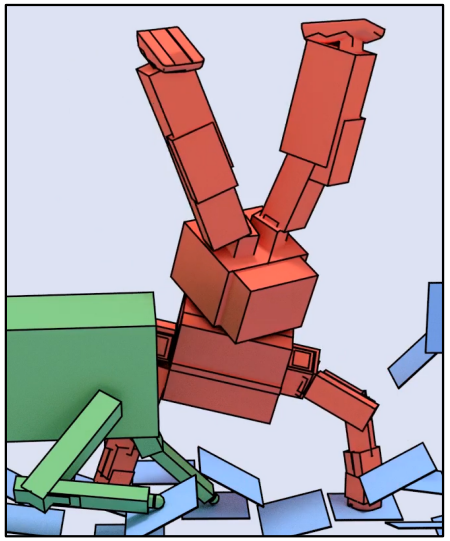
MULTI-CONTACT PLANNING USING REDUCTION PROPERTIES : WHY AND HOW


Martigny, january 30, 2019
Steve Tonneau

THE UNIVERSITY OF EDINBURGH

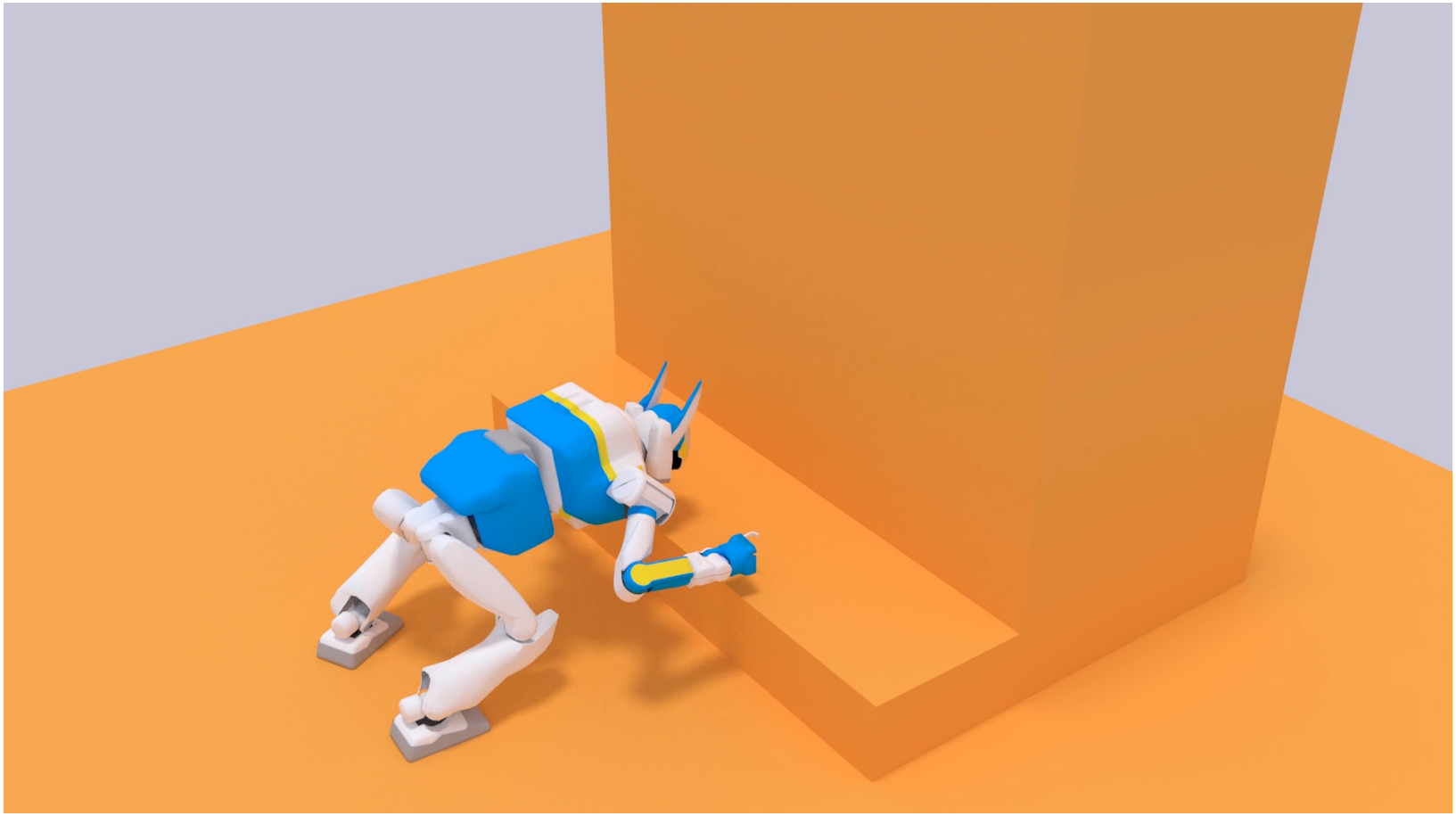
memmo

WHAT IS MISSING IN PREVIOUS CLASSES ?

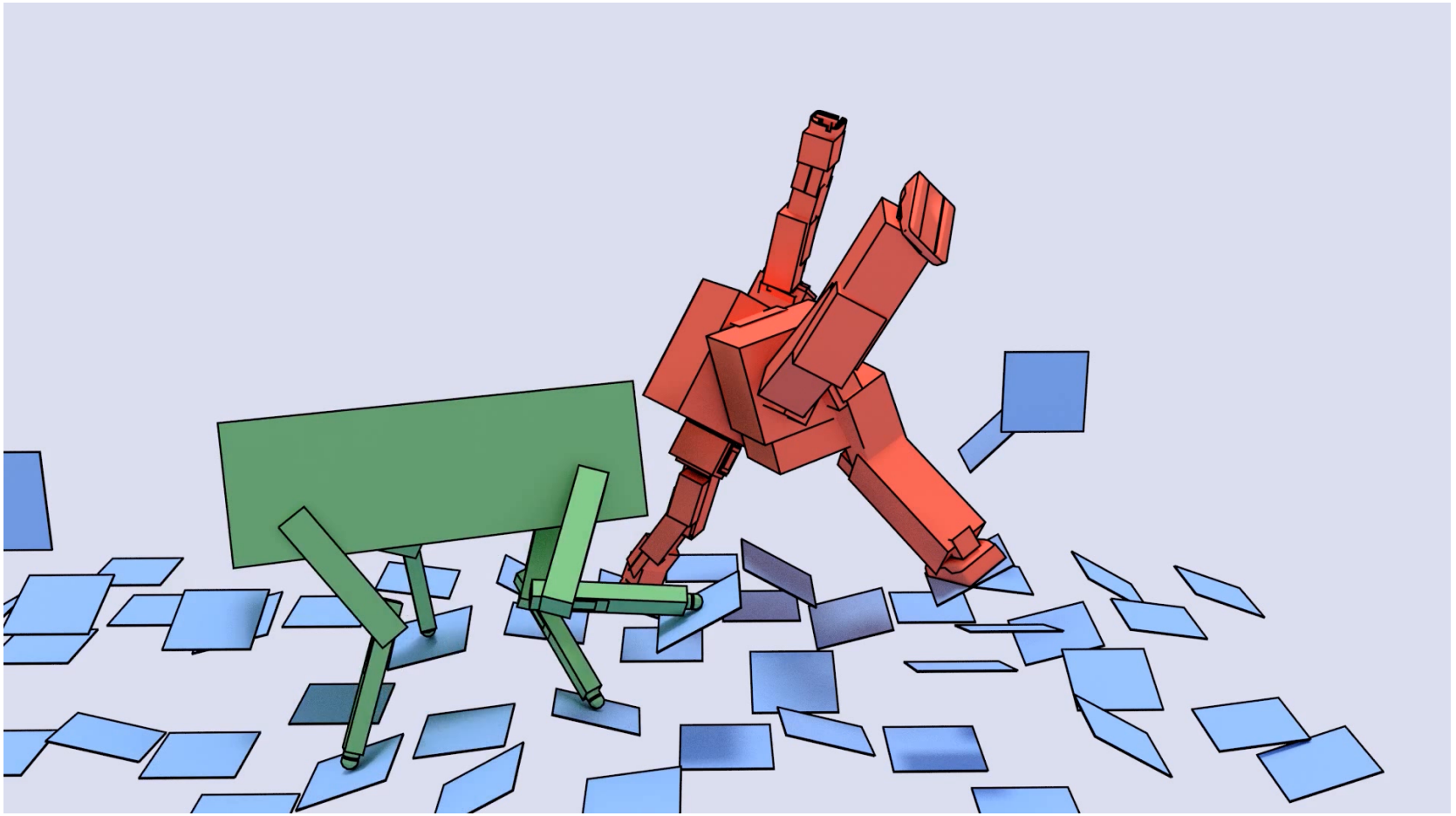
The reference:

- Contacts: locations, timings…
- Joint / end-effectors reference motion
- Environment constraints (collisions…)

WHAT IS MULTI CONTACT LOCOMOTION ?



WHAT IS MULTI CONTACT LOCOMOTION ?



IN GENERAL, LOCOMOTION SYNTHESIS IS HARD

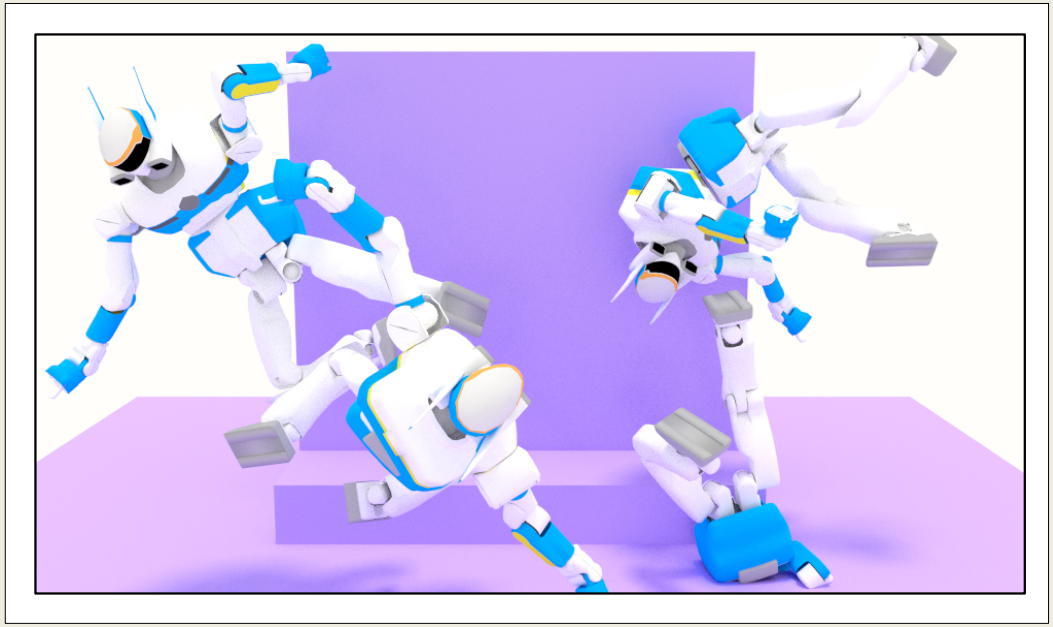
IN GENERAL, LOCOMOTION SYNTHESIS IS HARD

Several hard, interdependent sub-problems…

IN GENERAL, LOCOMOTION SYNTHESIS IS HARD

Several hard, interdependent sub-problems…

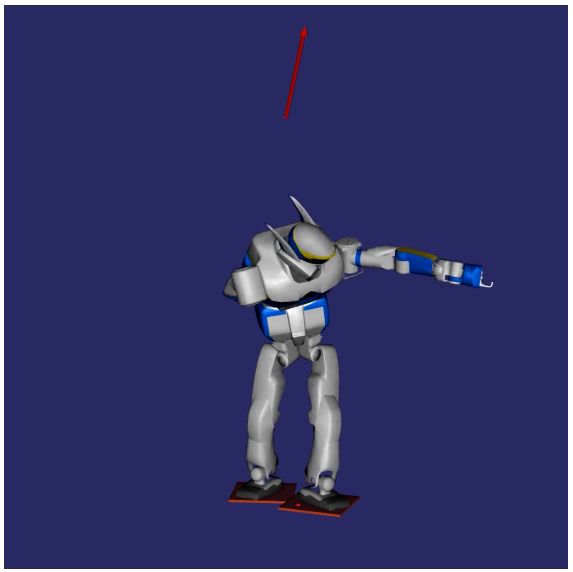
- High dimensional search space (> 100)



IN GENERAL, LOCOMOTION SYNTHESIS IS HARD

Several hard, interdependent sub-problems…

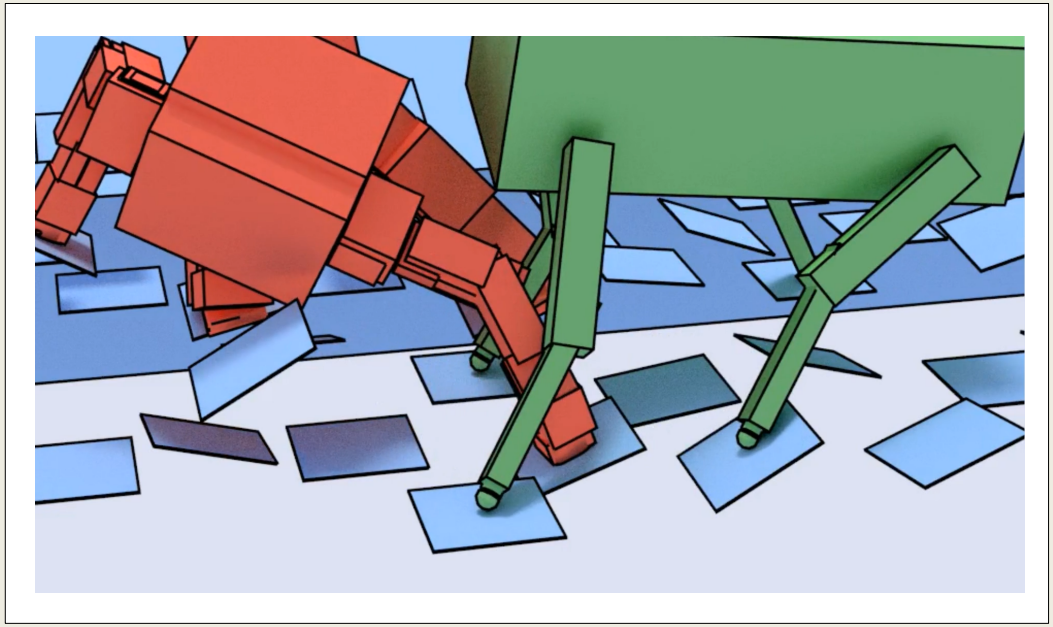
- High dimensional search space (> 100)
- Instable, discontinuous, non-convex contact dynamics



IN GENERAL, LOCOMOTION SYNTHESIS IS HARD

Several hard, interdependent sub-problems…

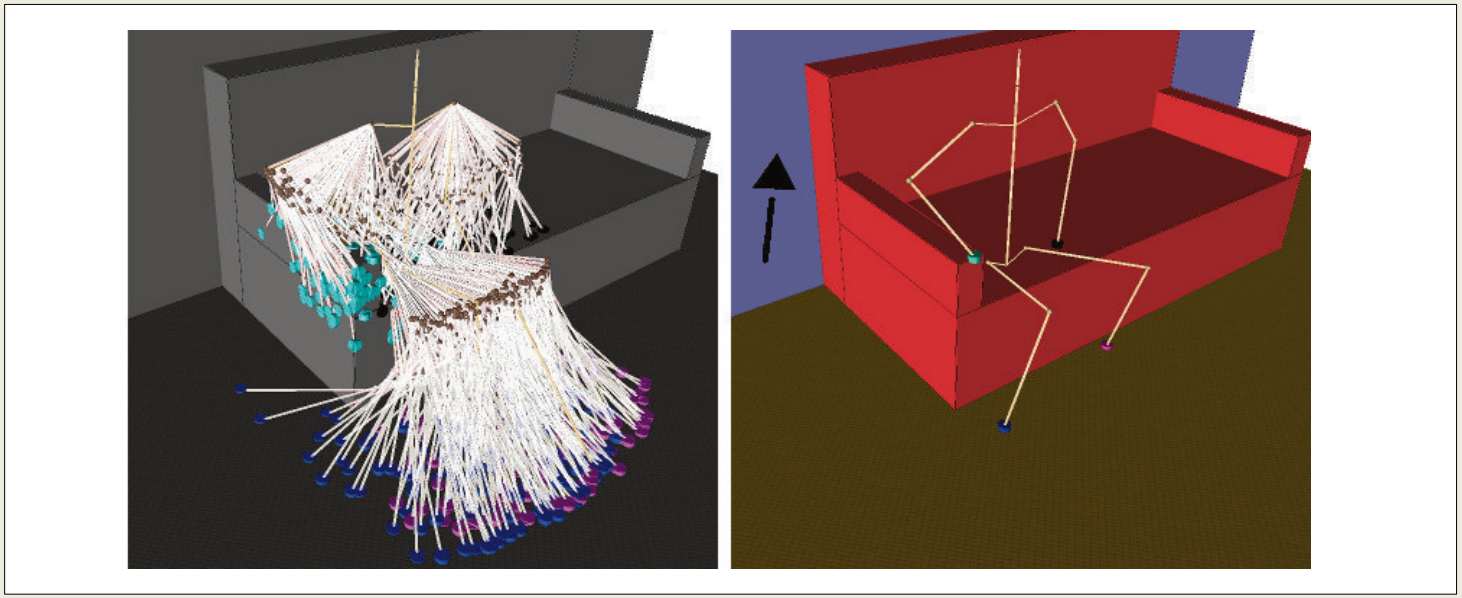
- High dimensional search space (> 100)
- Instable, discontinuous, non-convex contact dynamics
- Non-convex collision avoidance constraints



IN GENERAL, LOCOMOTION SYNTHESIS IS HARD

Several hard, interdependent sub-problems…

… within a combinatorial contact planning issue



STOP! THAT'S TOO HARD !

STOP! THAT'S TOO HARD !

Do we have other descriptions of motion?

Something pragmatic and not so elaborated?

STOP! THAT'S TOO HARD !

Do we have other descriptions of motion?

Something pragmatic and not so elaborated?

let's ask the French army:

"Walking is putting one foot in front of the other, and then starting over"

THE ARMY ALGORITHM

```
While(not_ordered_otherwise):
    PutBackFootForward()
```

OKAY, SO WHAT ?

Whole point of this class is to advocate for

- Abstraction / reduction properties
- Heuristics

  (A theoretically accurate heuristic is a reduction property)

Why?

- To efficiently discriminate over huge search space
- To provide generative methods (vs data-based)

THE PLAN FOR THIS CLASS

While(not_ordered_otherwise):

PutBackFootForward()

I. Where precisely is ``forward'' ?

a) Geometric constraints

b) Geometric and dynamic feasibility

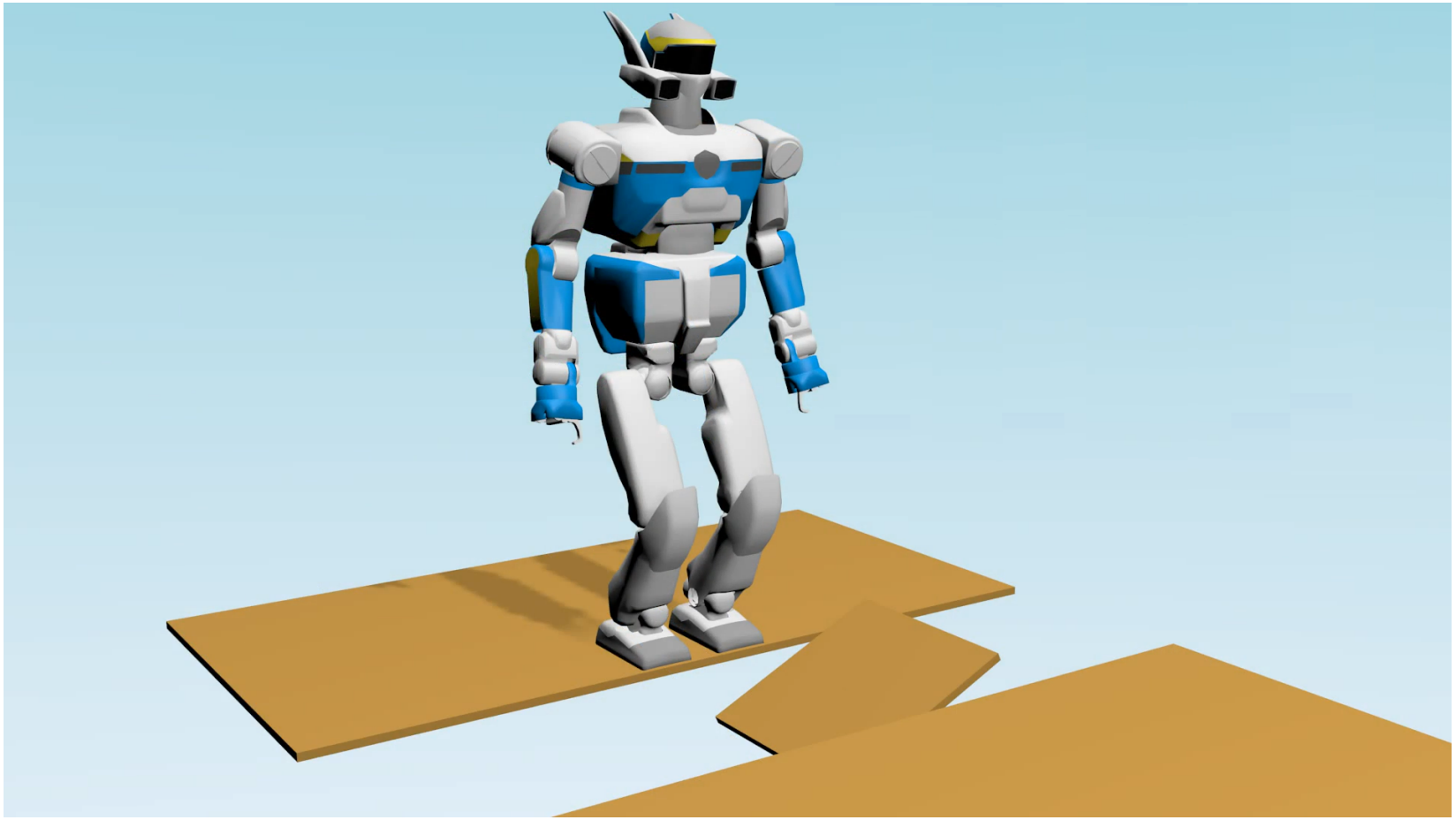
II. Who gives the orders ?

Global VS local path planning

III. Implementation: HPP-rbprm software

discussion

I. BACK FOOT FORWARD



A DISCRETE APPROACH

We can consider discrete events (contact states)

PutBackFootForward():

Given:

- Current State (feet locations)

Find:

- Next State (feet locations)

HOW DO WE CHOOSE THE FOOT LOCATION?

Geometric constraints:

- Other foot does not move
- Joint limits are not violated
- Foot is in contact (why ?)

A DISCRETE APPROACH

PutBackFootForward():

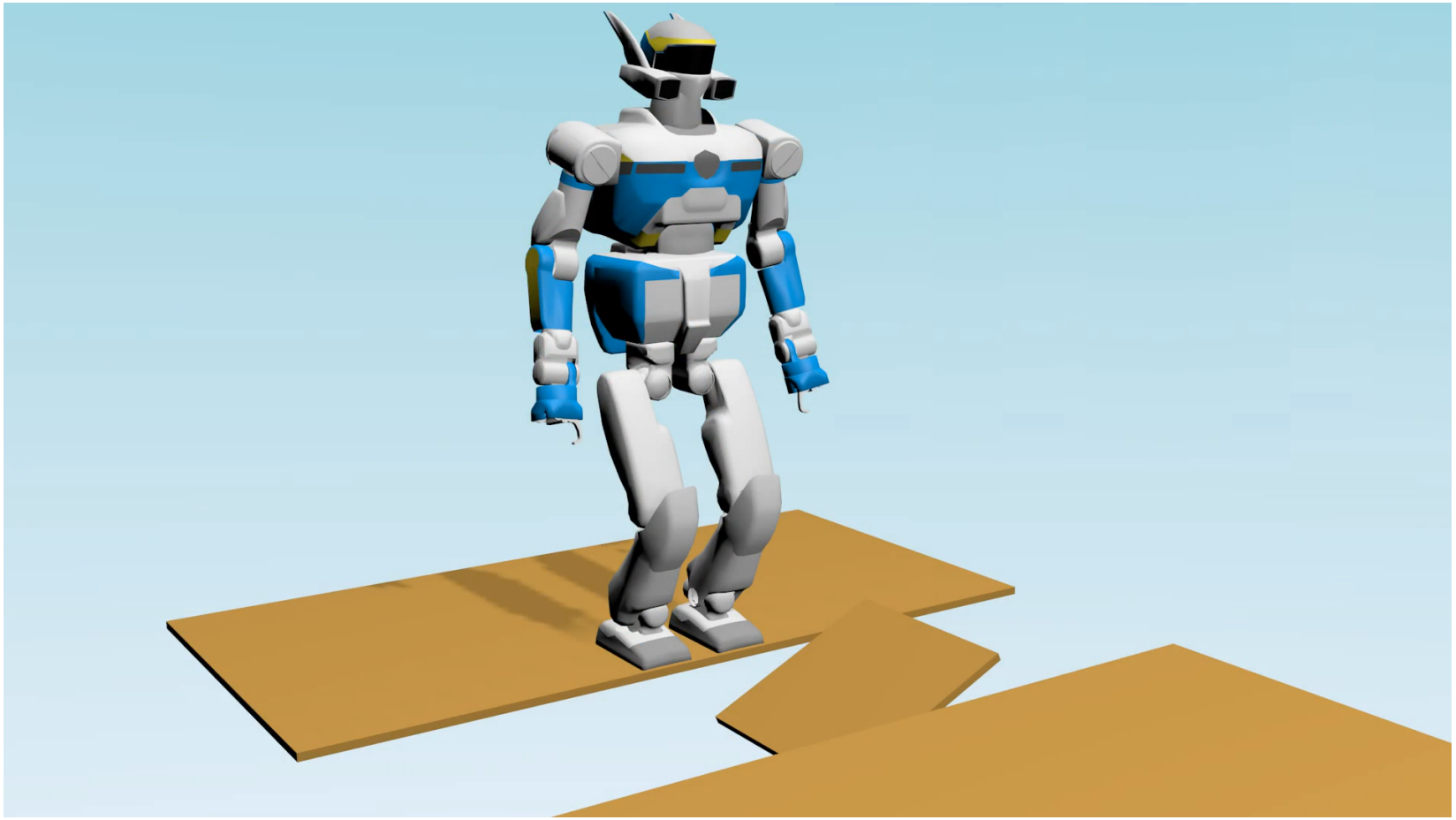
Given:

- Current Contact state (active contacts and locations)
- Possible contact surfaces

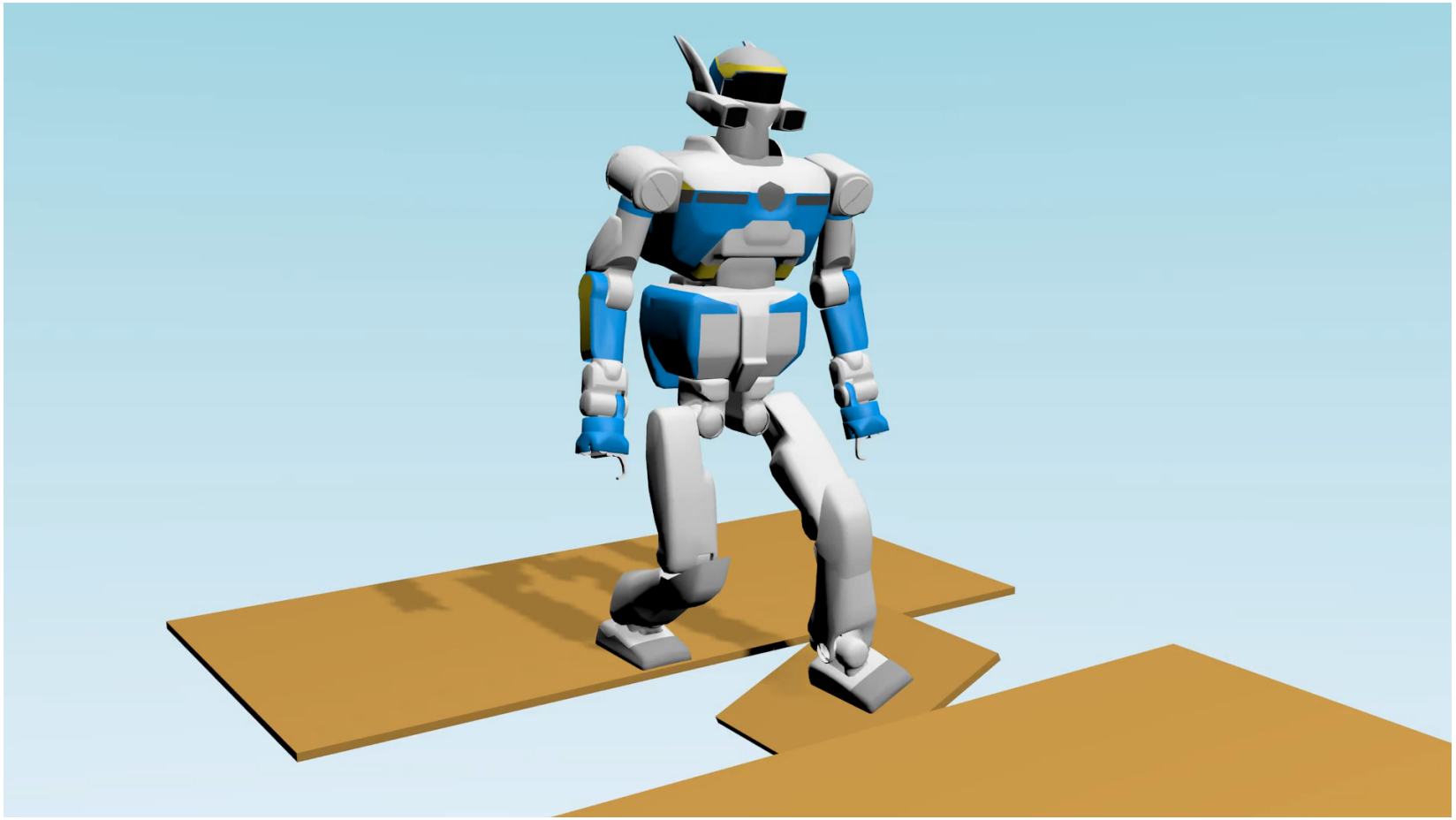
Find:

- Next contact state (active contact and locations)

ONE STEP OUTPUT



ONE STEP OUTPUT



HOW DO WE CHOOSE THE FOOT LOCATION?

HOW DO WE CHOOSE THE FOOT LOCATION?

```
function PutBackFootForward(State s)
    success = False
    r = s.rootPosition() + αx
    while (not success):
        target = SampleForwardContactSurface(r)
        constraint1 = Position(s.backFoot(), target)
        constraint2 = Position(s.frontFoot(), s.frontFoot().position())
        success, state = project(s, r, constraint1, constraint2):
    return state
```

HOW DO WE CHOOSE THE FOOT LOCATION?

```
function PutBackFootForward(State s)
    success = False
    r = s.rootPosition() + αx
    while (not success):
        target = SampleForwardContactSurface(r)
        constraint1 = Position(s.backFoot(), target)
        constraint2 = Position(s.frontFoot(), s.frontFoot().position())
        success, state = project(s, r, constraint1, constraint2):
    return state
```

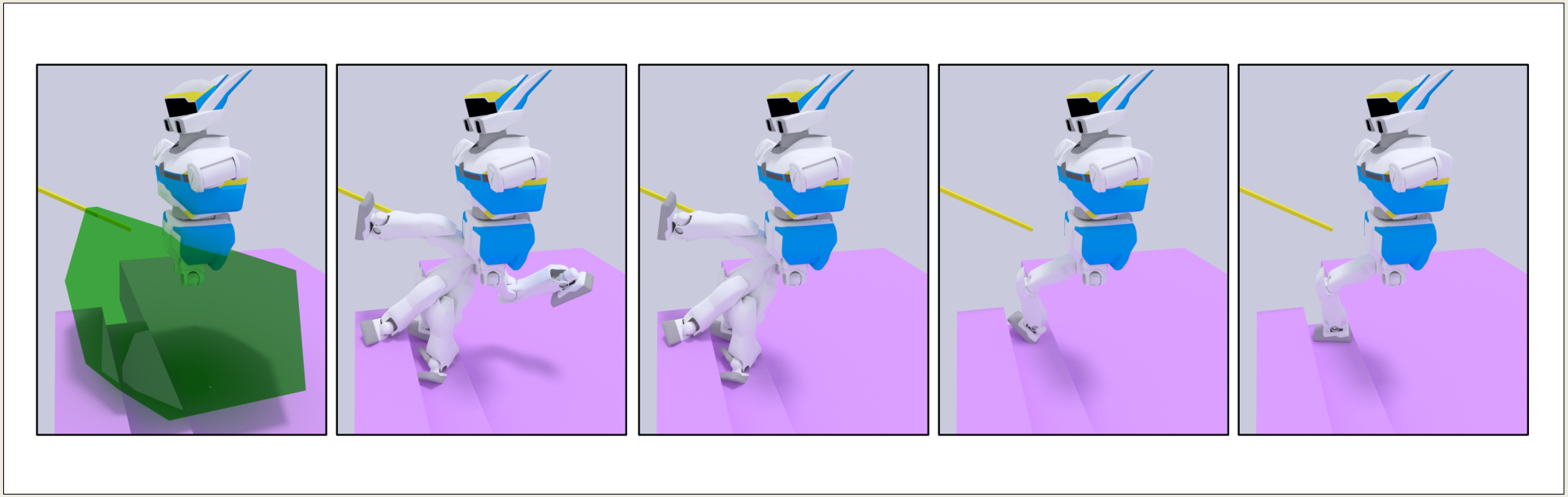
Drawbacks ?

HOW DO WE CHOOSE THE FOOT LOCATION?

```
function PutBackFootForward(State s)
    success = False
    r = s.rootPosition() + αx
    while (not success):
        target = SampleForwardContactSurface(r)
        constraint1 = Position(s.backFoot(), target)
        constraint2 = Position(s.frontFoot(), s.frontFoot().position())
        success, state = project(s, r, constraint1, constraint2):
    return state
```

Drawbacks ?

BIASING THE SAMPLING THROUGH HEURISTICS



Reduce sampling space

Reduce projection cost

ADAPT THE METHOD TO MULTI-CONTACT?

```
function PutBackFootForward(State s)
    success = False
    r = s.rootPosition() + αx
    while (not success):
        target = SampleForwardContactSurface(r)
        constraint1 = Position(s.backFoot(), target)
        constraint2 = Position(s.frontFoot(), s.frontFoot().position())
        success, state = project(s, r, constraint1, constraint2):
    return state
```

ADAPT THE METHOD TO MULTI-CONTACT?

```
function ContactSwitchForward(State s)
    success = False; r = s.rootPosition() + αx
    while (not success):
        nextContact = randomEffector()
        if nextContact ∈ s.activeContacts():
            maintainedContacts = s.activeContacts() \ nextContact
        else
            maintainedContacts = s.activeContacts()
            maintainedContacts = MaybeRemoveRandomContact(maintainedContacts, p)
        target = SampleForwardContactSurface(r)
        constraint1 = Position(nextContact, target)
        constraint2 = Positions(maintainedContacts, maintainedContacts().positions())
        success, state = project(s,r,constraint1, constraint2):
    return state
```

ADAPT THE METHOD TO MULTI-CONTACT?

```
function ContactSwitchForward(State s)
    success = False; r = s.rootPosition() + αx
    while (not success):
        nextContact = randomEffector()
        if nextContact ∈ s.activeContacts():
            maintainedContacts = s.activeContacts() \ nextContact
        else
            maintainedContacts = s.activeContacts()
            maintainedContacts = MaybeRemoveRandomContact(maintainedContacts, p)
        target = SampleForwardContactSurface(r)
        constraint1 = Position(nextContact, target)
        constraint2 = Positions(maintainedContacts, maintainedContacts().positions())
        success, state = project(s,r,constraint1, constraint2):
    return state
```

OUR GENERIC ONE STEP GENERATOR

```
states = []
While(not_ordered_otherwise):
    states.insert(ContactSwitchForward(states.last()))
```

OUR GENERIC ONE STEP GENERATOR

```
states = []
While(not_ordered_otherwise):
    states.insert(ContactSwitchForward(states.last()))
```

Issues ?

OUR GENERIC ONE STEP GENERATOR

```
states = []
While(not_ordered_otherwise):
    states.insert(ContactSwitchForward(states.last()))
```

Issues ?

- Viability of the reached state ?
- Feasibility of the transition ?

Conservative Viability Condition: Static Equilibrium Constraint

Sufficient condition for state viability:

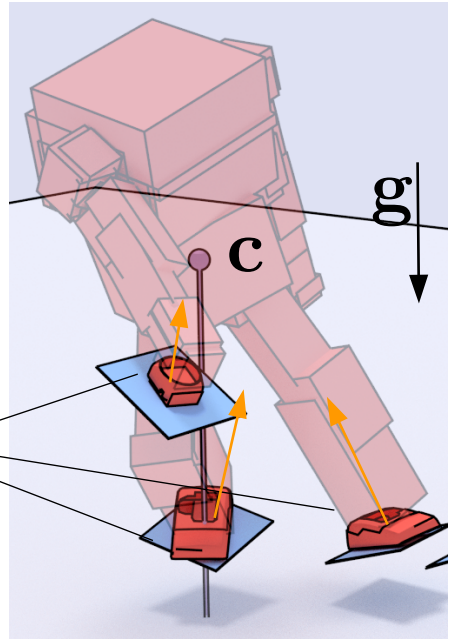
- Posture collision free
- Joint limits respected
- Static equililbrium is ensured:

  There exists a set of contact forces that compensate for gravity interaction while the COM is at 0 velocity

NEWTON EULER EQUATIONS

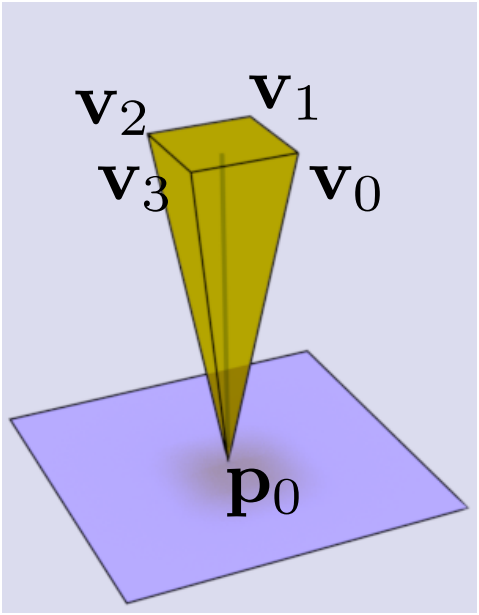
$$\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_{nc} \end{bmatrix} \mathbf{f}$$

$$\mathbf{f} = \left[\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_{nc}\right]^T \in \mathbb{R}^{3nc}$$



LINEARIZED FRICTION CONES CONSTRAINTS

$$\mathrm{f} = \mathrm{V}\ \beta, \text{ where } \mathbf{V} \in \mathbb{R}^{3nc \times 4nc}$$
$$\beta \in \mathbb{R}^{4nc} \succcurlyeq \mathbf{0}$$



STATIC EQUILIBRIUM PROBLEM

For a given $\mathbf{c}^*$, static equilibrium conditions give:

$$\underbrace{\begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_0 & \dots & \hat{\mathbf{p}}_{nc} \end{bmatrix}}_{\mathbf{G}} \mathbf{V}\,\beta = \underbrace{\begin{bmatrix} \mathbf{0}_{3\times 3} \\ m\hat{\mathbf{g}} \end{bmatrix}}_{\mathbf{D}} \mathbf{c}^* + \underbrace{\begin{bmatrix} -m\mathbf{g} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{d}}$$

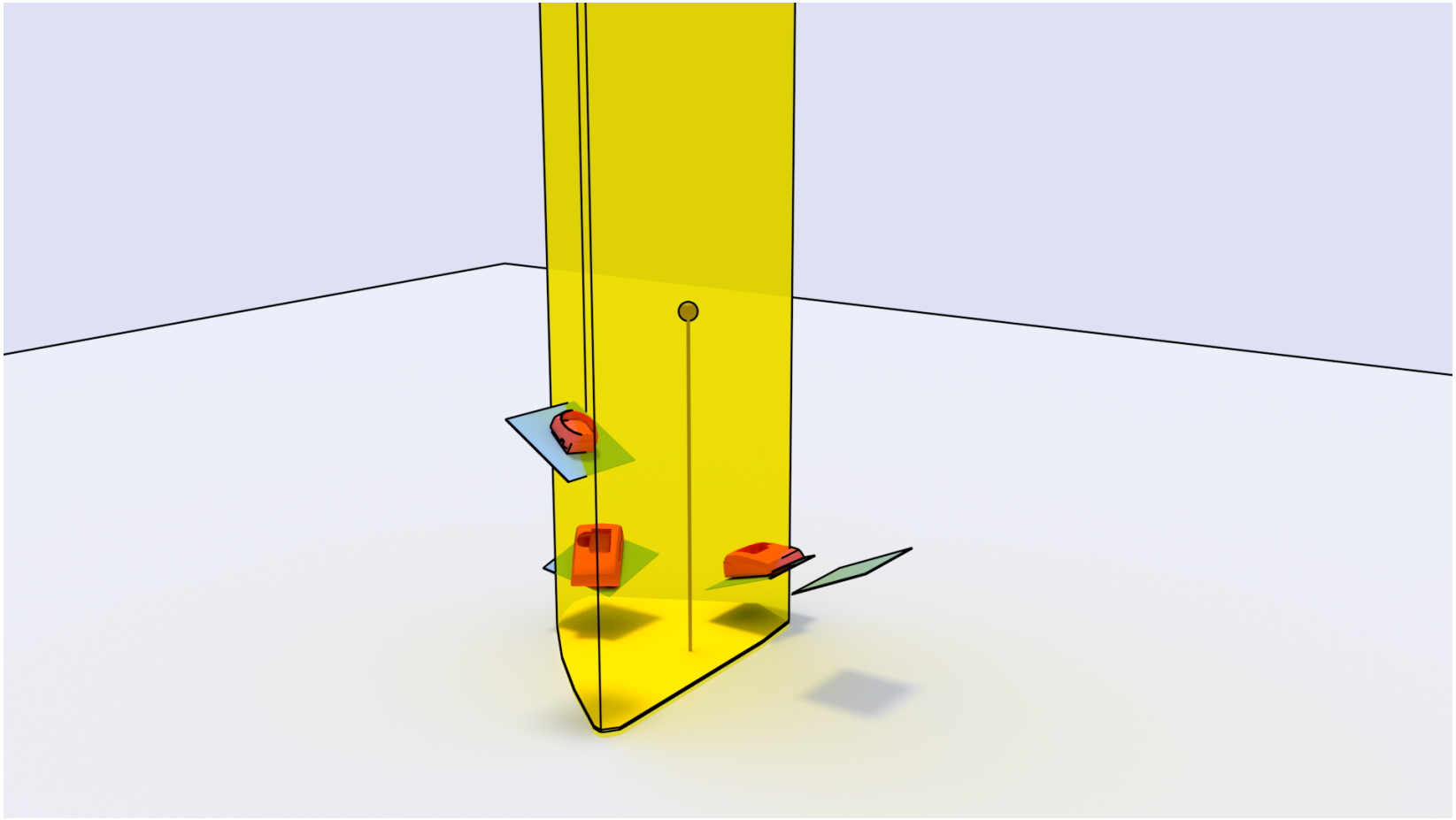
Feasiblity solved by the LP [1]:

$$\begin{aligned} \text{find} \quad & \beta \in \mathbb{R}^{4nc} \\ \text{s.t.} \quad & \mathbf{G}\beta = \mathbf{D}\mathbf{c}^* + \mathbf{d} \end{aligned}$$

[1] Del Prete et al., Fast Algorithms to Test Robust Static Equilibrium for Legged Robots, ICRA 16
[2] Fernbach et al., A Kinodynamic steering-method for legged multi-contact locomotion, IROS 17

CONE AND REPRESENTATION

Solution space is a 3D cone, (6D in general) [1,2]



[1] Qiu et al., Human motions analysis and simulation based on a general criterion of stability

[2] Caron et al., Leveraging Cone Double Description for Multi-contact Stability of Humanoids [...]

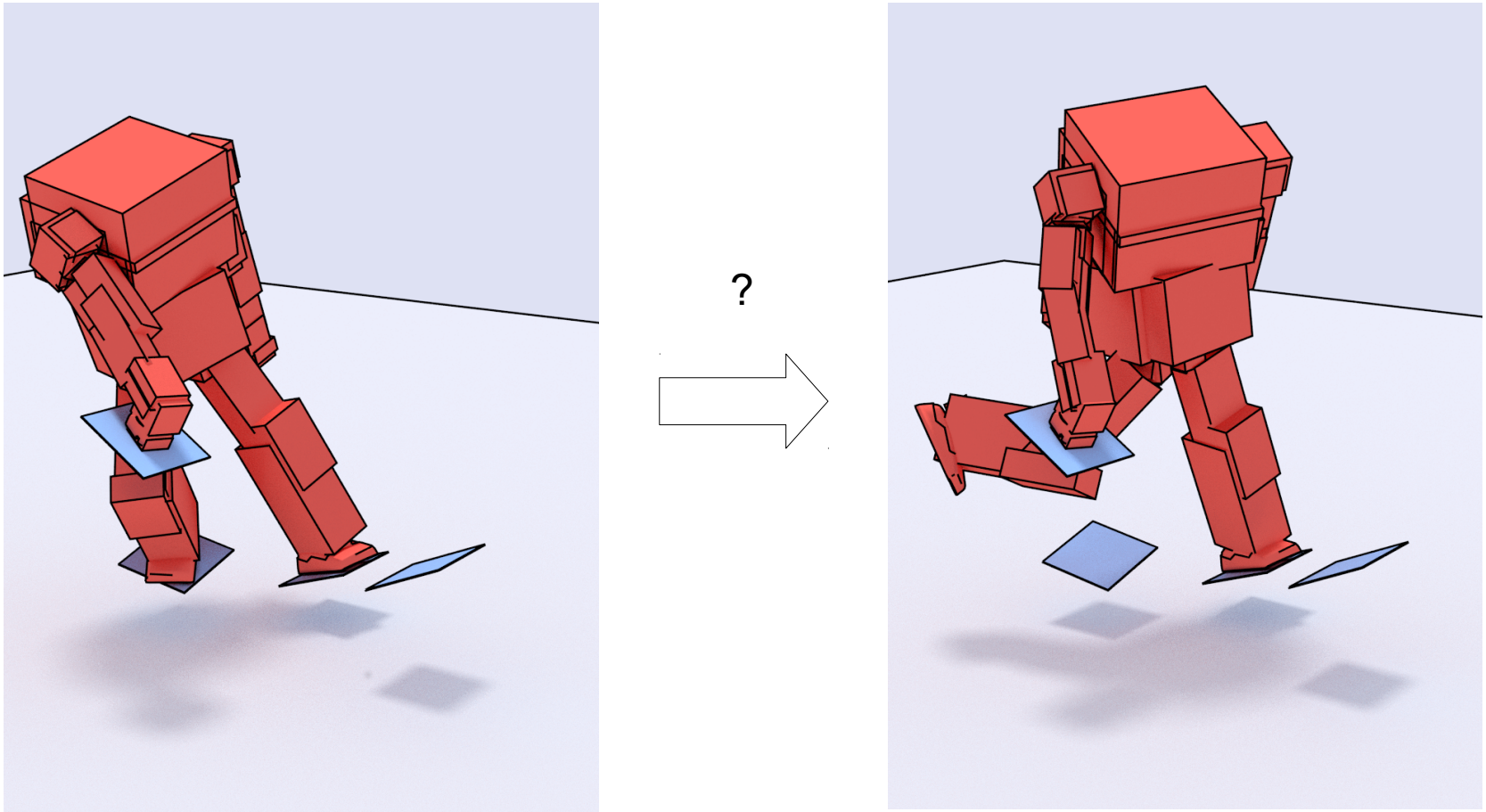
OUR GENERIC ONE STEP GENERATOR

```
states = []
While(not_ordered_otherwise):
    State s = ContactSwitchForward(states.last()))
    if Viable(s):
        states.insert(s)
```

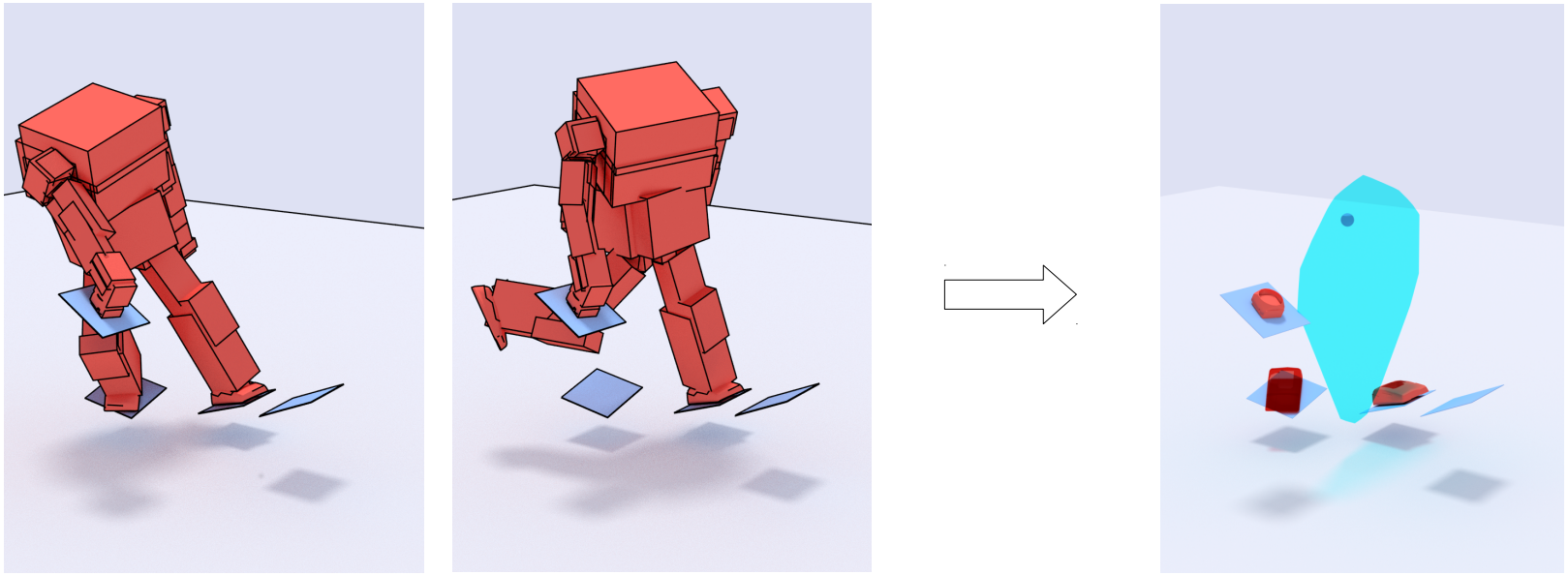
Issues ?

- Viability of the reached state ? ✓
- Feasibility of the transition ?

THE FEASIBILITY TRANSITION PROBLEM



CHECK FEASIBILITY AS EFFICIENTLY AS POSSIBLE

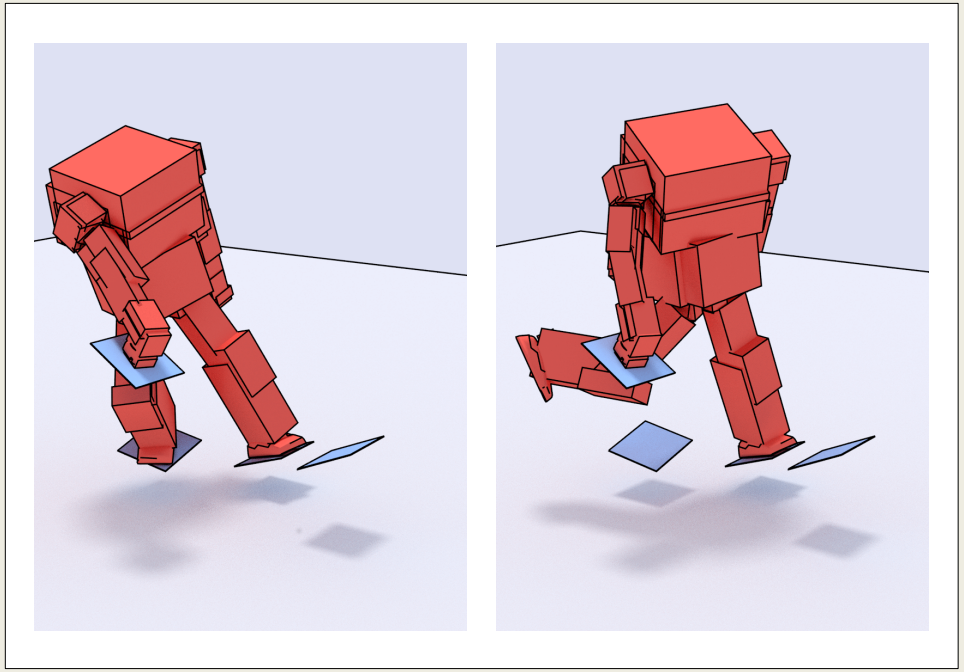


For instance, as a 3D linear program

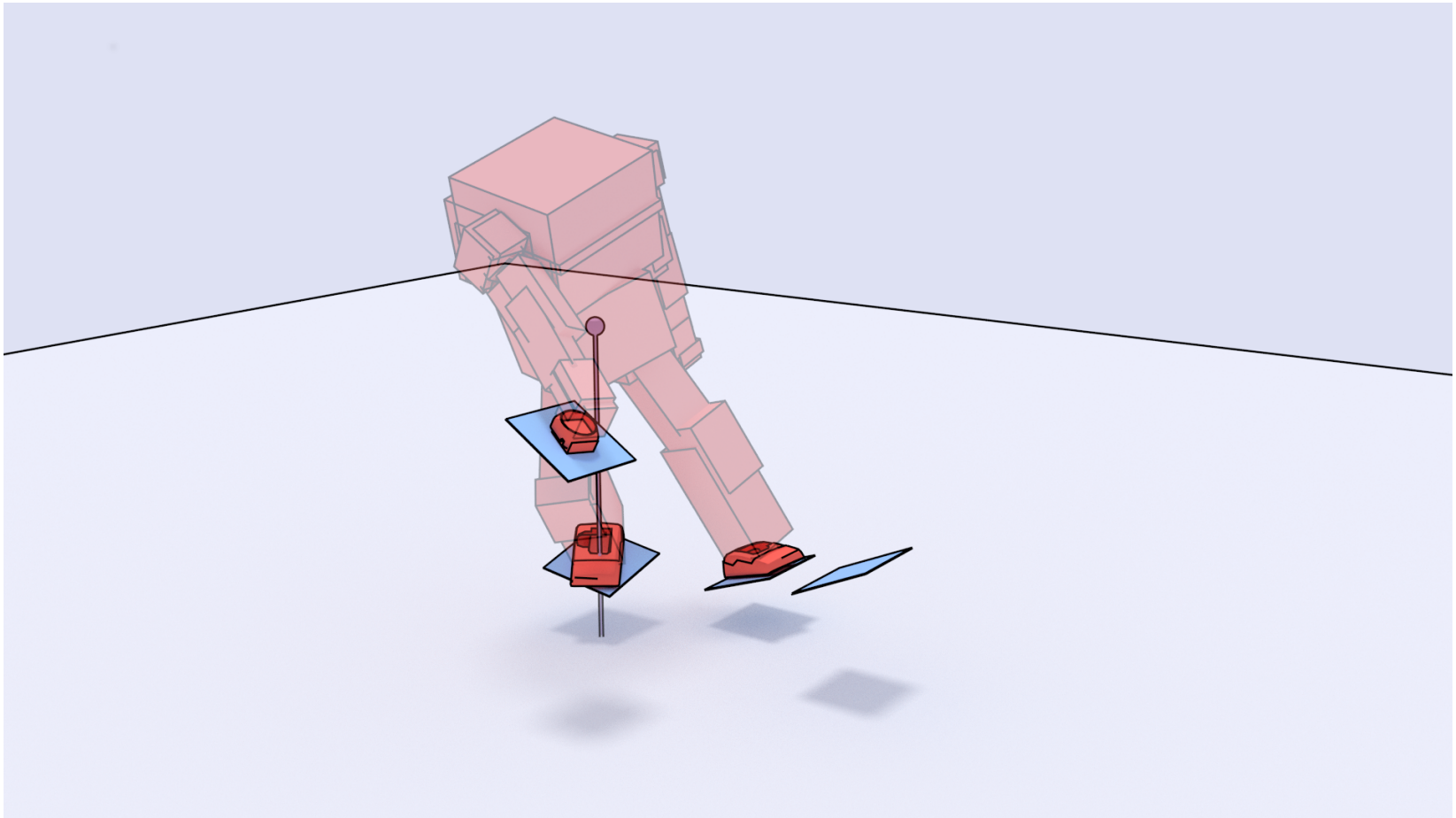
REDUCTION PROPERTY: THE TRANSITION INSTANT

Create / Break contact without:

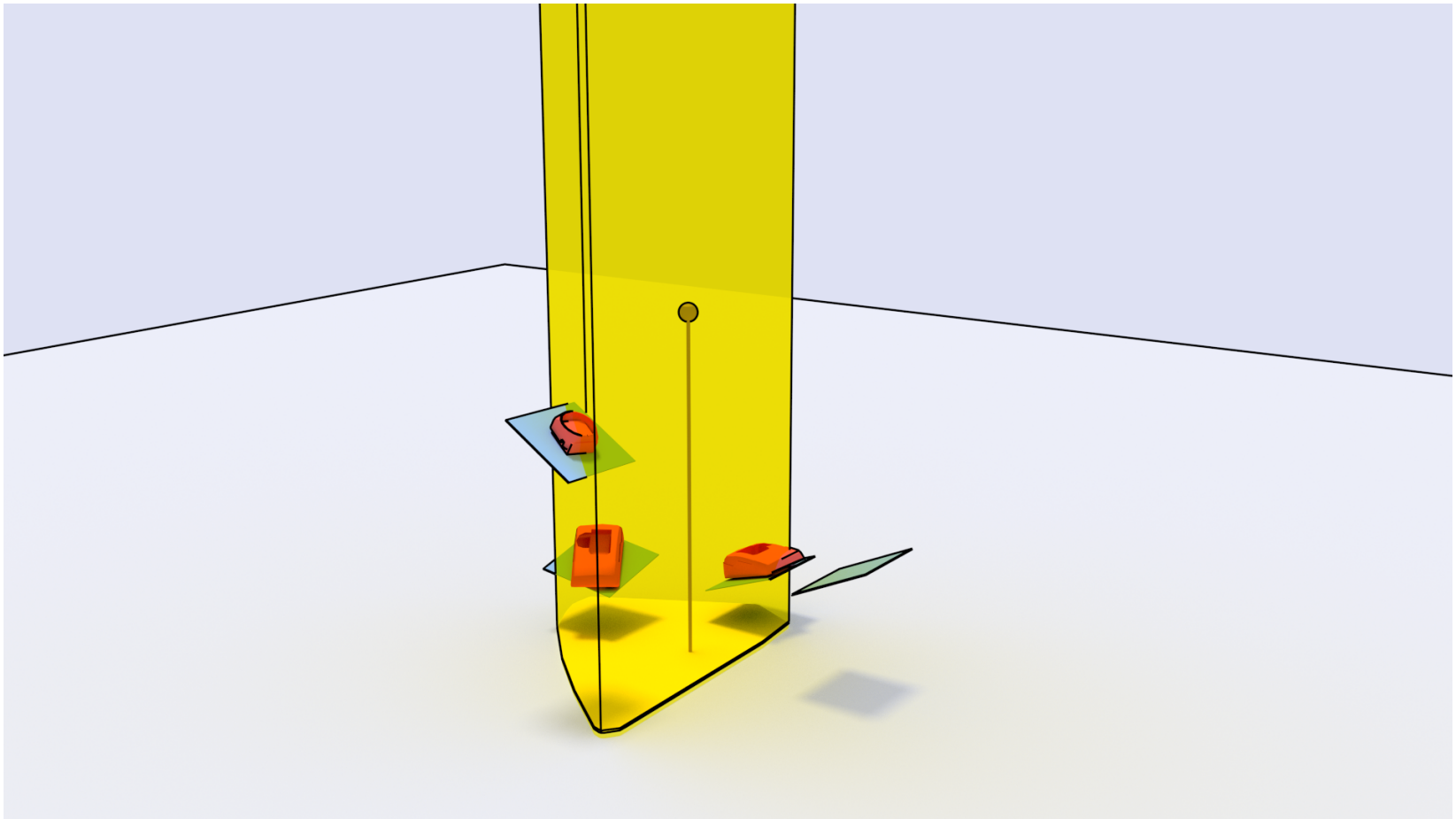
- Falling
- Violating geometric constraints



CONSTRAINTS EXPRESSED AT THE COM

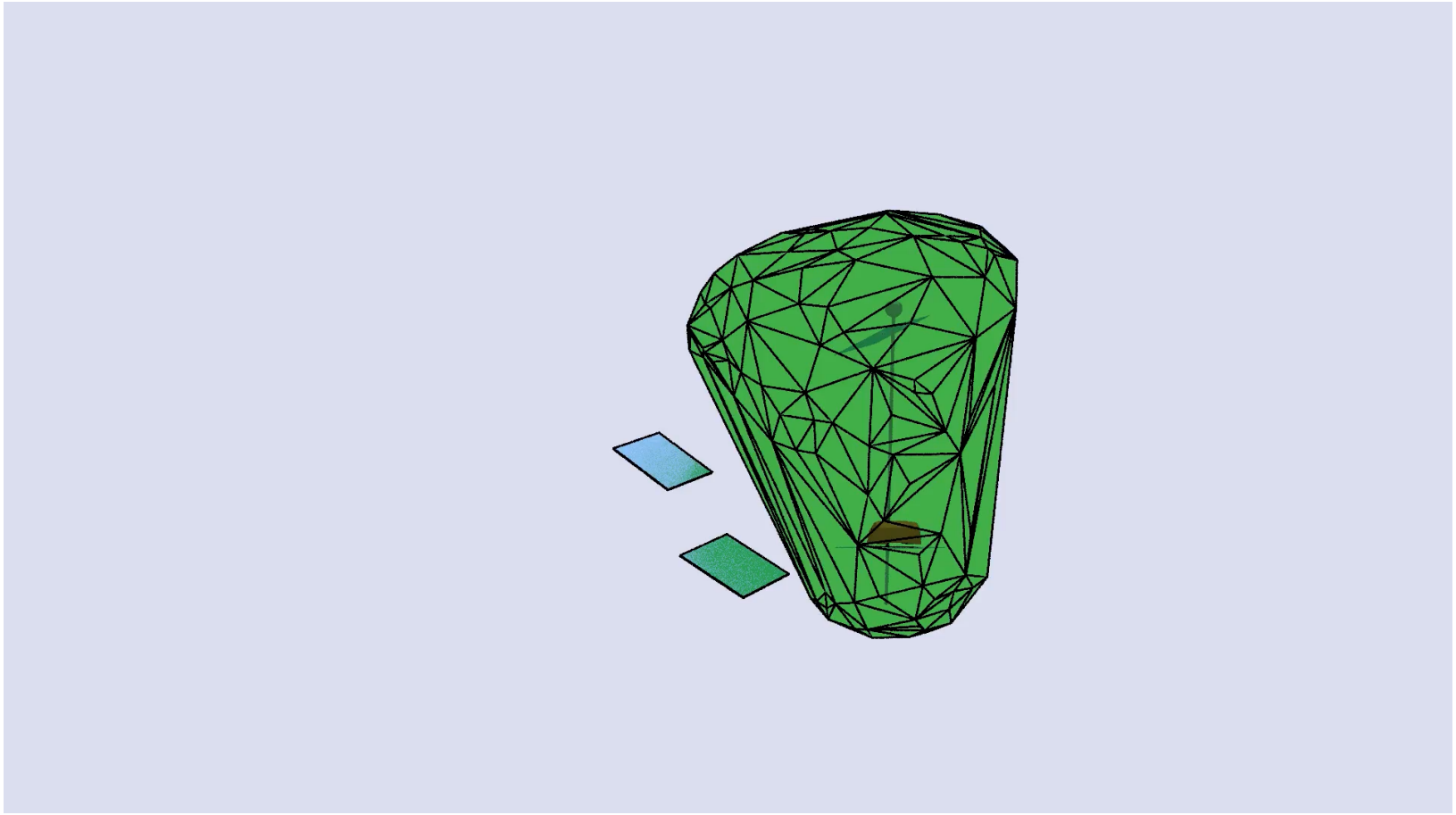


DYNAMIC FEASIBILITY (STATIC FOR NOW)

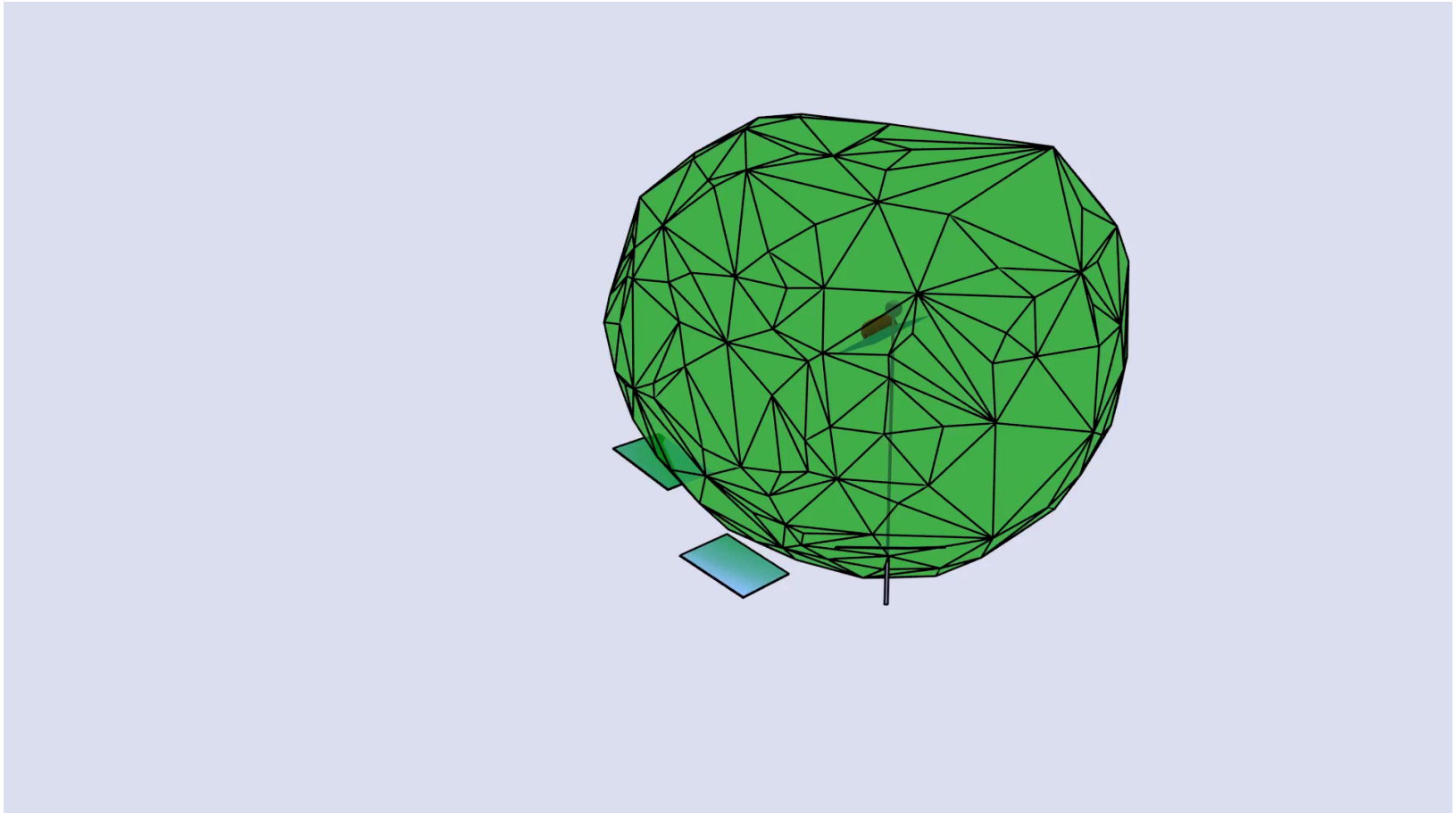


$\mathbf{Ac} \preceq \mathbf{a}$

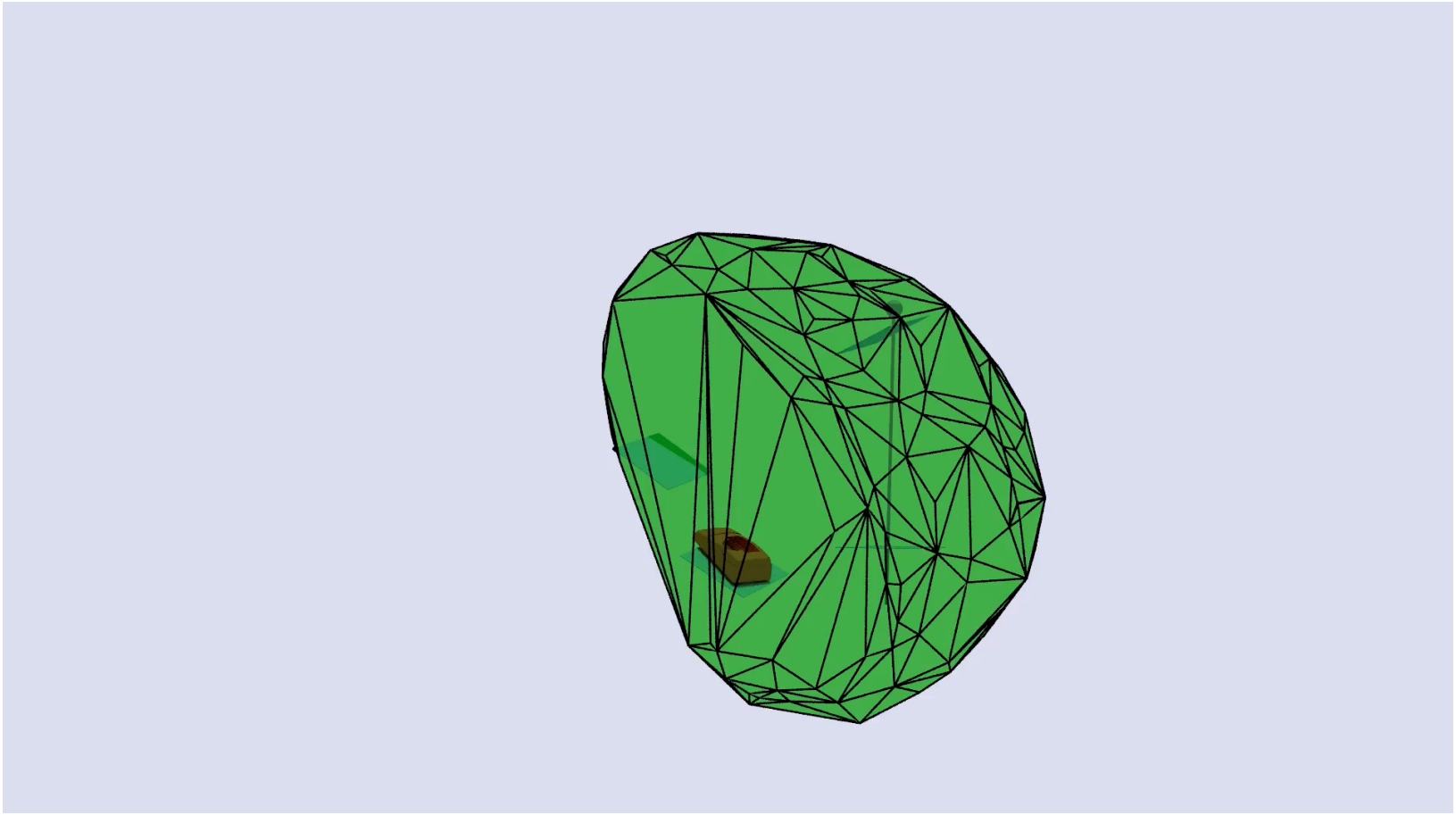
KINEMATIC FEASIBILITY FOR RIGHT LEG...



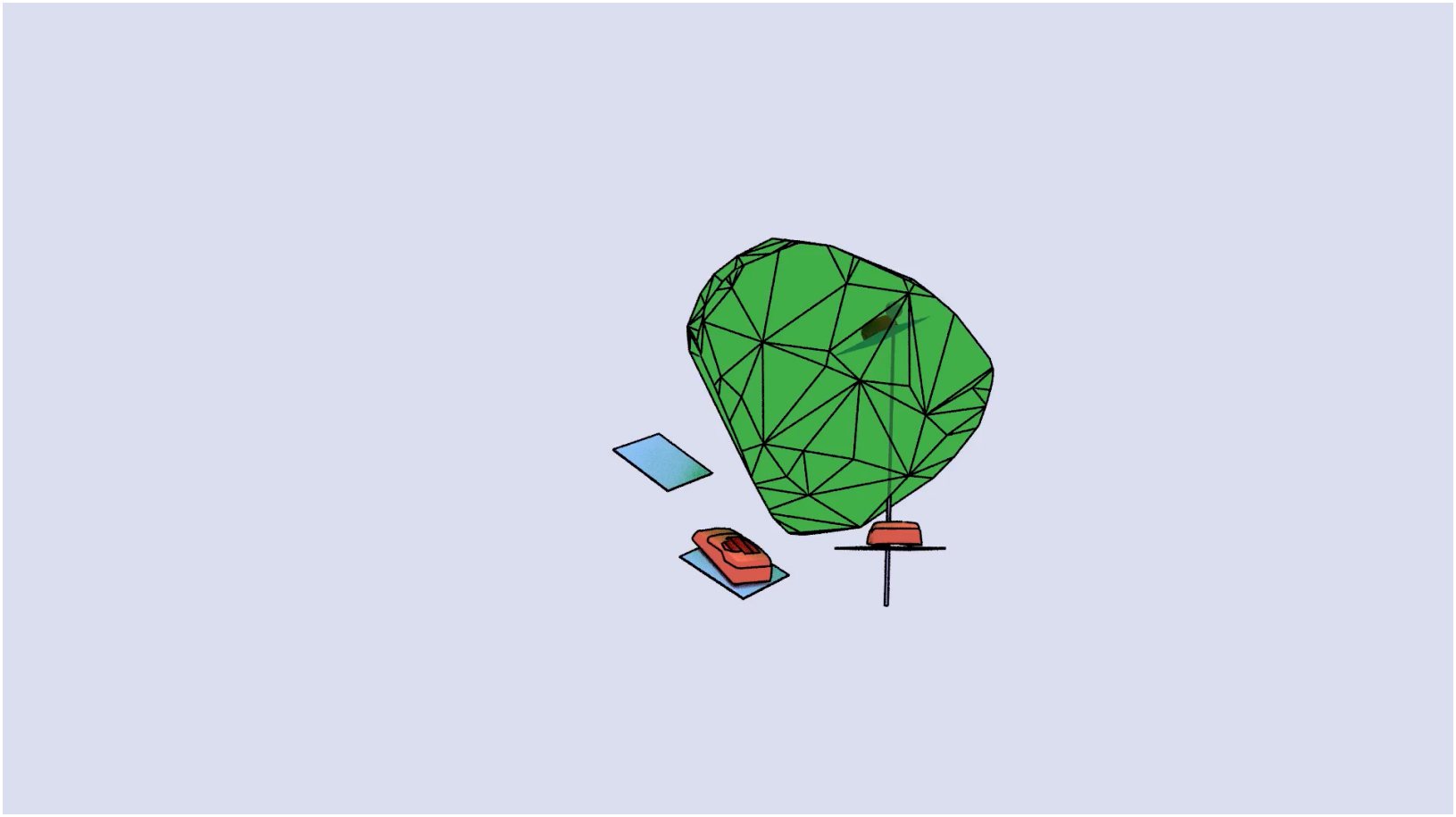
KINEMATIC FEASIBILITY FOR THE RIGHT ARM...



KINEMATIC FEASIBILITY FOR THE LEFT LEG...

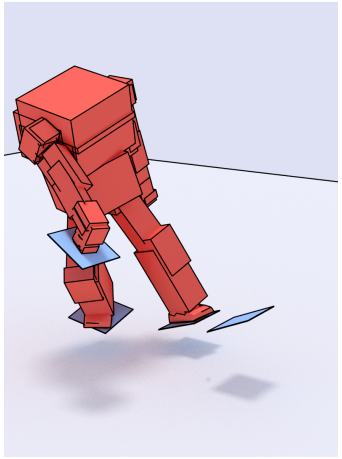


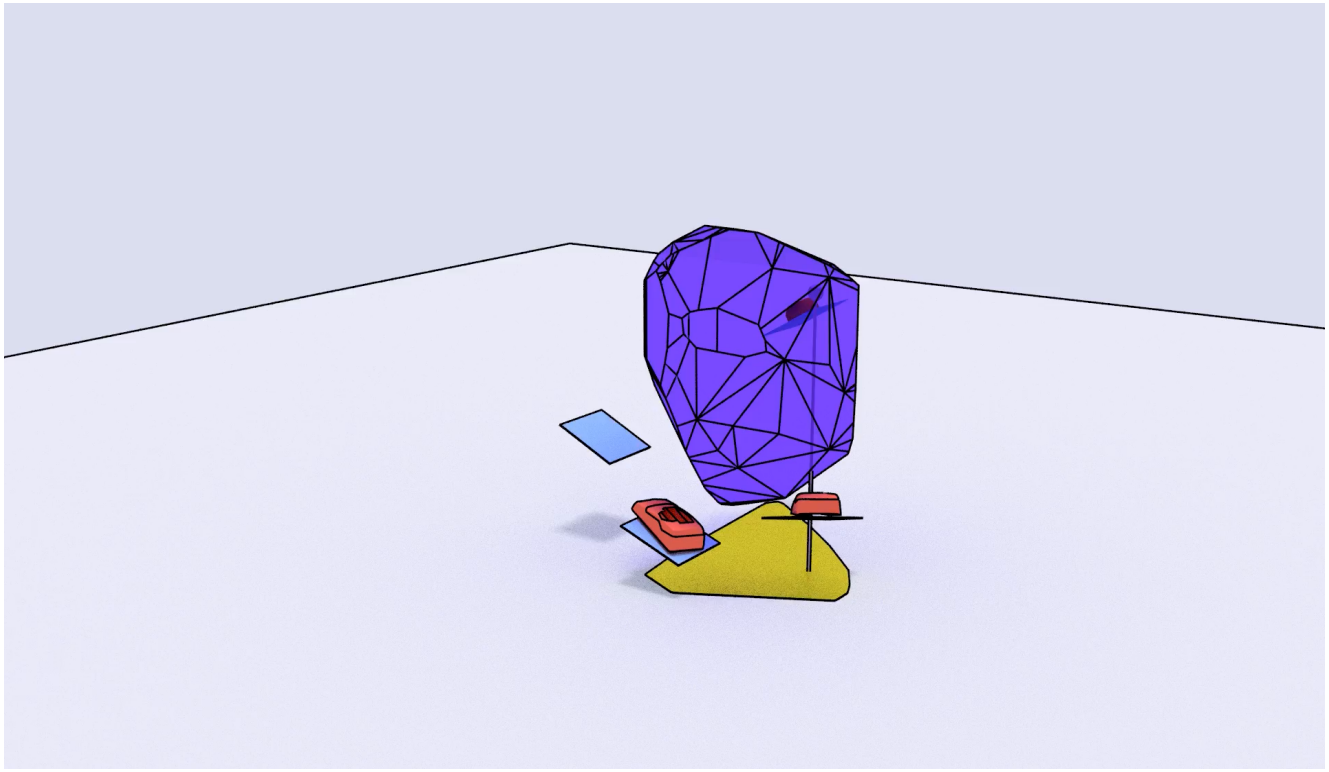
NECESSARY KINEMATIC CONDITION FOR FEASIBILITY GIVEN BY INTERSECTION



$$\mathbf{K}\mathbf{c} \preceq \mathbf{k}$$

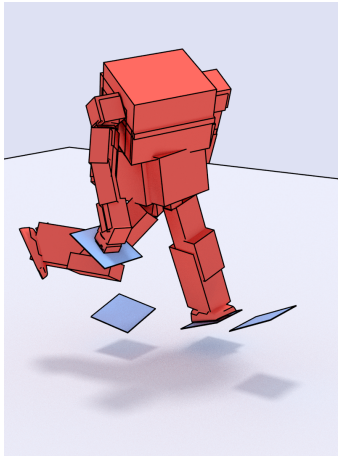
BOTH KINEMATIC AND EQUILIBRIUM FEASIBILITY => GREEN AND YELLOW POYTOPE INTERSECTION

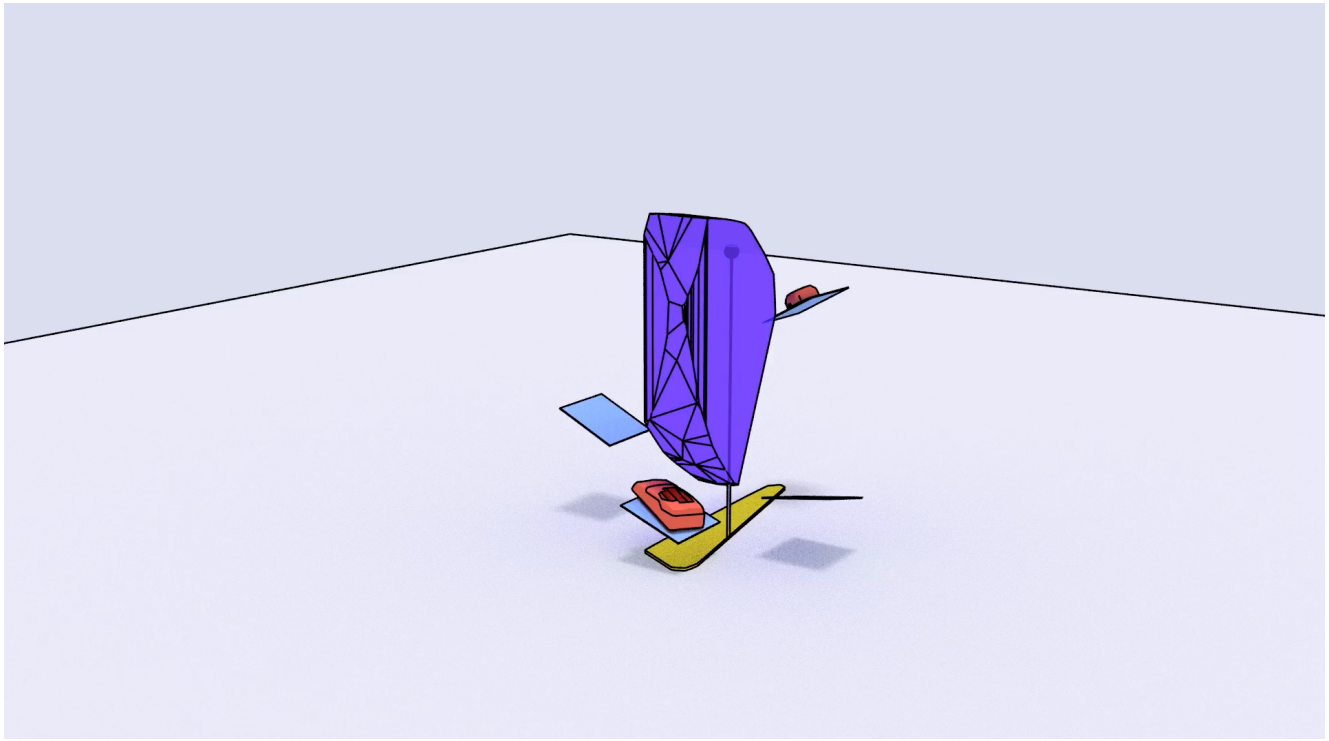




$$\mathbf{Ac} \preceq \mathbf{a} \wedge \mathbf{Kc} \preceq \mathbf{k} \Rightarrow \mathbf{Dc} \preceq \mathbf{d}$$

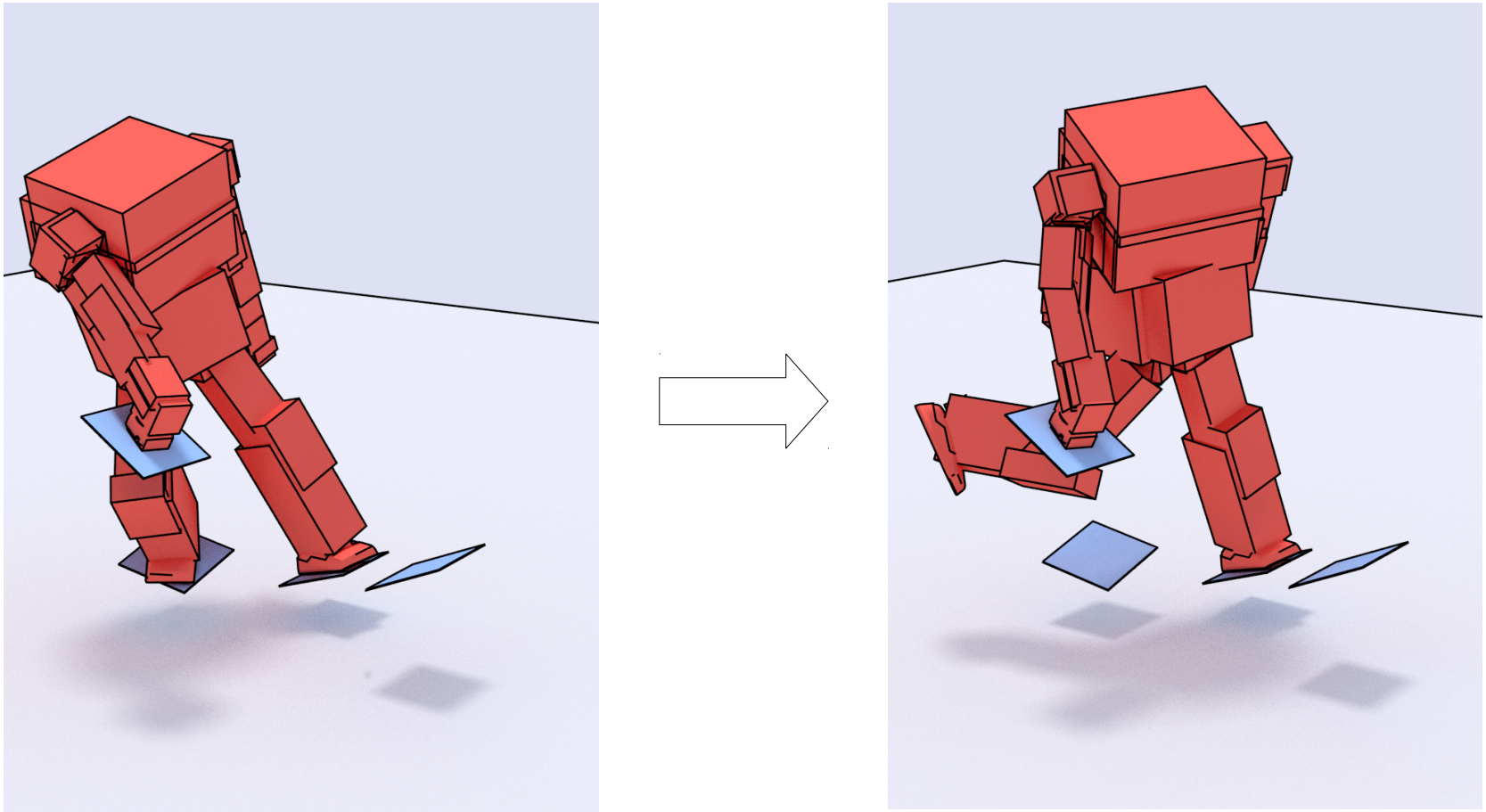
WE REPEAT THE OPERATION FOR PHASE 2



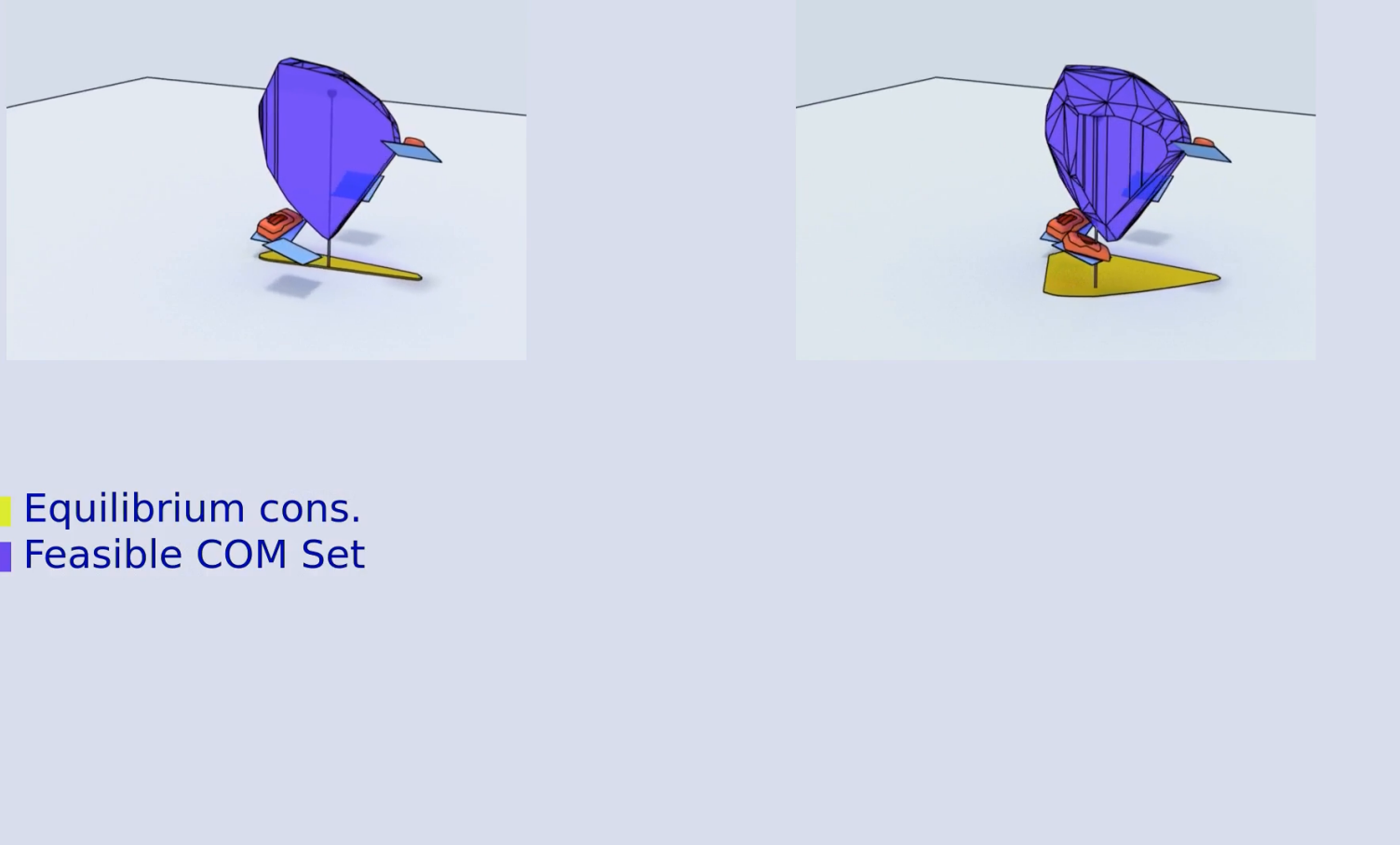


$$\mathbf{D}^2\mathbf{c} \preceq \mathbf{d}^2$$

WHAT CONSTRAINTS FOR THE TRANSITION INSTANT ?

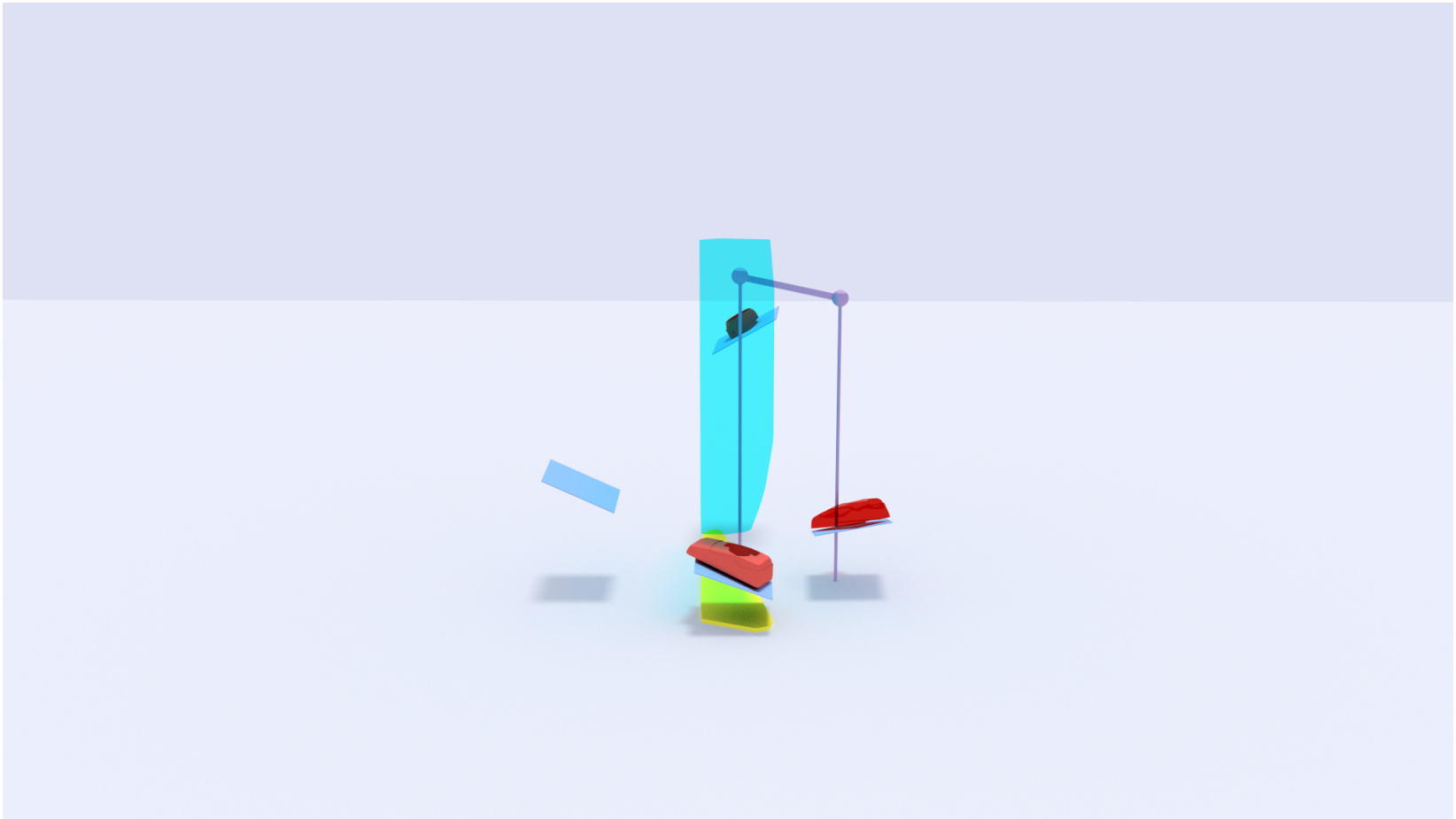


TRANSITION FEASIBILITY GIVEN BY INTERSECTION

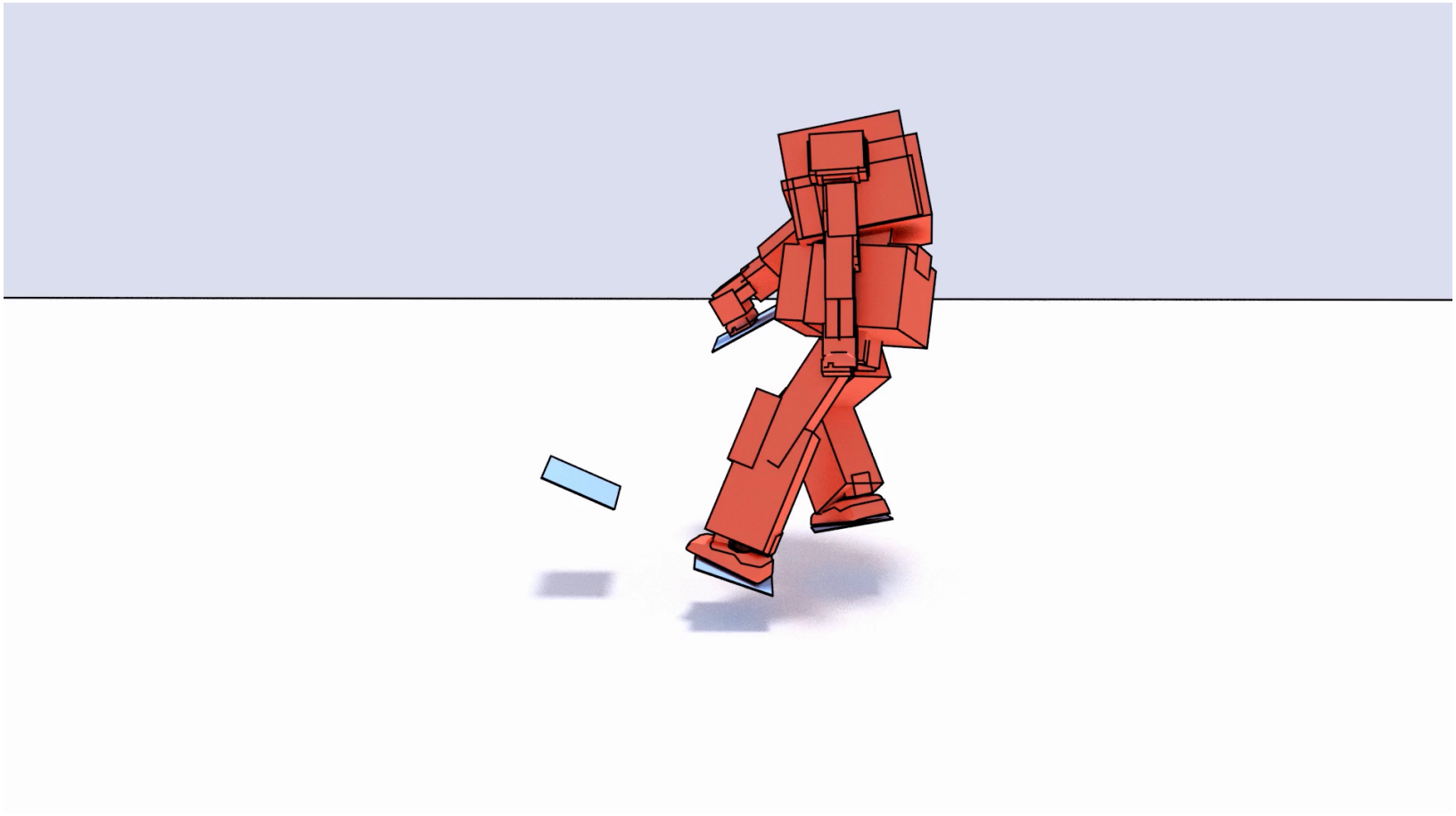


$$\mathbf{Dc} \preceq \mathbf{d} \wedge \mathbf{D}^2\mathbf{c} \preceq \mathbf{d}^2 \Rightarrow \mathbf{E}^{\{1,2\}}\mathbf{c} \preceq \mathbf{e}^{\{1,2\}}$$

BY CONVEXITY, STRAIGHT LINES ARE FEASIBLE TRAJECTORIES!



THIS RESULTS IN A QUASI STATIC MOTION



WHY ARE WE HAPPY?

A feasible trajectory $\Leftrightarrow$ the blue set is defined

The blue set defines a <u>reduction property</u>

$$\begin{aligned} &\text{find} \quad \mathbf{c} \in \mathbb{R}^3 \\ &\quad \text{s.t.} \ \mathbf{E}^{\{1,2\}}\mathbf{c} \preceq \mathbf{e}^{\{1,2\}} \end{aligned}$$

Any point in the blue set defines a feasible trajectory!

[1] Tonneau et al., 2PAC: 2 Point Attractors for Center of mass trajectories in multi-contact scenarios

WHY ARE WE HAPPY?

A feasible trajectory $\Leftrightarrow$ the blue set is defined

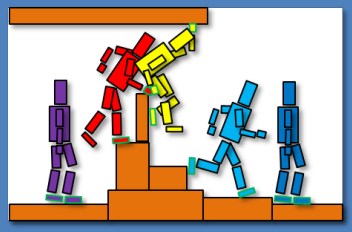
The blue set defines a <u>reduction property</u>

$$\begin{aligned} &\text{find} \quad \mathbf{c} \in \mathbb{R}^3 \\ &\quad \text{s. t.} \ \mathbf{E}^{\{1,2\}}\mathbf{c} \preceq \mathbf{e}^{\{1,2\}} \end{aligned}$$

Any point in the blue set defines a feasible trajectory!

Issues ?

[1] Tonneau et al., 2PAC: 2 Point Attractors for Center of mass trajectories in multi-contact scenarios



CAN WE REASON SIMILARLY FOR THE GENERAL DYNAMIC CASE?

Quasi-static case ⇒ Dynamic case

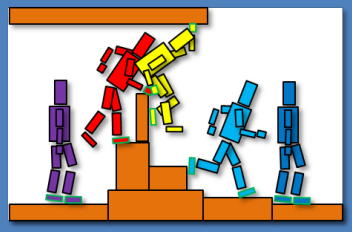
$$\mathbf{A}\mathbf{c} \preceq \mathbf{a}$$

$$\mathbf{H} \underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} \leq \mathbf{h}$$

Non linear

$$\mathbf{c}(t) \in \mathbb{R}^3$$

$$\mathbf{w}(t) \in \mathbb{R}^6$$



CAN WE REASON SIMILARLY FOR THE GENERAL DYNAMIC CASE?

Quasi-static case ⇒ Dynamic case

$$\mathbf{A}\mathbf{c} \preceq \mathbf{a}$$

$$\mathbf{H}\underbrace{\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix}}_{\mathbf{w}} \leq \mathbf{h}$$

Non linear

$$\mathbf{c}(t) \in \mathbb{R}^3 \qquad \mathbf{w}(t) \in \mathbb{R}^6$$

Remove non linearity with a conservative trick[1,2]

(if there's time and interest)

[1] Fernbach et al., CROC: Convex Resolution Of Centroidal dynamics trajectories[…], IROS 17

[2] Fernbach et al., C-CROC: Continuous and Convex Resolution Of Centroidal dynamics trajectories[…]

SO WHERE ARE WE ?

```
states = []
While(not_ordered_otherwise):
    State s = ContactSwitchForward(states.last()))
    if Viable(s) and TransitionFeasible(s, states.last()) :
        states.insert(s)
```

Issues ?

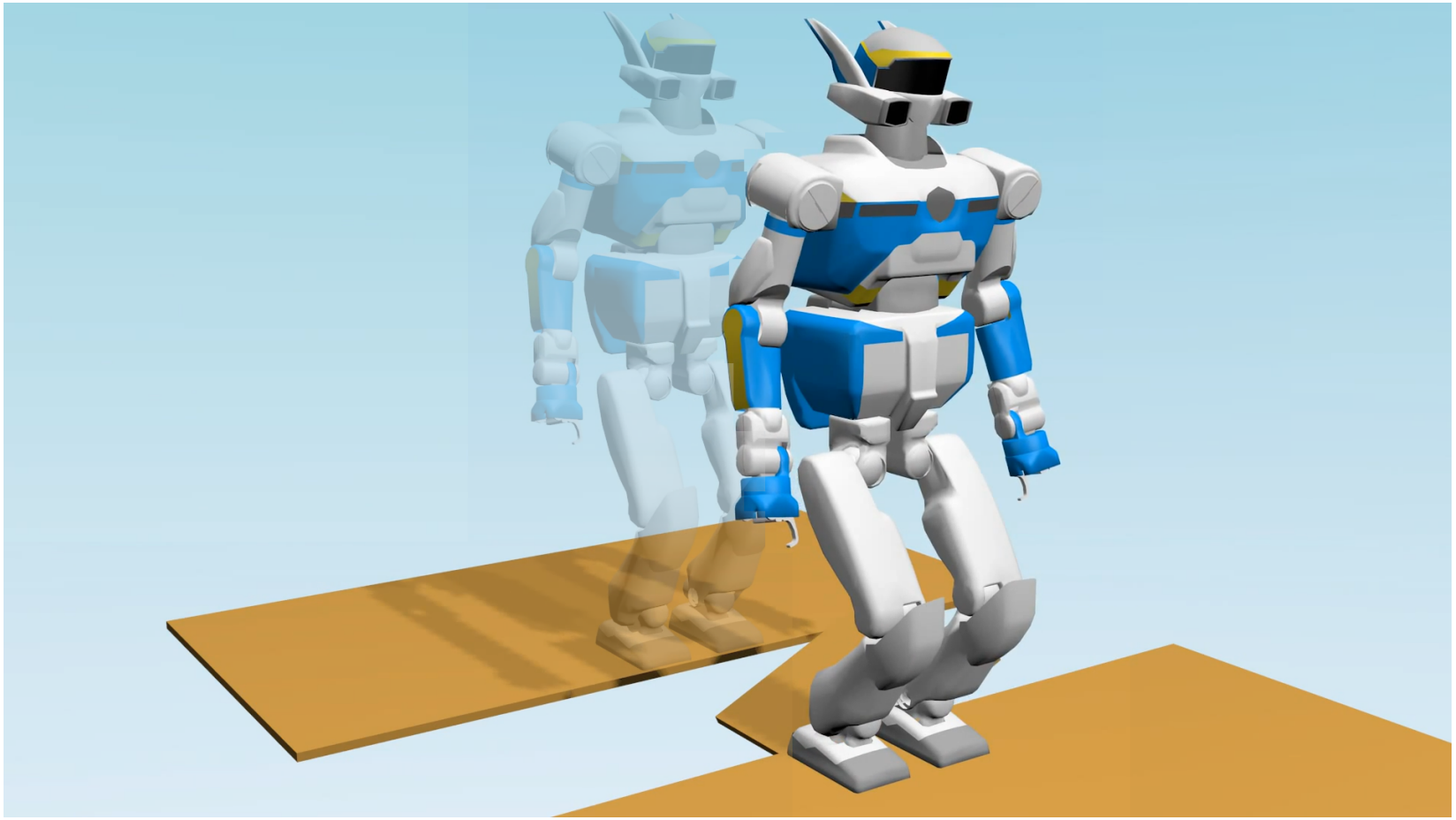
SO WHERE ARE WE ?

```
states = []
While(not_ordered_otherwise):
    State s = ContactSwitchForward(states.last()))
    if Viable(s) and TransitionFeasible(s, states.last()) :
        states.insert(s)
```

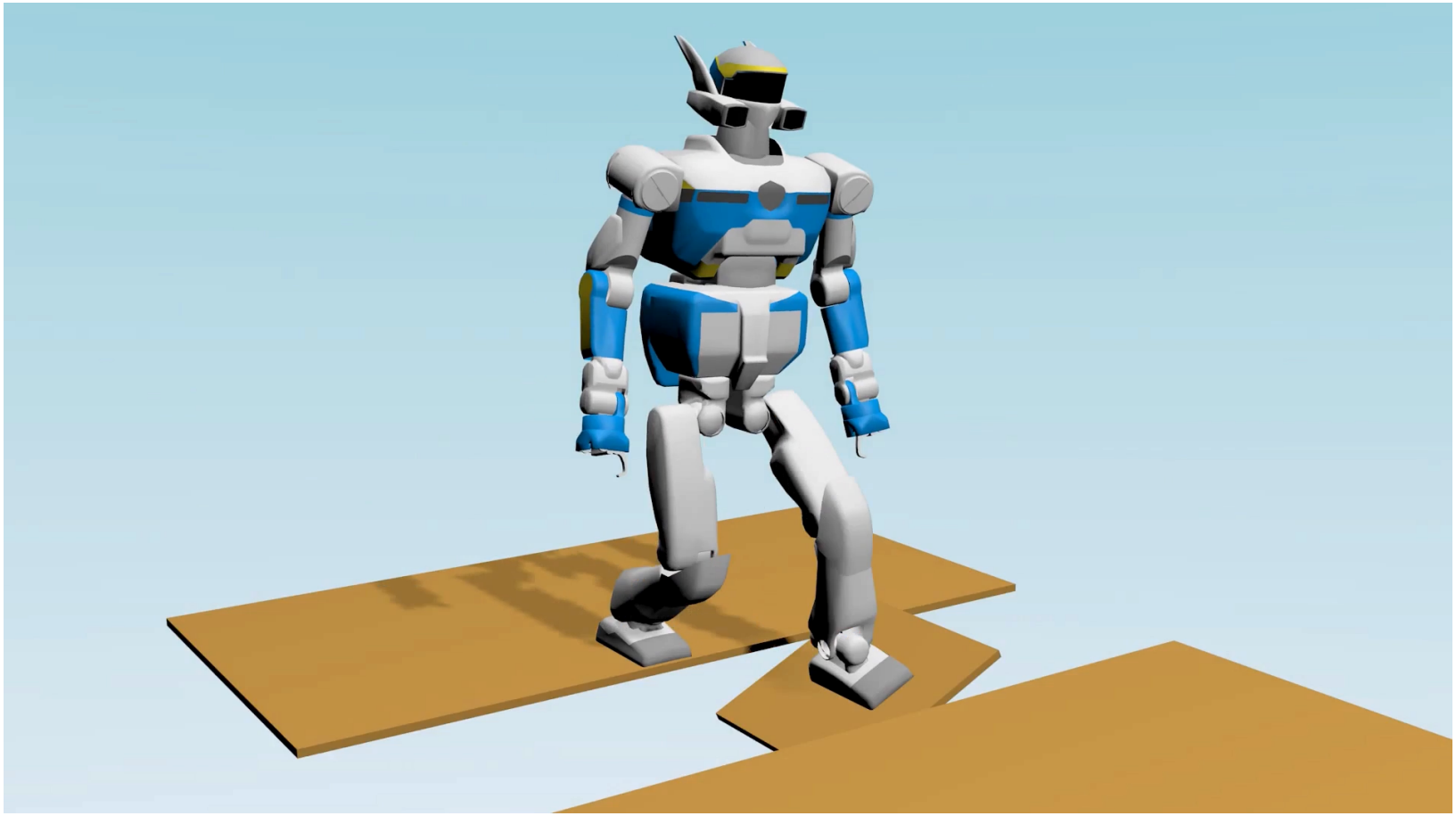
Issues ?

- Kinematic feasibility
- Angular momentum
- Collisions ?

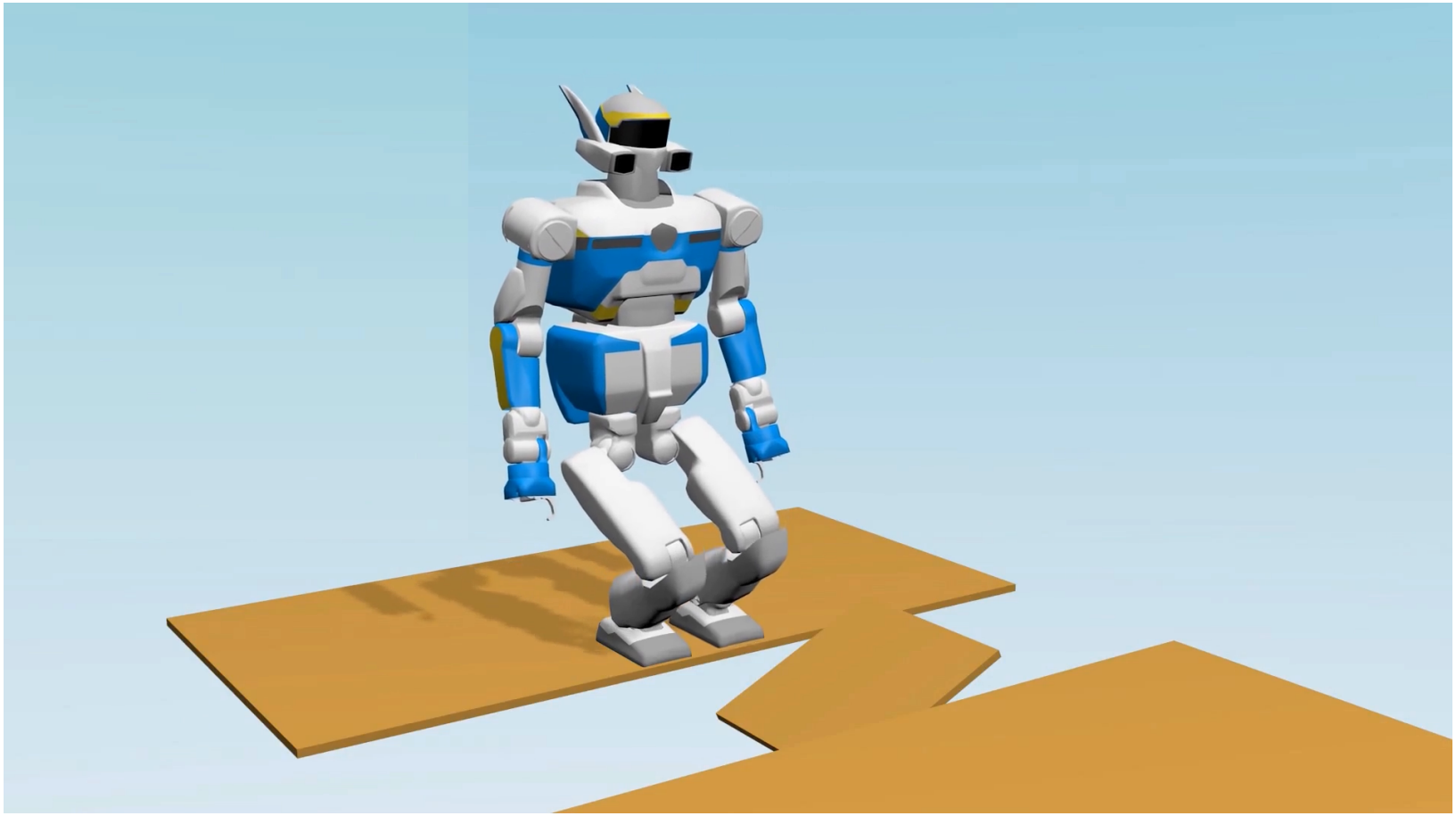
WALK STRAIGHT FOR ABOUT 1 METER



THE MOST LIKELY CONTACT SEQUENCE...

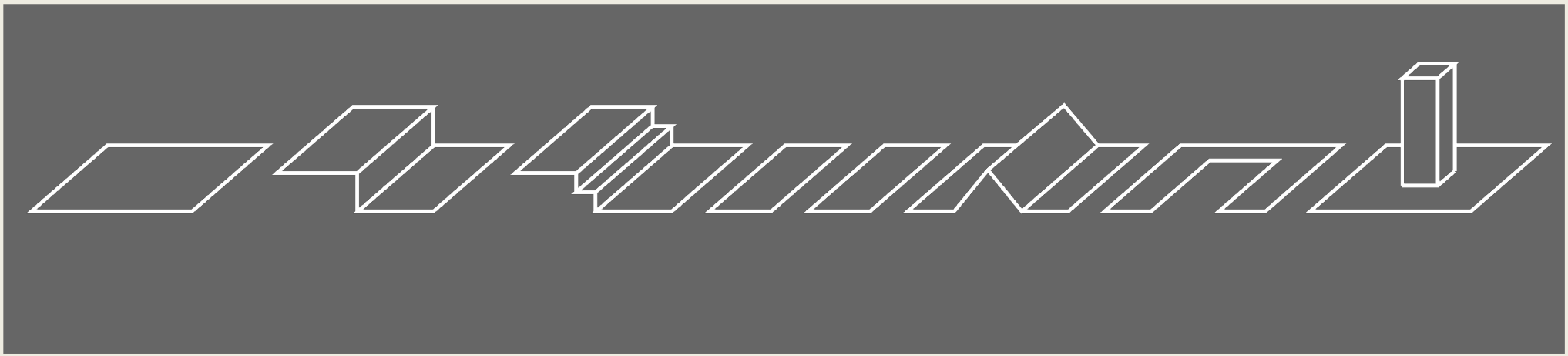


... RESULTS IN THE FOLLOWING MOTION

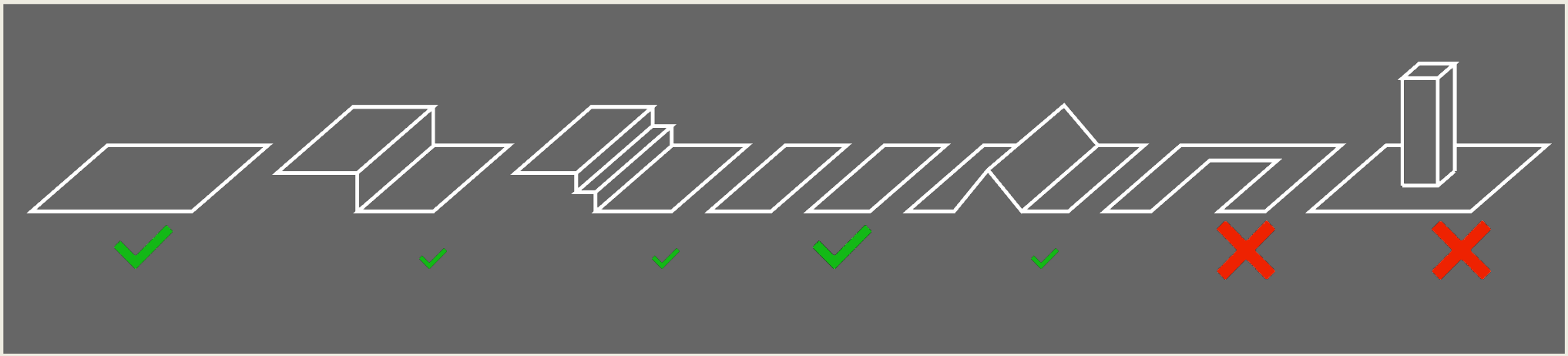


II. WHO GIVES THE ORDERS ?

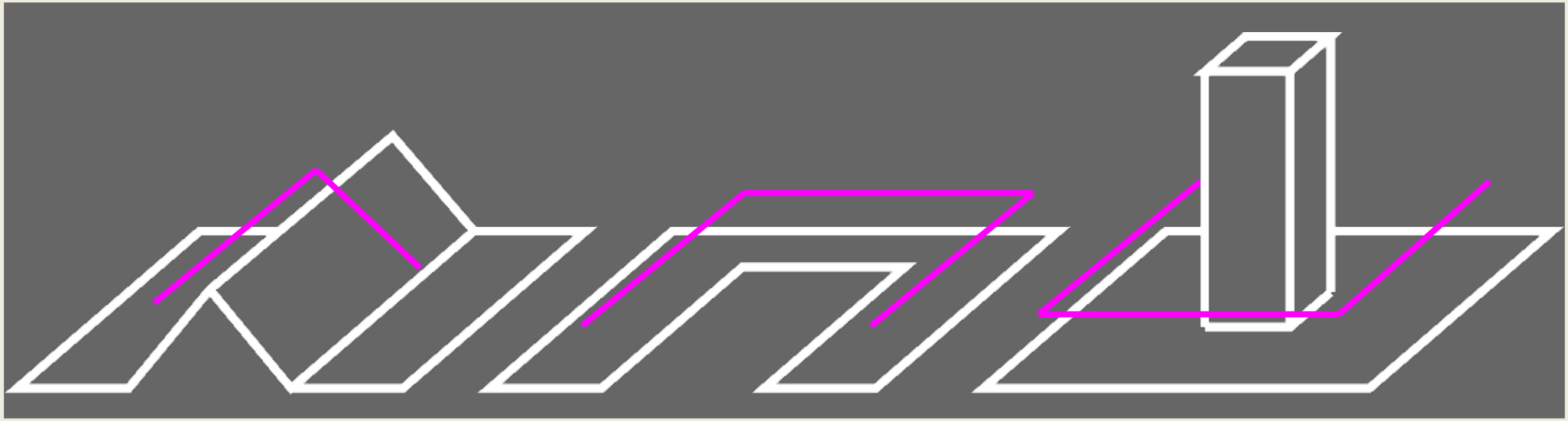
WHAT ABOUT MORE COMPLEX SCENES ?

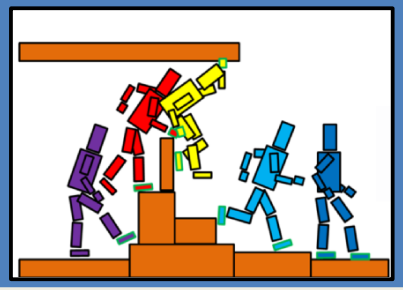


WHAT ABOUT MORE COMPLEX SCENES ?

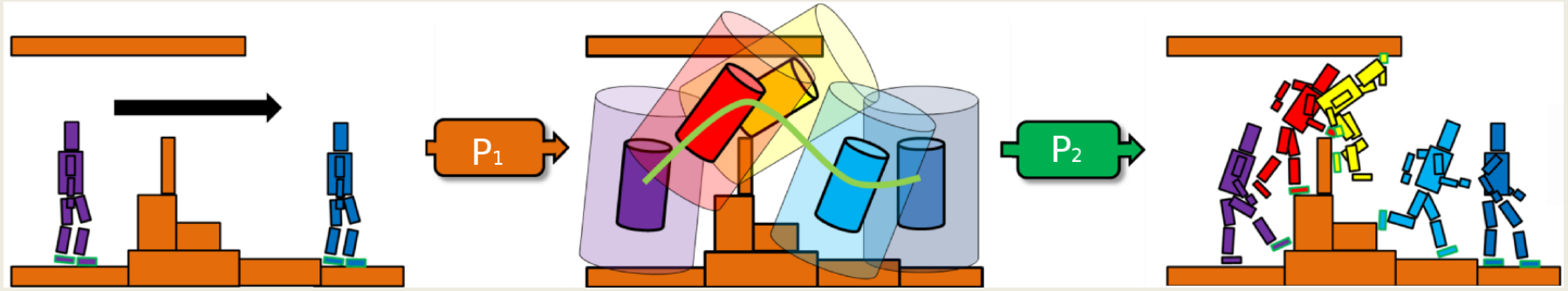


HEURISTIC: ORDERS COMPUTED AS A GUIDE PATH FOR THE ROOT





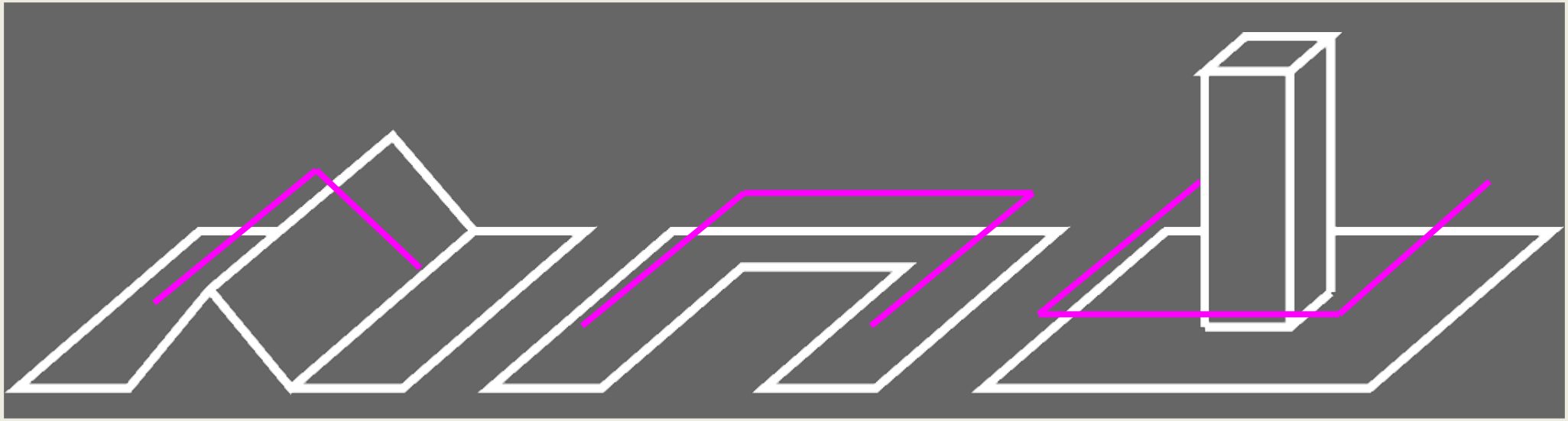
HEURISTIC: ORDERS COMPUTED AS A GUIDE PATH FOR THE ROOT



Guide path planner (orders)

Contact Planner (steps)

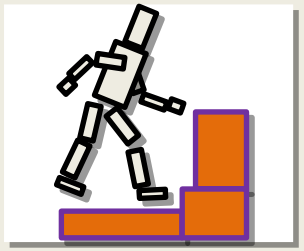
HEURISTIC: ORDERS COMPUTED AS A GUIDE PATH FOR THE ROOT

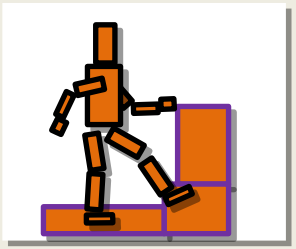


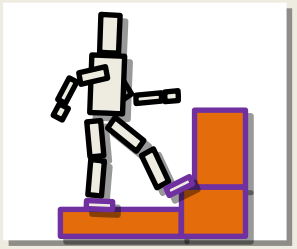
Constraint:

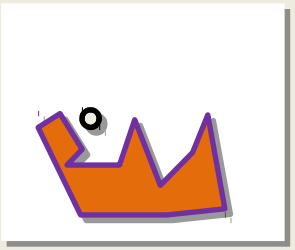
6D root path must allow contact creation

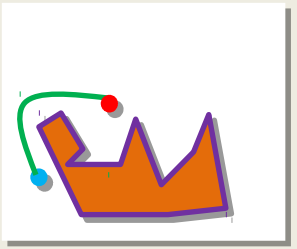
CONTACTS IN CONFIGURATION SPACE

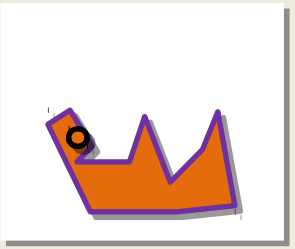


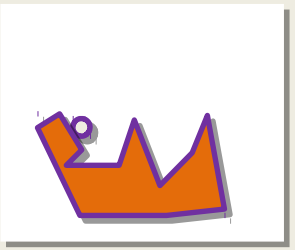


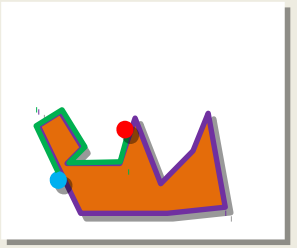




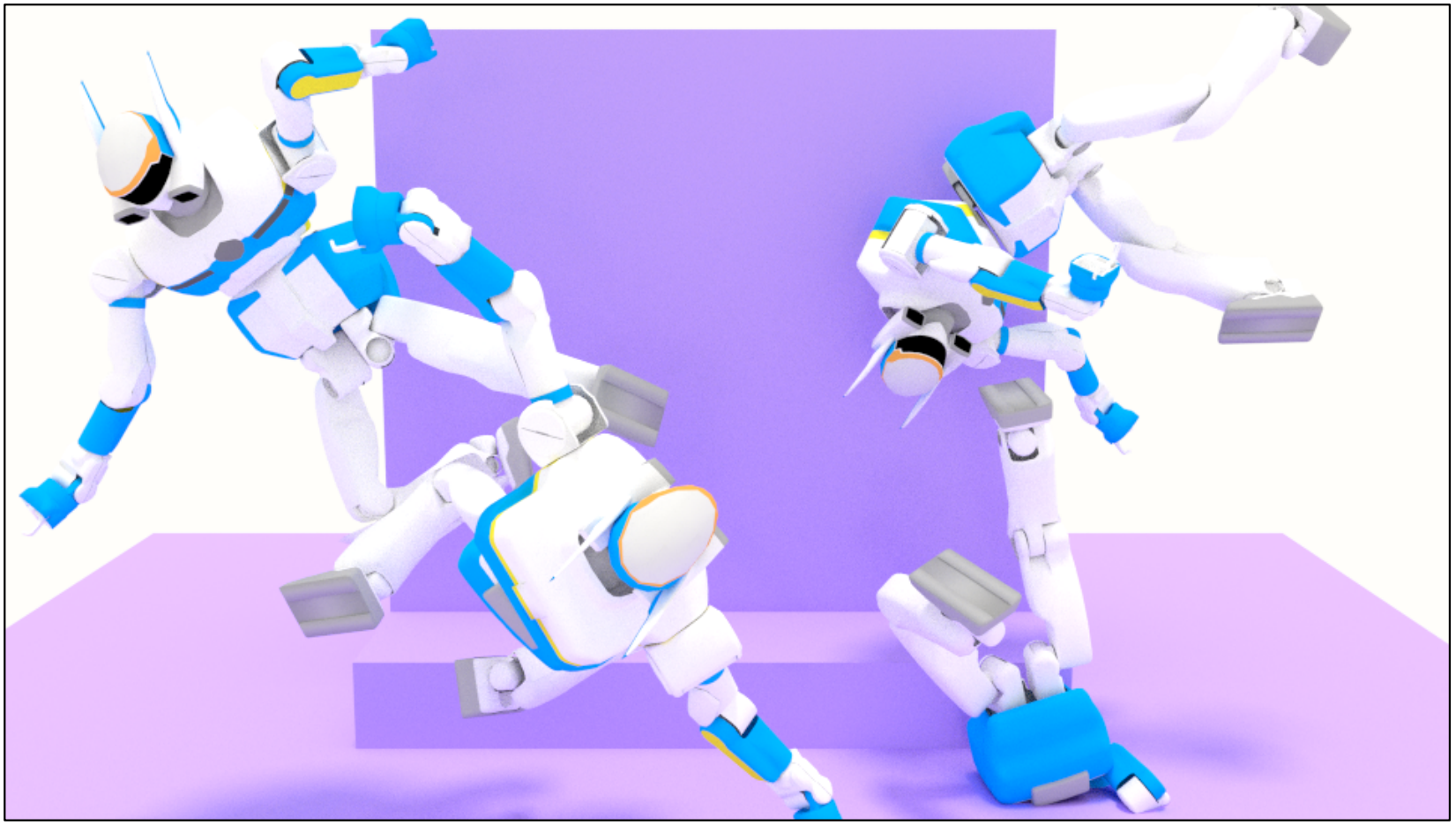


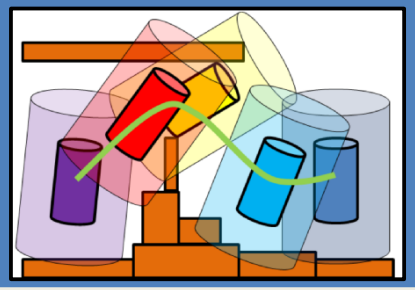






BACK TO THE CURSE OF DIMENSIONALITY

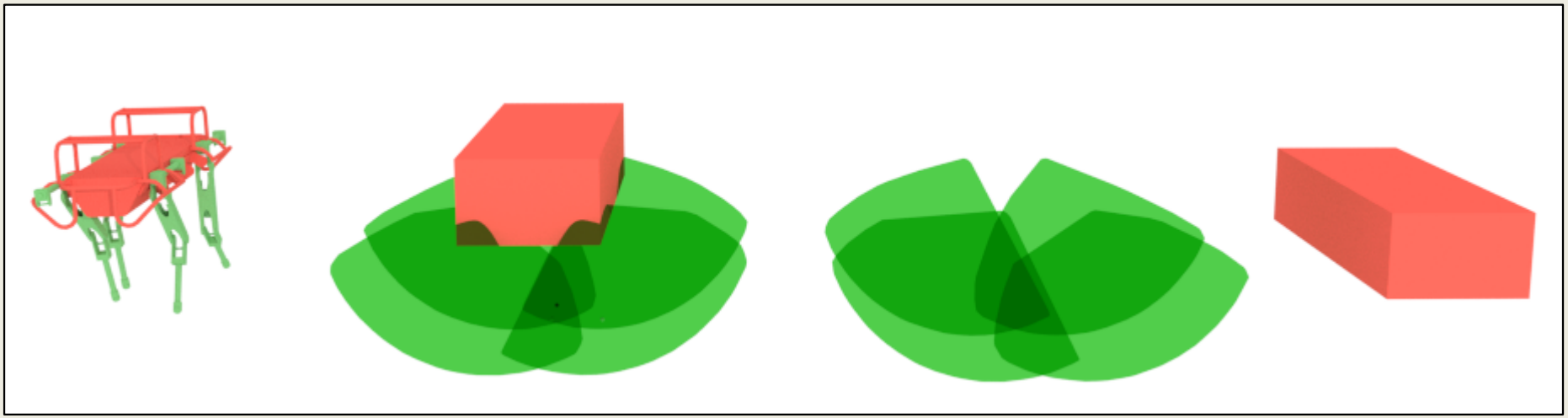




THE REACHABILITY CONDITION

Necessary vs sufficient cond. for *contact feasibility* [1]

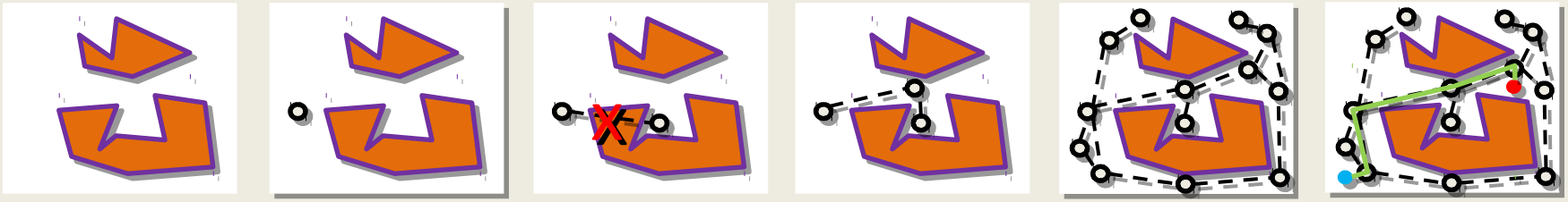
- Obstacles in reachable workspace…
- … while avoiding collisions



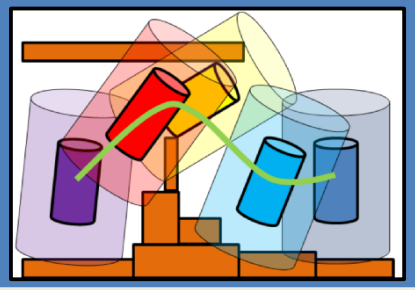
[1] Tonneau et al., an efficient acyclic contact planner for multiped robots, TRO 18

REMINDER ON SAMPLING BASED-PLANNING

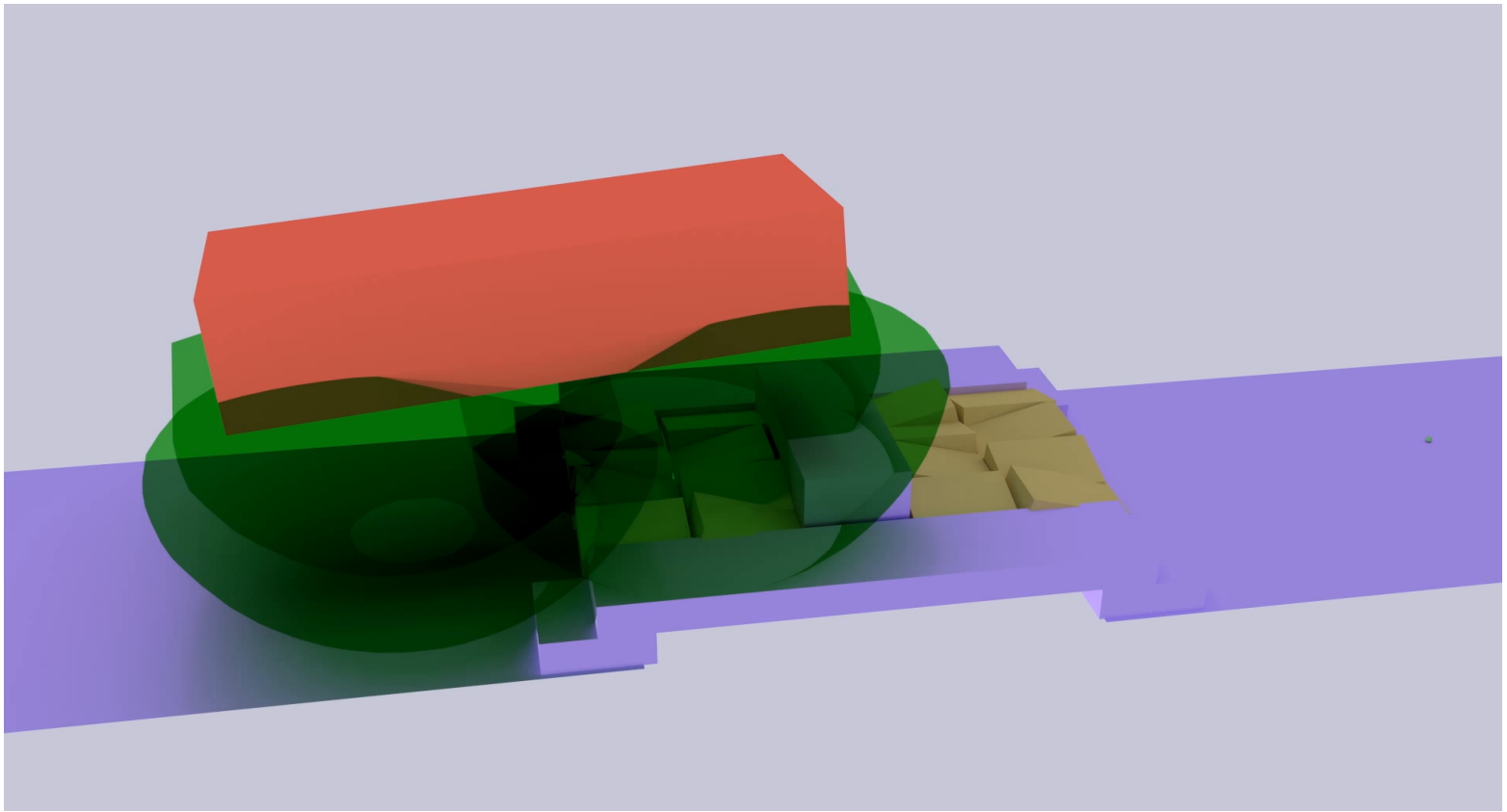
Probabilistic Road Maps (Kavraki et al. 96)
Rapidly exploring Random Trees (Lavalle 98)



DEMONSTRATION

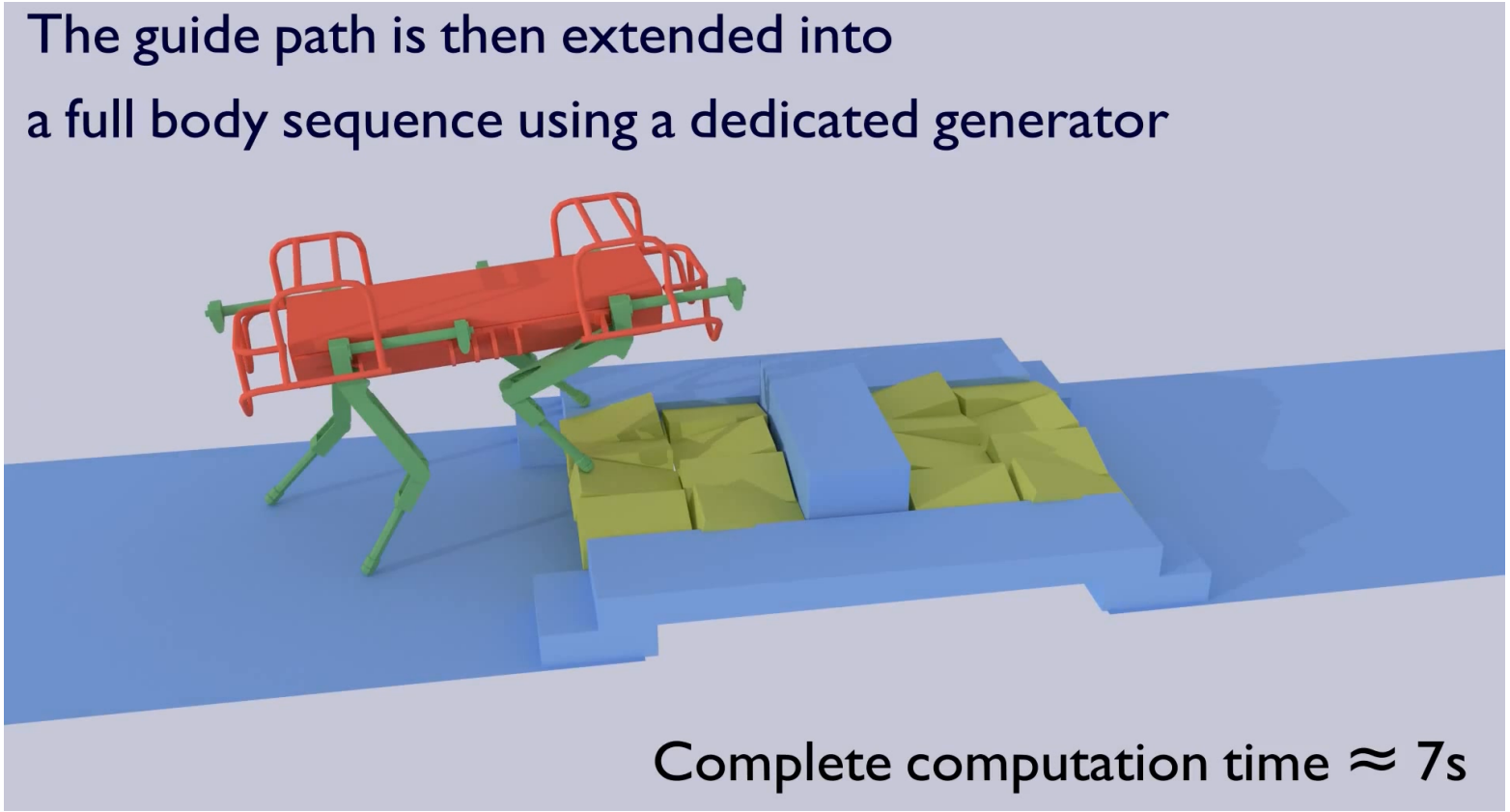


DEMONSTRATION



The guide path is then extended into

a full body sequence using a dedicated generator



Complete computation time ≈ 7s

REACHABILITY BASED CONTACT PLANNING

```
states = []
guide = ComputeGuidePath(s_start, s_goal)
rootPostures = guide.discretize()
for(r in rootPostures):
    while (True):
        State s = ContactSwitchForward(r, states.last()))
        if Viable(s) and TransitionFeasible(s, states.last()) :
            states.insert(s)
            break
```

Limitations?

HOW DOES THAT COMPARE WITH OPTIMIZATION BASED SOLVERS ?

Some insight from a collaboration with ETH Zurich

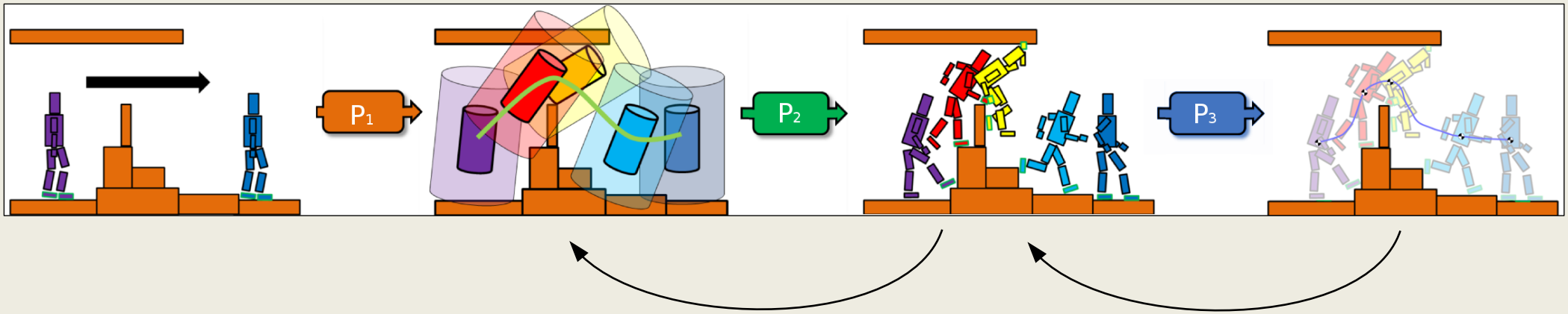
SOME CONCLUSIONS

Heuristics / reduction properties are required to allow for efficient “divide and conquer” approach

Global path planner

Contact Planner

Whole body motion generator



WHAT HAS NOT BEEN DISCUSSED, BUT IS THERE

Heuristics used for contact selection

Collision avoidance (limb rrt)

Reference trajectories

INTERESTING RESEARCH QUESTIONS

Collision avoidance at whole-body level

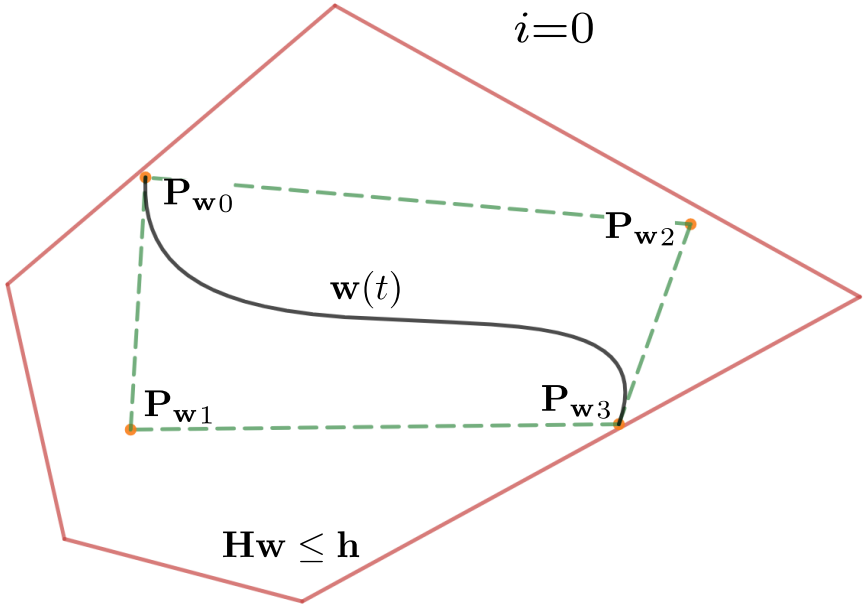
Relaxing guide constraints / improving guide

Bezier curve formulation within NLPs (TOWR)

CROC for Model Predictive Control

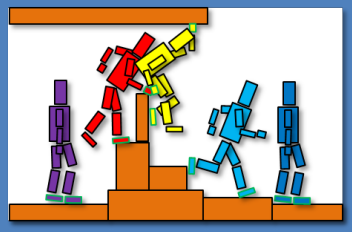
A FIRST TRICK: USING BEZIER CURVES ALLOWS FOR A CONTINUOUS FORMULATION OF THE PROBLEM

Lossless reformulation: $\mathbf{w}(t) = \sum_{i=0}^{n} B_i^n(t/T)\mathbf{P}_{\mathbf{w}i}$



"A Bézier curve is bounded by the convex hull of it's control points."

⇨ Check the constraints at the control points.

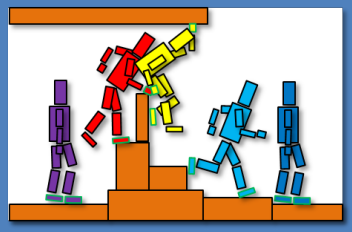


CONSERVATIVE REFORMULATION OF THE TRAJECTORY

$\mathbf{c}(t)$ Bezier curve with only one variable $\mathbf{y}$

$$\mathbf{c}(t, \mathbf{y}) = \underbrace{\sum_{i \in \{0,1,2,4,5,6\}} B_i^6(t/T)\mathbf{P}_i}_{\mathbf{b_c}} + \underbrace{B_3^6(t/T)}_{\mathbf{a_c}}\mathbf{y}$$

$\mathbf{P}_i$ constant terms given by initial / terminal constraints



CONSERVATIVE REFORMULATION OF THE TRAJECTORY

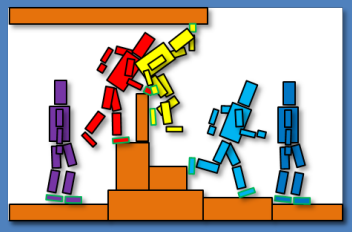
$\mathbf{c}(t)$ Bezier curve with only one variable $\mathbf{y}$

$$\mathbf{c}(t,\mathbf{y}) = \underbrace{\sum_{i\in\{0,1,2,4,5,6\}} B_i^6(t/T)\mathbf{P}_i}_{\mathbf{b_c}} + \underbrace{B_3^6(t/T)}_{\mathbf{a_c}}\mathbf{y}$$

$\mathbf{P}_i$ constant terms given by initial / terminal constraints

$$\mathbf{P}_0 = \mathbf{c}_s$$
$$\mathbf{P}_1 = \frac{\dot{\mathbf{c}}_s/n}{T} + \mathbf{P}_0$$
$$\mathbf{P}_2 = \frac{\ddot{\mathbf{c}}_s/(n(n-1))}{T^2} + 2\mathbf{P}_1 - \mathbf{P}_0$$

$$\mathbf{P}_n = \mathbf{c}_g$$
$$\mathbf{P}_{n-1} = \mathbf{P}_n - \frac{\dot{\mathbf{c}}_g/n}{T}$$
$$\mathbf{P}_{n-2} = \frac{\ddot{\mathbf{c}}_g/(n(n-1))}{T^2} + 2\mathbf{P}_{n-1} - \mathbf{P}_n$$



A SECOND TRICK: CONSERVATIVE REFORMULATION OF THE TRAJECTORY

$$\mathbf{c}(t,\mathbf{y}) = \mathbf{a_c}(t)\mathbf{y} + \mathbf{b_c}(t), \mathbf{y} \in \mathbb{R}^3$$

$$\Rightarrow \quad \ddot{\mathbf{c}}(t,\mathbf{y}) = \mathbf{a_{\ddot{c}}}\mathbf{y} + \mathbf{b_{\ddot{c}}}$$

$$\Rightarrow \quad \mathbf{c}(t,\mathbf{y}) \times \ddot{\mathbf{c}}(t,\mathbf{y}) = \mathbf{a_{c\times\ddot{c}}}\mathbf{y} + \mathbf{b_{c\times\ddot{c}}}$$

$\mathbf{w}(t,\mathbf{y})$ linearly dependent of $\mathbf{y}$ $\qquad \mathbf{My} \preceq \mathbf{m}$