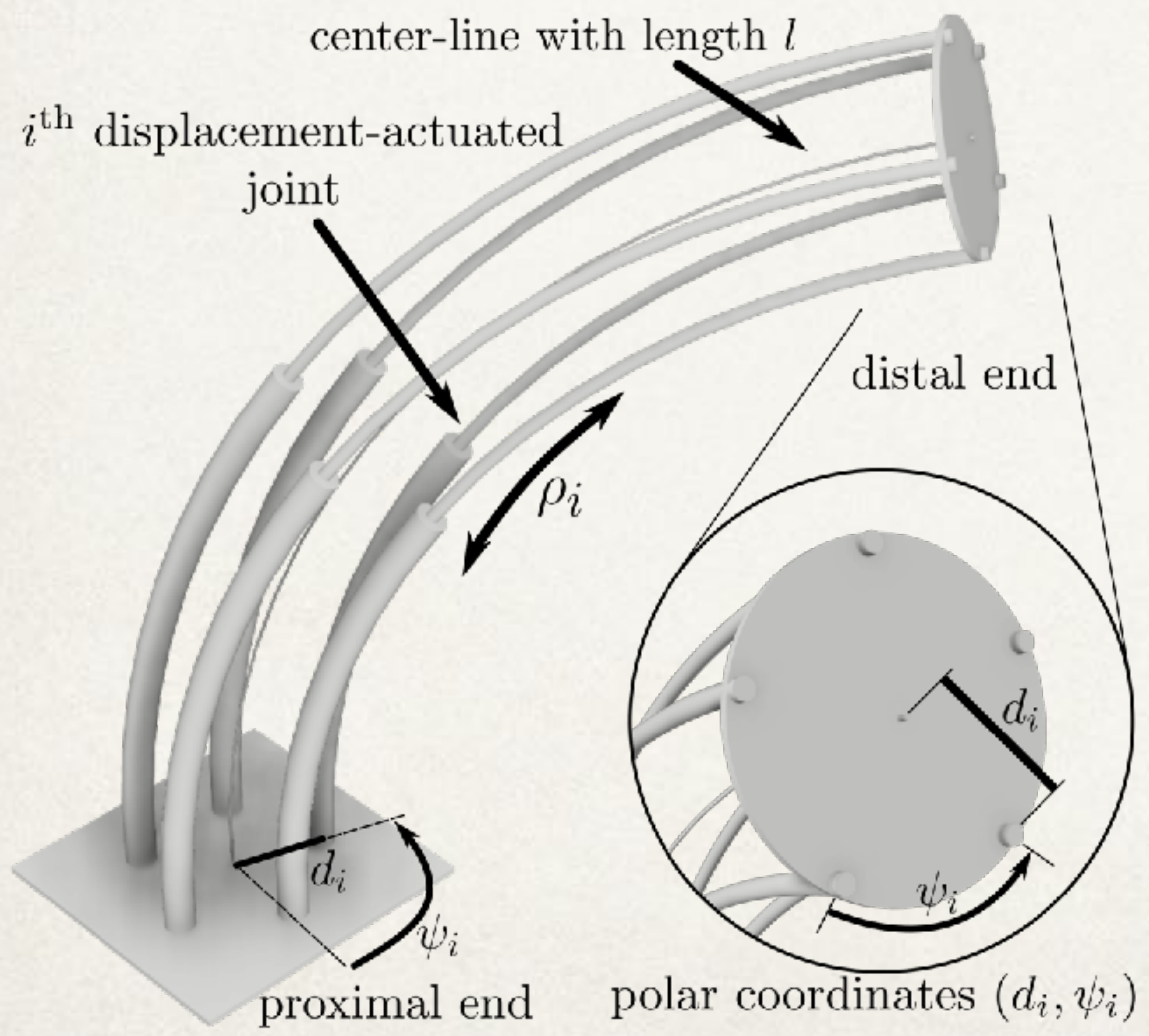


[Grassmann & Burgner-Kahrs, arXiv (under review)] "Using Clarke Transform to Create a Framework on the Manifold"

Takeaway: Prelude of Benchmarking

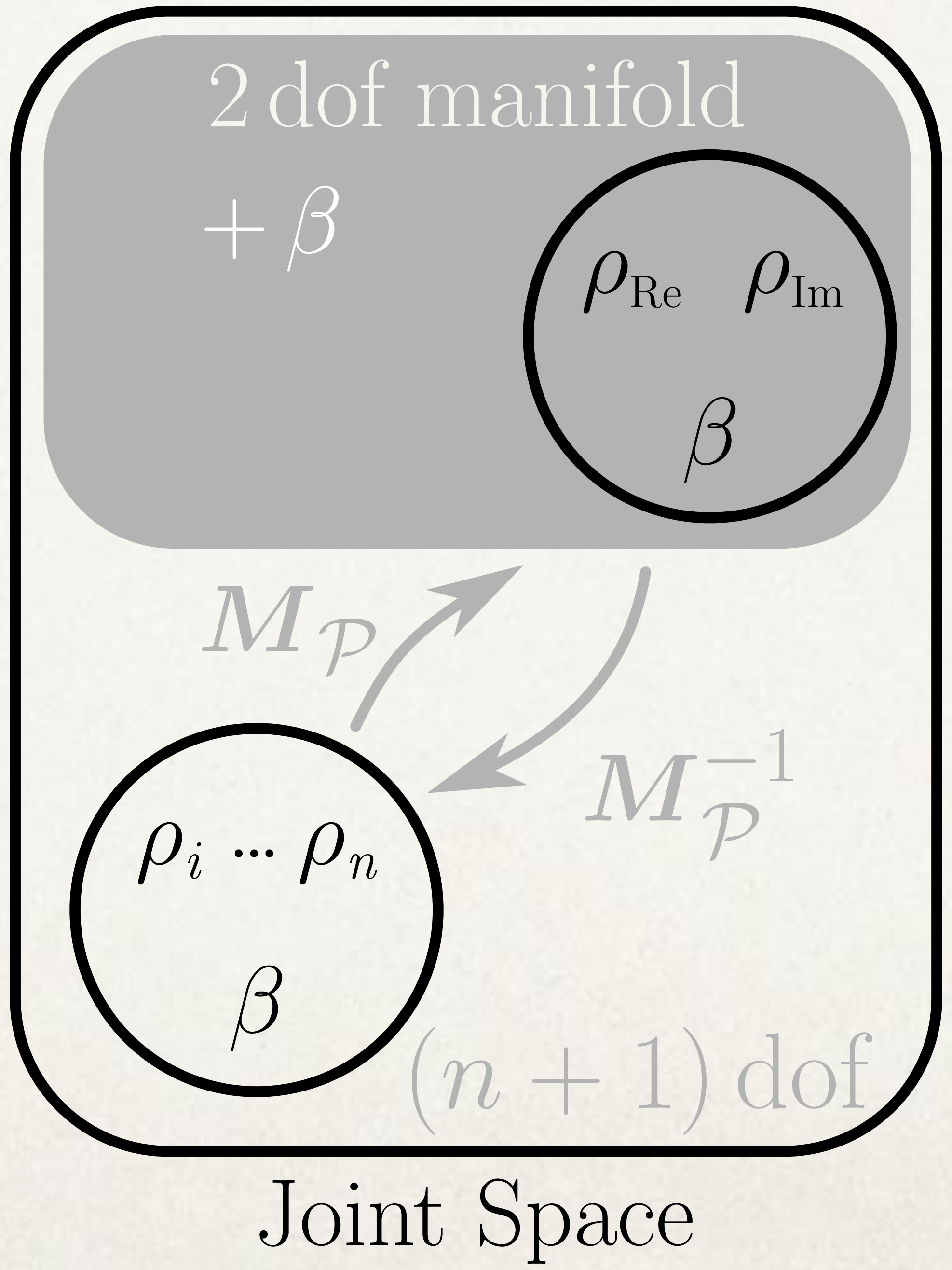
I.

right definition and abstraction



II.

manifolds and parameterization



III.

testable units