GDC

From Motion Matching to Motion Synthesis

Fabio Zinno
Sr. Software Engineer – Electronic Arts

GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19

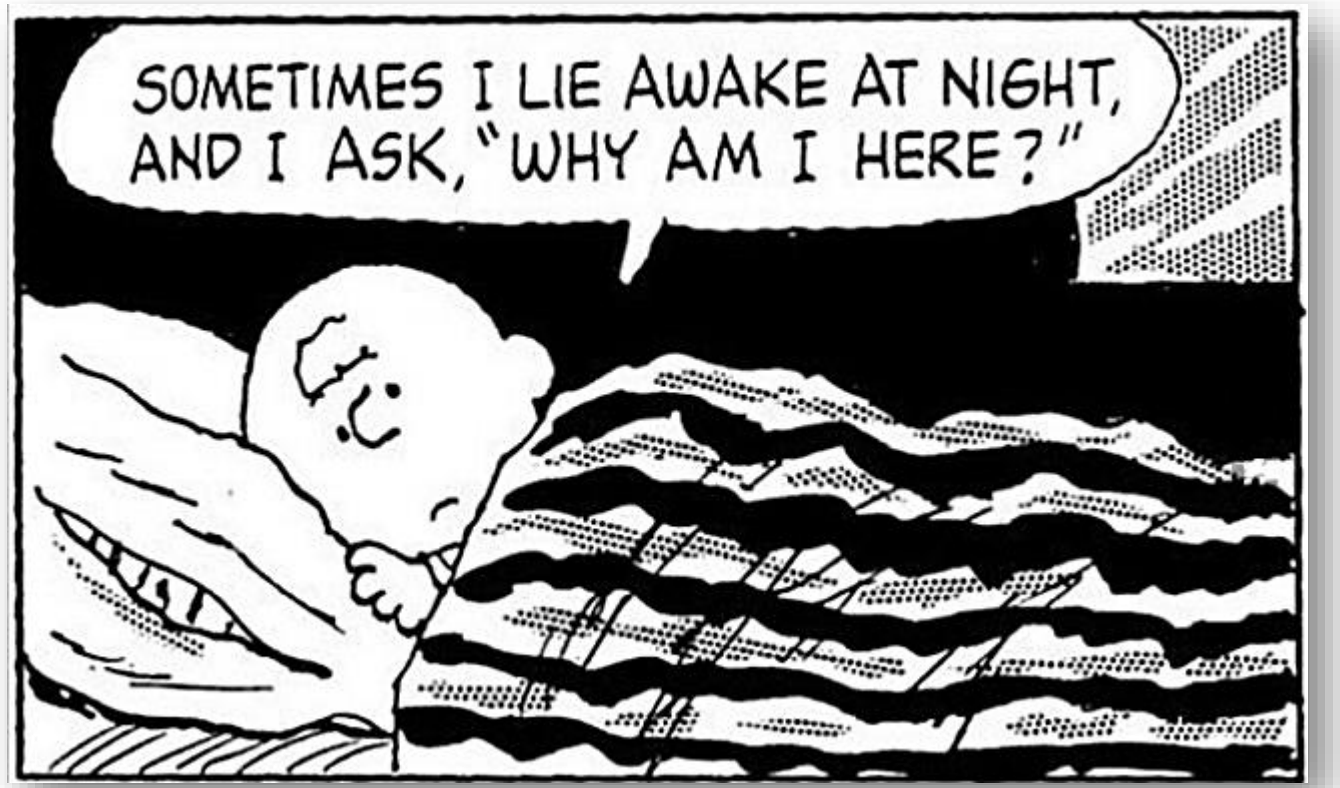
Why are you here?

Motion Synthesis state of the art:

- **PFNN** paper/**MANN** paper
  - spelled out details
  - tips to avoid common pitfalls
- experiments

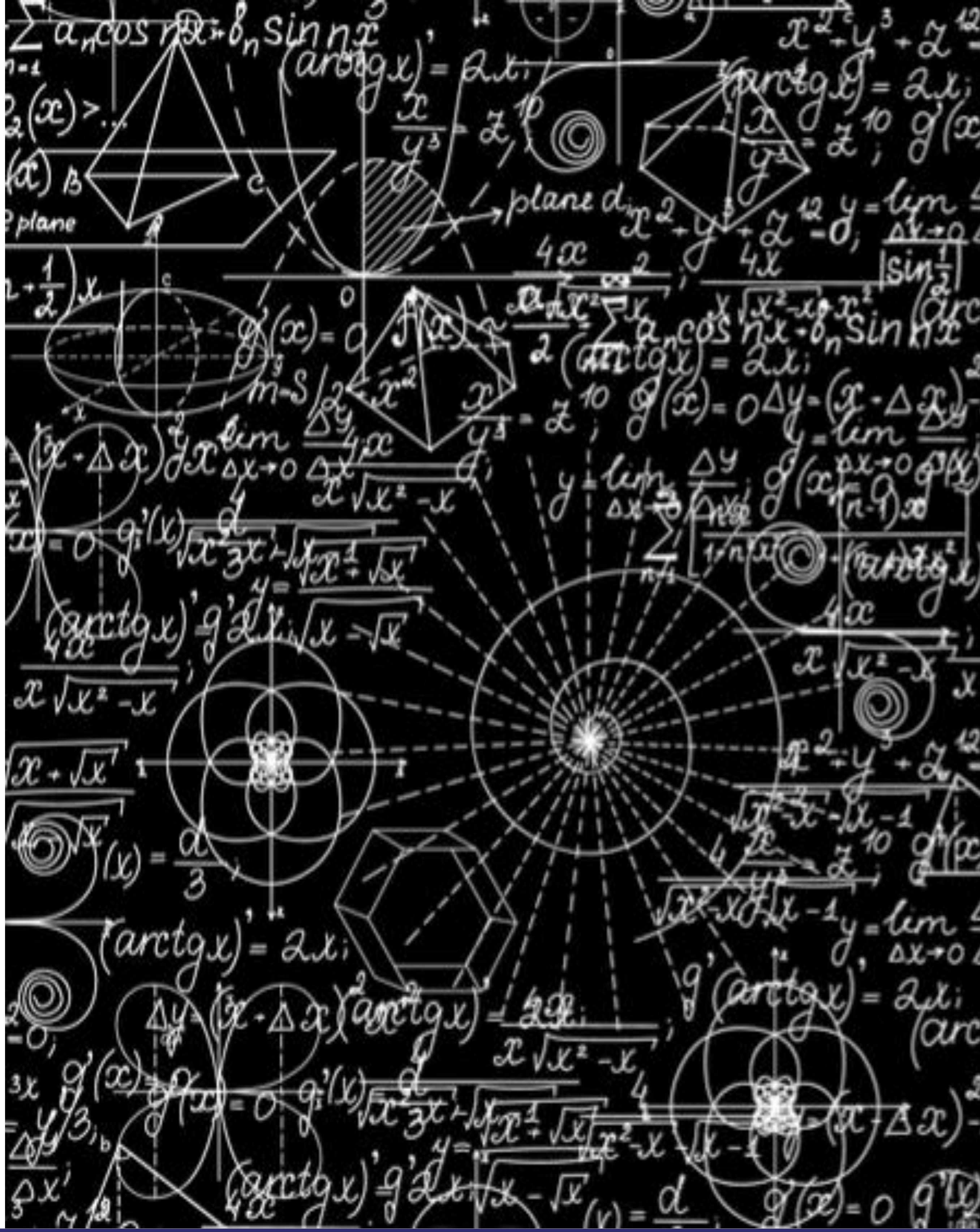
Discussion:

- Motion Synthesis in development
- research in the game industry



The problem

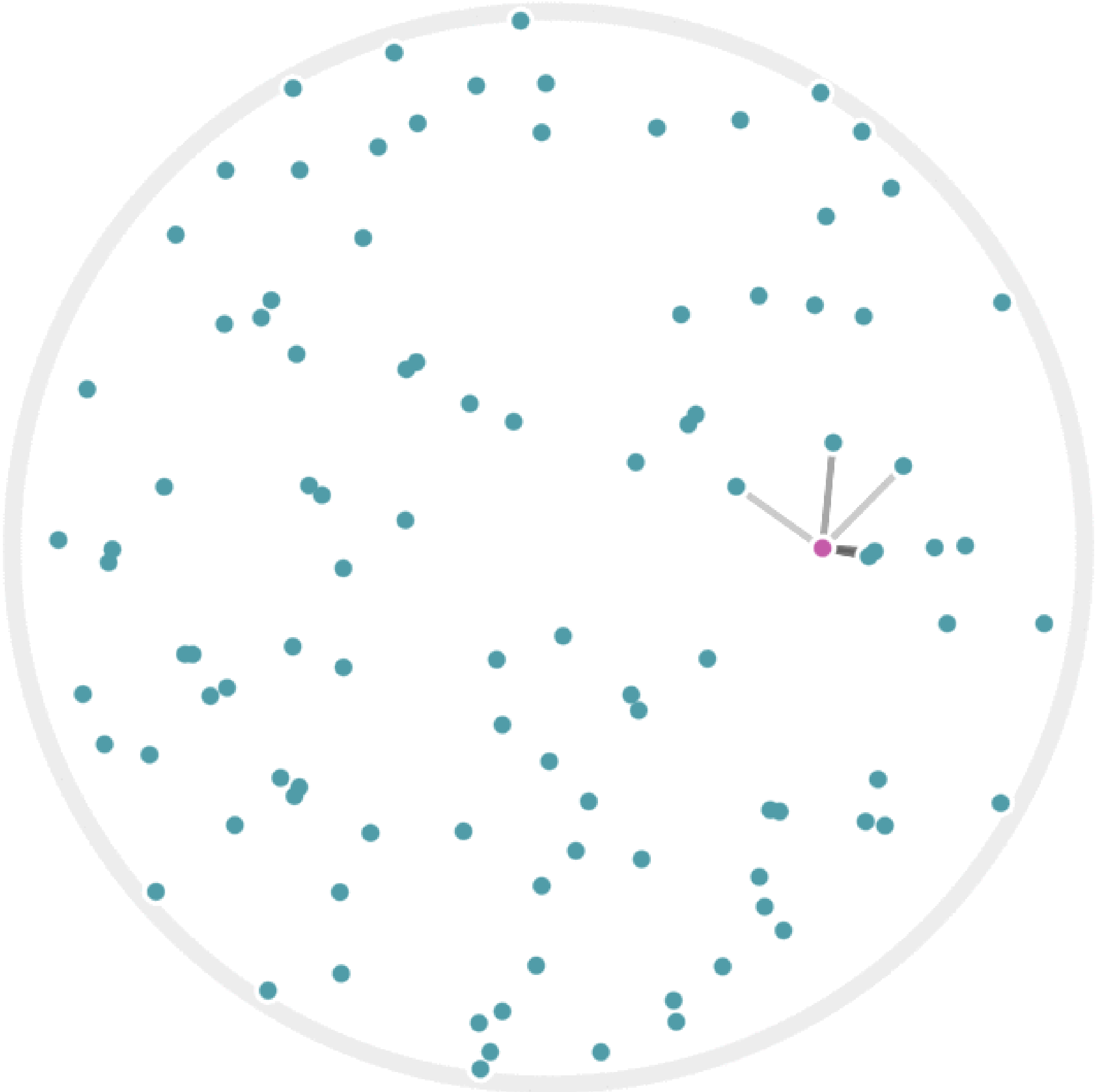
- moCap is high quality data but…
  - …we can’t capture everything!
- motion trees:
  - time consuming…
  - …and complicated
- blending, IK, procedural animation:
  - this is where we lose quality!



PES 2019 vs FIFA 19 | Realism Animation Comparison | Fujimarupes

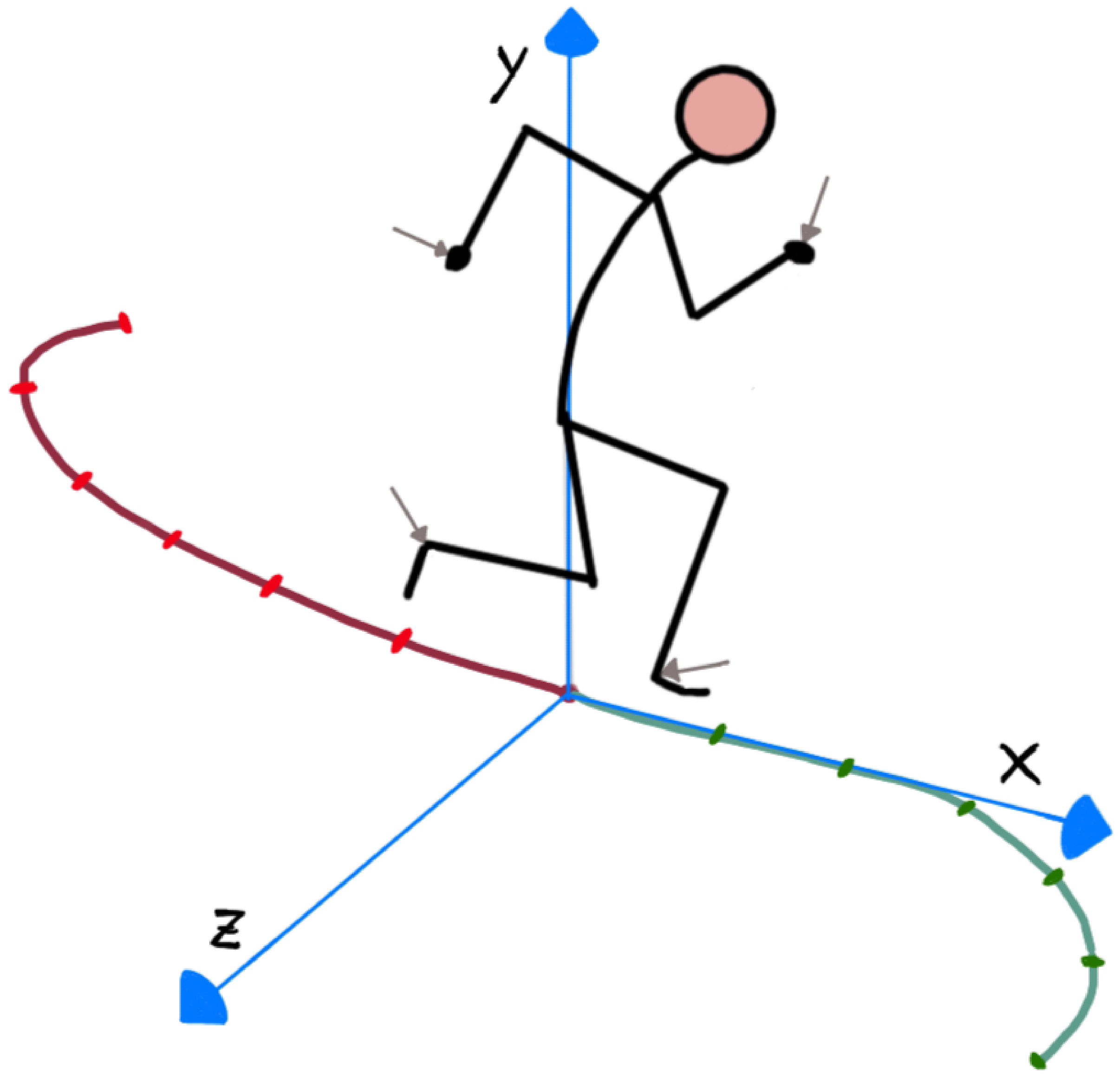
Motion Matching

- **knn** over database of poses
- query: pose + future/past trajectory
- new choice (possibly) every frame



Motion Matching - query

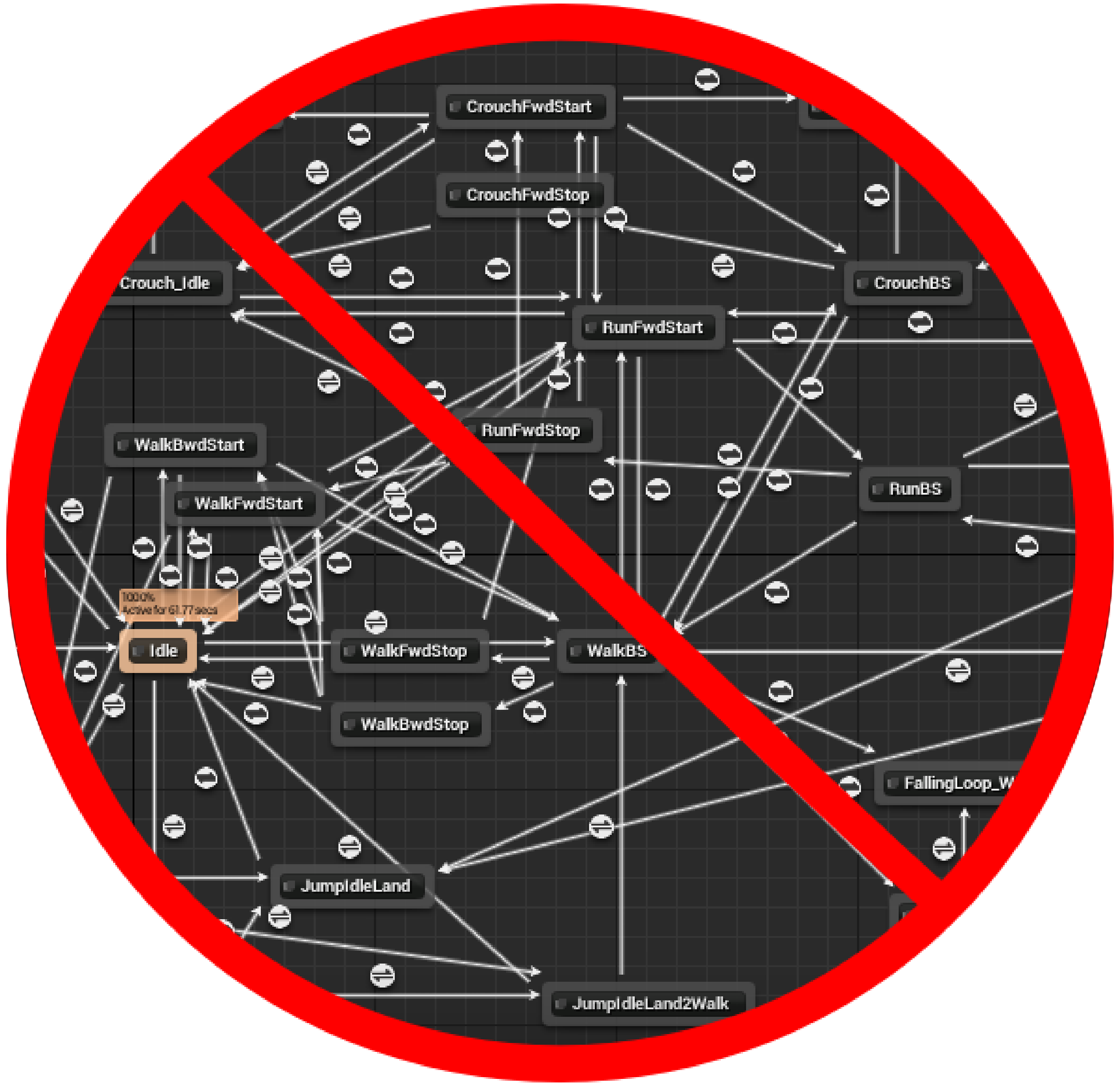
- **pose:**
  - joints positions + velocities
  - root facing and velocities
  - everything in character root space!
- **trajectory:**
  - 1 second samples in the future
  - 1 second samples in the past
  - all relative to current character root space



Real Player Motion

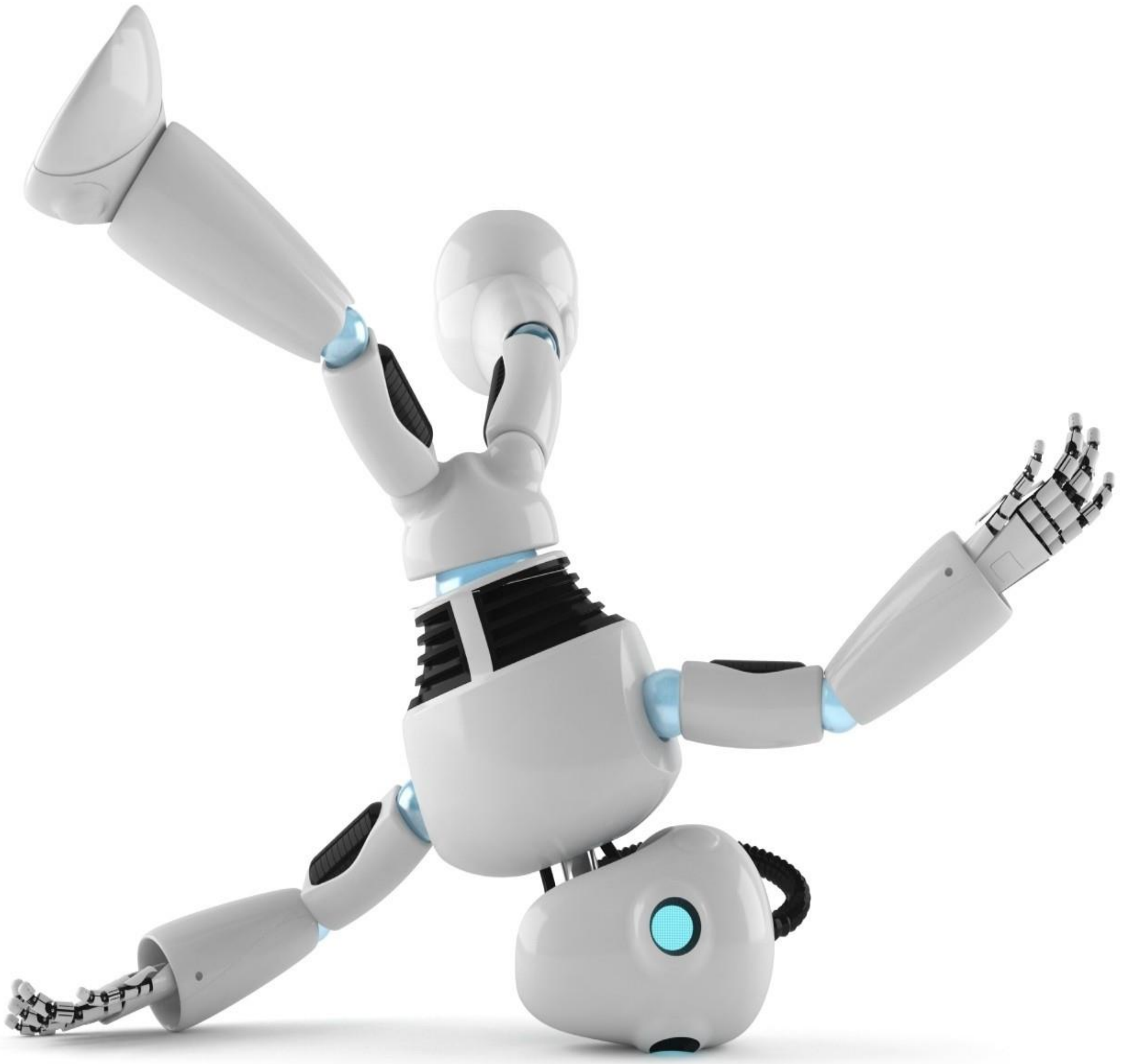
Motion Matching

- no more motion trees
- no more logic
- higher quality transitions
- more details



Motion Matching

- can’t create new motion
- original animation in memory
- can’t generalize



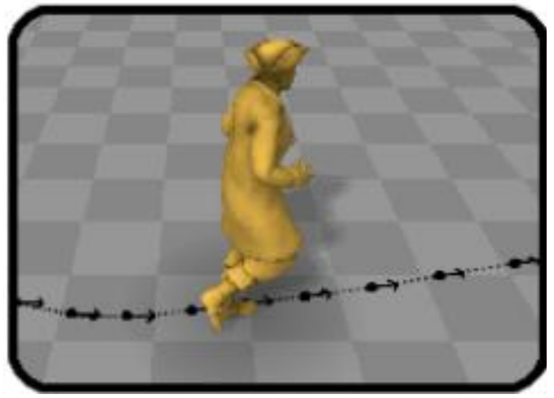
Motion Synthesis

- learn from examples
- generalize to new situations
- no animation data in memory
- that sounds like Machine Learning…

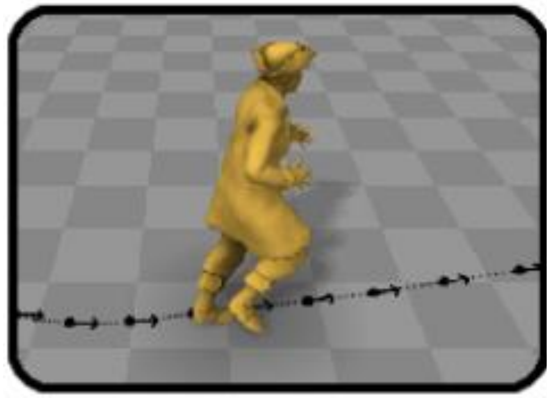
Neural Network

- simple: fully connected, feed forward
- **in:** pose + future/past trajectory
- **out:** next pose + future/past trajectory

Frame *i*

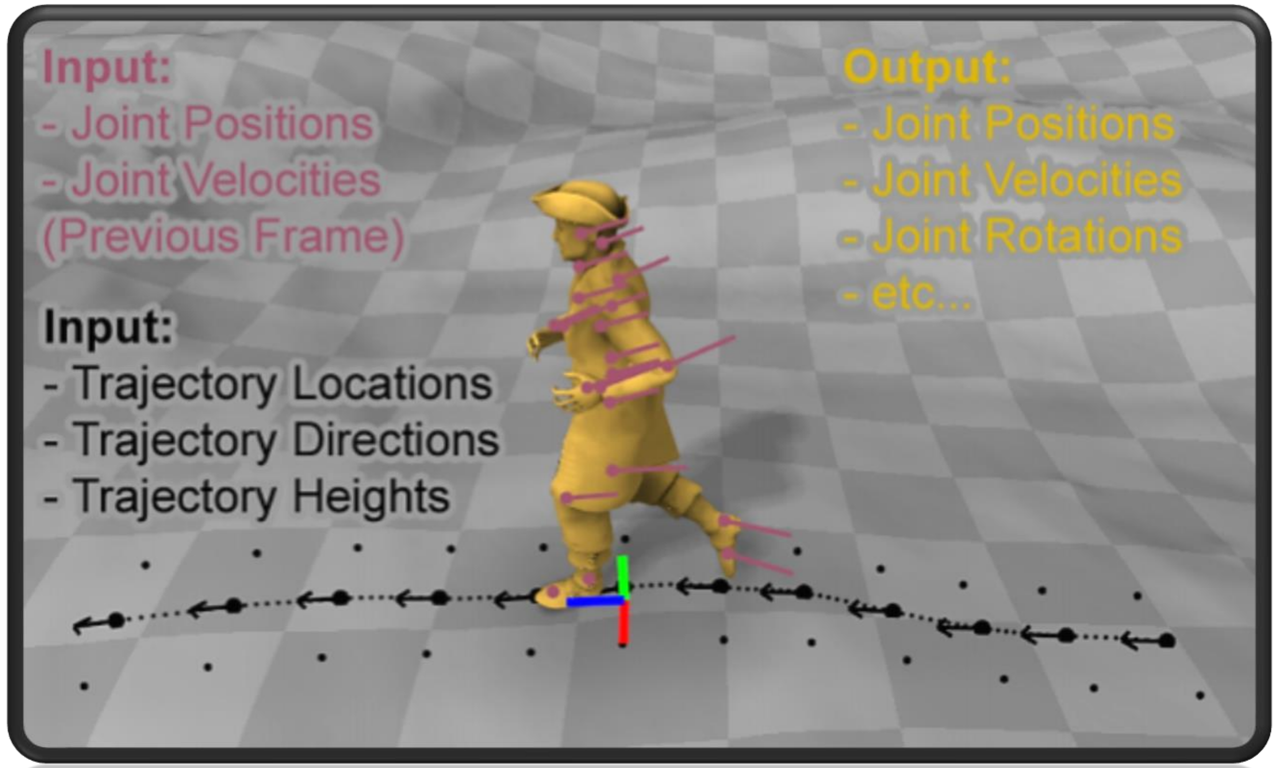


Frame *i+1*



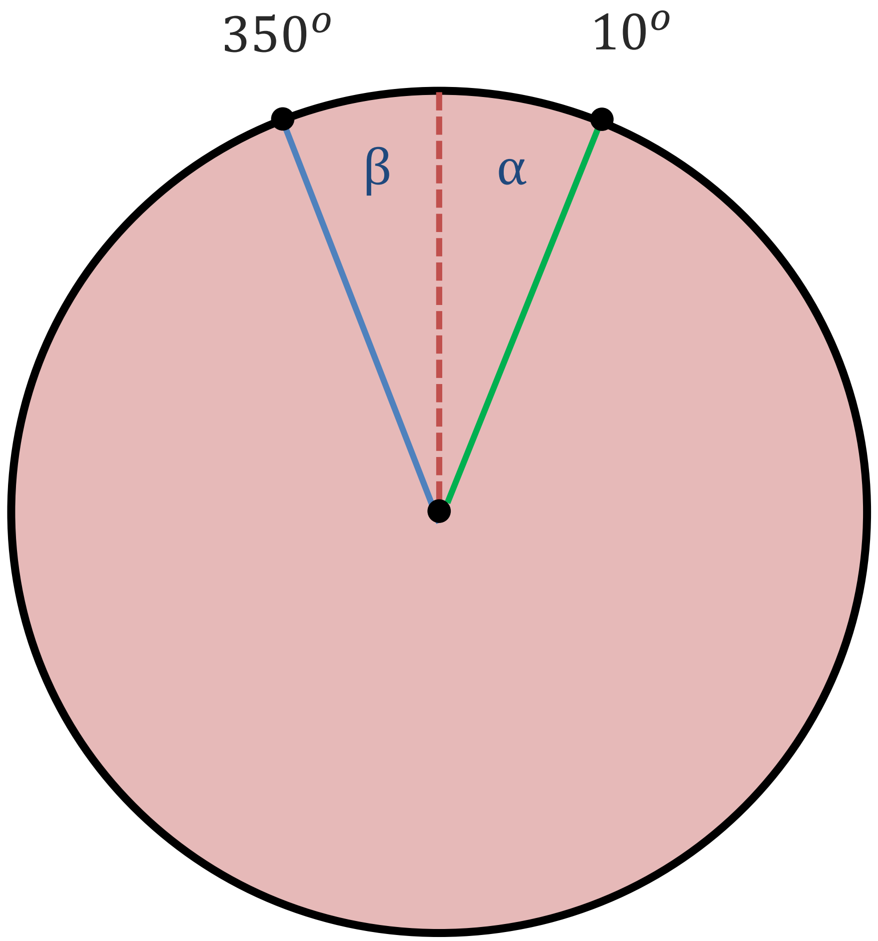
What's in the "pose"?

- Input:
  - **joint positions** – 3D points
  - **joints velocities** – 3D vectors
- Output:
  - **joint positions** – 3D points
  - **joints velocities** – 3D vectors
  - **joint orientations** – ?



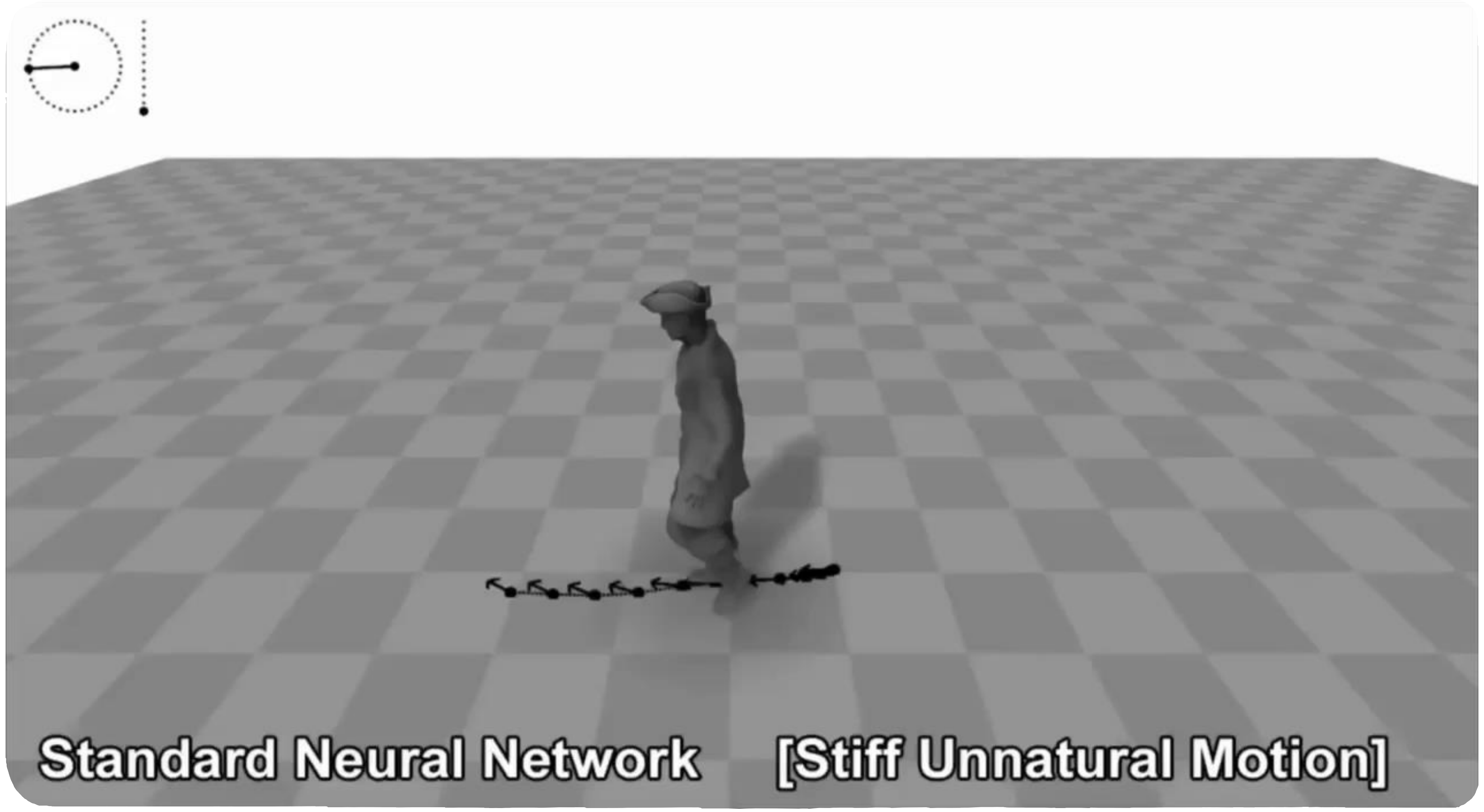
Animation Data encoding

- why not local joint transforms?
  - error propagates!
- rotations encoding
  - **quaternions** - complex algebra
  - **axis-angle** - euler angles are discontinuous
- alternatives?
  - vector difference
  - 3x3 matrices! (2 axes suffice)



$|\boldsymbol{\alpha} - \boldsymbol{\beta}| = 20^o \; or \; 340^o?$

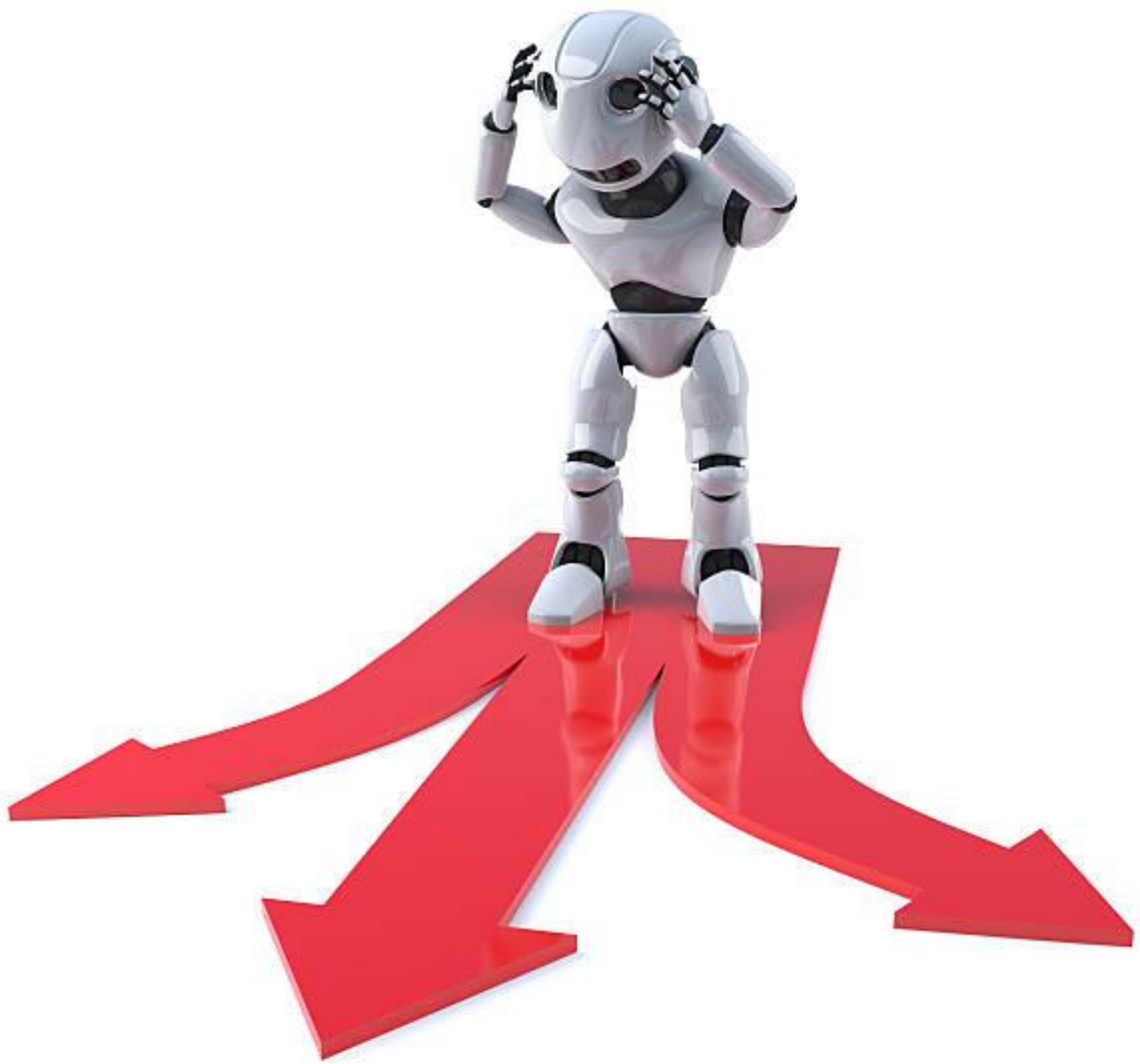
Vanilla Network



Phase-Functioned Neural Networks for Character Animation – Daniel Holden

What's wrong?

- input is not uniquely defined
- more than 1 output for each input
- MSE will average to minimize error
- result is floaty, frozen pose



Phase-Functioned Neural Networks

SIGGRAPH 2017

Phase-Functioned Neural Networks for Character Control

DANIEL HOLDEN, University of Edinburgh
TAKU KOMURA, University of Edinburgh
JUN SAITO, Method Studios

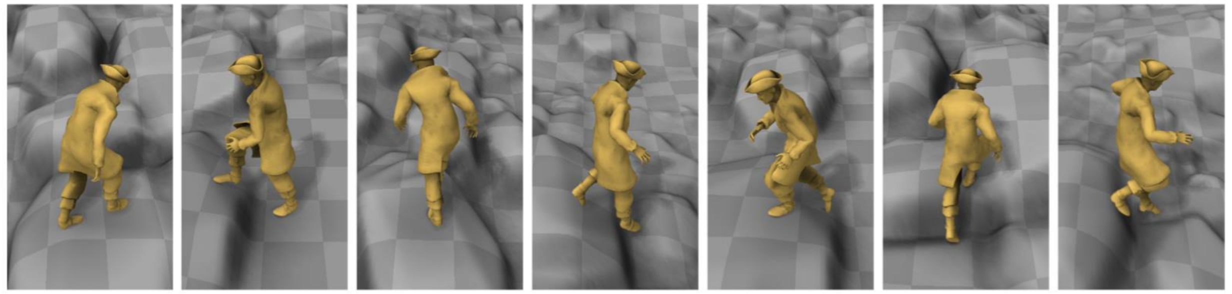


Fig. 1. A selection of results using our method of character control to traverse rough terrain: the character automatically produces appropriate and expressive locomotion according to the real-time user control and the geometry of the environment.

We present a real-time character control mechanism using a novel neural network architecture called a Phase-Functioned Neural Network. In this network structure, the weights are computed via a cyclic function which uses the phase as an input. Along with the phase, our system takes as input user controls, the previous state of the character, the geometry of the scene, and automatically produces high quality motions that achieve the desired user control. The entire network is trained in an end-to-end fashion on a large dataset composed of locomotion such as walking, running, jumping, and climbing movements fitted into virtual environments. Our system can therefore automatically produce motions where the character adapts to different geometric environments such as walking and running over rough terrain, climbing over large rocks, jumping over obstacles, and crouching under low ceilings. Our network architecture produces higher quality results than time-series autoregressive models such as LSTMs as it deals explicitly with the latent variable of motion relating to the phase. Once trained, our system is also extremely fast and compact, requiring only milliseconds of execution time and a few megabytes of memory, even when trained on gigabytes of motion data. Our work is most appropriate for controlling characters in interactive scenes such as computer games and virtual reality systems.

CCS Concepts: • **Computing methodologies → Motion capture**;

Additional Key Words and Phrases: neural networks, locomotion, human motion, character animation, character control, deep learning

ACM Reference format:
Daniel Holden, Taku Komura, and Jun Saito . 2017. Phase-Functioned Neural Networks for Character Control. *ACM Trans. Graph.* 36, 4, Article 42 (July 2017), 13 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073663

This work is supported by Marza Animation Planet.

1 INTRODUCTION

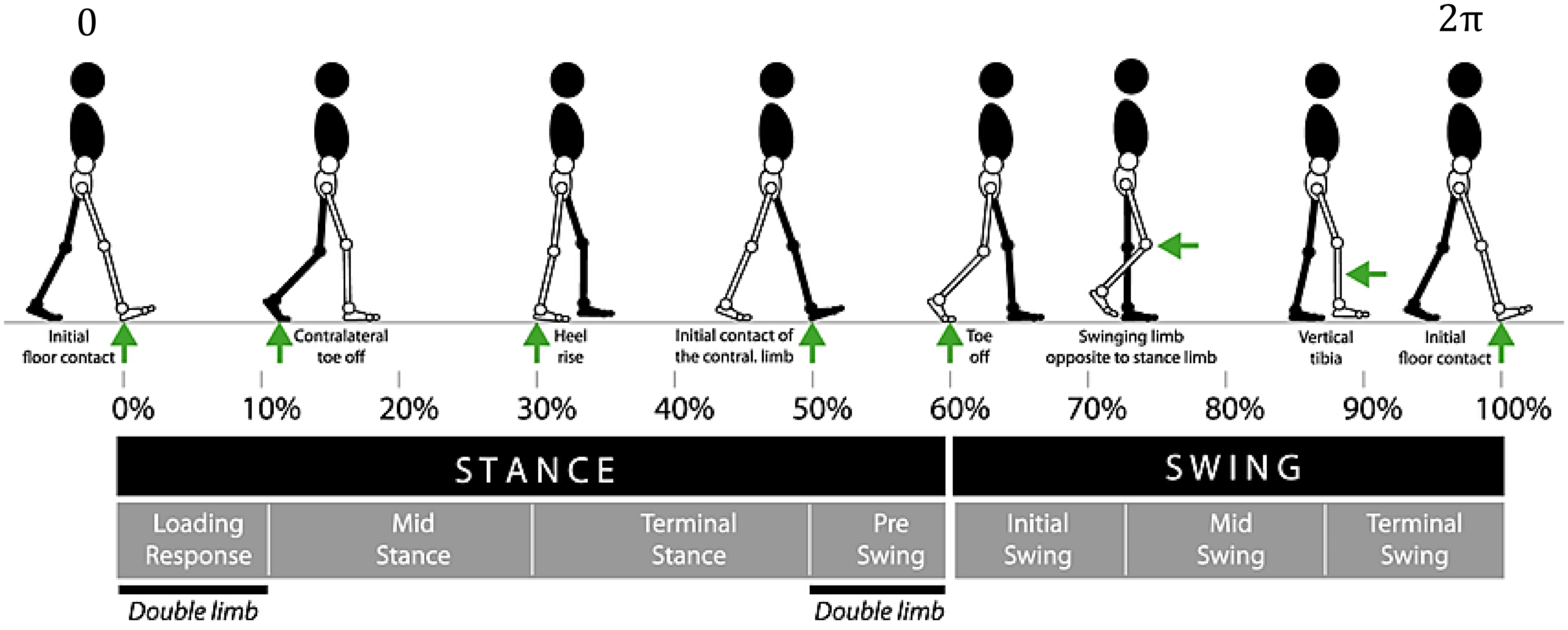
Producing real-time data-driven controllers for virtual characters has been a challenging task even with the large amounts of readily available high quality motion capture data. Partially this is because character controllers have many difficult requirements which must be satisfied for them to be useful - they must be able to learn from large amounts of data, they must not require much manual pre-processing of data, and they must be extremely fast to execute at runtime with low memory requirements. While a lot of progress has been made in this field almost all existing approaches struggle with one or more of these requirements which has slowed their general adoption.

The problem can be even more challenging when the environment is composed of uneven terrain and large obstacles which require the character to perform various stepping, climbing, jumping, or avoidance motions in order to follow the instruction of the user. In this scenario a framework which can learn from a very large amount of high dimensional motion data is required since there are a large combination of different motion trajectories and corresponding geometries which can exist.

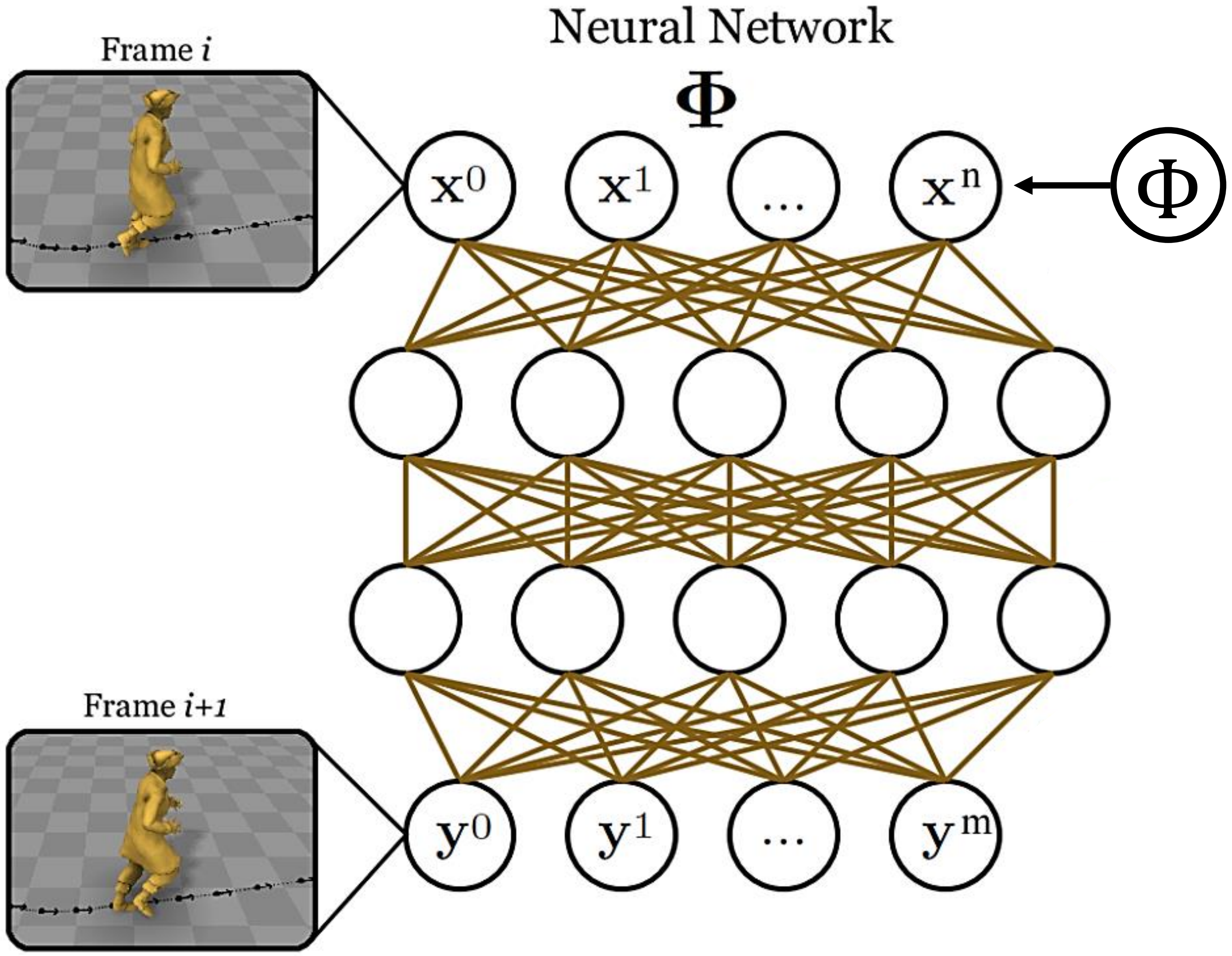
Recent developments in deep learning and neural networks have shown some promise in potentially resolving these issues. Neural networks are capable of learning from very large, high dimensional datasets and once trained have a low memory footprint and fast execution time. The question now remains of exactly how neural networks are best applied to motion data in a way that can produce high quality output in real time with minimal data processing.

Previously some success has been achieved using convolutional models such as CNNs [Holden et al. 2016, 2015], autoregressive models such as RBMs [Taylor and Hinton 2009], and RNNs [Fragkiadaki et al. 2015]. CNN models perform a temporally local transformation on each layer, progressively transforming the input signal until the desired output signal is produced. This structure naturally lends itself to an offline, parallel style setup where the whole input is

Phase for locomotion



Let's add phase!



- a bit better but...
- phase is one drop in an ocean
- no guarantees the network will use it
- dropout doesn't help...

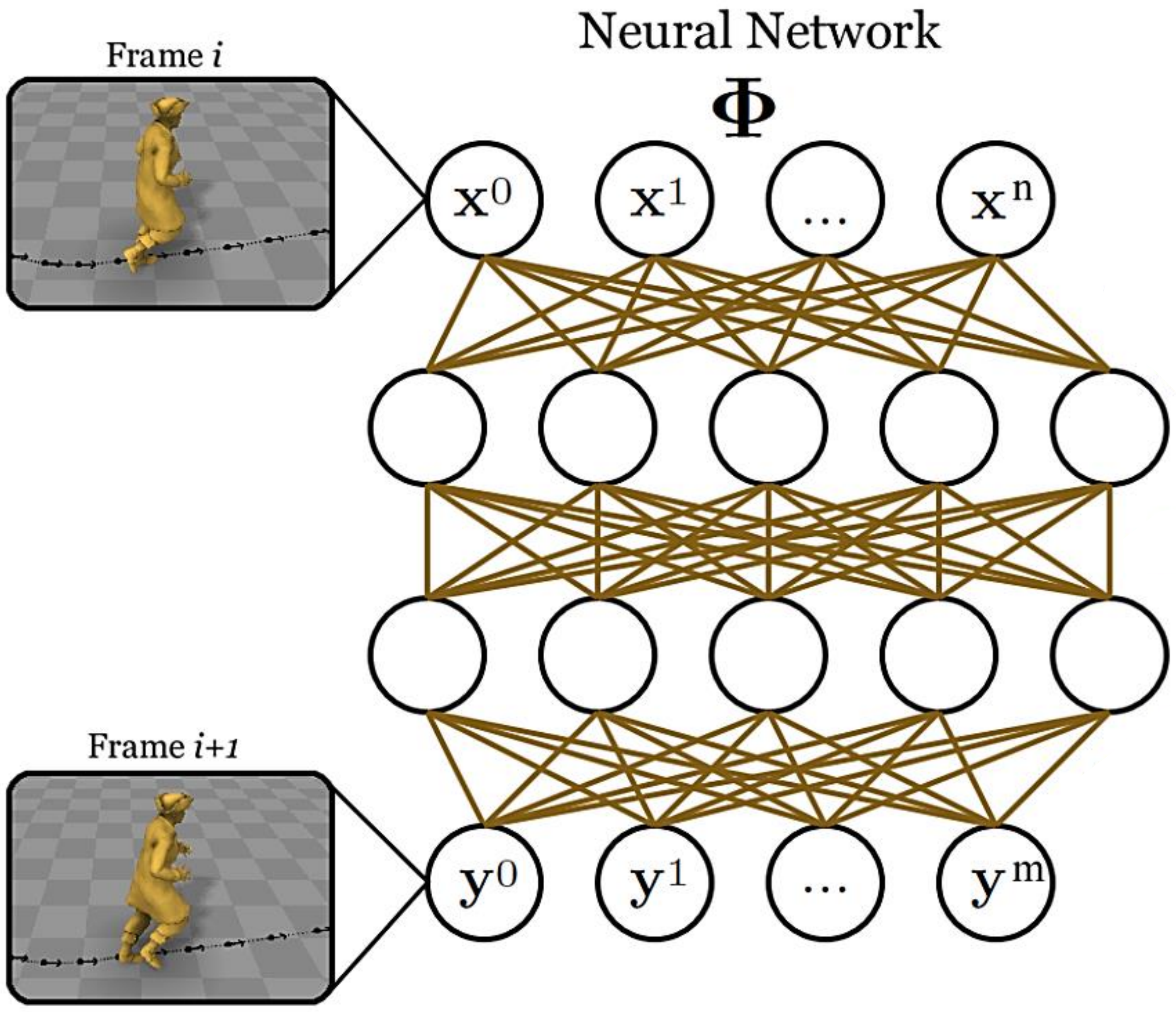
Phase-binning

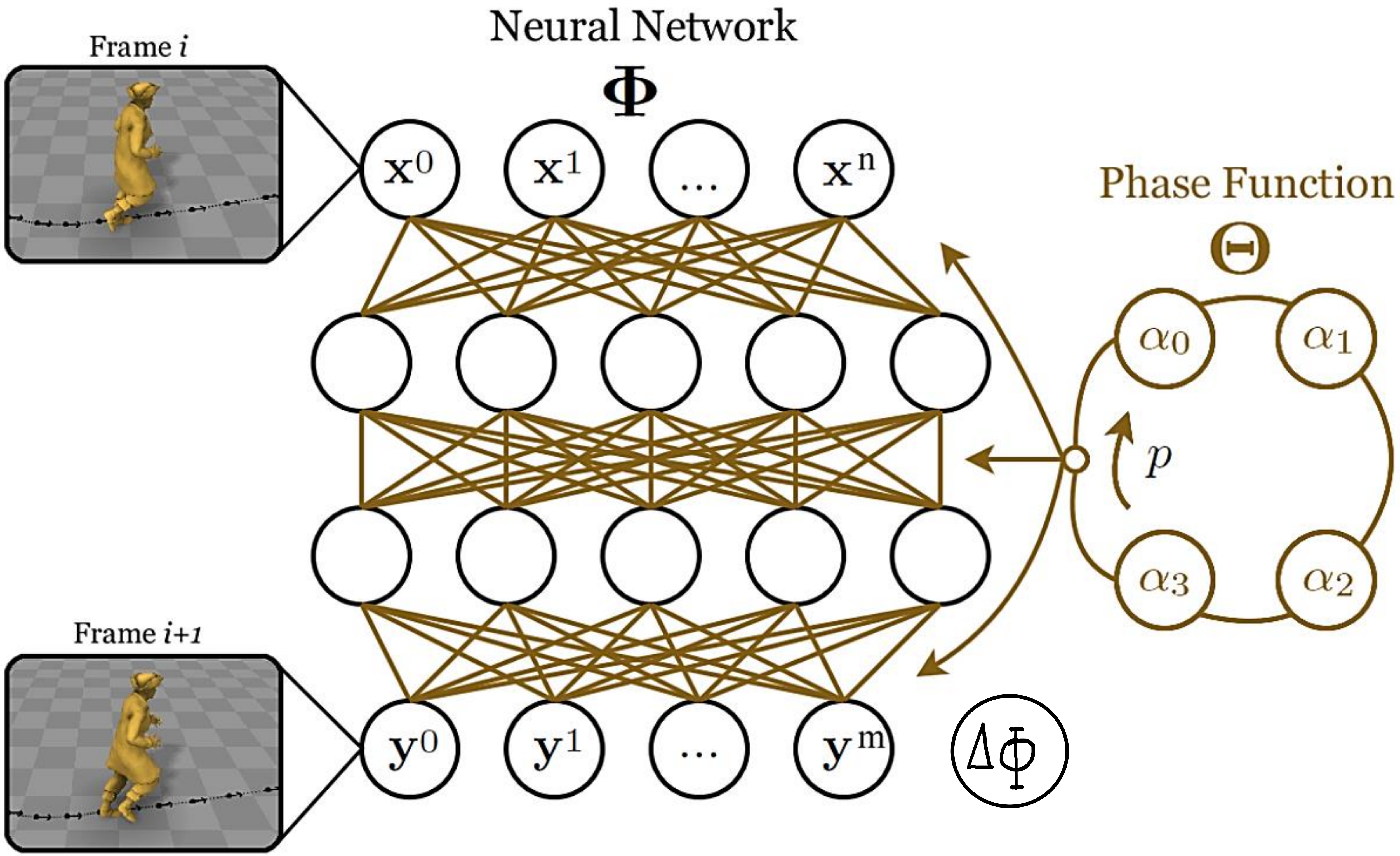
- divide phase in **n** sections
- train **n** separate networks
- switch NN according to phase

\+ works better than vanilla!

– can stutter when switching NN

x **n**





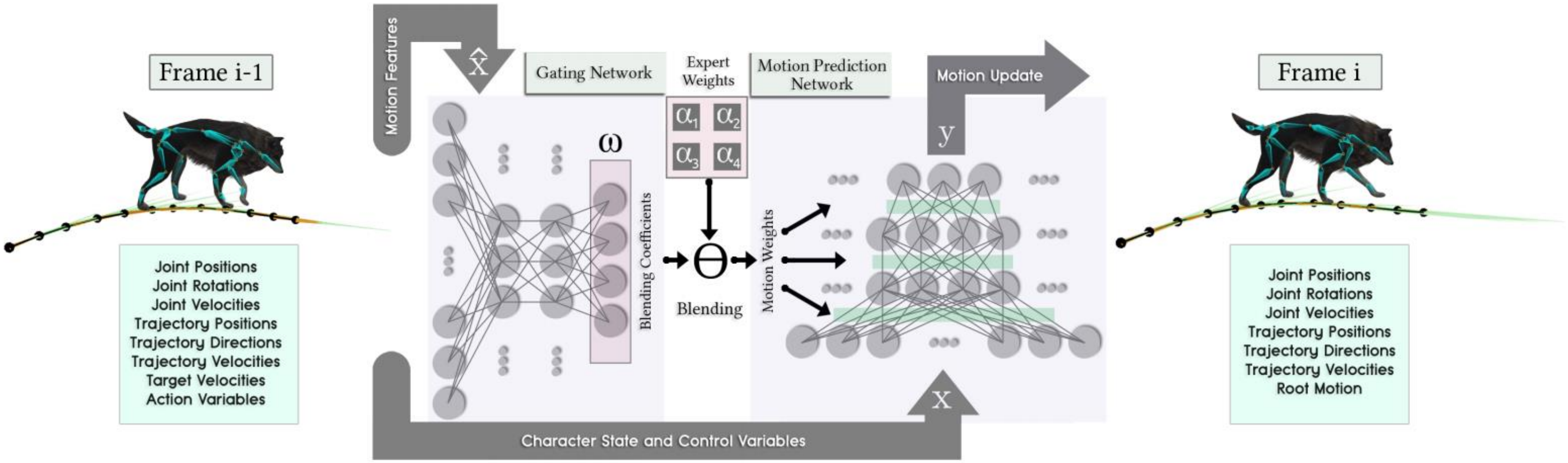
Frame i
Neural Network
$\mathbf{\Phi}$
$\mathbf{x}^0$
$\mathbf{x}^1$
...
$\mathbf{x}^n$
Phase Function
$\mathbf{\Theta}$
α_0
α_1
p
α_3
α_2
Frame $i+1$
$\mathbf{y}^0$
$\mathbf{y}^1$
...
$\mathbf{y}^m$
$\Delta\Phi$

Phase-Functioned Neural Networks for Character Control

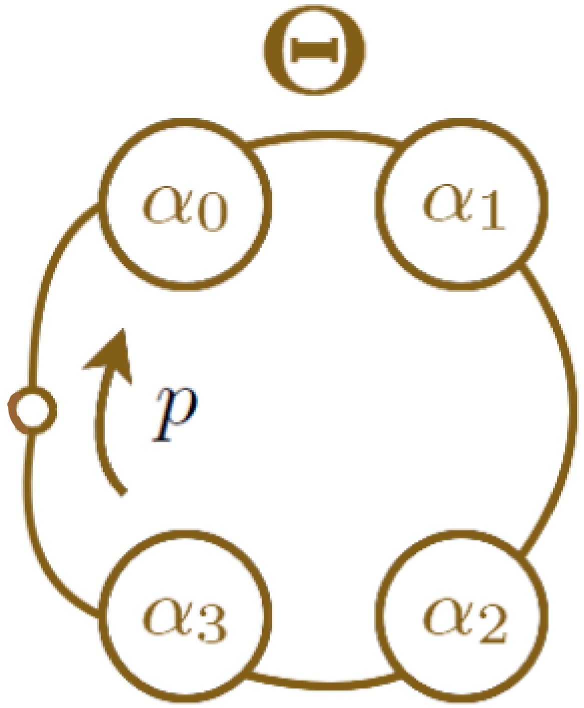
Daniel Holden [1], Taku Komura [1], Jun Saito [2]

[1] University of Edinburgh, [2] Method Studios

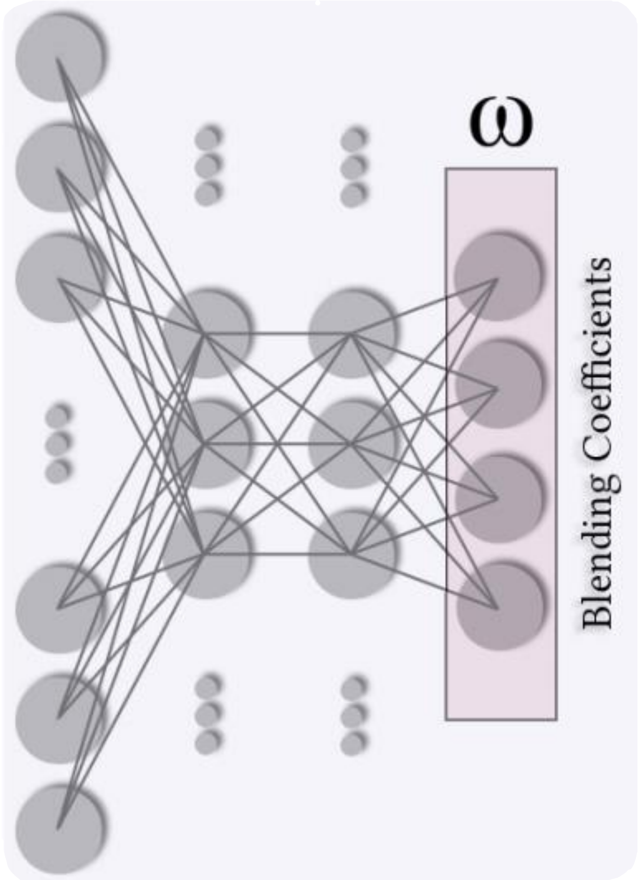

Mode-Adaptive Neural Networks for
Quadruped Motion Control
- SIGGRAPH 2018: Vancouver, Canada -
He Zhang*
Sebastian Starke*
Taku Komura
Jun Saito
*Joint First Authors
THE UNIVERSITY of EDINBURGH
Adobe


Frame i-1
Joint Positions
Joint Rotations
Joint Velocities
Trajectory Positions
Trajectory Directions
Trajectory Velocities
Target Velocities
Action Variables
Motion Features
x̂
Gating Network
ω
Blending Coefficients
Expert Weights
α1 α2 α3 α4
Θ
Blending
Motion Prediction Network
Motion Weights
Motion Update
y
x
Character State and Control Variables
Frame i
Joint Positions
Joint Rotations
Joint Velocities
Trajectory Positions
Trajectory Directions
Trajectory Velocities
Root Motion

Different approach



VS



Cubic Interpolator

Gating Network

In theory...

Cubic Spline:

- data needs phase annotation
- interpolator is cyclic by design
- function of phase alone

VS

Gating Network:

- no phase annotation required
- arbitrary function
- more info as input

MANN with humans?

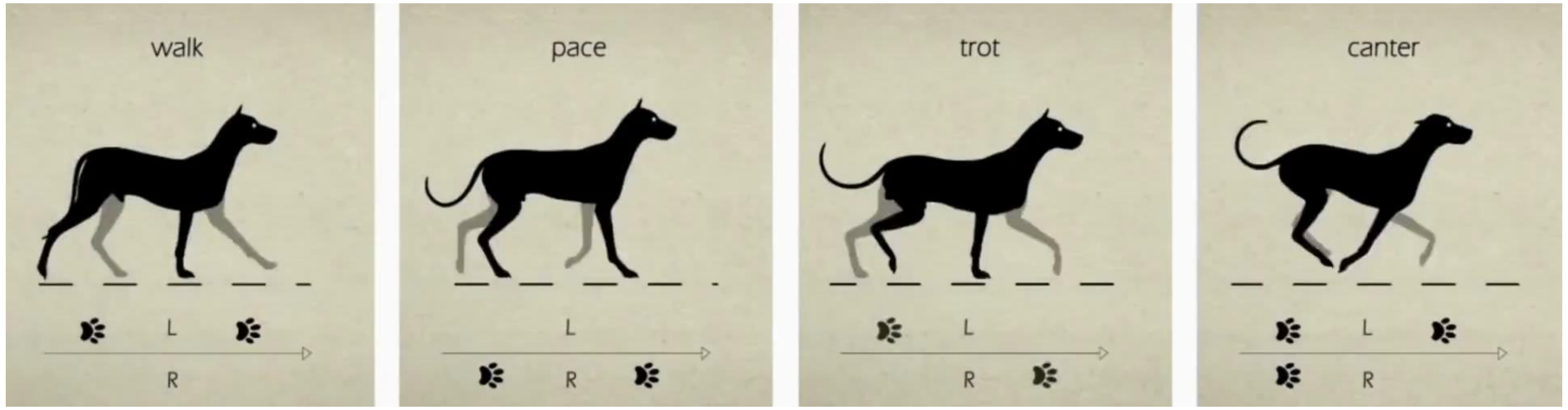
- phase annotation is annoying…
- …and prevents skipping/hopping
- MANN seems more generic…
- …and not limited by phase

…in practice

- full feature vector as Gating input
  - floaty, frozen pose
- selecting few features helps…
  - feet position, desired root velocity
- …but doesn't solve the issues

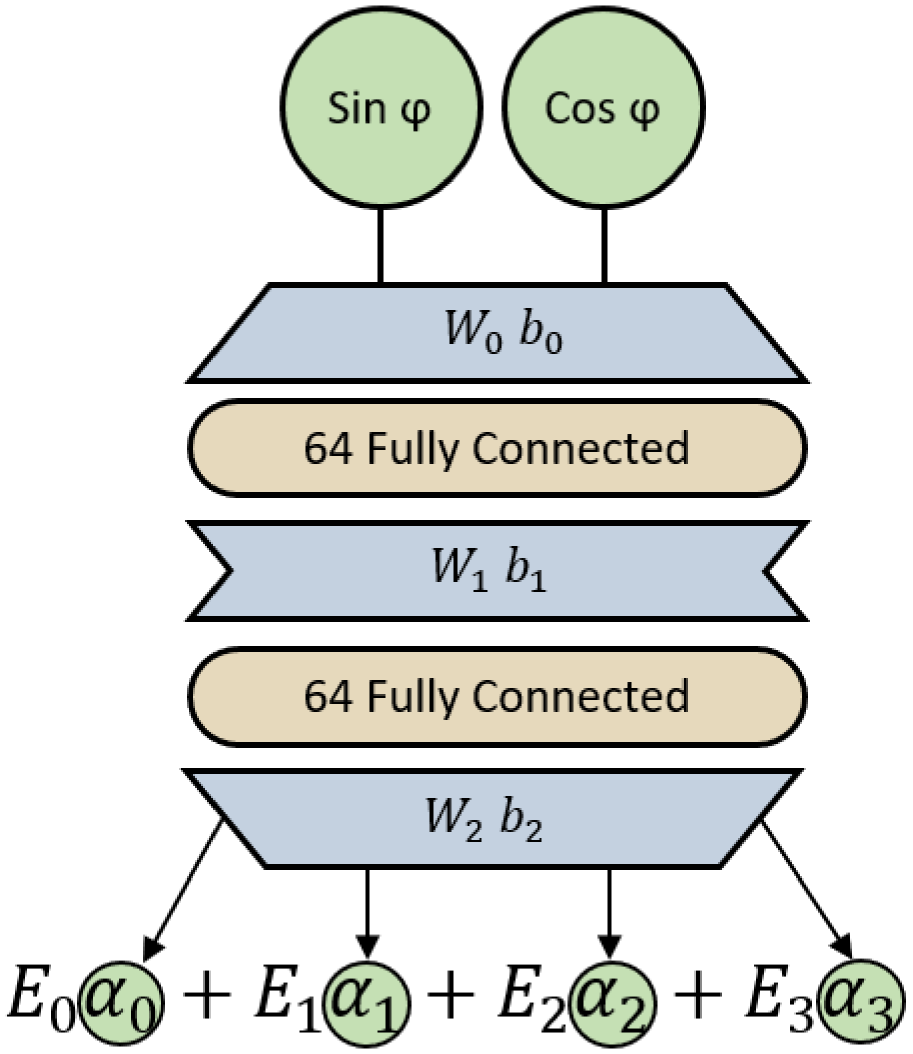
Dogs don't need phase?

- well defined modes +
- complex foot patterns =
- less ambiguity?



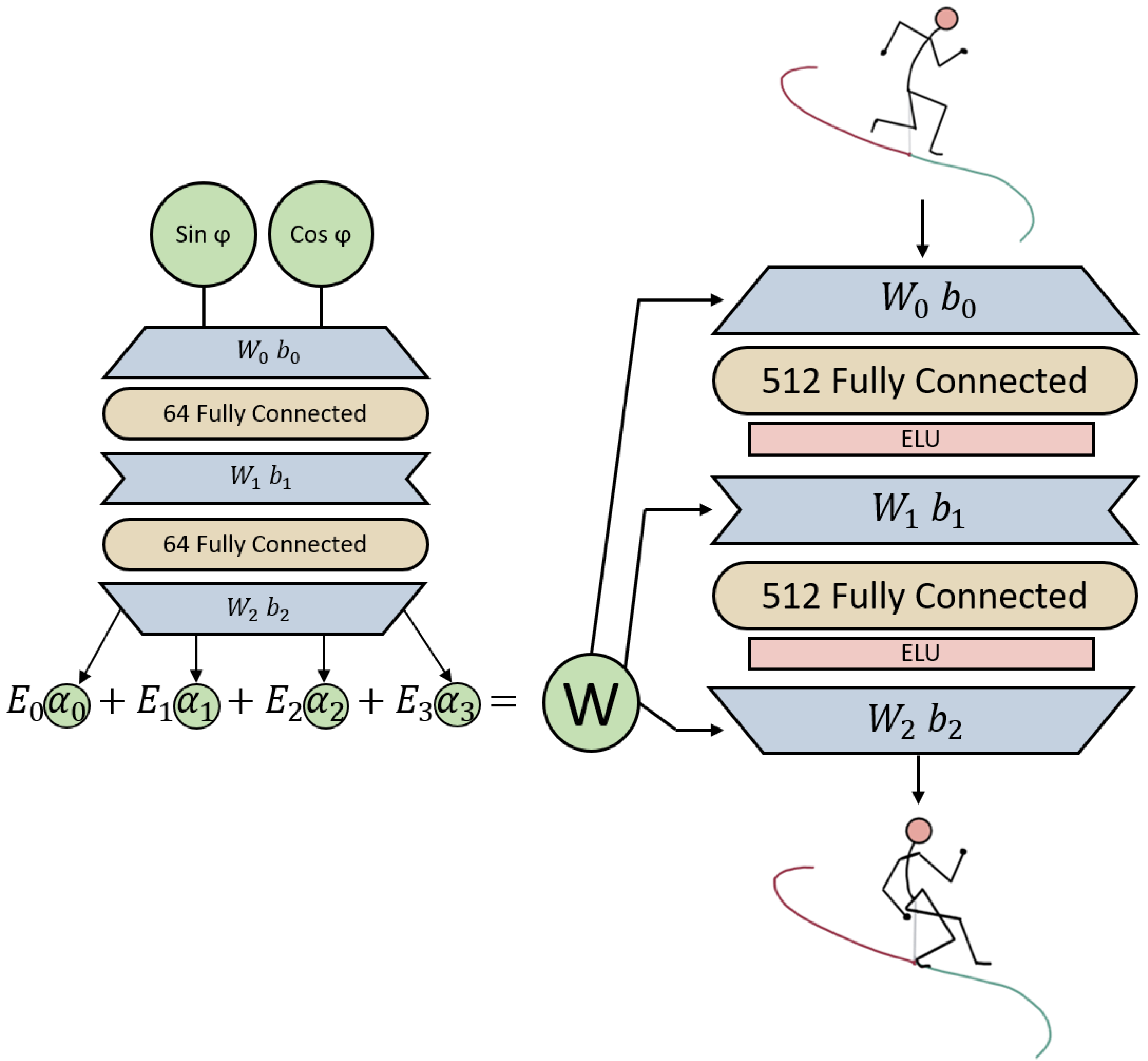
MANN with phase?

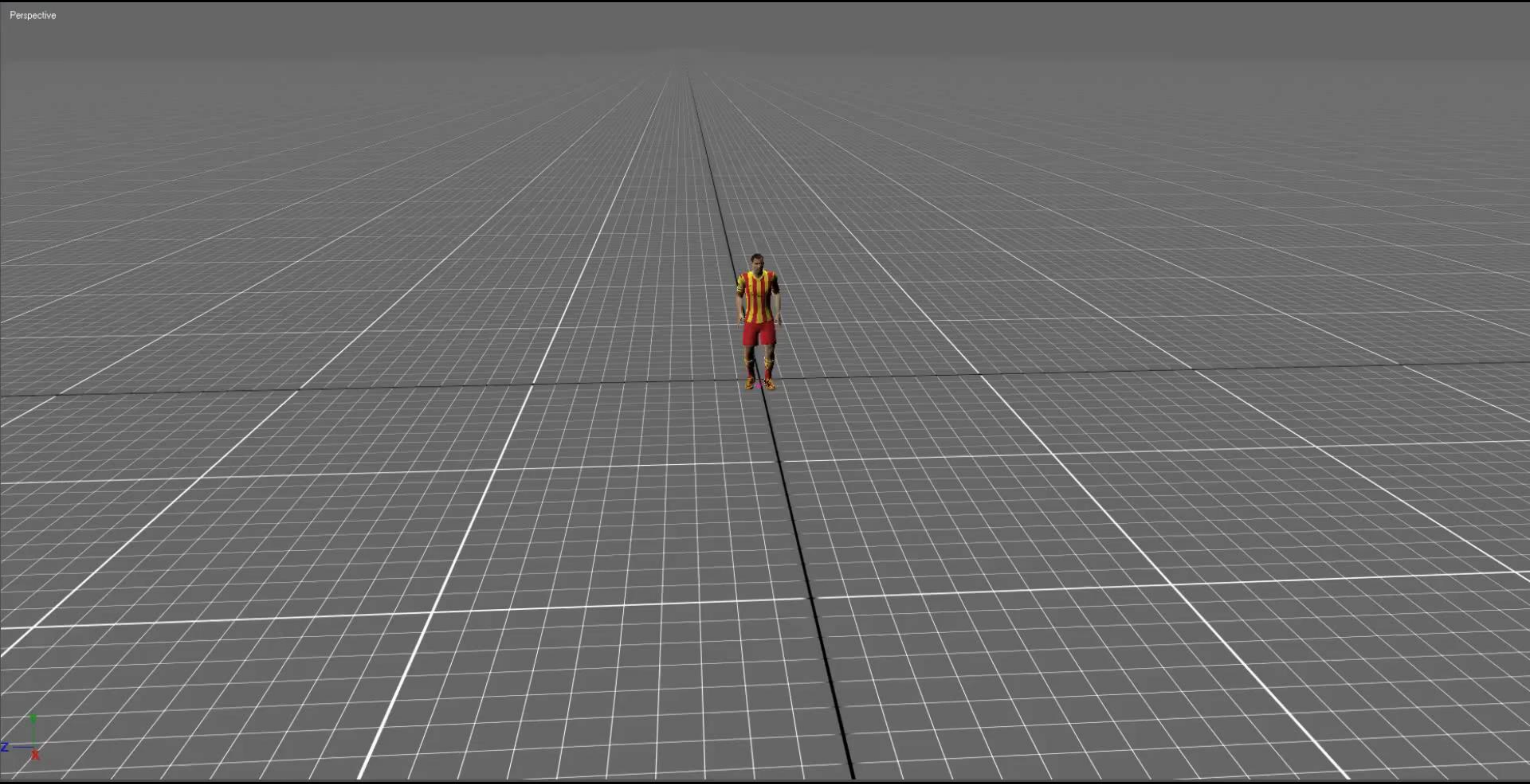
- *"...there are many potential choices for [phase function]...could be another neural network..."*
- use gating network as phase function
- phase is discontinuous: [*sin(ϕ), cos(ϕ)*]



Architecture

- dropout every layer (0.3)
- loss: $\boldsymbol{MSE}(\boldsymbol{y}, \boldsymbol{\hat{y}})$
- training time: ~6 hours
  - GTX 1080
  - 20 mins of MoCap




Perspective

Performance

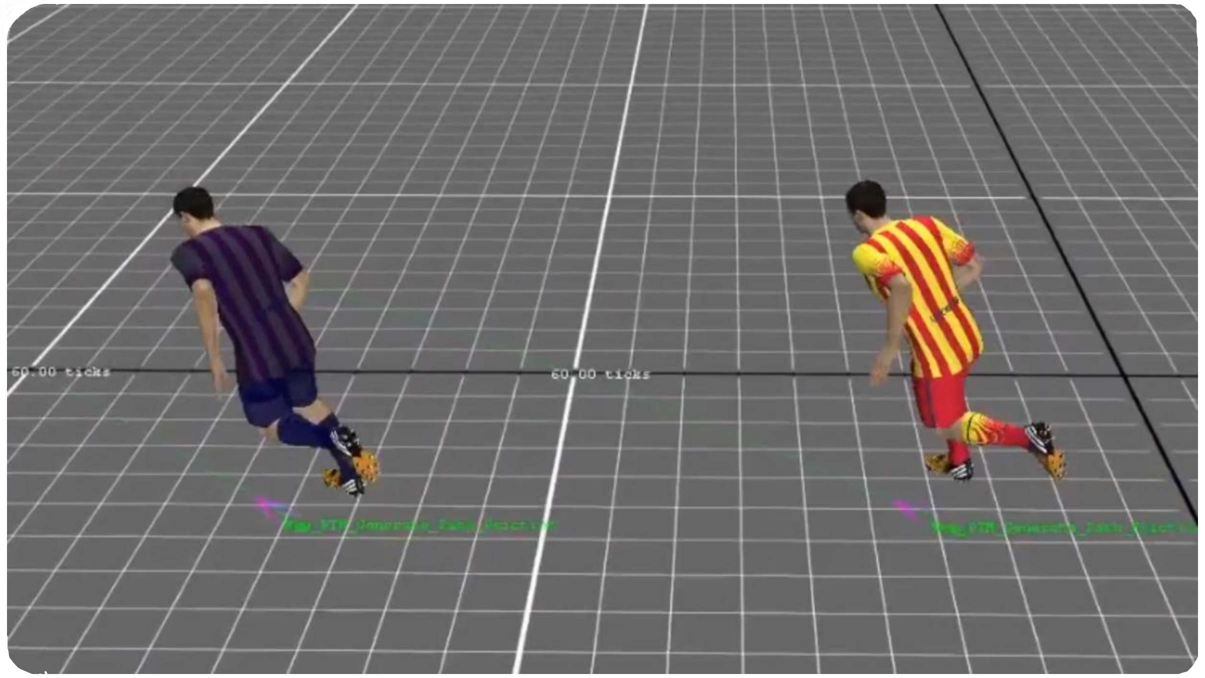
- real-time (CPU):
  - 3x Motion Matching
  - not optimized yet
- no training data in memory
  - only memory for network weights
  - *(from PFNN)* ~10 MB with 4 experts

Motion Quality

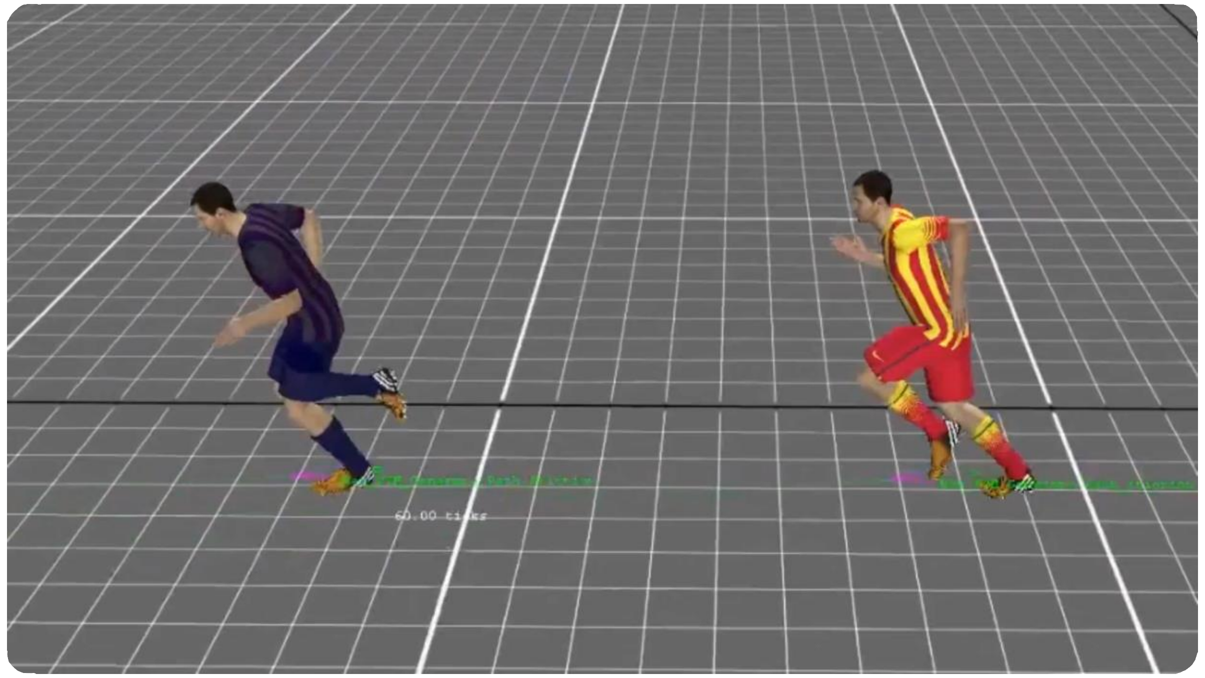
- comparable to Motion Matching
- **Motion Matching:** more details
- **NN:** generalizes better

Motion Quality

- Motion Matching shows more details:
  - leaning in acceleration
  - more precise foot steps



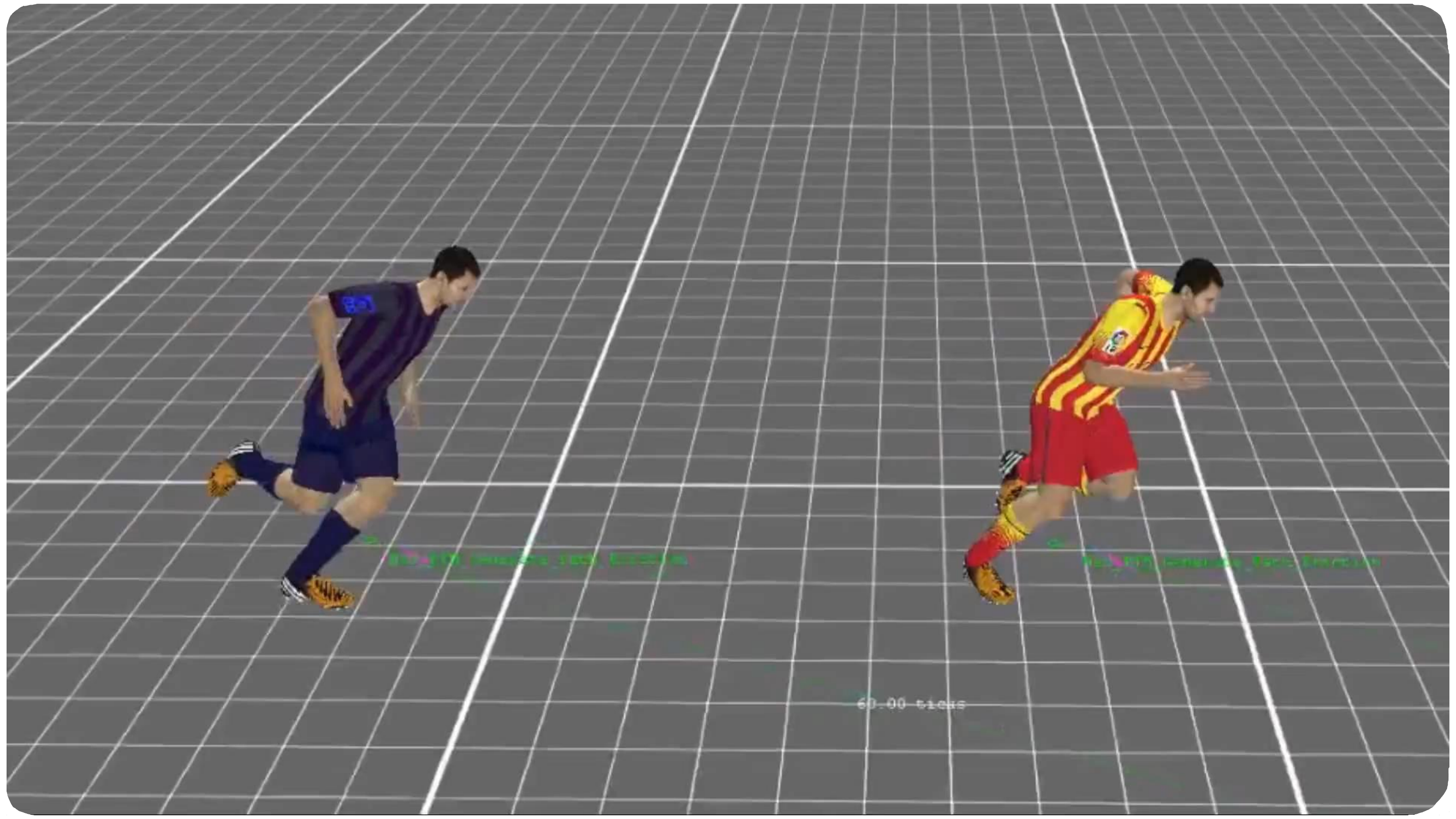
Neural Network Motion Matching



Neural Network Motion Matching

Better generalization

- NN generalizes to new situations
- like unrealistic trajectories…
  - …that games often need!
- smoother motion
- more sliding – feet IK helps



Neural Network Motion Matching

What’s next?

- smaller, specialized networks
- how do we combine them:
    - heuristics
    - gating hierarchies

Road to production

Challenges:

- integration in pipelines
- usability and artistic control
- fast iteration
- runtime efficiency

New approach?

- games are getting bigger
- brute force takes us only that far
- we will have to let go a bit

Research in the industry

- applied research please!
- it's on us to make this usable
- (most) academics will not care about it

Thank you!

Fabio Zinno – fzinno@ea.com