CS686 MOTION PLANNING & APPLICATION

PLANNER FOR MULTI-CONTACT LOCOMOTION

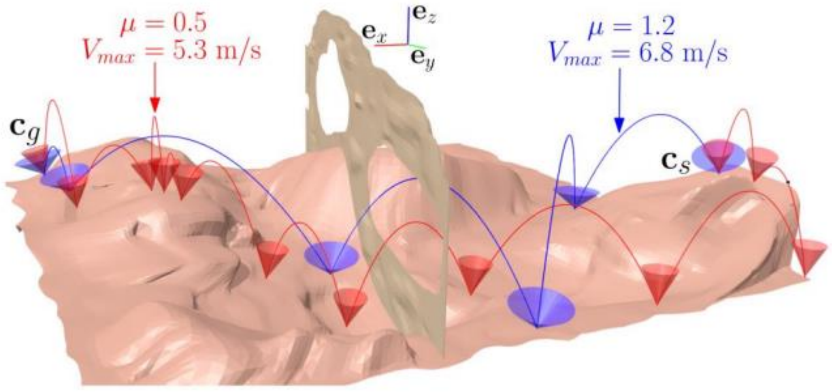
Zeehyo Kim
(김지효)

2019.12.03

KAIST

Recap

- 1. Ballistic Motion Planning
  - Simulation for ballistic motion of point robot in 3D environment.
  - Constraints from the slipping and velocity.



- 2. Single Leg Dynamic Motion Planning with Mixed-Integer Convex Optimization
  - Implemented ballistic motion planning for real robot.
  - Simplify constraints through Mixed-Integer convex optimization

An Efficient Acyclic Contact Planner for Multiped Robots

S.Tonneau, A. D. Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, *IEEE Transactions on Robotics*, 2018

Objective

- Make a contact planner for complex legged locomotion tasks.
  - Contact planner will give contact sequences and planned motion sequences.
    - Why is contact important?
  - It plans an acyclic contact sequence between a start and goal configuration in cluttered environments considering static equilibrium.

Previous works on contact planning

- Key issue: Avoiding combinatorial explosion while considering the possible contacts and potential paths.
- [T. Bretl. 2006], [K. Hauser, et al. 2006], [H. Dai, et al. 2014]
  - Papers proposed effective algorithms on simple situations, but not applicable to arbitrary environments.
- [R. Deits and R. Tedrake, 2014]
  - Solved contact planning globally as mixed-integer problem, but only cycle bipedal locomotion is considered, and equilibrium is not considered.

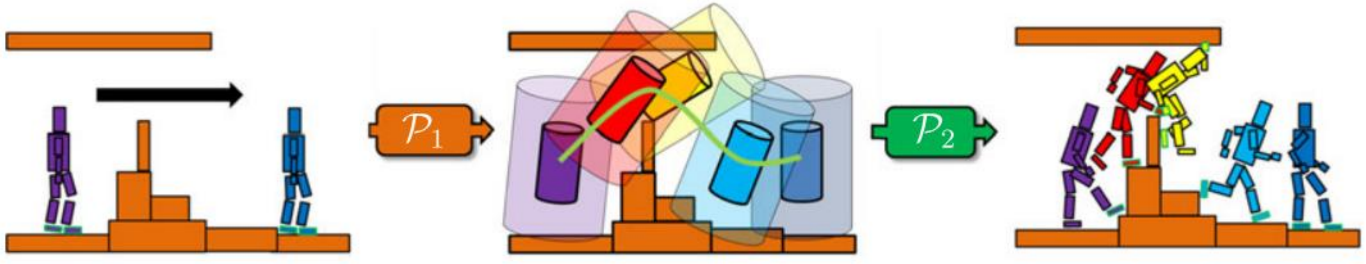
Problems for acyclic contact planning

- $\wp_{1}$: Computation of a root guide path
- $\wp_{2}$: Generating a discrete sequence of contact configurations.
- $\wp_{3}$: Interpolating a complete motion btwn two postures of the contact sequence.

- → In this work, problems are going to be decoupled into subproblems to reduce the complexity.
- In the paper, <u>solving the $\wp_{1}$, $\wp_{2}$ is introduced</u>, and planner uses existing method for $\wp_{3}$.
  - For $\wp_{3}$, they used [J. Carpentier, et al. 2016]

Overview of two-stage framework

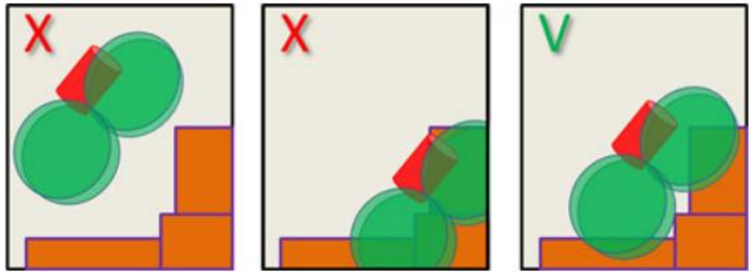
- 1. Plan feasible root guide path ($\wp_1$)
  - It will be achieved by defining geometric condition, the *reachability condition*.
  - Plan guide path in a low dimensional space by RRT.
- 2. Generate a discrete sequence of contact configurations($\wp_2$)
  - Extending the path to a discrete sequence of whole-body configuration in static equilibrium using an iterative algorithm.



(a) (b) (c)

Root Path Planning($\wp_1$)

- Conditions for *contact reachable workspace*
  - Compromise between the following conditions.
  - Necessary conditions $\mathcal{C}^0_{\text{Nec}} = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \text{ and } W^0(\mathbf{q}^0) \cap O = \emptyset\}.$
    - Torso of robot: Collision free
    - Obstacle need to intersects reachable workspace of a robot.
  - Sufficient conditions $\mathcal{C}^0_{\text{Suf}} = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \text{ and } B^{\text{Suf}}(\mathbf{q}^0) \cap O = \emptyset\}.$
    - Bounding volume of whole robot except effector surfaces: not intersect with object.
  - ➢ Reachability condition:
    - $\boldsymbol{W_s^0}$: scaling transformation of vol $C_s^0 = \{\mathbf{q}^0 : W(\mathbf{q}^0) \cap O \neq \emptyset \textbf{ and } W_s^0(\mathbf{q}^0) \cap O = \emptyset\}.$



Root Path Planning($\wp_1$)

- Computing the Guide Path in contact reachable workspace.
  - Motion planning with any standard motion planner is possible. Here, authors used Bi-RRT Planner.
  - Only difference from classic implementation is that instead of collision detection, authors validate it with *reachability condition*.

Generate a discrete sequence of contact configurations($\wp_2$)

- Root path $\boldsymbol{q^0(t)}$ discretized into a sequence of $\boldsymbol{j}$ key configurations: $\boldsymbol{Q^0} = [\boldsymbol{q_0^0};\ \boldsymbol{q_1^0}; \dots ; \boldsymbol{q_{j-1}^0}]$
  - $\boldsymbol{j}$: Discretization step.
- Contact sequence follows below.
  - At most one contact is broken between two consecutive configurations.
  - At most one contact is added between two consecutive configurations.
  - Each configuration is in static equilibrium.
  - Each configuration is collision-free.
- They create contacts using first in, first out approach
  - First try to create contact with the limb that has been contact free longest.

Generate a discrete sequence of contact configurations($\wp_2$)

Algorithm 1: Discretization of a Path.

```
function INTERPOLATE(s0, Q^0, MAX_TRIES)
    states ← (s0)  ▷ List of states initialized with s0
    nb_fail ← 0
    i ← 1;                        ▷ Current index in the list
    while i < Length(Q^0) do
        pState ← LastElement(states)
        s ← GENFULLBODY(pState, Element(Q^0, i))
        if s ≠ 0 then
            nb_fail ← 0
            i ← i + 1
        else
            nb_fail ← nb_fail + 1
            s ← INTERMEDIATECONTACTSTATE(pState)
            if s == 0 ∨ nb_fail == MAX_TRIES then
                return FAILURE
        PushBack(states, s)
    return states
```

Algorithm 3: Adds or Repositions a Contact for One Limb.

```
function INTERMEDIATECONTACTSTATE(state)
    newState ← state
    for each k in FreeLimbs(newState) do
        if GenerateContact(newState, k) then
            MarkContact(newState, k)
            return newState
    for each k in ContactLimbs(newState) do
        if GenerateContact(newState, k) then
            /*Account for repositioning in FIFO queue*/
            MarkContact(newState, k)
            return newState
    /*Fails if impossible to relocate any effector*/
    return 0
```

Algorithm 2: Full Body Contact Generation Method.

```
function GENFULLBODY(pState, q^0)
newState ← CreateState(q^0,
                       ContactLimbs(pState))
    nConBroken ← 0
    for each k in ContactLimbs(pState) do
        if ¬MaintainContact(pState, q^0, k) then
            nConBroken ← nConBroken + 1
            if nConBroken > 1 then
                return 0
            MarkFree(newState, k)
        else
            MarkContact(newState, k)
    for each k in FreeLimbs(newState) do
        if GenerateContact(newState, k) then
            MarkContact(newState, k)
            return newState
    if IsInStaticEquilibrium(newState) then
        return newState
    else
        return 0
```

Generate a discrete sequence of contact configurations($\wp_2$)

- Contact Generator
  - Input: a configuration of the root and the list of effectors that should be in contact
  - Output: configuration of the limbs

Generate a discrete sequence of contact configurations($\wp_2$)

- Contact Generator
  - $C^{\epsilon}_{Contact}$ is the set of configuration that the minimum distance between an effector and an obstacle is less then ϵ
  - 1) Generate offline N valid sample limb configurations
  - 2) When contact creation is required, retrieve the list of samples $S \subset C^{\epsilon}_{Contact}$ close to contact.
  - 3) Use a user-defined heuristic to sort S.
  - 4) S is empty, stop. Else select the first configuration of S, and project it onto contact using inverse kinematics.
  - Until a configuration is in static equilibrium!

Conditions for Static Equilibrium

- They solved LP on Newton-Euler equations to judge static equilibrium.

1) $\mathbf{c} \in \mathbb{R}^3$ is the robot center of mass (COM);
2) $m \in \mathbb{R}$ is the robot mass;
3) $\mathbf{g} = [0, 0, -9.81]^T$ is the gravity acceleration;
4) μ is the friction coefficient;
5) for the ith contact point $1 \leq i \leq e$:
   a) $\mathbf{p_i}$ is the contact position;
   b) $\mathbf{f_i}$ is the force applied at $\mathbf{p_i}$;
   c) $\mathbf{n}_i, \gamma_{i1}, \gamma_{i2}$ form a local Cartesian coordinate system centered at $\mathbf{p_i}$. $\mathbf{n}_i$ is aligned with the contact surface normal, and the γ_i s are tangent vectors.

$$\mathbf{V}_i = \left[\mathbf{n}_i + \mu\gamma_{i1} \ \ \mathbf{n}_i - \mu\gamma_{i1} \ \ \mathbf{n}_i + \mu\gamma_{i2} \ \ \mathbf{n}_i - \mu\gamma_{i2} \right]^T$$

Stacked non-slipping constraints: $\exists \boldsymbol{\beta} \in \mathbb{R}^{4e}, \boldsymbol{\beta} \geq 0$ and $\mathbf{f} = \mathbf{V}\boldsymbol{\beta}$

Newton-Euler equations

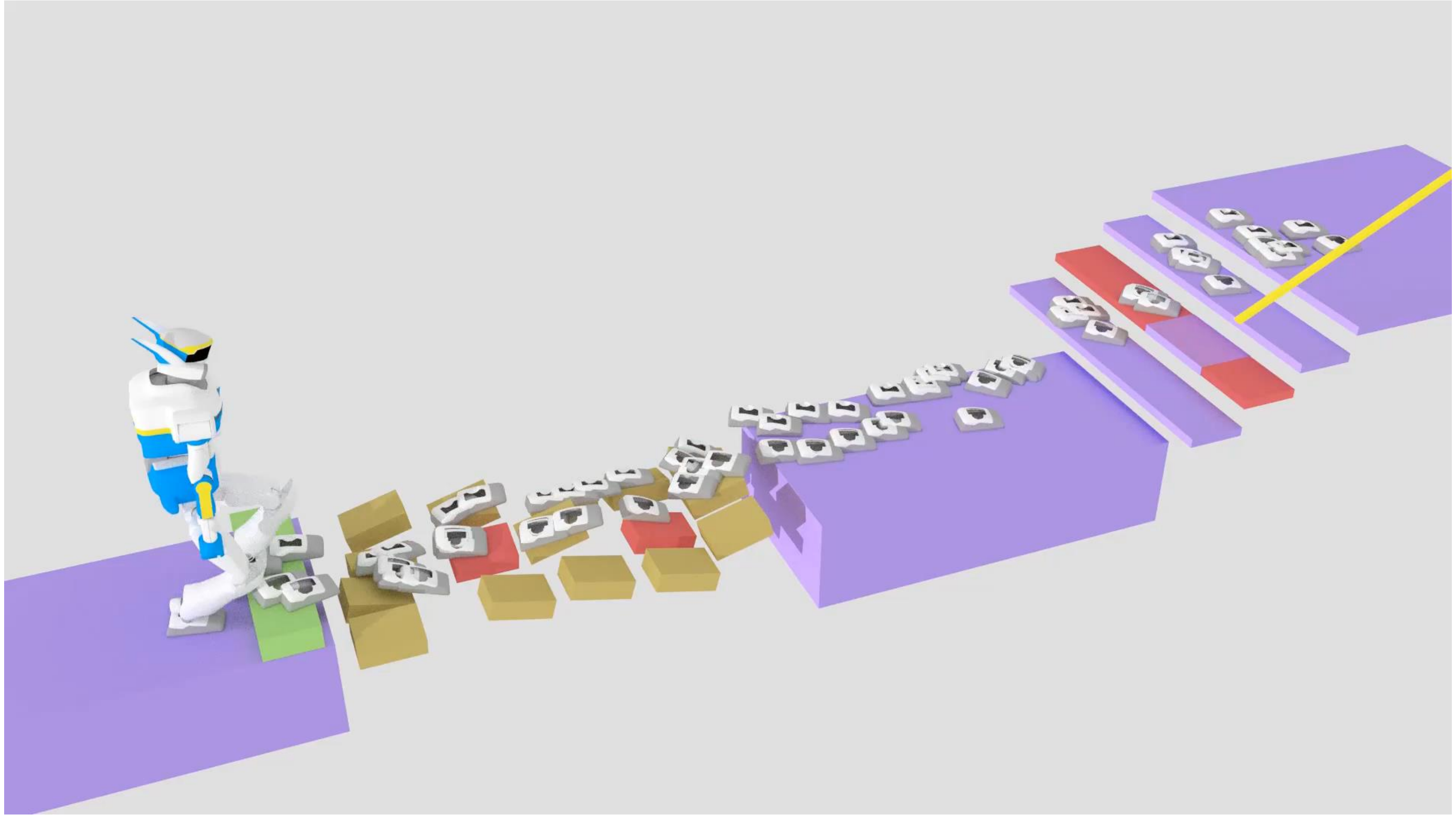
$$\underbrace{\begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_e \end{bmatrix}}_{\mathbf{G}} \mathbf{V}\,\boldsymbol{\beta}, = \underbrace{\begin{bmatrix} \mathbf{0}_{3\times 3} \\ m\hat{\mathbf{g}} \end{bmatrix}}_{\mathbf{D}} \mathbf{c} + \underbrace{\begin{bmatrix} -m\mathbf{g} \\ \mathbf{0} \end{bmatrix}}_{\mathbf{d}}$$

Robust LP formulation

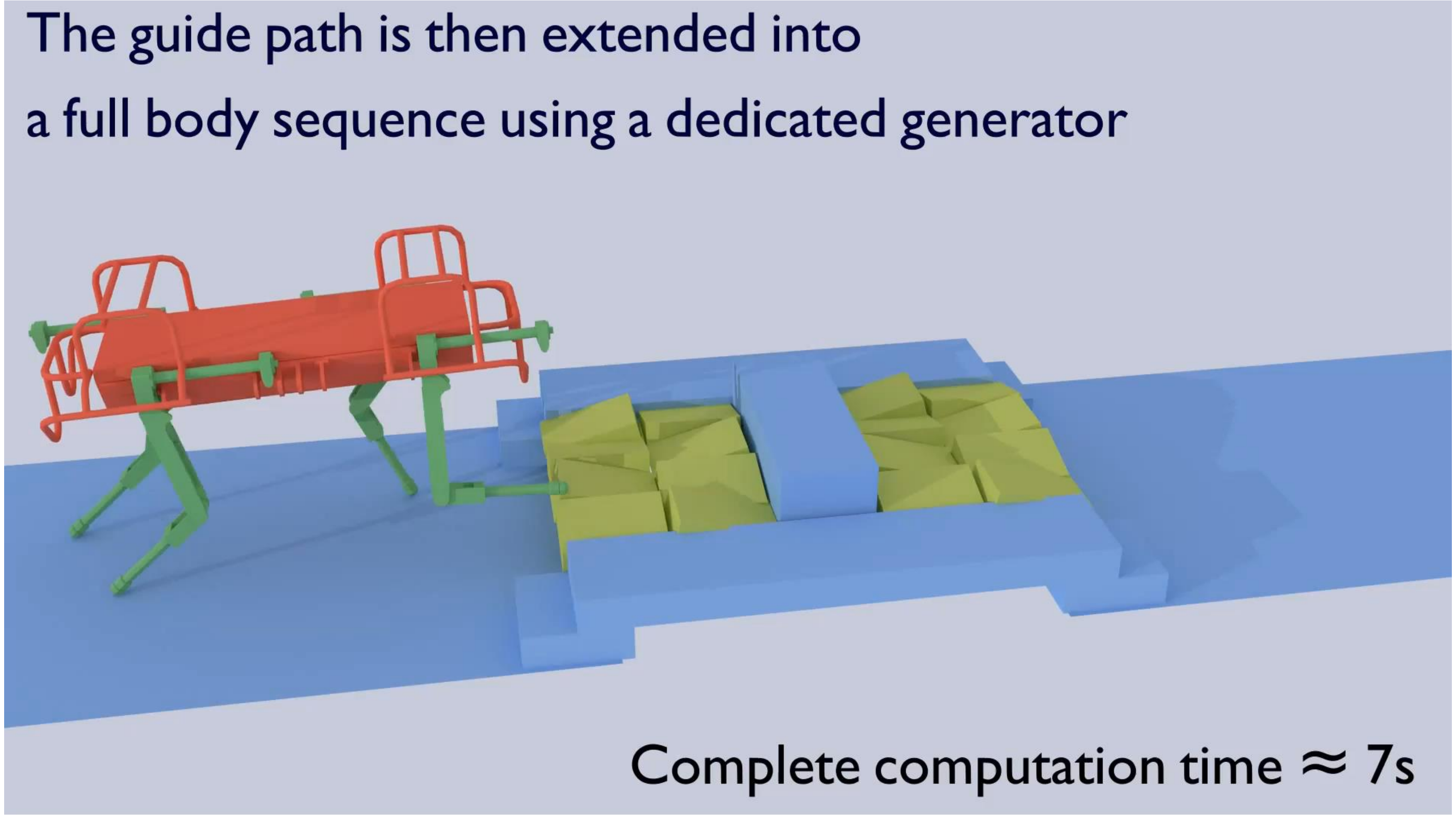
$$\begin{aligned} \text{find} \quad & \boldsymbol{\beta} \in \mathbb{R}^{4e}, b_0 \in \mathbb{R} \\ \text{minimize} \quad & -b_0 \\ \text{subject to} \quad & \mathbf{G}\boldsymbol{\beta} = \mathbf{D}\mathbf{c} + \mathbf{d} \\ & \boldsymbol{\beta} \geq b_0 \mathbf{1}. \end{aligned}$$

- If b_0 is positive, configuration is in static equilibrium, and otherwise, no.

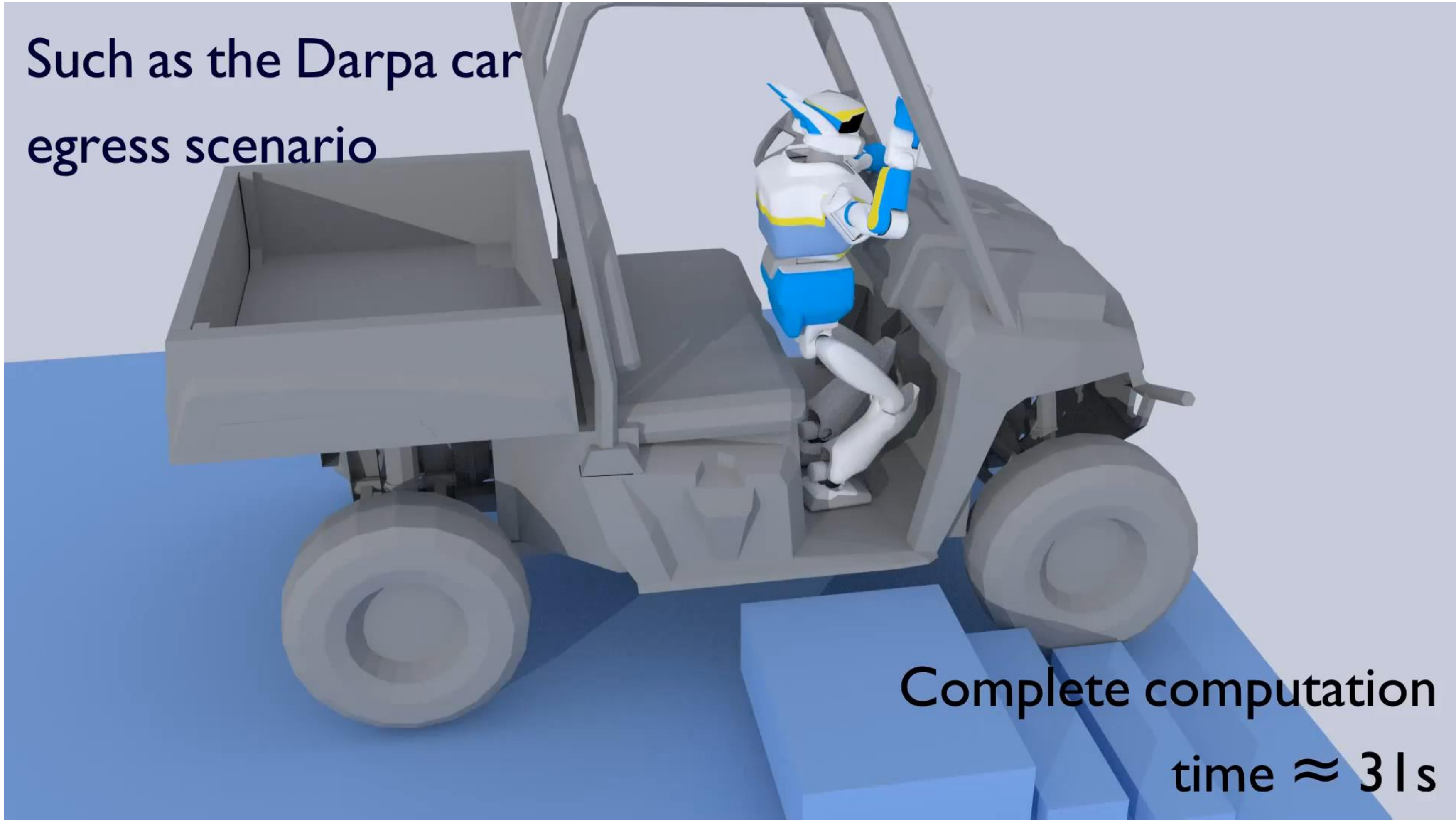
Result videos



Result videos



Result videos



Conclusion

- They consider the multi-contact planning problem formulated as three subproblem $\wp_1$, $\wp_2$, $\wp_3$.
- Their contributions
  - $\wp_1$: Simply computing root path by introduction of a low-dimensional space.
  - $\wp_2$: fast contact generation scheme.
- Their approach was successful compare to previous approach.

Scenario	Method	Computation time
Stair 20 cm	Hauser [9]	5.42 min
	Mordatch et al.[12]	2 to 10 min
	Ours + [1]	$<$ **2s**
Stair 30 cm	Hauser [9]	4.08 min
	Mordatch et al.[12]	2 to 10 min
	Ours	$<$ **2s**
Stair 40 cm	Hauser [9]	10.08 min
	Mordatch et al.[12]	2 to 10 min
	Ours	$<$ **5s**
Table (car) egress	Bouyarmane et al. [19], [18]	3.5 hours
	Ours	$<$ **60 s**

A kinodynamic steering-method for legged multi-contact locomotion

P. Fernbach, S. Tonneau, A. D. Prete, and M. Ta¨ıx, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017

Objective

- Make a kinodynamic planner for multi-contact motions that could produce highly-dynamic motion in complex environments.

- The kinodynamic planning requires:
  - A steering method that heuristically generates a trajectory between two states of the robot.
  - A trajectory validation method that verifies that the trajectory is collision-free, dynamically feasible.

Why is it difficult?

- Dynamic constraints of the robot are state-dependent.
- The contact points are not coplanar: cannot apply simplified dynamic models.
- Local optimization methods could get stuck in local minima.

Previous works for multi-contact planning

- Regard the problem as a optimization problem
  - I. Mordatch, E. Todorov, and Z. Popovic (2012)
  - Problem: Be able to result in local minima.
- Plan the root path heuristically, then generate a contact sequence along this path.
  - Paper 1
  - Problem: Each contact phase to be in static equilibrium.

Definitions

- State $\mathbf{S} =< \mathbf{X}, \mathbf{P}, \mathbf{N}, \mathbf{q} >$
    - $\mathbf{X} =< \mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}} >$: position, velocity, acceleration of COM
    - $\mathbf{P} =< \mathbf{p_1}, \dots, \mathbf{p_k} >$: $\mathbf{p_i} \in \mathbb{R}^3$, position of the i-th contact point
    - $\mathbf{N} =< \mathbf{n}_1, \dots, \mathbf{n}_k >$: $\mathbf{n_i} \in \mathbb{R}^3$, surface normal of the i-th contact point
    - $\mathbf{q} \in \boldsymbol{SE}(\mathbf{3}) \times \mathbb{R}^{\boldsymbol{n}}$: configuration of the robot.
- All positions are expressed in the world frame.

Steering method & Trajectory validation

- Proposed two-step method
  - 1. Use DIMT method with constraints computed for the initial state $\boldsymbol{S_0}$.
    - DIMT (Double Integrator Minimum time) method
      - Input: User-defined constant & symmetric bounds on the COM dynamics along orthogonal axis, initial state, target state
      - Output: minimum time trajectory $\mathbf{X}(\mathbf{t})$ that connects exactly $\mathbf{X_0}$ and $\mathbf{X_1}$. (w/o considering collision avoidance)
    - Increase probability that the trajectory $\boldsymbol{X}(\boldsymbol{t})$ be dynamically feasible in the neighborhood of $\boldsymbol{S_0}$.

Steering method & Trajectory validation

- Proposed two-step method
  - 2. In trajectory validation phase, verify the dynamic equilibrium of the root, collision avoidance along the trajectory.
    - Inputs: Two states $\mathbf{S_0}$, $\boldsymbol{S_1}$ and a trajectory $\mathbf{X(t)}$ connecting them.
    - Output: dynamically feasible, collision free sub-trajectory of $\mathbf{X}$, $\mathbf{X'(t)}$

KAIST

Trajectory validation

- They checked dynamic equilibrium by solving the LP.
  - Detail → Refer to paper

2) *Dynamic equilibrium:* We formulate a test for dynamic equilibrium as an LP, extending the static equilibrium test proposed in [19]. The Newton-Euler equations verify:

$$\begin{bmatrix} m(\ddot{\mathbf{c}} - \mathbf{g}) \\ m\mathbf{c} \times (\ddot{\mathbf{c}} - \mathbf{g}) + \dot{\mathbf{L}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_k \end{bmatrix} \mathbf{f} \qquad (4)$$

Where :

- m is the total mass of the robot;
- $\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_k]^T \in \mathbb{R}^{3k}$ is the stacked vector of contact forces $\mathbf{f}_i$, applied at contact position $\mathbf{p}_i$;
- $\mathbf{g} = [0 \quad 0 \quad -9.81]^T$ is the gravity vector;
- $\mathbf{L} \in \mathbb{R}^3$ is the angular momentum (expressed at $\mathbf{c}$).
- $\hat{\mathbf{p}}_i$ denotes the skew-symmetric matrix associated to $\mathbf{p}_i$.

To respect the non-slipping condition, we constrain the contact forces $\mathbf{f}_i$ to lie inside a linearized friction cone of generators $\mathbf{V}_i$, such that [4]:

$$\mathbf{f}_i = \mathbf{V}_i \boldsymbol{\beta}_i \text{ with } \boldsymbol{\beta}_i \in \mathbb{R}^4 \text{ and } \boldsymbol{\beta}_i \geq \mathbf{0} \qquad (5)$$

which leads to

$$\mathbf{f} = \mathbf{V}\boldsymbol{\beta} \text{ with } \boldsymbol{\beta} \in \mathbb{R}^{4k} \text{ and } \boldsymbol{\beta} \geq \mathbf{0} \qquad (6)$$

where $\mathbf{V} = diag([\mathbf{V}_1 \quad \dots \quad \mathbf{V}_k]) \in \mathbb{R}^{3k \times 4k}$.
We set $\dot{\mathbf{L}} = 0$ as classically done [25] and rewrite (4):

$$\underbrace{m\begin{bmatrix} \mathbf{I}_3 \\ \hat{\mathbf{c}} \end{bmatrix}}_{\mathbf{H}} \ddot{\mathbf{c}} + \underbrace{m\begin{bmatrix} -\mathbf{g} \\ \mathbf{c} \times -\mathbf{g} \end{bmatrix}}_{\mathbf{h}} = \underbrace{\begin{bmatrix} \mathbf{I}_3 & \dots & \mathbf{I}_3 \\ \hat{\mathbf{p}}_1 & \dots & \hat{\mathbf{p}}_k \end{bmatrix} \mathbf{V}}_{\mathbf{G}} \boldsymbol{\beta} \qquad (7)$$

Thus, if there exists a $\boldsymbol{\beta}^*$ such that $\boldsymbol{\beta}^* \geq \mathbf{0}$ and (7) is satisfied, it means the robot is in dynamic equilibrium. Our test then boils down to solving the following LP:

$$\begin{aligned} &\text{find } \boldsymbol{\beta} \\ &\text{s.t. } \mathbf{G}\boldsymbol{\beta} = \mathbf{H}\ddot{\mathbf{c}} + \mathbf{h} \qquad (8) \\ &\quad\ \boldsymbol{\beta} \geq 0 \end{aligned}$$

KAIST

Computing acceleration bounds for the steering method

- 1. Call DIMT with arbitrary large bounds, then obtain a trajectory. $\mathbf{a}$ is the direction of the first phase.
- 2. Computes maximum acceleration $\ddot{\mathbf{c}}^{max} = \alpha^* \mathbf{a}$, and $\alpha^* \in \mathbb{R}^+$ satisfying non-slipping condition at $\mathbf{S_0}$.
  - α^* can be achieved by LP extending previous slide, also refer to paper.
- 3. Compute $\mathbf{X}(\boldsymbol{t})$ by calling again the DIMT, using bound $-(\alpha^* \mathbf{a})_{\{x,y,z\}} \leq \ddot{\mathbf{c}}_{\{x,y,z\}} \leq (\alpha^* \mathbf{a})_{\{x,y,z\}}$

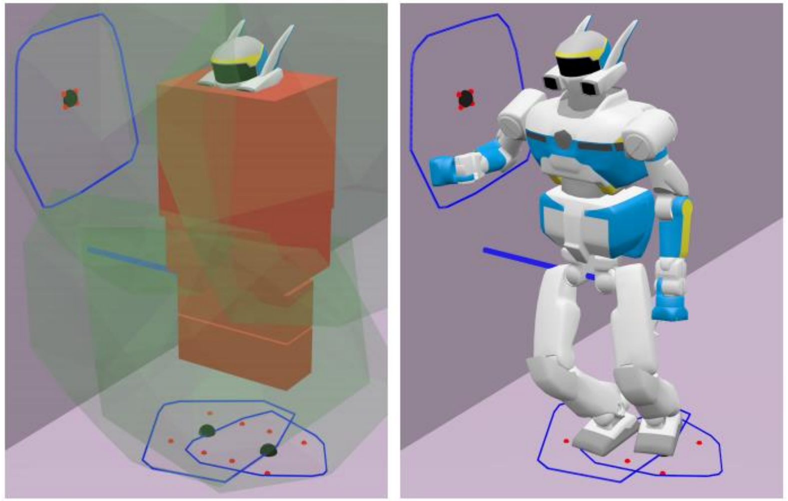
KAIST

Application to multi-contact planning

- They integrated their methods within the [Paper 1] to generate multi-contact motions.
  - They are going to modify planner from PAPER 1 that decouples motion planning problem to three phases.
    - $\wp_1$: Planning a root trajectory by RRT
    - $\wp_2$: Generation of a discrete contact sequence along this root path
    - $\wp_3$: Interpolating contact sequence into continuous motion for the robot.

Application to multi-contact planning

- Planning a root trajectory($\wp_1$)
  - Integration of the steering method.
    - For given configuration, compute the intersection between the reachable workspace of each effector and the environments. Then assume that a contact exists at the center of this intersection.
    - Approximate COM as position of the root.



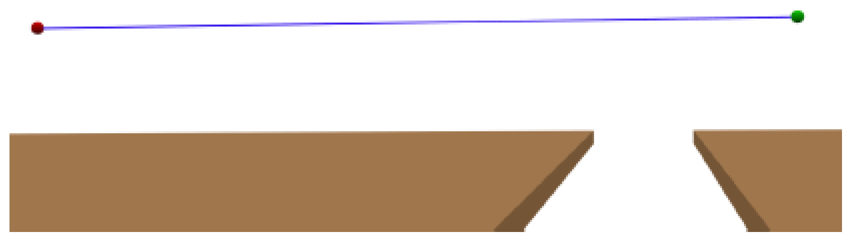
◀ Approximation of contact locations in $\wp_1$

Application to multi-contact planning

- Planning a root trajectory($\wp_{\mathbf{1}}$)
  - Trajectory validation in $\wp_{\mathbf{1}}$
    - Approximate contact phases $\mathbf{P}(\boldsymbol{t})$, $\mathbf{N}(\boldsymbol{t})$.
    - Assume that contacts sliding along with COM trajectory. - If it could not be continued, estimates new contacts and continue.

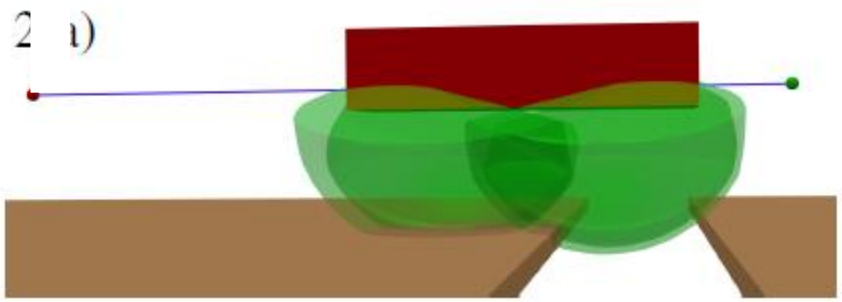
Application to multi-contact planning

- Planning a root trajectory($\wp_1$)
  - Generating jumps
    - If the produced trajectory between to states results in phases where the number of active contact falls under a user-defined threshold, the planner could try to generate a jump

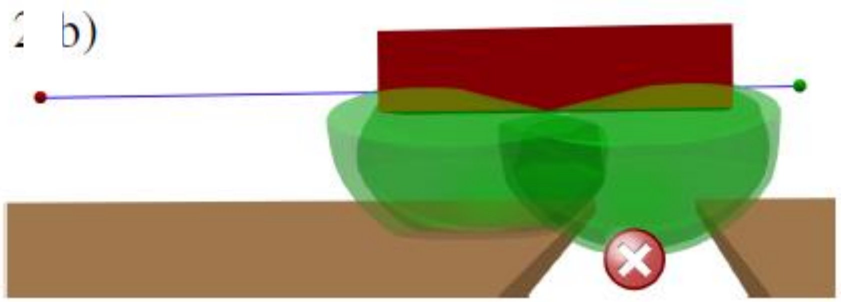


Application to multi-contact planning

- Planning a root trajectory($\wp_1$)
  - Generating jumps
    - 1) Identify the last state $X(t_{to})$ where the desired effectors were still in contact.



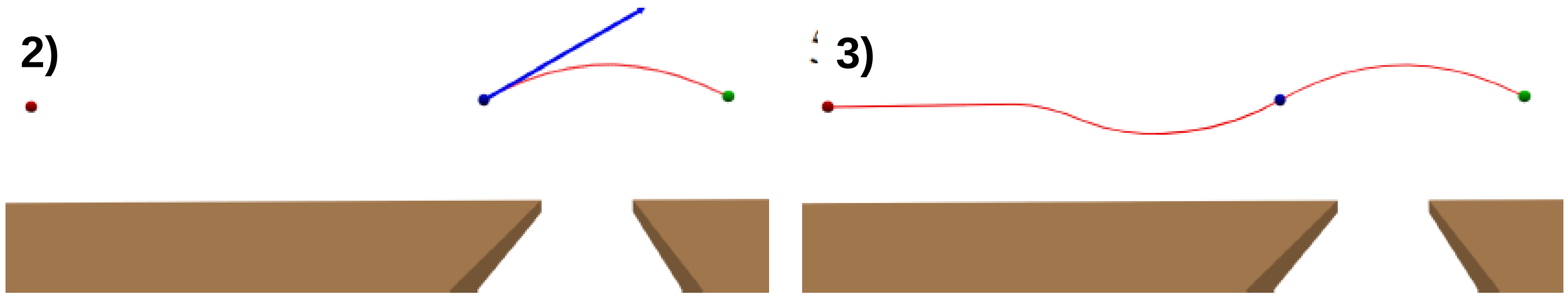
▲ Last state that desired effectors were still in contact



▲ First invalid state

Application to multi-contact planning

- Planning a root trajectory($\wp_1$)
  - Generating jumps
    - 2) Try to compute a feasible take-off velocity $\dot{c}_{to}$ such that a collision free ballistic motion between $\mathbf{S}_{\mathbf{to}}$ and $\mathbf{S}_1$ is feasible.
      - If contact force at jumping is in the centroidal convex cone, slip would not occur.
    - 3) Compute a trajectory from $\mathbf{S}_0$ to $\mathbf{S}_{\mathbf{to}}$, and connect with 2).

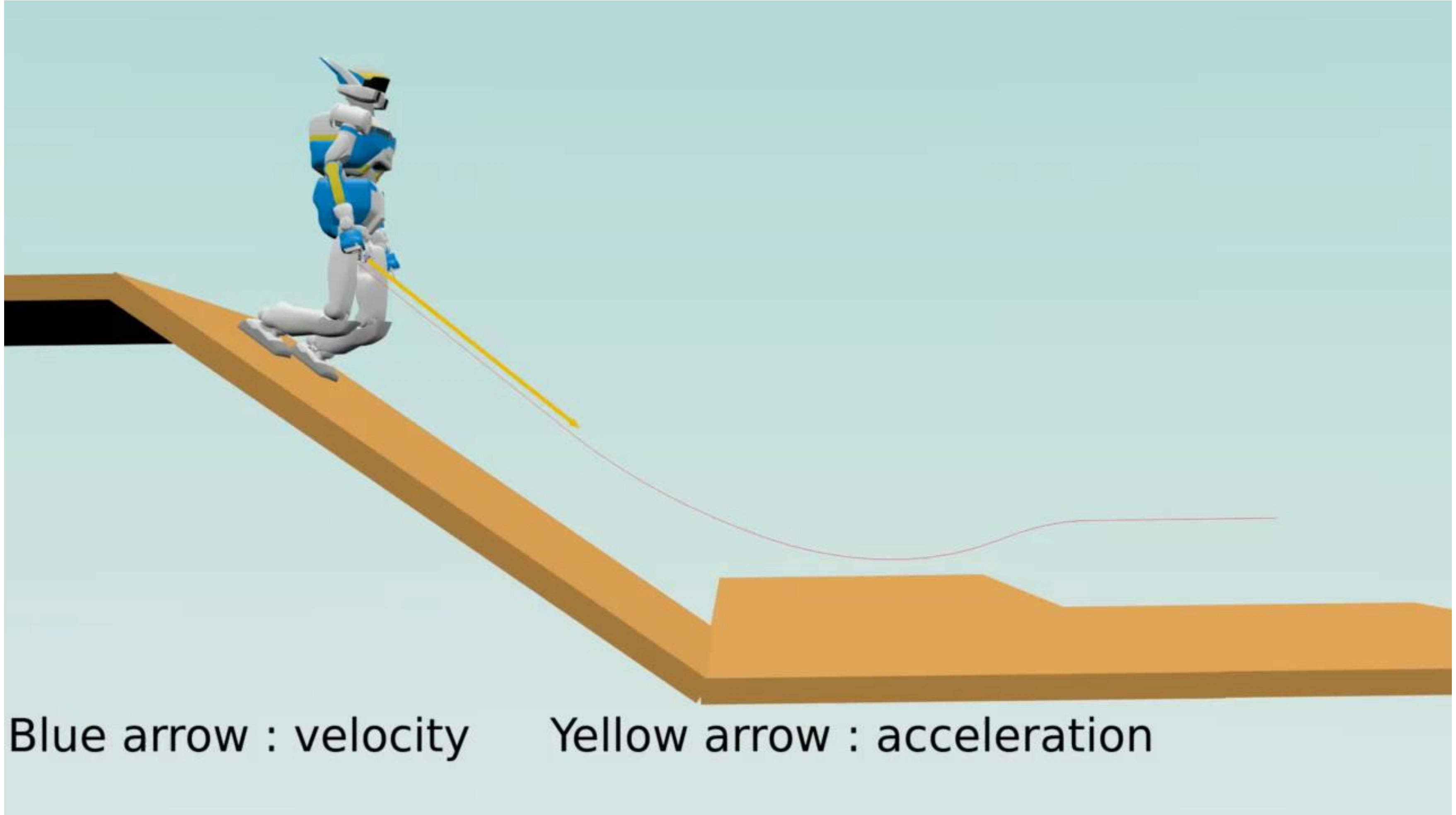


Application to multi-contact planning

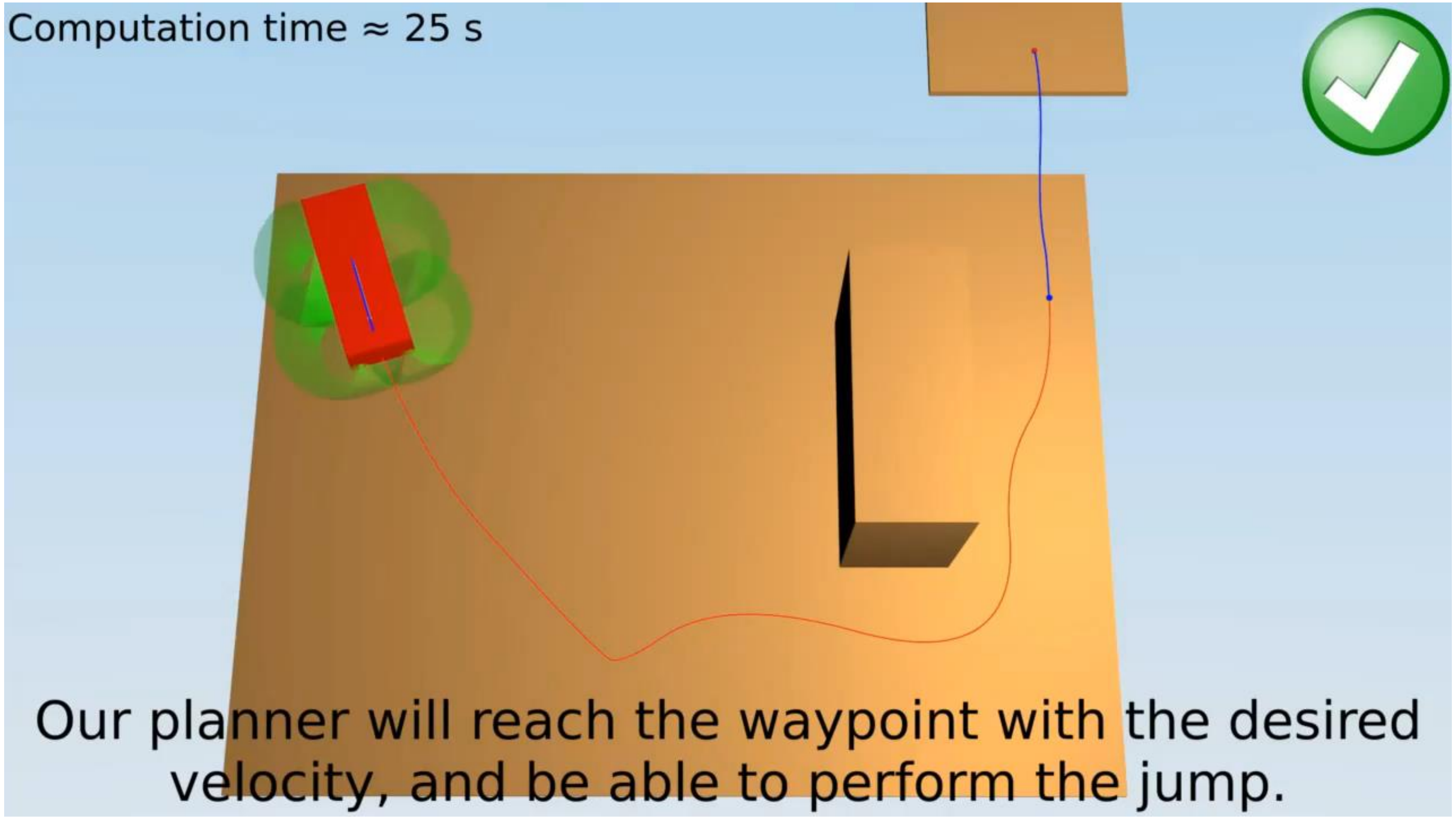
- Generating dynamic contact configurations ($\wp_2$)
  - Using same approach from paper 1, but in this work, authors are using an LP solving dynamic constraints to generate dynamically feasible contact configurations.

- Trajectory interpolation($\wp_3$) could be applied same with [Paper 1].

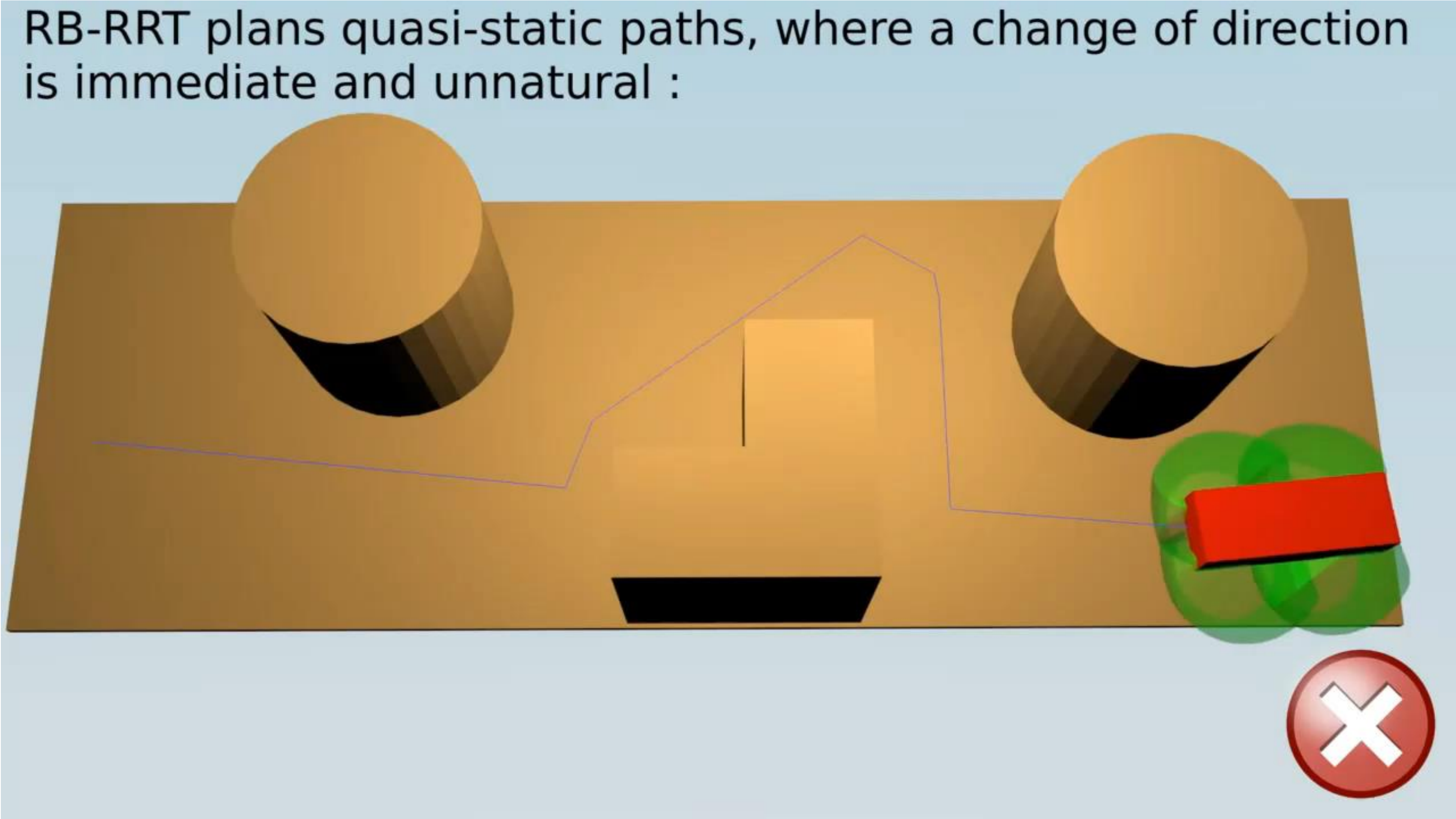
Result videos



Result videos



Result videos



KAIST

Conclusion

- They presented a method for synthesizing collision-free, various dynamic behaviors for multi-contact robots.
- Their contributions are
  - An extension of the static equilibrium contact planner to dynamic cases, faster than previous approaches.
  - An efficient LP to determine the acceleration bounds on the COM of a robot, given its active contacts and a desired direction of acceleration.