The University of Edinburgh

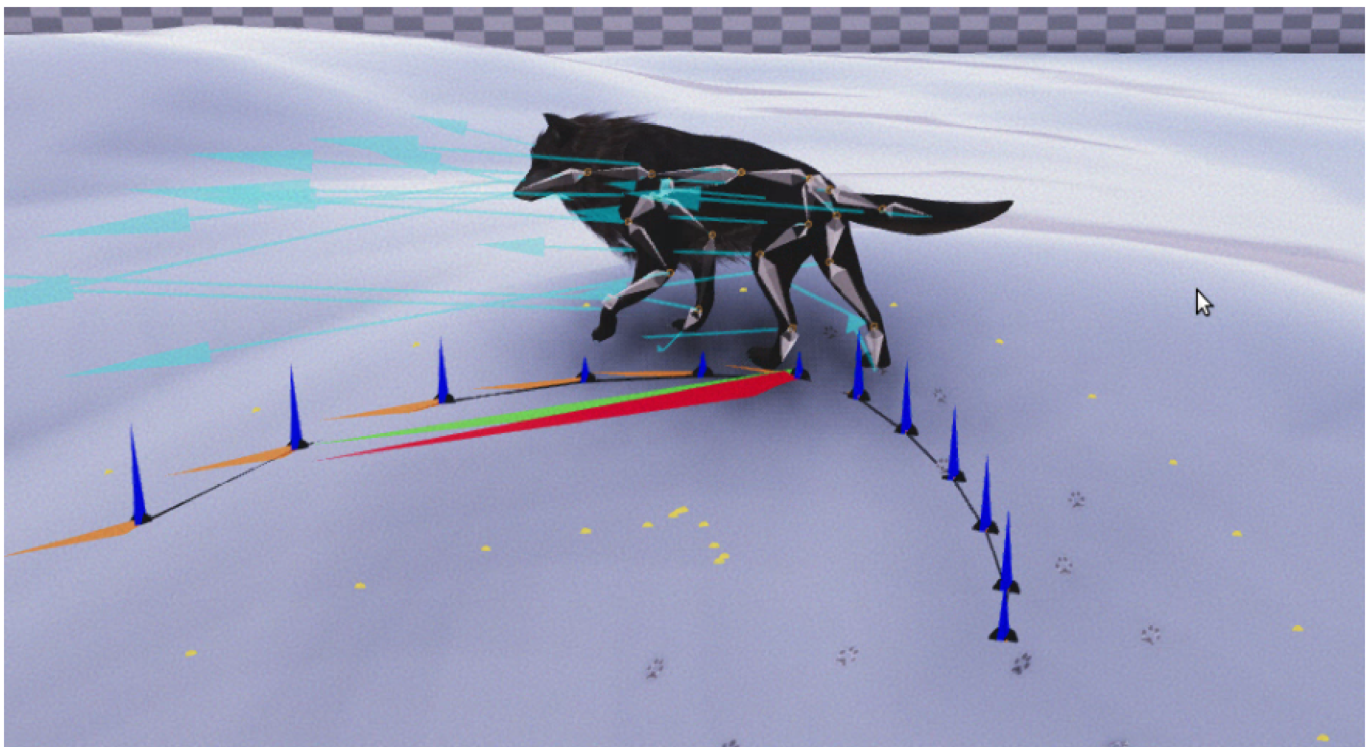
Phase-Functioned Neural Networks for Motion Learning

TAMS
University of Hamburg
03.01.2018

Sebastian Starke
University of Edinburgh
School of Informatics
Institue of Perception, Action and Behaviour
Sebastian.Starke@ed.ac.uk

Introduction

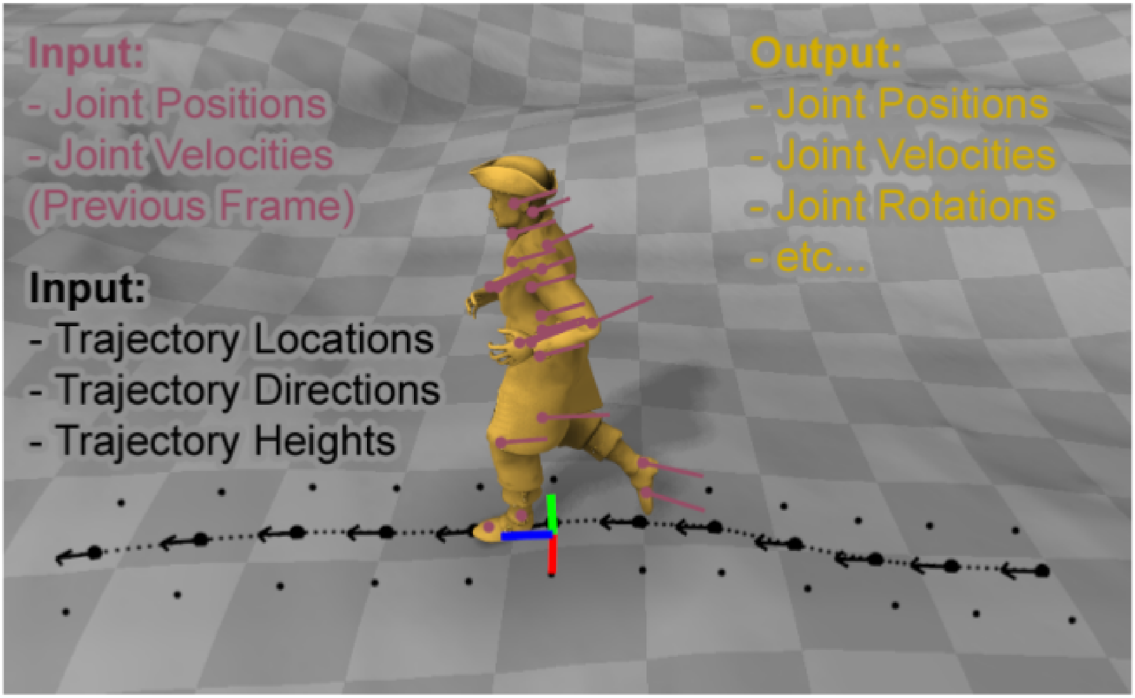
- Learning motion on articulated bodies is a task which aims to robustly generate or reproduce valid and efficient movements.



Introduction

- Typical applications in robotics and animation include locomotion, manipulation, interaction, …

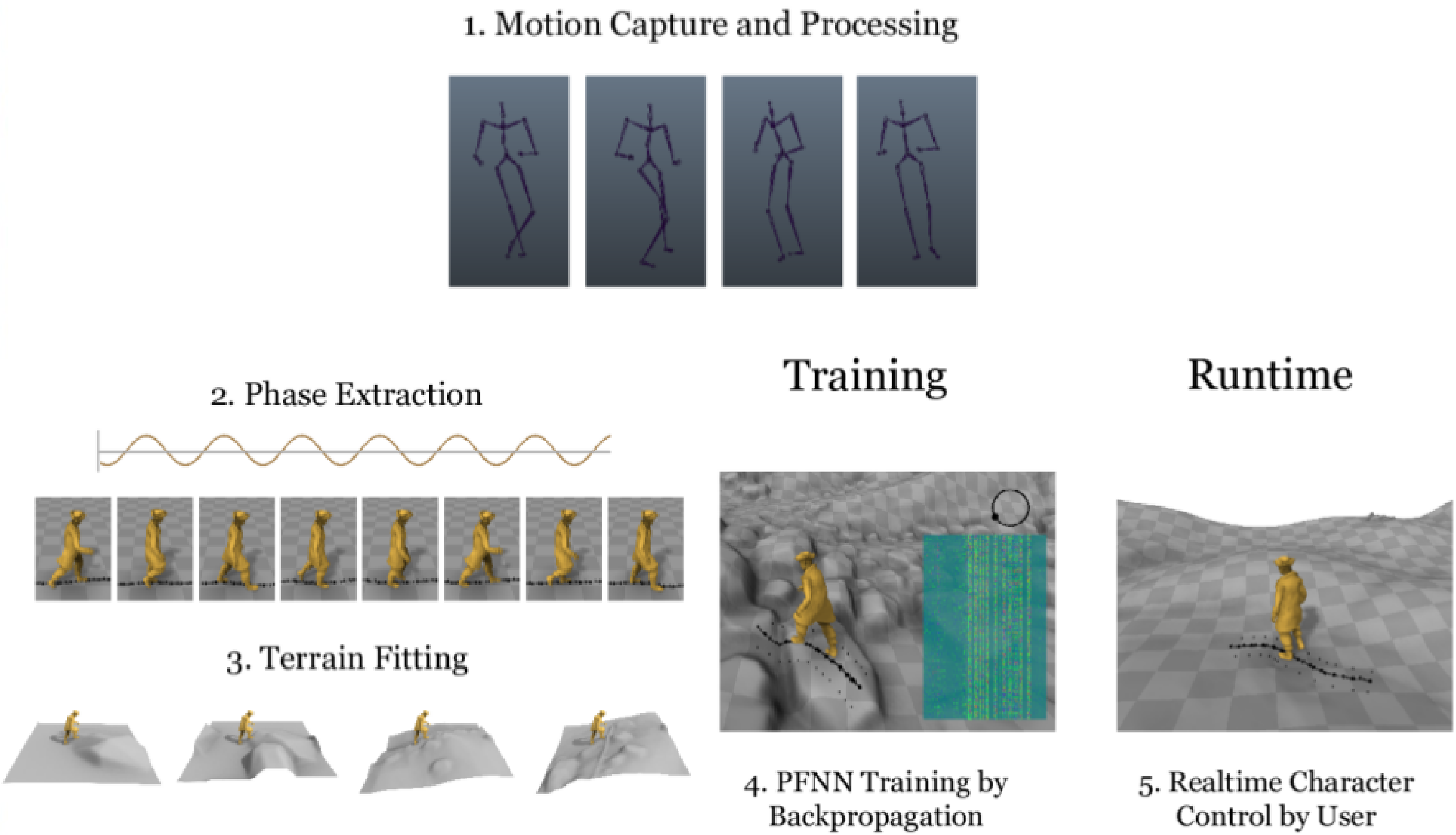
→ Geometry of the body (serial chain, biped, quadruped)
→ Adapting to environments
→ Control signals produce motion



Related Work

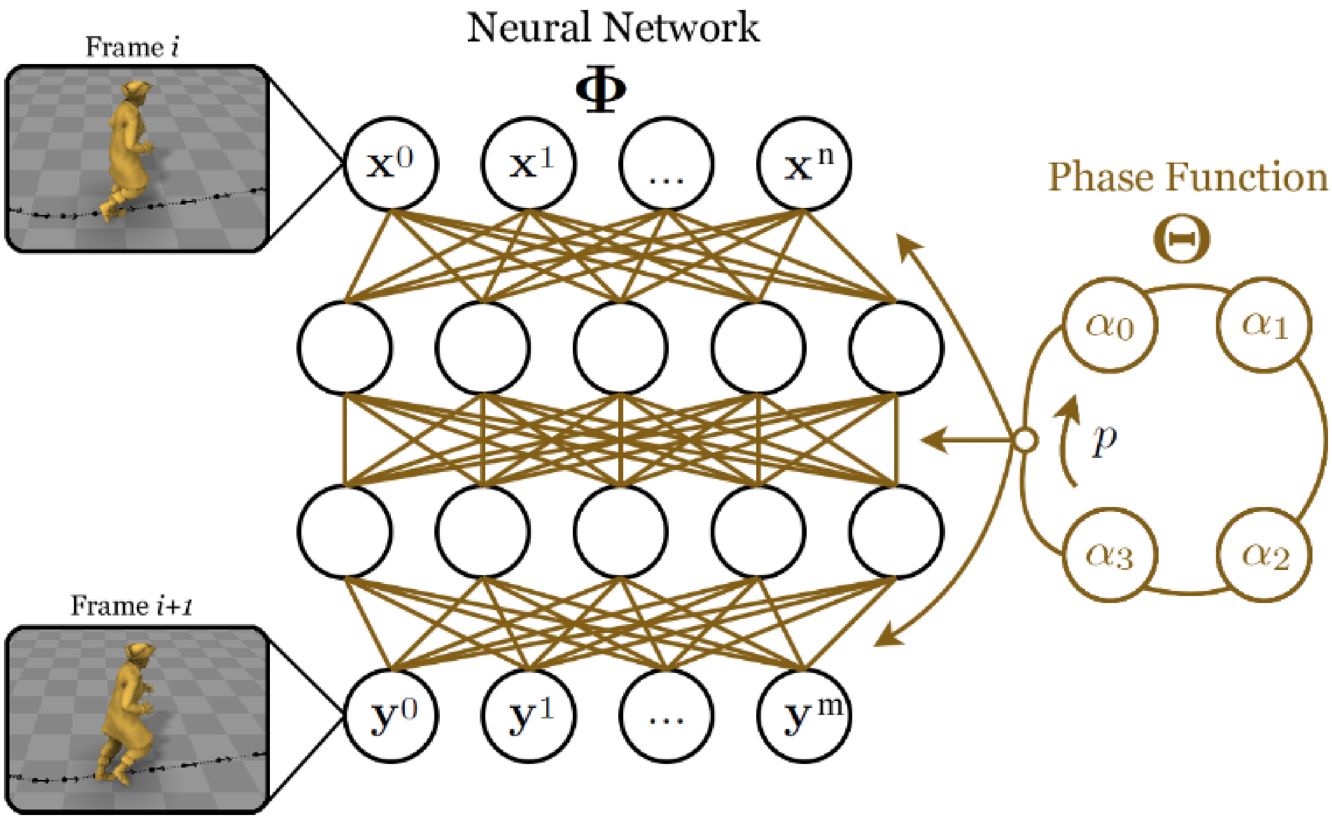
- Data-driven motion synthesis based on PCA (Howe et. al. 1999, Safonova et. al. 2004) by projecting motion on a lower-dimensional manifold (global vs. local PCA)
- Kernel-based approaches overcome limitations of linear-based methods using RBF (Radial Basis Functions) or GP (Gaussian Processes)
  → Motion Blending (Mukai 2011, Grochow et. al. 2004)
  → Planning Movements (Levine et. al. 2012)
- Auto-regressive models such as conditional RBM (Taylor et. al. 2009) and Encoder-Recurrent-Decoder using LSTM (Fragkiadaki et. al., 2015)
  → More scalable, but tend to drift from the original motion
- Deep Reinforcement Learning in the control space of physically-based animation to handle high-dimensional state spaces (Peng et. al. 2016), but the system is only tested in 2D environments.

Approach


1. Motion Capture and Processing
2. Phase Extraction
3. Terrain Fitting
Training
Runtime
4. PFNN Training by Backpropagation
5. Realtime Character Control by User

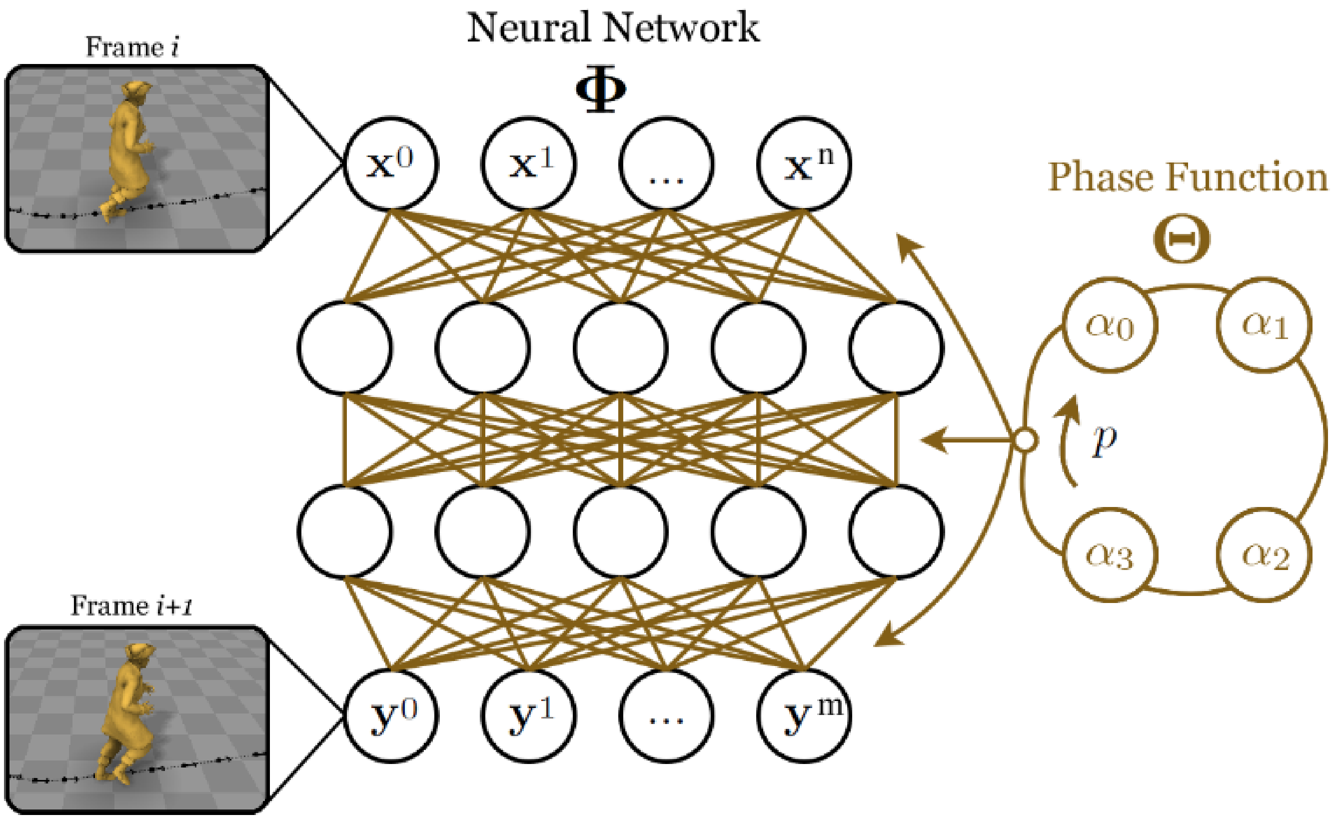
Approach

- Phase-Functioned Neural Network to learn predictions from one state i to i+1 while handling different styles of motion.



Approach

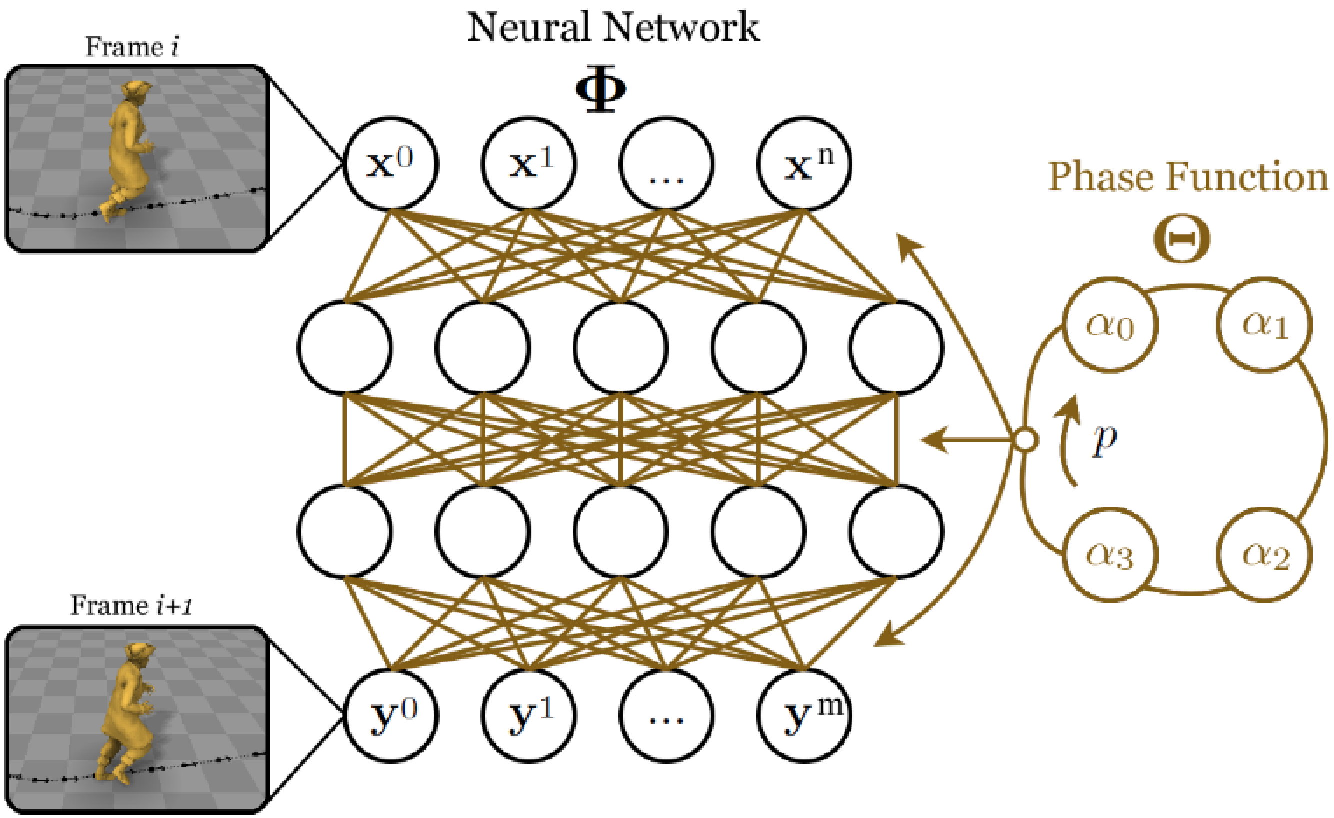
- Intuitvely, the phase is used to learn a function of weights rather than a single weight distribution in order to prevent false motion interpolations.



Approach

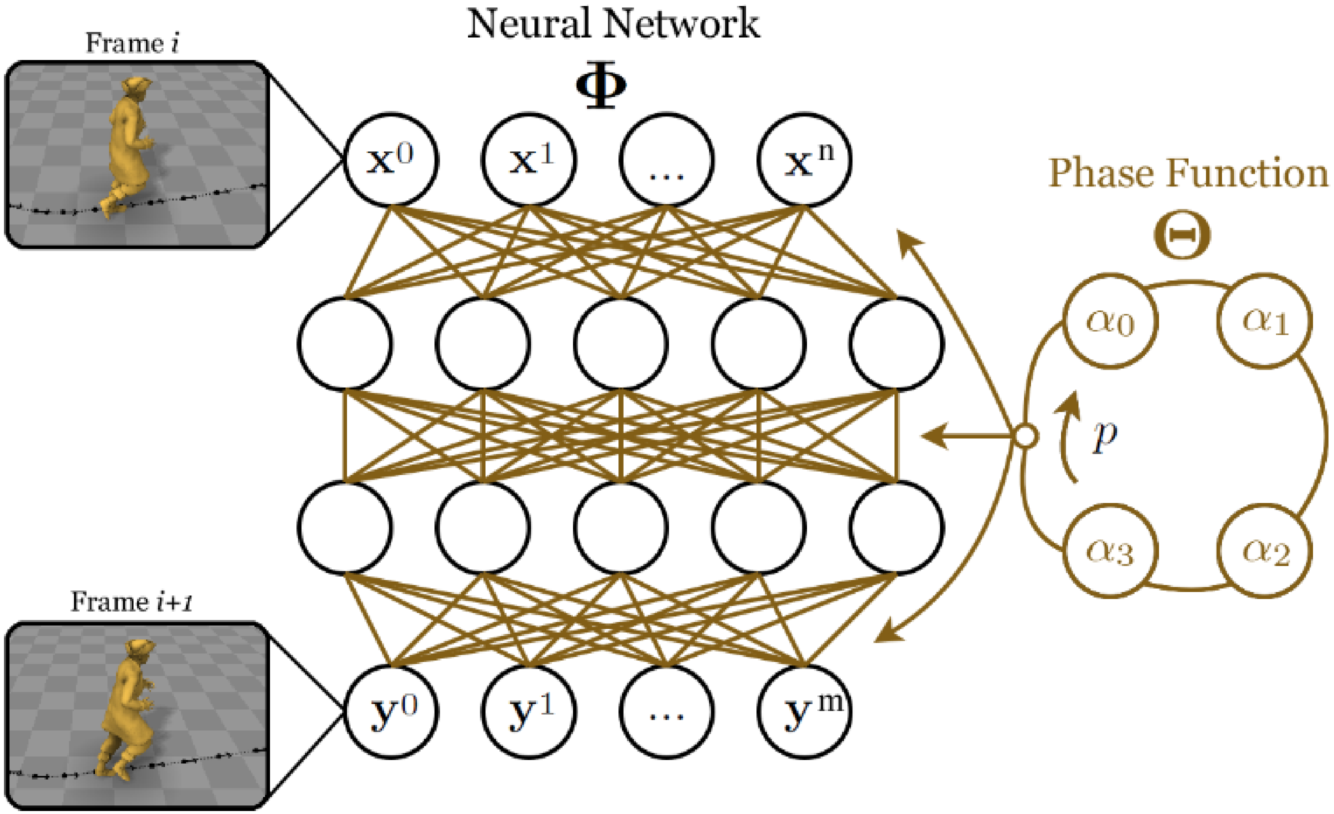
$$\Phi(\mathbf{x};\boldsymbol{\alpha}) = \mathbf{W}_2\ \mathrm{ELU}(\ \mathbf{W}_1\ \mathrm{ELU}(\ \mathbf{W}_0\ \mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1) + \mathbf{b}_2,$$

$$\mathrm{ELU}(x) = \max(x, 0) + \exp(\min(x, 0)) - 1.$$



Approach

- Full Parametrisation: Trajectory {Positions, Directions}, Styles, Joint {Positions, Rotations, Velocities}, Environment Information
- Prediction from state i to i+1



Approach

- The phase can be expressed as the Catmull-Rom spline function theta given four control points (a1, a2, a3, a4), the network weights beta for an arbitrary cyclic phase p.

$$
\begin{aligned}
\Theta(p;\boldsymbol{\beta}) = \quad & \boldsymbol{\alpha}_{k_1} \\
& + w\left(\tfrac{1}{2}\boldsymbol{\alpha}_{k_2} - \tfrac{1}{2}\boldsymbol{\alpha}_{k_0}\right) \\
& + w^2\left(\boldsymbol{\alpha}_{k_0} - \tfrac{5}{2}\boldsymbol{\alpha}_{k_1} + 2\boldsymbol{\alpha}_{k_2} - \tfrac{1}{2}\boldsymbol{\alpha}_{k_3}\right) \\
& + w^3\left(\tfrac{3}{2}\boldsymbol{\alpha}_{k_1} - \tfrac{3}{2}\boldsymbol{\alpha}_{k_2} + \tfrac{1}{2}\boldsymbol{\alpha}_{k_3} - \tfrac{1}{2}\boldsymbol{\alpha}_{k_0}\right)
\end{aligned}
$$

$$
w = \frac{4p}{2\pi} \pmod{1}
$$

$$
k_n = \left\lfloor \frac{4p}{2\pi} \right\rfloor + n - 1 \pmod{4}.
$$

Approach

- The network is trained using the Adam optimiser, with respect to the following cost function:

$$\text{Cost}(\mathbf{X}, \mathbf{Y}, \mathbf{P}; \boldsymbol{\beta}) = \|\mathbf{Y} - \Phi(\mathbf{X}; \Theta(\mathbf{P}; \boldsymbol{\beta}))\| + \gamma\ |\boldsymbol{\beta}|.$$

where

→ X = input control parameters
→ Y = output parameters
→ P = phase parameters
→ beta = phase function parameters
→ drop-out rate of 0.7
→ Backpropagation using AdamWR *(Loshchilov, Huttner 2017)*

- 1 full training takes approximately 10-15 hours using NVIDIA GeForce GTX 980M using 10GB of processed motion data.

Discussion

- What is the crucial part?

→ **Phase Labeling / Extraction** (can be difficult)

→ **Style Labeling / Extraction** (is rather easy)

→ **Amount of Data** (not too hard)

→ **Data Completeness** (think before capture…)

Phase

- Biped Motion…

  → Phase encoding is straight forward for humans

  → Locomotion cycles can be extracted using foot contact patterns

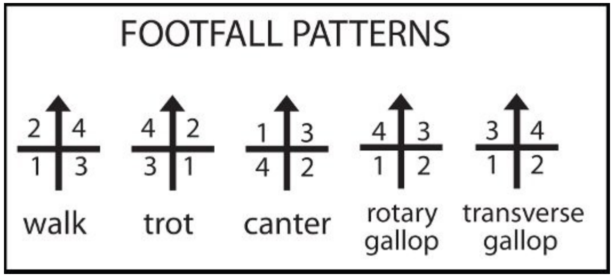
  → We can capture loads of data and suitable for our tasks

  → Motion retargeting is rather easy even for varying geometries

Phase

- Quadruped Motion…

  → Phase encoding is very difficult since many gait patterns exist



  → Motion is much more noisy and foot contacts are often unsharp

  → Difficult to capture specific motions from real animals

  → Motion retargeting is more complex due to high-frequency components

Phase

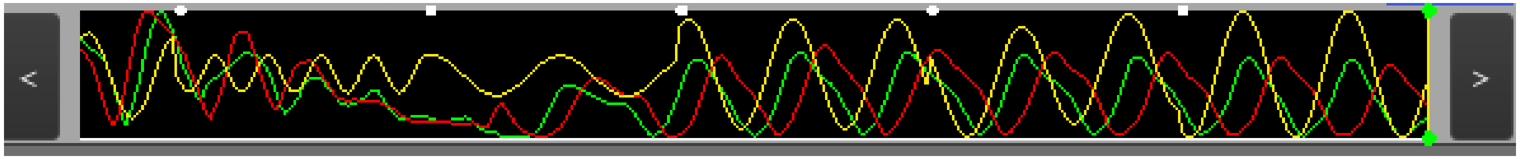
- Quadruped Motion…

  → Phase can be extracted by optimising a linearised trigonometric function to determine motion cycles within joint movements.

  → Positive **!or!** Negative turning points represent start/end of phase cycle

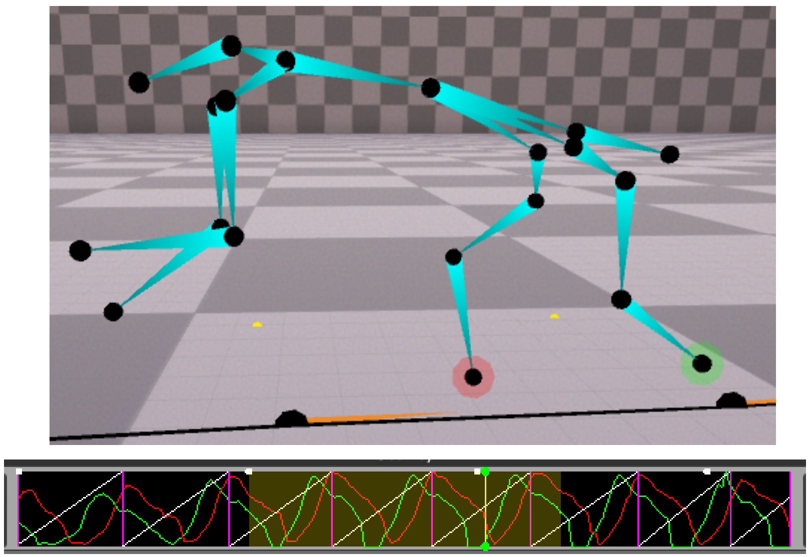
  → Fit the free variables for amplitude, phase, shift, offset, slope

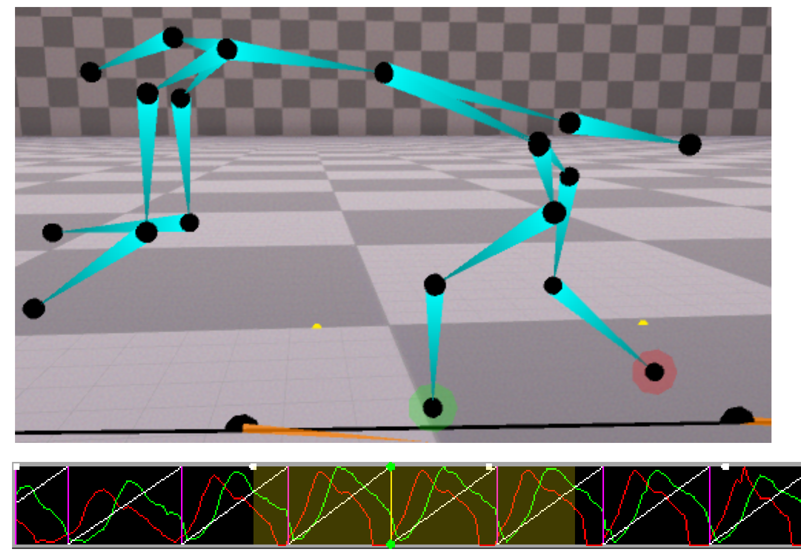
- y = a * sin(b*x – c) + m*x + b
- Loss = RMSE(y_red – y, -y_green – y)
- Yellow Curve = Fitted cyclic function over frequency windows with t=1s



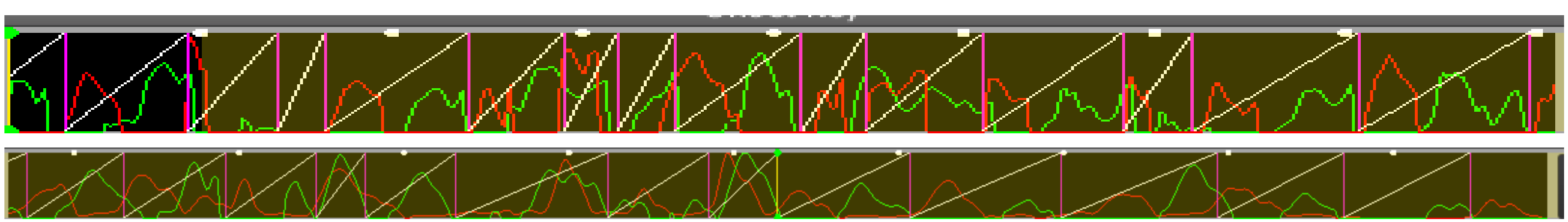
Phase

- Quadruped Motion…



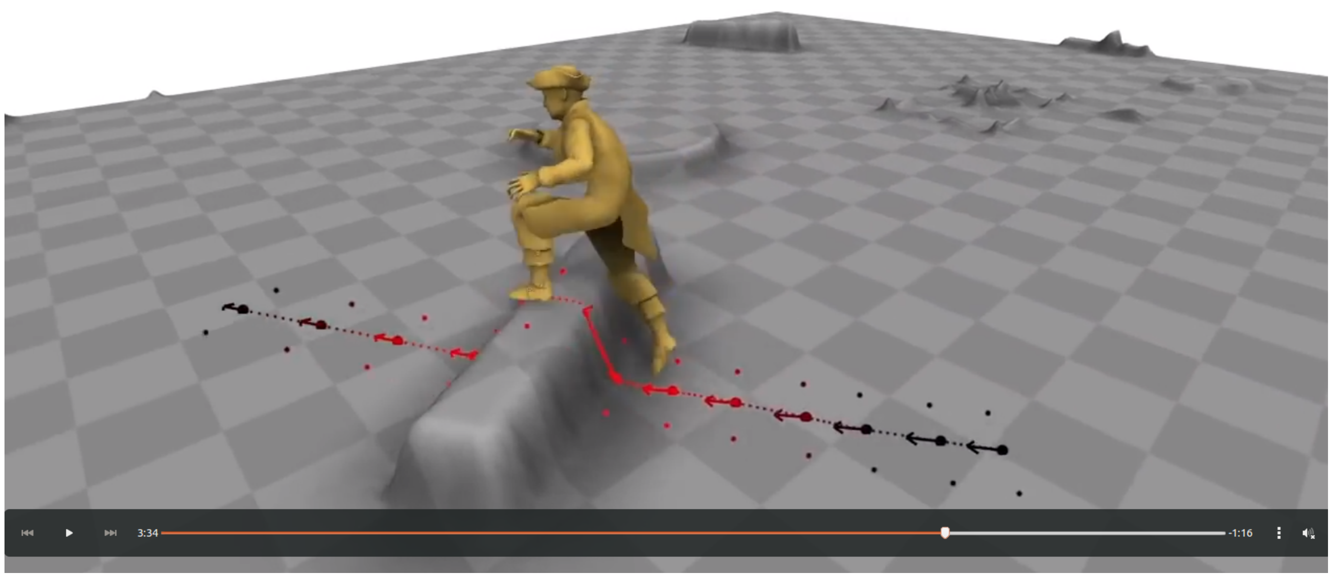


- More complex…



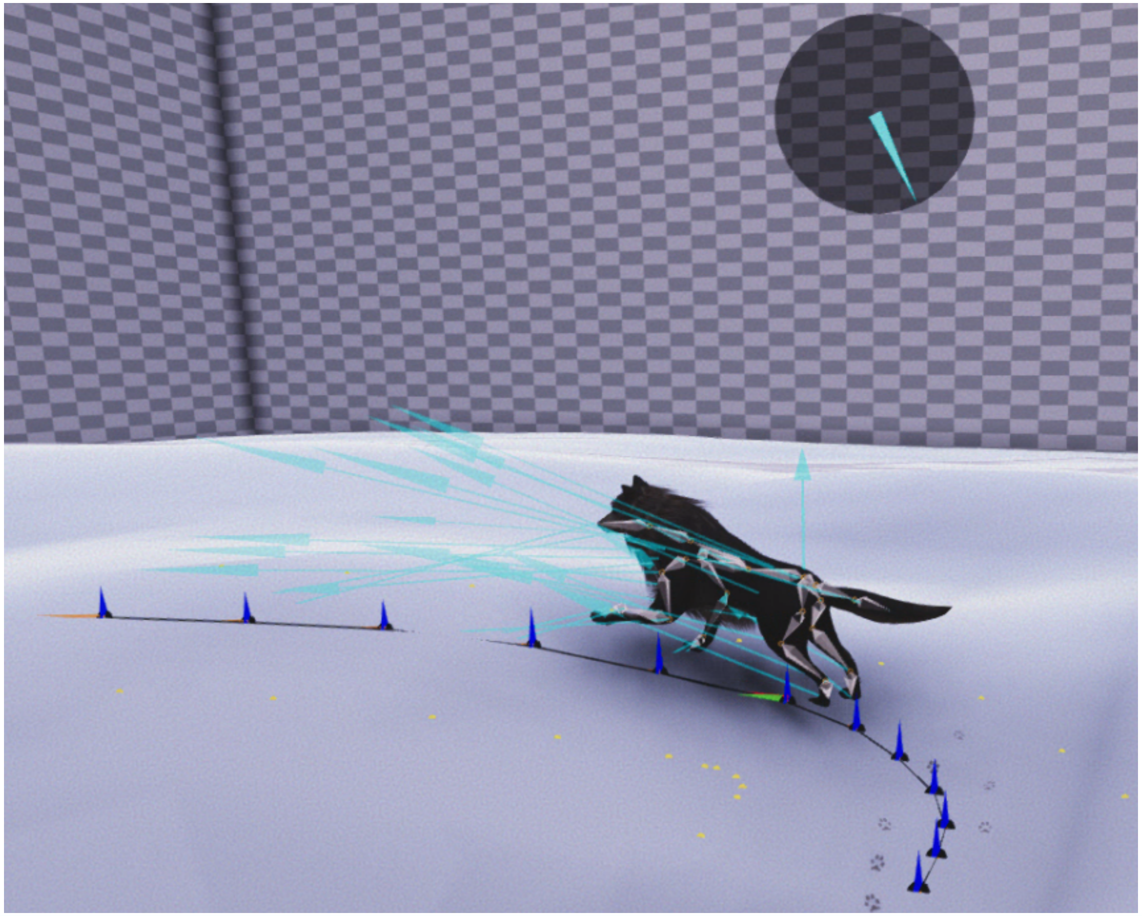
Results

Demo Biped Locomotion



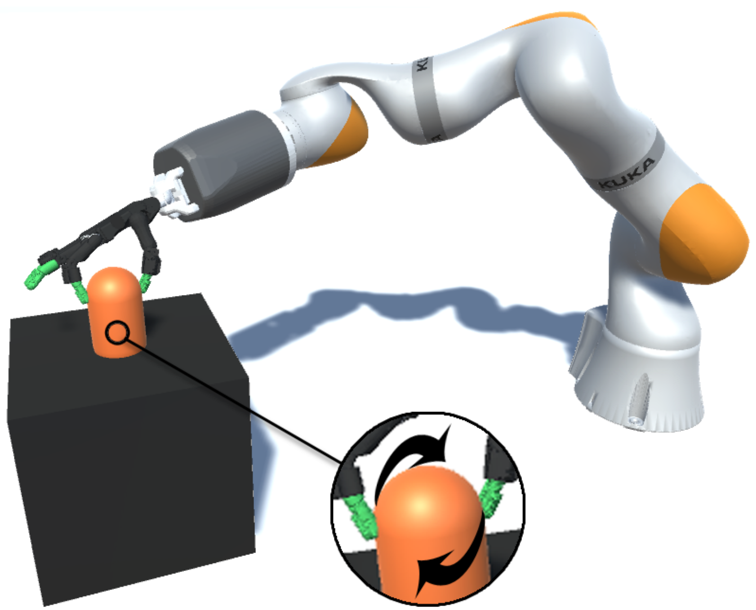
Results

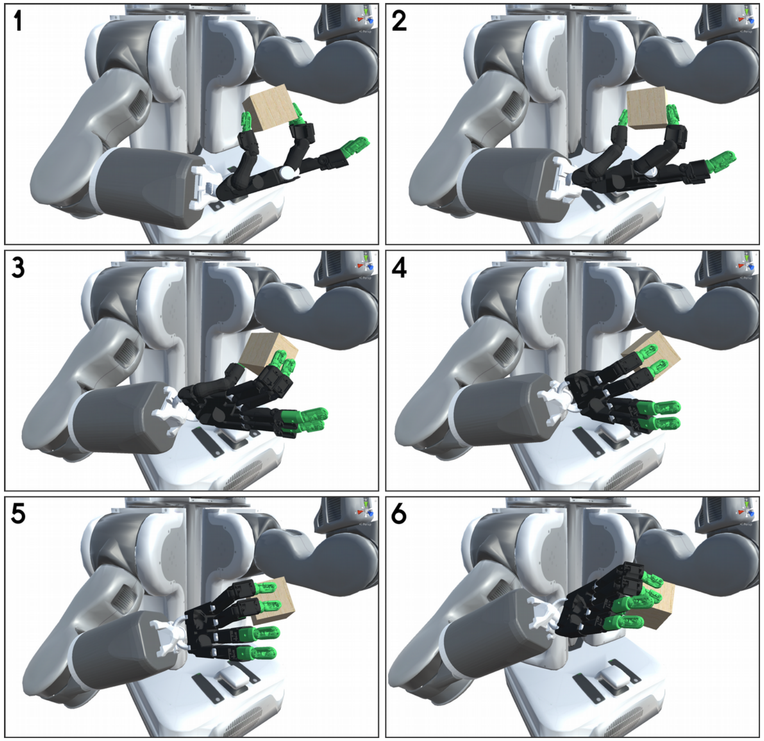
Demo Quadruped Locomotion



Setup Sketch for Robot Manipulation

- Dexterous manipulation by learning motion from humans.
- Data can be generated either purely supervised or through imitation.





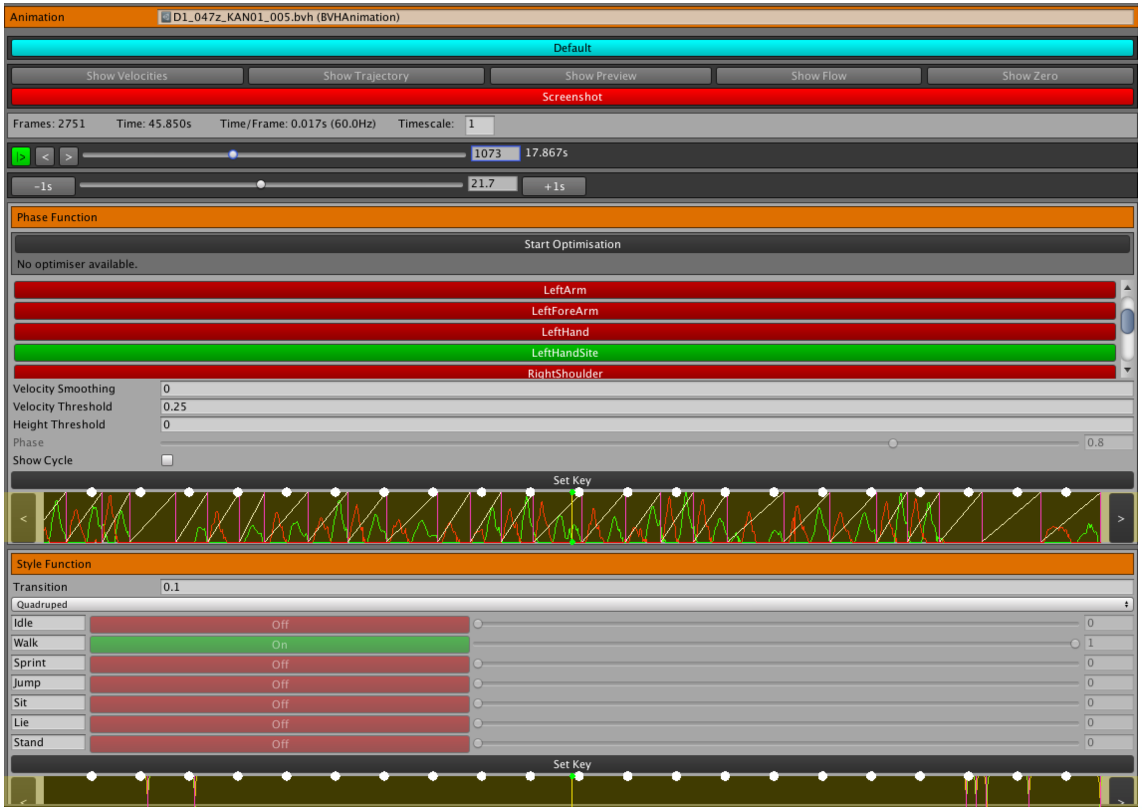
Setup Sketch for Robot Manipulation

- Perform motion capture using your favourite hardware or human-demonstration (e.g. Perception Neuron is quite good for hand motion recording)
- Make sure to record a variety of motion trajectories which can later be controlled by user input control signals
- If desired, also record motions which avoid obstacles
- Save joint positions, rotations, velocities relative to the root, and store environment geometry (e.g. use a CNN-Octree representation as in Wang et. al. 2017)

The University of Edinburgh

Setup Sketch for Robot Manipulation

- Export motion as .bvh and perform phase and style labeling, either (semi)-automatically scripted or using the developed tools in Unity3D



Setup Sketch for Robot Manipulation

- Export labelled motion sequences as state feature vectors in a data file, and use the PFNN tensorflow code for training


```
1 1 0 0.00000 0.46706 0.00000 0.99892 -0.02616 -0.03828 0.02653 0.99960 0.00944 0.00067 -0.00004 0.00083 0.00000 0.46706 0.00000 0.96865 -0.03063 -0.24652 -0.03540 -0.99926 -0.01491 0.000
1 2 0.0166667 0.00000 0.46673 0.00000 0.99883 -0.02753 -0.03978 0.02806 0.99952 0.01282 0.00068 -0.00007 0.00086 0.00000 0.46673 0.00000 0.96835 -0.03389 -0.24728 -0.03831 -0.99918 -0.013
1 3 0.0333334 0.00000 0.46693 0.00000 0.99885 -0.02781 -0.03909 0.02834 0.99951 0.01303 0.00075 0.00000 0.00086 0.00000 0.46693 0.00000 0.96695 -0.03432 -0.25266 -0.03923 -0.99913 -0.0144
1 4 0.0500001 0.00000 0.46680 0.00000 0.99881 -0.03012 -0.03845 0.03066 0.99944 0.01366 0.00085 0.00006 0.00086 0.00000 0.46680 0.00000 0.96582 -0.03679 -0.25658 -0.04220 -0.99899 -0.0155
1 5 0.0666668 0.00000 0.46706 0.00000 0.99878 -0.03114 -0.03821 0.03168 0.99940 0.01377 0.00087 0.00010 0.00081 0.00000 0.46706 0.00000 0.96459 -0.03926 -0.26083 -0.04476 -0.99888 -0.0151
1 6 0.08333349 0.00000 0.46742 0.00000 0.99868 -0.03385 -0.03875 0.03439 0.99932 0.01350 0.00089 0.00009 0.00076 0.00000 0.46742 0.00000 0.96363 -0.04125 -0.26405 -0.04786 -0.99868 -0.018
1 7 0.1000002 0.00000 0.46732 0.00000 0.99847 -0.03863 -0.03953 0.03917 0.99915 0.01285 0.00098 0.00017 0.00072 0.00000 0.46732 0.00000 0.96253 -0.04412 -0.26755 -0.05169 -0.99844 -0.0213
1 8 0.1166669 0.00000 0.46749 0.00000 0.99823 -0.04396 -0.04007 0.04451 0.99893 0.01293 0.00103 0.00011 0.00071 0.00000 0.46749 0.00000 0.96148 -0.04772 -0.27071 -0.05660 -0.99808 -0.0250
1 9 0.1333336 0.00000 0.46784 0.00000 0.99795 -0.04950 -0.04057 0.05007 0.99866 0.01317 0.00100 0.00006 0.00069 0.00000 0.46784 0.00000 0.96040 -0.05137 -0.27384 -0.06119 -0.99775 -0.0274
1 10 0.1500003 0.00000 0.46775 0.00000 0.99769 -0.05377 -0.04146 0.05438 0.99842 0.01393 0.00096 0.00007 0.00064 0.00000 0.46775 0.00000 0.95994 -0.05428 -0.27491 -0.06502 -0.99743 -0.030
1 11 0.166667 0.00000 0.46775 0.00000 0.99742 -0.05703 -0.04360 0.05771 0.99823 0.01446 0.00093 0.00002 0.00057 0.00000 0.46775 0.00000 0.95983 -0.05659 -0.27480 -0.06826 -0.99712 -0.0331
1 12 0.1833337 0.00000 0.46773 0.00000 0.99722 -0.05998 -0.04426 0.06072 0.99803 0.01565 0.00086 -0.00006 0.00055 0.00000 0.46773 0.00000 0.95930 -0.05868 -0.27621 -0.07130 -0.99681 -0.03
1 13 0.2000004 0.00000 0.46762 0.00000 0.99697 -0.06367 -0.04474 0.06449 0.99777 0.01717 0.00080 -0.00009 0.00049 0.00000 0.46762 0.00000 0.95885 -0.06109 -0.27725 -0.07466 -0.99646 -0.03
1 14 0.2166671 0.00000 0.46746 0.00000 0.99686 -0.06424 -0.04627 0.06512 0.99772 0.01787 0.00073 -0.00013 0.00043 0.00000 0.46746 0.00000 0.95888 -0.06176 -0.27703 -0.07621 -0.99622 -0.04
1 15 0.2333338 0.00000 0.46721 0.00000 0.99680 -0.06465 -0.04697 0.06563 0.99765 0.01974 0.00070 -0.00016 0.00035 0.00000 0.46721 0.00000 0.95900 -0.06234 -0.27647 -0.07787 -0.99592 -0.04
1 16 0.2500005 0.00000 0.46698 0.00000 0.99671 -0.06596 -0.04708 0.06704 0.99751 0.02192 0.00066 -0.00017 0.00030 0.00000 0.46698 0.00000 0.95883 -0.06363 -0.27678 -0.08005 -0.99561 -0.04
1 17 0.2666672 0.00000 0.46680 0.00000 0.99663 -0.06772 -0.04622 0.06885 0.99735 0.02332 0.00059 -0.00018 0.00025 0.00000 0.46680 0.00000 0.95867 -0.06503 -0.27699 -0.08177 -0.99543 -0.04
1 18 0.2833339 0.00000 0.46658 0.00000 0.99668 -0.06778 -0.04518 0.06893 0.99732 0.02441 0.00050 -0.00011 0.00018 0.00000 0.46658 0.00000 0.95867 -0.06519 -0.27696 -0.08235 -0.99531 -0.05
1 19 0.3000006 0.00000 0.46639 0.00000 0.99673 -0.06714 -0.04486 0.06829 0.99736 0.02460 0.00047 -0.00007 0.00013 0.00000 0.46639 0.00000 0.95857 -0.06506 -0.27734 -0.08256 -0.99523 -0.05
1 20 0.3166673 0.00000 0.46652 0.00000 0.99677 -0.06688 -0.04448 0.06804 0.99737 0.02502 0.00045 -0.00008 0.00009 0.00000 0.46652 0.00000 0.95830 -0.06477 -0.27833 -0.08274 -0.99515 -0.05
1 21 0.333334 0.00000 0.46656 0.00000 0.99676 -0.06700 -0.04462 0.06814 0.99737 0.02460 0.00040 -0.00006 0.00007 0.00000 0.46656 0.00000 0.95820 -0.06439 -0.27875 -0.08273 -0.99508 -0.054
1 22 0.3500007 0.00000 0.46631 0.00000 0.99671 -0.06762 -0.04475 0.06872 0.99736 0.02365 0.00039 -0.00002 0.00005 0.00000 0.46631 0.00000 0.95808 -0.06473 -0.27908 -0.08303 -0.99507 -0.05
1 23 0.3666674 0.00000 0.46622 0.00000 0.99667 -0.06787 -0.04527 0.06897 0.99735 0.02326 0.00043 -0.00004 0.00006 0.00000 0.46622 0.00000 0.95768 -0.06474 -0.28045 -0.08298 -0.99511 -0.05
1 24 0.3833341 0.00000 0.46625 0.00000 0.99657 -0.06903 -0.04568 0.07014 0.99727 0.02324 0.00042 -0.00006 0.00010 0.00000 0.46625 0.00000 0.95714 -0.06507 -0.28221 -0.08306 -0.99517 -0.05
1 25 0.4000008 0.00000 0.46628 0.00000 0.99645 -0.07089 -0.04535 0.07201 0.99712 0.02365 0.00037 -0.00002 0.00015 0.00000 0.46628 0.00000 0.95641 -0.06579 -0.28453 -0.08395 -0.99511 -0.05
1 26 0.4166675 0.00000 0.46619 0.00000 0.99634 -0.07189 -0.04615 0.07305 0.99704 0.02394 0.00036 0.00000 0.00017 0.00000 0.46619 0.00000 0.95587 -0.06599 -0.28628 -0.08426 -0.99509 -0.051
1 27 0.4333342 0.00000 0.46620 0.00000 0.99631 -0.07157 -0.04730 0.07279 0.99704 0.02473 0.00037 0.00001 0.00018 0.00000 0.46620 0.00000 0.95529 -0.06538 -0.28834 -0.08398 -0.99508 -0.052
1 28 0.4500009 0.00000 0.46623 0.00000 0.99624 -0.07232 -0.04770 0.07360 0.99695 0.02584 0.00033 -0.00003 0.00018 0.00000 0.46623 0.00000 0.95479 -0.06528 -0.29001 -0.08406 -0.99506 -0.05
1 29 0.4666676 0.00000 0.46628 0.00000 0.99616 -0.07315 -0.04808 0.07447 0.99688 0.02619 0.00026 -0.00004 0.00017 0.00000 0.46628 0.00000 0.95430 -0.06523 -0.29163 -0.08416 -0.99505 -0.05
1 30 0.4833343 0.00000 0.46608 0.00000 0.99612 -0.07300 -0.04923 0.07437 0.99688 0.02651 0.00025 -0.00002 0.00012 0.00000 0.46608 0.00000 0.95396 -0.06503 -0.29280 -0.08388 -0.99510 -0.05
1 31 0.500001 0.00000 0.46594 0.00000 0.99610 -0.07210 -0.05092 0.07356 0.99692 0.02736 0.00027 -0.00007 0.00005 0.00000 0.46594 0.00000 0.95392 -0.06455 -0.29303 -0.08327 -0.99518 -0.051
1 32 0.5166677 0.00000 0.46605 0.00000 0.99605 -0.07224 -0.05172 0.07376 0.99688 0.02814 0.00023 -0.00010 0.00001 0.00000 0.46605 0.00000 0.95398 -0.06441 -0.29286 -0.08317 -0.99517 -0.05
1 33 0.5333344 0.00000 0.46581 0.00000 0.99599 -0.07299 -0.05170 0.07448 0.99684 0.02746 0.00018 -0.00004 -0.00002 0.00000 0.46581 0.00000 0.95426 -0.06470 -0.29188 -0.08364 -0.99509 -0.0
1 34 0.5500011 0.00000 0.46569 0.00000 0.99597 -0.07300 -0.05204 0.07453 0.99683 0.02798 0.00018 -0.00003 -0.00008 0.00000 0.46569 0.00000 0.95446 -0.06492 -0.29119 -0.08364 -0.99512 -0.0
1 35 0.5666678 0.00000 0.46587 0.00000 0.99605 -0.07186 -0.05221 0.07340 0.99690 0.02835 0.00020 -0.00002 -0.00[illegible]2 0.00000 0.46587 0.00000 0.95445 -0.06493 -0.29123 -0.08326 -0.99522 -0.0
1 36 0.5833344 0.00000 0.46576 0.00000 0.99607 -0.07152 -0.05230 0.07305 0.99694 0.02794 0.00017 0.00003 -0.00013 0.00000 0.46576 0.00000 0.95493 -0.06527 -0.28957 -0.08328 -0.99525 -0.05
1 37 0.6000012 0.00000 0.46596 0.00000 0.99611 -0.07148 -0.05157 0.07292 0.99698 0.02654 0.00013 0.00005 -0.00013 0.00000 0.46596 0.00000 0.95483 -0.06514 -0.28994 -0.08304 -0.99530 -0.04
1 38 0.6166679 0.00000 0.46599 0.00000 0.99616 -0.07112 -0.05110 0.07252 0.99703 0.02606 0.00015 0.00003 -0.00015 0.00000 0.46599 0.00000 0.95475 -0.06517 -0.29019 -0.08289 -0.99534 -0.04
1 39 0.6333346 0.00000 0.46597 0.00000 0.99620 -0.07062 -0.05096 0.07197 0.99709 0.02520 0.00018 0.00010 -0.00016 0.00000 0.46597 0.00000 0.95498 -0.06496 -0.28946 -0.08246 -0.99541 -0.04
1 40 0.6500013 0.00000 0.46608 0.00000 0.99614 -0.07190 -0.05036 0.07320 0.99702 0.02443 0.00017 0.00007 -0.00012 0.00000 0.46608 0.00000 0.95474 -0.06467 -0.29034 -0.08197 -0.99549 -0.04
1 41 0.6666679 0.00000 0.46636 0.00000 0.99607 -0.07351 -0.04947 0.07476 0.99692 0.02390 0.00015 0.00002 -0.00007 0.00000 0.46636 0.00000 0.95433 -0.06448 -0.29171 -0.08178 -0.99552 -0.04
1 42 0.6833346 0.00000 0.46638 0.00000 0.99603 -0.07400 -0.04938 0.07525 0.99688 0.02395 0.00017 0.00003 -0.00004 0.00000 0.46638 0.00000 0.95437 -0.06523 -0.29141 -0.08228 -0.99552 -0.04
1 43 0.7000014 0.00000 0.46612 0.00000 0.99603 -0.07386 -0.04975 0.07512 0.99689 0.02386 0.00018 0.00003 -0.00002 0.00000 0.46612 0.00000 0.95455 -0.06490 -0.29090 -0.08207 -0.99551 -0.04
1 44 0.7166681 0.00000 0.46615 0.00000 0.99592 -0.07522 -0.04993 0.07649 0.99678 0.02406 0.00015 -0.00005 0.00004 0.00000 0.46615 0.00000 0.95469 -0.06514 -0.29039 -0.08250 -0.99544 -0.04
1 45 0.7333348 0.00000 0.46626 0.00000 0.99581 -0.07690 -0.04955 0.07819 0.99663 0.02474 0.00010 -0.00007 0.00005 0.00000 0.46626 0.00000 0.95474 -0.06597 -0.29004 -0.08346 -0.99534 -0.04
1 46 0.7500015 0.00000 0.46604 0.00000 0.99576 -0.07725 -0.04992 0.07856 0.99660 0.02485 0.00012 -0.00007 0.00004 0.00000 0.46604 0.00000 0.95477 -0.06668 -0.28978 -0.08410 -0.99530 -0.04
1 47 0.7666681 0.00000 0.46595 0.00000 0.99573 -0.07710 -0.05079 0.07849 0.99658 0.02597 0.00011 -0.00008 0.00002 0.00000 0.46595 0.00000 0.95505 -0.06645 -0.28891 -0.08426 -0.99521 -0.04
1 48 0.7833349 0.00000 0.46570 0.00000 0.99564 -0.07820 -0.05086 0.07962 0.99648 0.02636 0.00006 -0.00012 0.00002 0.00000 0.46570 0.00000 0.95524 -0.06677 -0.28821 -0.08490 -0.99509 -0.05
1 49 0.8000016 0.00000 0.46565 0.00000 0.99564 -0.07881 -0.04991 0.08023 0.99641 0.02711 0.00000 -0.00011 0.00001 0.00000 0.46565 0.00000 0.95580 -0.06841 -0.28594 -0.08670 -0.99489 -0.05
1 50 0.8166683 0.00000 0.46551 0.00000 0.99569 -0.07814 -0.04991 0.07959 0.99644 0.02765 -0.00002 -0.00009 -0.00001 0.00000 0.46551 0.00000 0.95603 -0.06866 -0.28513 -0.08684 -0.99488 -0.
1 51 0.833335 0.00000 0.46536 0.00000 0.99581 -0.07643 -0.05022 0.07792 0.99656 0.02827 -0.00006 -0.00004 -0.00004 0.00000 0.46536 0.00000 0.95642 -0.06789 -0.28398 -0.08590 -0.99497 -0.0
1 52 0.8500016 0.00000 0.46541 0.00000 0.99589 -0.07540 -0.05024 0.07686 0.99665 0.02787 -0.00012 -0.00005 -0.00005 0.00000 0.46541 0.00000 0.95668 -0.06709 -0.28331 -0.08488 -0.99509 -0.
1 53 0.8666683 0.00000 0.46546 0.00000 0.99604 -0.07375 -0.04972 0.07522 0.99677 0.02822 -0.00014 -0.00004 -0.00005 0.00000 0.46546 0.00000 0.95674 -0.06628 -0.28328 -0.08355 -0.99528 -0.
```

Setup Sketch for Robot Manipulation

Wait ~24 hours training, e.g. using your favourite coffee…

Setup Sketch for Robot Manipulation

- Import saved binary files from tensorflow which represent the precomputed control points and weights for the PFNN
- Provide a suitable input to the network, including the past and estimated future trajectory states (do not need to be accurate) as well as the robot/environment state (needs to be accurate)
- Update and correct the estimated states using the PFNN output (improves accuracy of successive predictions)
- Keep predicting all trajectory states until the goal is reached
- Start manipulating…

Conclusion

- Phase-Functioned Neural Network as a novel method for motion learning in animation and robotics
- Can handle high-dimensional motion with different style types and including environment information for creating realistic motion in real-time
- Current state-of-the-art for humanoid character animation
- First successful tests also on quadruped geometries
- Promising to be applicable for dexterous robot manipulation

Implementations

- Phase-Functioned Neural Network Visualisation and Motion Capture Labeling Editor of .bvh files in Unity3D

- General data-preprocessing production pipeline which can be used for different geometries (bipeds, quadrupeds, robots, …)
  https://github.com/sebastianstarke/AI4Animation
  (IP, Code and Data belongs to the UoE, and is only available for non-commerical use)

- Phase-Functioned Neural Network in Tensorflow (including variations for transfer and multi-task learning)
  https://github.com/ShikamaruZhang
  (IP, Code and Data belongs to the UoE, and is only available for non-commerical use)