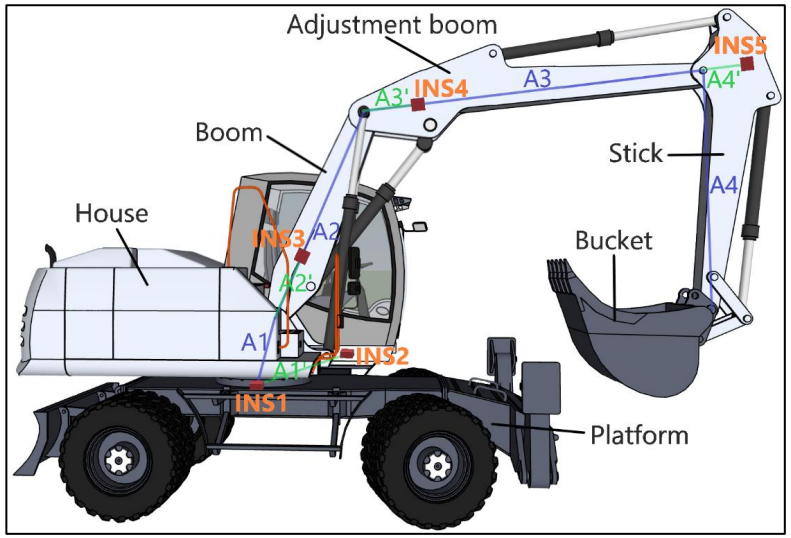
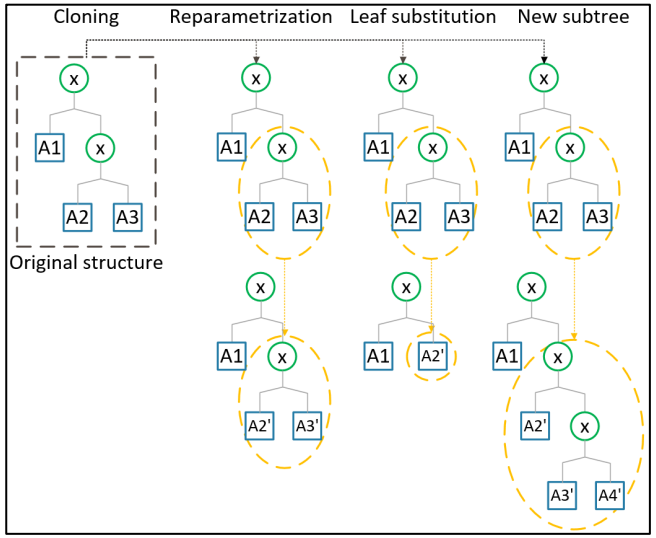
Estimation of the kinematic chain of a construction vehicle

Kinematic chain estimation

- Forward kinematic chain estimation of an excavator by bacterial programming based on randomly placed inertial navigation systems per segments with microelectromechanical sensors within.
- The unknown model structure and parameters are established and identified by BP without any a priori or given information about the device according to Denavit-Hartenberg transformation conventions



+



=

$$A^{G}_{INS_{AB}} =$$

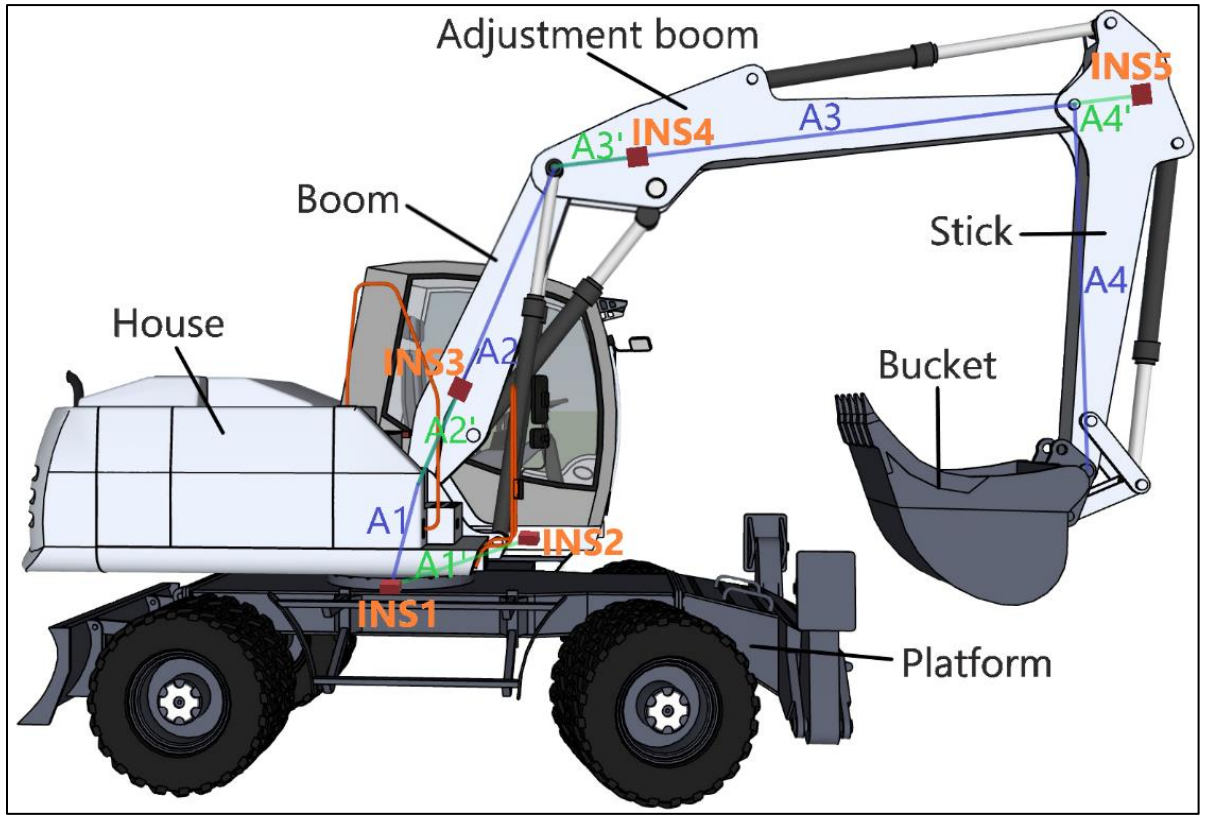
$$Analytic \rightarrow \begin{Bmatrix} 0 \\ 0 \\ 0 \\ x_W \end{Bmatrix} \begin{Bmatrix} \psi_H \\ 0 \\ 0 \\ -1.250 \end{Bmatrix} \begin{Bmatrix} 3.141 \\ 0 \\ -1.571 \\ -0.400 \end{Bmatrix}$$

$$\begin{Bmatrix} \theta_B \\ 0 \\ 0 \\ 0.580 \end{Bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1.285 \end{Bmatrix} \begin{Bmatrix} \theta_{AB} \\ 0 \\ 0 \\ 0.330 \end{Bmatrix}$$

$$BP \rightarrow \begin{Bmatrix} 0 \\ 0 \\ 0 \\ x_W \end{Bmatrix} \begin{Bmatrix} \psi_H \\ 0 \\ 0.001 \\ -1.262 \end{Bmatrix} \begin{Bmatrix} 3.141 \\ 0 \\ -1.571 \\ -0.400 \end{Bmatrix}$$

$$\begin{Bmatrix} \theta_B \\ 0 \\ 0 \\ 0.575 \end{Bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1.286 \end{Bmatrix} \begin{Bmatrix} \theta_{AB} \\ 0 \\ 0 \\ 0.340 \end{Bmatrix}$$

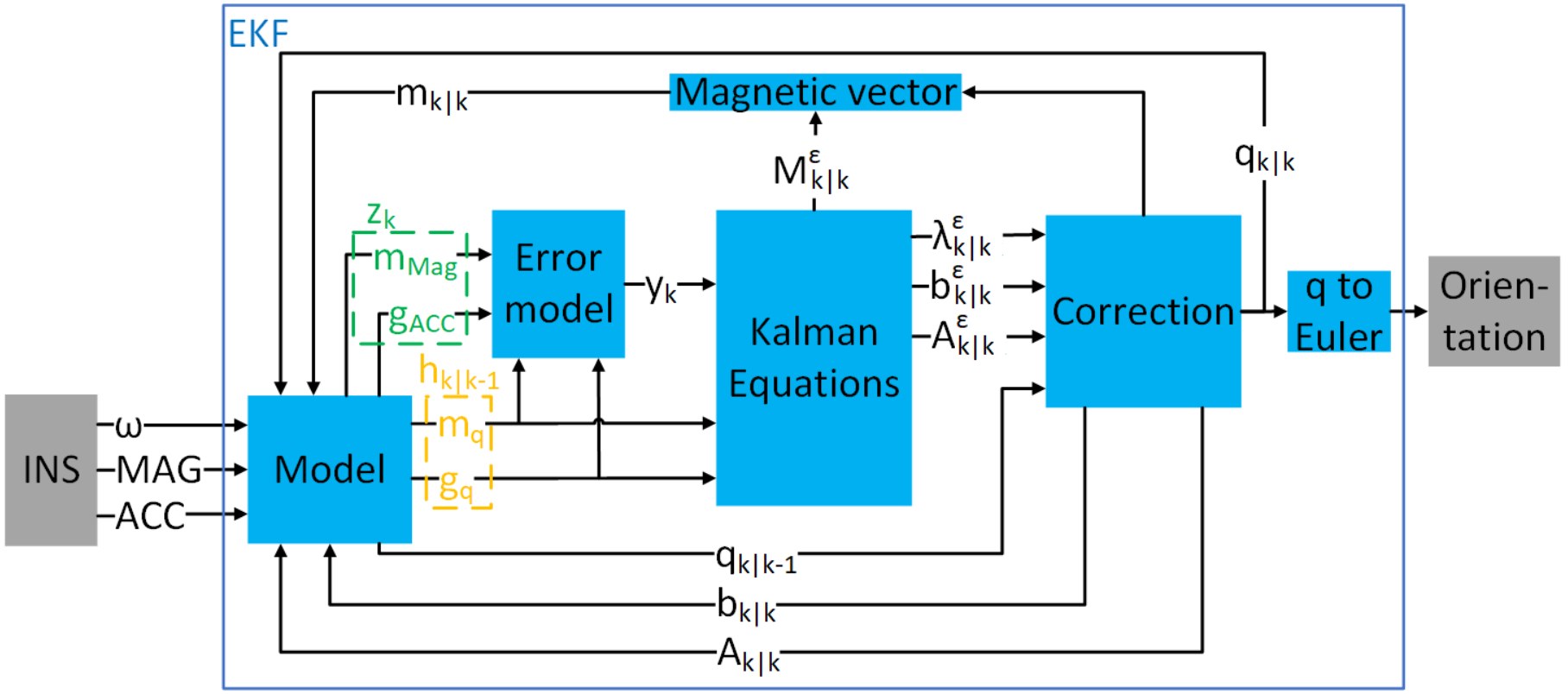
Kinematic analysis of a construction vehicle



- Illustration of analyzed system with links, transformations and INS positions. Blue lines highlight transformations from INS to joints, green lines represent transformations from joints to INS.

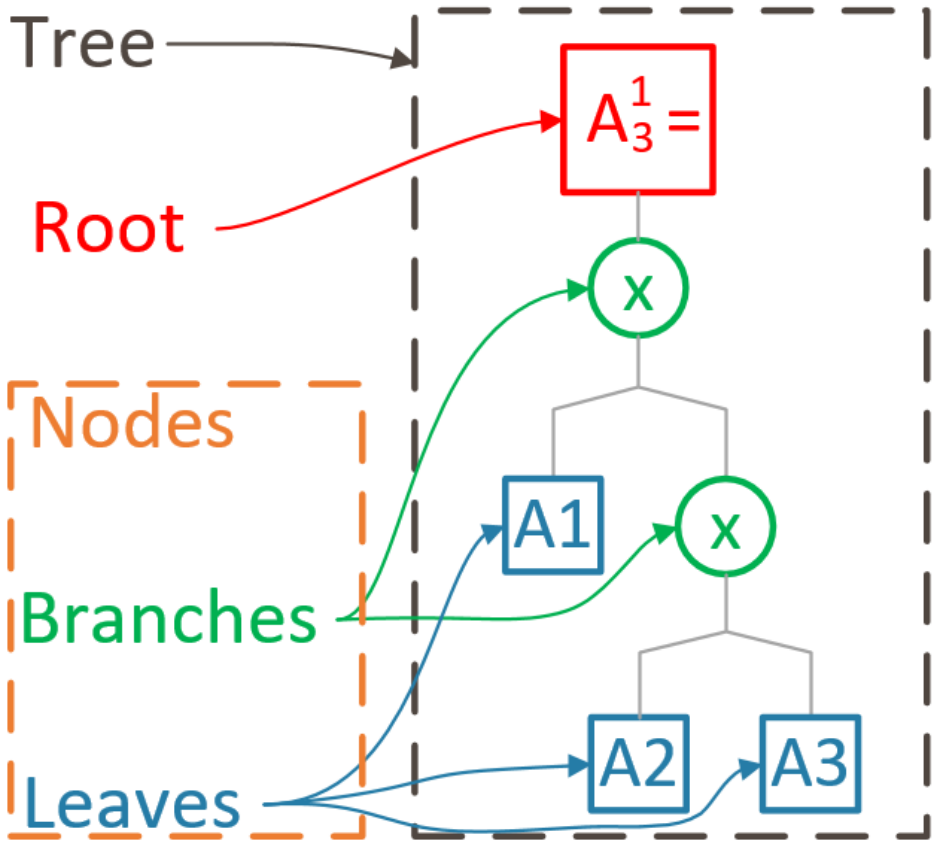
$$A^{G}_{INS_T} = A^{G}_{INS_P} \cdot A^{INS_P}_{INS_H} \cdots A^{T}_{INS_T}$$

Sensor fusion and angle estimation



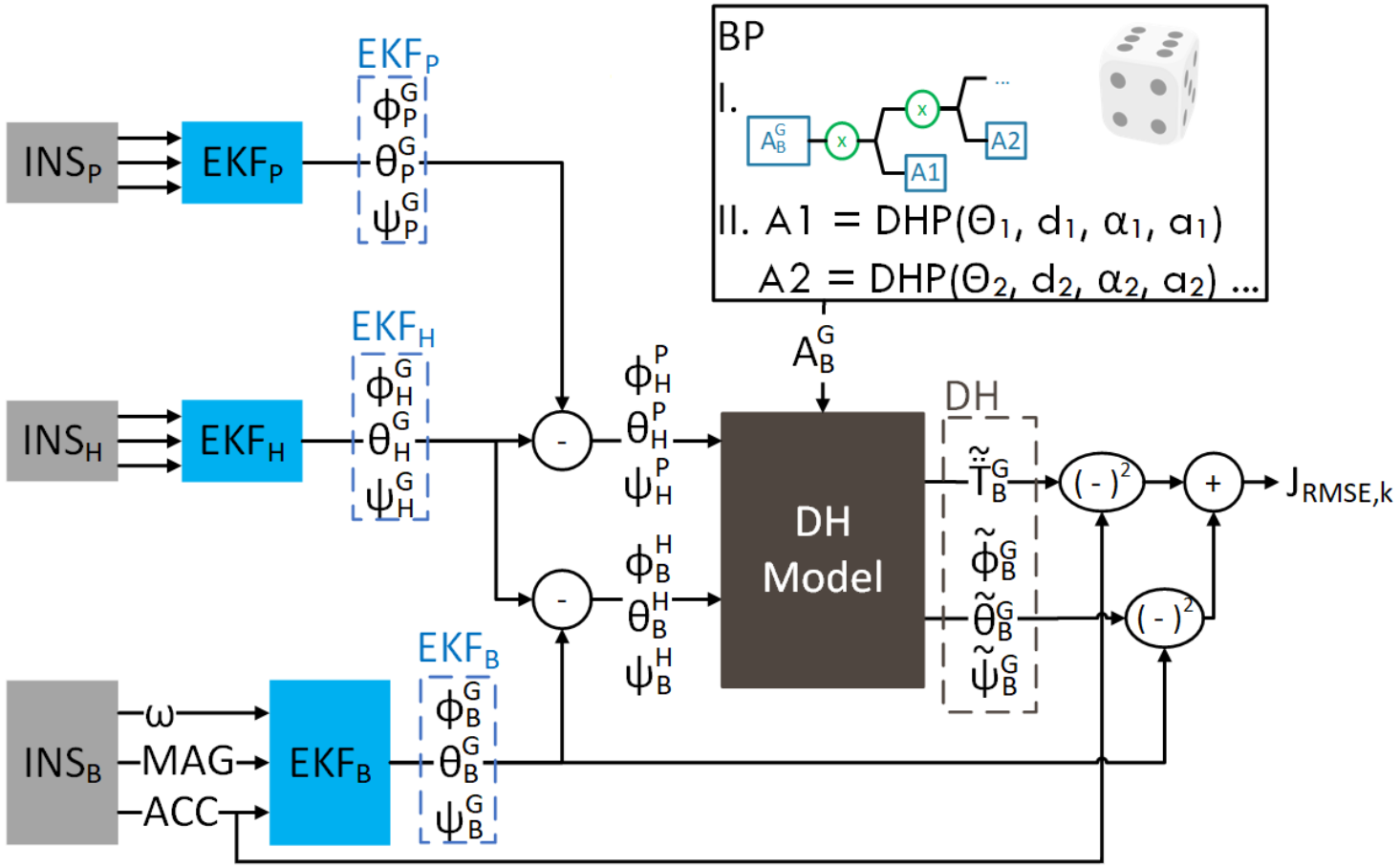
- Block diagram of error state estimation with Extended Kalman Filter with orientations represented in quaternions.

Representation of individuals in BP



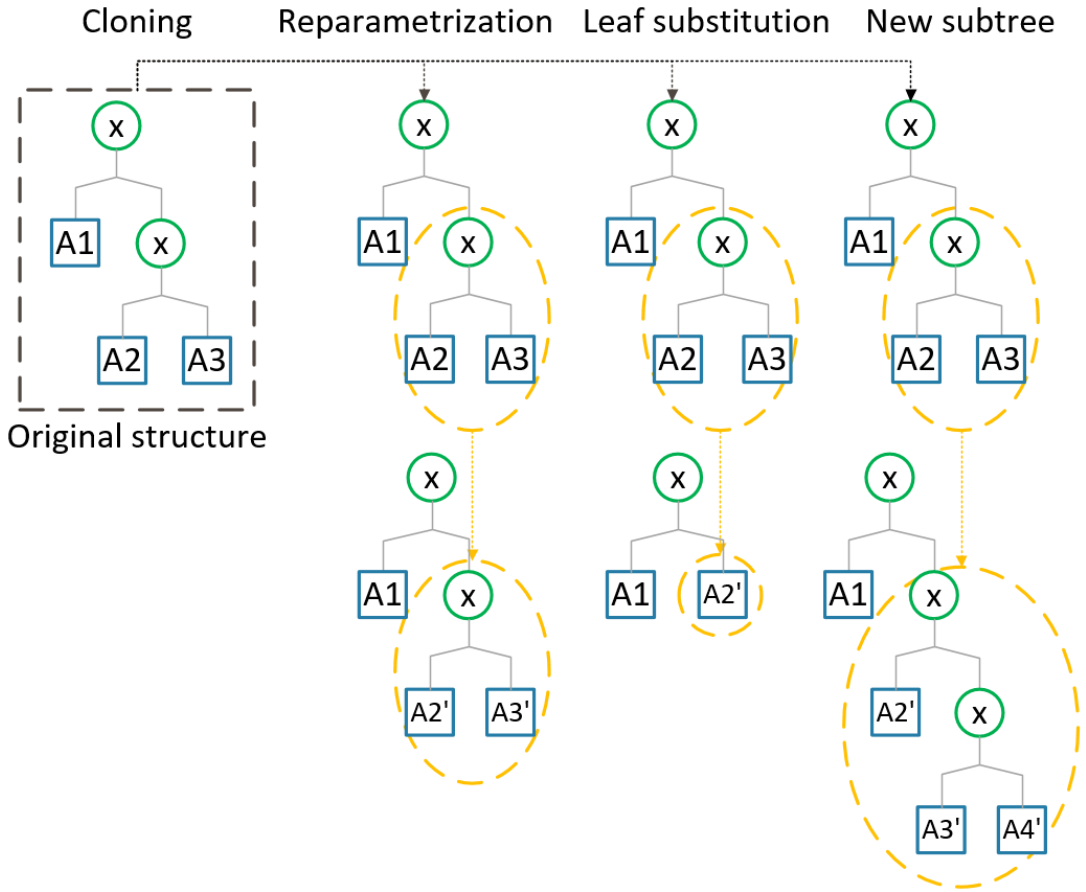
- Representation of BP terminology.
- Each BP individual also called as tree represents a DH transformation between two points.
- The tree root is the 4 × 4 DH matrix.
- The tree contains several nodes, that can be branches with mathematical operations or leaves with DH parameters belonging to one transformation.

Process flow



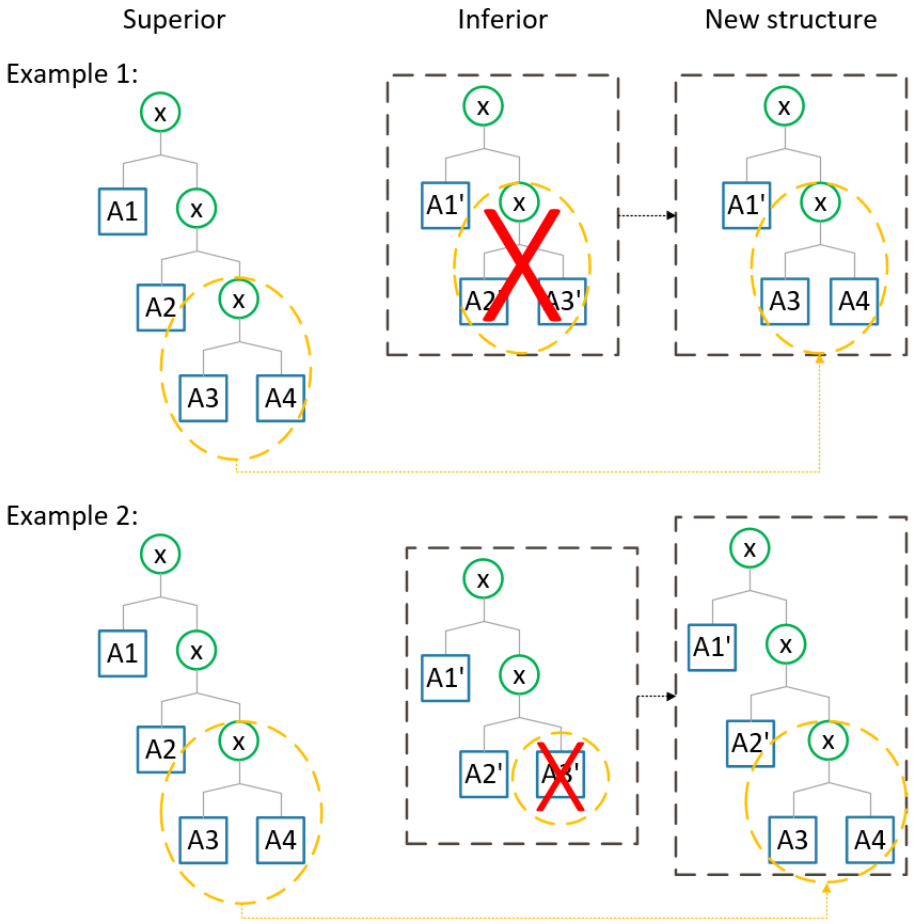
- Process flow from INS sensors to error (J_{RMSE}) calculation for a single *k* time step.
- Relative angles are calculated between EKF filters and substituted into BP determined DH model.
- Accelerations and orientations are calculated by DH model.
- They are then compared with sensor measured accelerations and EKF angles in ground coordinate system.

Bacterial mutation



- Three types of clone mutation during bacterial mutation after a node is selected.
- Tree reduction happens if a sub-tree is replaced by a leaf, expansion is possible through regrowing new sub-tree.
- The tree must be supervised and controlled in all cases.

Gene transfer



- Example about gene transfer of bacterial programming, when superior is passing sub-tree section to inferior individual.
- Just like in bacterial mutation, tree depth must be controlled in gene transfer.

Results

$$A^{G}_{INS_{AB}} =$$

$$Analytic \rightarrow \begin{Bmatrix} 0 \\ 0 \\ 0 \\ x_W \end{Bmatrix} \begin{Bmatrix} \psi_H \\ 0 \\ 0 \\ -1.250 \end{Bmatrix} \begin{Bmatrix} 3.141 \\ 0 \\ -1.571 \\ -0.400 \end{Bmatrix} \begin{Bmatrix} \theta_B \\ 0 \\ 0 \\ 0.580 \end{Bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1.285 \end{Bmatrix} \begin{Bmatrix} \theta_{AB} \\ 0 \\ 0 \\ 0.330 \end{Bmatrix}$$

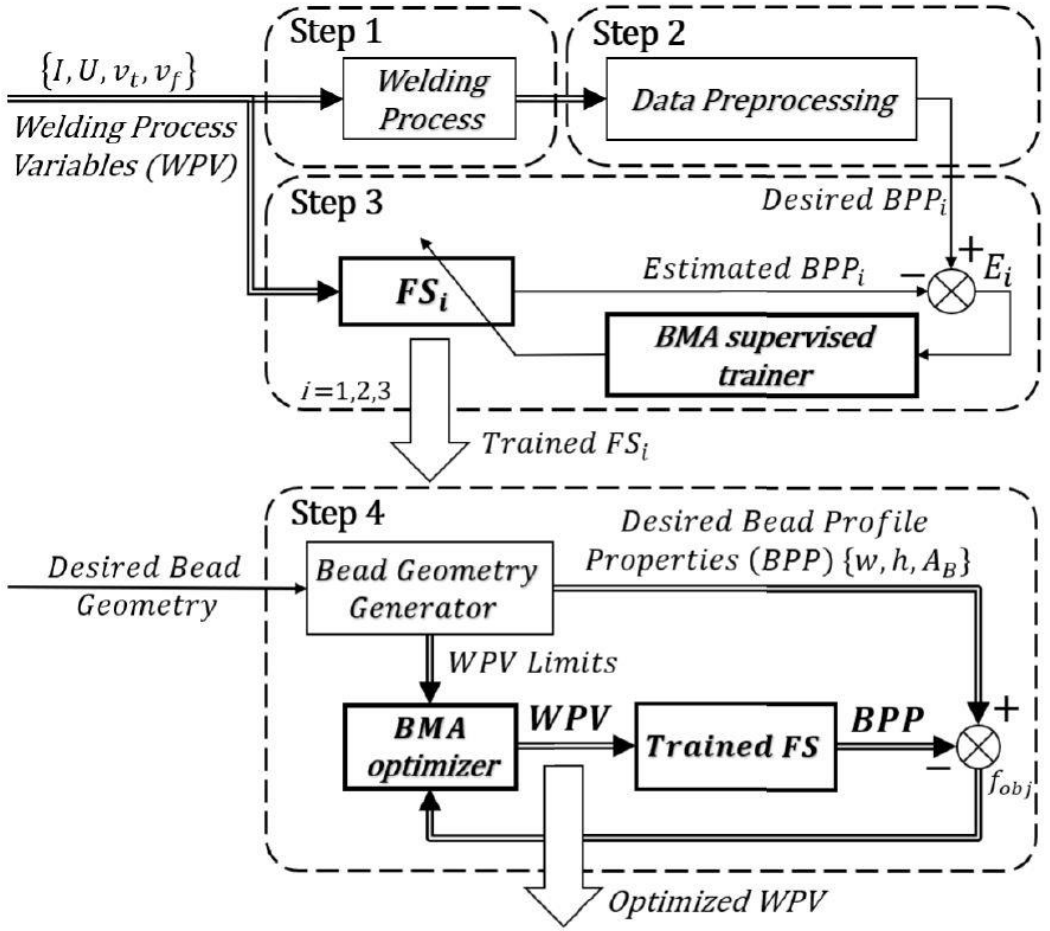
$$BP \rightarrow \begin{Bmatrix} 0 \\ 0 \\ 0 \\ x_W \end{Bmatrix} \begin{Bmatrix} \psi_H \\ 0 \\ 0.001 \\ -1.262 \end{Bmatrix} \begin{Bmatrix} 3.141 \\ 0 \\ -1.571 \\ -0.400 \end{Bmatrix} \begin{Bmatrix} \theta_B \\ 0 \\ 0 \\ 0.575 \end{Bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 1.286 \end{Bmatrix} \begin{Bmatrix} \theta_{AB} \\ 0 \\ 0 \\ 0.340 \end{Bmatrix}$$

- The exact theoretical Equation represents the solution with nominal geometric values from excavator data sheet.
- The estimated DH model is obtained by BP.
- The structure of the equations are identical, only small deviations in constants are observable causing some measurable error in cost functions.

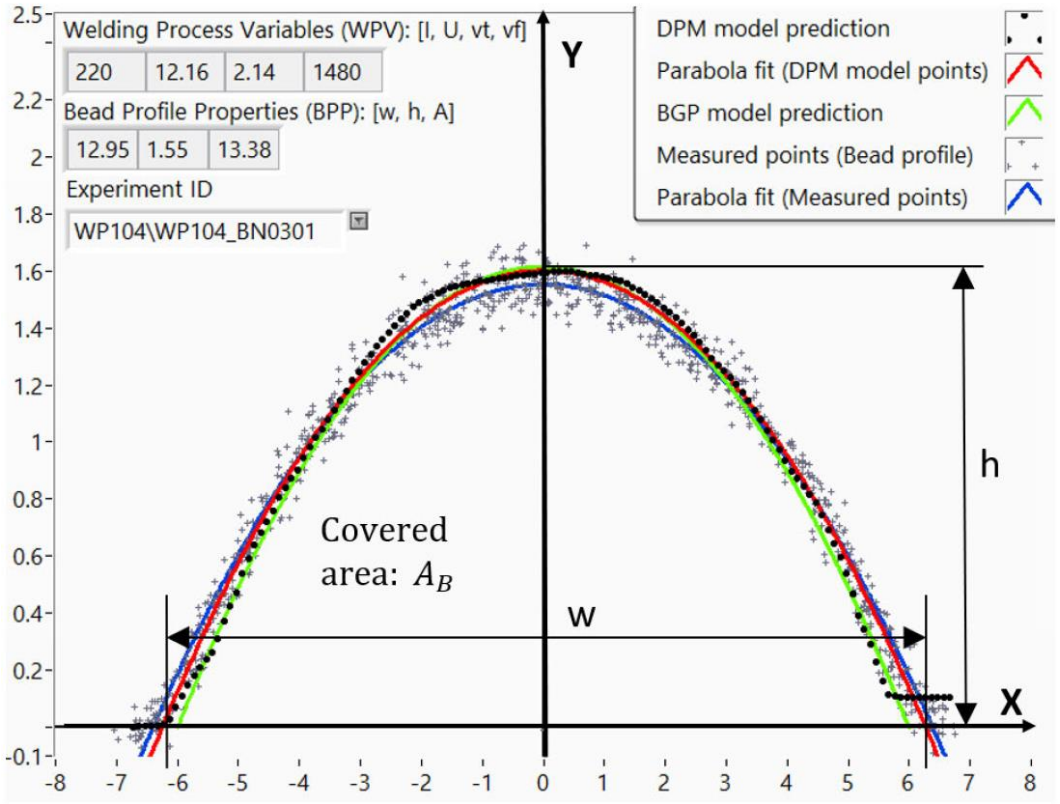
Welding robot

Modeling weld bead geometry

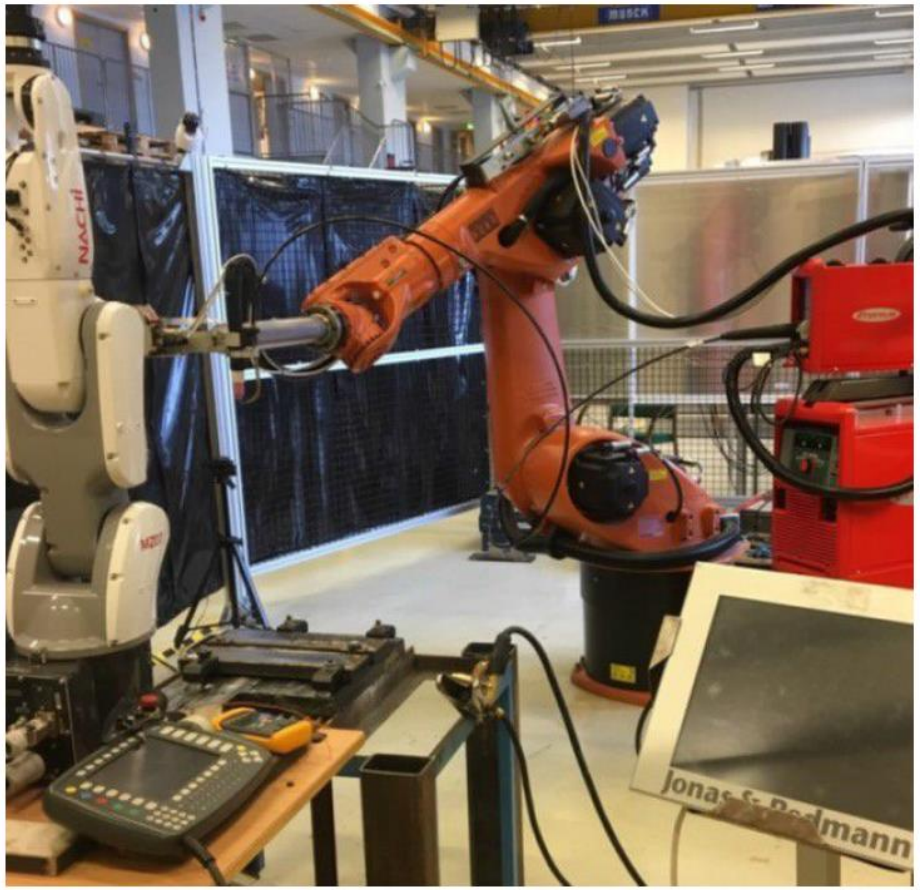
1. Database from welding experiments
2. Preprocessing of weld bead profiles
3. Train fuzzy systems with BMA (modeling)
    1. Bead Geometry Properties Model (BGP)
    2. Direct Profile Measurements Model (DPM)
4. Welding Process Variables optimization to achieve a specific bead geometry (planning)



Experiment design and execution

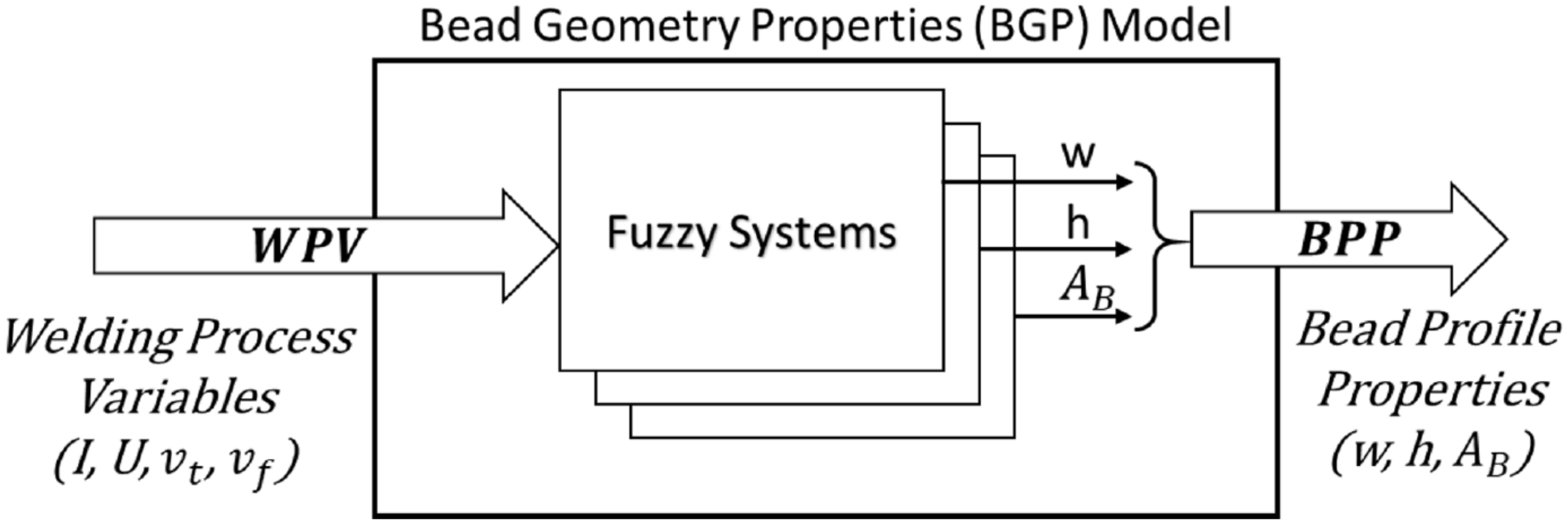


A weld bead cross-section as the result of the measurements and the models.

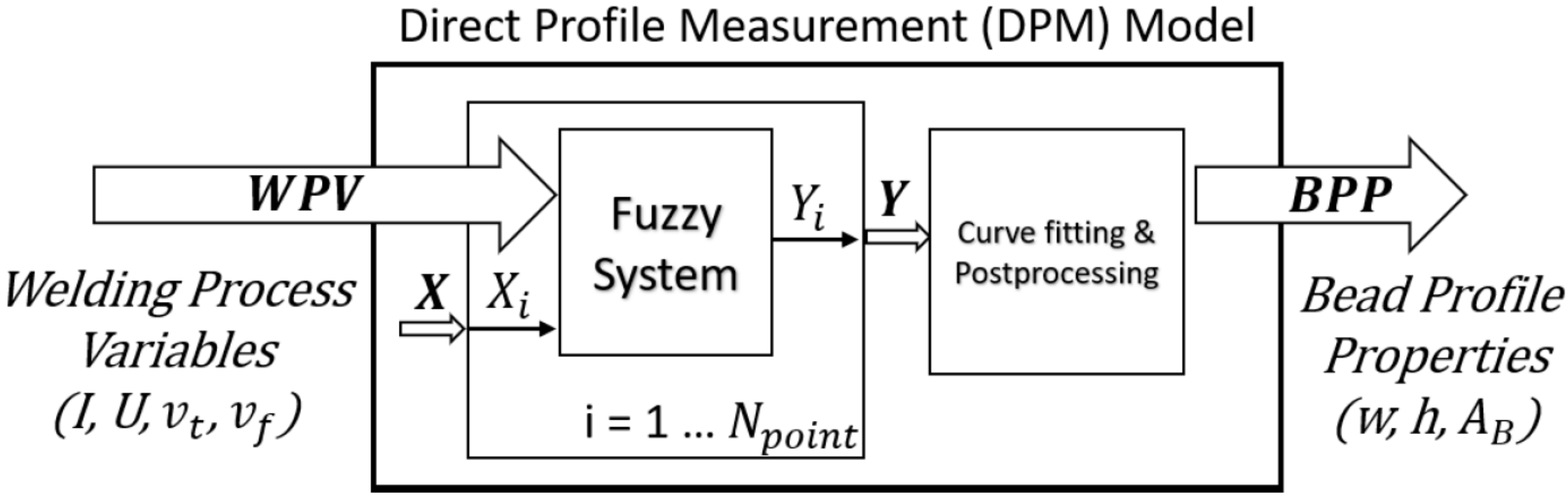


Robotic welding cell, where the experiments were carried out.

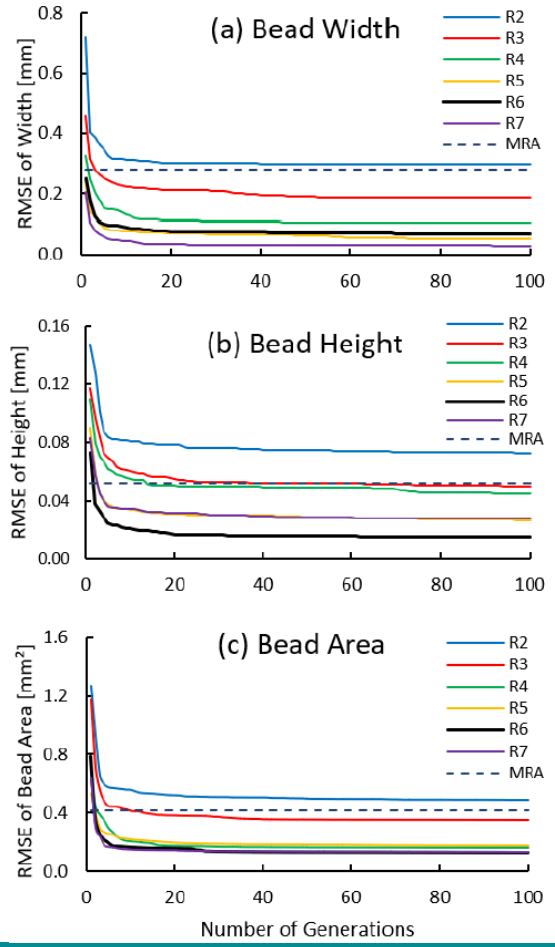
Bead Geometry Properties Model



Direct Profile Measurements Model



Results - Illustration



Evolutionary process with different number of rules for each Bead Profile Property (BPP) and comparison to the multiple regression analysis (MRA) model output (RMSE values).

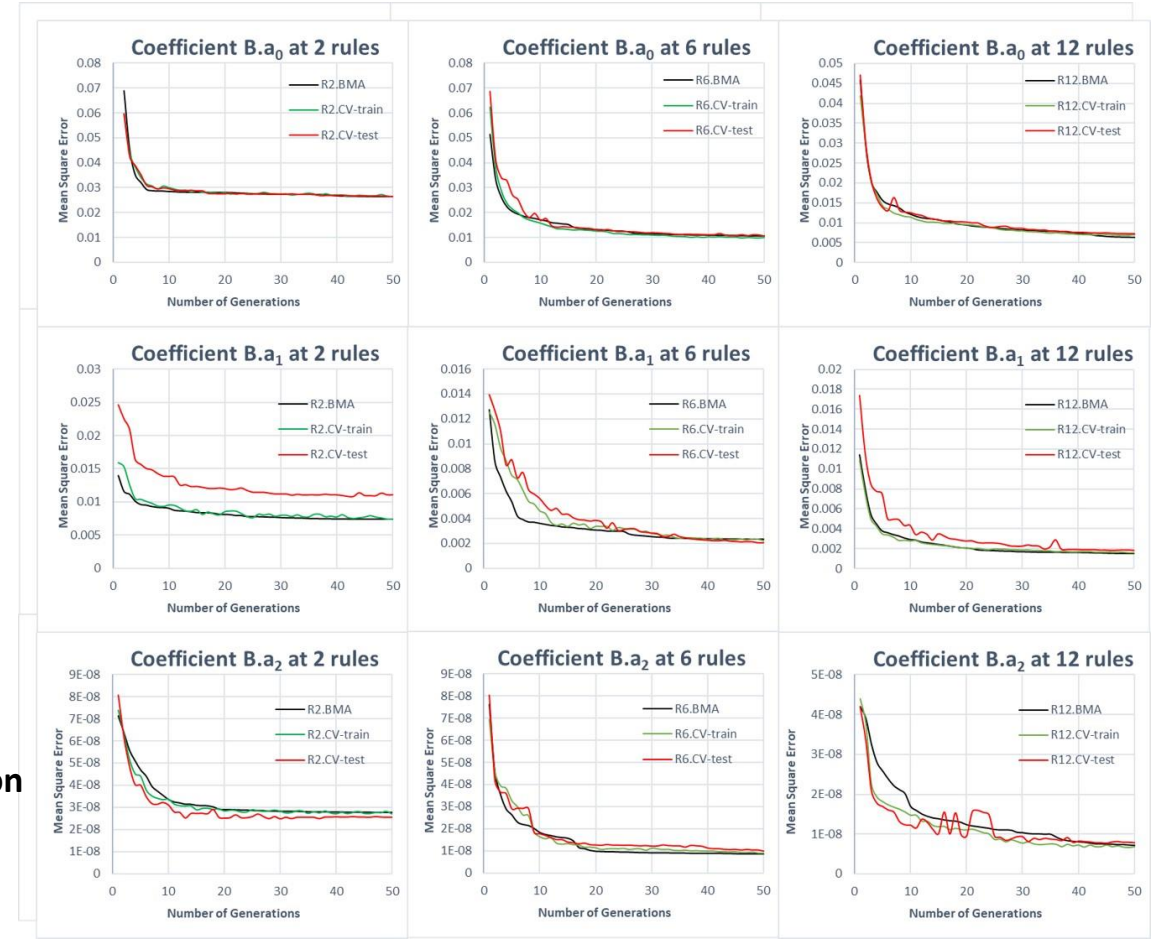
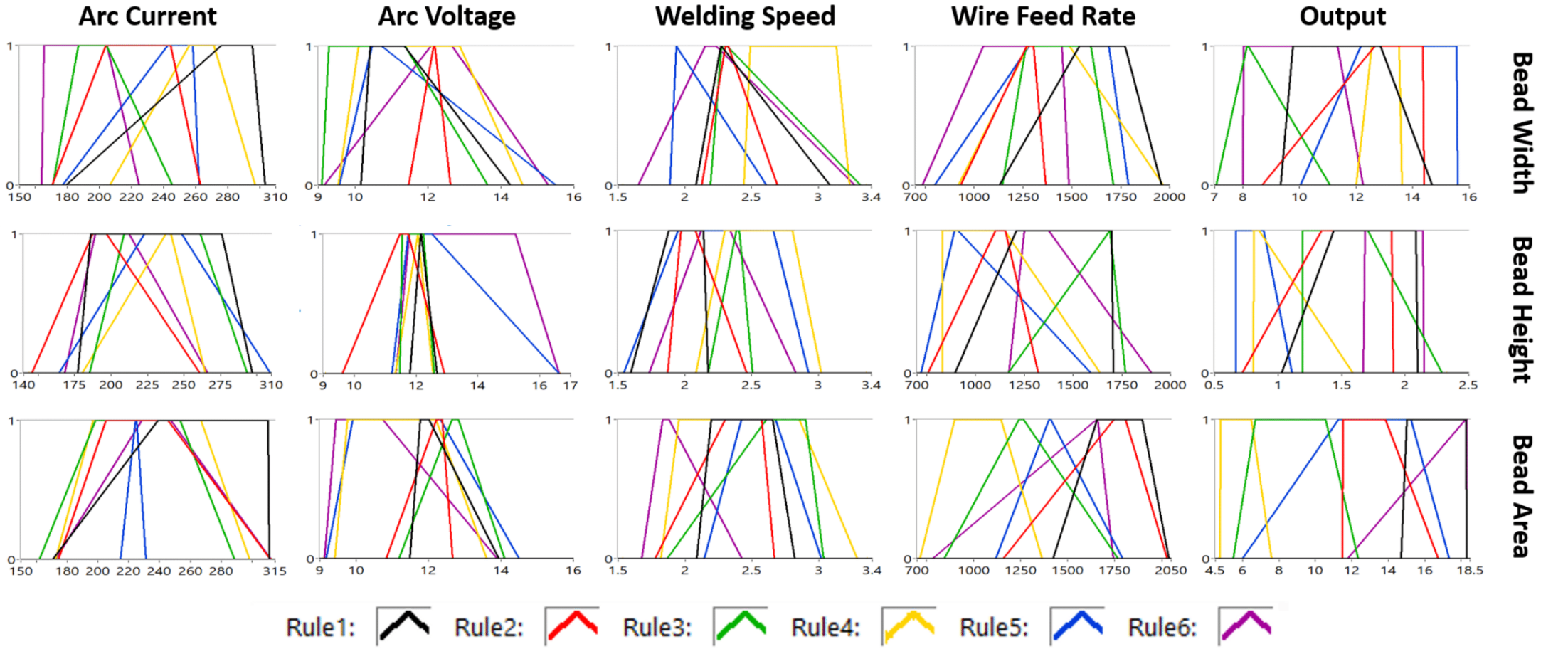


Illustration of the evolutionary process using BMA with and without cross-validation for $B.a_0$, $B.a_1$, $B.a_2$ variables at various number of rules

Obtained rule base



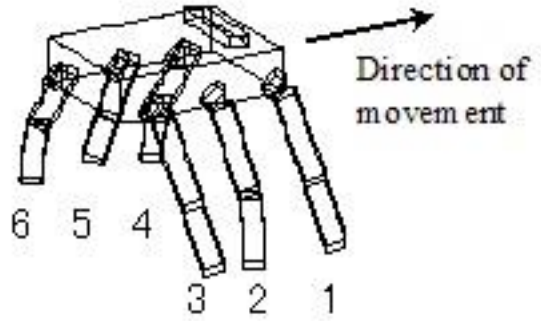
Evolutionary-based robot locomotion

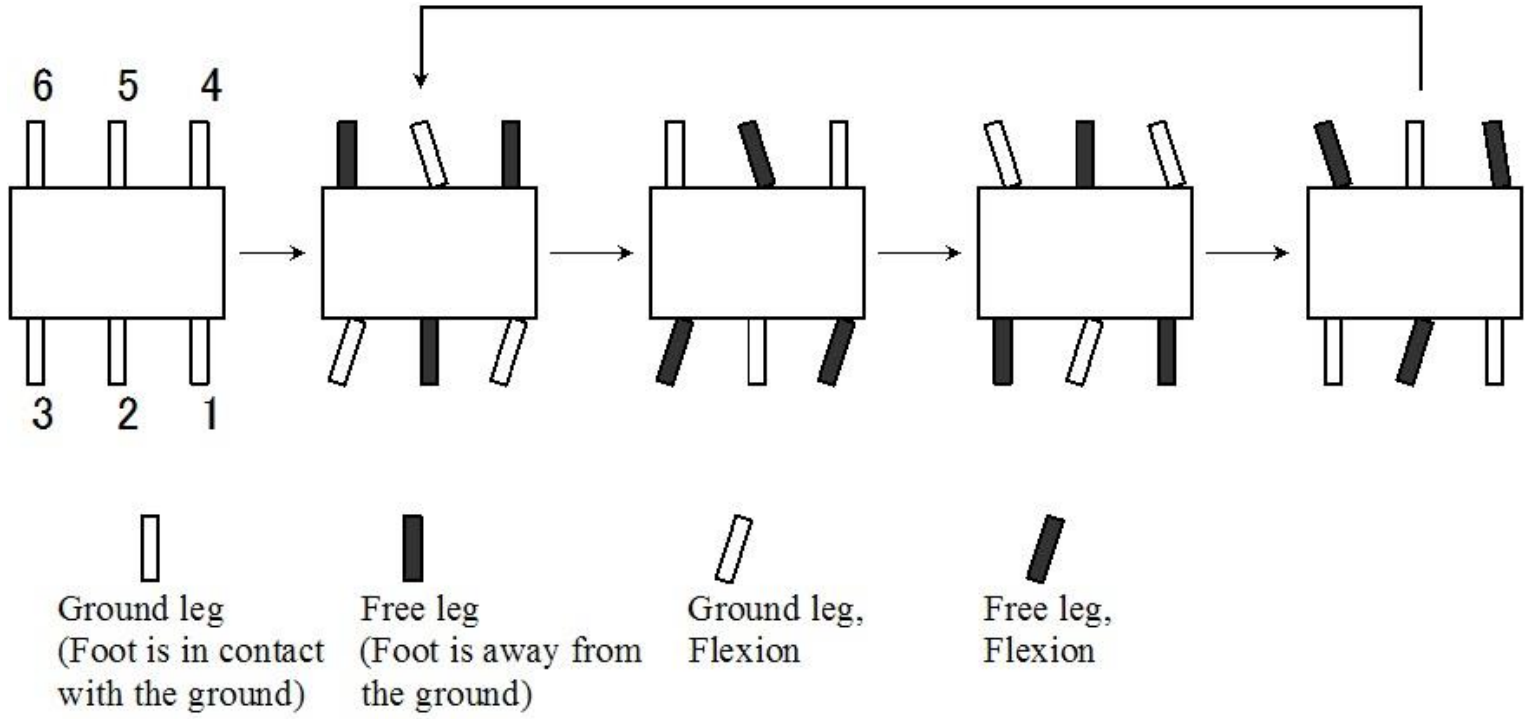
Introduction

- Motion generation is an essential task in robotics
- Solving it by hand is very difficult and time-consuming
- Evolutionary computation can be used to solve the problem
- Steady state genetic algorithm is proposed for motion generation
- The number of intermediate postures is also optimized
- The robot can adapt to different environments and maximize the moving distance
- A computer simulation environment is used before a real robot is applied

Robot model

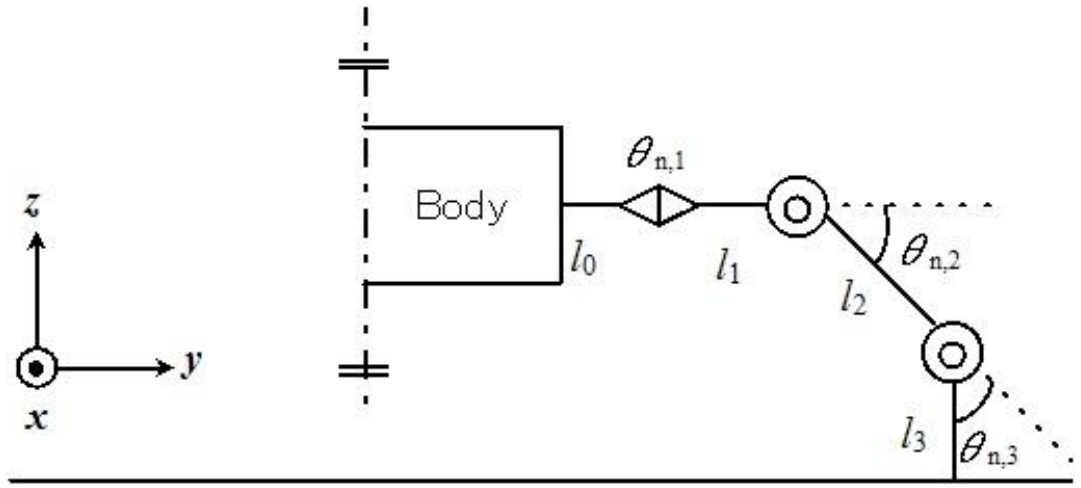
- A six-legged robot is applied
- The motion is based on tripod gait





Robot model

- The robot model is based on forward kinematics



$$\mathbf{P} = Tr(0, l_0, 0) \cdot Rot(y, \theta_{n,1}) \cdot Tr(0, l_1, 0) \cdot Rot(x, \theta_{n,2}) \cdot Tr(0, l_2, 0) \cdot Rot(x, \theta_{n,3}) \cdot Tr(0, l_3, 0) \cdot \mathbf{u} =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_{n,1}) & 0 & \sin(\theta_{n,1}) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_{n,1}) & 0 & \cos(\theta_{n,1}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

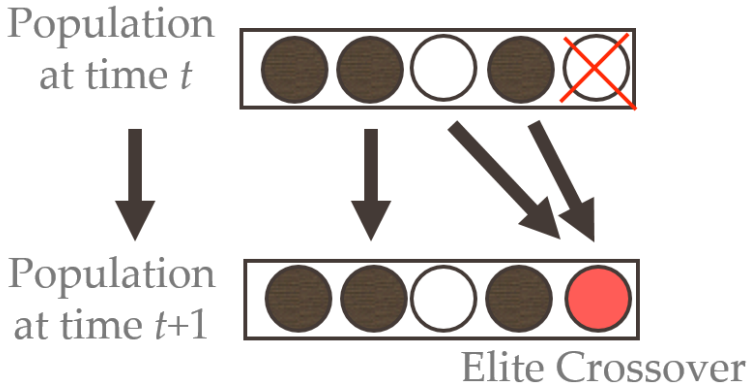
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_{n,2}) & -\sin(\theta_{n,2}) & 0 \\ 0 & \sin(\theta_{n,2}) & \cos(\theta_{n,2}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_{n,3}) & -\sin(\theta_{n,3}) & 0 \\ 0 & \sin(\theta_{n,3}) & \cos(\theta_{n,3}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Steady State Genetic Algorithm

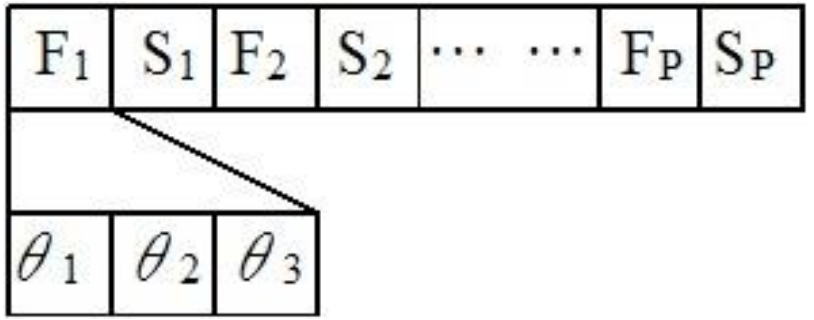
- SSGA is a model of continuous alternation of generations, where few individuals with low fitness values are replaced by new individuals produced by genetic manipulation



- In order to produce the best walking it is necessary to form intermediate positions
- It is difficult to predict the number of intermediate positions in dynamic or undetermined environment

Encoding method

- The chromosome contains the intermediate positions of the joint variables



- F represents the combination of three foots in the first phase of the tripod gait, while S means the combination in the second phase
- P stands for the number of postures
- Each posture has $2 \cdot 3$ parameters, so the total number of parameters to be optimized is $6 \cdot P$

Fitness calculation

- The fitness of each individual is calculated from the moving distance of the robot in one trial, the posture angle of the robot, and the penalty due to the height of the robot body:

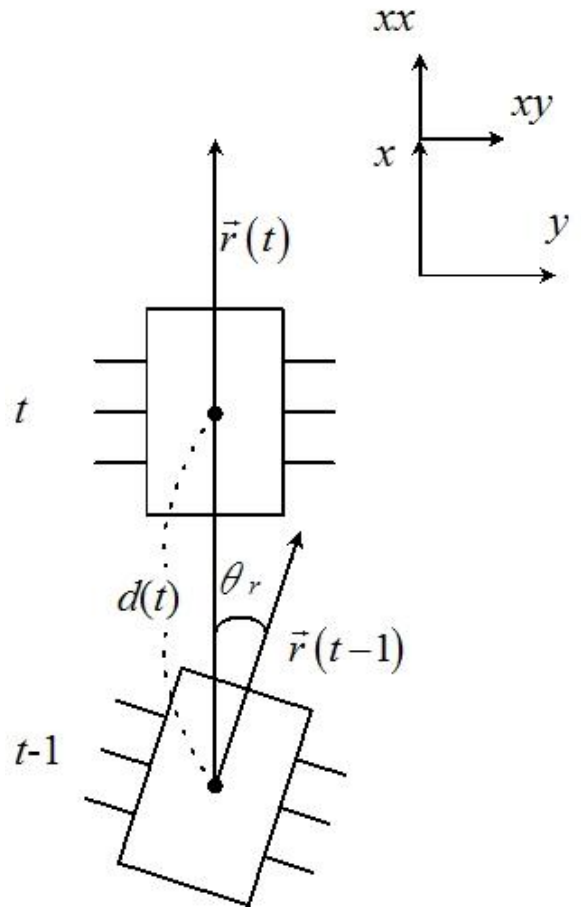
$$f_i = \eta^a \cdot f_a + \eta^q \cdot f_q + \eta^d \cdot f_d + \eta^p \cdot f_p$$

- $\eta^a, \eta^q, \eta^d, \eta^p$ are weights, f_a is related to the direction of movement of the robot, f_q is the inner product of the robot's posture and the moving direction vector, f_d is the moving distance $\big(d(t)\big)$, f_p is penalty for the posture of the robot:

$$f_p = \begin{cases} z & if\ z_{min} < z < z_{max} \\ -1{,}0 & otherwise \end{cases}$$

- For straight motion:

$$f_a = \exp(-\theta_r^2)$$



Crossover

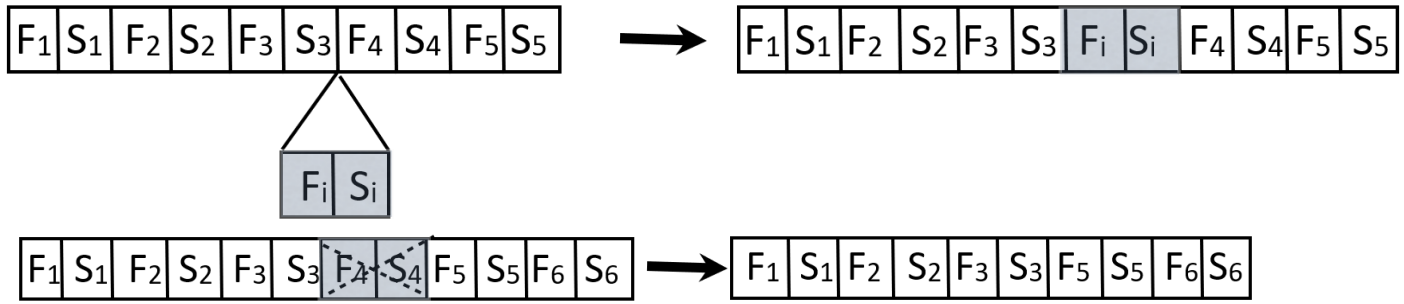
- The worst individual's genes are replaced by the crossed genes of the best individual and a random individual:

$$x_{worst,j} = \begin{cases} x_{rand,j} + \alpha \cdot q_j \cdot N(0,1) & if\ (r_j < r)\ \wedge (m_j < l_{rand}) \\ x_{best,j} + \beta \cdot q_j \cdot N(0,1) & otherwise \end{cases}$$

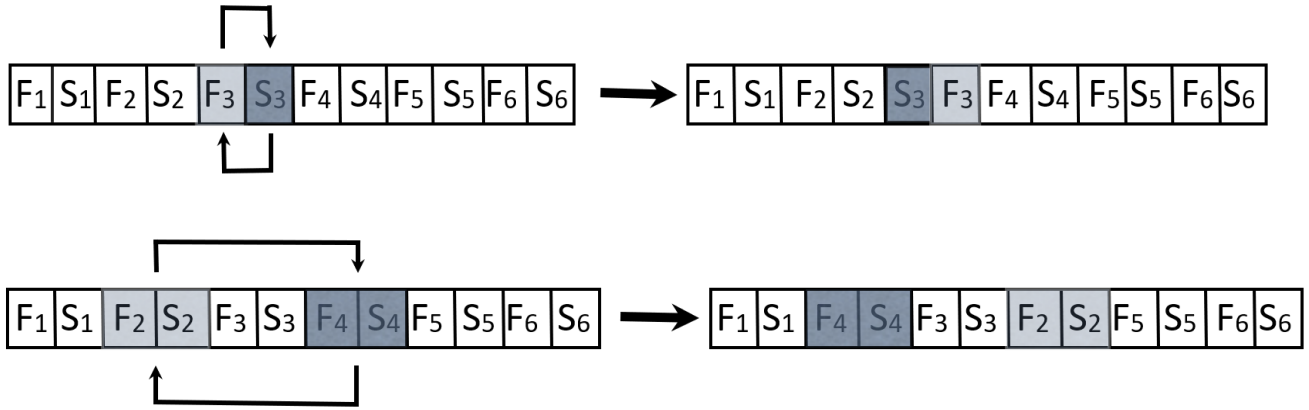
- $x_{worst,j}$, $x_{best,j}$, $x_{rand,j}$ are the j-th gene of the worst, the best and a randomly selected individual; l_{rand} is the number of postures of the random individual; m_j is the posture ID related to the j-th gene; r_j is a uniform random number between 0 and 1; r is a uniform random number between 0 and 0.5; $N(0,1)$ is a normal random number; q_j is the range of the j-th gene j; α and β are parameters

Mutation

- Insertion:
- Deletion:

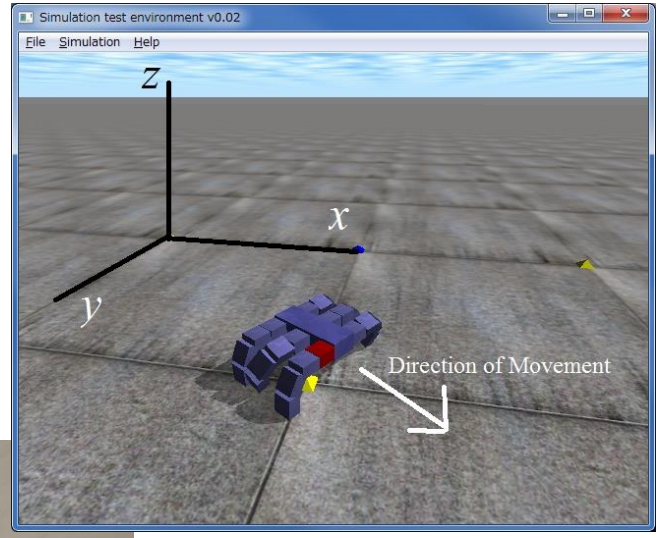


- Phase exchange:
- Order exchange:

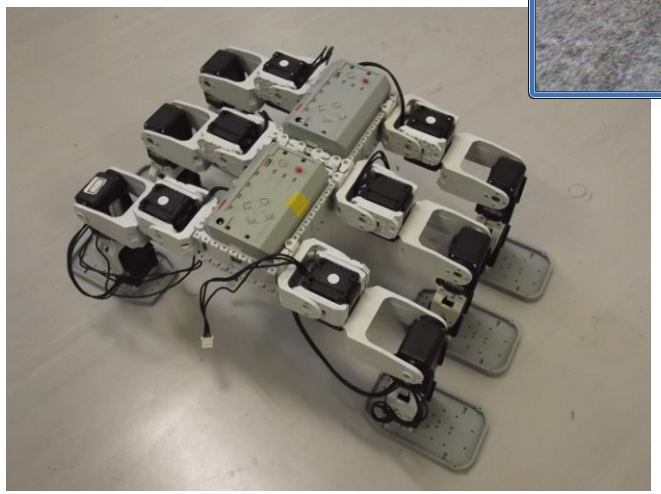


Experimental results

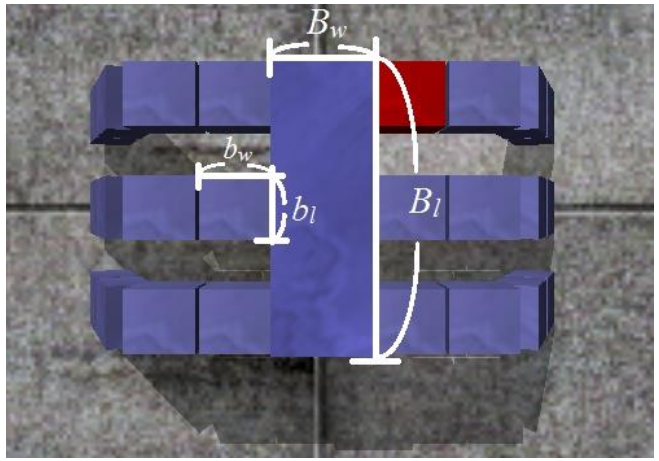
- First, computer simulations were performed using the Open Dynamics Engine (ODE)

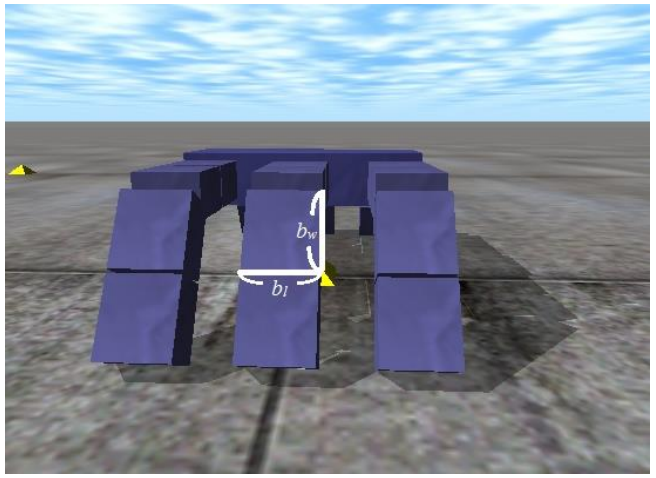


- Real experiment was also performed by Bioloid

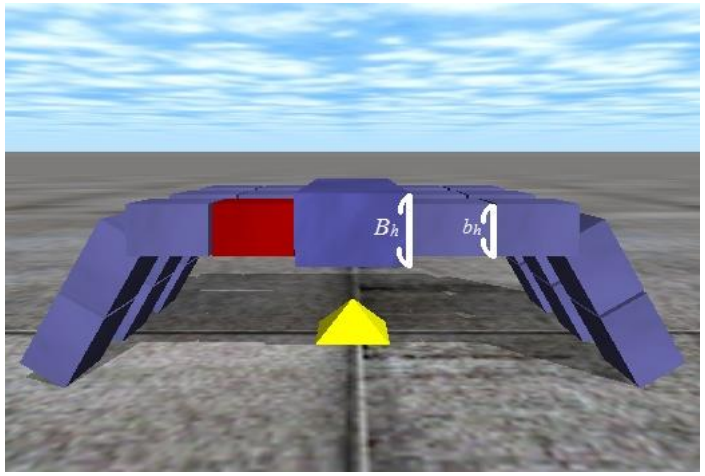


Parameters of the robot model





Parameter	Body (B)	Leg (b)
Mass B_m, b_m [kg]	0.700	0.055
Length B_l, b_l [m]	0.215	0.048
Width B_w, b_w [m]	0.075	0.053
Height B_h, b_h [m]	0.048	0.034
Max Torque b_τ [N · m]	—	1.62



Parameters of the robot model

Angle parameters

Parameter	1st joint	2nd joint	3rd joint
range[°]	$-30 \le \theta_1 \le 30$	$45 \le \theta_2 \le 90$	$0 \le \theta_3 \le 60$
initial [°]	0	60	0

Fitness parameters

Parameter	Value	
	Str.	Turn
η^a	1	10
η^d	10	0.5
η^p	1	1
η^q	1	
z_{min}	0.1	0.1
z_{max}	0.2	0.2

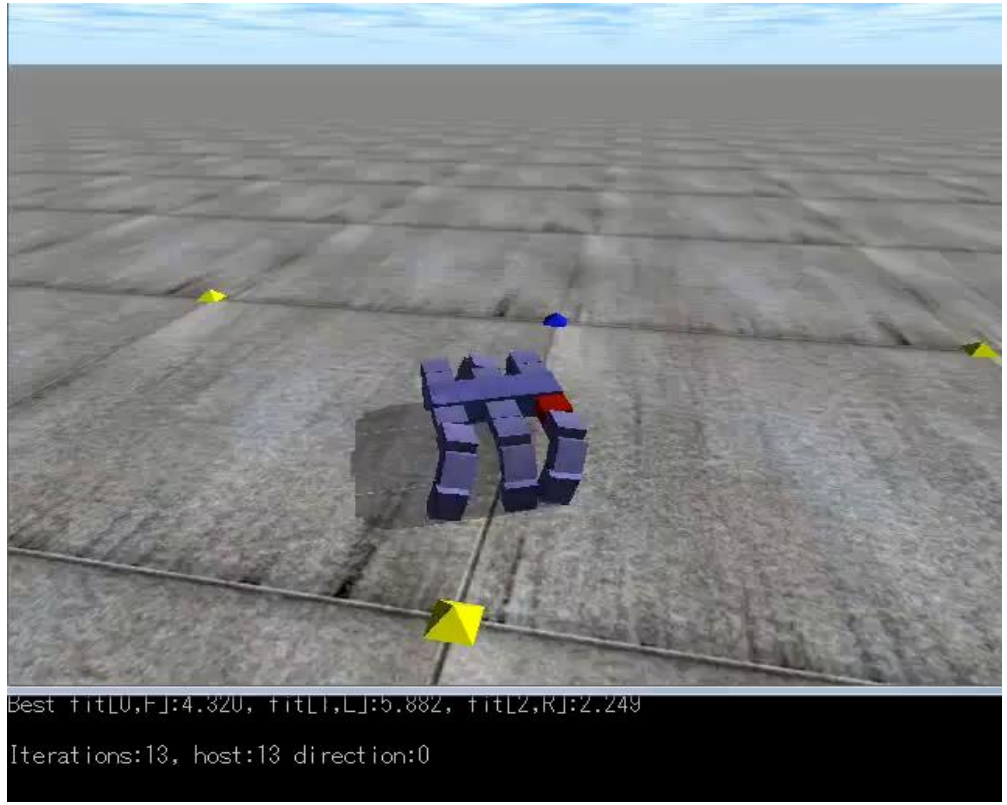
ODE parameters

Parameter	Value
Time of 1 Trial [sec.]	1.38
Environment ERP	0.9
Environment CFM	0.001
Depth of Ground Surface	0.001
Contact ERP	0.9
Contact CFM	0.001

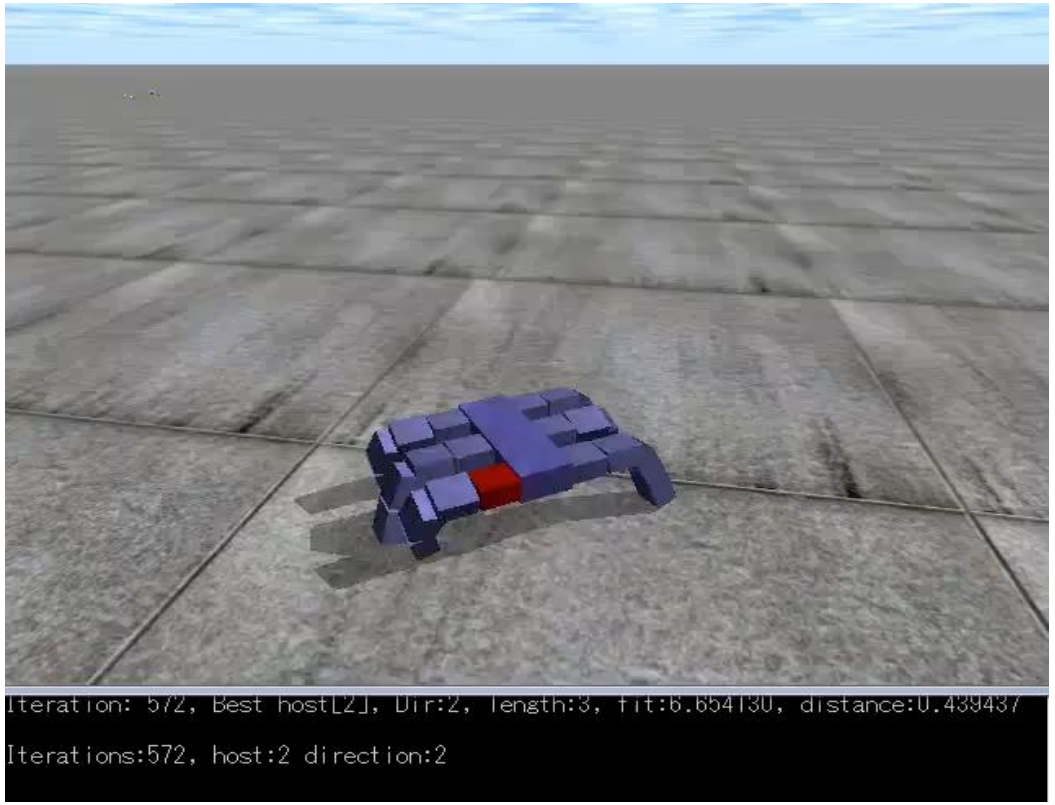
SSGA parameters

Parameter	Value
Number of Individuals	20
Max Number of Postures	10
Trial Number (time)	500
Coefficient α	0.2
Coefficient β	0.3
Probability of Insertion Mutation	0.15
Probability of Deletion Mutation	0.15
Probability of Phase Exchange Mutation	0.1
Probability of Order Exchange Mutation	0.1

Simulation results

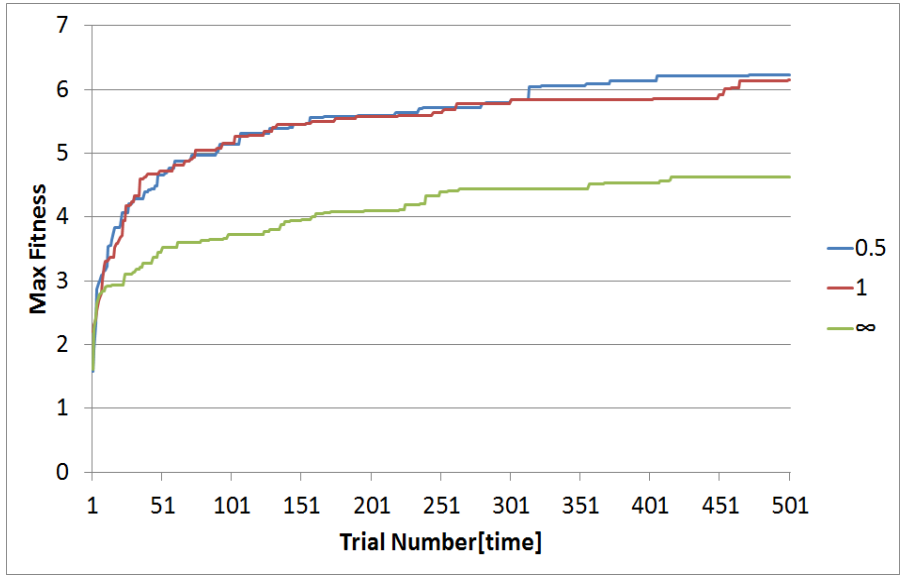


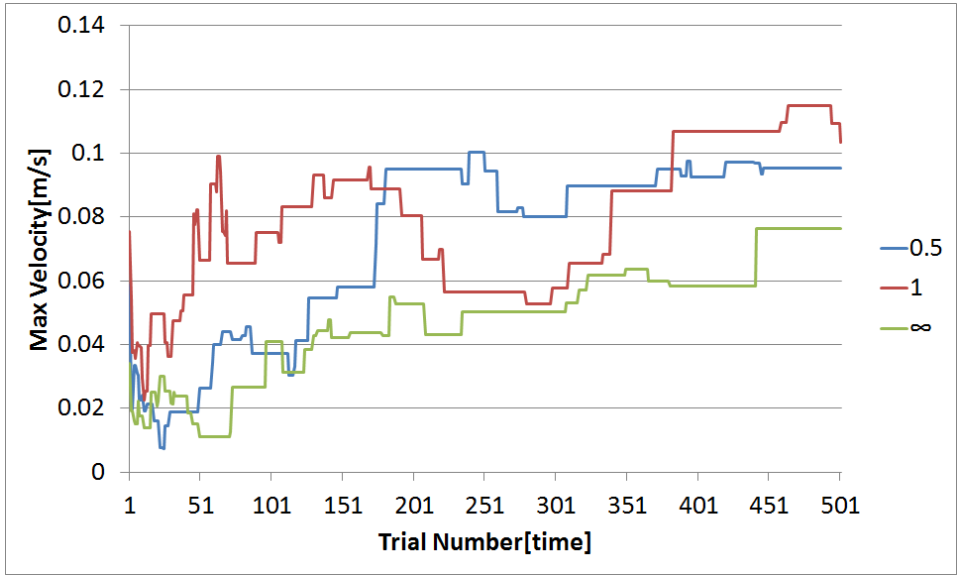
After training



Simulation results

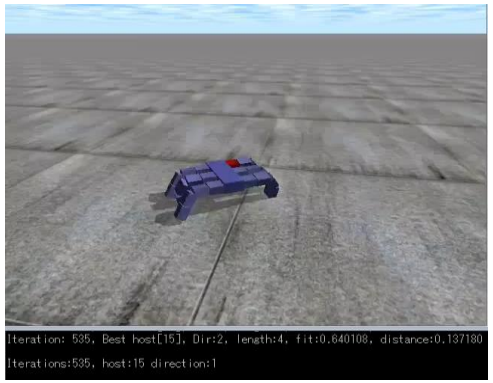
- Simulations in different environments
- Different friction coefficients (μ): 0.5; 1; ∞
- The average of maximum fitness values and maximum velocities during the evolutionary process based on ten simulations:

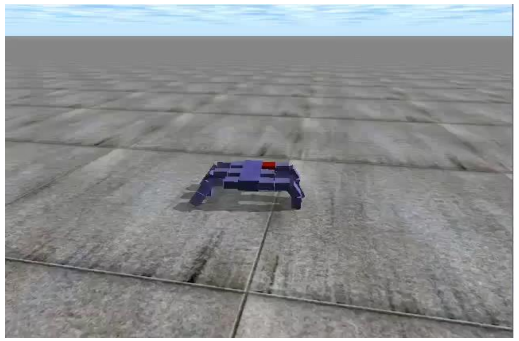




Simulation results

- μ=0.5 μ=1 μ=∞





Best fitness and number of postures:

7.24 (4) 6.65 (3) 5.34 (3)

- If the friction coefficient is small, there is a risk of tipping and slipping, however, the movement distance and the velocity increase
- If the coefficient is large, the robot can walk without tipping, however the increase of the movement distance and the velocity is less

Simulation results

- $\mu = 0.5$

F_1		S_1		F_2		S_2		F_3		S_3		F_4		S_4	
θ_1	29.998	θ_1	-29.993	θ_1	-14.127	θ_1	9.263	θ_1	-29.998	θ_1	22.837	θ_1	14.018	θ_1	-21.360
θ_2	66.372	θ_2	64.107	θ_2	89.994	θ_2	45.010	θ_2	45.001	θ_2	86.387	θ_2	51.806	θ_2	45.001
θ_3	56.638	θ_3	41.614	θ_3	9.963	θ_3	0.001	θ_3	5.726	θ_3	7.889	θ_3	48.093	θ_3	0.010

- $\mu = 1$

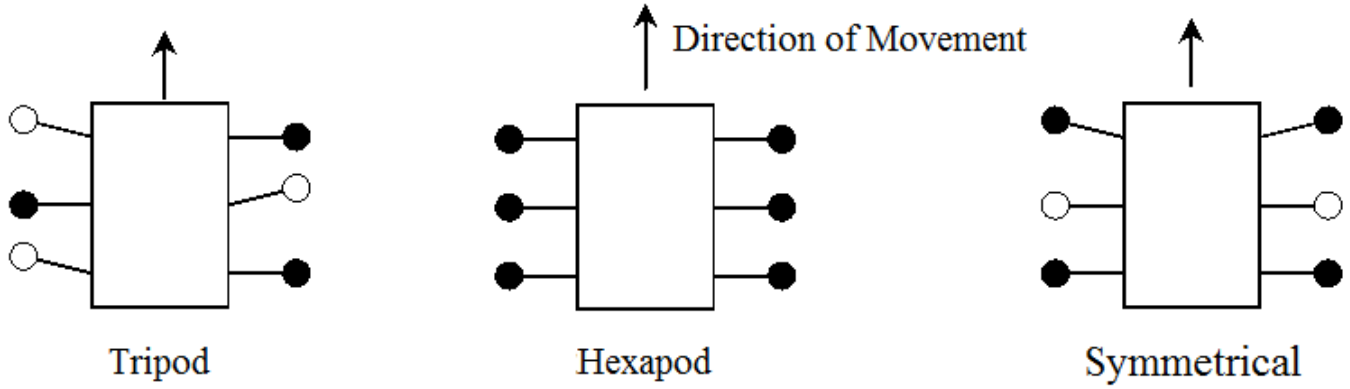
F_1		S_1		F_2		S_2		F_3		S_3	
θ_1	-29.991	θ_1	29.997	θ_1	-29.996	θ_1	-29.993	θ_1	29.993	θ_1	-19.218
θ_2	59.063	θ_2	45.962	θ_2	59.157	θ_2	79.540	θ_2	70.724	θ_2	52.672
θ_3	0.002	θ_3	6.741	θ_3	59.999	θ_3	0.008	θ_3	0.004	θ_3	25.214

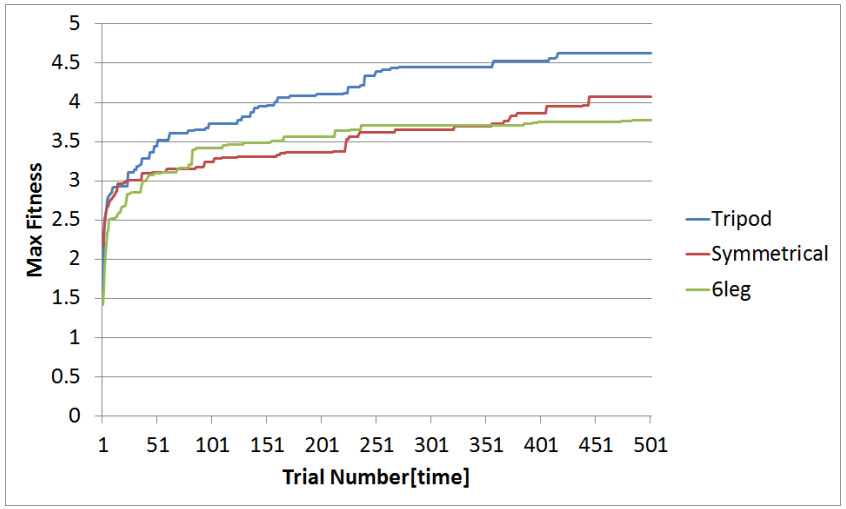
- $\mu = \infty$

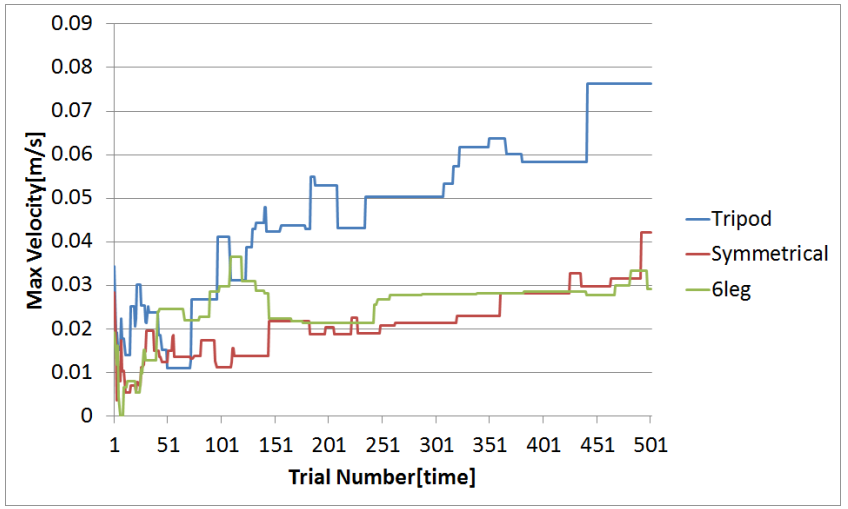
F_1		S_1		F_2		S_2		F_3		S_3	
θ_1	-29.991	θ_1	-12.980	θ_1	18.906	θ_1	-29.993	θ_1	-17.989	θ_1	-29.992
θ_2	66.638	θ_2	67.009	θ_2	45.010	θ_2	63.461	θ_2	77.062	θ_2	45.004
θ_3	16.112	θ_3	0.004	θ_3	59.999	θ_3	0.000	θ_3	0.002	θ_3	0.005

Different gait types are also compared

The average of maximum fitness values and maximum velocities during the evolutionary process based on ten simulations (μ=∞):

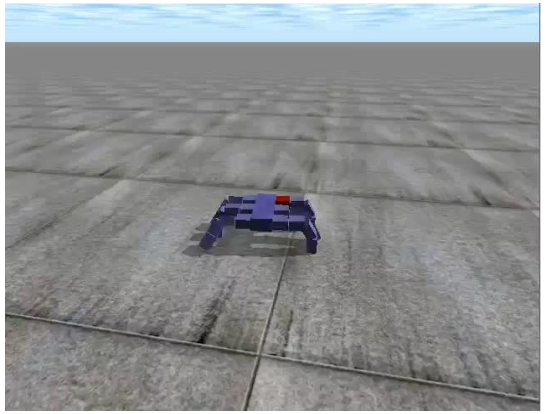


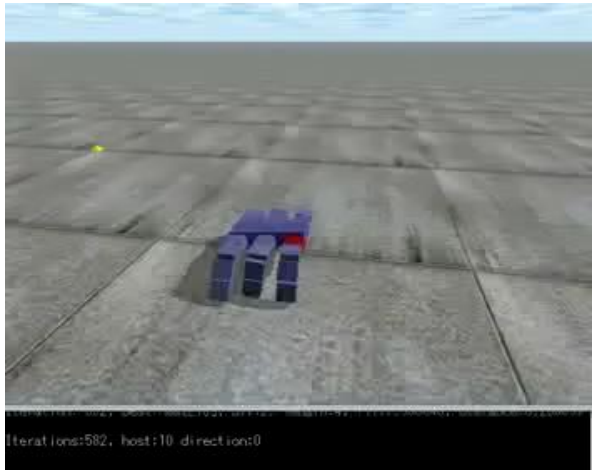


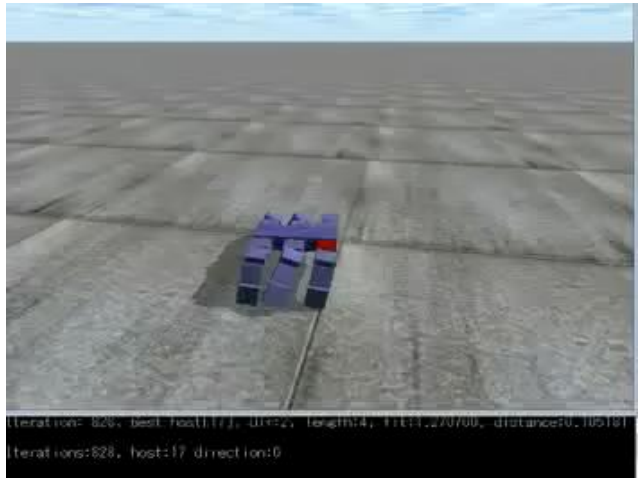


Different gait types are also compared

tripod | symmetrical | hexapod







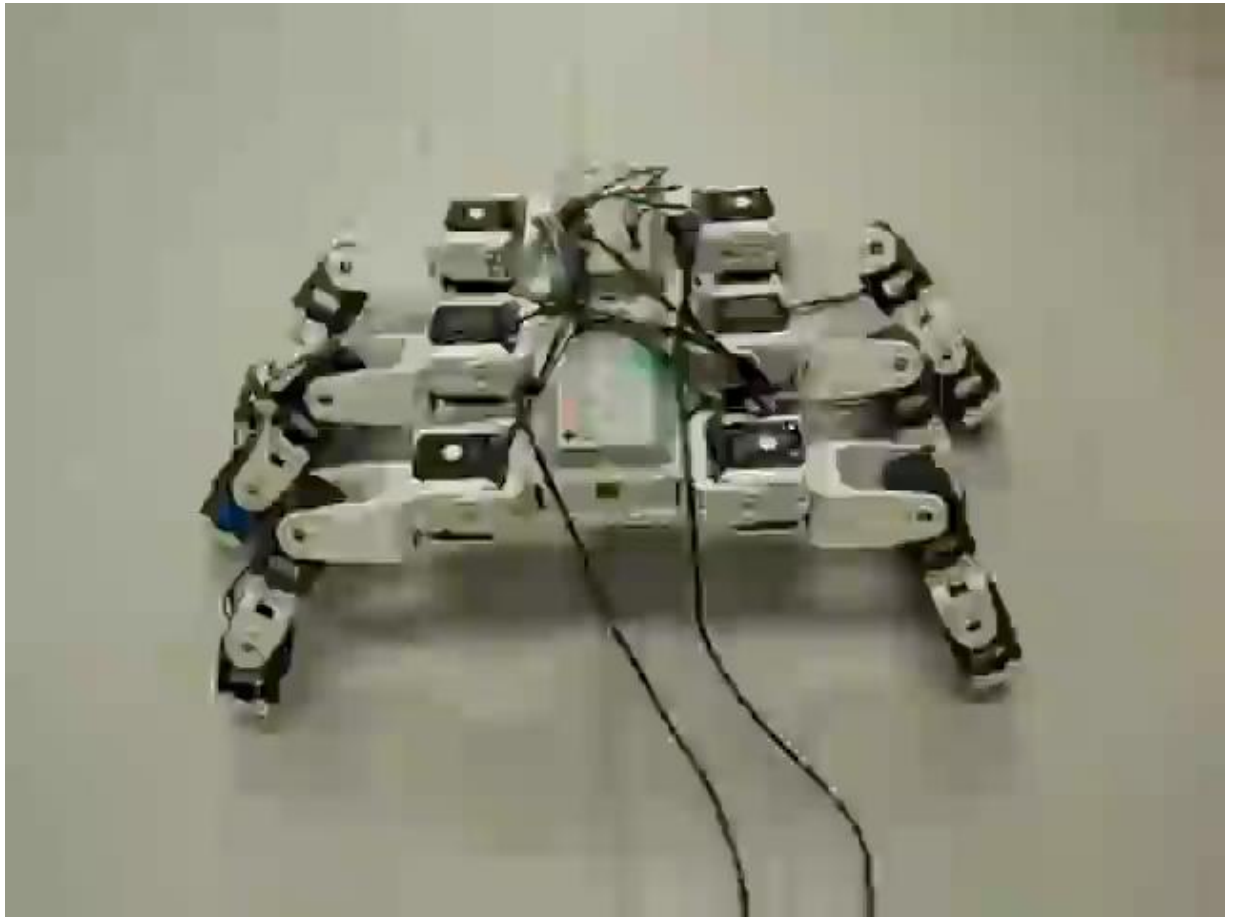
- Best fitness:

  5.34 | 5.05 | 5.03
- Tripod gait has the maximum fitness and velocity.

Bioloid

- A ready-to-assemble type robot manufactured by ROBOTIS
- The components of robot kits include the main unit of micro-controller, actuators, sensor unit, frames, connection cables, bolts, and nuts
- The robot can perform wireless communication by ZigBee through RS-232C with host computers and other robots
- It can be used for edutainment, serving as a robot partner, illustrating path planning problem etc.

Experimental result

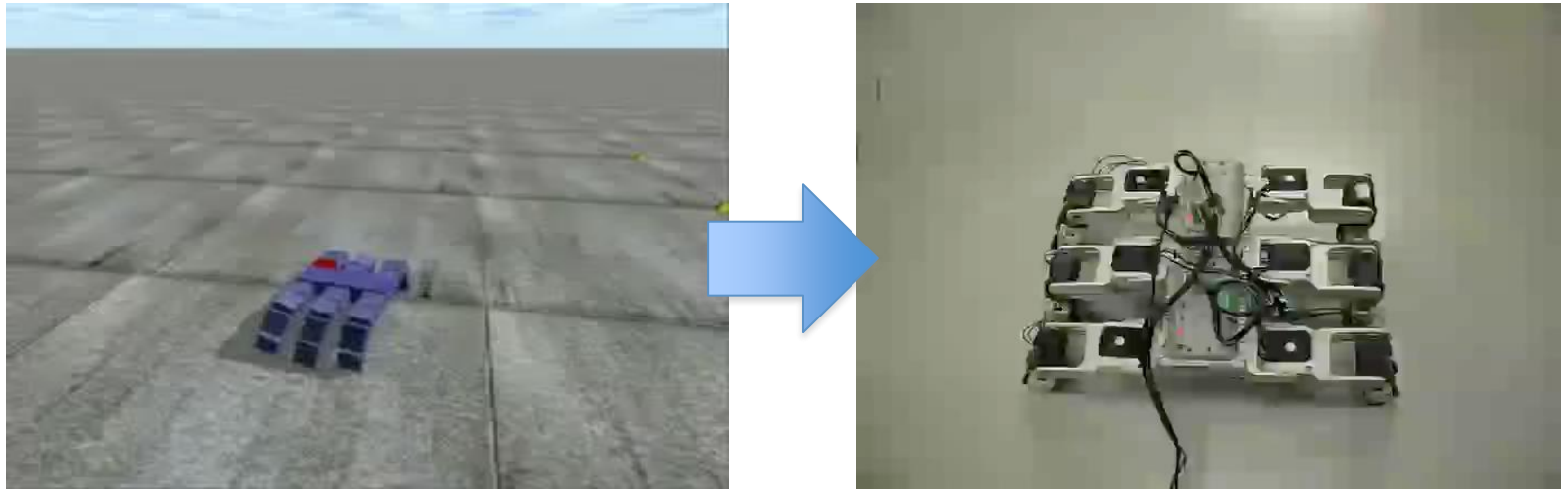


Conclusions

- The motion generation of a six-legged robot was discussed
- SSGA is applied for solving the problem
- ODE computer simulation environment was used to test the method with different friction types and different gait types
- After obtaining the optimal result in computer simulation it is tried in a real robot as well

Robot model

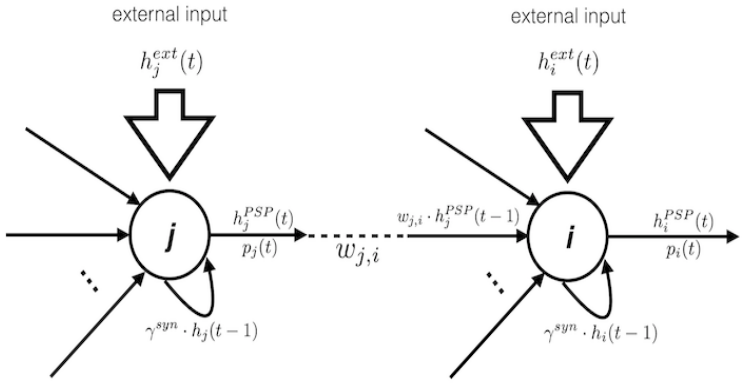
- Generated in simulation → applied to a real robot → there are some shortcomings during simulation (angular limits, motor speed, etc.) → generation of motion pattern (rhythm) by SNN in simulation and with the real robot

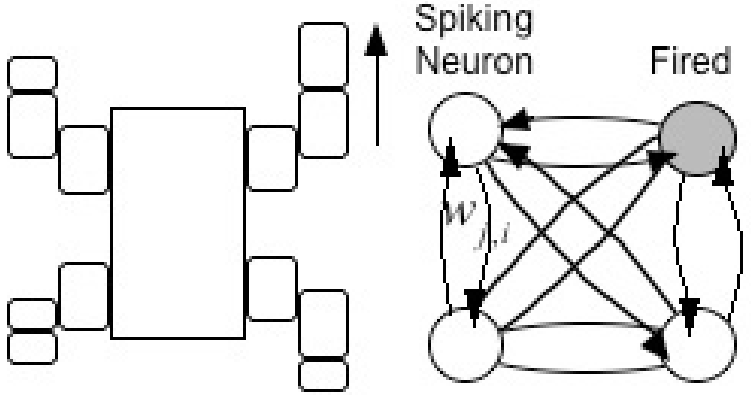


	F_1	S_1	F_2	S_2	F_3	S_3
θ_1[°]	-29.991	29.997	-29.996	-29.993	29.993	-19.218
θ_2[°]	59.063	45.962	59.157	79.540	70.724	52.672
θ_3[°]	0.002	6.741	59.999	0.008	0.004	25.214

Spiking Neural Networks

Simple Spike response model





Spiking neural network for robot locomotion

$$h_i(t) = \tanh\left(h_i^{syn}(t) + h_i^{ext}(t) + h_i^{ref}(t)\right)$$

$h_i(t)$: internal state (membrane potential) of neuron *i*

t: discrete time

$h_i^{syn}(t)$: weighted outputs from other neurons and reduced value of internal state from previous step

$h_i^{ext}(t)$: input to neuron *i* from the environment

$h_i^{ref}(t)$: represents the refractory phase in the neuron

Spiking Neural Networks

$$h_i^{syn}(t) = \gamma^{syn} \cdot h_i(t-1) + \sum_{j=1, j \neq i}^{N} w_{j,i} \cdot h_j^{PSP}(t-1)$$

- γ^{syn}: reduction factor of h_i
- $w_{j,i}$: weight between neuron j and neuron i
- $h_j^{PSP}(t)$: postsynaptic potential (PSP) from neuron j at time t
- N: number of neurons

$$h_i^{ref} = \begin{cases} \gamma^{ref} \cdot h_i^{ref}(t-1) - R & if\ p_i(t-1) = 1 \\ \gamma^{ref} \cdot h_i^{ref}(t-1) & otherwise \end{cases}$$

- R: constant, $R > 0$
- γ^{ref}: reduction factor of h_i^{ref}; $0 \leq \gamma^{ref} \leq 1$
- $p_i\ (t)$: output impulse of neuron i at time t

$$h_i^{ext}(t) = \max(A_i \cdot \sin(\omega_i \cdot t + \phi_i), 0)$$

- A_i: amplitude, $A_i > 0$
- ω_i: frequency, $\omega_i > 0$
- ϕ_i: phase

Spiking Neural Networks

$$p_i(t) = \begin{cases} 1 & if \;\; h_i(t) \geq \theta \\ 0 & otherwise \end{cases}$$

θ: firing threshold

$$h_i^{PSP}(t) = \exp\left(-\frac{t - t_i^{(f)}}{\tau_i}\right)$$

- $t_i^{(f)}$: the last firing time of the neuron *i*
- τ_i: length of firing effect for the associated neurons

Training algorithm

- Learning weights between neurons ($w_{i,j}$) by Hebbian learning
- $\tau_i, A_i, \omega_i, \varphi_i$ SNN parameter optimization with (1+1) evolution strategy

$x_{t+1} = x_t + N(0, \sigma)$

  - x_t: parent
  - x_{t+1}: offspring
  - $N(0, \sigma)$: normally distributed random number with 0 mean and standard deviation σ
  - The chromosomes of the parent and the offspring contain the values of $\tau_i, A_i, \omega_i, \varphi_i$ for each neuron
  - There is only one variance parameter σ for each component within the chromosome
  - The parameter σ varies adaptively following the 1:5 rule
  - If more than one out of five mutations are successful (the offspring is a better fit than the parent) then the variance σ is multiplied by a parameter α ($\alpha > 1$).
  - If the success rate is less than 20% then σ is divided by α.

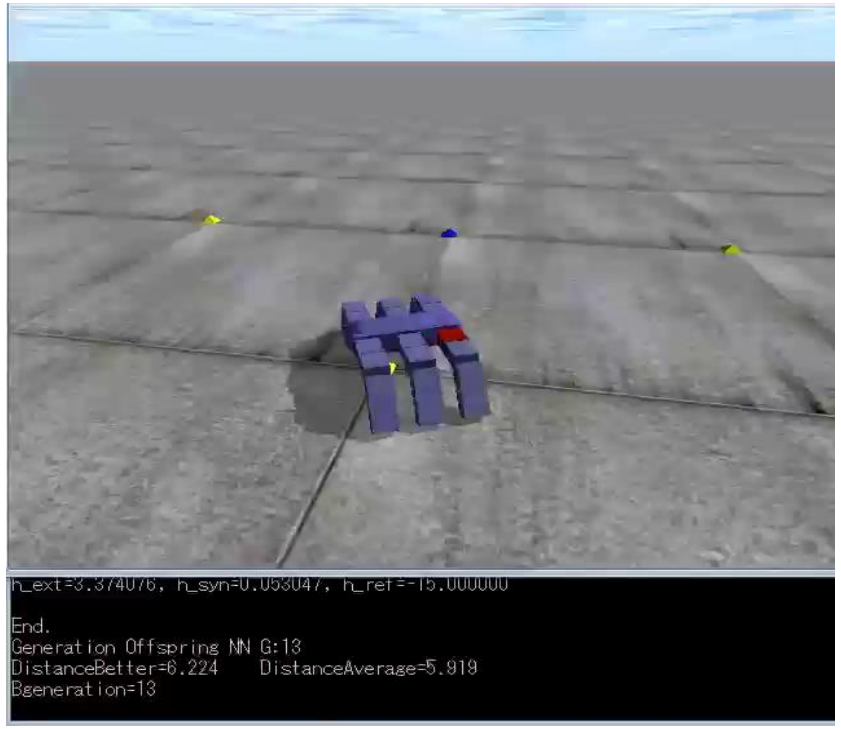
Training algorithm

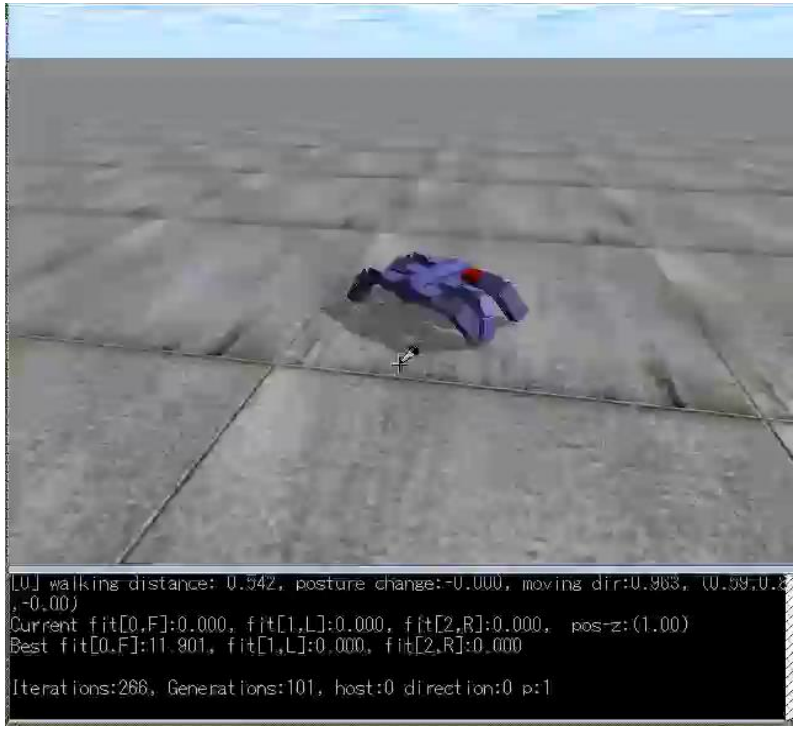
Algorithm 1 Training Algorithm

NumberOfSuccessfulMutations $\leftarrow 0$
FitnessAverage $\leftarrow 0$
MutationVariance $\leftarrow \sigma$
for $gen = 1, 2 \ldots N_{gen}$ **do**
    Parent $\leftarrow$ *current values of* $\tau_i, A_i, \omega_i, \phi_i$
    FitnessParent $\leftarrow$ *FitnessCalculation(Parent)*
    Offspring $\leftarrow$ *Mutate(Parent)*
    FitnessOffspring $\leftarrow$ *FitnessCalculation(Offspring)*
    if *FitnessParent* $\geq$ *FitnessOffspring* **then**
        FitnessBetter $\leftarrow$ *FitnessParent*
        Keep Parent as the solution
    else
        FitnessBetter $\leftarrow$ *FitnessOffspring*
        Keep Offspring as the solution
        NumberOfSuccessfulMutations++
    if *FitnessBetter* $>$ *FitnessAverage* **then**
        $w_{j,i}^{new} = w_{j,i}^{old} + \xi \cdot h_j^{PSP}(t-1) \cdot h_i^{PSP}(t)$
    else
        $w_{j,i}^{new} = w_{j,i}^{old} - \xi \cdot h_j^{PSP}(t-1) \cdot h_i^{PSP}(t)$
    FitnessAverage $\leftarrow \frac{FitnessAverage \cdot gen + FitnessBetter}{gen+1}$
    if *gen* MOD $5 = 0$ **then**
        if *NumberOfSuccessfulMutations* ≥ 1 **then**
            MutationVariance $\leftarrow$ *MutationVariance* $\cdot \alpha$
        else
            MutationVariance $\leftarrow$ *MutationVariance* $: \alpha$
        NumberOfSuccessfulMutations $\leftarrow 0$

Simulation results – ODE (Open Dynamics Engine)

- Simulation target:
  - Increase robot speed (distance traveled)
  - To obtain the straight forward type of movement
- Simulation conditions:
  - Three friction coefficients in each simulation $\mu(0.5;1;\infty)$





Parameters

SNN and algorithm parameters

Parameter	Value
Number of Generations (N_{gen})	200
Γ^{ref}	0.2
Γ^{syn}	0.05
Θ	0.995
R	15
τ_i interval	$\tau_i>0$
A_i interval	$A_i>0$
ω_i interval	$\omega_i>0$
time interval	0<t<30
σ	1
α	1.22
ξ	1.125

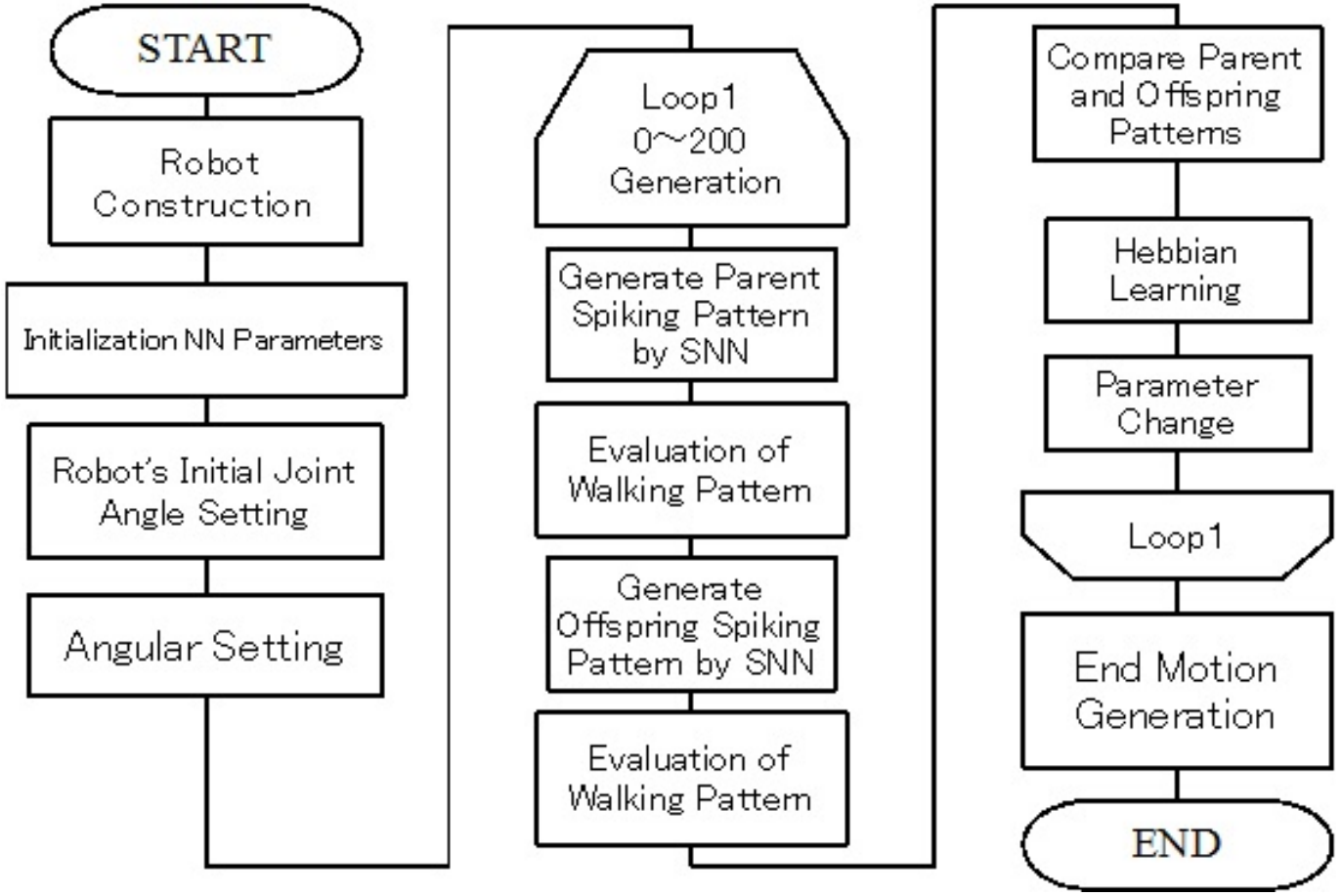
Fitness parameters

Parameter	Value
η_a	3.0
η_d	10.0
η_q	3.0
η_p	1.0
z_{min}	0.1
z_{max}	0.2

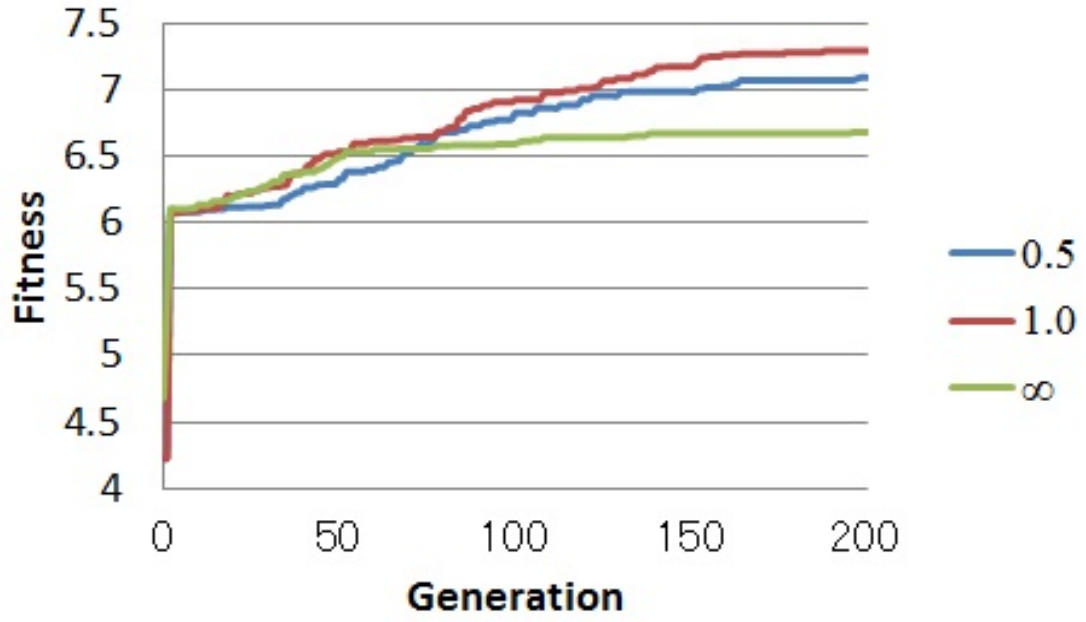
ODE parameters

Parameters	Value
Test time (s)	2.3
Environmental ERP	0.9
Environmental CFM	0.001
Depth of ground level	0.001
Contact ERP	0.9
Contact CFM	0.001

Simulation flowchart

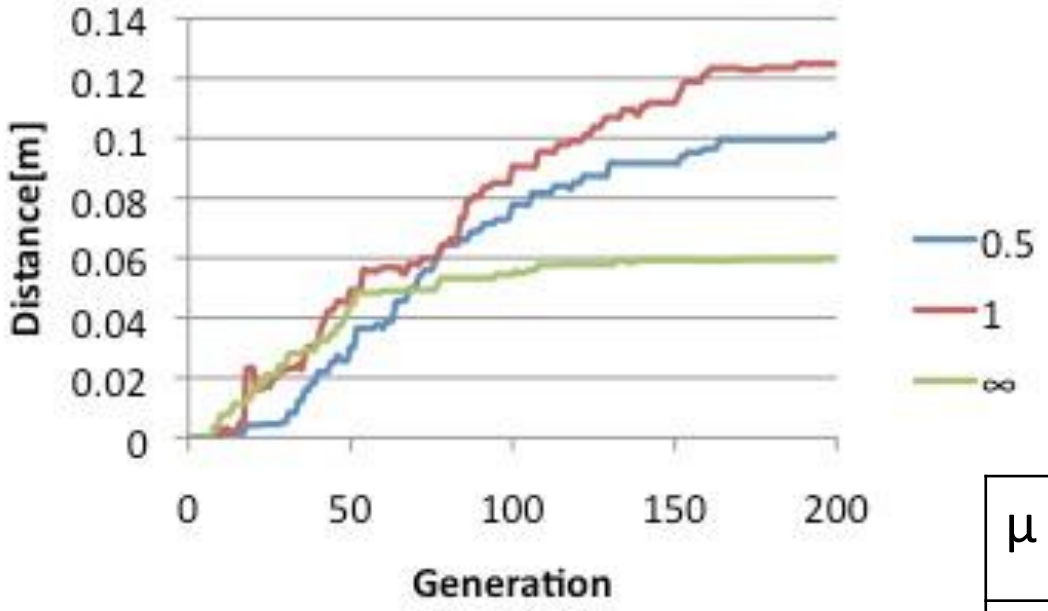


Evolutionary process – 4-legged robot



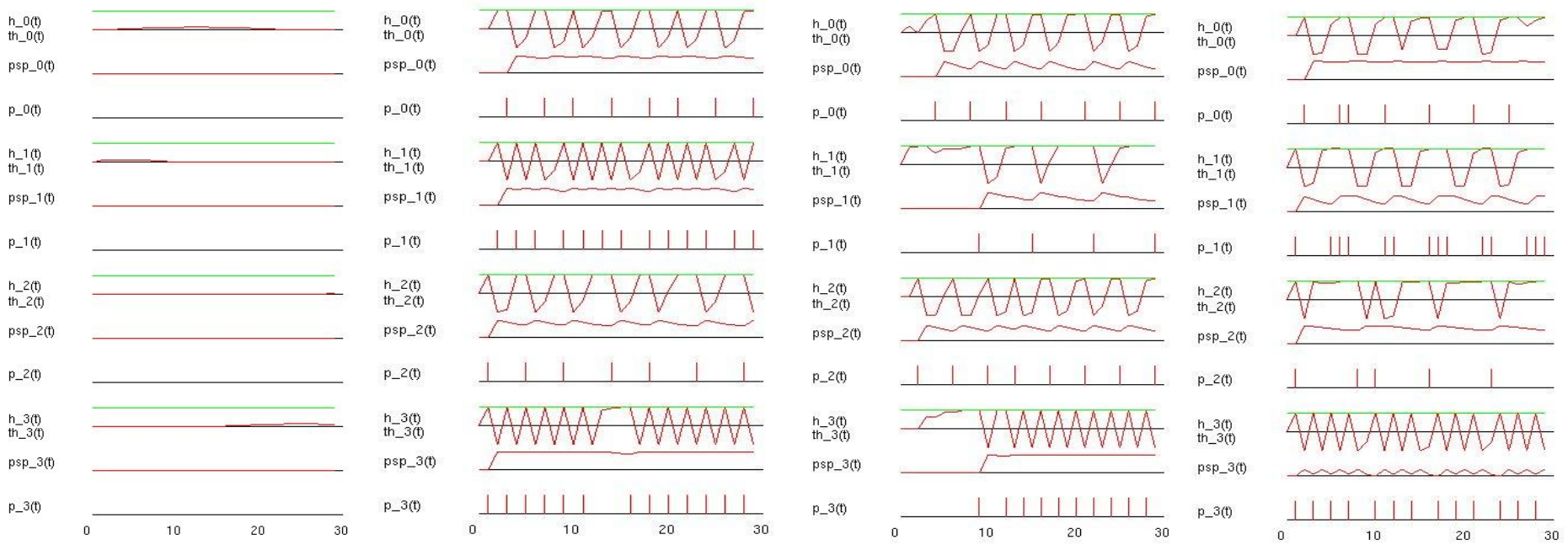
μ	0.5	1.0	∞
Average fitness	7.088	7.290	6.679
Maximum fitness	7.289	7.708	6.871

Distance



μ	0.5	1.0	∞
Average distance [m]	0.10131	0.12476	0.05973
Maximum distance [m]	0.12483	0.16081	0.07687

Spike functions



Initial μ=0.5 μ=1.0 μ=∞

Neuron parameters

Initial

i-th neuron (*i-th* leg)	1	2	3	4
T_i	6	8	10	12
A_i	0.15	0.15	0.15	0.15
Ω_i	0.1257	0.1257	0.1257	0.1257
Φ_i	0	1.5707	3.1415	4.7123

μ=1.0

i-th neuron (*i-th* leg)	1	2	3	4
τ_i	5	8	6	21
A_i	7.34	2.15	6.85	6.39
ω_i	7.78	0.90	4.65	6.37
ϕ_i	1.58	-0.03	11.55	18.33

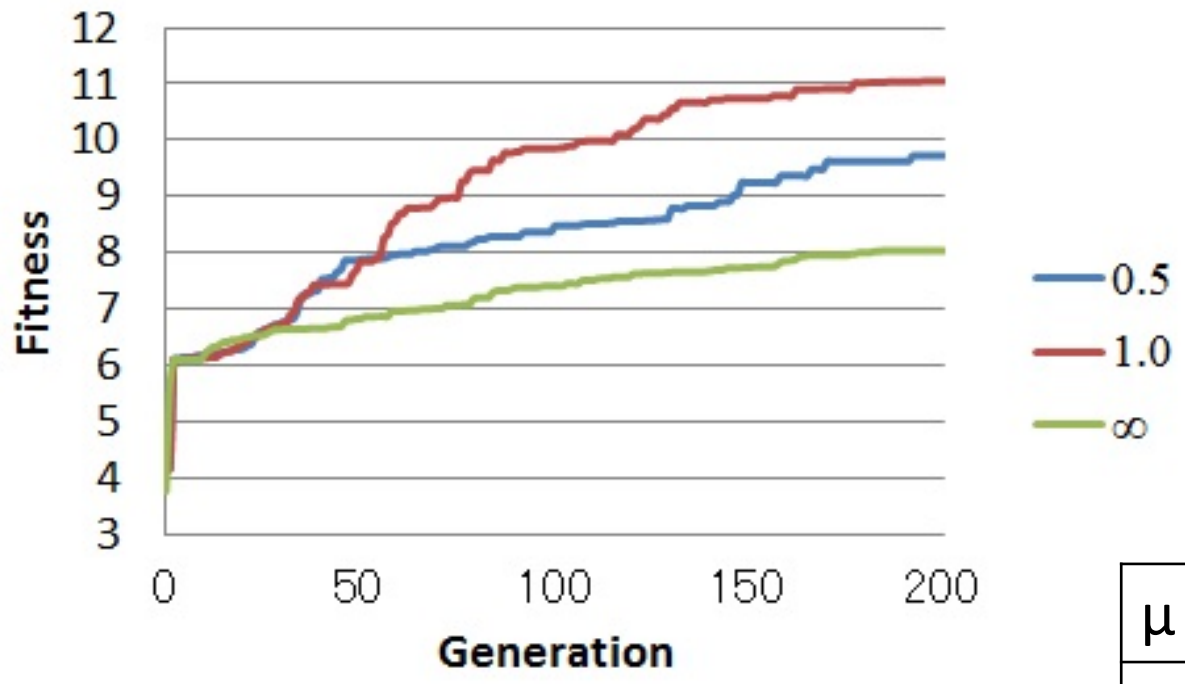
i-th neuron (*i-th* leg)	1	2	3	4
τ_i	14	10	10	29
A_i	6.41	15.48	5.71	10.14
ω_i	17.14	9.06	11.20	3.26
ϕ_i	-17.75	-3.51	-4.01	-1.76

μ=0.5

i-th neuron (*i-th* leg)	1	2	3	4
τ_i	54	5	20	1
A_i	16.52	47.28	6.13	50.04
ω_i	74.07	13.69	55.69	9.88
ϕ_i	35.97	26.25	22.08	15.40

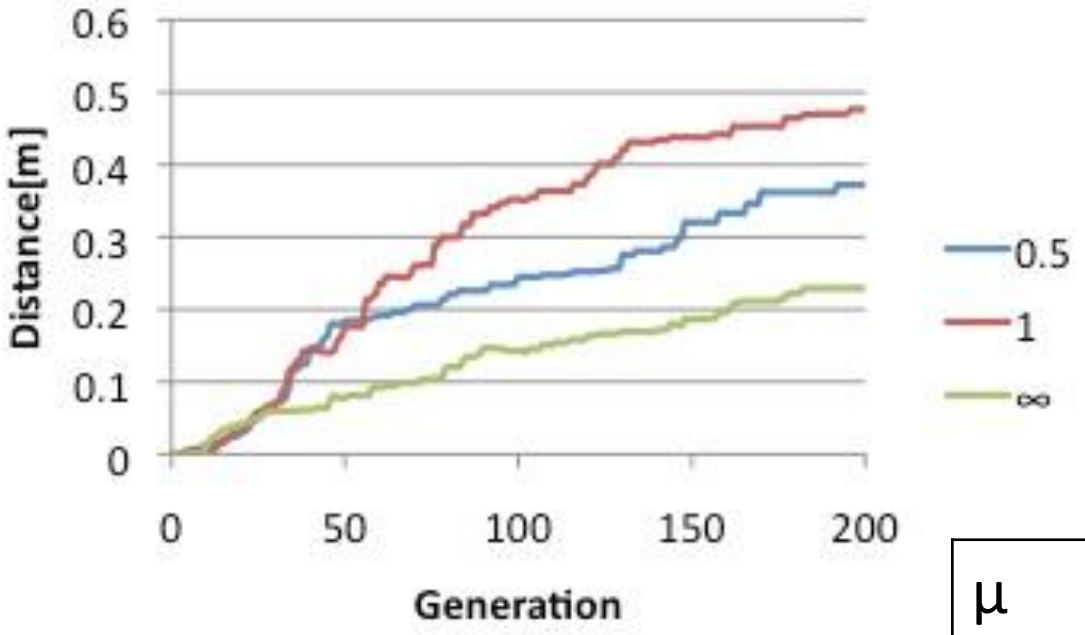
μ=∞

Evolutionary process – 6-legged robot



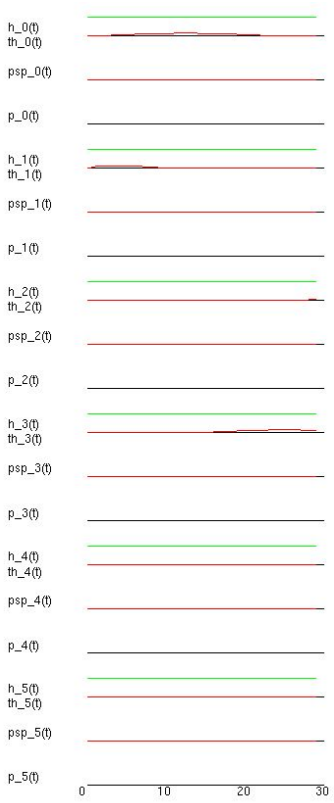
μ	0.5	1.0	∞
Average fitness	9.729	11.051	8.036
Maximum fitness	12.291	11.901	8.927

Distance

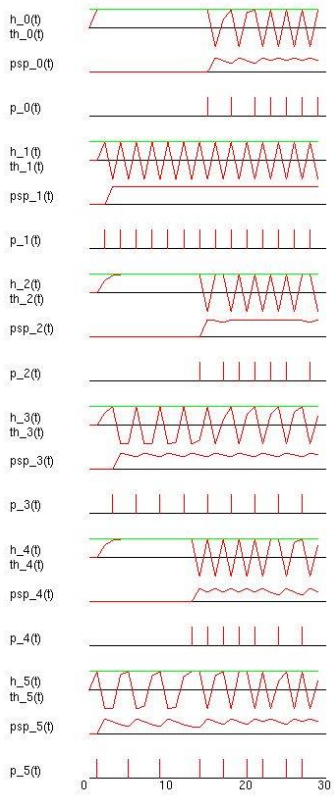


μ	0.5	1.0	∞
Average distance [m]	0.3729	0.4764	0.2292
Maximum dist. [m]	0.6441	0.5878	0.3698

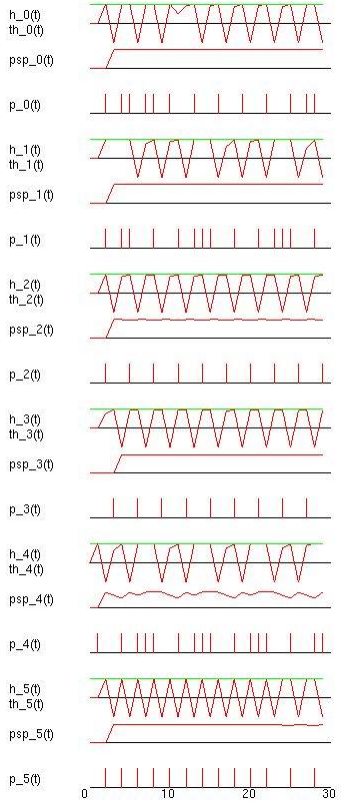
Spiking function



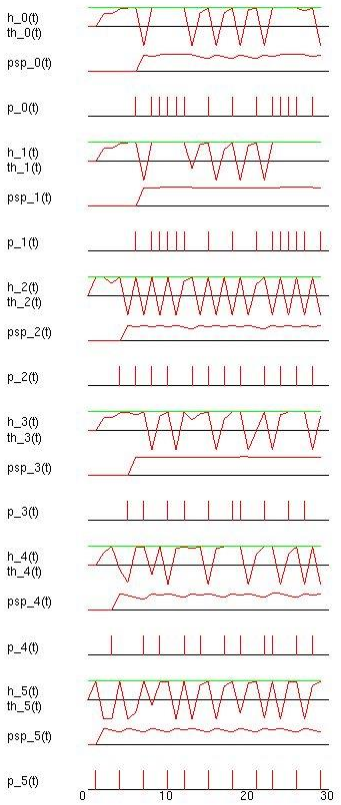
Initial



μ=0.5



μ=1.0



μ=∞

Neuron parameters

Initial

i-th neuron (*i-th* leg)	1	2	3	4	5	6
τ_i	6	8	10	12	14	16
A_i	0.15	0.15	0.15	0.15	0.15	0.15
ω_i	0.1257	0.1257	0.1257	0.1257	0.1257	0.1257
ϕ_i	0	1.5707	3.1415	4.7123	6.2832	7.8540

μ=0.5

i-th neuron (*i-th* leg)	1	2	3	4	5	6
τ_i	4	20	13	8	3	5
A_i	3.15	12.23	3.32	6.2	5.82	5.33
ω_i	0.21	3.13	0.16	8.33	0.25	4.78
ϕ_i	8.53	-10.39	-1.99	8.46	-2.96	-2.58

μ=1.0

i-th neuron (*i-th* leg)	1	2	3	4	5	6
T_i	59	138	45	36	5	25
A_i	44.73	20.25	6.37	0.17	25.74	5.04
ω_i	16.66	5.66	27.19	28.64	32.3	3.23
ϕ_i	18.08	35.67	3.07	19.42	-4.9	26.01

μ=∞

i-th neuron (*i-th* leg)	1	2	3	4	5	6
τ_i	7	29	6	32	7	7
A_i	23.56	32.67	11.57	14.67	14.69	10.86
ω_i	24.7	18.44	34.22	30.47	13.86	20.62
ϕ_i	-12.81	30.94	10.02	-5.65	34.37	-6.57

Conclusions

- The movements of four- and six-legged robots were discussed
- SNN was used to generate the motion patterns
- Parameter optimization and weight learning were solved using evolution strategy and Hebbian learning
- The model was implemented in ODE and real environment
- The robot was able to learn the motion patterns and increase the movement distance

Robot locomotion

Introduction

Background

Locomotion

Trajectory based Locomotion

It uses a mathematical model to generate the locomotion trajectory in Cartesian level

Biologically inspired based Locomotion

It uses a mathematical model inspired by animal or human morphologies to develop the locomotion system. In this case, we use neural oscillator as the signal generator

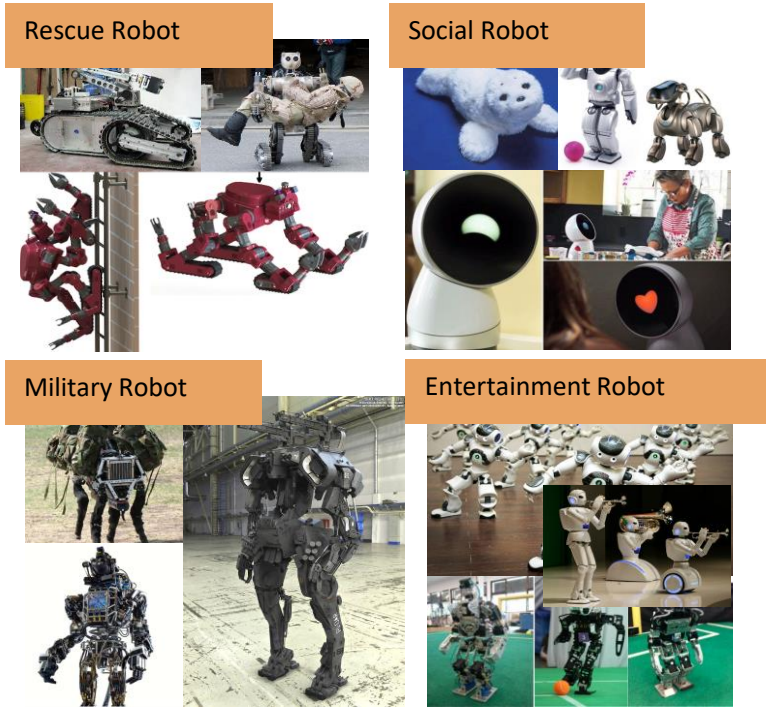
Problems

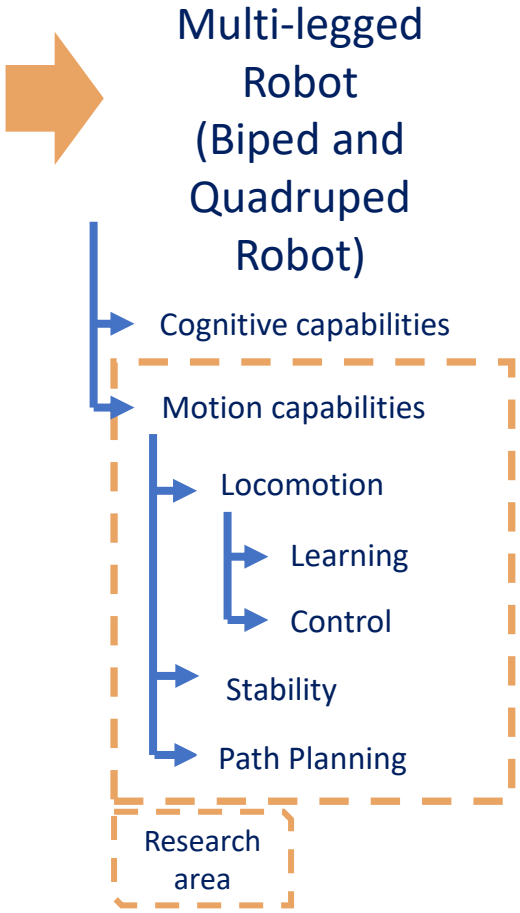
- Required behavioral learning to generate the locomotion pattern.
- The stability level is difficult to be reached in biologically inspired locomotion.
- The stability level was decreased when the walking speed was increased.

Solution

- Improving the connectivity structure of motor neurons based on preliminary tests
- Designing the structure of sensor neurons connected to motor neurons using an evolutionary algorithm.
- Development of stabilization support
- Application of a multi-objective evolutionary algorithm to find a trade-off between good stability and high speed of movement.
- MLP-based gait mode generator that generates synaptic weights for neural connections

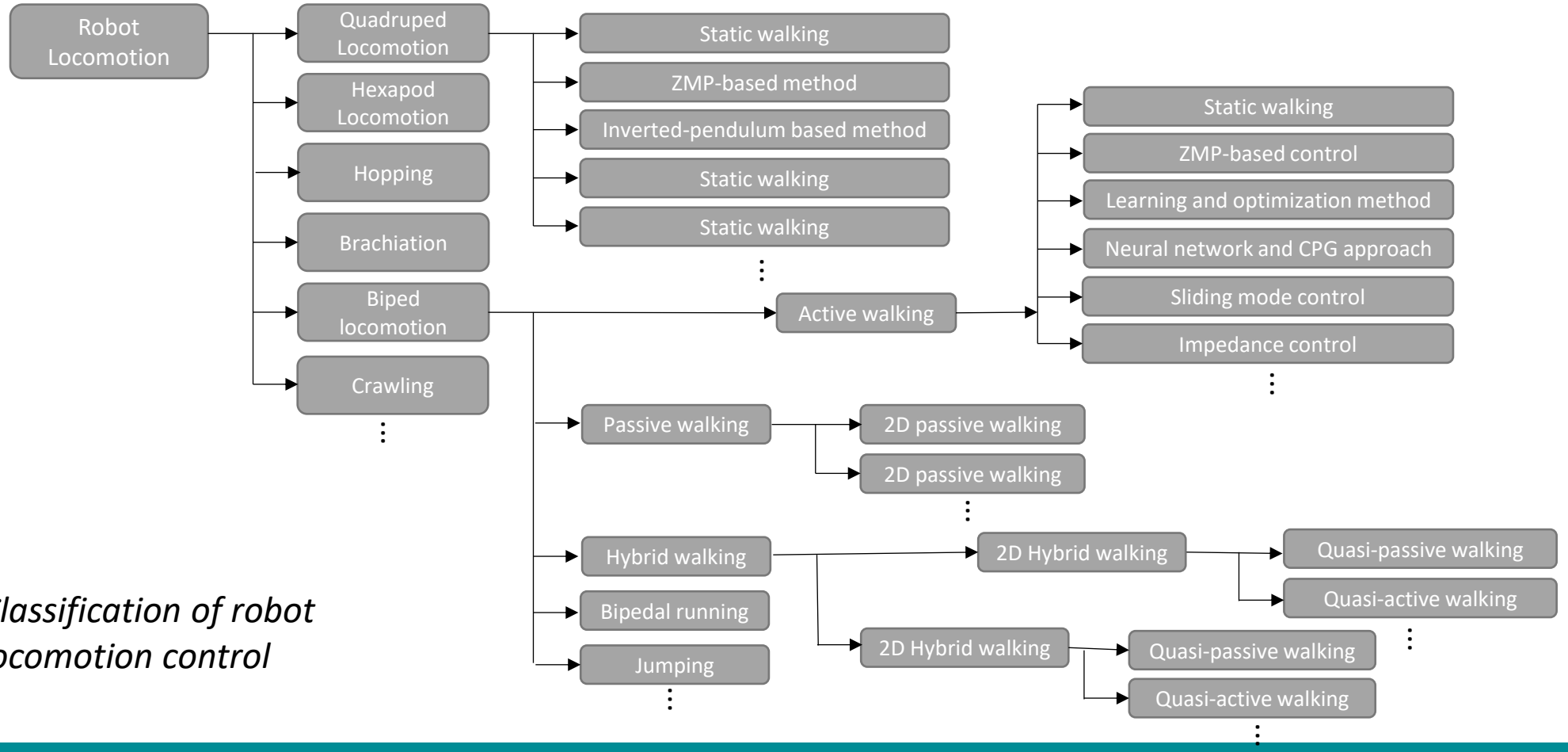
Background





- The multi-legged robot represented by proposed model is a suitable robot in many fields: social life, rescue, military purposes, or entertainment (soccer, dancing).
- Possible for applying both biped walking and quadruped walking
- The cognitive capabilities of humanoid robots are important, but their motion capabilities are also important to support its activity

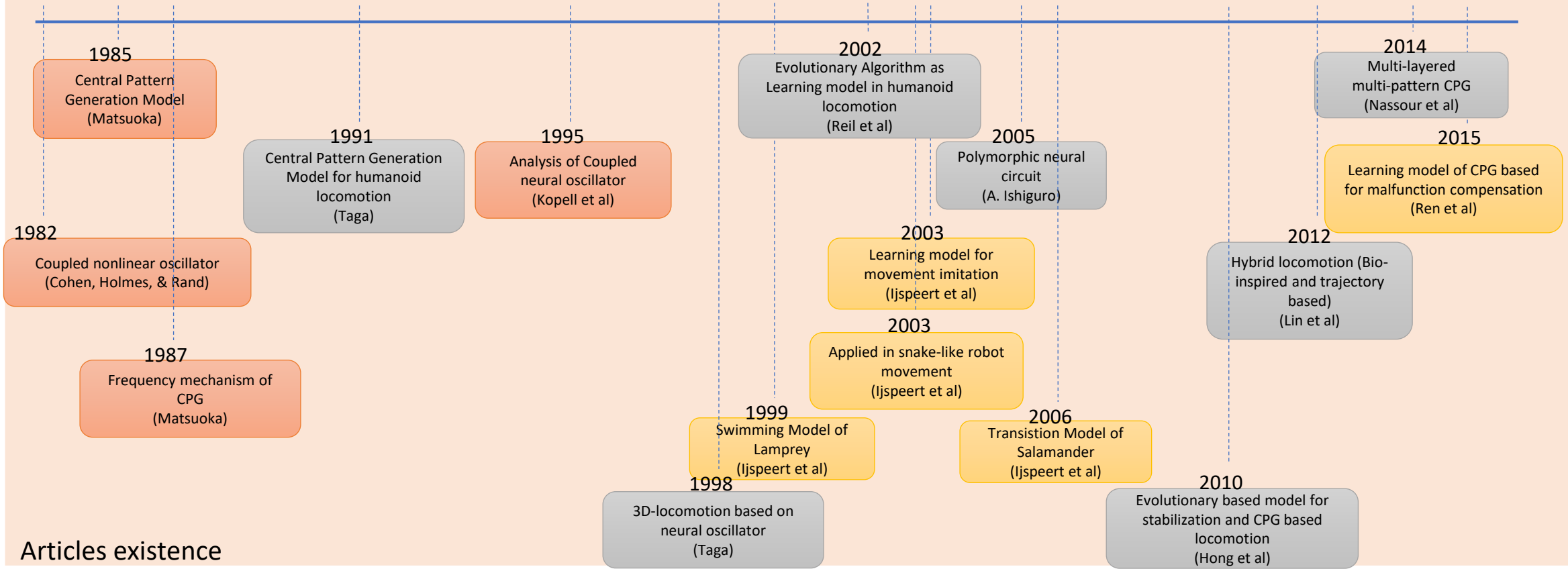
Study of Locomotion



Classification of robot locomotion control

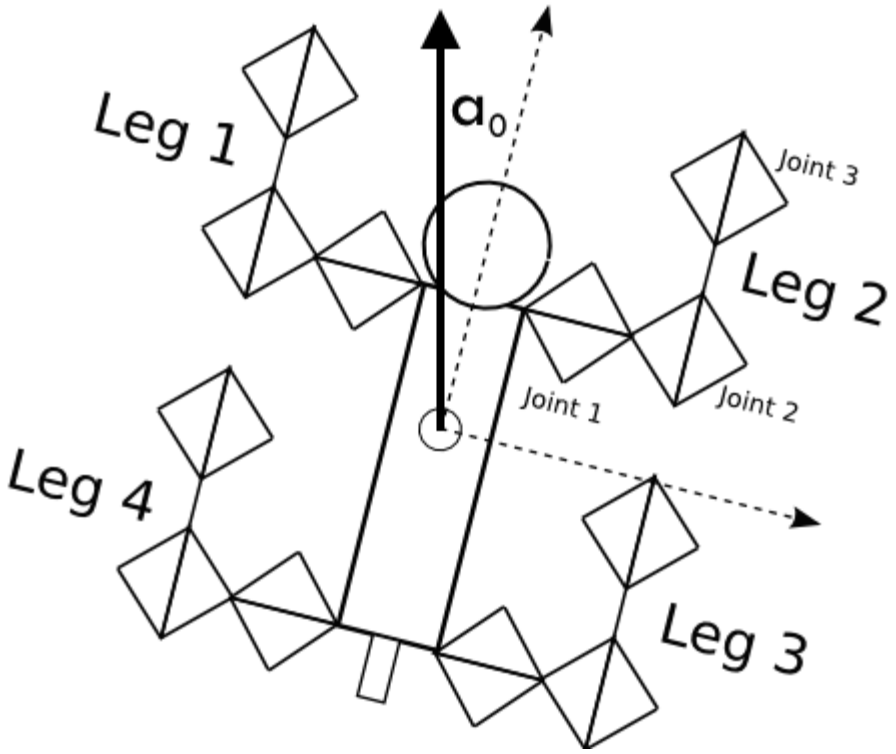
Study of Bio-inspired Locomotion

Research and development timeline

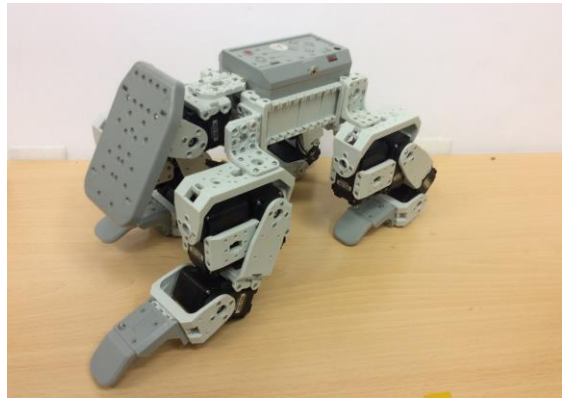


Robot design (quadruped)

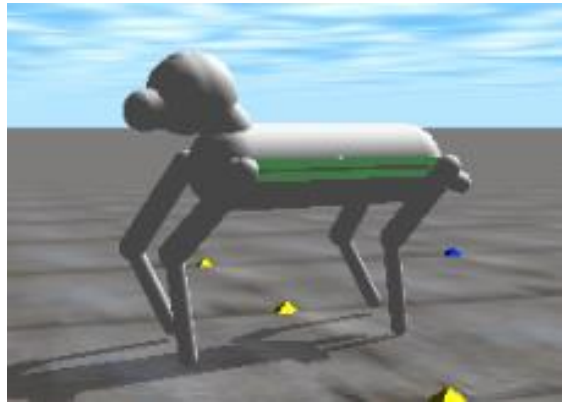
- The mechanical structure of the robot is adapted from the animal structure.
- Each leg has 3 joints represented by a coupled neuron
- Robot is equipped with inertial sensor, such as: gyroscope, accelerometer, and inertial measurement unit (IMU).



Mechanical structure

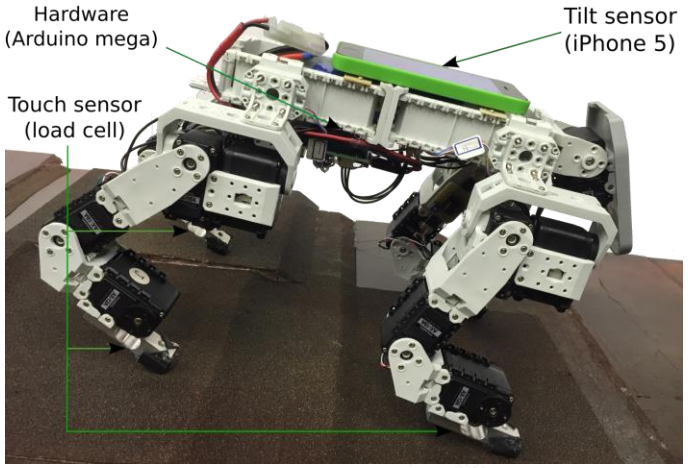


Real design

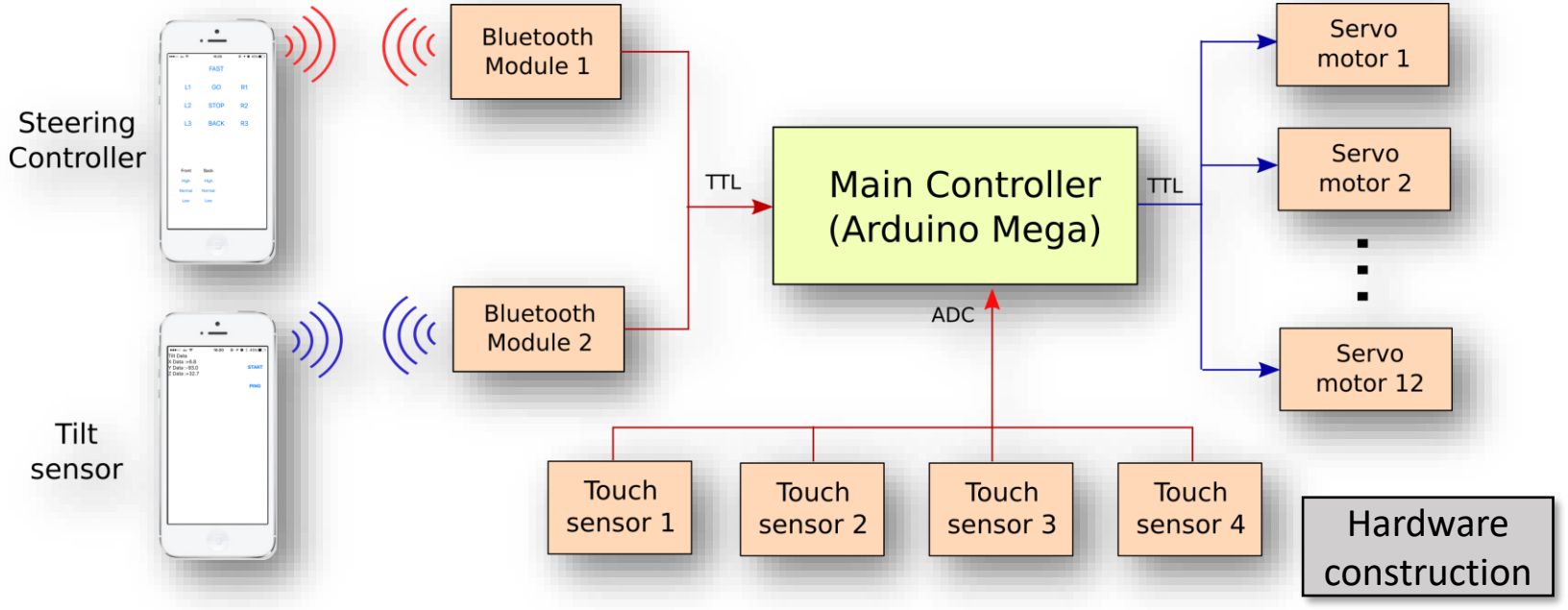


Simulation design

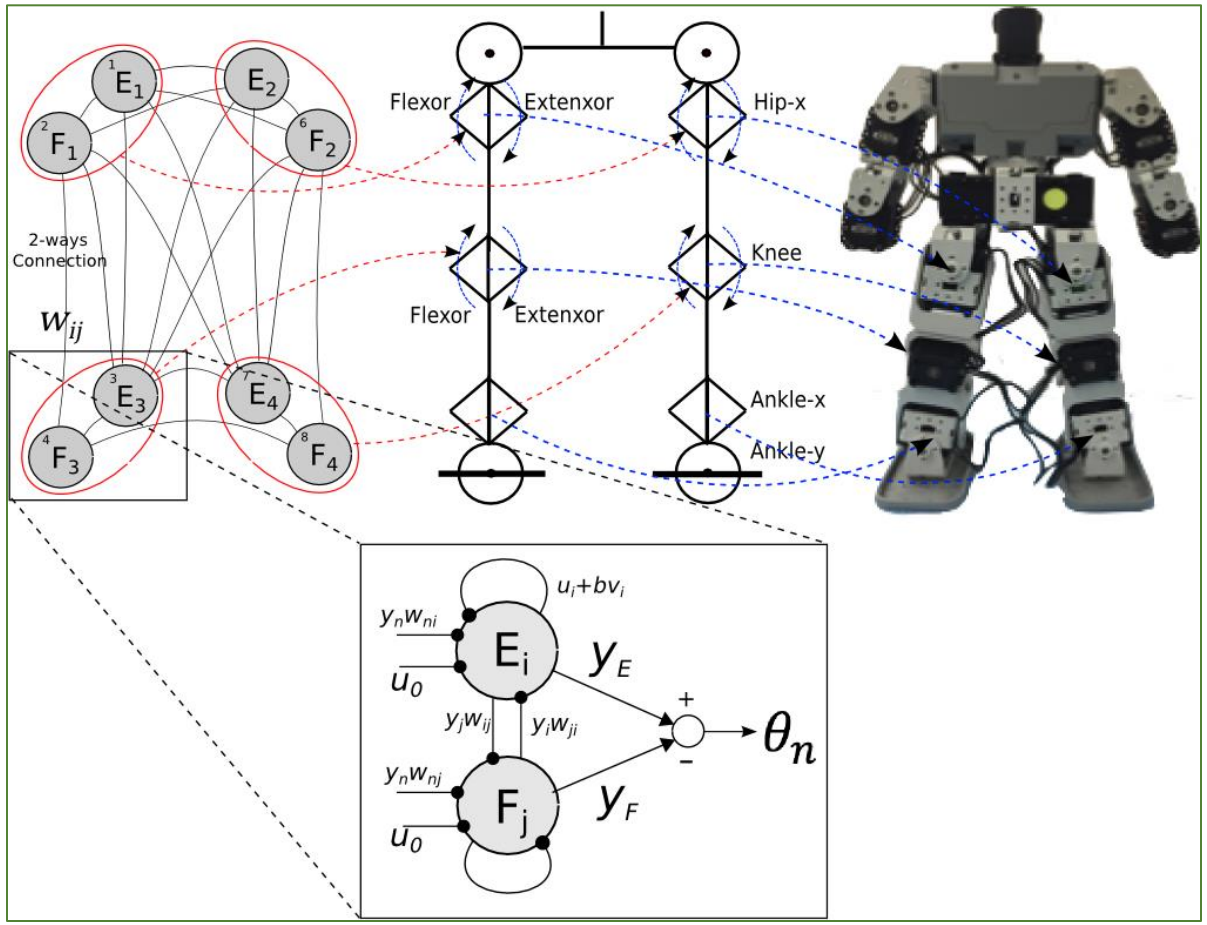
Robot design (quadruped)



- Used simple and cheap 4 legged robot.
- In 4 legged robot, there are 3 degrees of freedom (hipx, hip-y, and knee joint) in each leg.
- The robot's size is approximately 30cm X 18cm X 20cm, and its weight is 2 kg.



Robot design (biped)

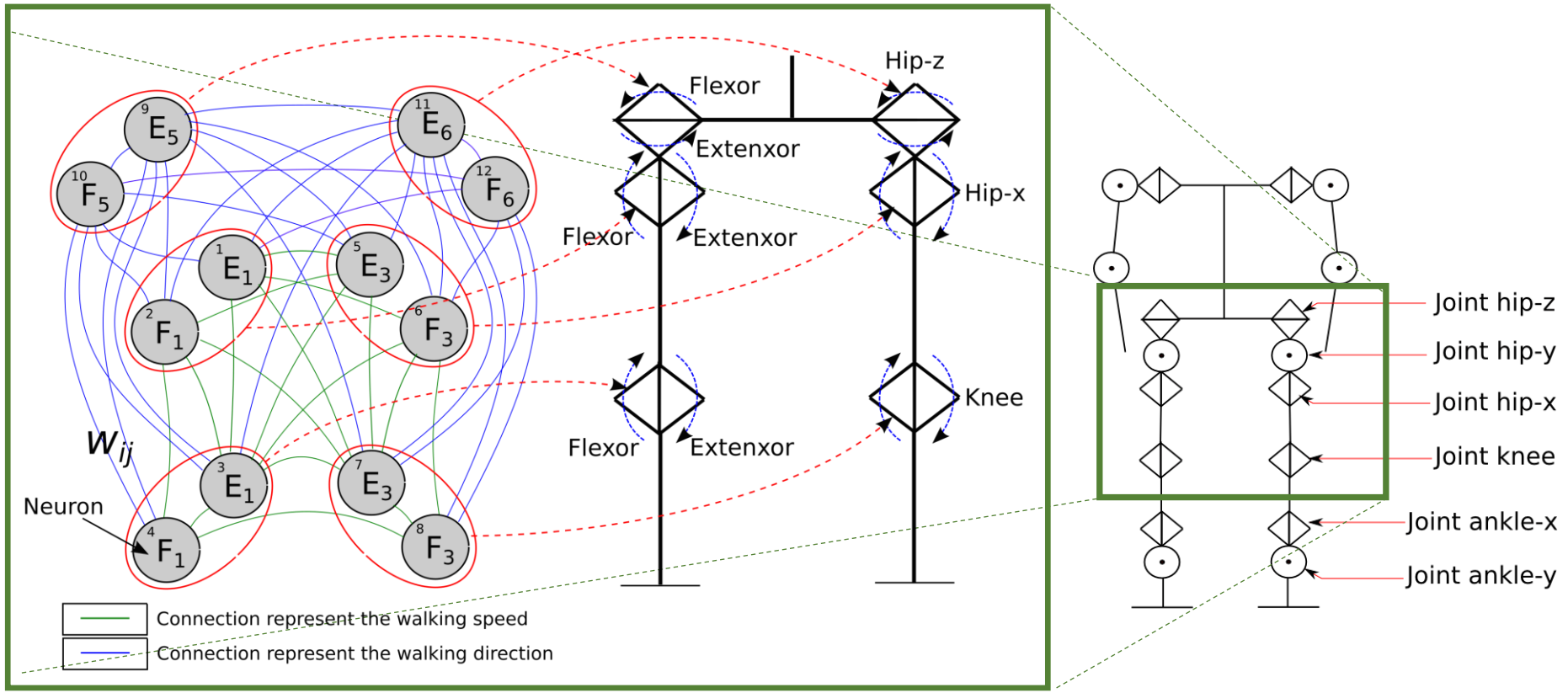


- The robot has 6 DoF in each leg.
- Only 2 important joints for forward walking in each leg are required to be optimized (hip-x and knee joint).
- Each joint is represented by 2 coupled neuron.
- The robot is equipped with inertial sensor, such as: gyroscope, accelerometer, and inertial measurement unit (IMU).
- The robot design is created in computer simulation open dynamic engine (ODE).

Neurooscillator based locomotion generation

Locomotion Generator – Robot design

- The robot has 6 DoF in each leg.
- Only 2 important joints for forward walking in each leg are required to be optimized (hip-x and knee joint).
- Hip-z joint is the important joint for representing the walking direction.
- Each joint is represented by 2 coupled neuron.
- The robot is equipped with inertial sensor, such as: gyroscope, accelerometer, and inertial measurement unit (IMU).
- The robot design is created in computer simulation open dynamic engine (ODE).

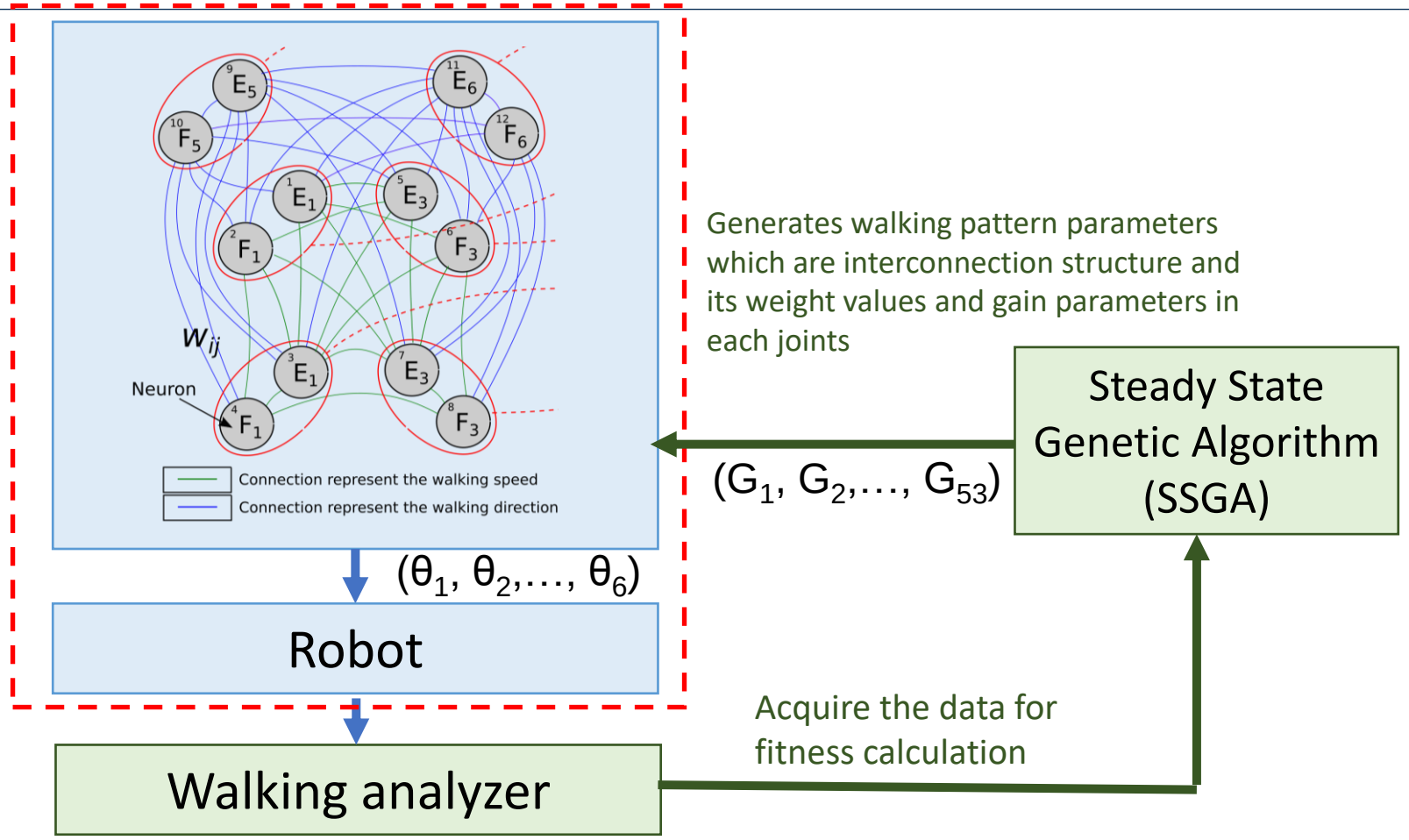


Walking model – Walking pattern optimization

Interconnection structure

Transferring the signal output of neuron to the robot in joint angle level

(walking speed, direction and stability level)



Locomotion Generator – Joint angle level

$$\tau_l \dot{x}_i = x_i - \sum_{j=1}^{n} w_{ij} y_j + S_i - b v_i$$

$$\tau_r \dot{v}_i + v_i = y_i$$

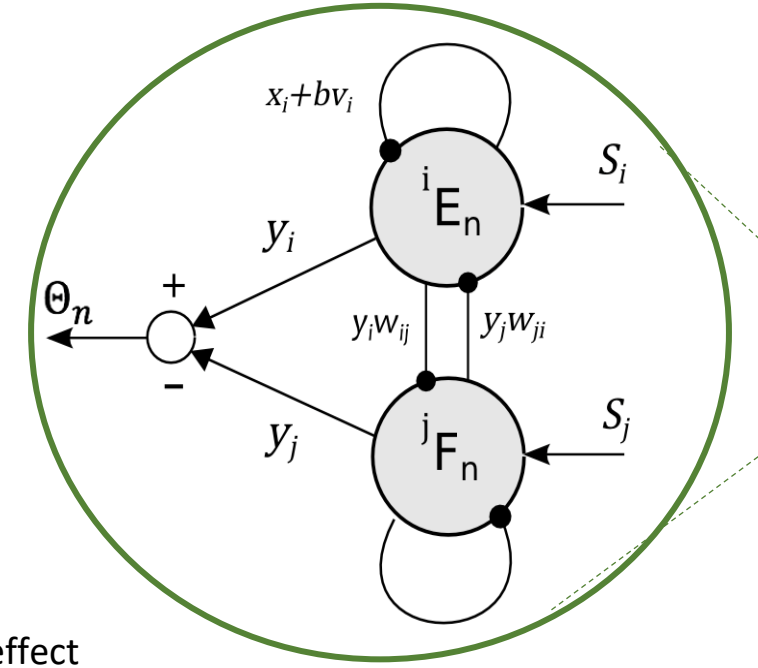
$$y_i = \max(x_i, 0)$$

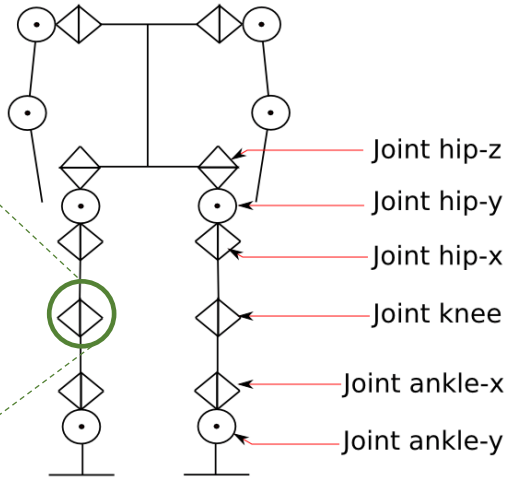
$$\theta_n = (y_{2n} - y_{2n+1}) \mu_n$$

$$\theta_4 = \theta_2 - \theta_1$$

$$\theta_5 = -\theta_3$$

To be optimized





θ_1 -> joint hip-x
θ_2 -> joint knee
θ_3 -> joint hip-y
θ_4 -> joint ankle-x
θ_5 -> joint ankle-z
θ_6 -> joint hip-z

- τ_l : constant time of the inner state
- τ_r : contant time of the adaptation effect
- x_i : inner state of the *i*th motor neuron
- v_i : self-inhibition effect of *i*th neuron
- y_i : output of the *i*th motor neuron
- S_i : constant value effect from outside

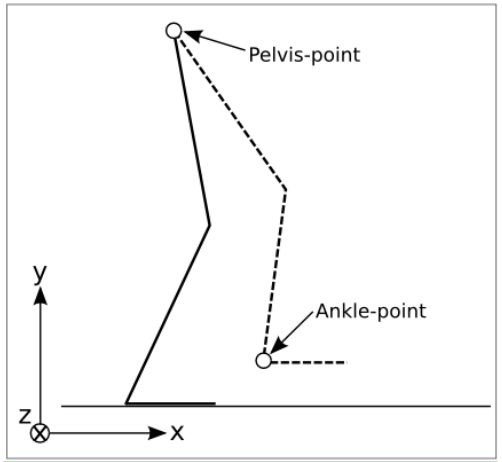
w_{ij} : strength of inhibitory connection

b : coefficient of self-inhibition effect

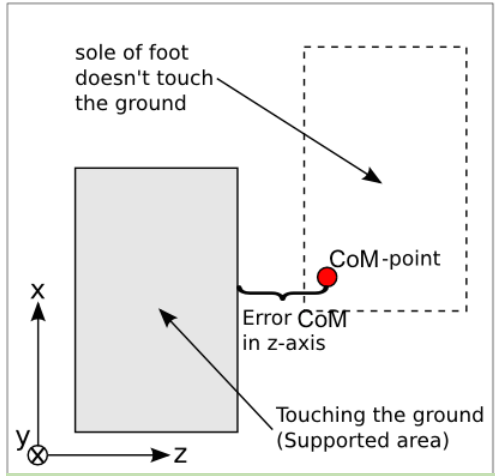
θ_n : the value of the *n*th joint

μ_n : Gain parameter

Interconnection optimization – Fitness calculation



2-D pose walking robot



Center of Mass representation

The fitness function is defined by the minimization of the error of walking speed ($E^{(v)}$), the error of desired step length ($E^{(l)}$), the height of step (h), the Center of Mass (CoM) error in x-axis ($E_x^{(CoM)}$) and z-axis ($E_z^{(CoM)}$).

$$E^{(v)} = \sum_{j=1}^{T} \|d_v - v(t)\|$$

$$E^{(l)} = \|d_l - l\|$$

$$h = \begin{cases} y(t) & \text{if } h < y(t) \ \& \ x_p(t) < x_a(t) \\ h & \text{otherwise} \end{cases}$$

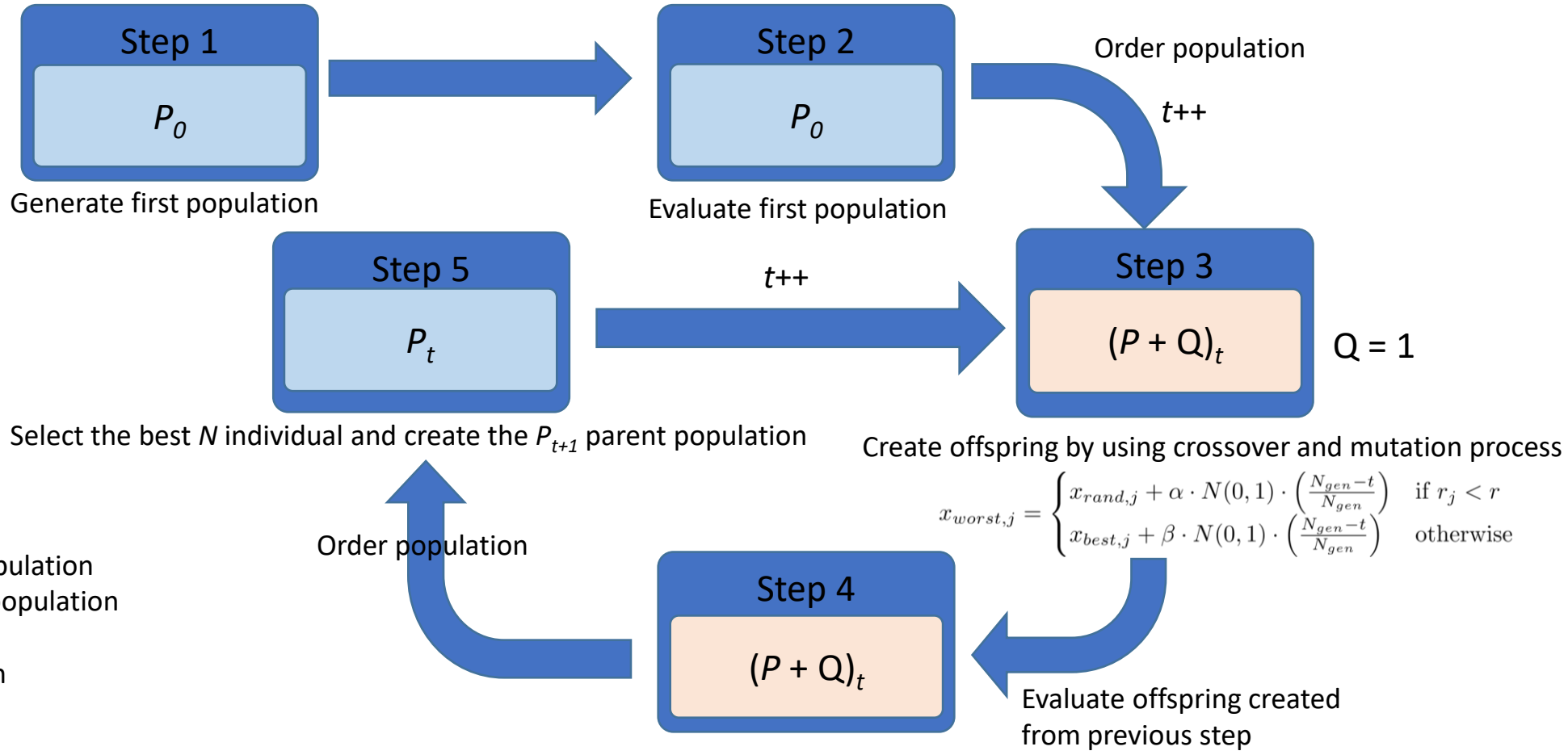
$$E_z^{(CoM)} = \sum_{t=1}^{T} \|CoM_z(t)|$$

$$E_x^{(CoM)} = \sum_{t=1}^{T} \|CoM_x(t)|$$

$$f = w_1^{(f)} E^{(v)} + w_2^{(f)} E^{(l)} - w_3^{(f)} h + w_4^{(f)} E_z^{(CoM)} + w_5^{(f)} E_x^{(CoM)}$$

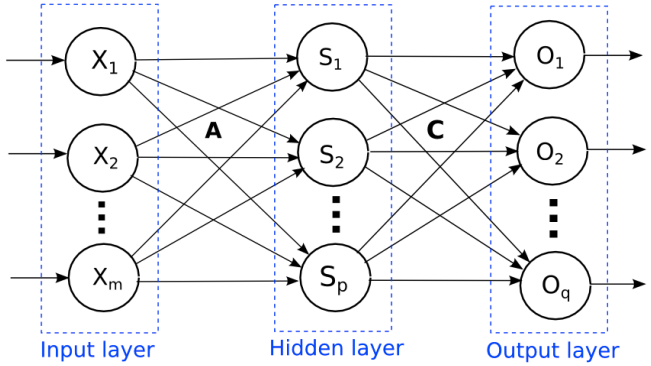
- d_v : the desired speed
- $v(t)$: the current speed
- d_l : the desired length of step
- l : the current length of step
- $y(t)$: the current height of step illustrated in Figure,
- $x_p(t)$: the current pelvis position in x-axis
- $x_a(t)$: the current ankle position in x-axis
- $CoM_z(t)$: the length of CoM from supported area in z-axis
- $CoM_x(t)$: the length of CoM from supported area in x-axis
- T : the maximum time cycle
- t : the current time cycle

Interconnection optimization – SSGA process



Where :
P : parent population
Q : offspring population
x : individual
t : generation

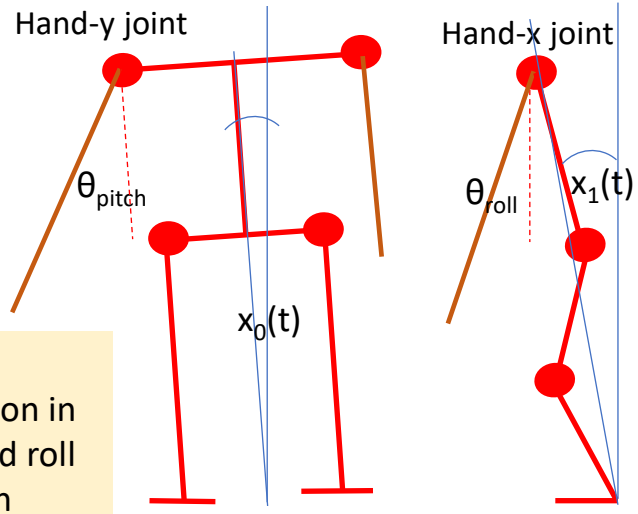
Stability support system



The stability system uses the robot's hand as response actuator to minimize the oscillation of the robot's tilt angle

$$S_y(t) = f(S'_y(t)) = f\left(\sum_i^m x_i(t)A_{ij} + b_j\right)$$

$$O_k(t) = g(O'_k(t)) = g\left(\sum_j^p s_j(t)C_{jk} + b_k\right)$$



Robot illustration in pitch and roll direction

- $x_i(t)$: the input value of the tilt sensor from the *i*-th input neuron is illustrated in the figure
- S'_y (t) : the input value of the y-th hidden neuron
- S_y : the output value of the y-th hidden neuron
- O_k(t) : the output value representing the value of joint angle of hands
- δ_k : the error propagation in output neuron
- δ_j : the error propagation in hidden neuron
- d_k : the desired output
- $g_k(x)$: the output of the tilt sensor $g_0(O_0)$ for pitch (θ_{pitch}) and $g_1(O_1)$ for roll (θ_{roll}) direction

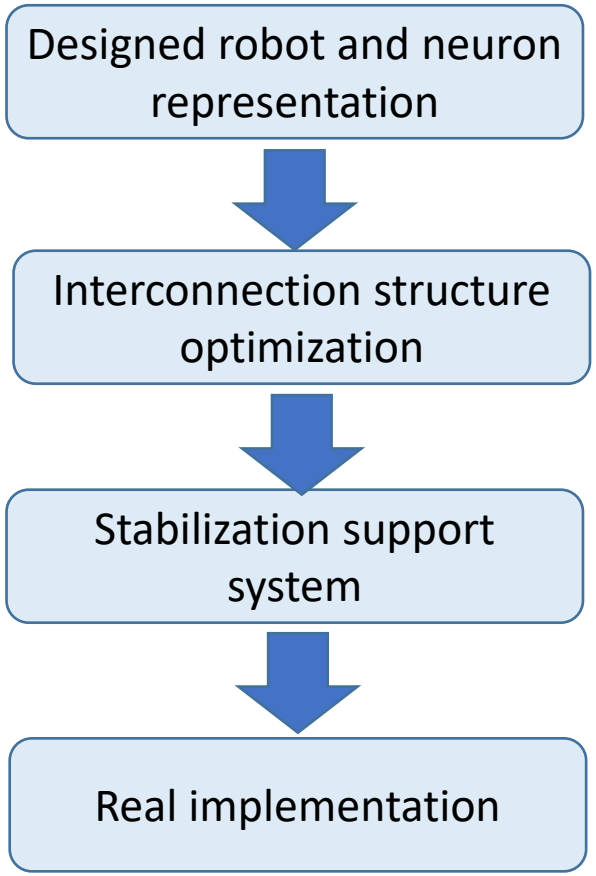
$$\delta_k = (d_k - g_k(O_k))f'(O'_k)$$

$$\delta_j = \sum_k^q \delta_k C_{jk} f'(S'_j)$$

$$\mathbf{C}(t+1) = \mathbf{C}(t) + \eta \mathbf{s}(t)\delta_k^T$$

$$\mathbf{A}(t+1) = \mathbf{A}(t) + \eta \mathbf{s}(t-1)\delta_j^T$$

Experimental result



- In this research, the experiments were conducted in computer simulation using open dynamic engine (ODE). We designed the robot with parameters as: weight, height, center of mass, environmental parameter etc.

- Design robot structure and neurons represent each joint angle based on preliminary study
- We optimize the interconnection structure of neural oscillator for acquiring stable speed, good stabilization, and length of step appropriate to the desired length of step. We used 2-D robot simulation
- Conducted in computer simulation ODE.

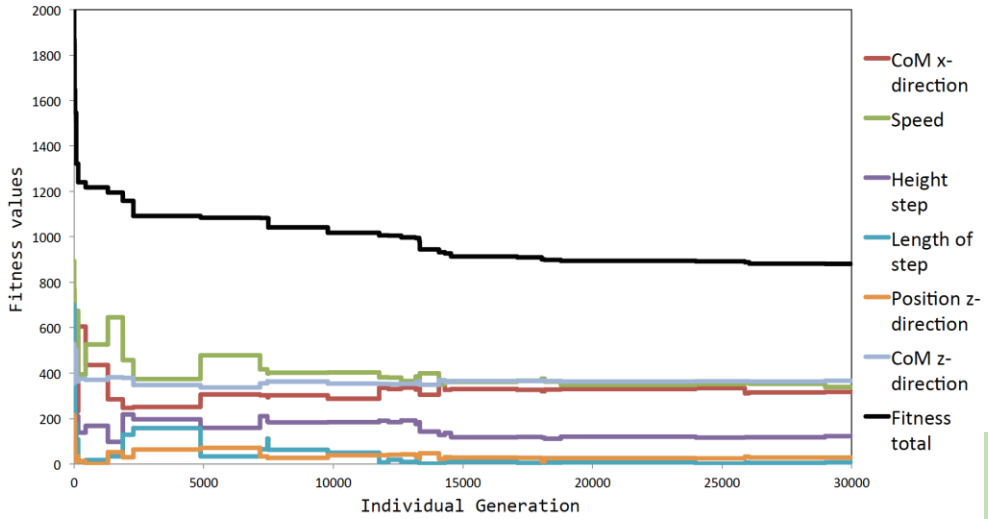
Parameter	Time cycle	τ_l	τ_r	b	η
Value	0.01 second	12.0	1.2	2.5	0.01

- Conducted in small humanoid robot for the next research

Experimental results – Optimization process

Parameter	Value
N_{ind}	5000
N_{gen}	20000
α	0.01
β	0.01
$x_{min,(1-50)}$; $x_{min,(51-53)}$	-3.5; 0.8
$x_{max,(1-50)}$; $x_{max,(51-53)}$	3.5; 3.0
$w_1^{(f)} - w_5^{(f)}$	{0.2, 0.3, 0.2, 0.2, 0.1}

Parameters used in the SSGA

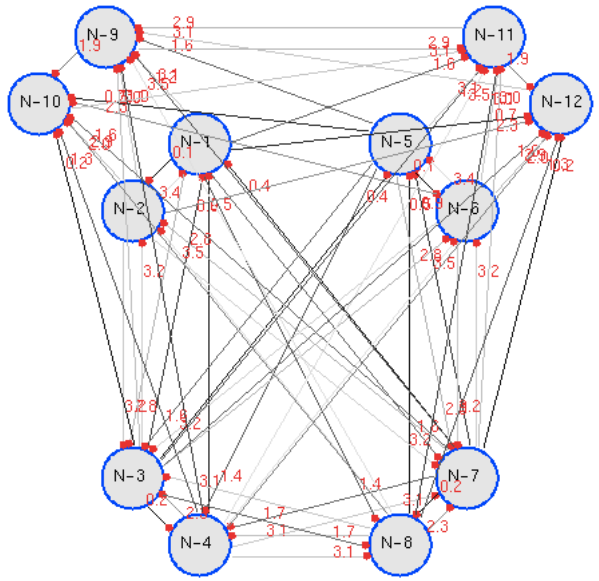


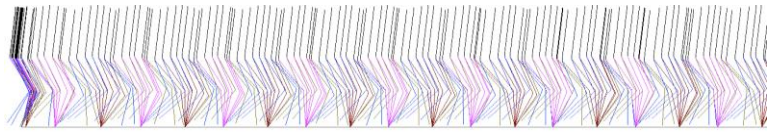
Fitness evolution

In this experiment, we optimize the interconnection structure of neural oscillator for acquiring stable speed, good stabilization, and length of step appropriate to the desired length of step. We used 2-D robot simulation to optimize the structure

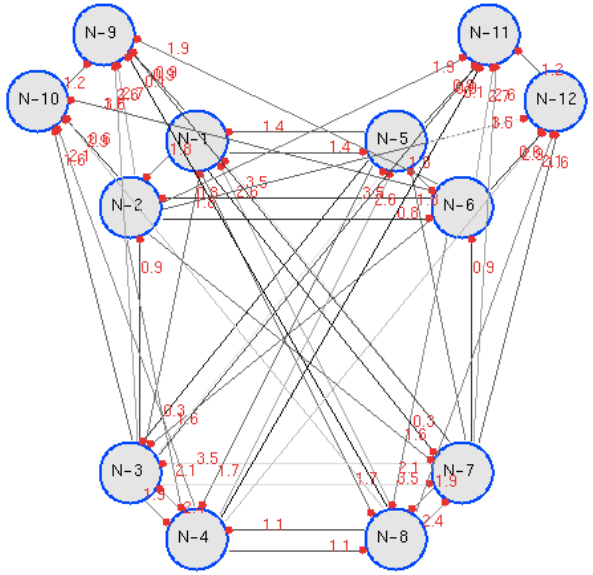
Source neuron (j) \ Destination neuron (i)	1	2	3	4	5	6	7	8	9	10	11	12
1	*0*	0.00	0.00	0.00	2.09	0.00	3.07	0.87	0.00	1.23	1.02	1.09
2	0.00	*0*	3.45	0.00	1.96	0.00	3.50	2.21	0.92	2.92	0.00	0.00
3	1.65	0.00	*0*	1.58	0.67	0.43	1.50	0.23	2.92	1.97	0.38	2.77
4	0.00	0.00	0.00	*0*	1.39	0.00	0.00	0.51	3.32	0.00	2.02	3.50
5	2.09	0.00	3.07	0.87	*0*	0.00	0.00	0.00	1.02	1.09	0.00	1.23
6	1.96	0.00	3.50	2.21	0.00	*0*	3.45	0.00	0.00	0.00	0.92	2.92
7	0.67	0.43	1.50	0.23	1.65	0.00	*0*	1.58	0.38	2.77	2.92	1.97
8	1.39	0.00	0.00	0.51	0.00	0.00	0.00	*0*	2.02	3.50	3.32	0.00
9	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0.80	2.89	1.97
10	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	0.00	*0*	2.81	0.00
11	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	2.89	1.97	*0*	0.80
12	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	2.81	0.00	0.00	*0*

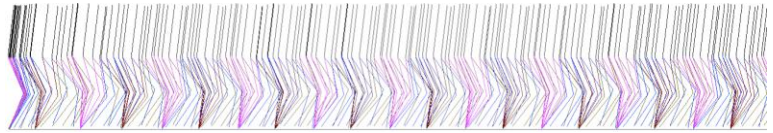
Experimental results – Optimization results



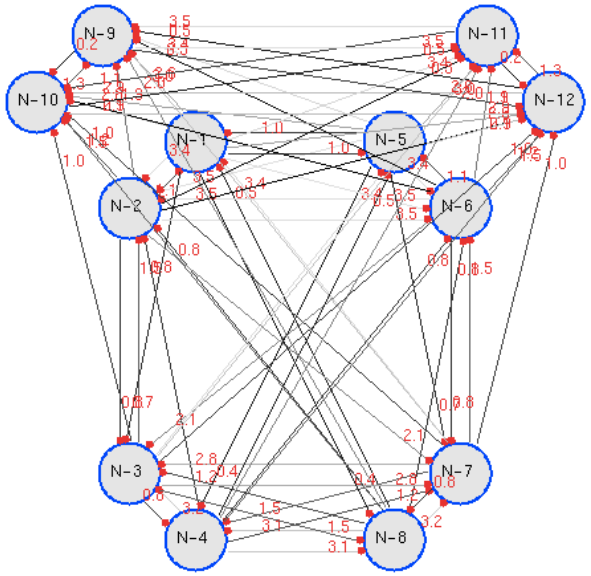


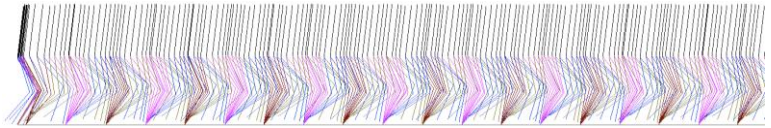
10000 generation





20000 generation

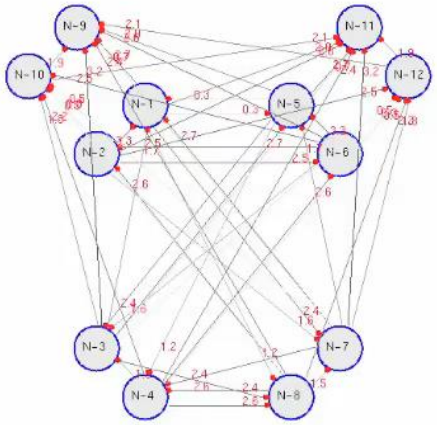




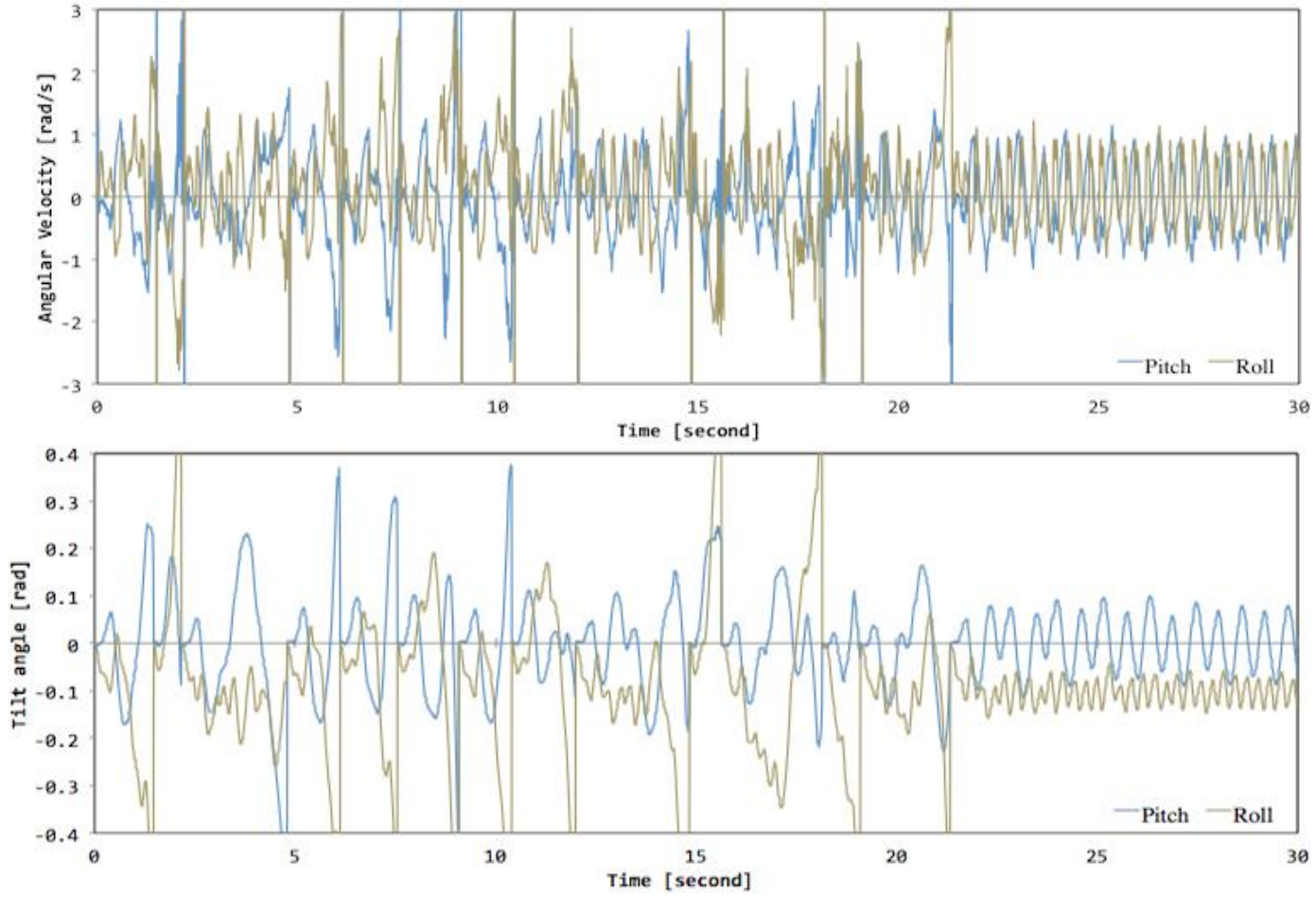
30000 generation

The pelvis speed of locomotion pattern resulted in 10000-th and 20000-th generation are still not stable, but in 30000-th generation, the pelvis speed is stable enough.

Experimental results - Video on optimization



Experimental results - Stabilization learning system



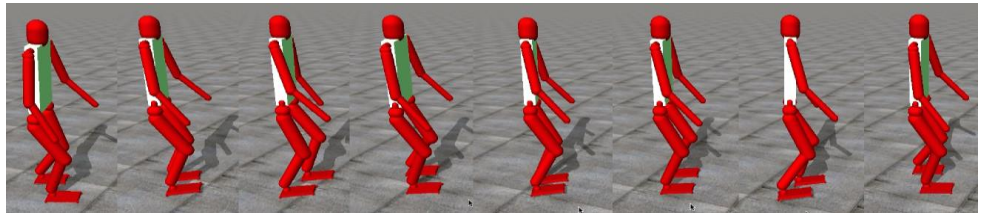
(a) Angular velocity signal in pitch and roll direction

- These figures represent the learning process.
- Figures (a) and (b) represent the signal from angular velocity sensor and tilt angle sensor, respectively.
- The robot can walk stably after learning the environmental condition in 21 seconds. Before that, the robot fell down several times.

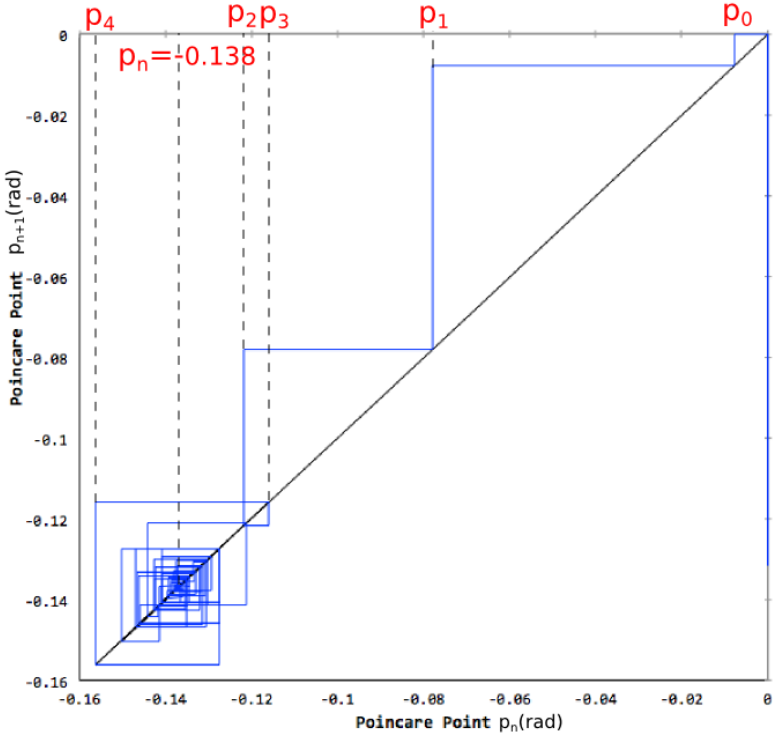
(b) Tilt angle of robot body in pitch and roll condition

Experimental results - Stabilization analysis

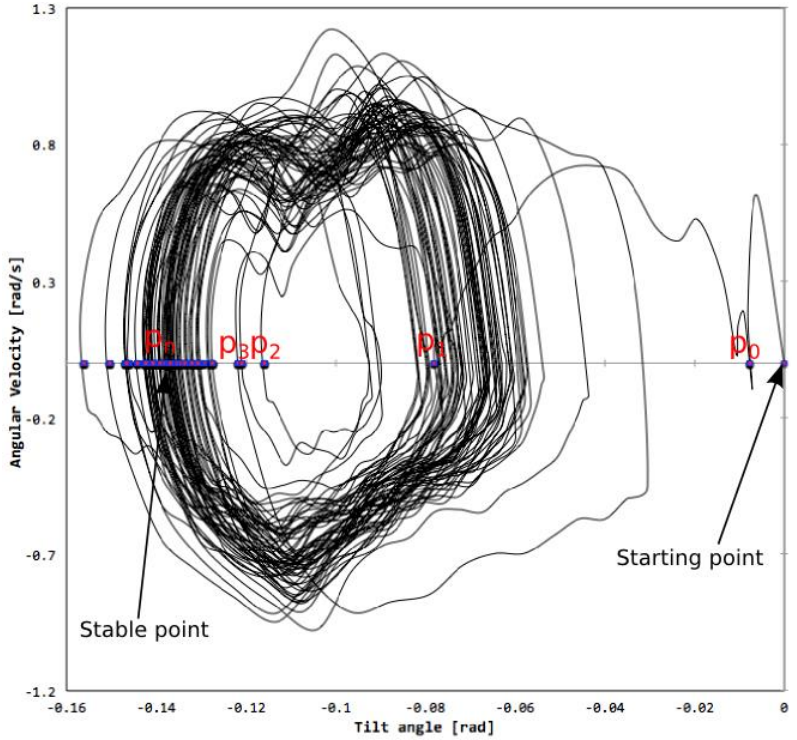
- The initial condition of robot's tilt when the robot stands up was 0 rad, when the robot was walking, the robot's tilt changed to stable tilt angle (0.138 rad).



Simulation view

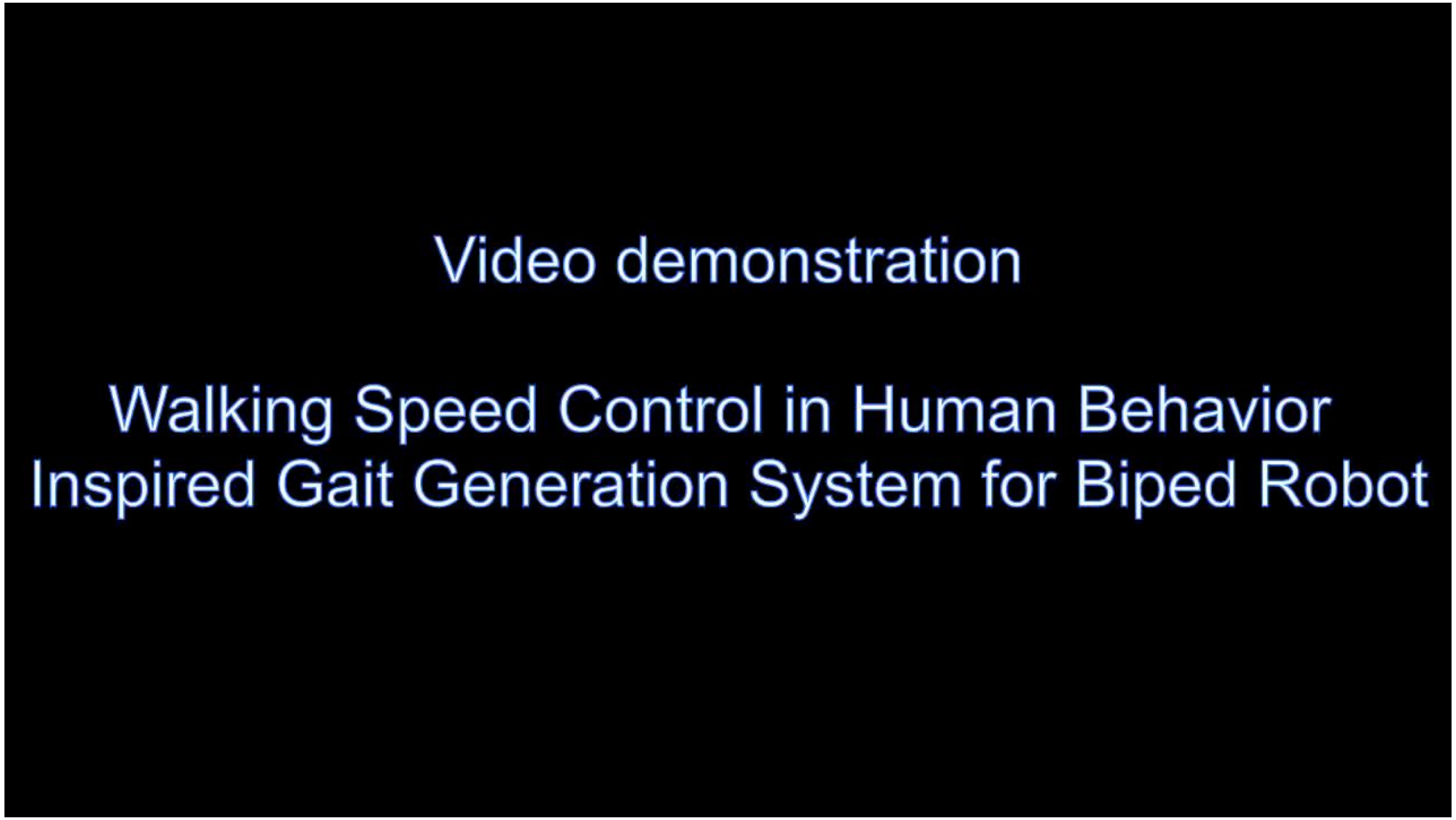


Cobweb diagram



Poincare diagram

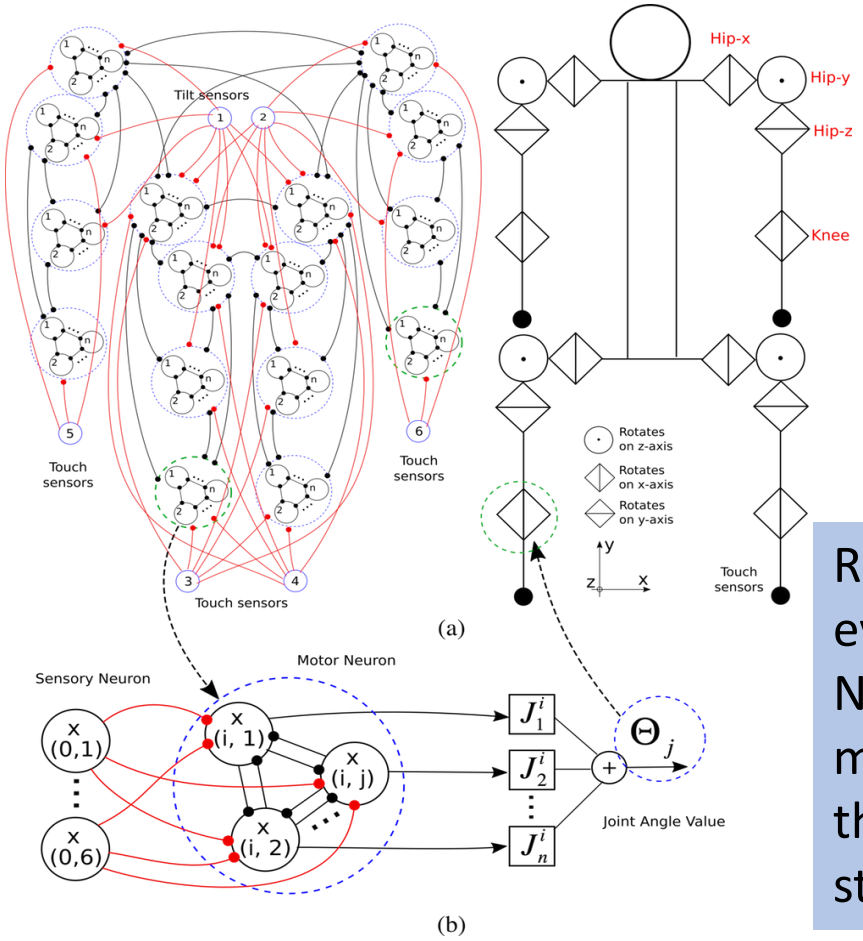
Video biped robot locomotion



Interconnection structure is changed depending on the walking speed

Evolving a sensory-motor interconnection structure

Proposed model



Representation of the evolving neural model (a) Neural structure and mechanical structure of the robot. (b) Neuron structure in joint angle

$$\tau \dot{u}_i = \left(u_0 - u_i - \sum_{j=1}^{n} w_{ij} y_j J_j^k + \sum_{l=1}^{n} m_{il} s_l - b v_i \right) \tau_f$$

$$\tau' \dot{v}_i = (y_i - v_i) \tau_f$$

$$y_i = \max(u_i, 0)$$

$$\Theta_i^{(l,r)} = \sum_{j=1}^{N_i^{neuron}} J_j^{l,r} y_j$$

Inner state calculation

Notation	Definition
w_{ij}	motor neurons interconnection
m_{il}	sensory motor neurons interconnection
u_i	inner state
y_i	output value
v_i	variable of self-inhibition effect of neurons
J_j^k	operator representing +1, -1, or 0
s_l	output of the lth sensory neuron
b	rate of the adaptation value
u_0	external input for coupled neurons
τ and τ'	time constant of the inner state and the adaptation effect

$$\theta_0 = \begin{cases} \theta_1^{(l)} - \theta_1^{(r)} & \text{if } N^{joint} < 3 \\ \Theta_3 & \text{otherwise} \end{cases}$$

$$\theta_1^{(l,r)} = \Theta_0$$

$$\theta_3 = \begin{cases} \theta_2^{(l,r)} - \theta_1^{(l,r)} & \text{if } N^{joint} < 4 \\ \Theta_4 & \text{otherwise} \end{cases}$$

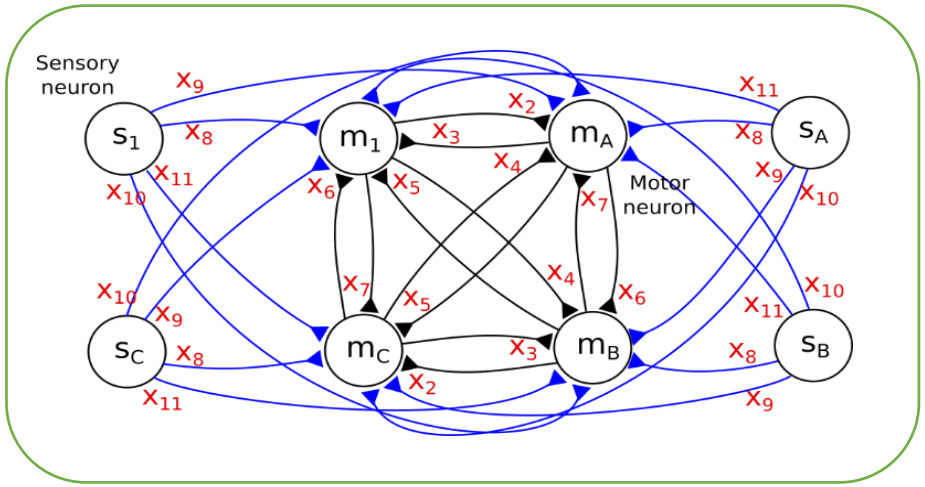
Joint angle calculation

The model so far

$$Number\ of\ link = N^2 - N$$

20 Neurons → 380 Links

Has longer learning process

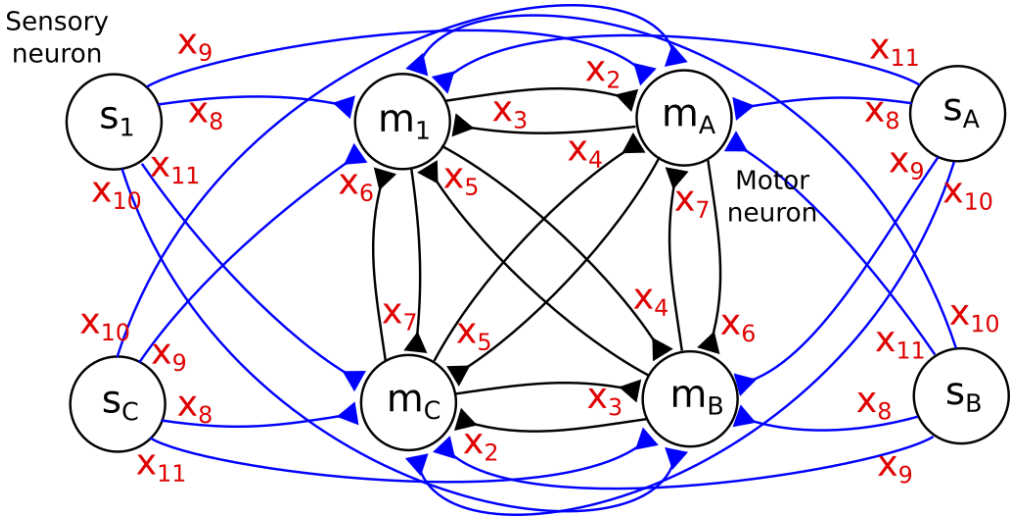


Distributed Tree Structure Model

Distribution of neuron structure

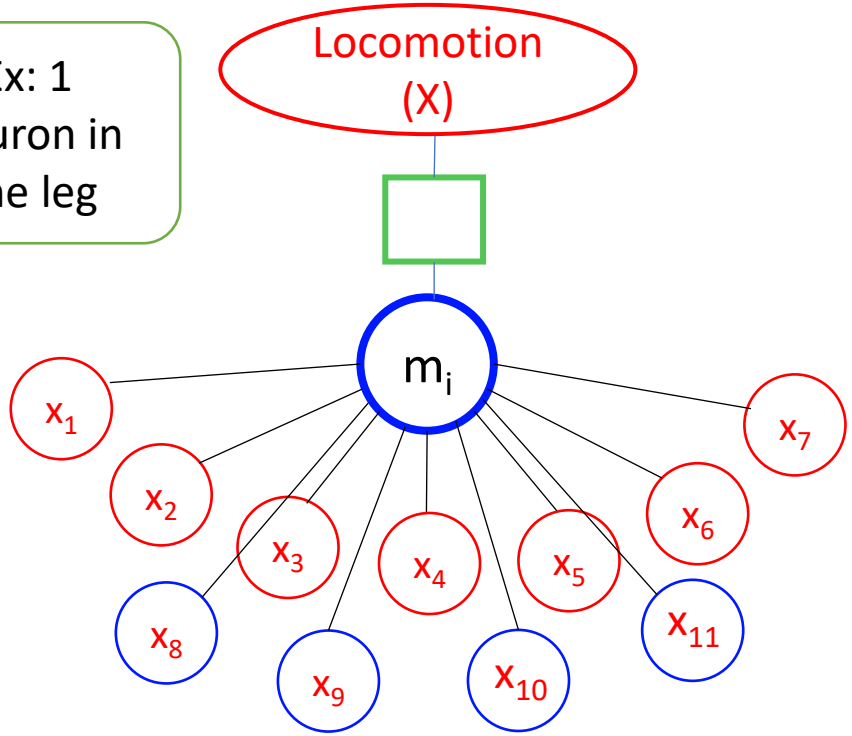
if $X = 1$, $A = 2$; $B = 3$; $C = 4$
if $X = 2$, $A = 2$; $B = 4$; $C = 3$
if $X = 3$, $A = 3$; $B = 2$; $C = 4$
if $X = 4$, $A = 3$; $B = 4$; $C = 2$
if $X = 5$, $A = 4$; $B = 2$; $C = 3$
if $X = 6$, $A = 4$; $B = 3$; $C = 2$

Neuron Structure model



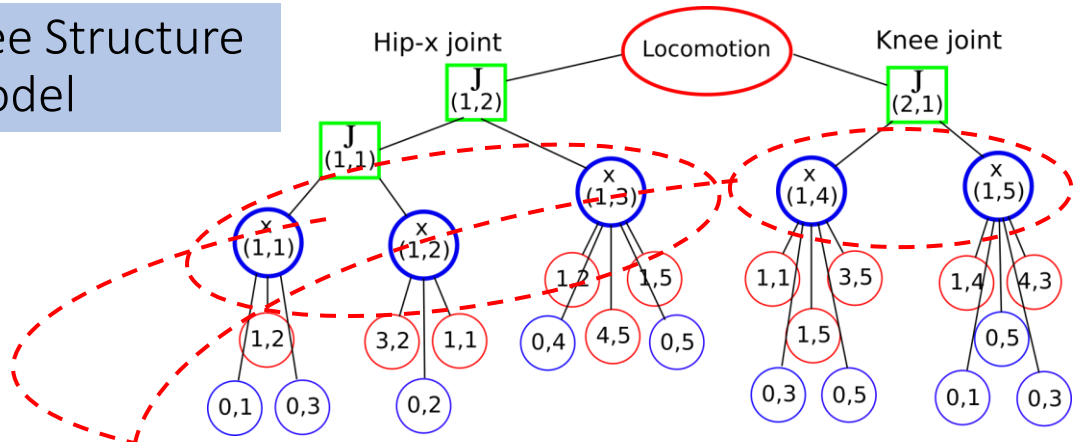
Tree Structure model

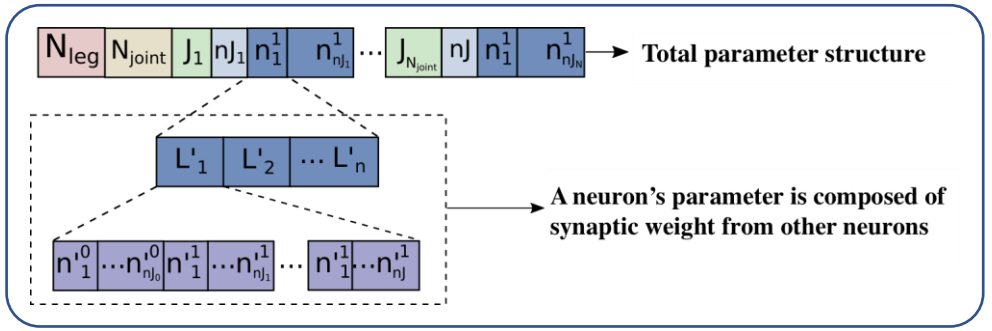
Ex: 1 Neuron in one leg



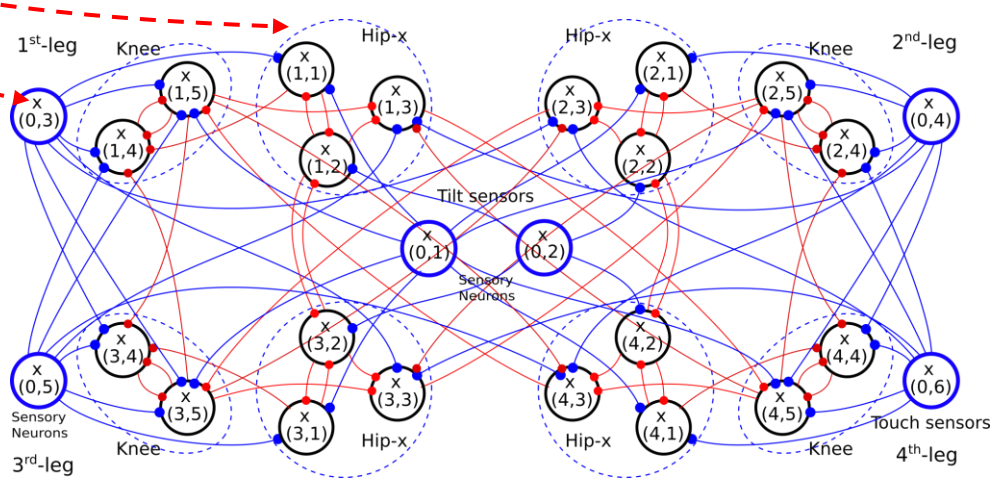
Tree Structure Model

Tree Structure model

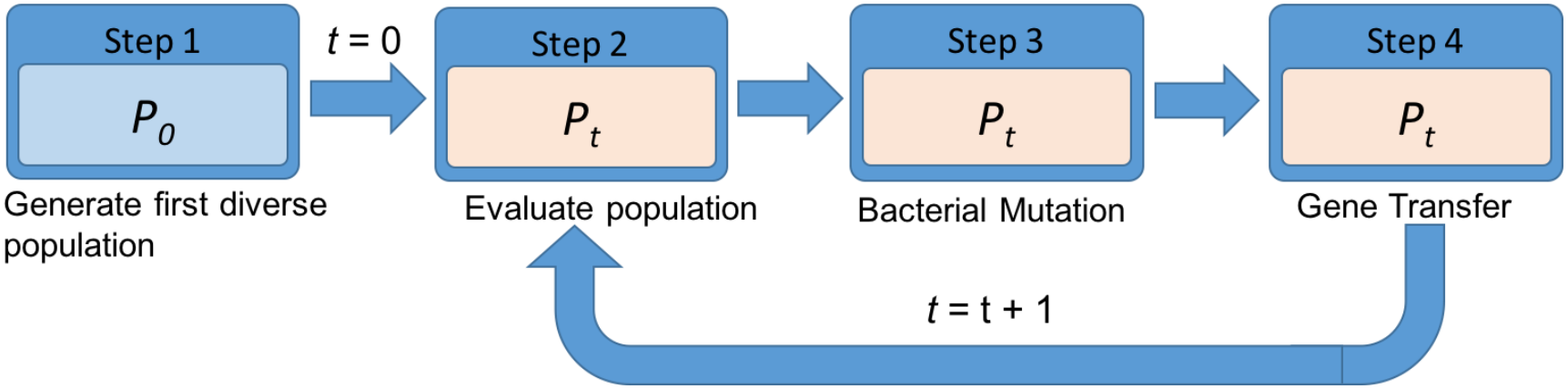




Neuron inter-connection structure



Bacterial Programming



Algorithm 1 Process of Bacterial Programming

```
1: //Generating an initial population:
2: for i ← 1 to N_ind do
3:     Generating tree structure of the ith individual
4:     Evaluating the ith individual
5: //Evolutionary loop:
6: for i ← 1 to N_gen do
7:     Bacterial Mutation operation
8:     Gene Transfer operation
9:     Evaluating the population
```

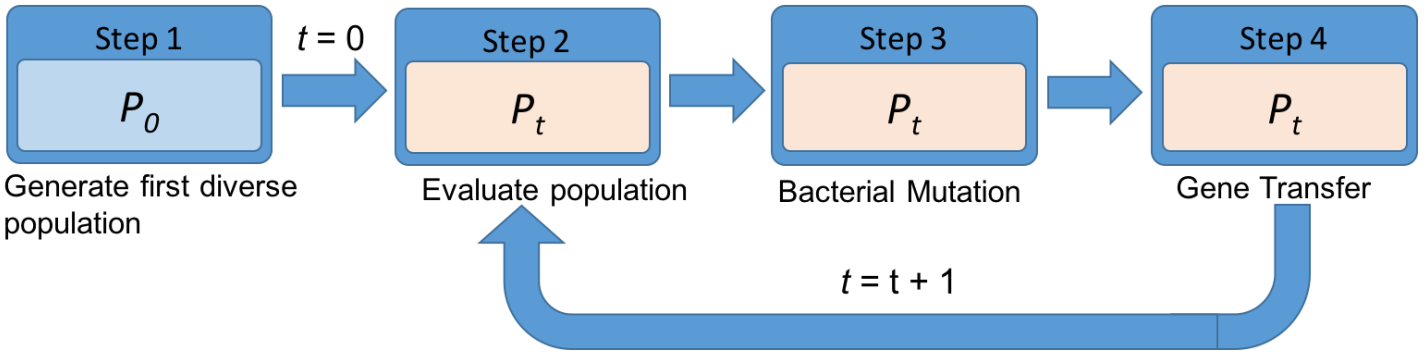
Process of BP

- This algorithm is based on the bacterial operations, however, it uses tree structures similar to the ones in the GP.

- The steps of checking the structure similarities between two bacteria are as follows:
  1. Checking the number of joint nodes at every joint branch
  2. Checking the number of neuron nodes at every neuron branch
  3. Checking the number of neuron leaves at every neuron node
  4. Checking the ID of neuron leaves at every neuron node.

- The similarity check is performed in a hierarchical way. If there is a dissimilarity in a step, we do not continue to the next step. If there is still similarity, next step checking is required.

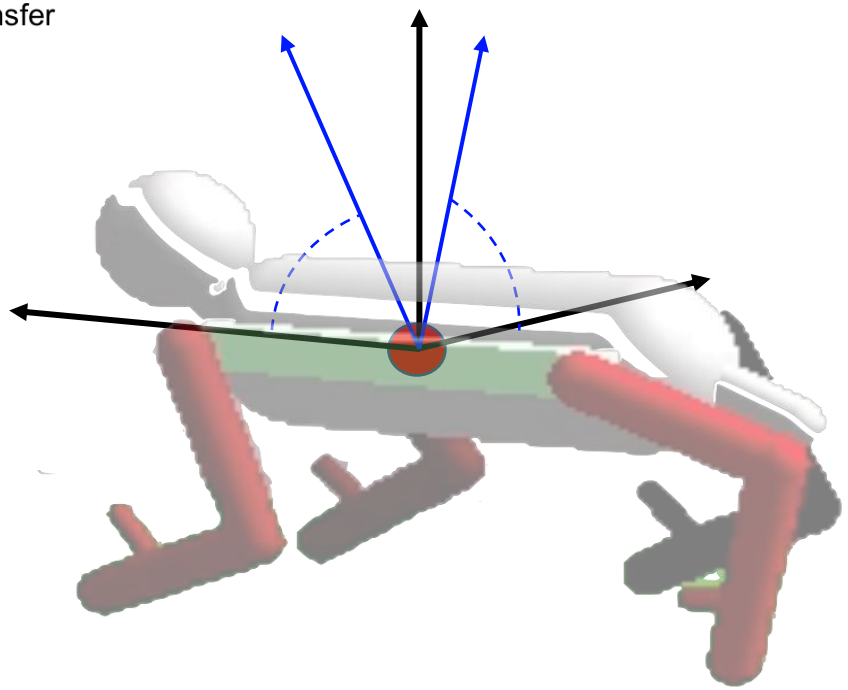
Bacterial Programming - Evaluation



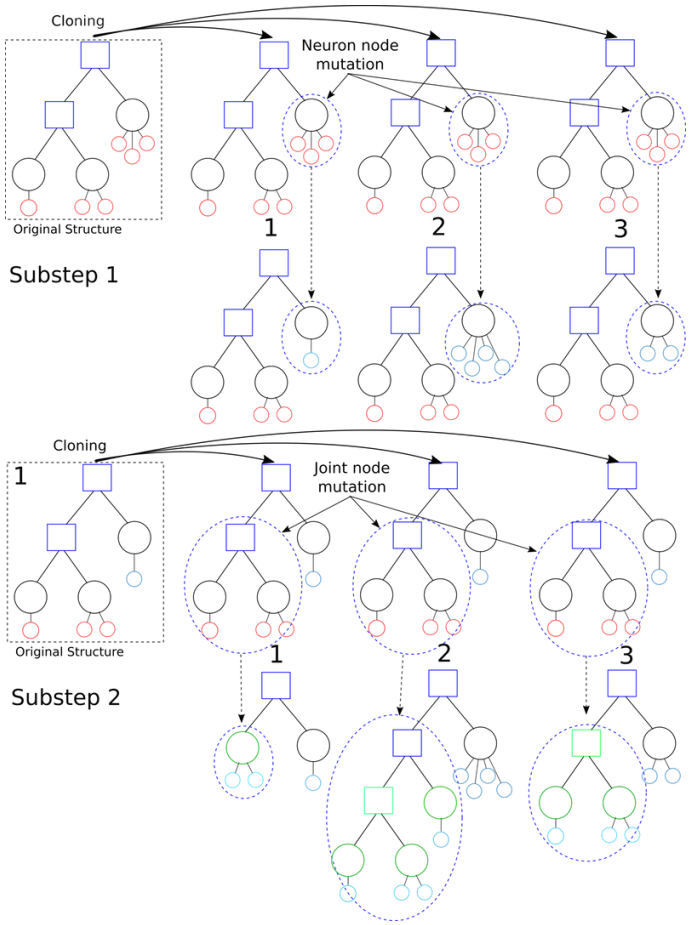
$$\bar{\dot{\beta}} = \frac{1}{T}\sum_{t=0}^{T}\left(\beta_{pitch}(t) + \beta_{roll}(t)\right)$$

$$\bar{v}_{(w_{ij})} = \Upsilon - \frac{1}{T}\sum_{t=0}^{T}\frac{\delta}{\delta t}\ell\left(t, w_{ij}, \mathrm{x}(t), \mathrm{y}(t)\right)$$

$$f = \bar{\dot{\beta}}\alpha_1 + \bar{v}\alpha_2$$



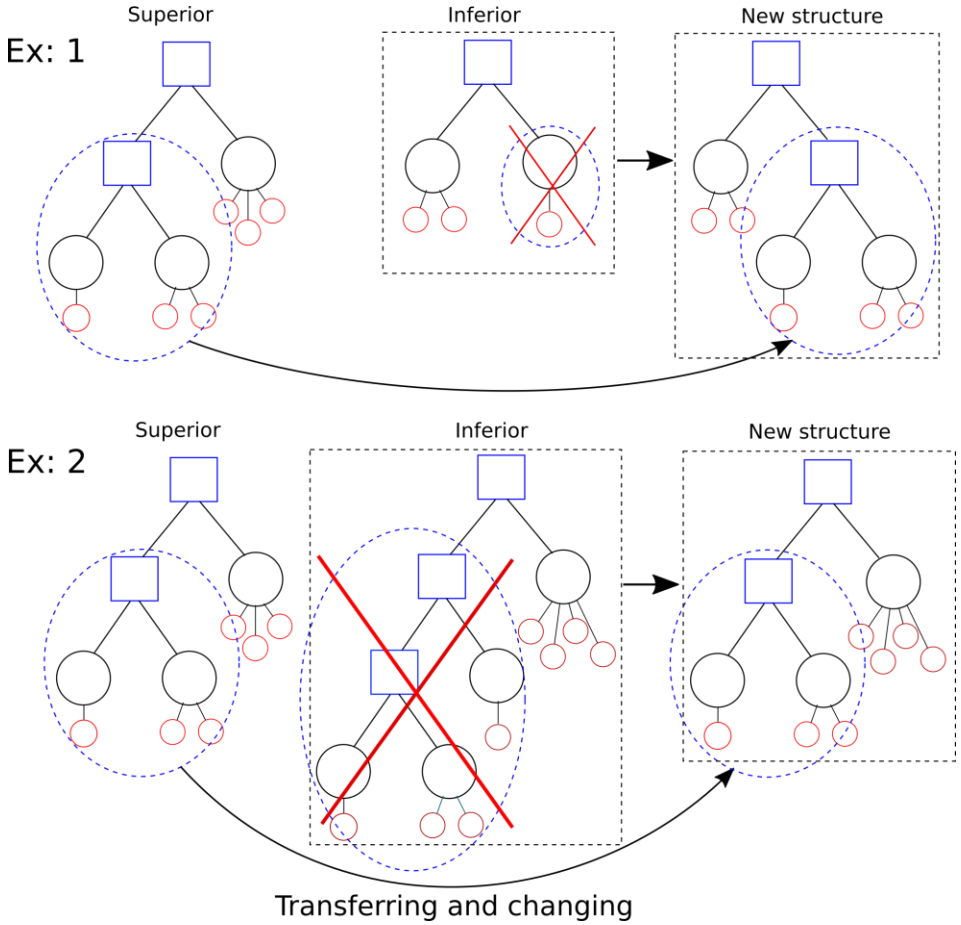
Bacterial Programming - Bacterial Mutation

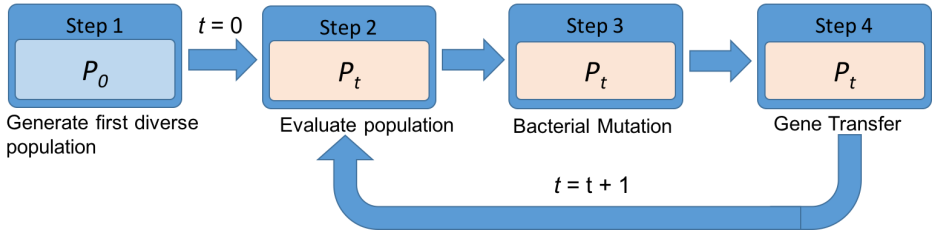


Algorithm 2 Bacterial Mutation Operation

```
for i ← 1 to N_ind do
    Cloning the ith individual
    repeat
        Defining a node randomly
        if node is a Joint node then
            //Joint node mutation in the clones
            for j ← 1 to N_clones do
                Processing Joint node mutation
                Evaluating the mutated clone
        else
            //Neuron node mutation in the clones
            for j ← 1 to N_clones do
                Processing Neuron node mutation
                Evaluating the mutated clone
        Selecting the best individual
        The best individual transfers the mutated subtree
        to the other individuals
    until all subtrees are mutated
    Keeping the best individual as the new ith individual
    Deleting the clones
```

Bacterial Programming - Gene Transfer



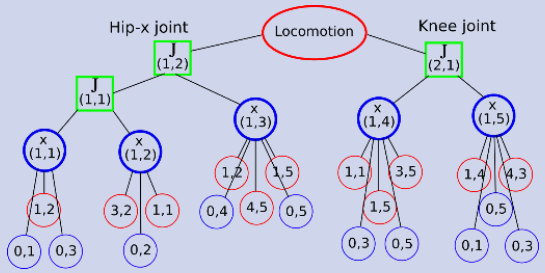
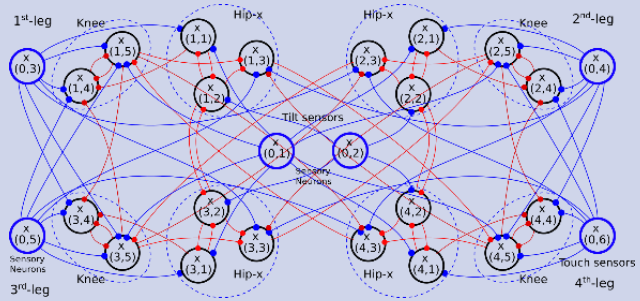


Algorithm 3 Gene Transfer Operation

1: **for** $i \leftarrow 1$ to N_{inf} **do**
2: Ordering the population
3: //Selecting a random superior bacterium
4: $Superior = RandomValue(1, N_{ind}/2)$
5: //Selecting a random inferior bacterium
6: $Inferior = RandomValue(N_{ind}/2+1, N_{ind})$
7: Defining the subtree to be transferred
8: from the Superior bacterium
9: Defining the subtree to be overwritten
10: in the Inferior bacterium
11: Transferring and overwriting
12: Evaluating the new Inferior bacterium

Bacterial Programming – Gene transfer

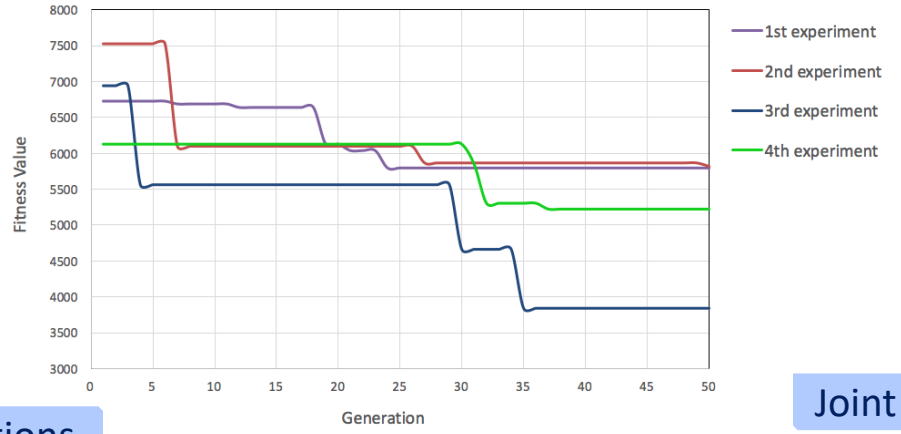
- Gene transfer represents the exchange of genetic information between two bacteria.
- The bacterial population is sorted according to the criteria.
- The source bacterium is randomly chosen from the superior (better performing) half of the population, while the target bacterium is chosen from the inferior (poorer performing) half.
- The source bacterium transfers one of its subtrees to the target bacterium, overwriting its subtree.
- The whole process is repeated (sorting, selection, gene transfer).

Objective analysis	Proposed Locomotion model	Current Locomotion Model
Optimizational model	Tree structure for the optimization strategy 	Neural Network model for the optimization strategy 
Generation of initial population	Diverse generation	Random generation
Number of neurons in one joint	Dynamic	Static
Considered sensors	Tilt, acceleration, ground sensor	Tilt, acceleration, ground sensor
Neural structure model	Adaptive interconnection structure depending on the walking speed, direction, and environmental condition.	Adaptive interconnection structure depending on the walking speed, direction, and environmental condition.

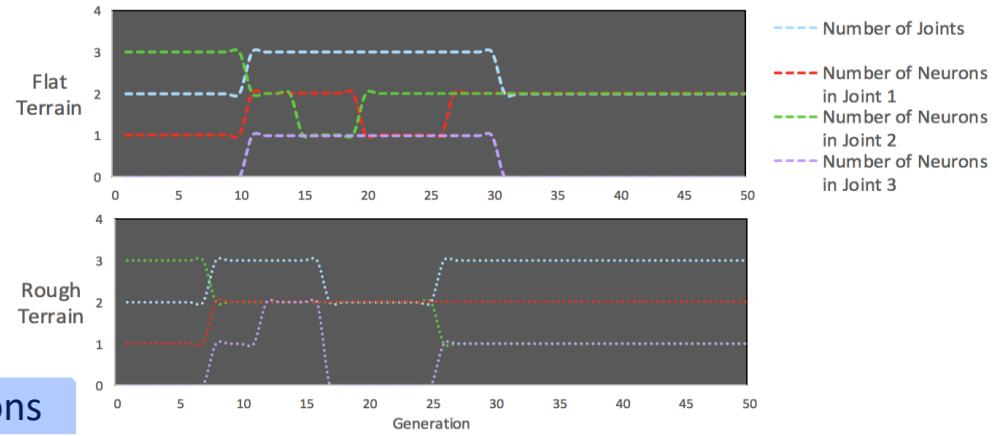
Experimental results – gait optimization

- Two experiments were conducted to validate the proposed model, 1) locomotion optimization 2) its application in the middle size quadruped robot.
- The experiment was conducted on a rough terrain with different slope degrees

Parameter	Value
N_{ind}, N_{gen}, N_{clones}, N_{inf}	100, 50, 10, 30
p^{min}; p^{max}	0; 4
N^{joint}_{min}; N^{joint}_{max}	1; 3
N^{neuron}_{min}; N^{neuron}_{max}	1; 4
α_1; α_2	0.75; 0.25

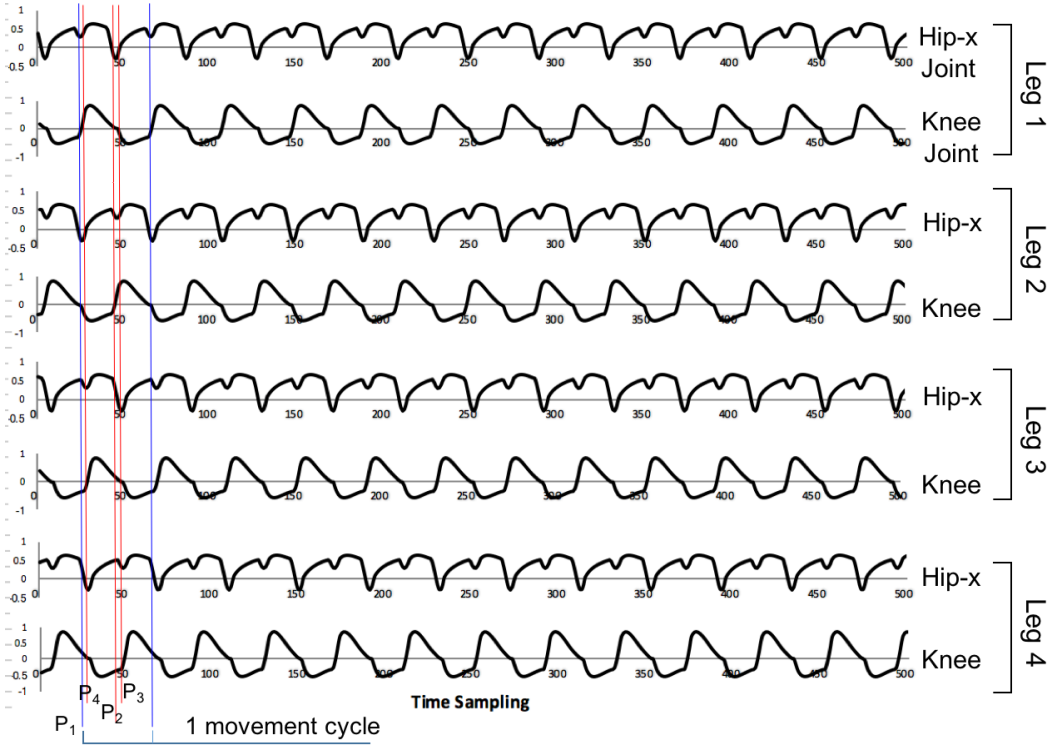


Error evolutions

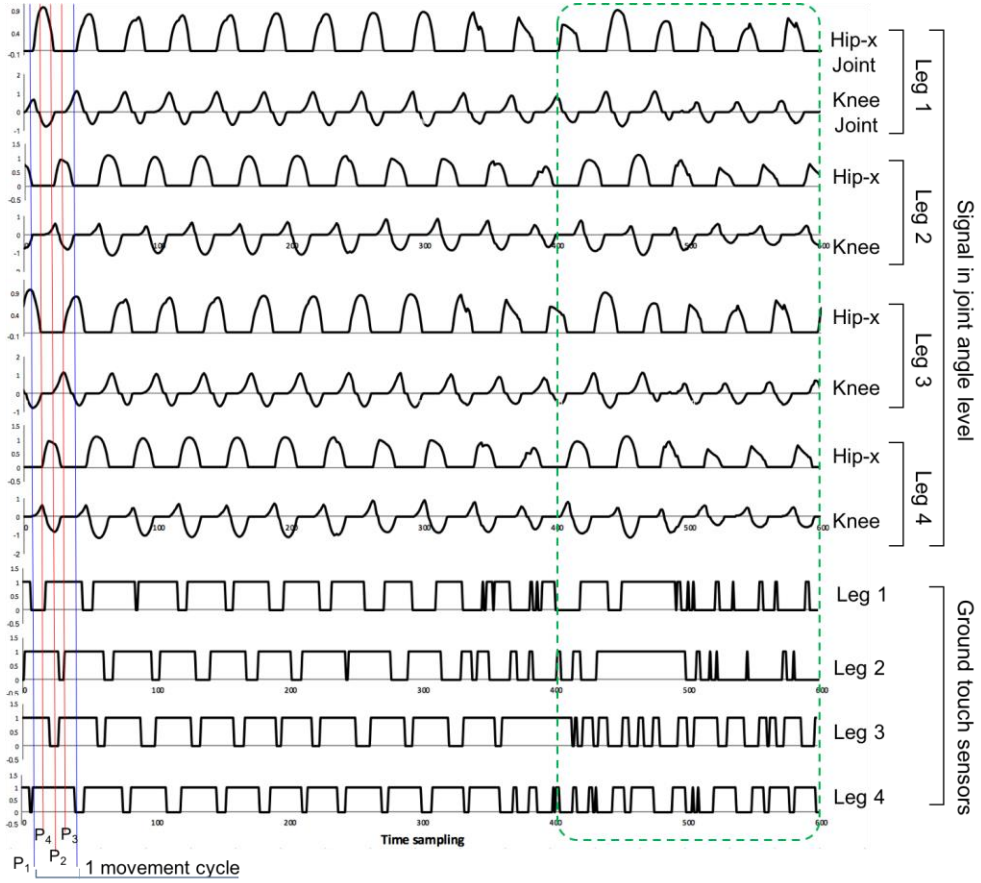


Joint evolutions

Experimental results



Joint angle signal

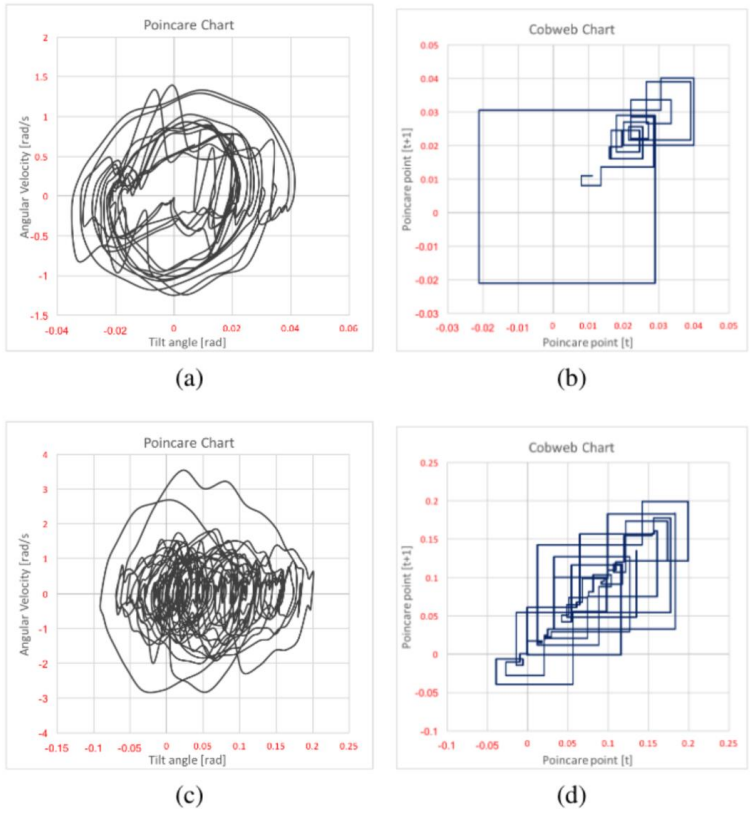


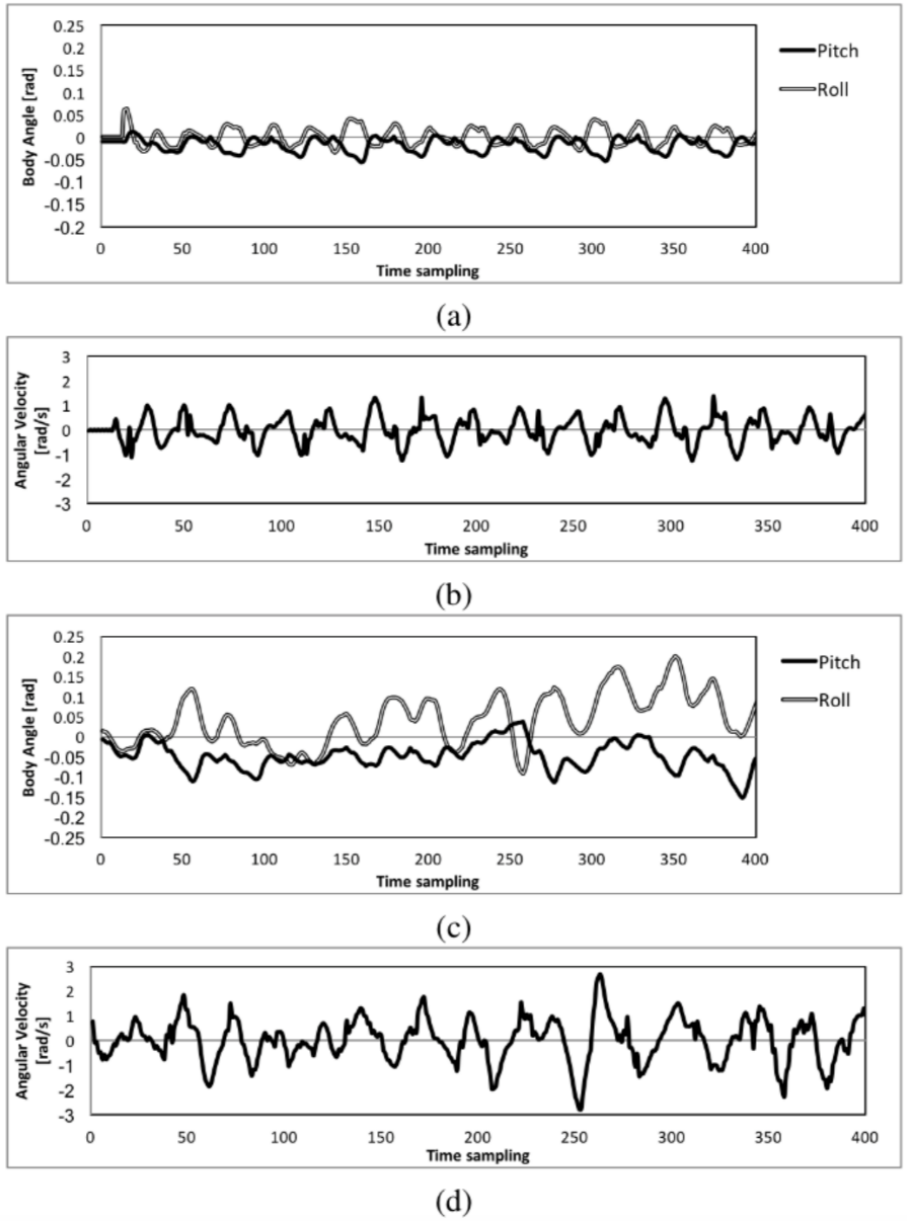
Effect with sensory input

Experimental results

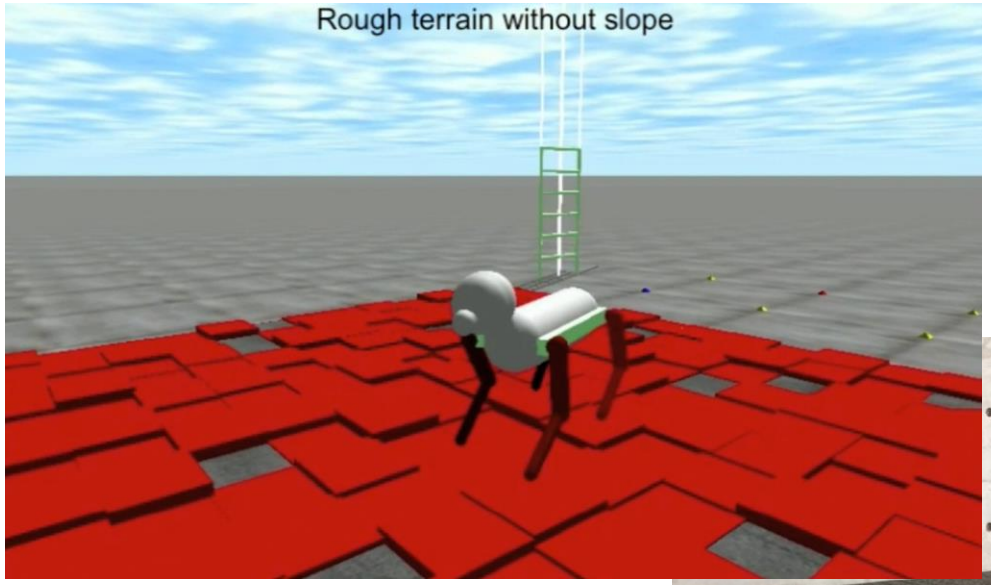
Poincare diagram

Cobweb diagram

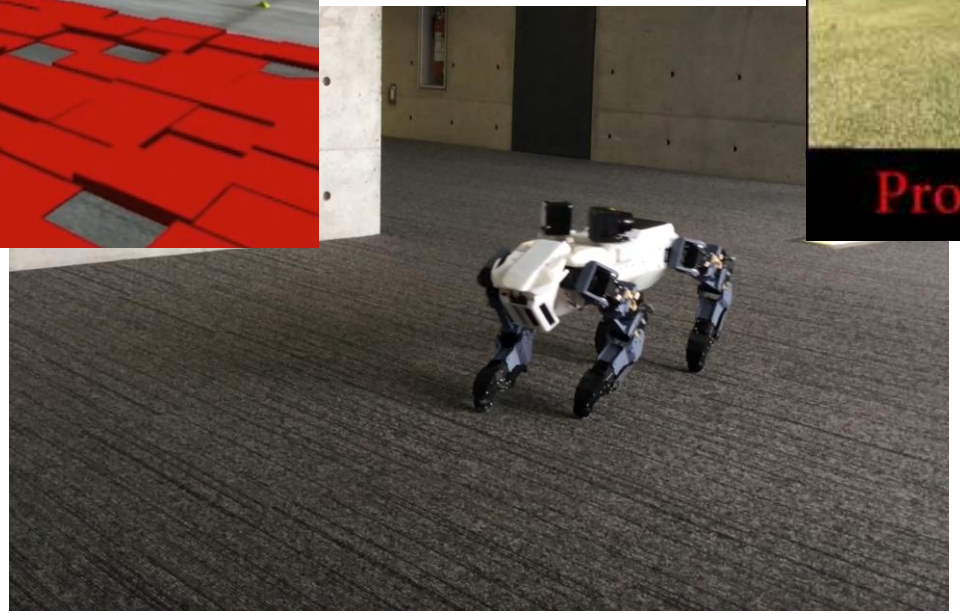


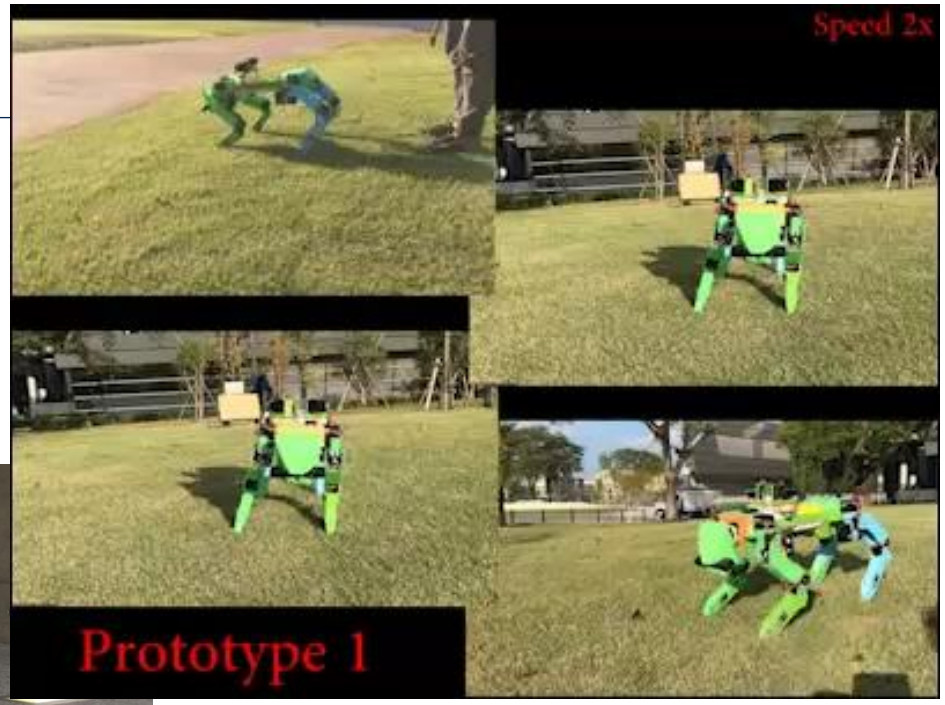


Results



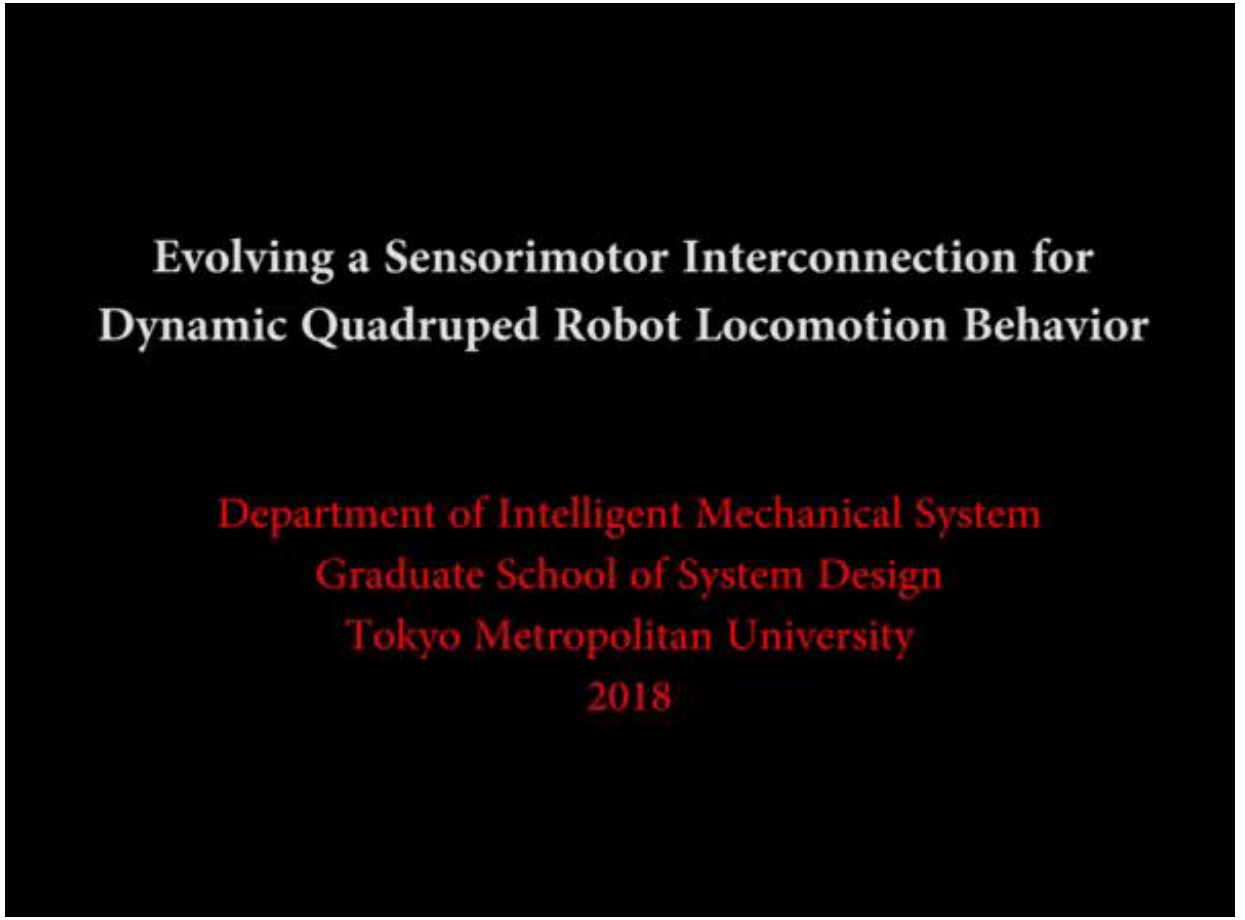
Simulation



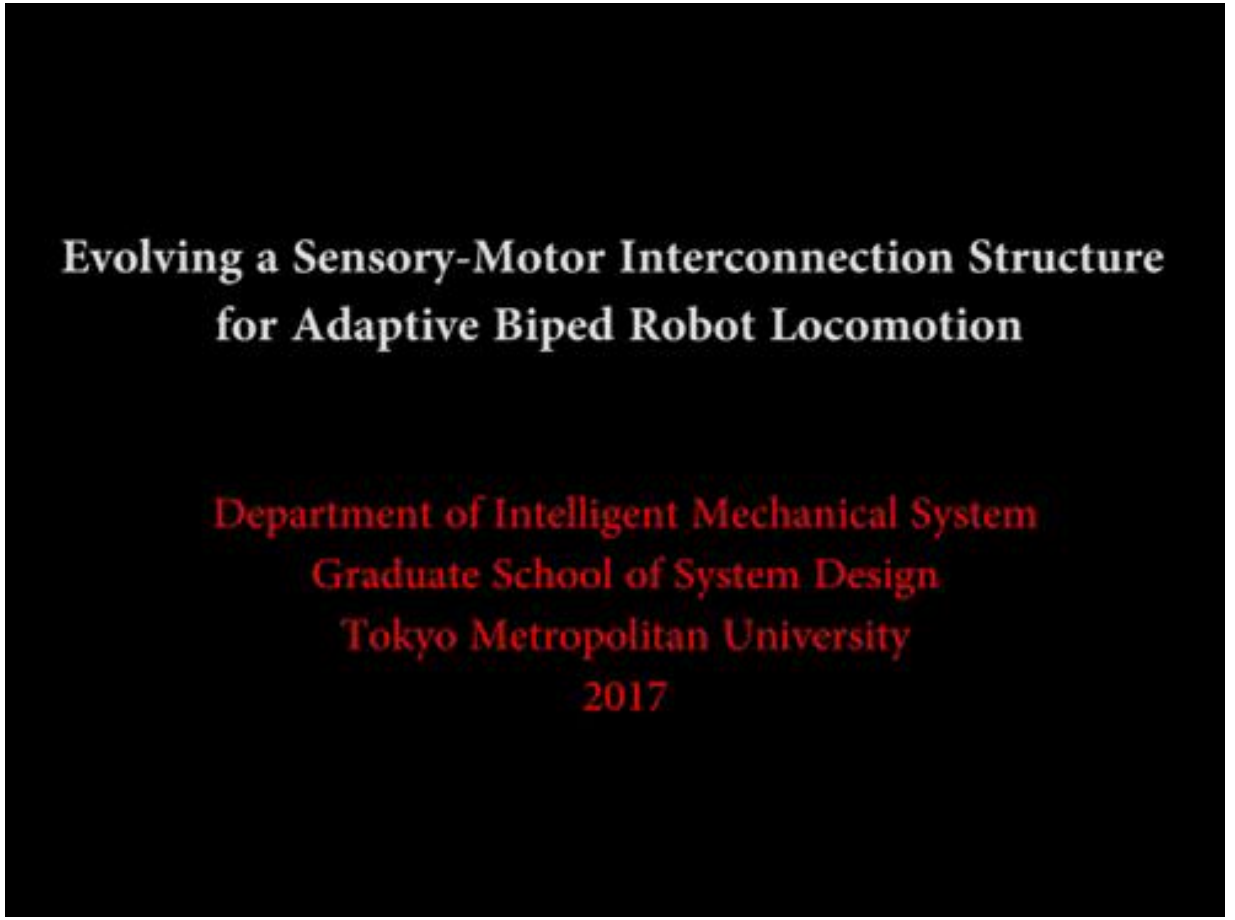


Real experiment

Results


Evolving a Sensorimotor Interconnection for
Dynamic Quadruped Robot Locomotion Behavior
Department of Intelligent Mechanical System
Graduate School of System Design
Tokyo Metropolitan University
2018

Results


Evolving a Sensory-Motor Interconnection Structure
for Adaptive Biped Robot Locomotion
Department of Intelligent Mechanical System
Graduate School of System Design
Tokyo Metropolitan University
2017