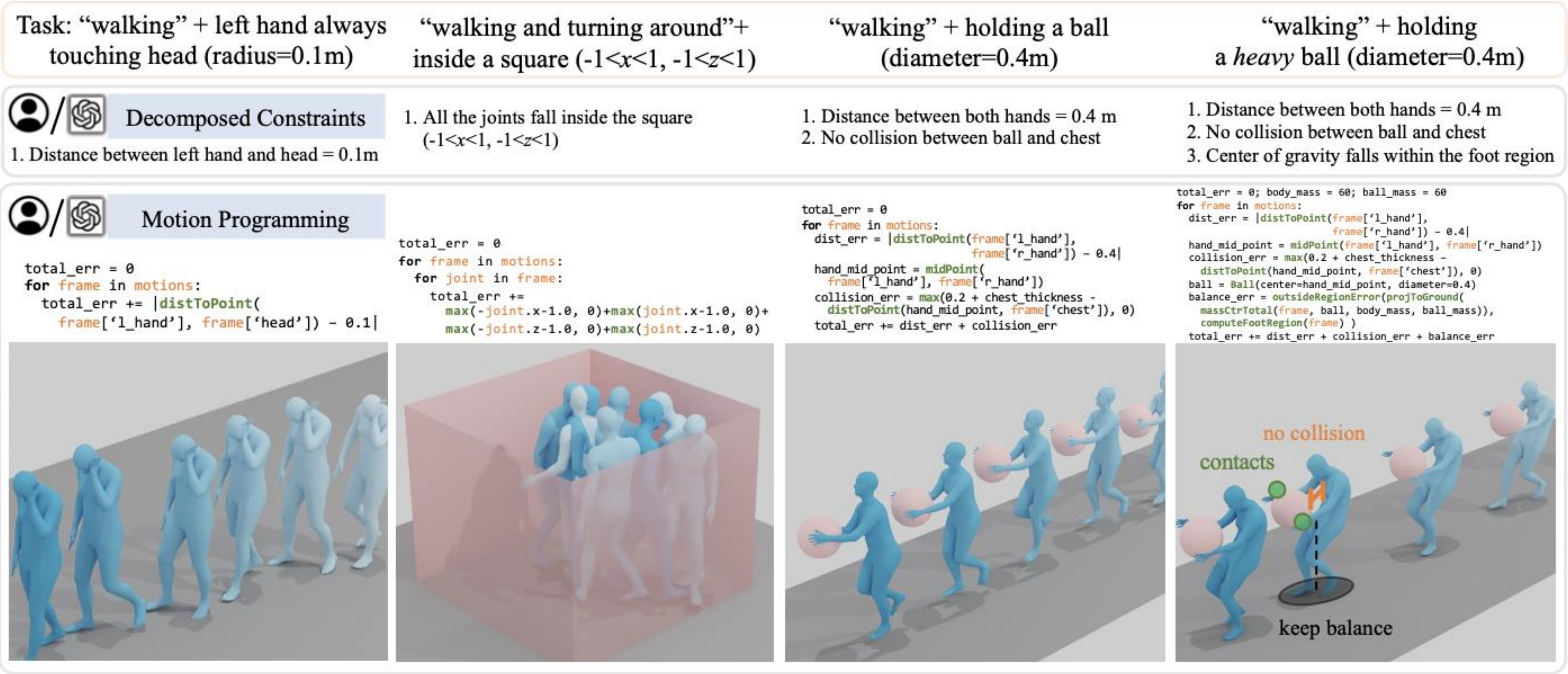


Programmable Motion Generation for Open-Set Motion Control Tasks

In CVPR 2024 Highlight

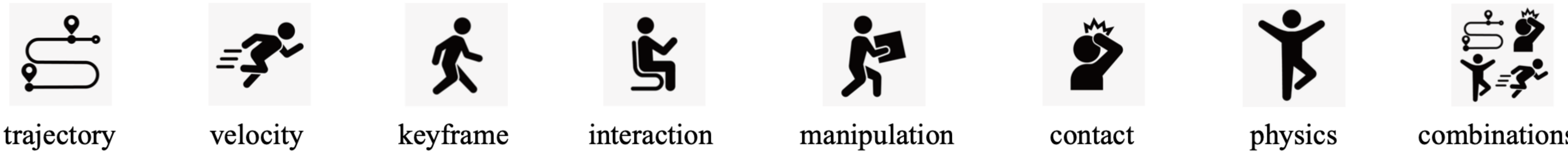
Hanchao Liu, Xiaohang Zhan, Shaoli Huang, Tai-Jiang Mu, Ying Shan
BNRist Tsinghua University Tencent AI Lab

Contents

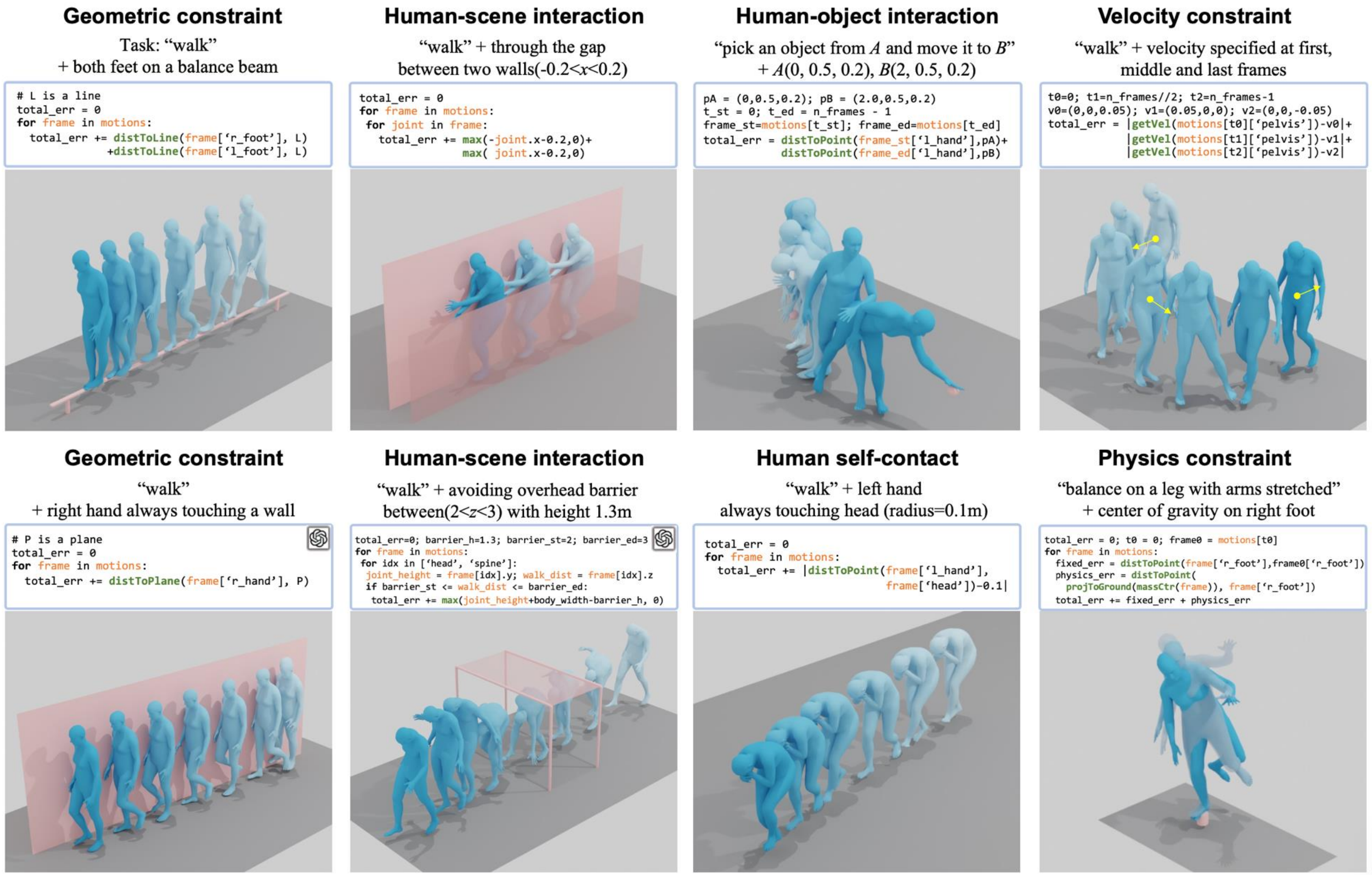
Introduction – Motivation

Human Motion = Subjective Behaviors + External Constraints

Constraints in Character Animation



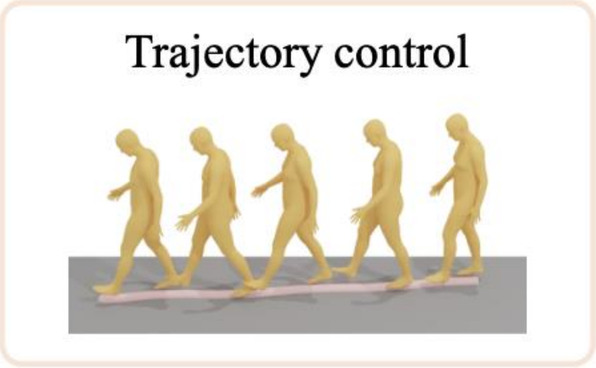
Human Motion = Subjective Behaviors + External Constraints

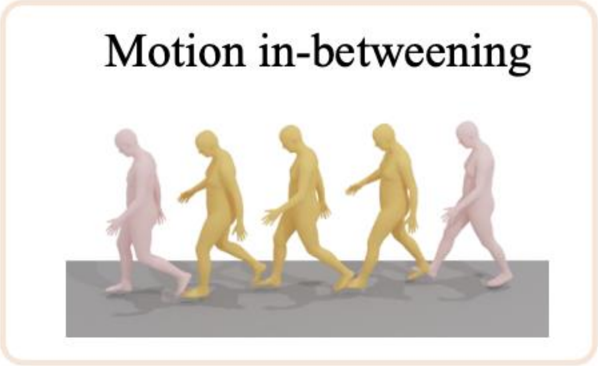


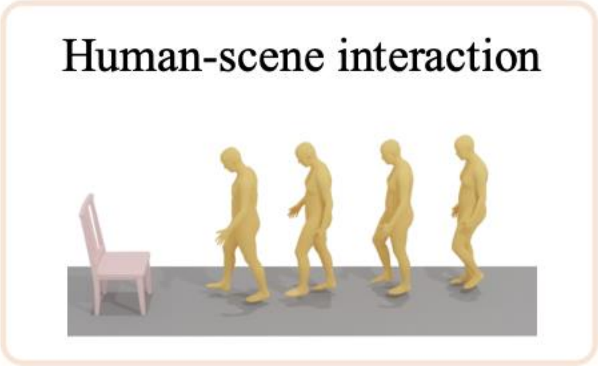
Introduction – Problem definition

Definition of Problem

Close-set Motion Control Problem







Open-set Motion Control Problem

? ? ? ? ?

Introduction – Problem definition

Close-set Motion Control Problem

pre-define single or a finite set of constraints and formulate it as individual tasks

<Close-set Motion Control Tasks List>

- Trajectories/velocity control
- Motion in-betweening
- Human-scene/object interactions
- physics-based animation
- etc

<Limitation: Previous ai-based animation method>

1. For each task, the dataset and the methodology are specifically designed and individually trained.
2. Those methods intrinsically cannot deal with customized constraints or arbitrary combinations of them.
3. seldom extendable or customizable.

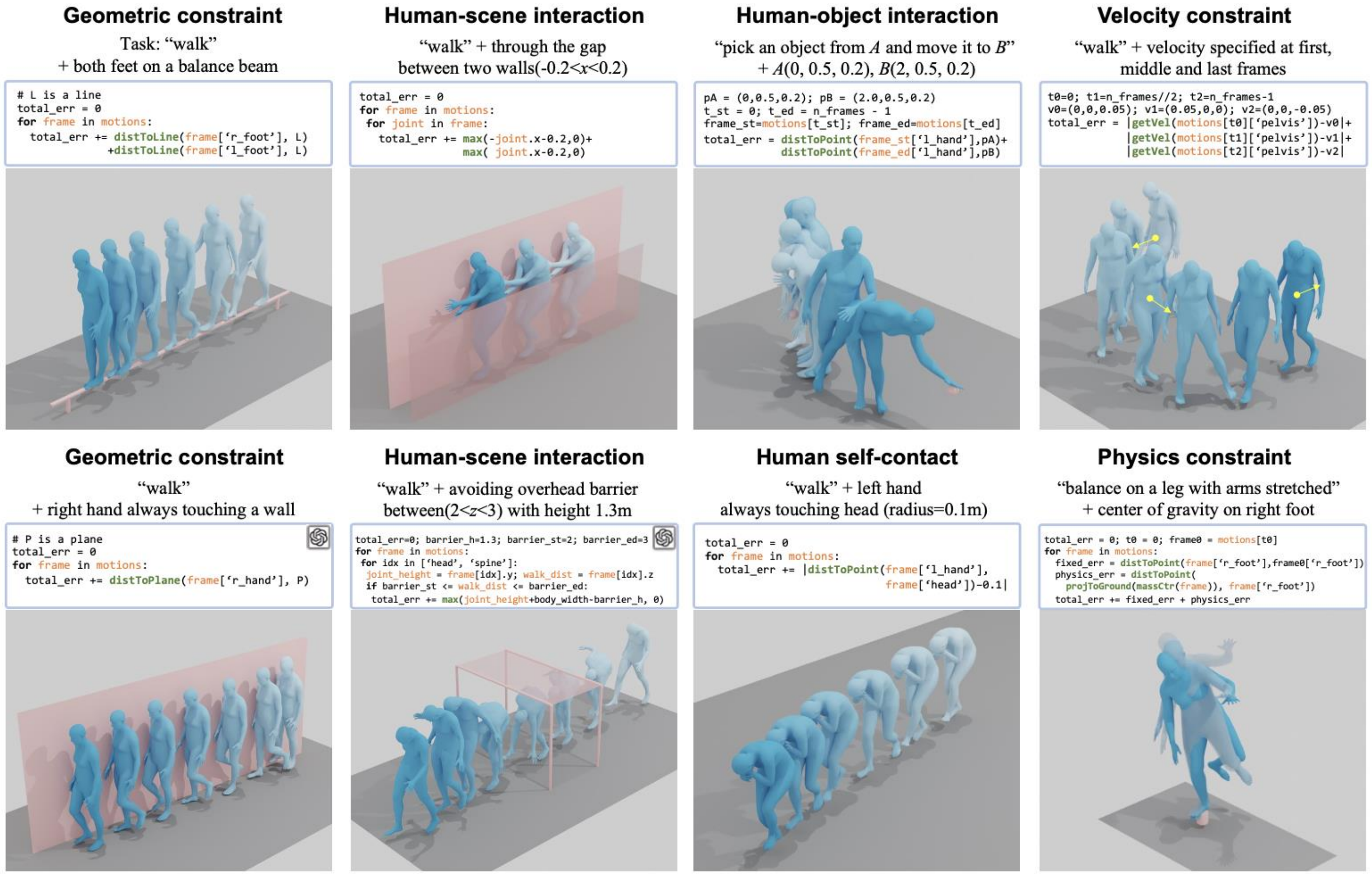
Introduction – Problem definition

Open-set Motion Control Problem

set of motion control tasks is open and fully customizable

<Open-set Motion Control Tasks List>

- Everything you want

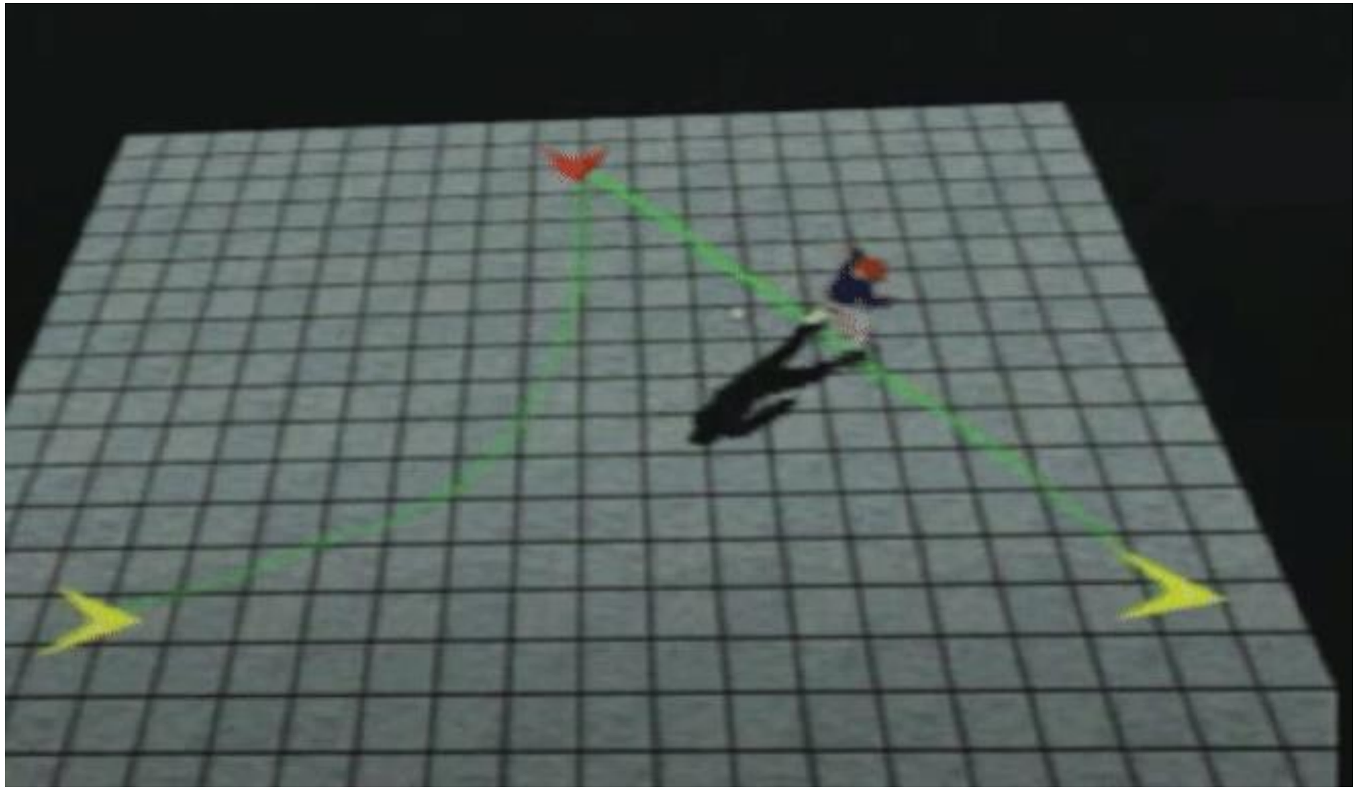


Introduction – Problem definition

Close-set Motion Control Problem

pre-define single or a finite set of constraints and formulate it as individual tasks

Trajectories/velocity control



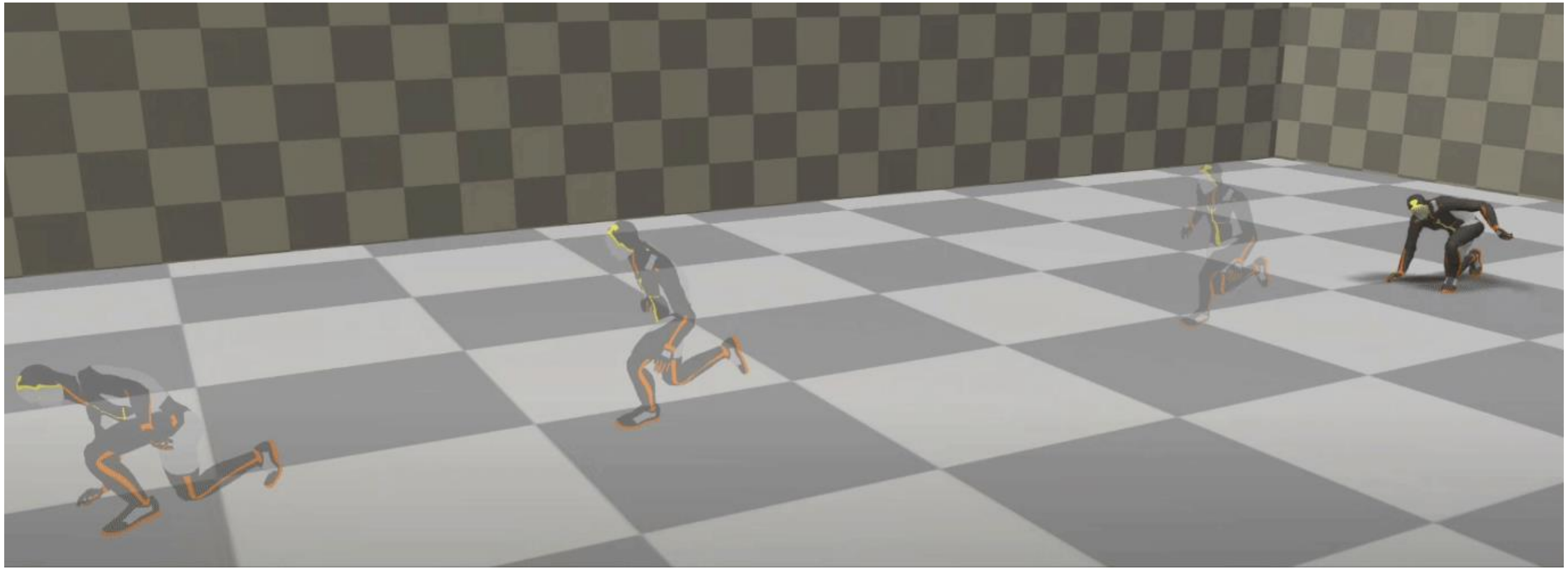
<Interactive motion generation from examples, *ACM Transactions on Graphics (TOG)*, 2002>

Introduction – Problem definition

Close-set Motion Control Problem

pre-define single or a finite set of constraints and formulate it as individual tasks

Motion in-betweening



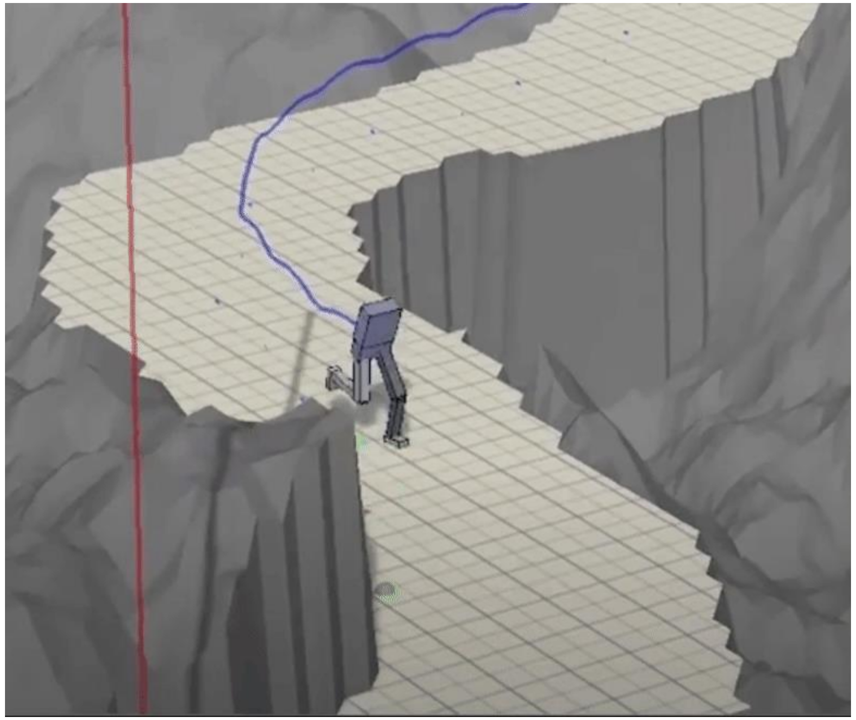
<Robust motion in-betweening. *ACM Transactions on Graphics (TOG)*, 2020>

Introduction – Problem definition

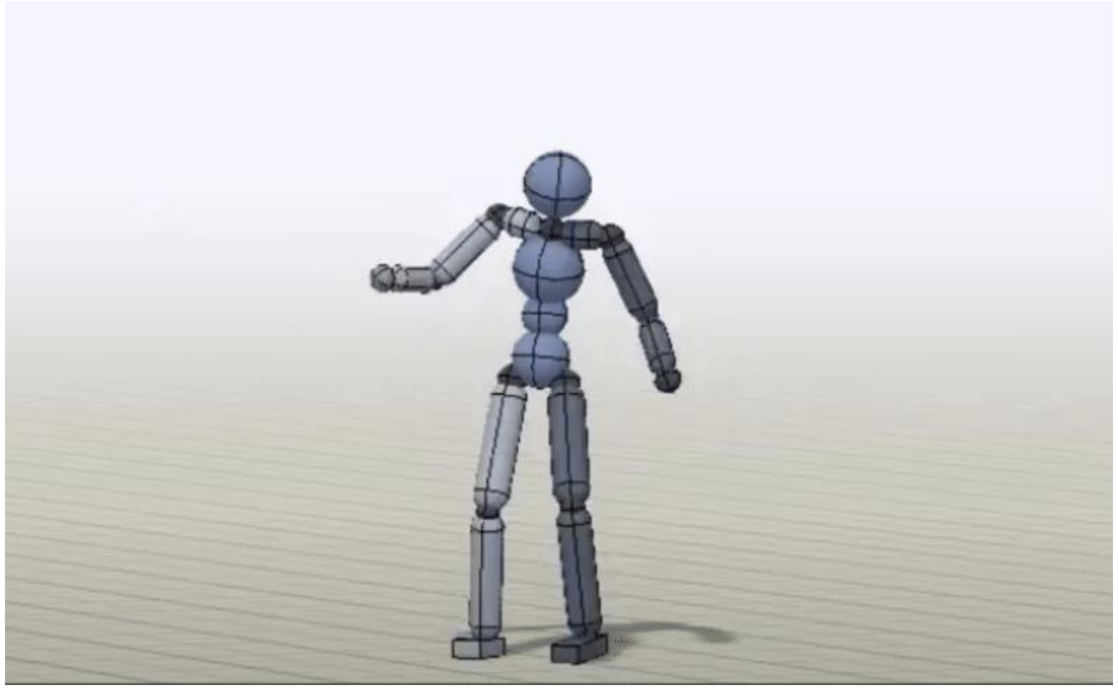
Close-set Motion Control Problem

pre-define single or a finite set of constraints and formulate it as individual tasks

Physics-based animation



<Deeploco, *ACM Transactions on Graphics (TOG)*, 2017>

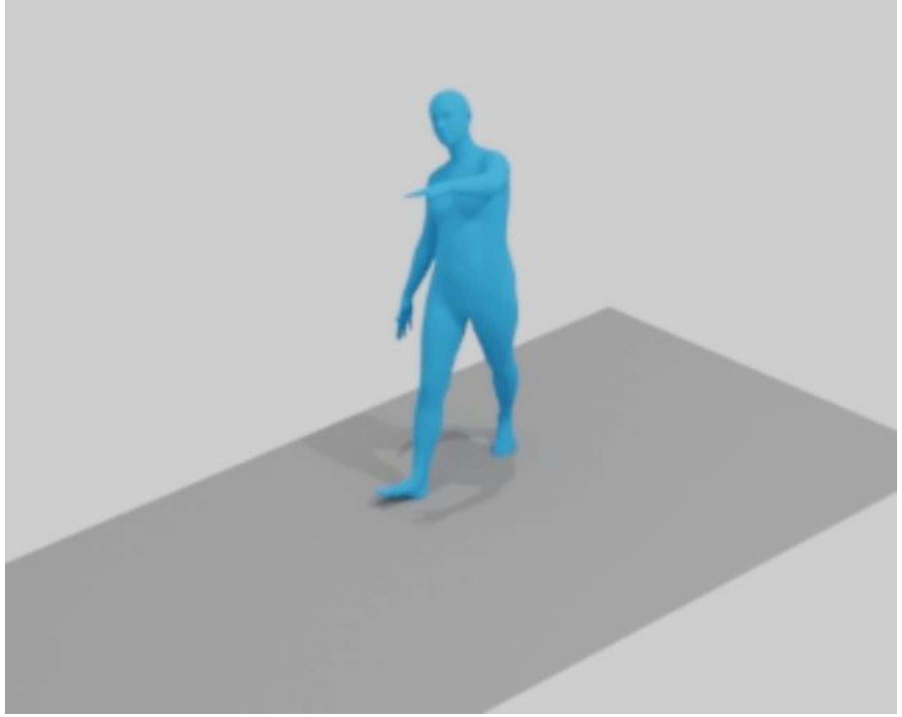


<Deepmimic, *ACM Transactions On Graphics (TOG)*, 2018>

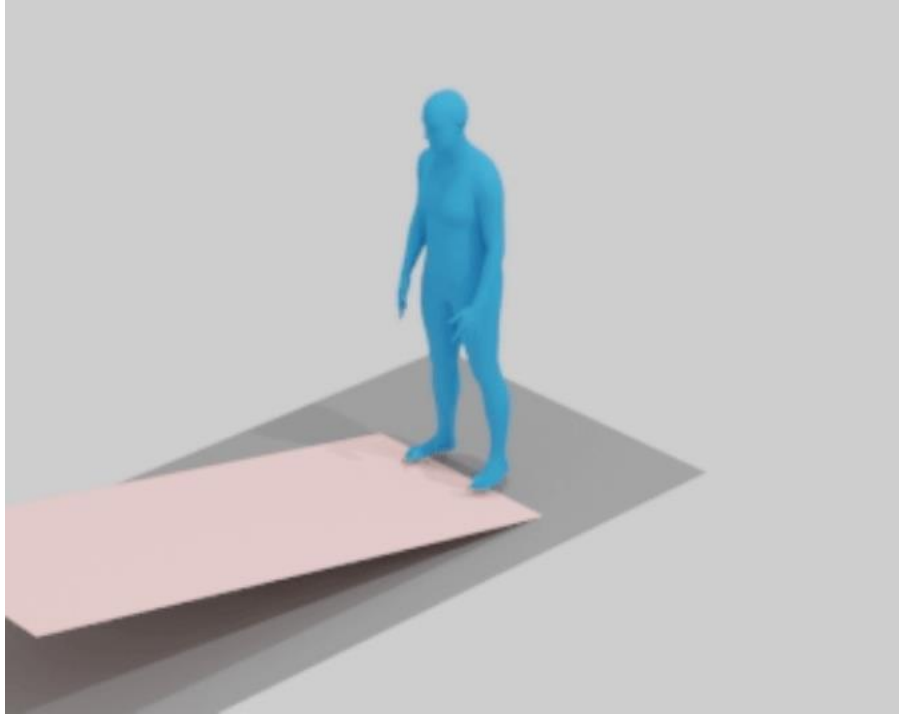
Introduction – Problem definition

Open-set Motion Control Problem

It is the Set of motion control tasks that is open and fully customizable



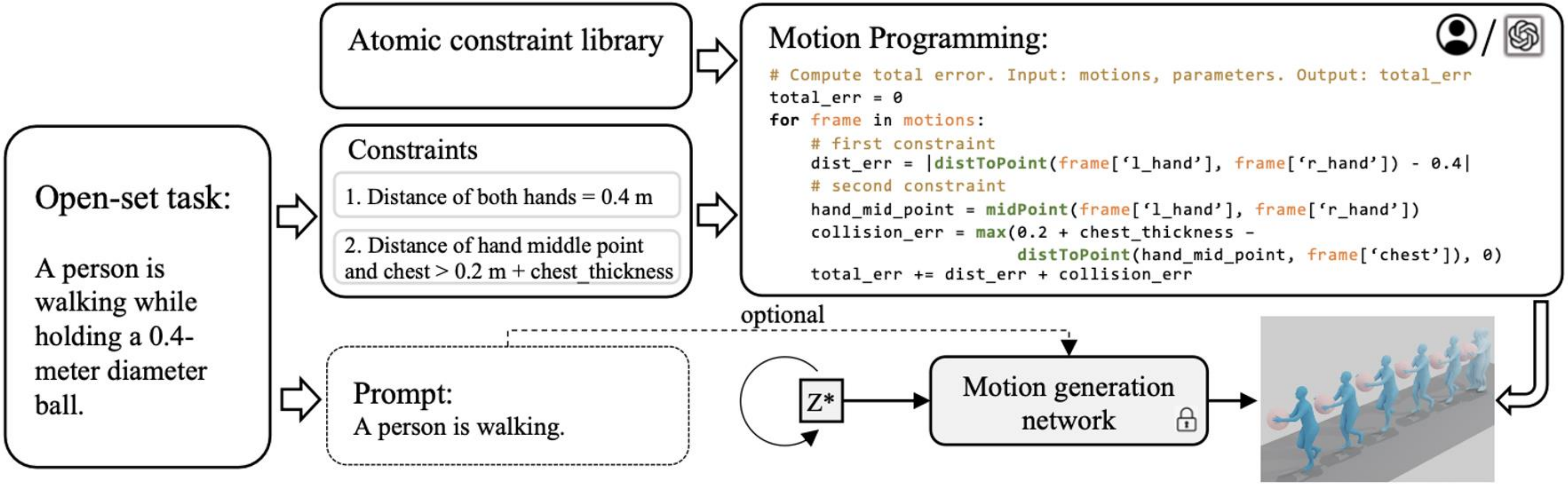
Task: walking + upper left



Task: walking on an inclined plane

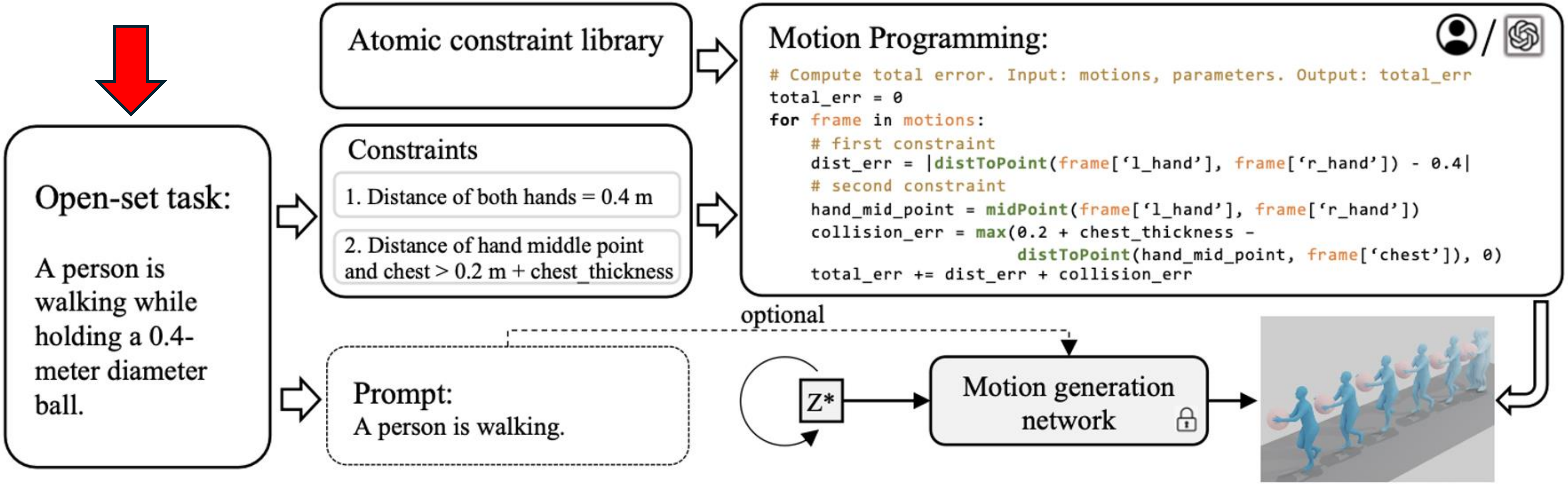
Method - Overview

Overview of Programmable Motion Generation



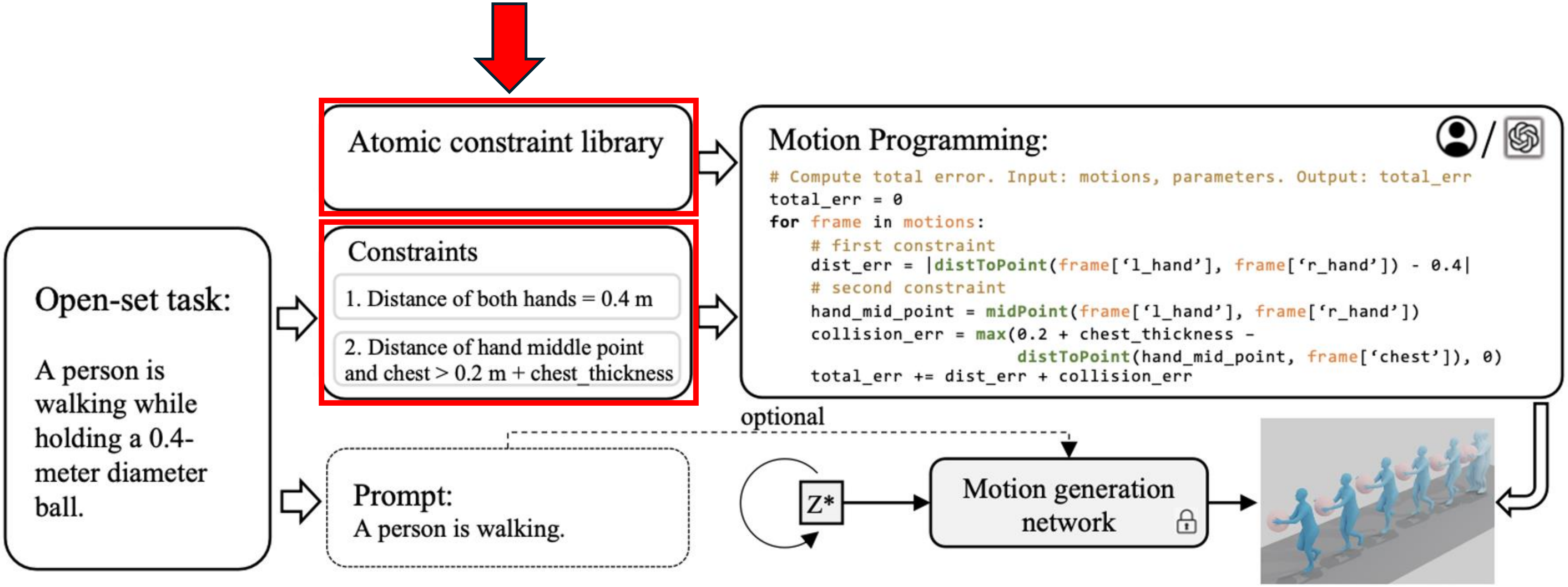
Method - Overview

Overview of Programmable Motion Generation



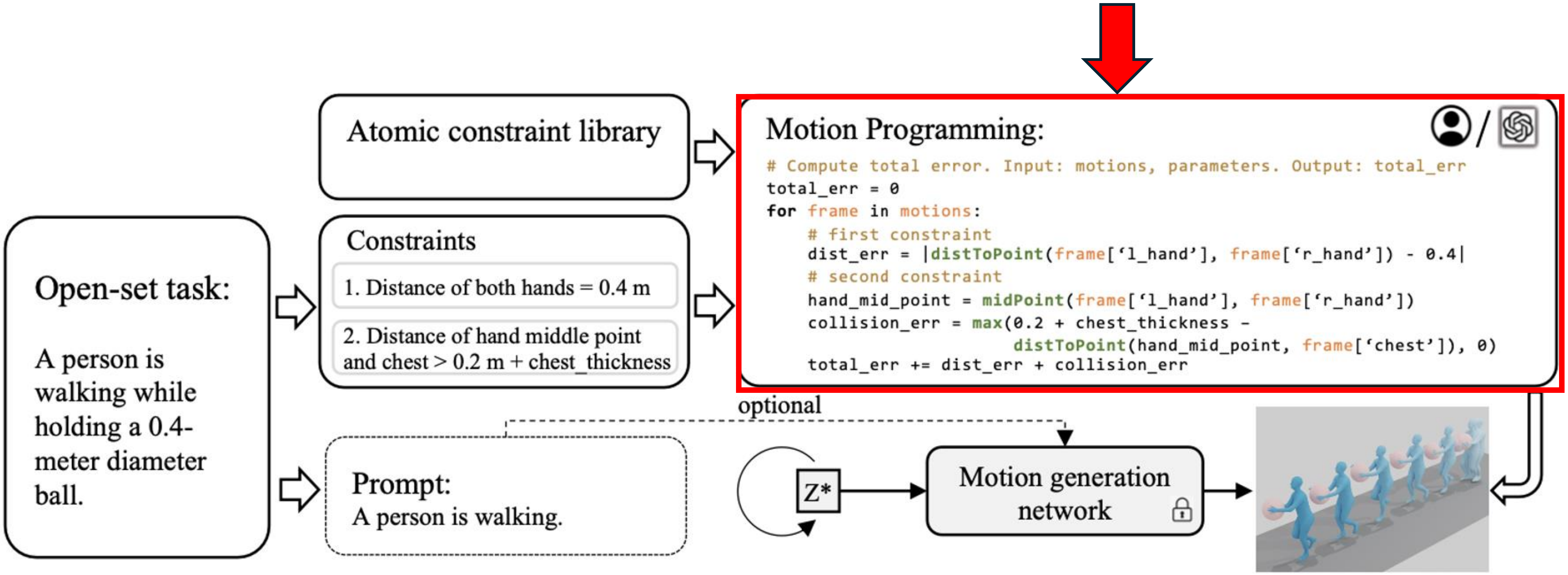
Method - Overview

Overview of Programmable Motion Generation



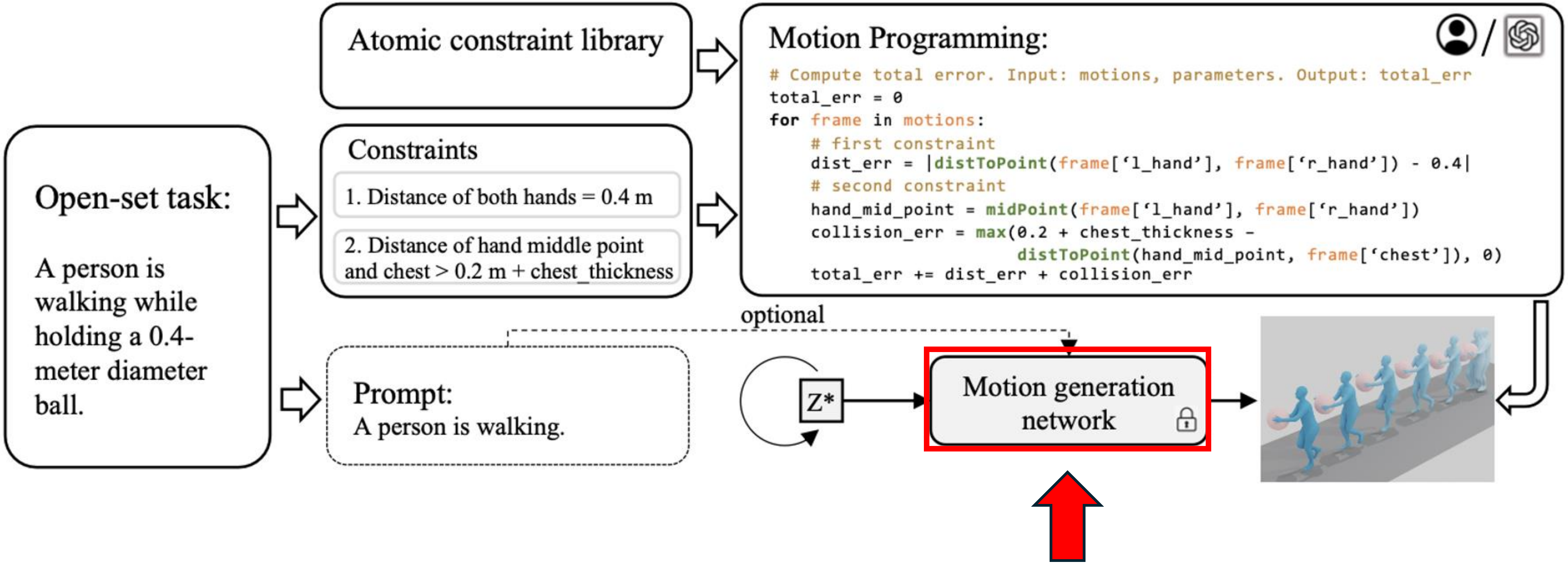
Overview of Programmable Motion Generation

Total Error Function for evaluating the generated motion



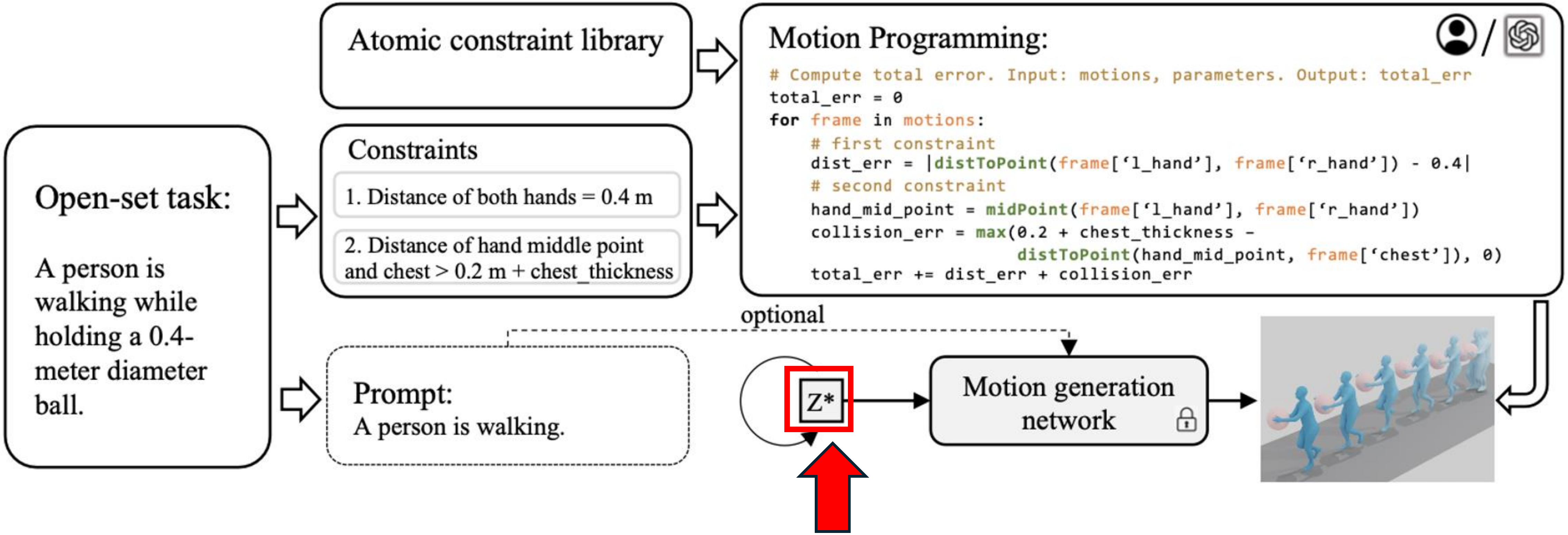
Method - Overview

Overview of Programmable Motion Generation



Method - Overview

Overview of Programmable Motion Generation

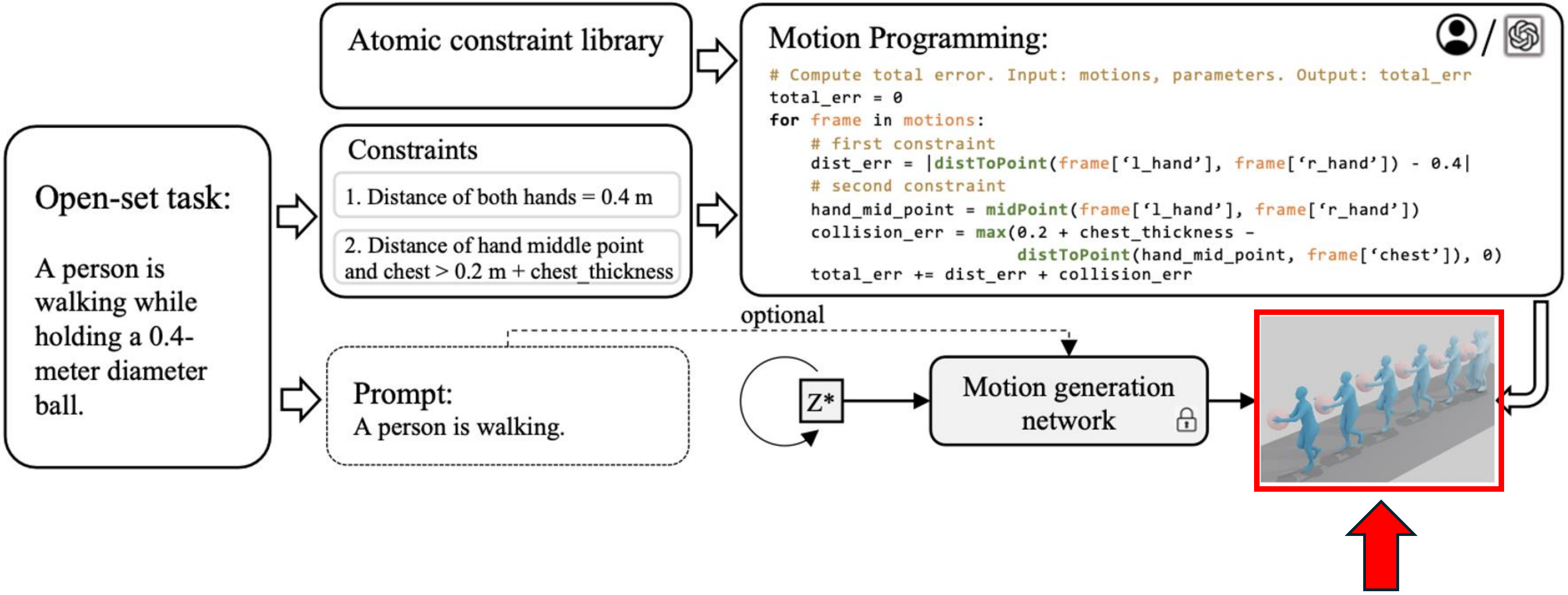


Find optimal Z(single vector)

optimization problem : $\min_{z} F(G_{\theta}(z, C), p)$

Method - Overview

Overview of Programmable Motion Generation



Method - Overview

Motion Programming(Total Error Function of the Constraints)

Motion Programming:

```
# Compute total error. Input: motions, parameters. Output: total_err
total_err = 0
for frame in motions:
    # first constraint
    dist_err = |distToPoint(frame['l_hand'], frame['r_hand']) - 0.4|
    # second constraint
    hand_mid_point = midPoint(frame['l_hand'], frame['r_hand'])
    collision_err = max(0.2 + chest_thickness -
                        distToPoint(hand_mid_point, frame['chest']), 0)
    total_err += dist_err + collision_err
```

Constraint Error Function *F*

Input: motions, parameters **Output:** a scalar value

```
def computeTotalError(motions, parameters):
    diameter, chest_thickness = parameters
    total_err = 0
    for frame in motions:
        dist_err = |distToPoint(frame['l_hand'], frame['r_hand']) - diameter|
        hand_mid_point = midPoint(frame['l_hand'], frame['r_hand'])
        collision_err = max(diameter / 2 + chest_thickness -
                            distToPoint(hand_mid_point, frame['chest']), 0)
        total_err += dist_err + collision_err
    return total_err
```

Programming Modules

Atomic Constraints		Logical Operations	
Absolute position	Relative distance	"=" E	AND E1+E2
High-order dynamics	Directional constraint	">" max(margin-E,0)	OR min(E1,E2)
Geometric constraint	Keyframe constraint	"<" max(E-margin,0)	NOT -E
…		…	

Method - Atomic Constraint library

Atomic Constraint library(+ Logical Operations)

Set of the Basic Constraints for generating various Human motions

<Atomic Constraint library>

- Absolute Position Constraint
- High-order Dynamic Constraint
- Geometric Constraint
- Relative Distance Constraint
- Directional Constraint
- key-frame Constraint
- etc

<Logical operations>

- ">"
- "<"
- "AND"
- "OR"
- "NOT"
- etc

Method - Atomic Constraint library

Atomic Constraint library(+ Logical Operations)

Set of the Basic Constraints for generating various Human motions

<Atomic Constraint library>

- **Absolute Position Constraint** requires the trajectory x_j^{pos} to be close to a given trajectory $\hat{x}_j^{pos}$ e.g.(L-n norms)

- **High-order Dynamic Constraint** constrains motion dynamics of joints(not position) e.g.(velocity, acceleration of certain joints)

- **Geometric Constraint** constrain a joint on a geometric primitive P in the global coordinate system e.g.(curve, surface)

- **Relative Distance Constraint** models relationships between two joints

- **Directional Constraint** requires a bone consisting of x_j and its parent joint $\mathrm{parent}(x_j)$ to point at a given direction

- **Key-frame Constraint** enforces constraint at certain timesteps

Method – Motion Programming Framework

Motion Programming Framework

<Input>

- motions
- parameters

Motion Programming:

```
# Compute total error. Input: motions, parameters. Output: total_err
total_err = 0
for frame in motions:
    # first constraint
    dist_err = |distToPoint(frame['l_hand'], frame['r_hand']) - 0.4|
    # second constraint
    hand_mid_point = midPoint(frame['l_hand'], frame['r_hand'])
    collision_err = max(0.2 + chest_thickness -
                        distToPoint(hand_mid_point, frame['chest']), 0)
    total_err += dist_err + collision_err
```

<output>

- total error (scalar value)

- The "motions" is a list of dictionaries containing information of joints : {"joint_name" : (x, y, z)}
- The "parameters" includes task-related constants

Method – Motion Programming Framework

Task : pick object / constraint : first Frame(object position : point A), last frame(object position : point B)

```python
def compute_error(motion, point_A, point_B):
    total_error = 0

    # Constants and thresholds
    max_allowable_distance_A = 0.1  # Maximum allowable distance between left hand and point A in the first frame
    max_allowable_distance_B = 0.1  # Maximum allowable distance between left hand and point B in the last frame

    # Iterate through frames
    for frame_index, joints in enumerate(motion):
        # Get the left hand coordinates for the current frame
        left_hand = joints.get("left_hand", None)

        if left_hand is not None:
            # Geometric constraint: Distance from left hand to point A in the first frame
            if frame_index == 0:
                distance_left_hand_to_A = DistToPoint(left_hand, point_A)
                total_error += max(distance_left_hand_to_A - max_allowable_distance_A, 0)

            # Geometric constraint: Distance from left hand to point B in the last frame
            elif frame_index == len(motion) - 1:
                distance_left_hand_to_B = DistToPoint(left_hand, point_B)
                total_error += max(distance_left_hand_to_B - max_allowable_distance_B, 0)

            # You may add more constraints based on specific requirements for intermediate frames

    return total_error
```

(1)

(2)

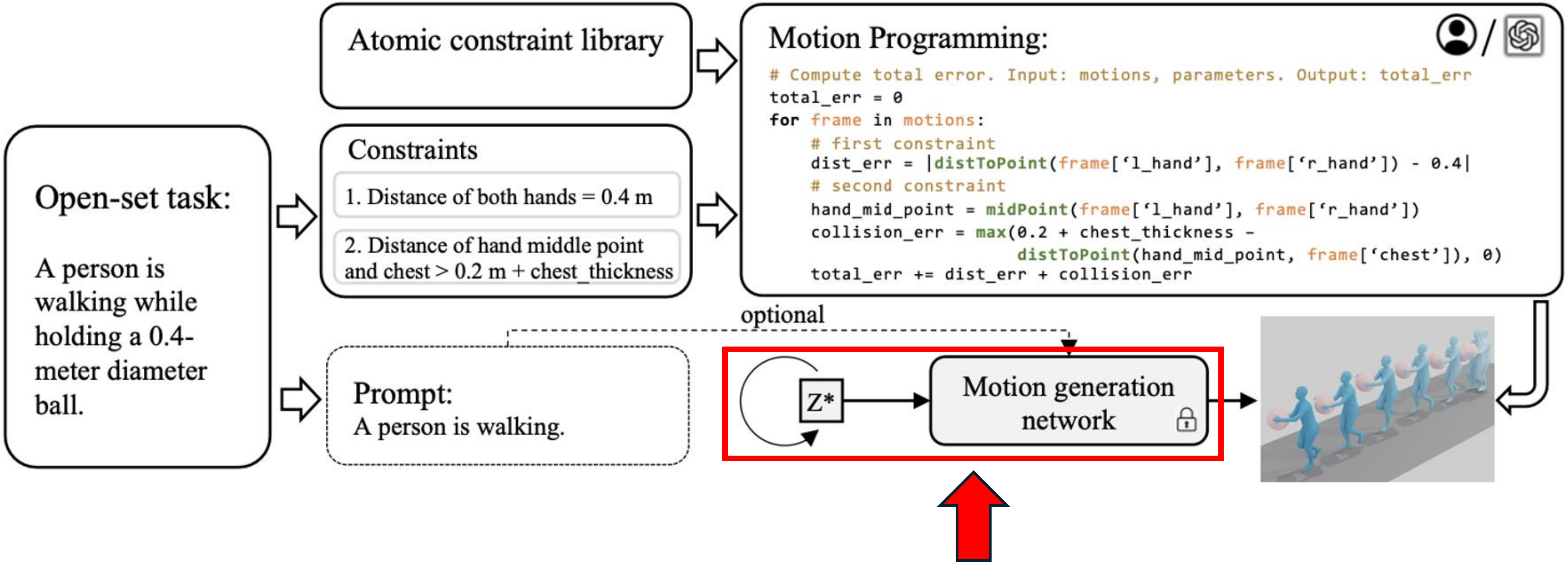
Example usage

```
# Example usage:
motion_data = [
    {"left_hand": (x1, y1, z1), ...},  # Frame 0
    {"left_hand": (x2, y2, z2), ...},  # Frame 1
    # ... (more frames)
    {"left_hand": (xn, yn, zn), ...},  # Last frame (Frame n)
]

point_A = (xA, yA, zA)
point_B = (xB, yB, zB)

total_error = compute_error(motion_data, point_A, point_B)
print("Total Error:", total_error)
```

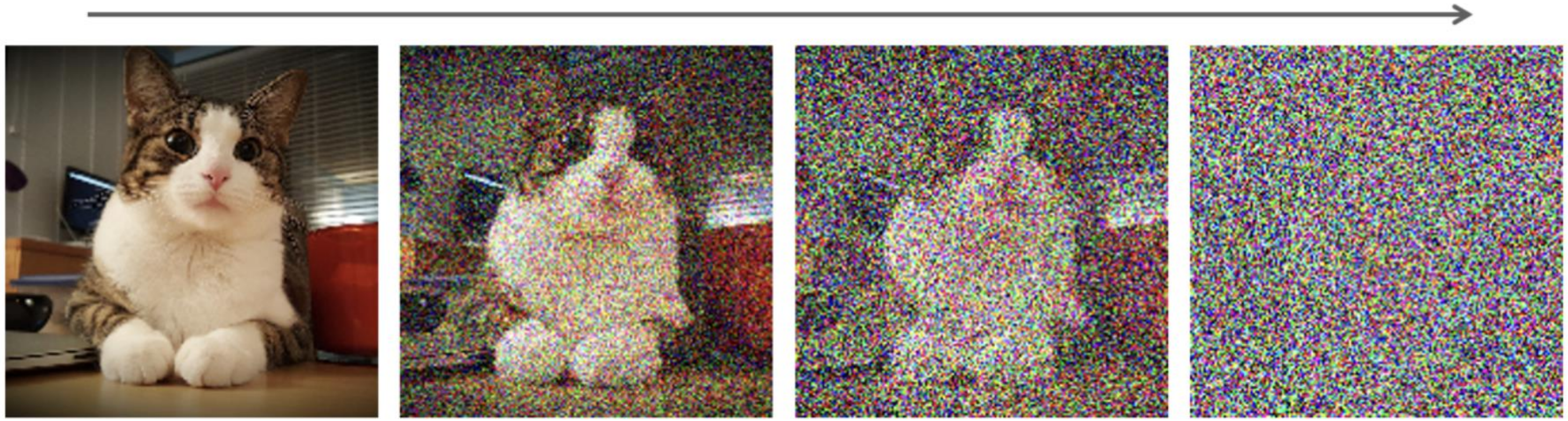
Method – Latent Noise Optimization



Method – Latent Noise Optimization

Diffusion model

Forward Diffusion Process



Reverse Denoising Process

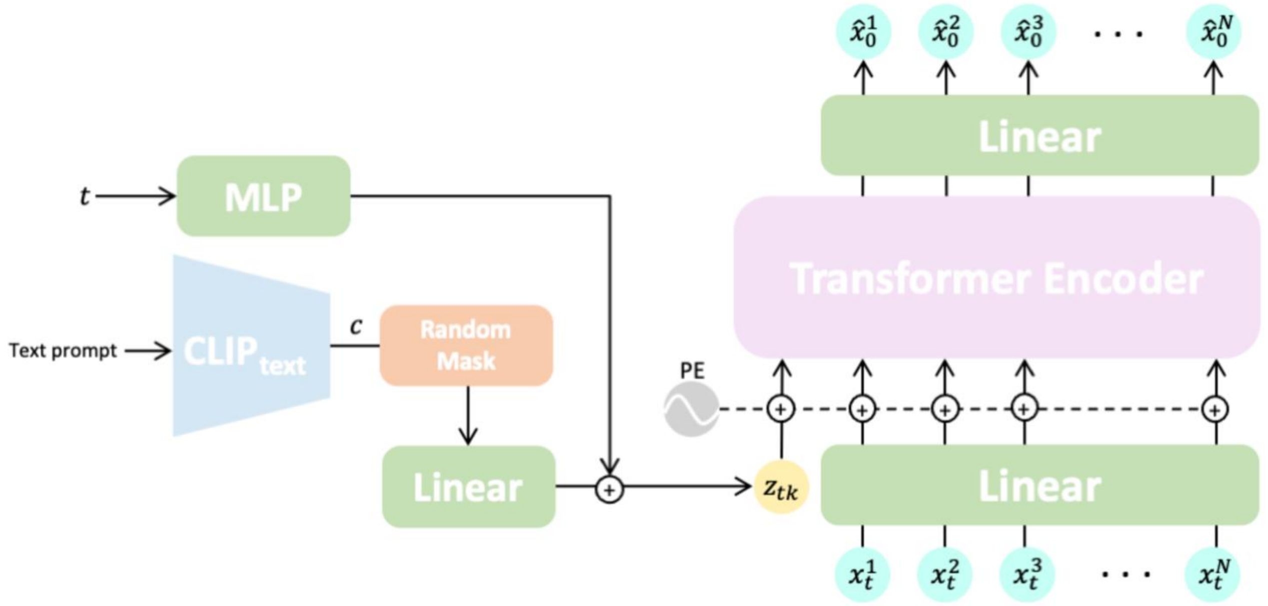
Noise following a specific distribution is added for each timestep.

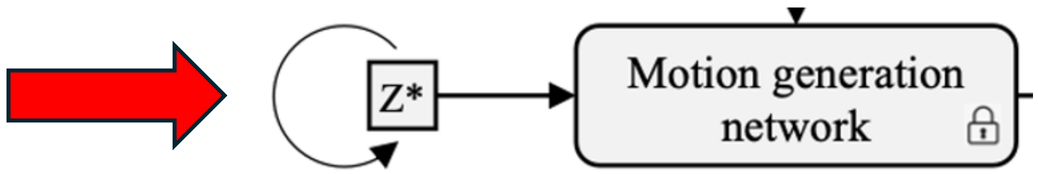
Goal : Creating a image with a probability distribution similar to the input image.

출처 : https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-1/

Method – Latent Noise Optimization

Motion Generation model





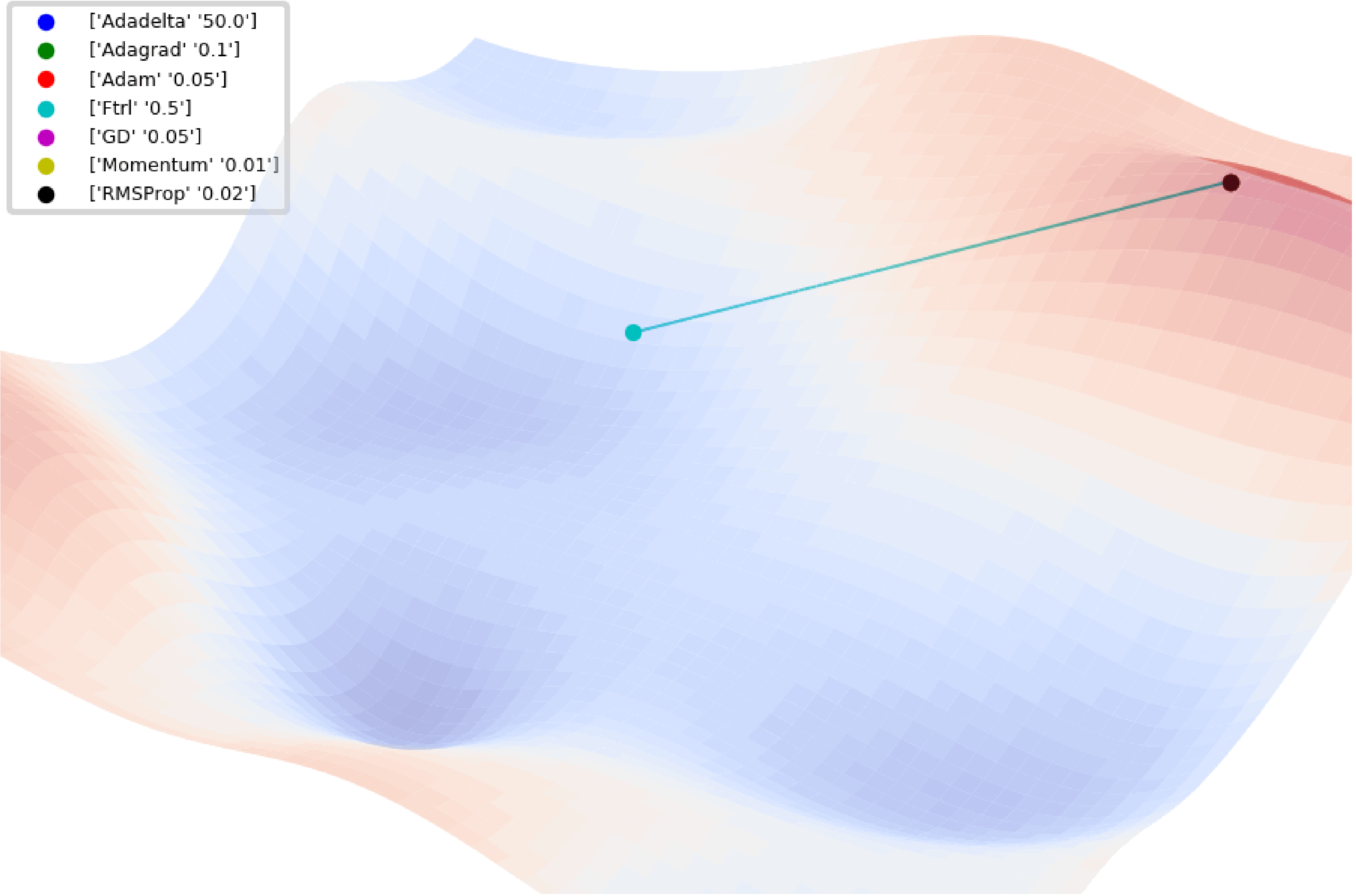
<Motion Diffusion Model(MDM)>

<Denosing Diffusion Implicit Model(DDIM)>
Z : latent noise that is single vector

optimization problem : $\min_{z} F(G_\theta(z, C), p)$

Mathis Petrovich, Michael J Black, and Gul Varol. Temos: ¨ Generating diverse human motions from textual descriptions. In European Conference on Computer Vision, pages 480– 497. Springer, 2022
Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In International Conference on Learning Representations, 2020

Method – Latent Noise Optimization



$\min_{z} F(G_\theta(z, C), p)$ -> Adam

https://github.com/j-w-yun/optimizer-visualization?tab=readme-ov-file

Experiments and Evaluation

Evaluation Metrics and Tasks

<Evaluation Metrics>

- Foot Skate

- Maximum Joint Acceleration(Max Acc)
-> To evaluate frame-wise consistency

-Constraint Error(C.Err)
-> MAE(Mean Absolute Error)

-Unsccess Rate(Unsucc. Rate)
-> The percentage of generated samples that fail to meaet all the constraints within 5cm threshold

-Frechet Inception Distance(FID)
-> To evaluate the quality of motion generation by measuring how similar the generated motion is to real motion data

-Diversity

- R-Precision(R-prec)
-> To evaluate how well the generated motion matches the conditions such as textual descriptions

<Evaluation Tasks>

- HSI-1 : head height constraint
- HIS-2 : avoiding barrier
- HIS-3 : walking inside a square
- GEO-1 : hand touching wall
- HOI-1 : moving object

Experiments and Evaluation

Evaluation

Task HSI-1: head height constraint							
Method	Foot Skate ↓	Max Acc. ↓	C.Err. ↓	Unsucc. Rate ↓	FID ↓	Diversity →	R-prec. (Top3) ↑
MDM (Unconstrained) [38]	0.086	0.097	0.118	0.718	0.545	9.656	0.610
MDM Edit [38]	0.094	0.148	0.109	0.645	0.554	9.656	0.614
IK	0.093	0.414	0.012	0.088	0.545	9.653	0.610
IK+Reg.	0.269	0.121	0.012	0.088	0.782	9.509	0.603
Ours	0.075	0.094	0.012	0.088	0.556	9.611	0.597

	Task HSI-2: avoiding barrier			Task HSI-3: walking inside a square		
Method	Foot Skate ↓	Max Acc. ↓	C.Err. ↓	Foot Skate ↓	Max Acc. ↓	C.Err. ↓
MDM (Unconstrained) [38]	0.096	0.126	0.454	0.096	0.126	0.301
IK	0.132	1.919	0.047	0.139	0.292	0.015
IK+Reg.	0.589	0.361	0.047	0.215	0.128	0.015
Ours	0.189	0.150	0.097	0.125	0.093	0.012

HIS-{number} : Human-Scene Interaction

HIS-1 : constraining the head heights on the first, central and last frames.
This task uses "**geometric constraint**" and "**key-frame constraint**".

Experiments and Evaluation

Evaluation

	Task GEO-1: hand touching wall			Task HOI-1: moving object		
Method	Foot Skate ↓	Max Acc. ↓	C.Err. ↓	Foot Skate ↓	Max Acc. ↓	C.Err. ↓
MDM (Unconstrained) [38]	0.096	0.126	0.233	0.029	0.026	1.701
MDM Edit [38]	0.161	0.147	0.141	0.029	0.032	1.739
PriorMDM [35]	0.350	0.197	0.185	0.327	0.213	1.884
IK	0.147	0.187	0.010	0.408	0.919	0.011
IK+Reg.	0.536	0.117	0.010	0.405	0.037	0.011
Ours	0.110	0.104	0.023	0.114	0.068	0.028

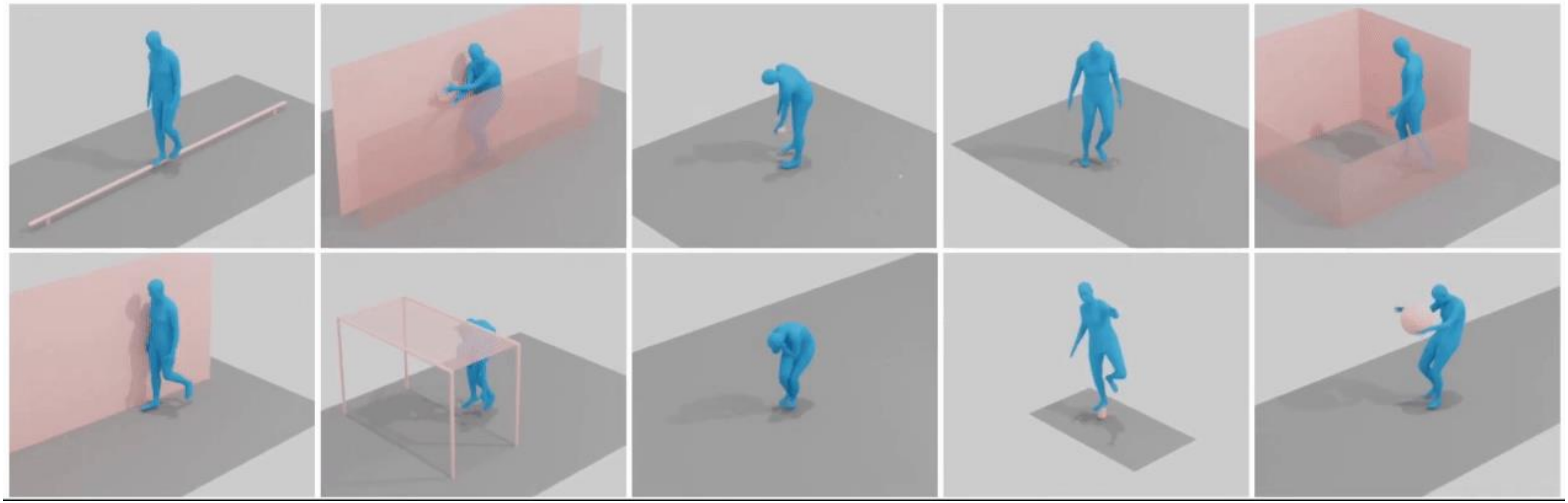
GEO-{number} : Motion Control with Geometric Constraints
HOI-{number} : Human-Object Interaction

GEO-1 : walking with hand touching a vertical wall

HOI-1 : moving an object from on place to another. Both starting and end positions for the controlled hand are specified. This task uses **"absolute position constraints"** and **"key-frame constraint"**.

Experiments and Evaluation

Evaluation



Reference

https://hanchaoliu.github.io/Prog-MoGen/

Okan Arikan and David A Forsyth. Interactive motion gen- eration from examples. *ACM Transactions on Graphics (TOG)*, 21(3):483–490, 2002.

Fe´lix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion in-betweening. *ACM Trans- actions on Graphics (TOG)*, 39(4):60–1, 2020.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforce- ment learning of physics-based character skills. *ACM Trans- actions On Graphics (TOG)*, 37(4):1–14, 2018.

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5152–5161, 2022.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denois- ing diffusion implicit models. In *International Conference on Learning Representations*, 2020

https://developer.nvidia.com/blog/improving-diffusion-models-as-an-alternative-to-gans-part-1/

https://github.com/j-w-yun/optimizer-visualization?tab=readme-ov-file