Evolutionary algorithms

General optimization algorithms

- Deterministic
  - Calculus based
  - Hill climbing
  - …
- Stochastic
  - Random search
  - Simulated annealing
  - …
- Evolutionary algorithms: Stochastic search methods, computationally simulate the natural evolutionary process using the idea of survival of the fittest

Evolutionary algorithms

- Their basic principle is the search on the population of solutions guided by laws known from biology
- The individuals in the population are the solutions of the given problem
- The population is evolving, we obtain better and better individuals

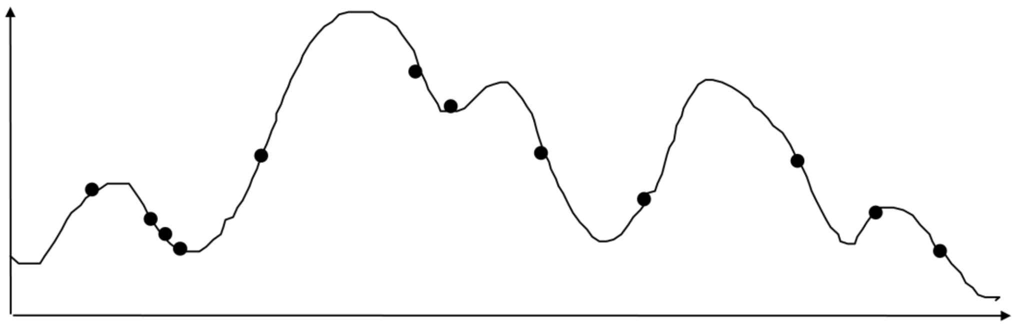
Evolutionary computation - history

- The idea of using simulated evolution to solve engineering and design problems have been around since the 1950’s
- However, it wasn’t until the early 1960’s that we began to see three influential forms of EA emerge:
    - Evolutionary programming (Lawrence Fogel, 1962)
    - Genetic algorithms (Holland, 1975)
    - Evolution strategies (Rechenberg, 1965 & Schwefel, 1968)
- The designers of each of the EA techniques saw that their particular problems could be solved via simulated evolution
    - Fogel was concerned with solving prediction problems
    - Rechenberg & Schwefel were concerned with solving parameter optimization problems
    - Holland was concerned with developing robust adaptive systems

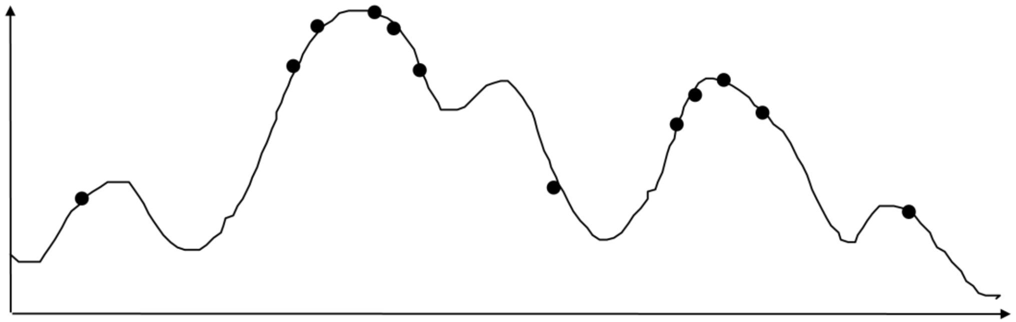
Terminology

- Gene: functional entity that encodes a specific feature of the individual (e.g. hair color)
- Allele: value of gene (e.g. blonde)
- Genotype: the specific combination of alleles carried by an individual
- Phenotype: the physical makeup of an organism
- Locus: position of the gene within the chromosome
- Individual: chromosome, represents a candidate solution for the problem
- Population: collection of individuals currently alive

Evolution of the population



Distribution of Individuals in Generation 0



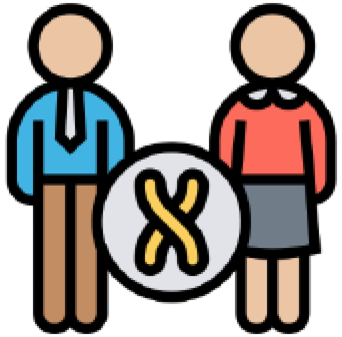
Distribution of Individuals in Generation N

Genetic algorithm

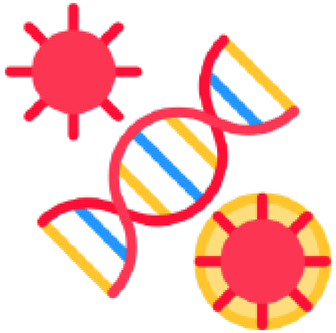
- Population-based optimization method
- Inspired by Darwinian theory of evolution
- Stochastic in nature
- Utilizes 3 bio-inspired operators

Selection

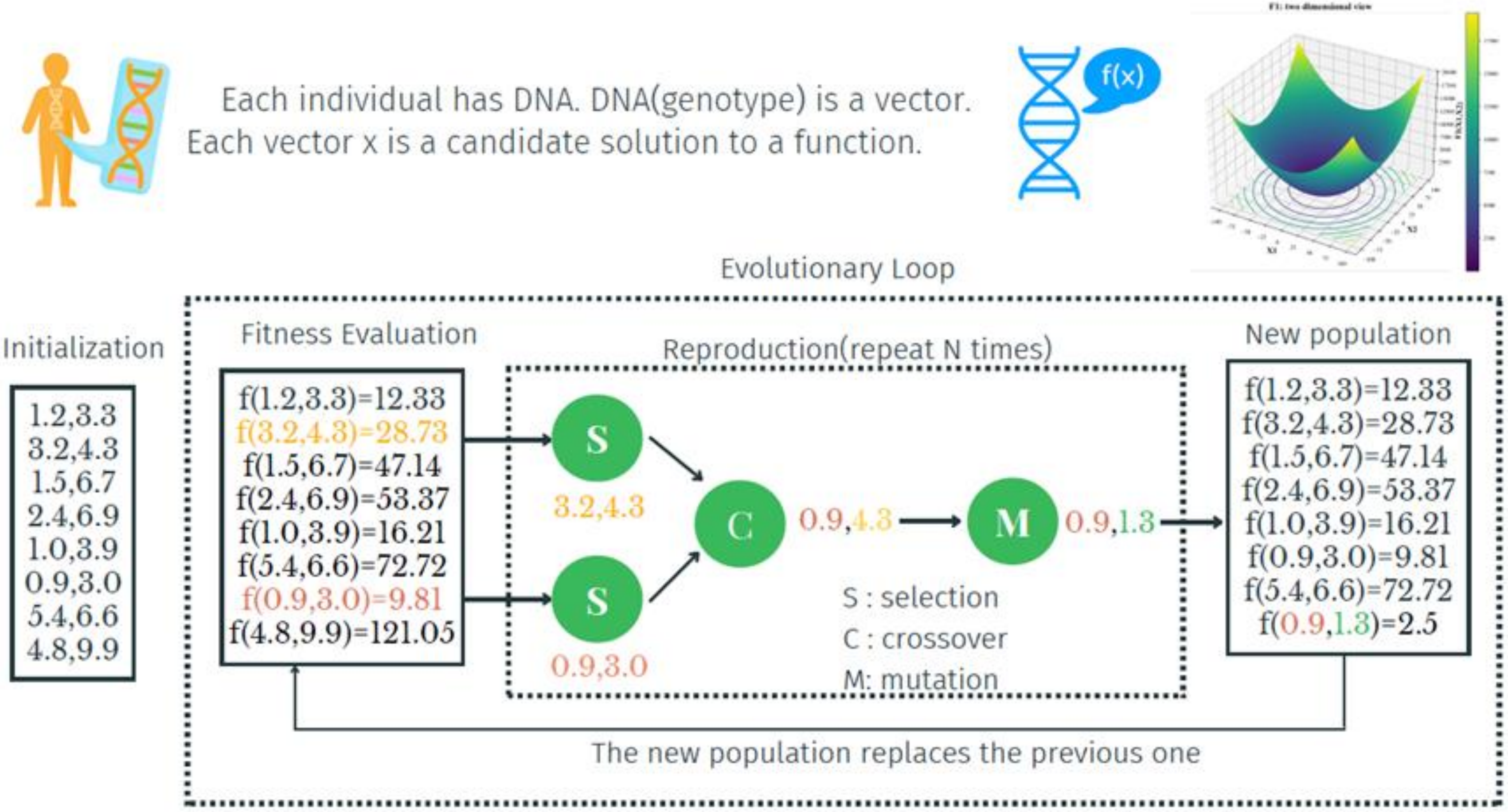
Crossover



Mutation



Genetic algorithms

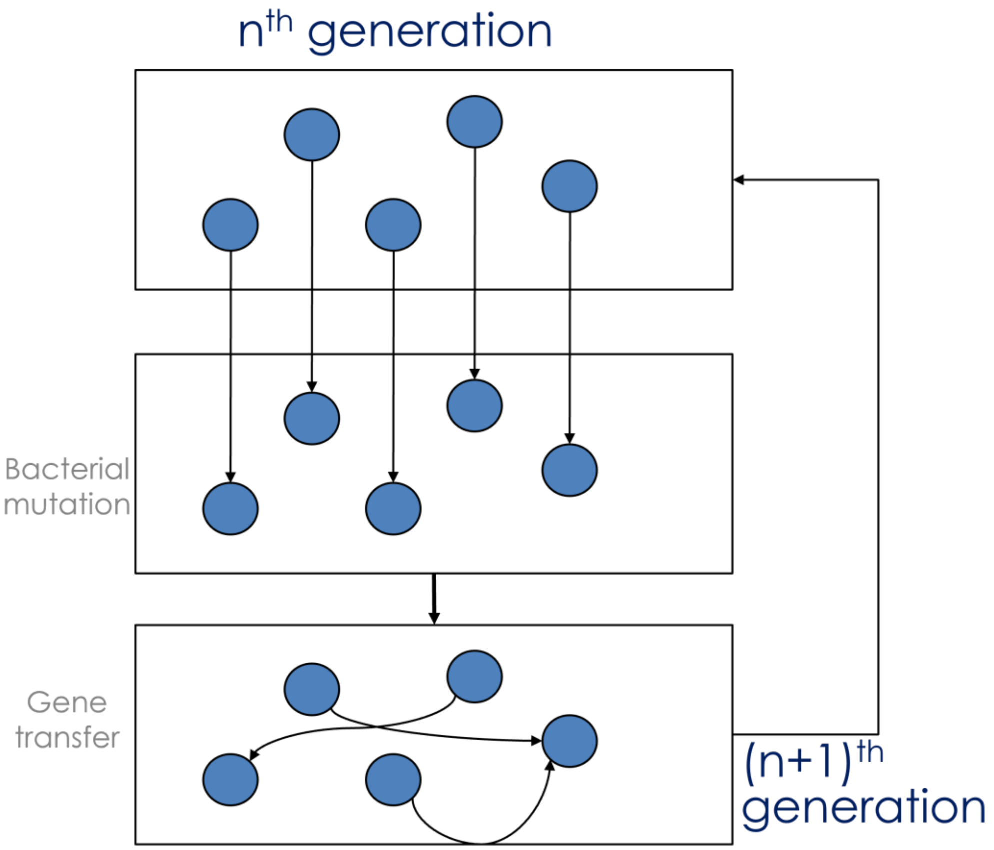


Bacterial evolutionary algorithms

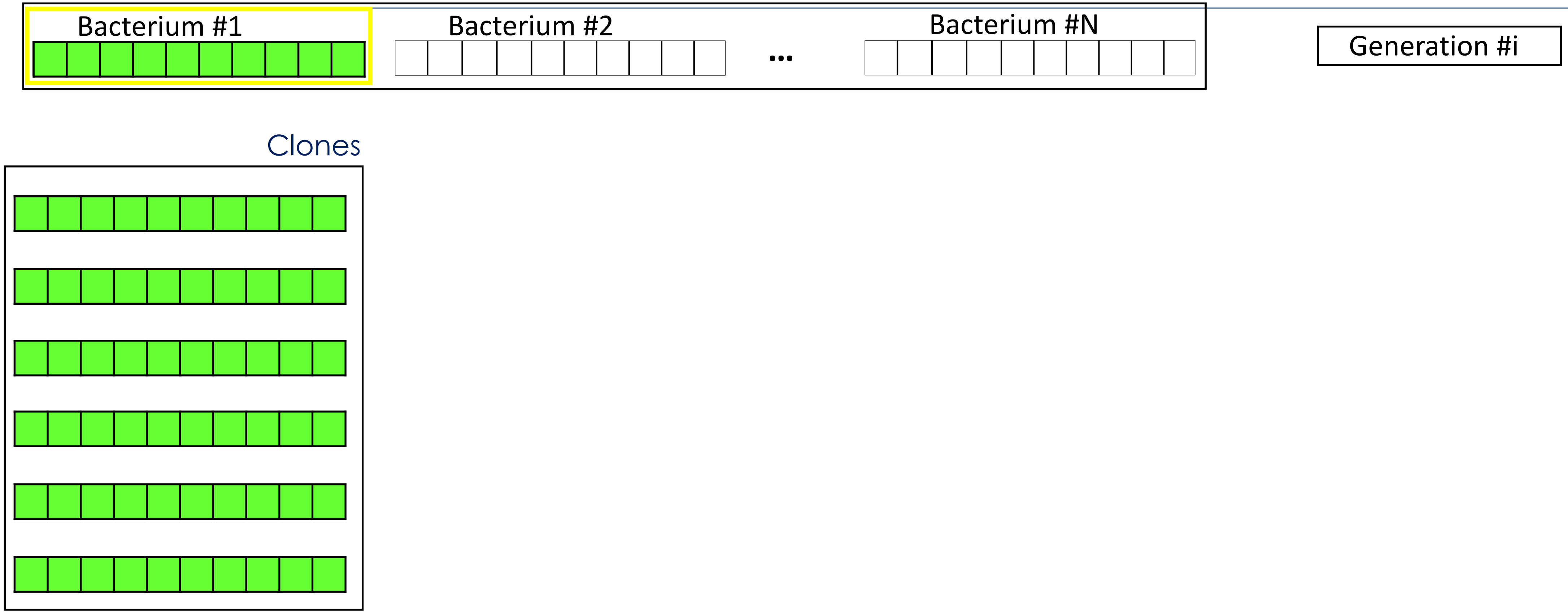
- Nature inspired optimization techniques
- Based on the process of microbial evolution
- Applicable for complex optimization problems
- Individual: one solution of the problem
- Intelligent search strategy to find *sufficiently good* solution (quasi optimum)
- Fast convergence (conditionally)

The algorithm

- Generating the initial population randomly
- Bacterial mutation is applied for each bacterium
- Gene transfer is applied in the population
- If a stopping condition is fullfilled then the algorithm stops, otherwise it continues with the bacterial mutation step



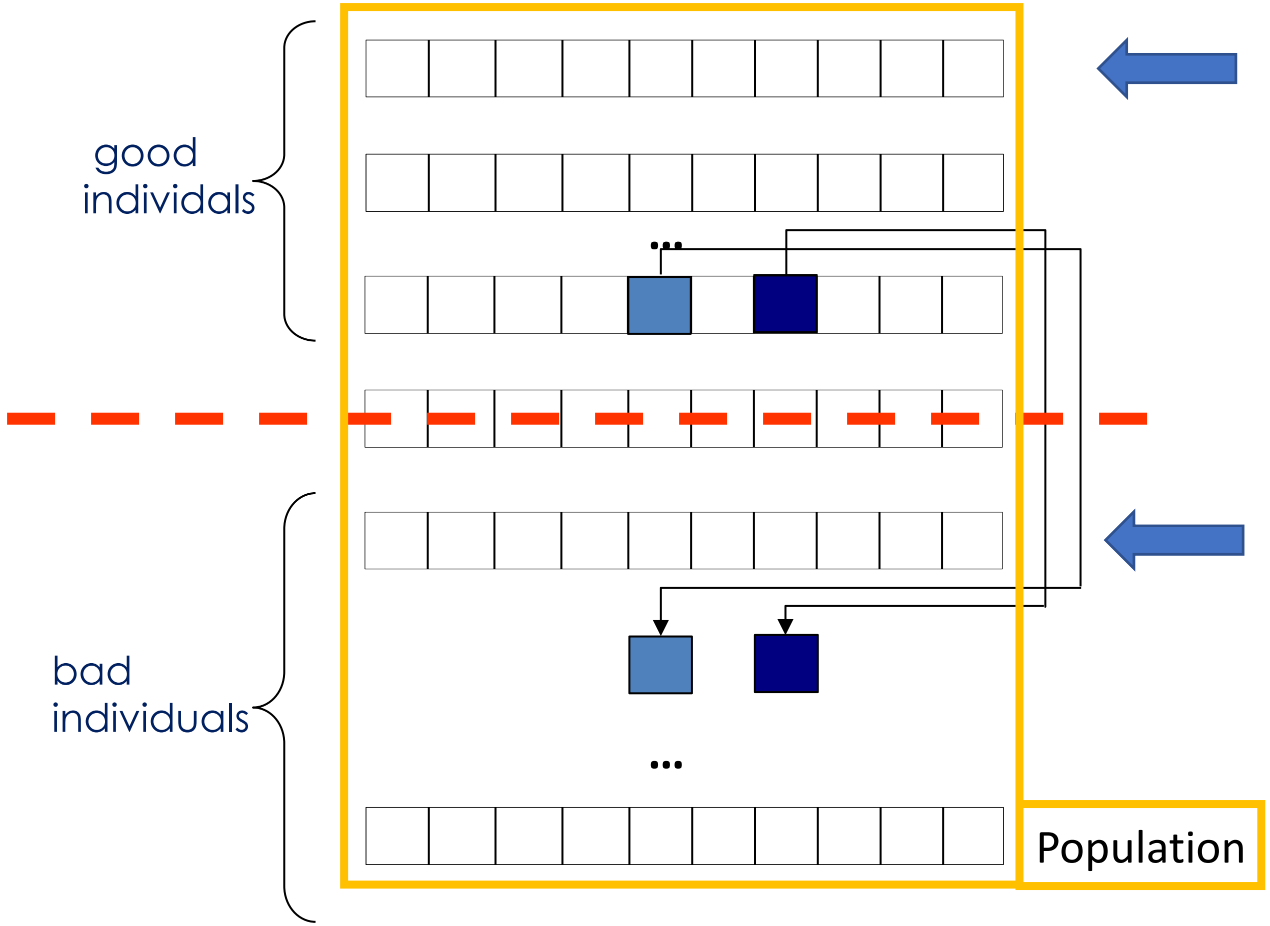
Bacterial Mutation



Gene transfer

1. Divide the population into two parts (superior and inferior)
2. Select one bacterium from the superior subpopulation (donor) and select one from the inferior subpopulation (acceptor)
3. Replace some of the acceptor’s genes with the donor’s genes

Repeat this cycle N_{inf} times
(Number of “infection”)



Parameters

- N_{gen}: number of generations
- N_{ind}: number of individuals
- N_{clone}: number of copies (clones) in the bacterial mutation
- N_{inf}: number of gene transfers (infections) in gene transfer

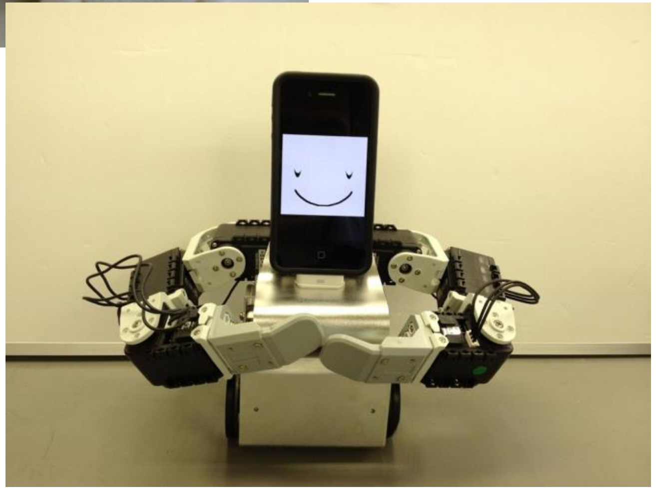
Cognitive model of the iPhonoid robot partner

Introduction

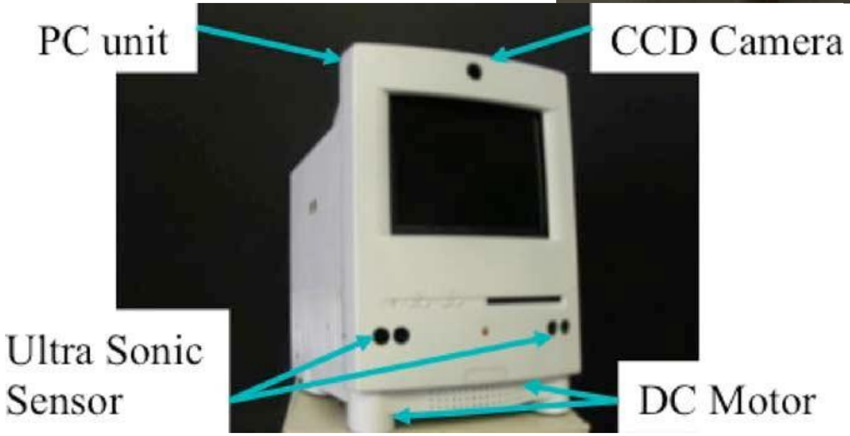
Kubota lab

Animaloid

MOBiFace

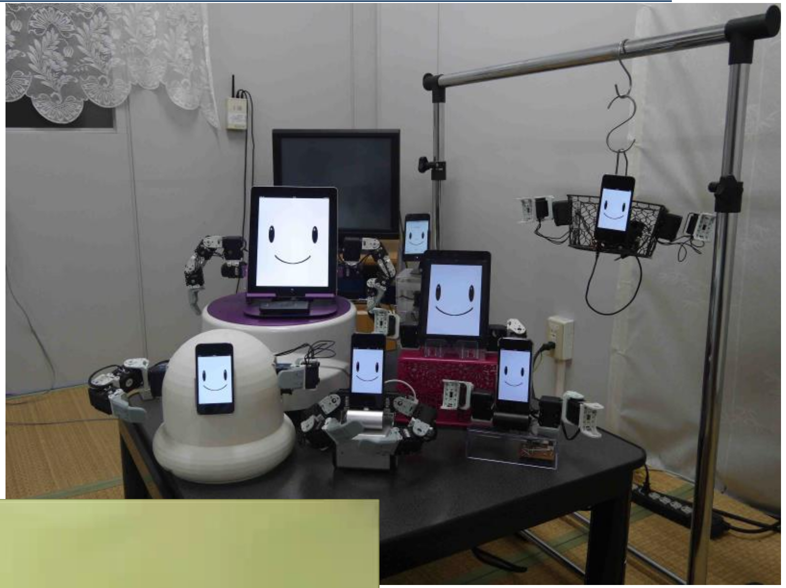


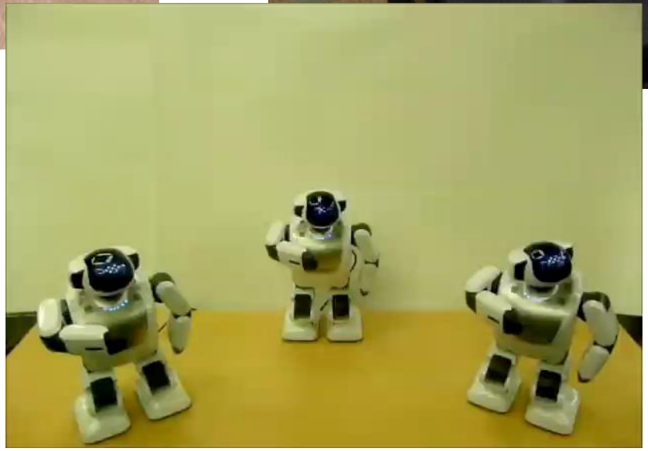
iPhonoid



MOBiMac

Kubota lab





History of iPhonoid

Fig 1. iPhonoid
(1st generation)

Fig 2. Cargonoid
(2nd generation)

Fig 3. iPhonoid
(2nd generation)

Fig 4. iPhonoid-B
(3rd generation)

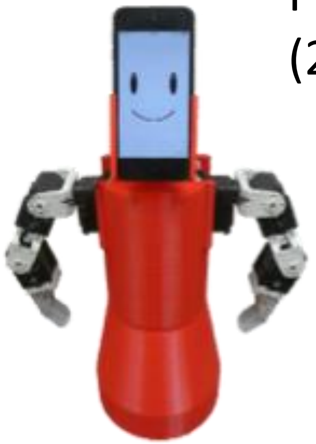


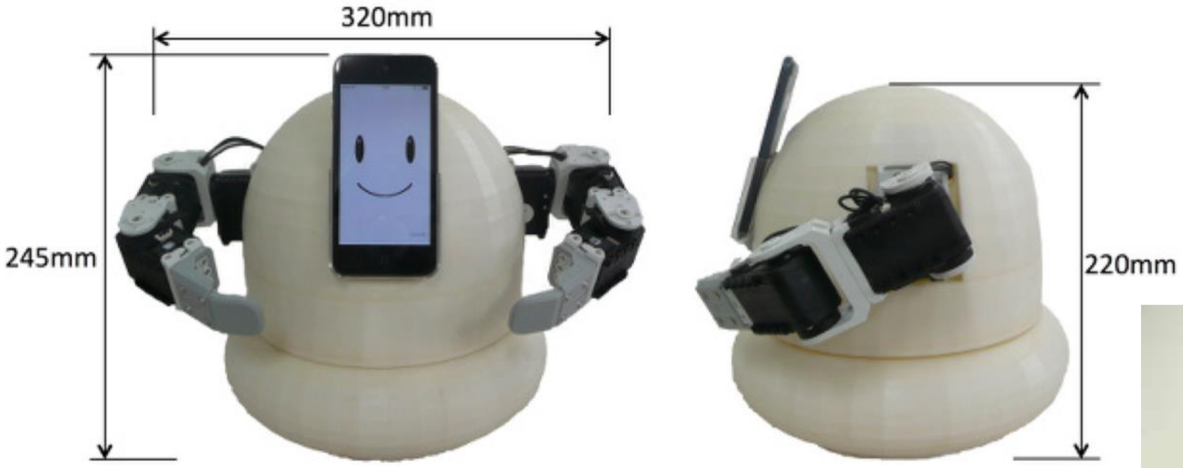
Fig 5. iPhonoid-C
(3rd generation)

Fig 6. iPhonoid (New)
(4th generation)

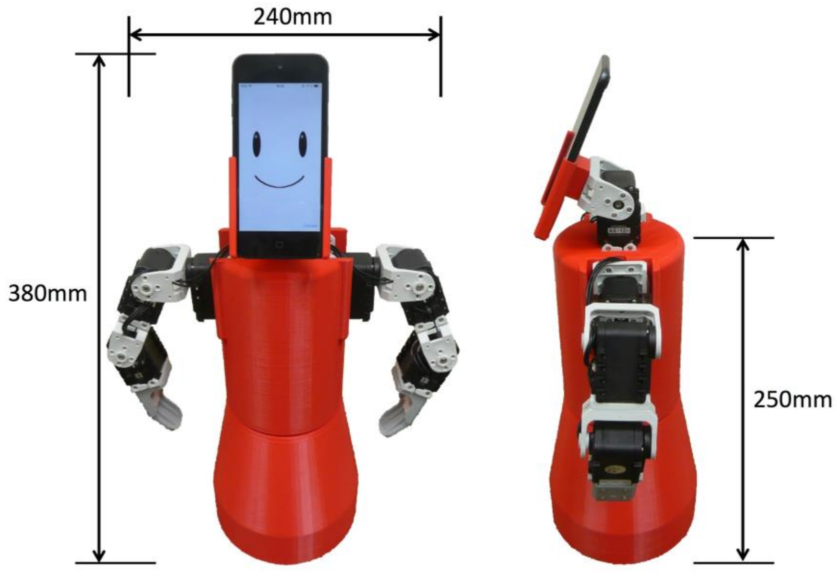
History of iPhonoid

	iPhonoid (Series A) (1st generation) 2010~2011	Cargonoid (Series A) (2nd generation) 2012~2013	iPhonoid (Series A) (2nd generation) 2012~2013	iPhonoid-B (3rd generation) 2014	iPhonoid-C (3rd generation) 2014	iPhonoid-D (3rd generation) 2015
Appearance						
Communication methods	Wireless LAN	Earphone signal or Bluetooth	Wired serial connection	Bluetooth	Bluetooth	Bluetooth
Movable mechanism	Arm	Arm	Arm・Waist	Arm・Waist	Neck・Arm・Waist	Neck・Arm・Waist
Degrees of freedom	3 (Movement) 6 (Arms)	4 (Arms)	6 (Arms) 1 (Waist)	6 (Arms) 1 (Waist)	1 (Neck) 6 (Arms) 1 (Waist)	1 (Neck) 6 (Arms) 1 (Waist)
Size [H x W x D]	23 cm x 25 cm x 13 cm	10 cm x 36 cm x 12.5 cm	37 cm x 26 cm x 12 cm	24.5 cm x 32 cm x 23 cm	38 cm x 24 cm x 24 cm	42.7 cm x 23 cm x 19 cm
Weight	Approx. 1.3kg	Approx. 1kg	Approx. 1kg	Approx. 1.6kg	Approx. 1.2kg	Approx. 1.1kg
Feature	Designed for moving in environment.	The robot can search human by using waist part.	The robot can search human by using waist part.	The robot can search human by using waist part.	The robot can search human by using neck and waist part.	Robot is designed by Mr. Mikael Jacquemont

Specification of iPhonoid

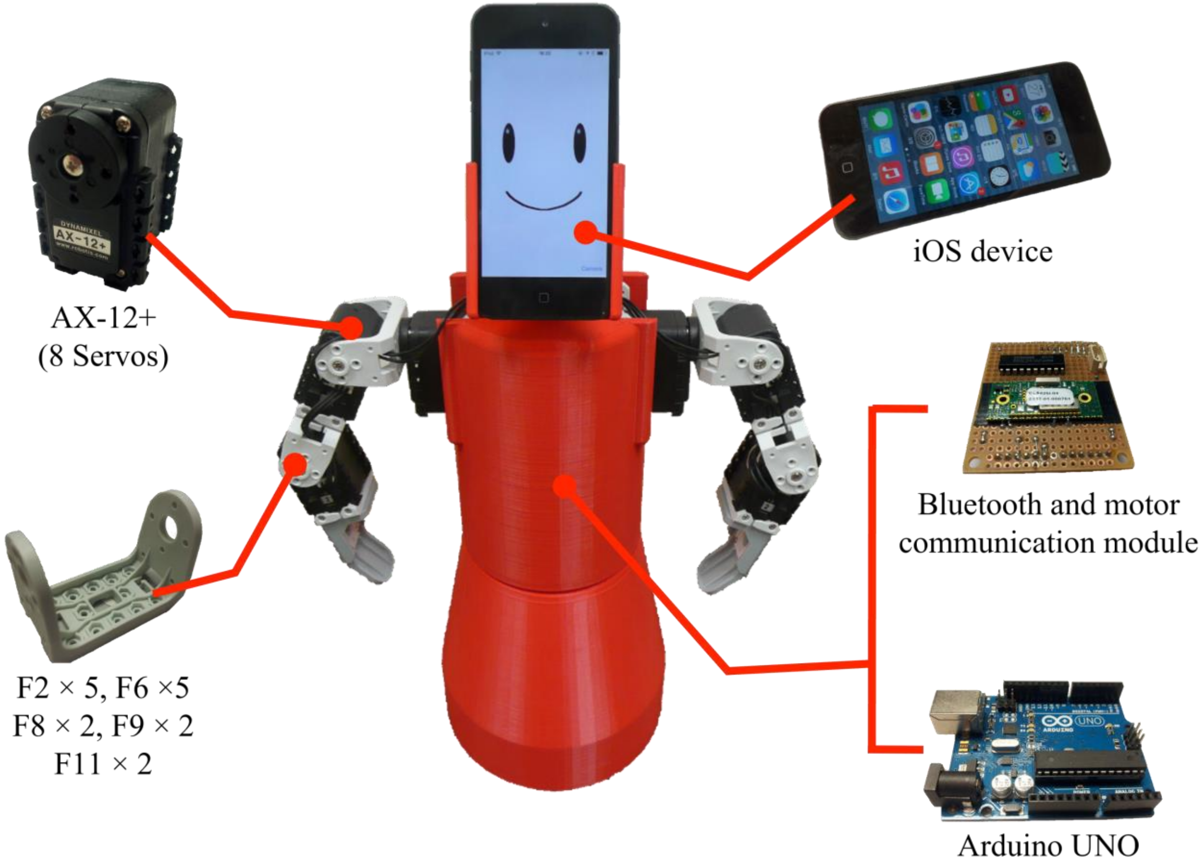


iPhonoid-B

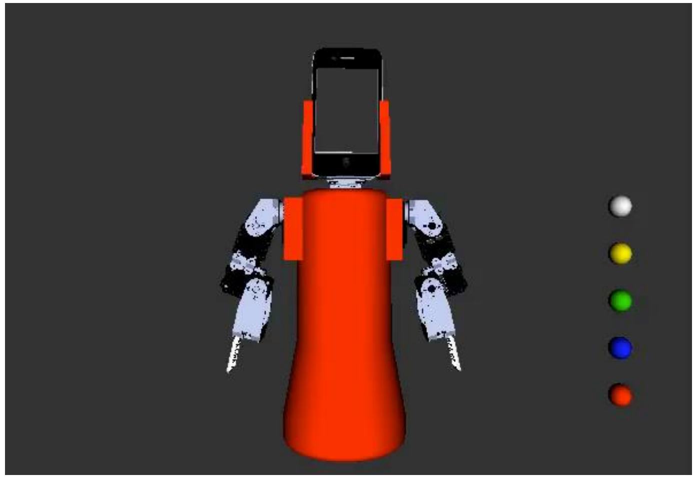


Example of demonstration

Specification of iPhonoid

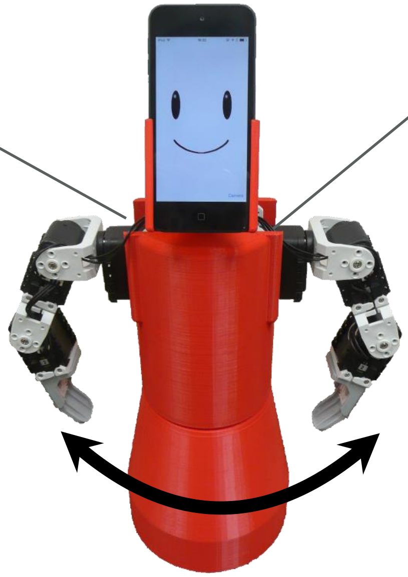


iPhonoid-C Design

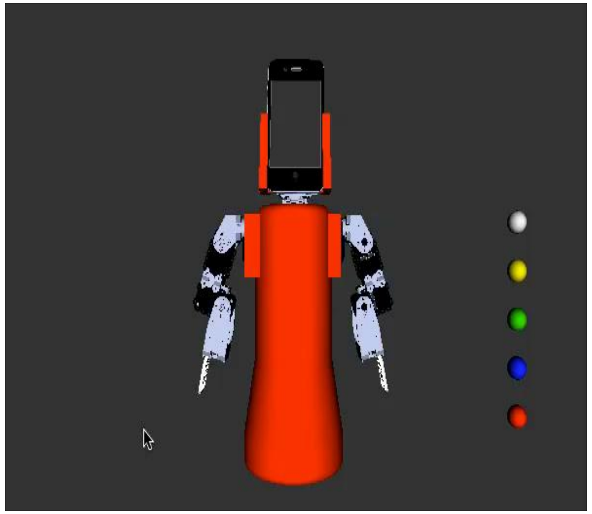


by Prof. Boris Tudjarov and Yusuf Bakhtiar

Panoramic photography using the waist joint

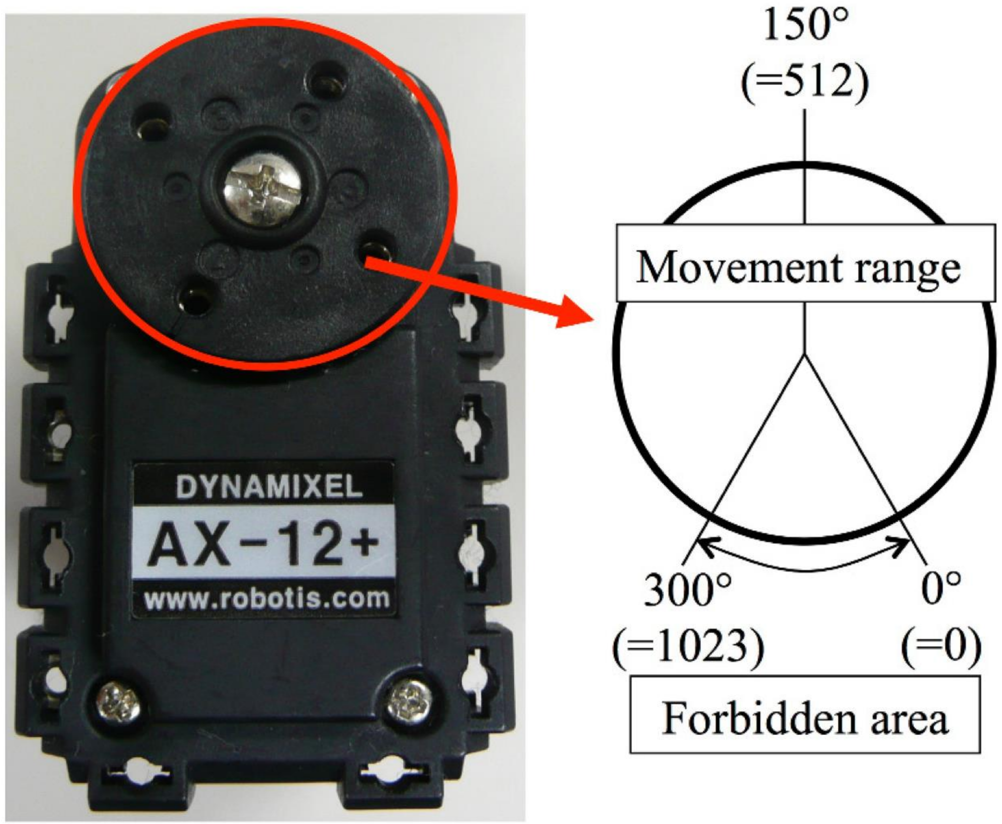


Web-based interaction

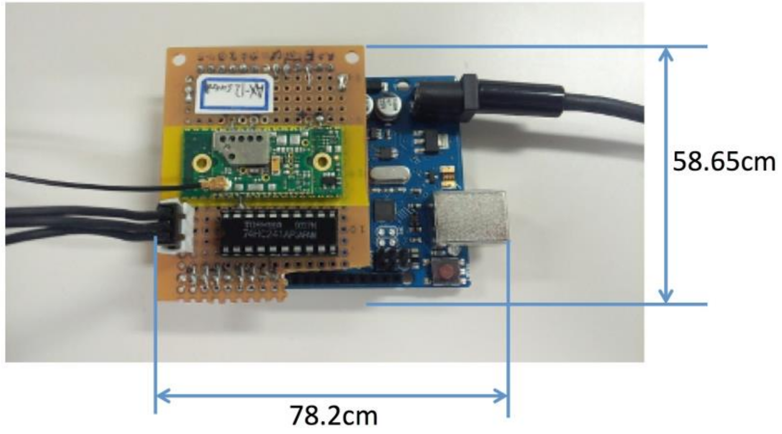


Specification of iPhonoid

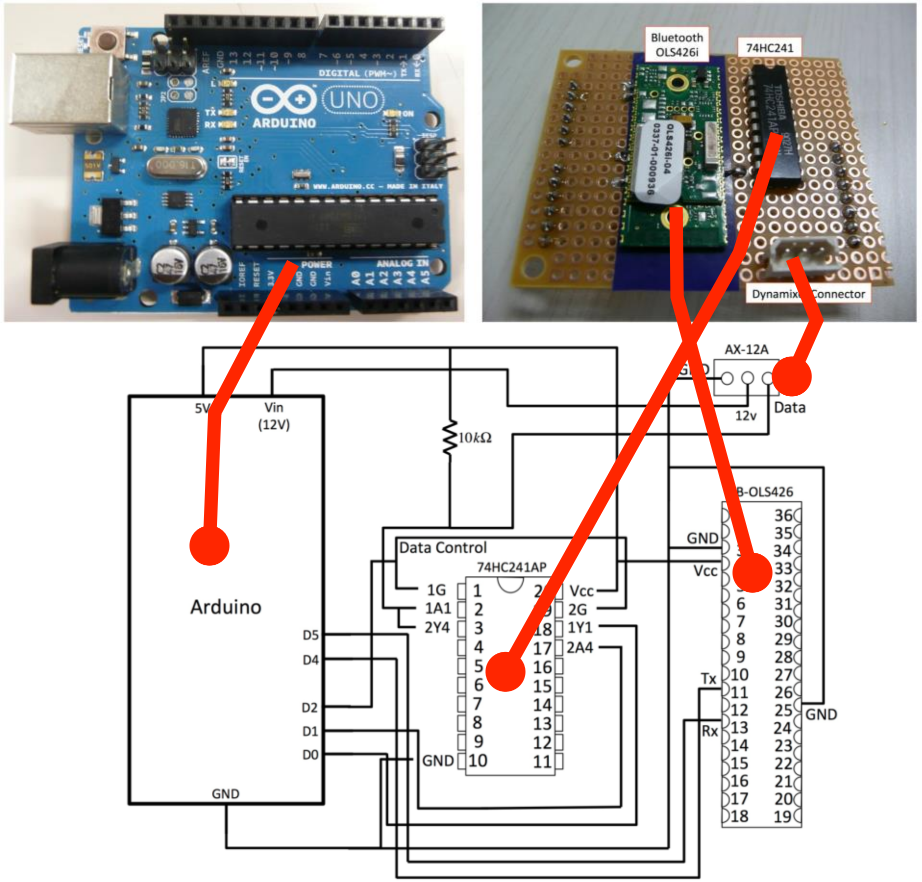
- Dynamixel AX-12 is a modular actuator from ROBOTIS that contains speed reducer, drivers, network function, and control unit
- The control range is between 0 and 300 degree, which can be encoded by integer values between 0 and 1023
- 7 motors are applied for realizing the 7 DOF of the robot for iPhonoid-B, and 8 motors for iPhonoid-C



Specification of iPhonoid



After connection

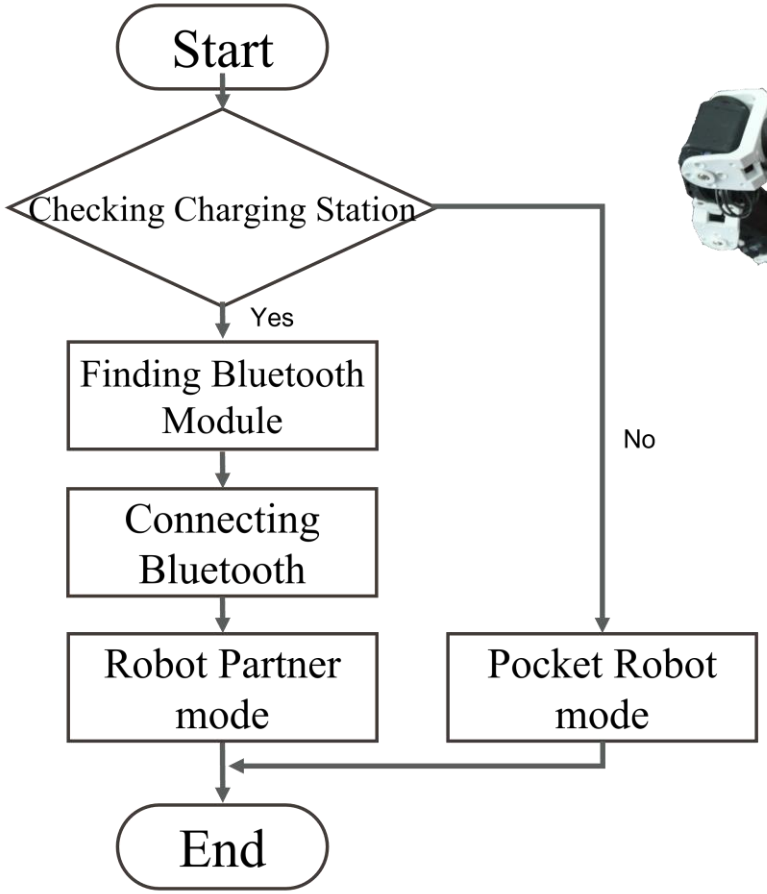


Microcontroller and Bluetooth module circuit for AX-12 control

- Arduino is a single-board microcontroller based on open source technology
- connectBlue OLS426 is applied as a bluetooth connection module
- 74HC241AP chip is applied to control the control signal for AX-12 motors

Device Selection for Robot Control

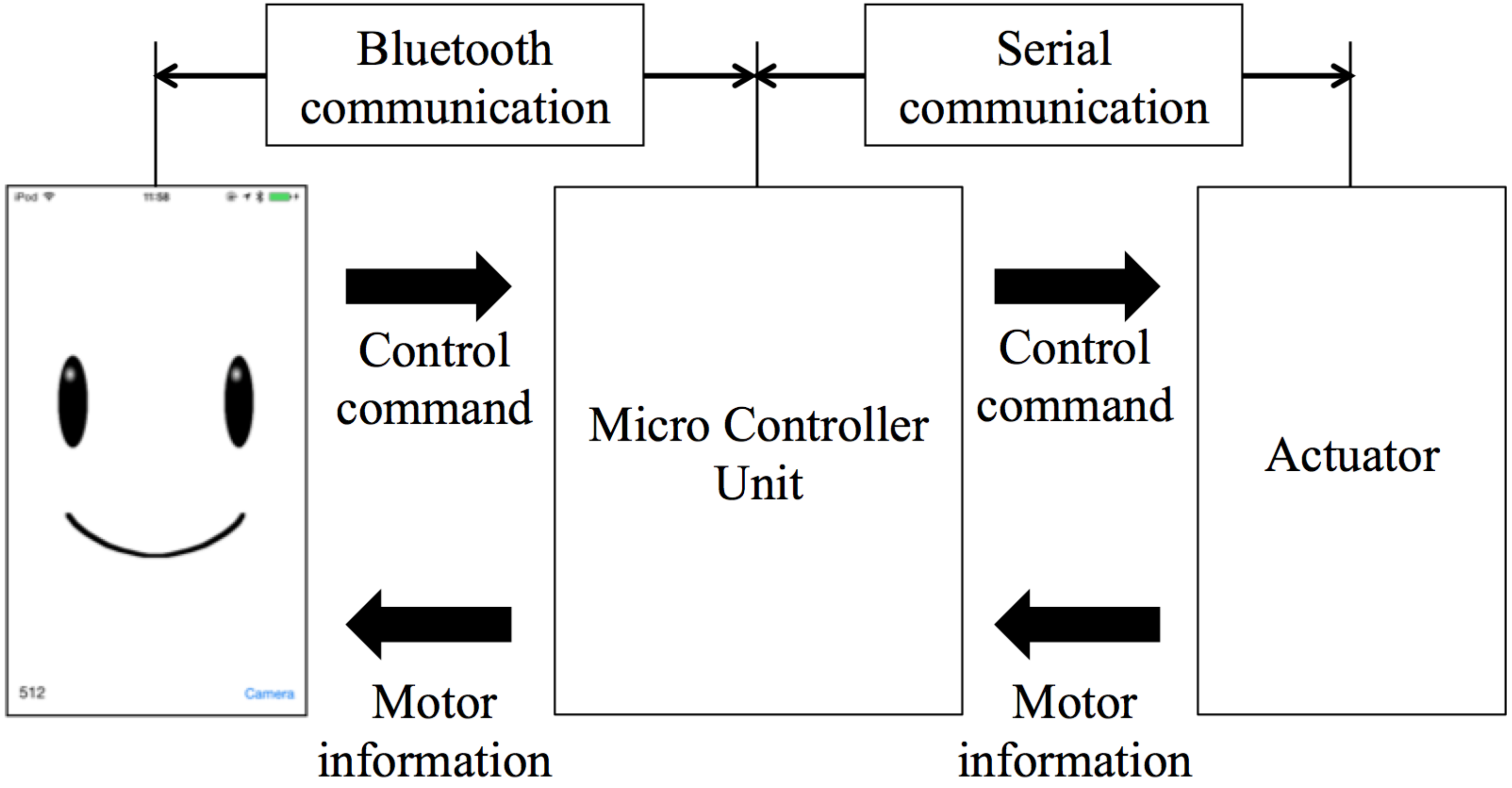
Robot Partner mode



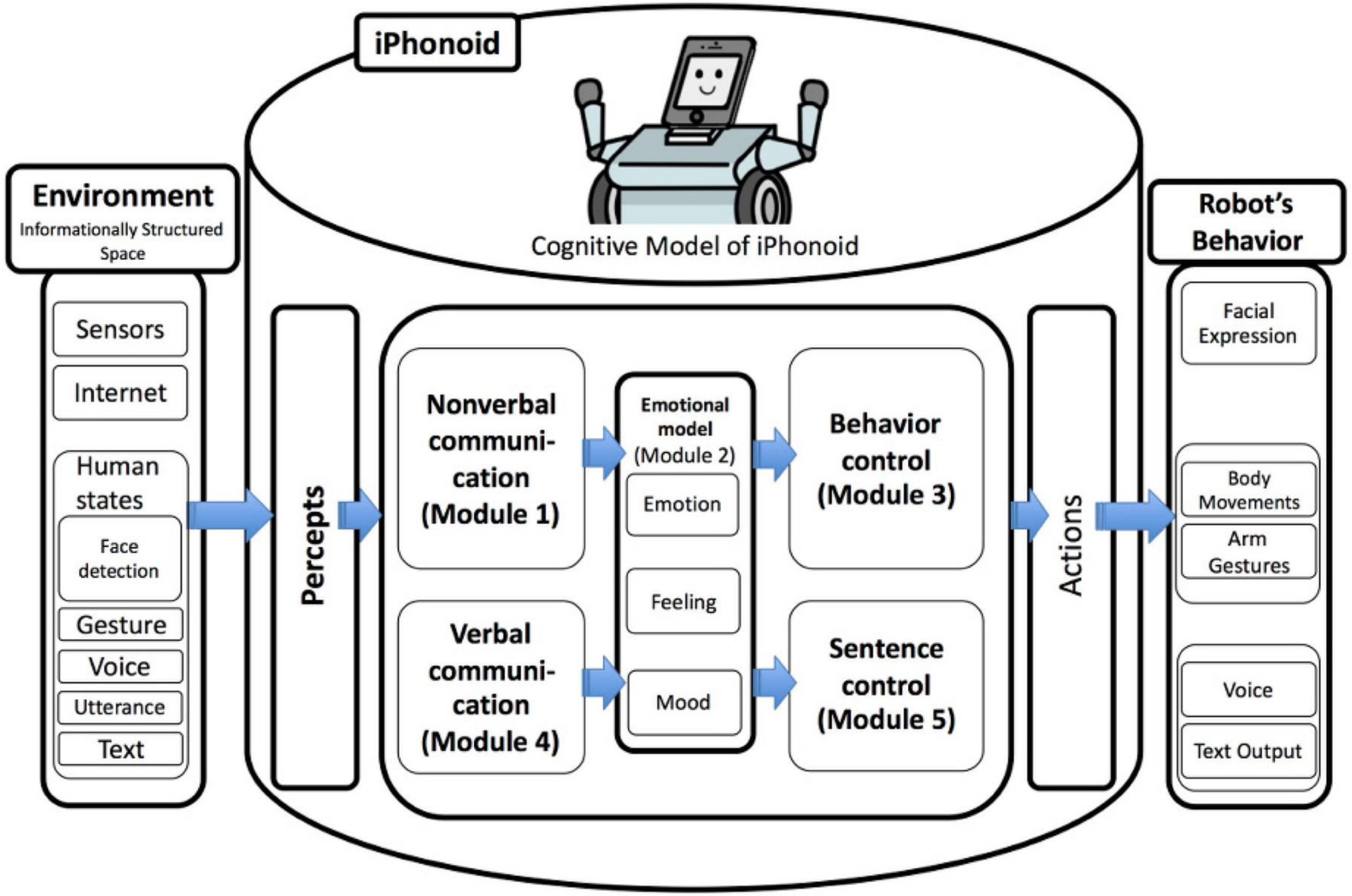
Charging Station

Pocket Robot mode

Specification of iPhonoid

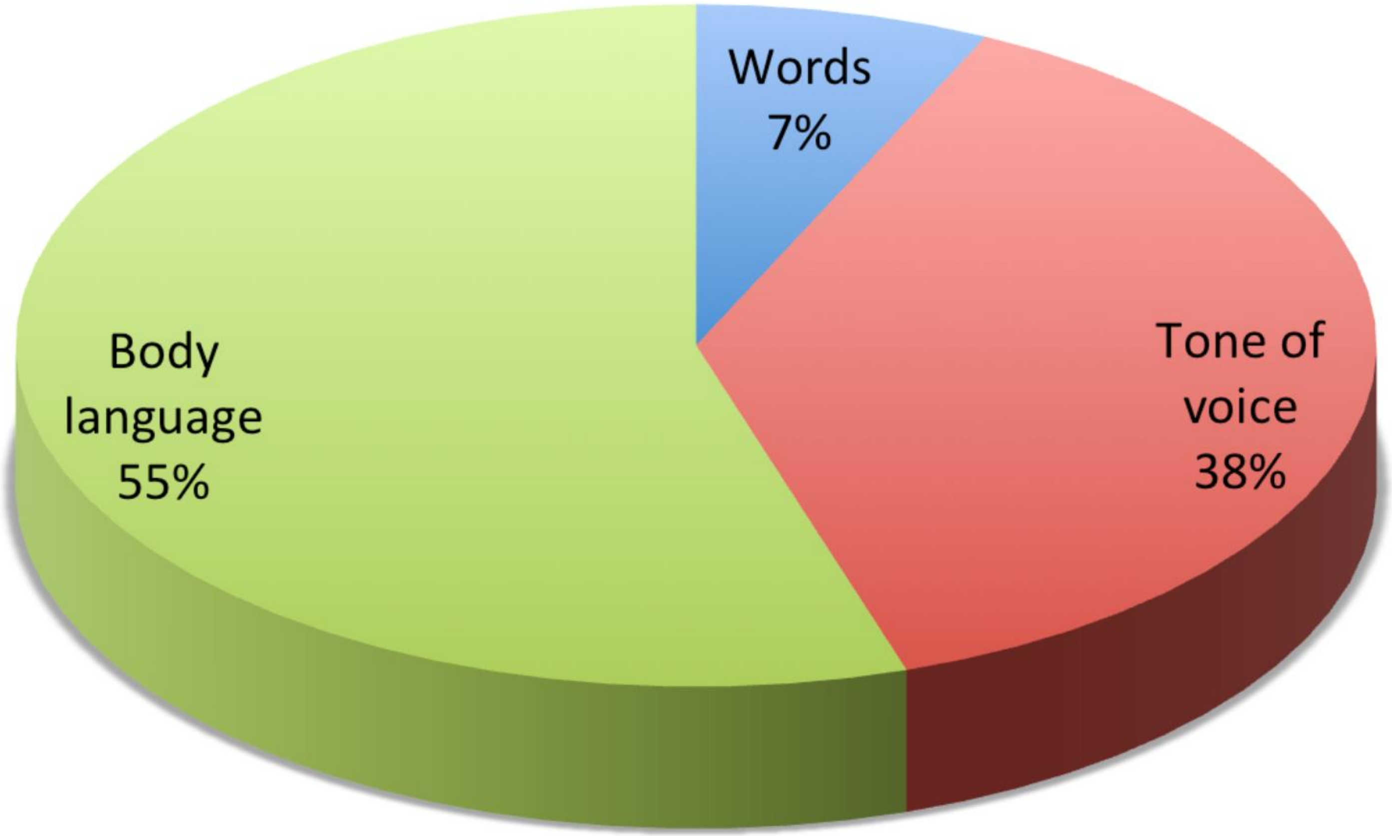


Cognitive Model of iPhonoid



Nonverbal communication module

The Law of Mehrabian



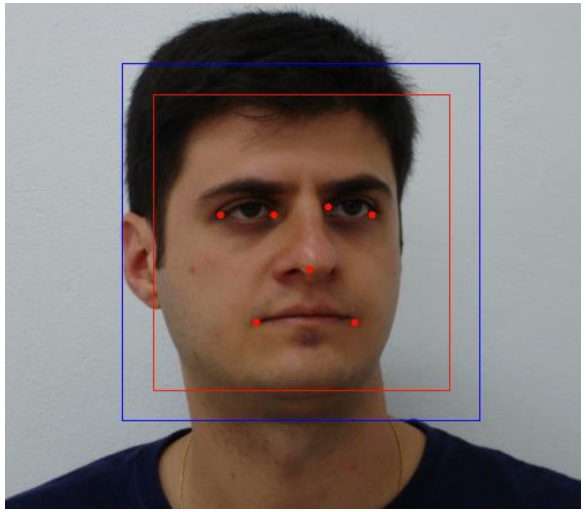
Nonverbal communication

- Robot partners can encourage elderly people to communicate with others
- Voice recognition is not enough in the daily conversation
- Nonverbal communications such as facial expressions, emotional gestures, and pointing gestures should also be understood
- Recognition of human gestures is important for smooth communication
- Human expressions used for natural communication are deeply related with emotional states

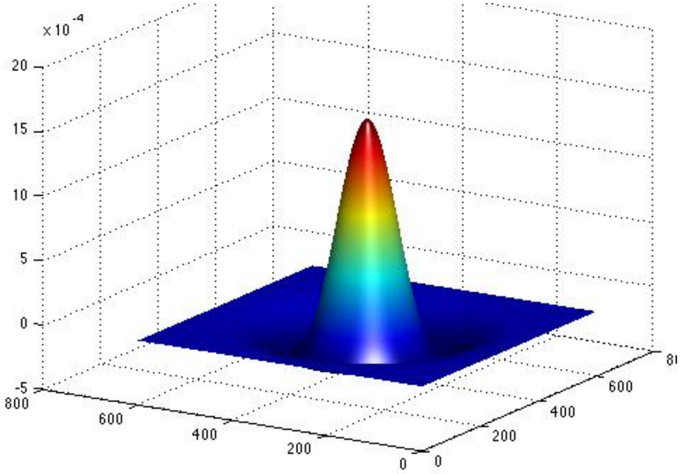
Face Classification

Face Detection via Viola-Jones Haar Cascade → Face Landmark detection by Deformable Face Model → Face Alignment using landmarks for the sides of eyes and nose → Standardize scale and translation → Difference of Gaussian filter applied to reduce effects of illumination → SVM for Gender Classification

Preprocessing

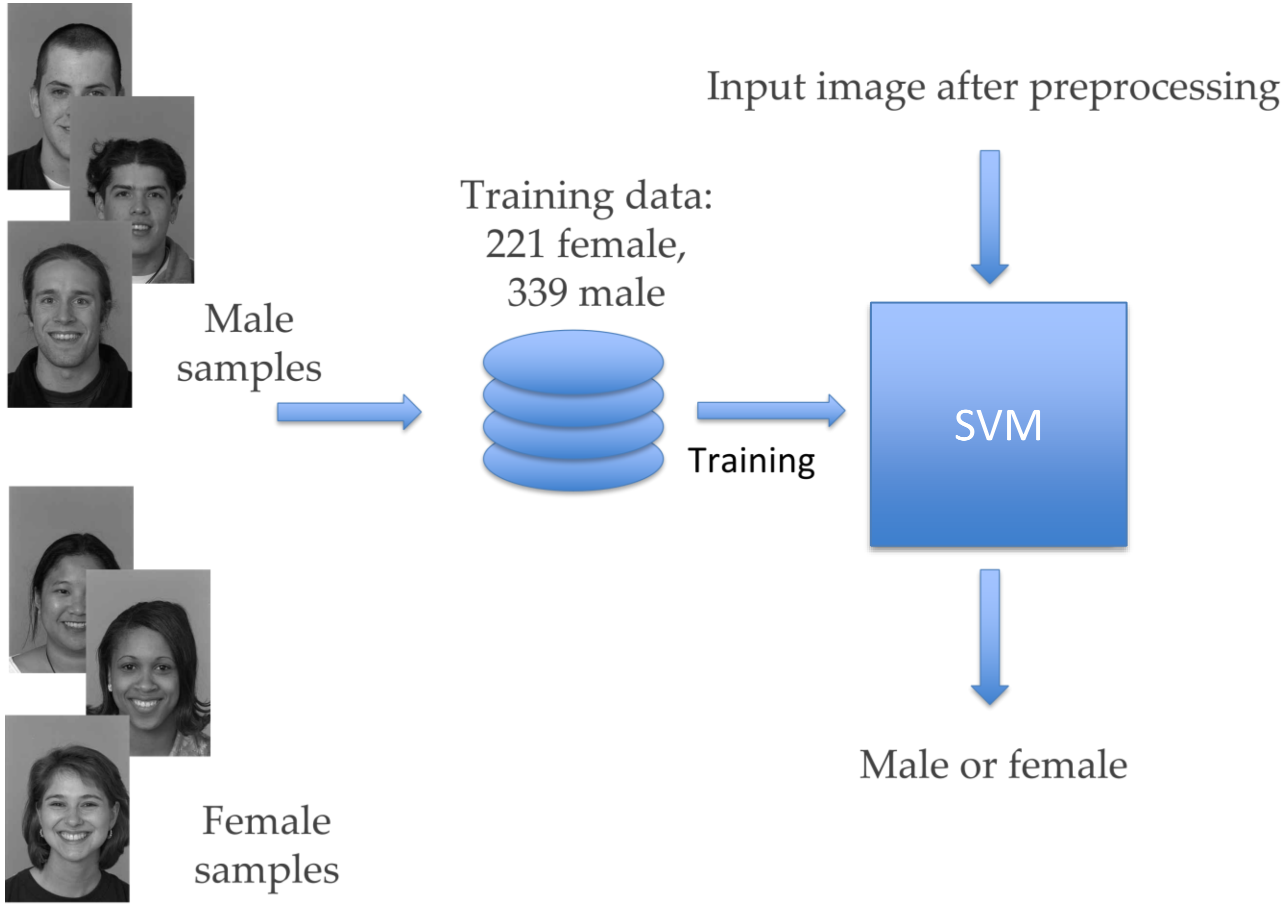


Face detection and landmark detection is performed. Landmarks are used to align the face for standardization. The aligned face will be cropped and scaled to a standard size



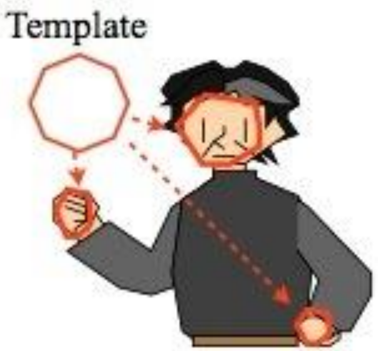
To reduce the effect of illumination, the image is convolved with the Difference of Gaussian filter as shown. The filter is a bandpass filter. Therefore it prevents high spatial frequency (from noise) and low spatial frequency (global illumination) from being passed on

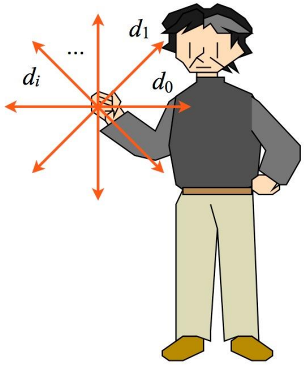
Classification



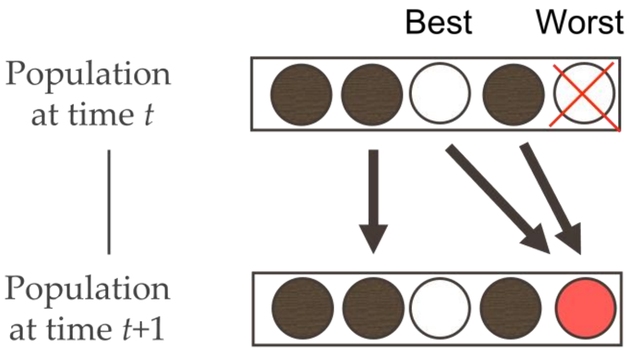
Human detection

1. Separate head and background based on consecutive frames
2. Using an evolutionary algorithm to identify the outline of the head
3. Detection of head and face motion using a spiking neural network
4. Gesture recognition with self-organizing map

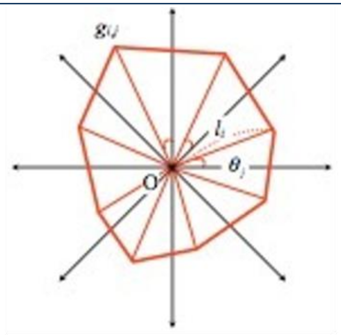




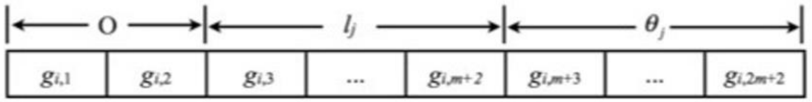
Identification of the outline of the head



Elite Crossover



Genotype: Template for human detection



Genotype in SSGA

Adaptive mutation: $g_{i,j} \leftarrow g_{i,j} + \left(\beta_{1,j} \cdot \frac{f_{max} - f_i}{f_{max} - f_{min}} + \beta_{2,j}\right) \cdot N(0,1)$

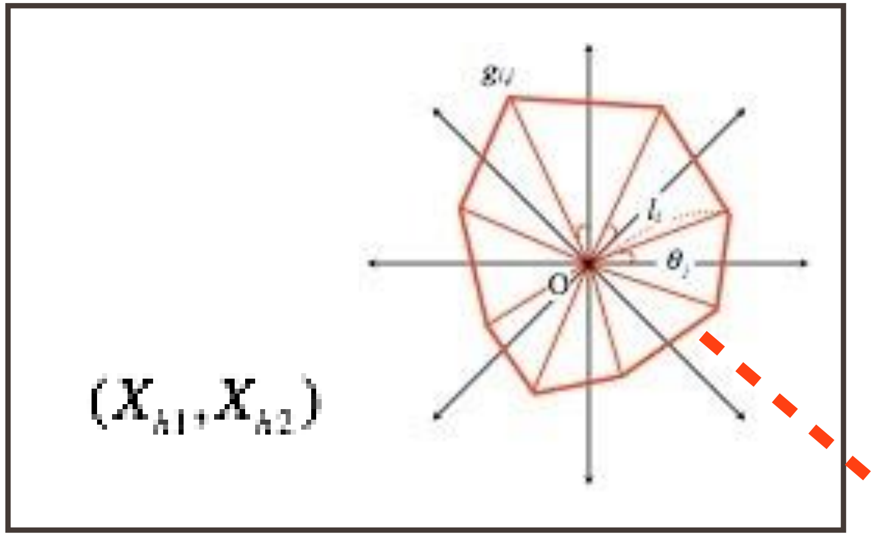
Fitness: $f_k(c) = \sum_{(x,y)\epsilon S_k} p_D(x,y) \cdot p_e(x,y,c)$

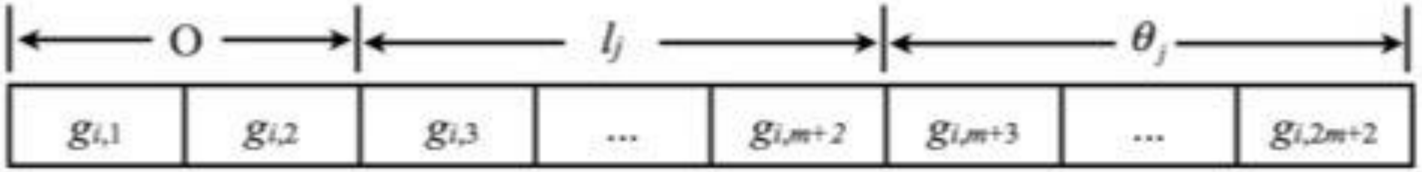
$$p_e(x,y,c) = \begin{cases} 1 & p_{clr}(x,y) = c \\ 0 & otherwise \end{cases}$$

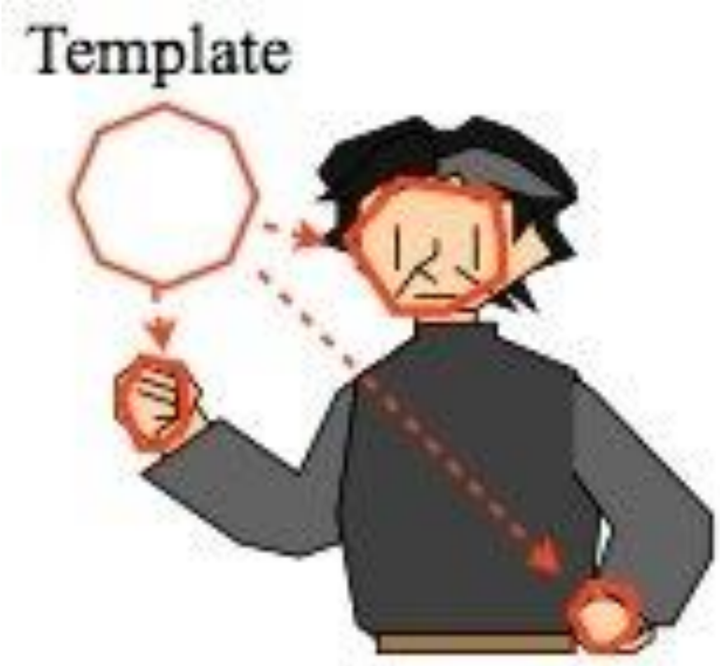
k: template; c: color; S_k: pixels associated with template k

$p_D(x,y)$: difference of successive frames

Steady State Genetic Algorithm







Tracking Position:

$$X_{h,k}(t) \leftarrow (1-\alpha_H)X_{h,k}(t-1) + \alpha_H \cdot g_{m,k}$$

Human tracking :

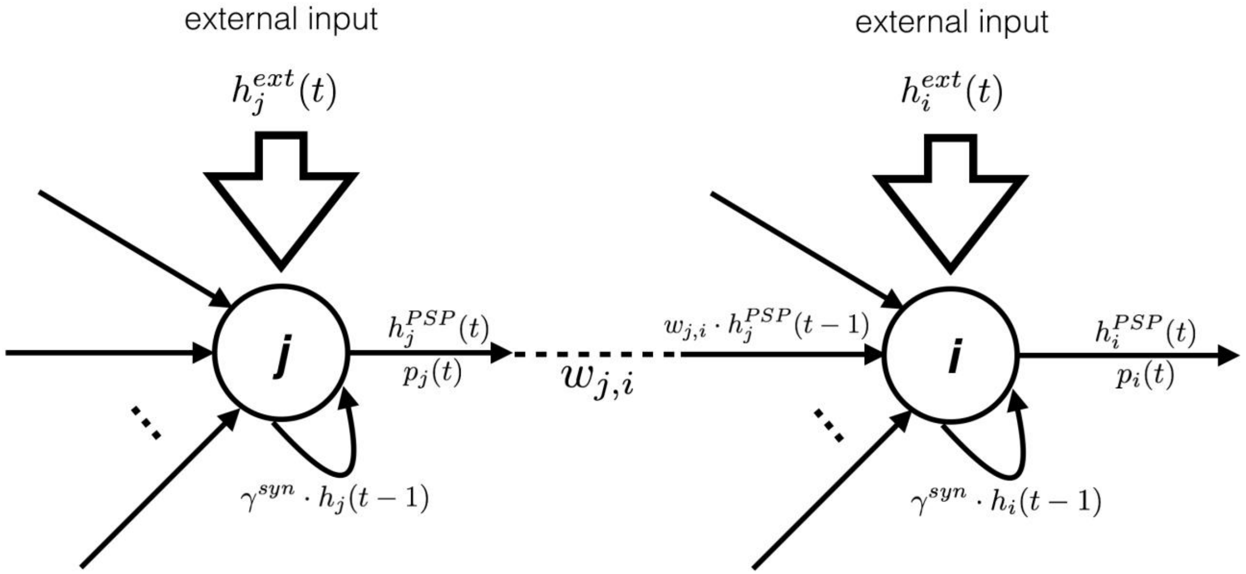
$$(X_{h1}, X_{h2})$$

Motion Extraction

- Spiking Neural Networks (SNN) are capable of temporal coding
  - they have been applied for memorizing spatial and temporal context
- Since the human gestures have spatiotemporal nature we can apply SNN for memorizing the gestures
- We use a simple spike response model to reduce the computational cost
- The membrane potential, or internal state $h_i(t)$ of the i-th spiking neuron at the discrete time t:

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

Simple Spike Response Model



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

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

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

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

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

Spiking Neural Networks

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

- where γ^{syn} is the temporal discount rate, $w_{j,i}$ is a weight from the *j*-th neuron to the *i*-th neuron, $p_j(t)$ is the spike output of the *j*-th neuron at the discrete time *t*, and *N* is the number of neurons

- The output of the *i*-th neuron:

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

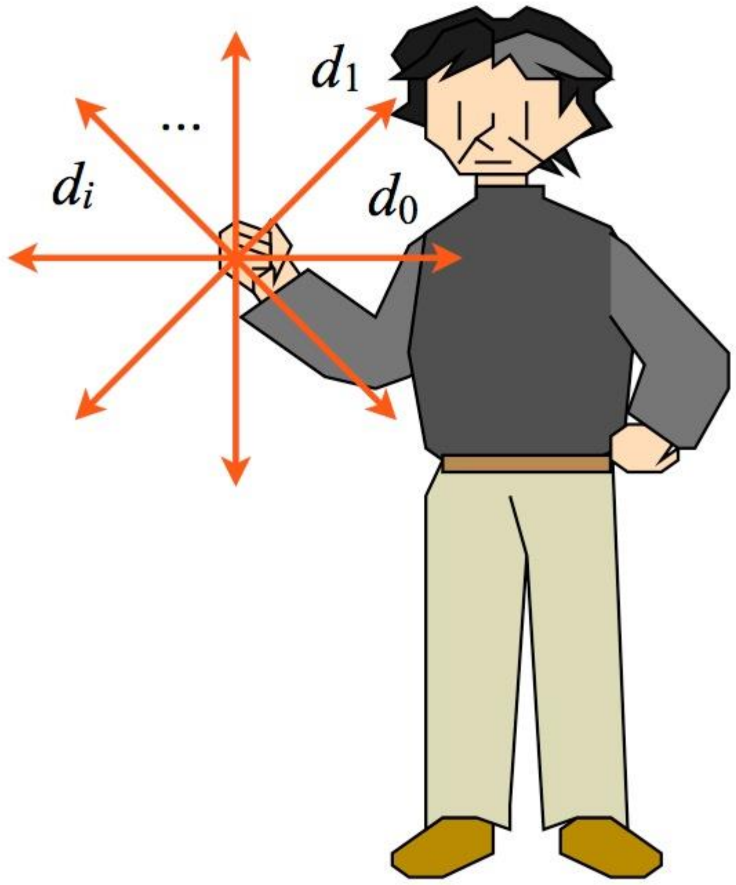
- where θ is a threshold for firing

- When the neuron fires:

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

- where γ^{ref} is a discount rate

Spiking Neural Networks



- The input to the *i*-th neuron is calculated from the structure of the SNN
- A directional structure with eight spiking neurons is applied

$$h_i^{ext}(t) = l(t) \cdot \exp\left(-\frac{\|\alpha_i - \alpha(t)\|^2}{\sigma^2}\right)$$

- where α_i is the directional information of the *i*-th neuron σ is the standard deviation, and $\alpha(t)$ and $l(t)$ are calculated from the trajectory of the human and human hand:

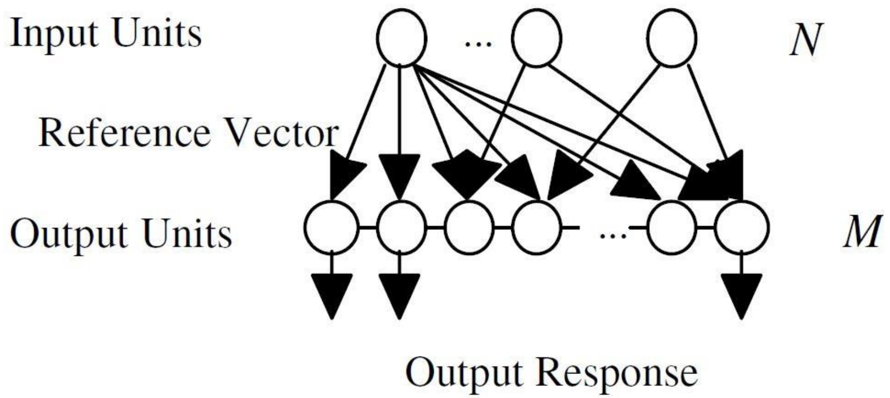
$$\alpha(t) = atan2\big(\Delta y(t), \Delta x(t)\big)$$

$$l(t) = \tanh\left(\frac{\Delta x(t)^2 + \Delta y(t)^2}{100}\right) \cdot 0{,}9$$

- where $\Delta x(t)$ and $\Delta y(t)$ are the changes of the *x* and *y* coordinates of the moving object at time *t*

Gesture Classification

- Self-organizing map (SOM) is often applied for extracting a relationship among observed data, since SOM can learn the hidden topological structure from the data
- Each input unit is connected to all output units in parallel via reference vectors
- Input data are distributed into output units
- The best matched output unit is selected according to the Euclidean distance



Self-organizing Map

- In the first step the reference vectors are initialized with small values
- In the training process in every iteration a training sample is shown to the SOM and the reference vectors are modified based on the training sample
- After the training process is finished the SOM can be used for mapping when every data including the training samples and other previously unseen samples are classified based on the SOM
- The input to the SOM is given as the weighted sum of pulse outputs from neurons:

$$v = (v_1, v_2, \dots, v_N)$$

$$v_i = \sum_{t=1}^{T} (\gamma^{SOM})^t \cdot p_i(t)$$

  - where $p_i(t)$ is the pulse output of the i-th neuron and γ^{SOM} is a weight parameter used for distinguishing the different directions in the time

Self-organizing Map

- In the training phase in every iteration a training sample (input) is used and the Euclidean distance between this input vector and the *i*-th reference vector is calculated:

$$d_i = \|v - r_i\|$$

  - where $\mathbf{r}_i=(r_{1,i},r_{2,i},\ldots,r_{N,i})$ and the number of reference vectors (output units) is M
- Next, the *k*-th output unit minimizing the distance d_i is selected:

$$k = \arg\min_i \|v - r_i\|$$

- Furthermore, the reference vectors are trained by:

$$r_i(t+1) = r_i(t) + \xi(t) \cdot \zeta_{k,i}(t) \cdot \big(v - r_i(t)\big)$$

  - where $\xi(t)$ is a learning rate ($0< \xi(t) <1$) , $\zeta_{k,i}(t)$ is a neighborhood function ($0< \zeta_{k,i}(t) <1$) describing the relationship between the winning *k*-th output unit and the other output units
  - The learning rate and the neighborhood function decrease with time
- After the training phase for any input data the output class can be determined by selecting the nearest output unit for the given input

Parameter Settings

- SNN: γ^{syn}=0.95, γ^{ref}=0.9, θ=0.8, R=1
- SOM:

number of iterations	2000
N	8
M	10
initial range	0.01
γ^{SOM}	0.98
learning rate (ξ_0)	0.3
τ	1000
ζ_1	0.9
ζ_2	0.7
ζ_3	0.5

- The learning rate for SOM is defined as: $\xi(t) = \xi_0 \cdot \exp(-t/\tau)$
- In the neighborhood function in the beginning stage the radius is 3 with parameter ζ_1, in the second stage the radius is 2 with parameter ζ_2, in the third stage the radius is 1 with parameter ζ_3, and in the final stage only the reference vector of the winning neuron is updated

Experimental result