Computer Vision

Genetic Algorithms

10 April 2018

j.van.de.loosdrecht@nhl.nl, jaap@vdlmv.nl

Genetic Algorithms (GA)

Overview:

- Introduction
- Example: Mother Nature
- Example: Finding best fit for a polynomial
- Example: Camera calibration
- Overview Operators
- Example Polynomial using fitness all function (*)
- Example: Travelling salesman
- Example: KnapSack
- Example: Optimising learn error Neural Network
- Example: Optimising Neural Network with evaluation set
- Example: Optimising Neural Network selecting features

Introduction

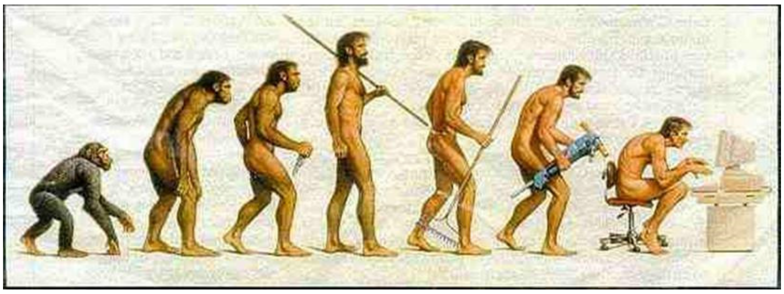
Idea:

In a population of beings the “strongest” are able to mate and to produce offspring. Disabled or sick beings have less opportunities to mate and have a smaller change to produce healthy offspring.

When environmental conditions are changing only offspring which can adapt to the new situation have good changes to survive and produce adapted offspring.

In short: “survival of the fittest”

Introduction Genetic Algorithms



Example Mother Nature

DNA represent the blueprint of a being

DNA mother: 11011010111010101001
DNA father: 10011010110001001011
Mating, DNA child:
- **Crossover: 11011010110001001011**
- **Mutation: 10011110110101001011**

Mutation makes it possible to get children that are better then their parents and/or can adapt to a changing environment

Fitness function determines whether a being can reproduce

27-aug-18 Genetic Algorithms 5

Genetic Algorithms (GA) for optimisation problems

GA can be used for optimisation problems if for the solution domain can be defined:
- **a genetic representation**
- **a fitness function**

27-aug-18 Genetic Algorithms 6

Genetic Algorithm (GA) for optimisation problems

Pseudo code:
- **Choose initial (random) population**
- **Evaluate the fitness of each individual in the population**
- **While not good enough best individual do**
  - **Select best ranking individuals to reproduce**
  - **Breed new generation through crossover and mutation**
  - **Add new generation to population**
  - **Evaluate the fitness of each individual in the population**

Example: finding best fit for a polynomial

- **Generate a point cloud of measurements (x,y)**
- **Use GA to find parameters a, b, and c of polynomial**

$$y = ax^2 + bx + c$$

- **Solution domain:**
  - **Genetic representation:**
    the floating point numbers a, b and c
  - **Fitness function:**
    the sum of errors of applying the solution to the point cloud

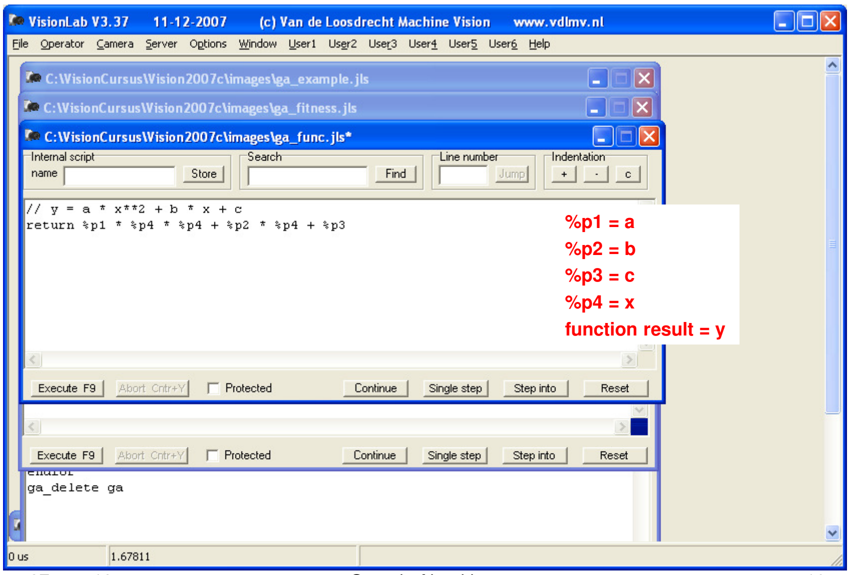
Demonstration: finding best fit for a polynomial

- Open scripts:
  - ga_func.jls:
    calculation of the polynomial
  - ga_example.jls: the main program
  - ga_fitness.jls: calculation the fitness of a solution
- Open variable window
- Run script ga_example.jls

- Change line to create ga in:
  ga_create ga ga_fitness 10 &$paramTab ***10*** tracefile.txt
- Run script for about 20 generations
- Examine file tracefile.txt to see the best 10 solutions for each generation
  Note: if tracefile indicates that n best solution are (for a large part) the same the best solution is found or the Genetic Algorithm is stuck in a local minimum.
  If the algorithm is stuck one or more of the parameters have to be changed

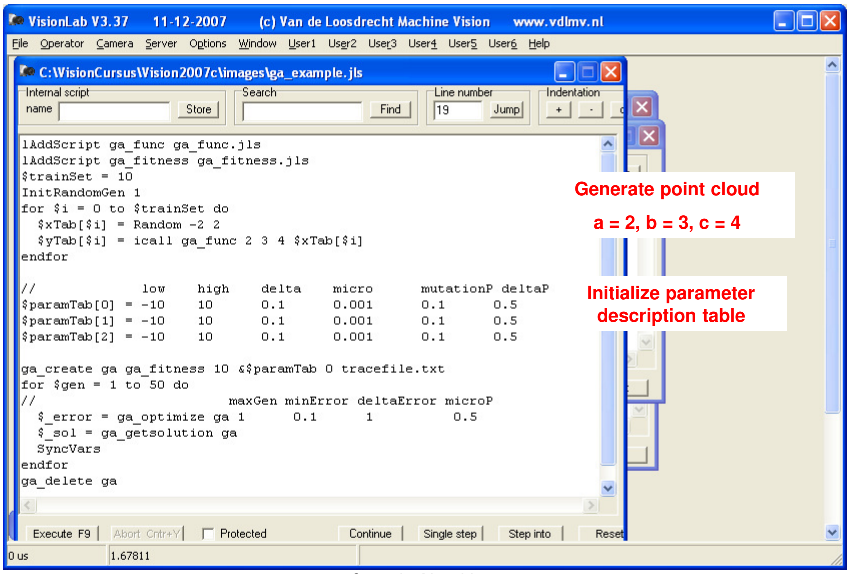
27-aug-18 Genetic Algorithms 9

ga_func.jls



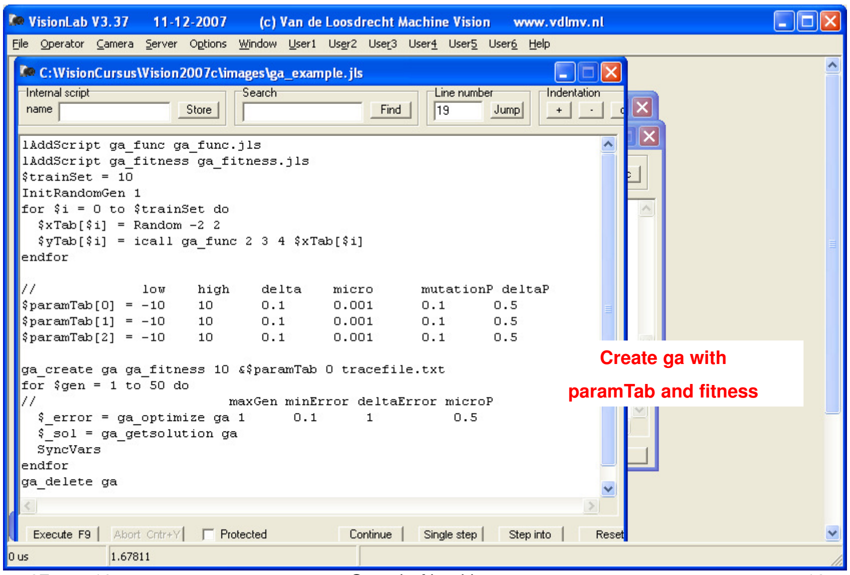
27-aug-18 Genetic Algorithms 10

ga_example.jls



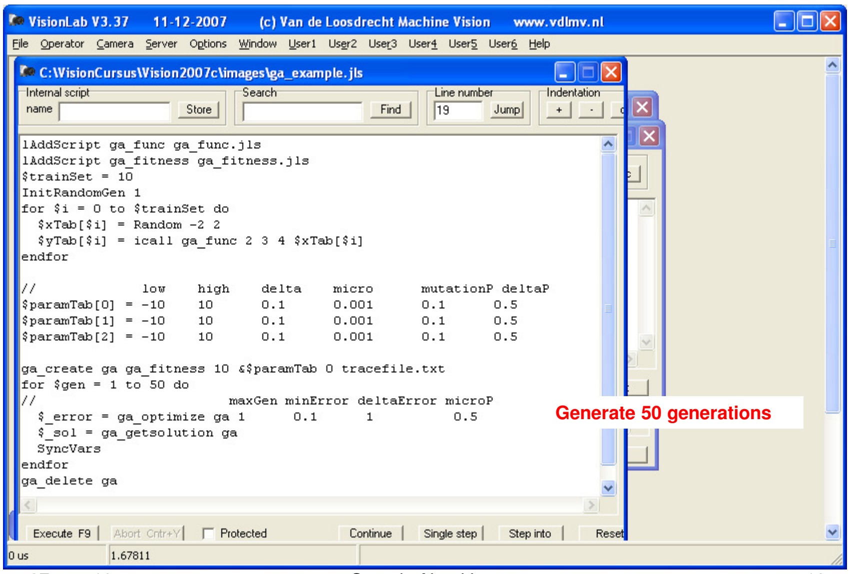
27-aug-18 Genetic Algorithms 11

ga_example.jls



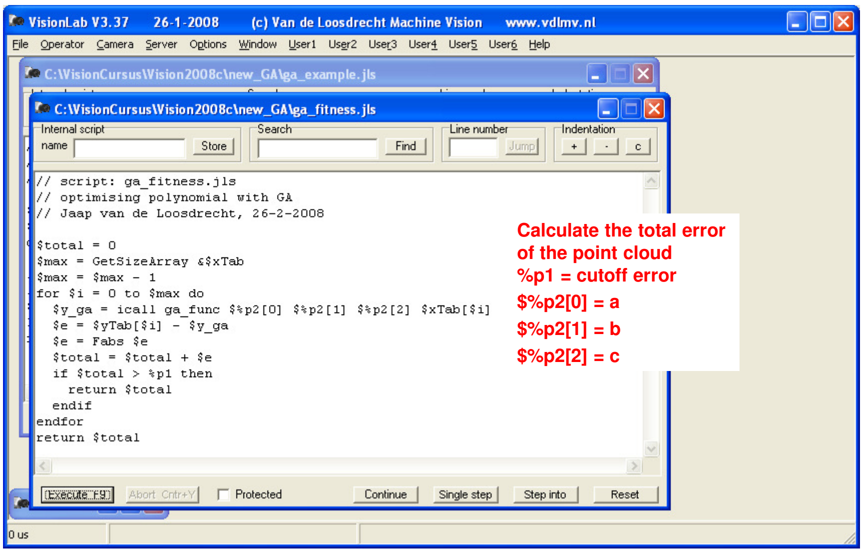
27-aug-18 Genetic Algorithms 12

ga_example.jls



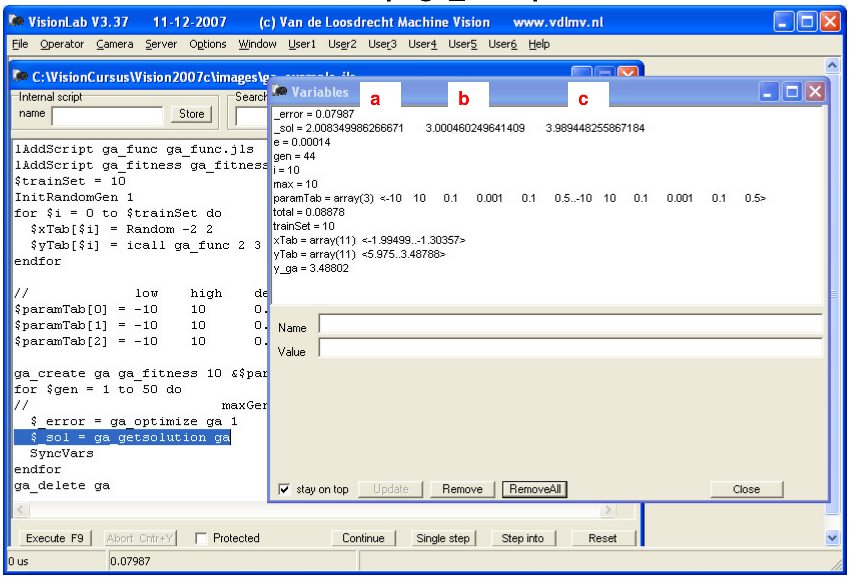
27-aug-18 Genetic Algorithms 13

ga_fitness.jls



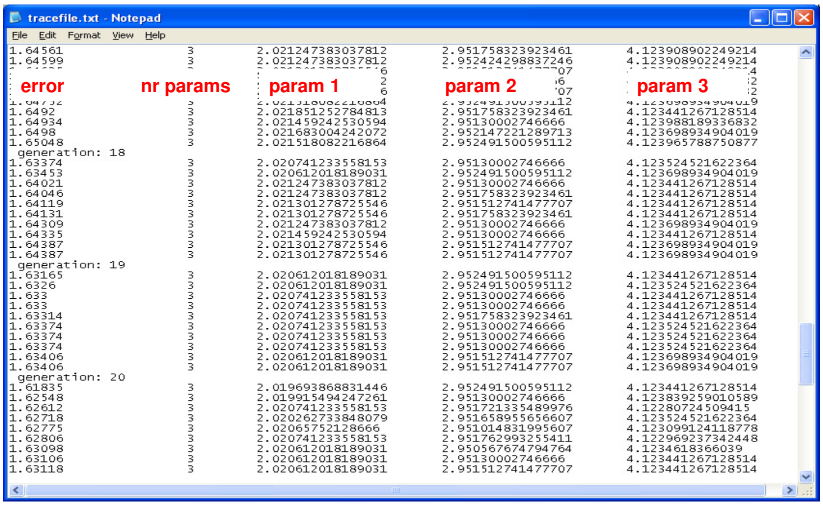
27-aug-18 Genetic Algorithms 14

Run script ga_example



27-aug-18 Genetic Algorithms 15

tracefile.txt



error	nr params	param 1	param 2	param 3
1.64561	3	2.021247383037812	2.951758323923461	4.123908902249214
1.64599	3	2.021247383037812	2.952424298837246	4.123908902249214
1.6492	3	2.021851252784813	2.951758323923461	4.123441267128514
1.64934	3	2.021459242530594	2.95130002746666	4.123988189336832
1.6498	3	2.021683004242072	2.952147221289713	4.123698934904019
1.65048	3	2.021518082216864	2.952491500595112	4.123965788750877
generation: 18				
1.63374	3	2.020741233558153	2.95130002746666	4.123524521622364
1.63453	3	2.020612018189031	2.952491500595112	4.123698934904019
1.64021	3	2.021247383037812	2.95130002746666	4.123441267128514
1.64046	3	2.021247383037812	2.951758323923461	4.123441267128514
1.64119	3	2.021301278725546	2.951512741477707	4.123441267128514
1.64131	3	2.021301278725546	2.951758323923461	4.123441267128514
1.64309	3	2.021247383037812	2.95130002746666	4.123698934904019
1.64335	3	2.021459242530594	2.95130002746666	4.123441267128514
1.64387	3	2.021301278725546	2.951512741477707	4.123698934904019
1.64387	3	2.021301278725546	2.951512741477707	4.123698934904019
generation: 19				
1.63165	3	2.020612018189031	2.952491500595112	4.123441267128514
1.6326	3	2.020612018189031	2.952491500595112	4.123524521622364
1.633	3	2.020741233558153	2.95130002746666	4.123441267128514
1.633	3	2.020741233558153	2.95130002746666	4.123441267128514
1.63314	3	2.020741233558153	2.951758323923461	4.123441267128514
1.63374	3	2.020741233558153	2.95130002746666	4.123524521622364
1.63374	3	2.020741233558153	2.95130002746666	4.123524521622364
1.63374	3	2.020741233558153	2.95130002746666	4.123524521622364
1.63406	3	2.020612018189031	2.951512741477707	4.123698934904019
1.63406	3	2.020612018189031	2.951512741477707	4.123698934904019
generation: 20				
1.61835	3	2.019693868831446	2.952491500595112	4.123441267128514
1.62548	3	2.019915494247261	2.95130002746666	4.123839259010589
1.62612	3	2.020741233558153	2.951721335489976	4.12280724509415
1.62718	3	2.020262733848079	2.951658955656607	4.123524521622364
1.62775	3	2.02065752128666	2.951014831995607	4.123099124118778
1.62806	3	2.020741233558153	2.951762993255411	4.122969237342448
1.63098	3	2.020612018189031	2.950567674794764	4.1234618366039
1.63106	3	2.020612018189031	2.95130002746666	4.123441267128514
1.63118	3	2.020612018189031	2.951512741477707	4.123441267128514

27-aug-18 Genetic Algorithms 16

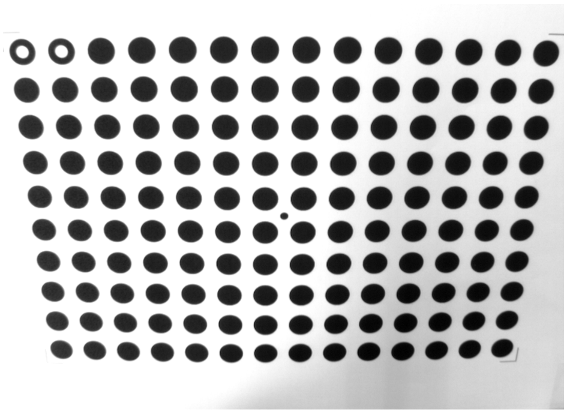
Example: Camera calibration

This example is from the chapter about camera calibration
See this chapter for more details

27-aug-18 Genetic Algorithms 17

Calibration Pattern

A fixed pattern of circles is used to calibrate the camera



27-aug-18 Genetic Algorithms 18

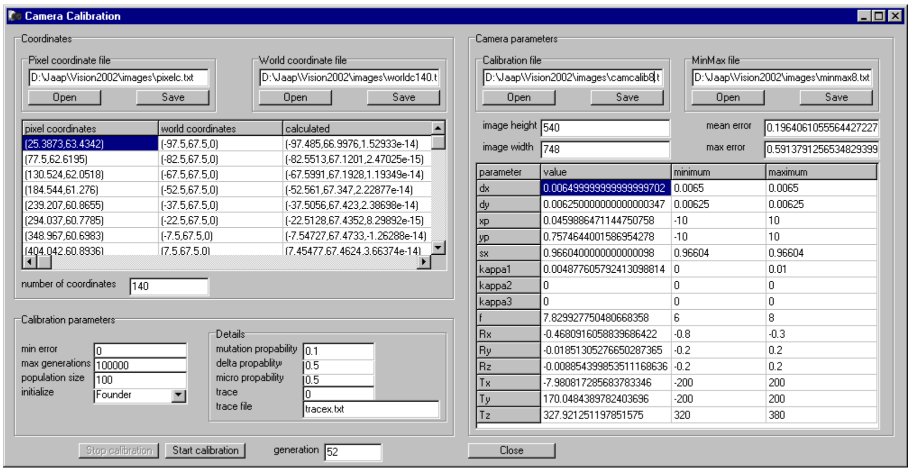
Camera Calibration

The 140 calibration points are used to find optimal values for the 15 camera parameters.

So there are 140 equations with 15 unknown parameters.
The parameter space has 15 dimensions and consists of polynomial and goniometric equations.

This non-linear optimalisation problem is solved in VisionLab with a combination of a Genetic Algorithm and Hill Climbing.

Demonstration Camera Calibration



GA in VisionLab

VisionLab uses a combination of genetic algorithm and "inverted hill climbing" to descend to the bottom of the local minima
All parameters are represented as floating point values

Overview of operators:
- GA_Create
- GA_Delete
- GA_RandomInitialize
- GA_SetFounder
- GA_SetFounders
- GA_Optimize
- GA_GetSolution
- GA_GetPopulation

GA_Create

GA_Create (gaName, fitnessName, popSize,
paramTab, trace, traceFileName)

Create an instance of a Genetic Algorithm
Parameters:
- gaName: name of variable
- fitnessName: name of internal script
  %p1 is cutoff error
  %p2 is the name of an array variable without the '$' containing the parameters to optimise, $%p2[0] is the first parameter, $%p2[1] is the second parameter, etc
- popSize: the population size
- trace: = 0 then no tracing, > 0 = number of best in population to trace
- traceFileName: filename to which trace log is written

GA_Create

Parameters continued:

- **paramTab: array with the description of the parameters:**
  - **low: the lowest possible value for the parameter**
  - **high: the highest possible value for the parameter**
  - **delta: step size used in finding the bottom in local minimum**
  - **micro: step size used by selfmating**
  - **mutationP: probability for mutation when mating, range [0..1]**
  - **deltaP: determines how much time will be used be used for "inverted hill climbing", range [0..1]**

GA_CreateAll

GA_CreateAll (gaName, fitnessName, popSize, paramTab, trace, traceFileName)

Create an instance of a Genetic Algorithm for with the fittness function will be called one time for each generation.

GA_CreateAll

Parameters:

- **gaName: name of variable**
- **fitnessName: name of internal script**
  %p1 is cutoff error
  %p2 is the name of an array variable without the '$'containing the parameters for each being to optimise.
  $%p2[0] contains the parameters for the first being, $%p2[1] for the second being etc.
  The fittnes script will have to insert the error for each being in front of theparameters for the being.
- **popSize: the population size**
- **trace: = 0 then no tracing, > 0 = number of best in population to trace**
- **traceFileName: filename to which trace log is written**

27-aug-18 Genetic Algorithms 25

GA_CreateAll

Parameters continued:

- **paramTab: array with the description of the parameters:**
  - **low: the lowest possible value for the parameter**
  - **high: the highest possible value for the parameter**
  - **delta: step size used in finding the bottom in local minimum**
  - **micro: step size used by selfmating**
  - **mutationP: probability for mutation when mating, range [0..1]**
  - **deltaP: determines how much time will be used be used for "inverted hill climbing", range [0..1]**

27-aug-18 Genetic Algorithms 26

GA_Delete

GA_Delete (gaName)

Delete an instance of a Genetic Algorithm

Parameter:
- **gaName: name of variable**

GA_RandomInitialize

GA_RandomInitialize (gaName, populationSize)

Random intialise the population

Parameters:
- **gaName: name of variable**
- **populationSize: the population size**

GA_SetFounder

GA_SetFounder (gaName, populationSize, founder)

Initialise the population with offspring from a parent

Parameters:
- **gaName: name of variable**
- **populationSize: the population size**
- **founder: the parameters of the founder**

GA_SetFounders

GA_SetFounders (gaName, parents)

Parameters:
- **gaName: name of variable**
- **parents: an array with the parents, one line for each parent**

GA_Optimize

GA_Optimize (gaName, maxGenerations, minError, deltaError, microP)

Use the genetic algorithm to find a best solution.
The function result is the fitness error of the best solution.

Parameters:
- **gaName: name of variable**
- **maxGenerations: maximum number of generations**
- **minError: if fitness error < minError the optimization is stopped**
- **deltaError: minimum error for start of inverted hillclimbing**
- **microP: probability that self mate is a micro mutation**

27-aug-18 Genetic Algorithms 31

GA_GetSolution

GA_GetSolution (gaName)

Function result is the best solution found

Parameter:
- **gaName: name of variable**

27-aug-18 Genetic Algorithms 32

GA_GetPopulation

GA_GetPopulation (gaName, population)

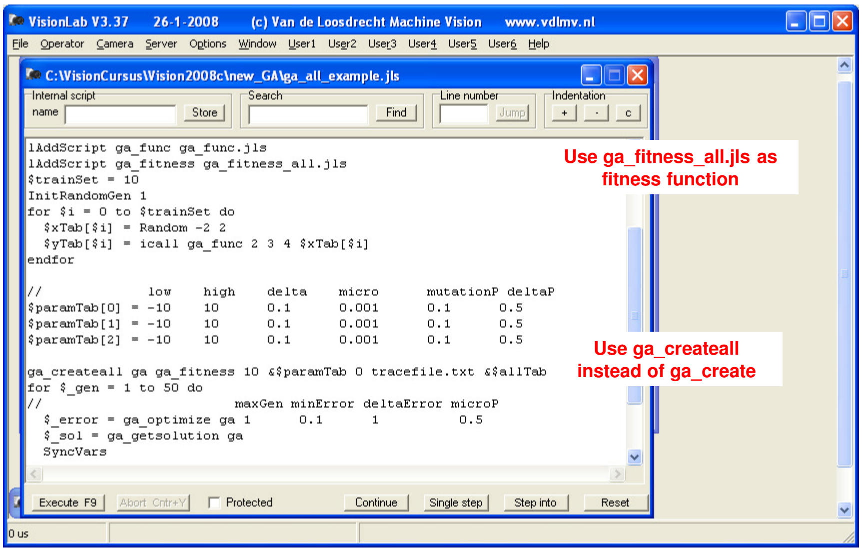
Retreive the current population

Parameter:
- **gaName: name of variable**
- **population: array to store the population, one line for each parent**

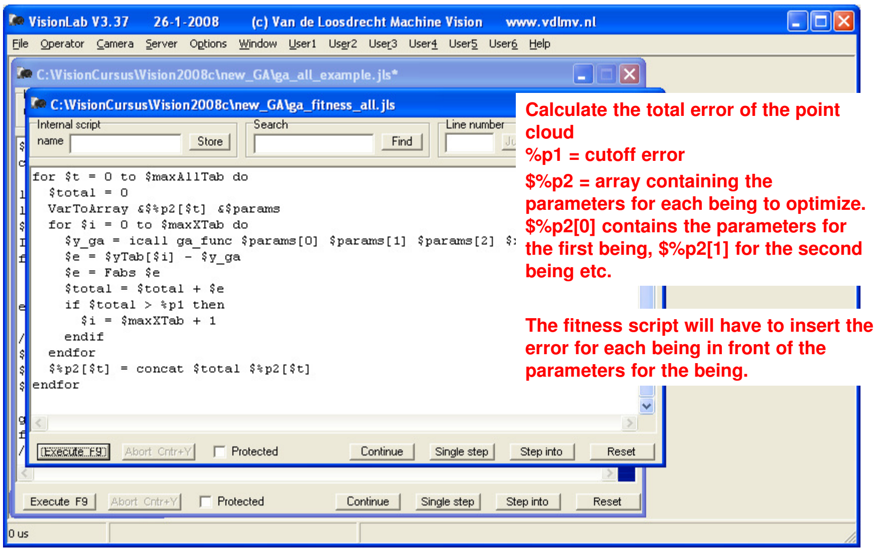
Demonstration: finding best fit for a polynomial using fitness all function

- **Open scripts:**
  - **ga_func.jls: calculation of the polynomial**
  - **ga_all_example.jls: the main program**
  - **ga_fitness_all.jls: calculation the fitness of a solution**
- **Open variable window**
- **Run script ga_all_example.jls**
- **Note: in this example the fitness function will evaluate all beings instead of one being in the previous example.**
- **The only differences between ga_example.jls and ga_all_example.jls are:**
  - **lAddScript ga_fitness ga_fitness_*all*.jls**
  - **ga_create*all* ga ga_fitness 10 &$paramTab 0 tracefile.txt &$allTab**

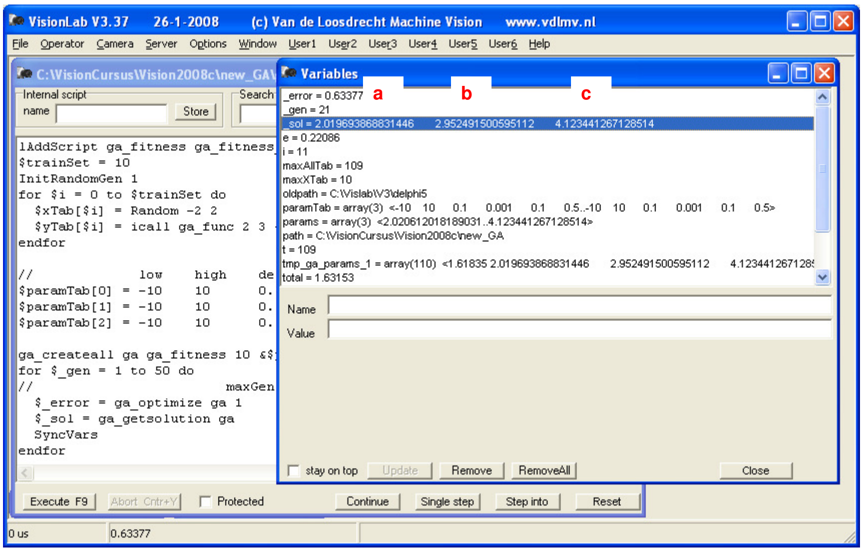
ga_all_example.jls



ga_fitness_all.jls



Run script ga_all_example



Example: Travelling Salesman Problem (TSP)

Optimizing problem:

- **Given a number of cities and the distance between the cityes, what is the least-cost round-trip route that visits each city and returns to the starting city?**

- **Solution domain:**
  - **Genetic representation:**
    a list indication the order in which the cities are visited
  - **Fitness function:**
    total distance of the round-trip

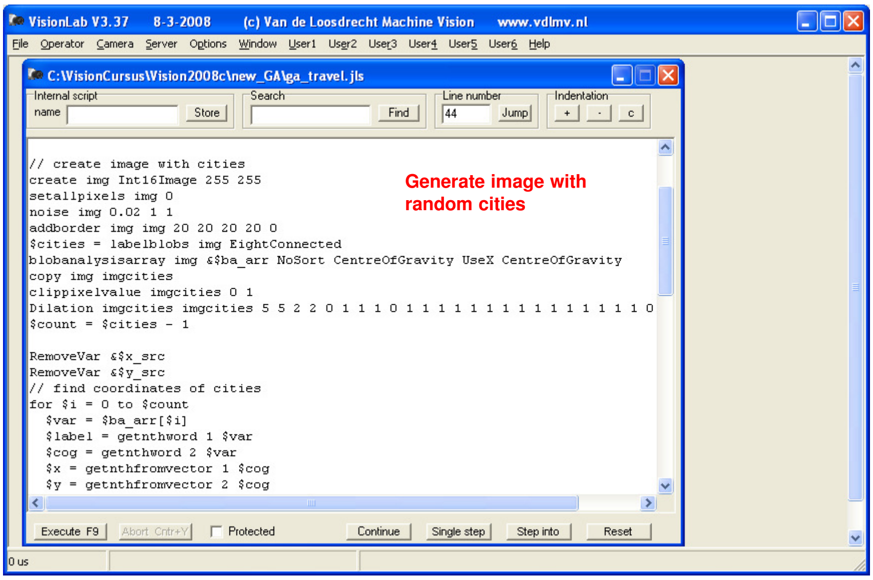
Example: TSP

- Cities are random pixel chosen in an empty image
- Genetic representation:
  a list indication the order in which the cities are visited
  - VisionLab implementation works with floating point numbers
  - an array with floating point values indicating the order in which the cities are visited list, 0 <= value <= 1, lowest values are visited first
- Fitness function:
  - Euclidian distance between cities

Demonstration: TSP

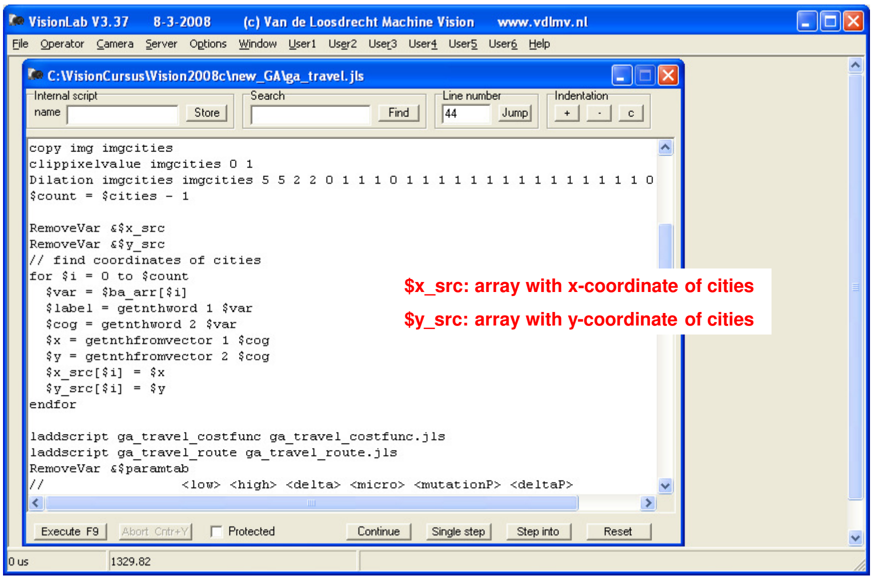
- Open scripts:
  - ga_travel.jls: the main program
  - ga_travel_costfunc.jls: calculation of the fitness of a solution
  - ga_travel_sort.jls: generate a route
- Open variable window
- Run script ga_travel.jls

ga_travel.jls (1)



27-aug-18 Genetic Algorithms 41

ga_travel.jls (2)



27-aug-18 Genetic Algorithms 42

ga_travel.jls (3)

VisionLab V3.37 8-3-2008 (c) Van de Loosdrecht Machine Vision www.vdlmv.nl

C:\VisionCursus\Vision2008c\new_GA\ga_travel.jls

```
laddscript ga_travel_costfunc ga_travel_costfunc.jls
laddscript ga_travel_route ga_travel_route.jls
RemoveVar &$paramtab
//                 <low> <high> <delta> <micro> <mutationP> <deltaP>
for $i = 0 to $count
  $paramtab[$i] =    0     1       0     0.1       0.2       0
endfor

GA_Create ga ga_travel_costfunc 6 &$paramtab 1 tracefile.txt
while true
  $$dist = GA_Optimize ga 1 0 0 0.5
  $sol = GA_GetSolution ga
  vartoarray &$sol &$sol_arr
  icall ga_travel_route $sol_arr $x_src $y_src $$x $$y
  // display result
  copy img img_disp
  setallpixels img_disp 0
  $count = $cities - 1
  for $i = 1 to $count
    $i1 = $i - 1
    $c_src = vector $$x[$i1] $$y[$i1]
    $c_dst = vector $$x[$i] $$y[$i]
```

Initialize parameter description table

Generate n generations

27-aug-18 Genetic Algorithms 43

ga_travel.jls (4)

VisionLab V3.37 8-3-2008 (c) Van de Loosdrecht Machine Vision www.vdlmv.nl

C:\VisionCursus\Vision2008c\new_GA\ga_travel.jls

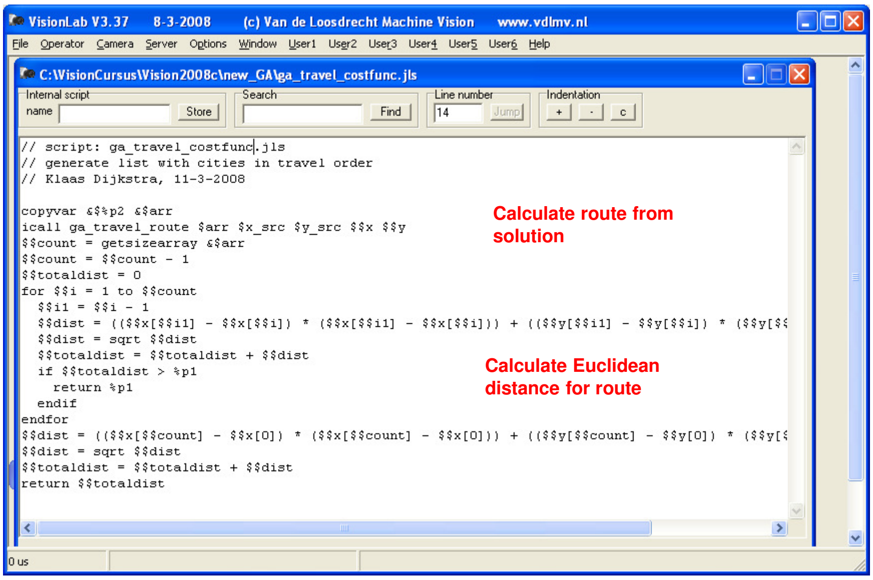
```
  $$dist = GA_Optimize ga 1 0 0 0.5
  $sol = GA_GetSolution ga
  vartoarray &$sol &$sol_arr
  icall ga_travel_route $sol_arr $x_src $y_src $$x $$y
  // display result
  copy img img_disp
  setallpixels img_disp 0
  $count = $cities - 1
  for $i = 1 to $count
    $i1 = $i - 1
    $c_src = vector $$x[$i1] $$y[$i1]
    $c_dst = vector $$x[$i] $$y[$i]
    drawline img_disp $c_src $c_dst 3 KeepOriginal
  endfor
  $c_src = vector $$x[$count] $$y[$count]
  $c_dst = vector $$x[0] $$y[0]
  drawline img_disp $c_src $c_dst 3 KeepOriginal
  or img_disp imgcities
  textonimage img_disp 10 10 courier10 $$dist
  display img_disp
endwhile
```

Generate route from GA solution

Display route in image

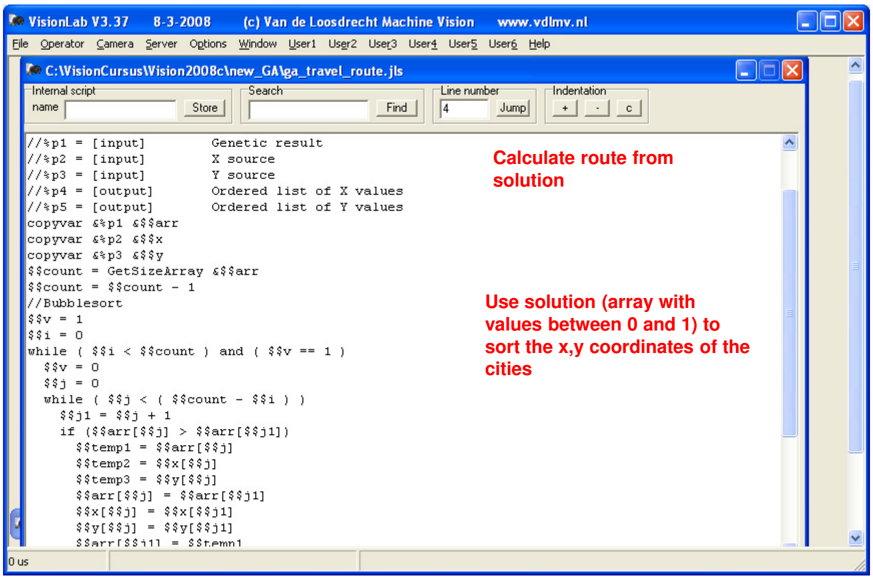
27-aug-18 Genetic Algorithms 44

ga_travel_costfunc.jls



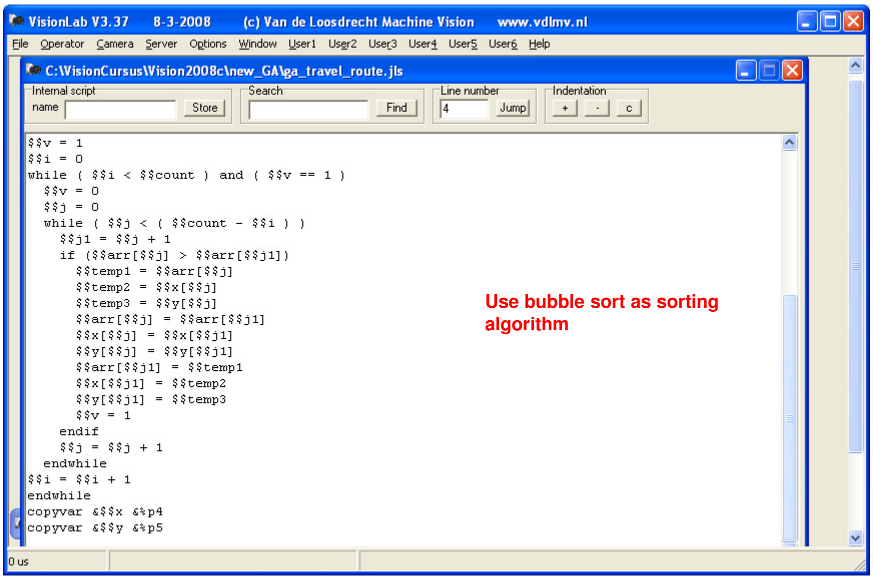
27-aug-18 Genetic Algorithms 45

ga_travel_route.jls (1)



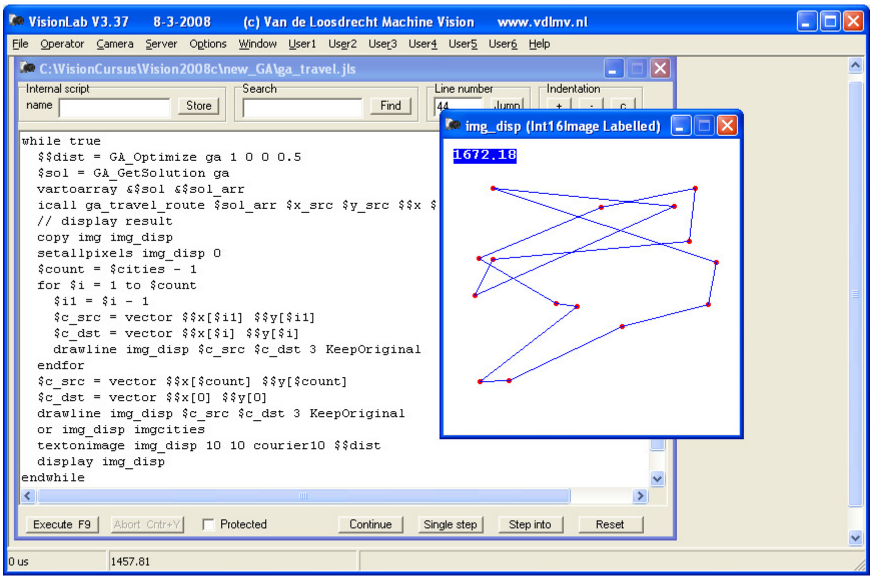
27-aug-18 Genetic Algorithms 46

ga_travel_route.jls (2)



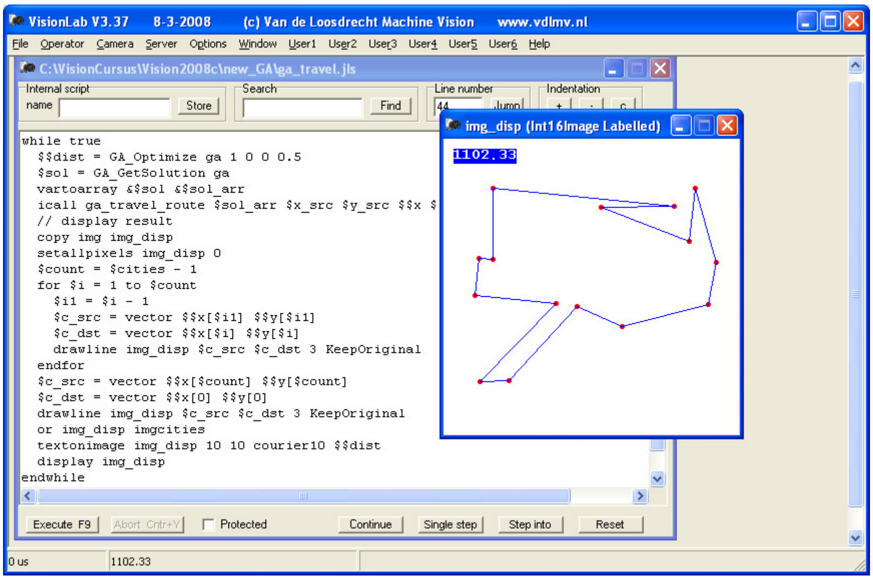
27-aug-18 Genetic Algorithms 47

Running TSP (1)



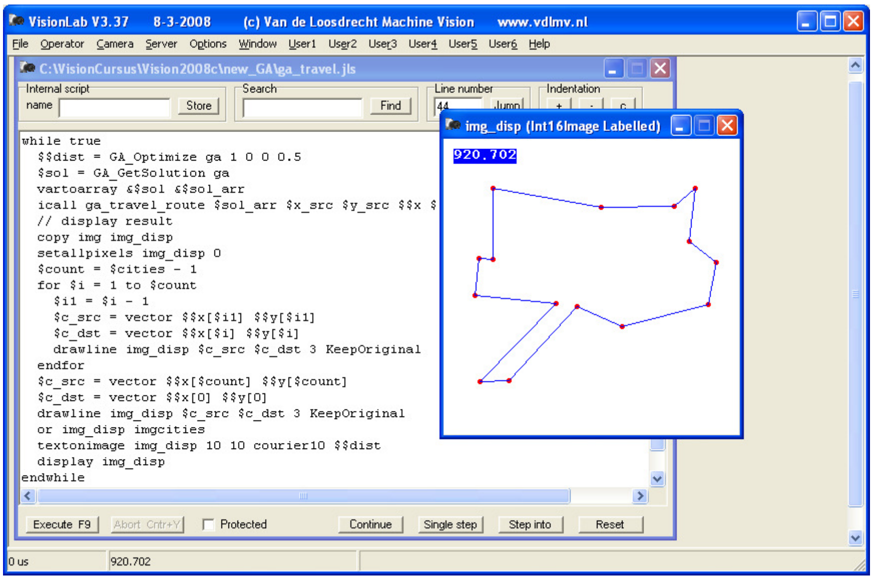
27-aug-18 Genetic Algorithms 48

Running TSP (2)



27-aug-18 Genetic Algorithms 49

Running TSP (3)



27-aug-18 Genetic Algorithms 50

Example: KnapSack

Optimizing problem:

- **Best choice of items that can fit in KnapSack**
- **KnapSack has maximum load in kg (= cost)**
- **Each item has a weight (kg) and a value**

- **"What is the maximum value V that can be achieved without exceeding the cost C"**

- **Solution domain:**
  - **Genetic representation:**
    a list of 0's or 1's indication if item is in KnapSack
  - **Fitness function:**
    total value of items in KnapSack not exceeding max weight

Example: KnapSack

- **Genetic representation:**
  a list of 0's or 1's indicating if item is in KnapSack

  - **VisionLab implementation works with floating point numbers**
  - **For each parameter (item):**
    - **Low = 0**
    - **High = 2**
    - **item included = (Value >= 1)**

Example: KnapSack

- **Fitness function:**
  total value of items in KnapSack not exceeding max weight
  - **Fitness function:**
    if total weight <= max weight then
    total value of items in KnapSack
    else
    0
  - **Note: this fitness function must be maximized**
    GA can only minimize a fitness function
  - **Redefined fitness function:**
    if total weight <= max weight then
    topValue - total value of items in KnapSack
    else
    topValue
  - **Where topValue > sum value of all items**

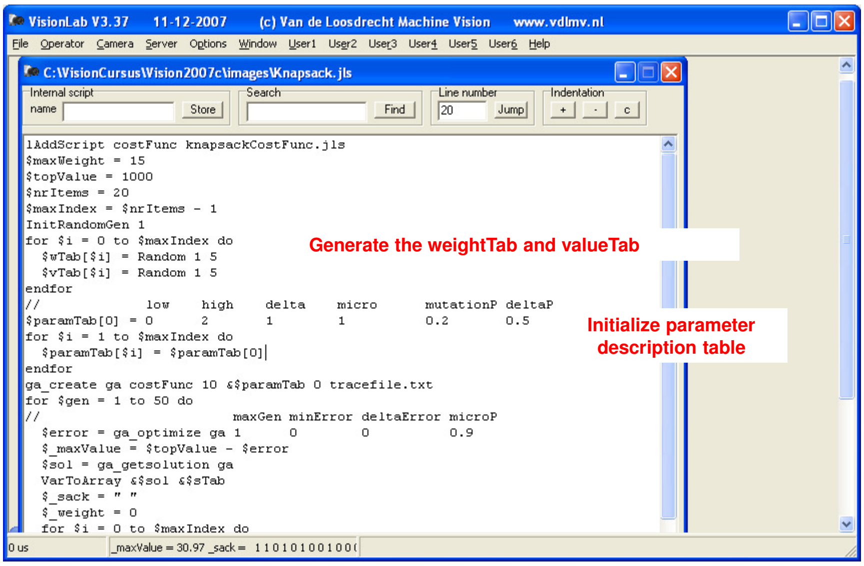
27-aug-18 Genetic Algorithms 53

Demonstration: KnapSack

- **Open scripts:**
  - **knapsack.jls:** **the main program**
  - **knapsackCostFunc.jls:** **calculation of the fitness of a solution**
- **Open variable window**
- **Run script knapsack.jls**

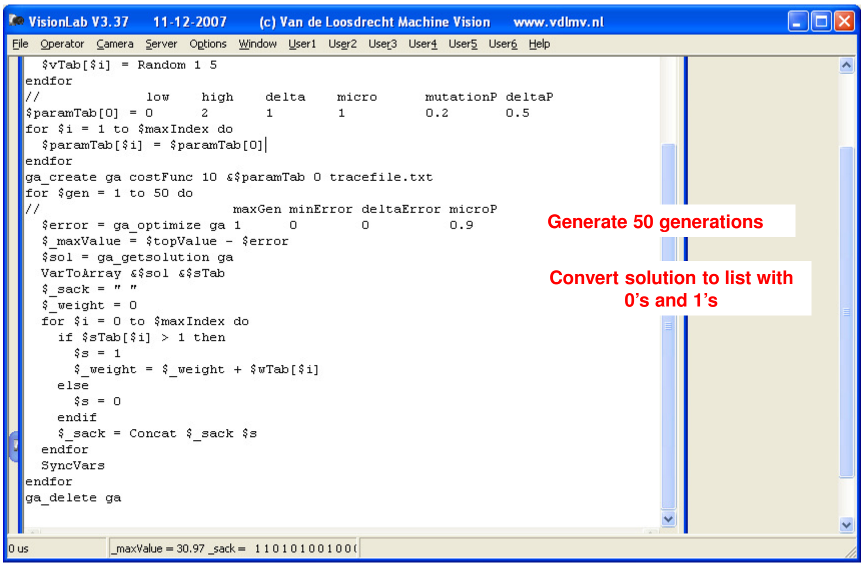
27-aug-18 Genetic Algorithms 54

knapsack.jls



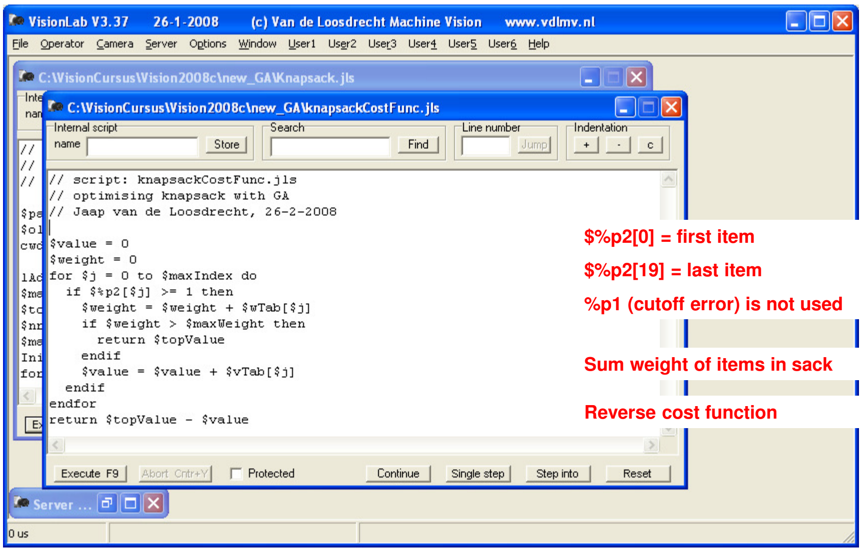
27-aug-18 Genetic Algorithms 55

knapsack.jls



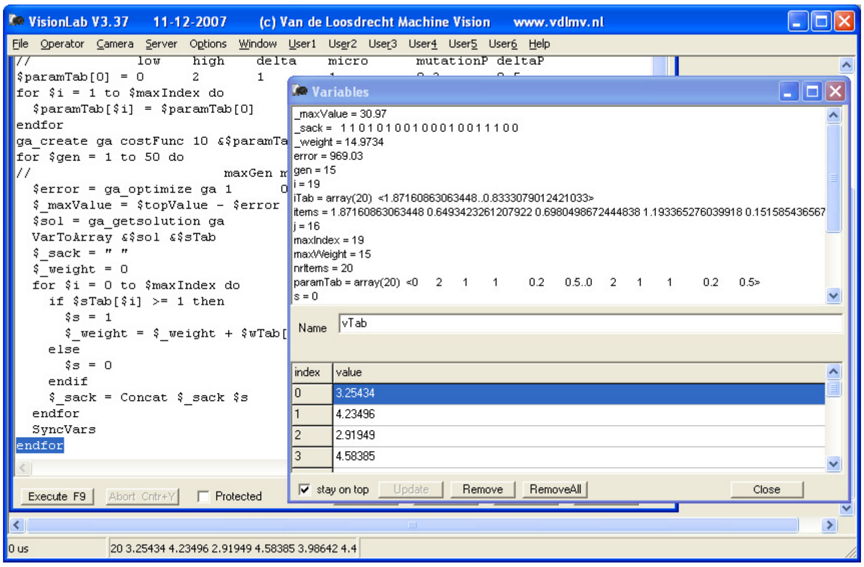
27-aug-18 Genetic Algorithms 56

knapsackCostFunc.jls



27-aug-18 Genetic Algorithms 57

Run knapsack.jls



27-aug-18 Genetic Algorithms 58

Example: Optimising learn error Neural Network

In this example the neural network for recognizing the OCR will be optimized from the chapter about Neural Networks

The training error will be used as fitness criteria

Parameters to optimize:
- Nr nodes hidden layer
- Learn rate
- Momentum

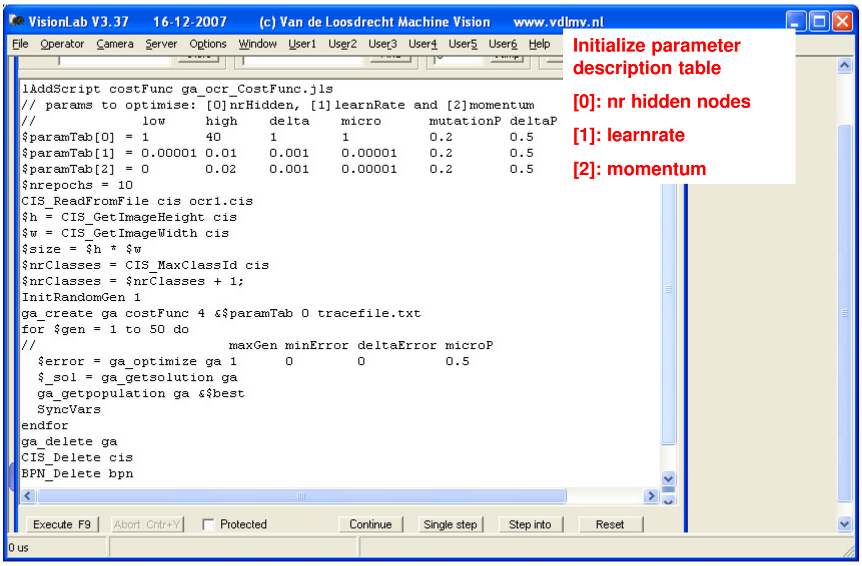
27-aug-18 Genetic Algorithms 59

Demonstration GA optimizing Neural Network

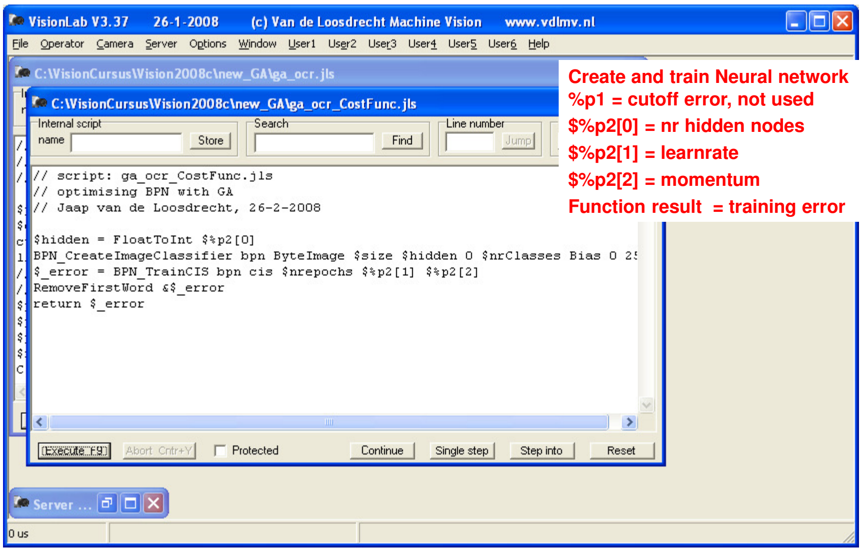
- Open scripts:
  - ga_ocr.jls: the main program
  - ga_ocr_CostFunc.jls: calculation the fitness of a solution
- Open variable window
- Run script ga_ocr.jls
- Note 1: this is time consuming,
  for real optimisation $nrepochs and population size should be greater
- Note 2: if calculations at server side takes much time the client will generated a timeout exception. The timeout value can be changed in the Server Options form in the Server menu.

27-aug-18 Genetic Algorithms 60

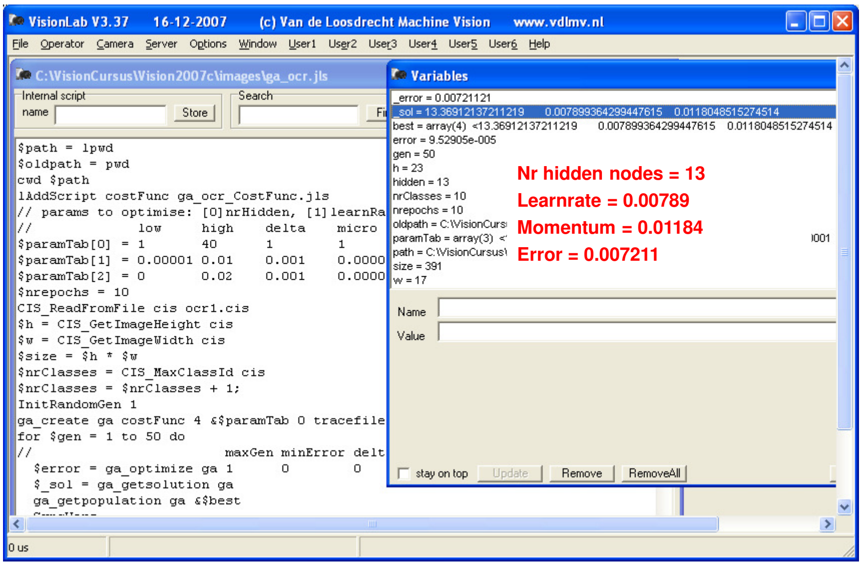
ga_ocr.jls



ga_ocr_CostFunc.jls



Result of GA optimizing



Example: Optimising Neural Network with evaluation set

In this example the neural network for recognizing the OCR will be optimized from the chapter about Neural Networks

The fitness criteria is chosen as:

- **Number of false classifications * 1000 +**
- **Number of low confidences +**
- **Error in output layer**

Parameters to optimize:

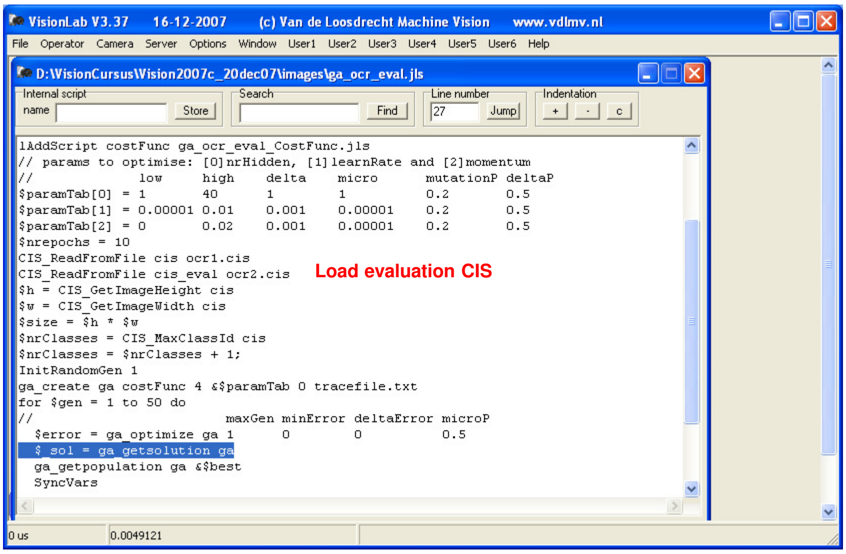
- **Nr nodes hidden layer**
- **Learn rate**
- **Momentum**

Demonstration Optimising Neural Network with evaluation set

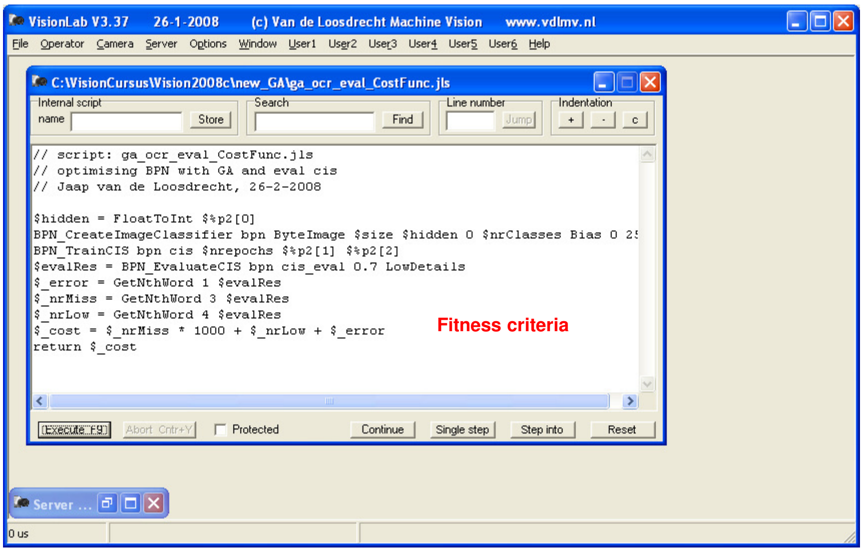
- Open scripts:
  - ga_ocr_eval.jls: the main program
  - ga_ocr_eval_CostFunc.jls: calculation the fitness of a solution
- Open variable window
- Run script ga_ocr_eval.jls

- Note 1: this is time consuming,
  for real optimisation $nrepochs should be greater
- Note 2: if calculations at server side takes much time the client will generated a timeout exception. The timeout value can be changed in the Server Options form in the Server menu.

ga_ocr_eval.jls

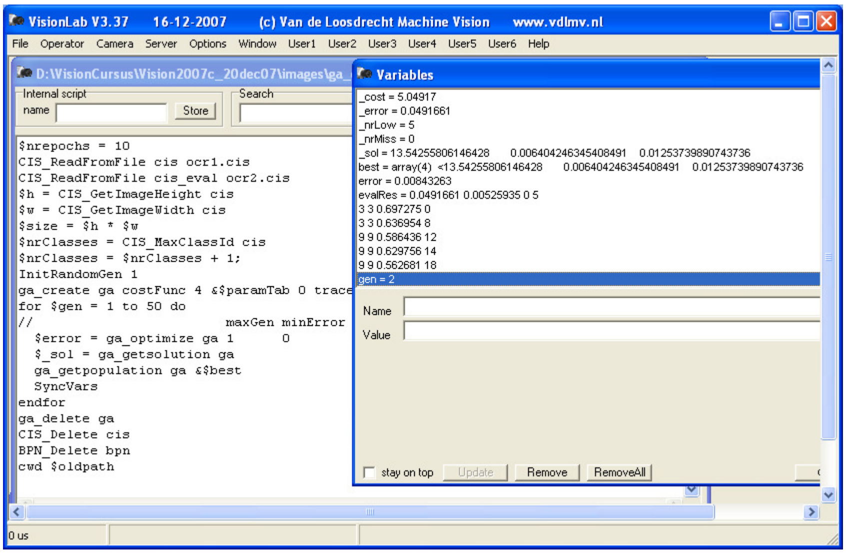


ga_ocr_eval_CostFunc.jls



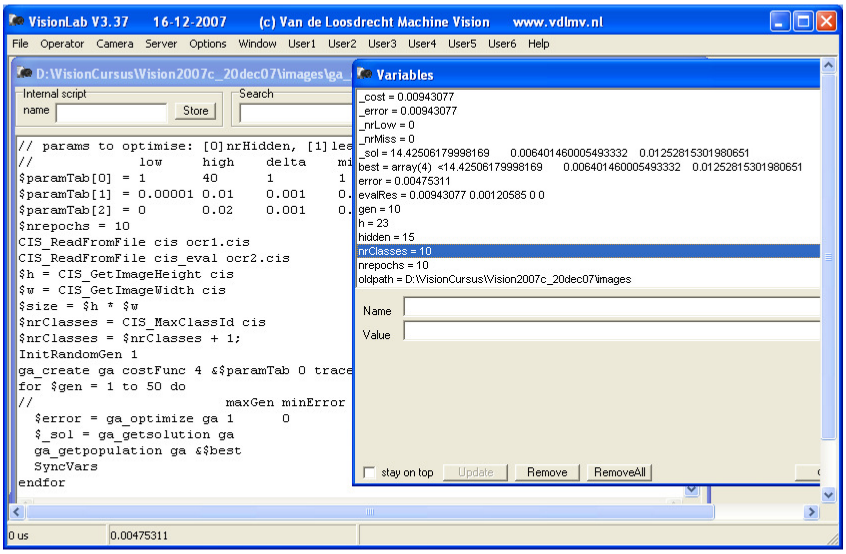
27-aug-18 Genetic Algorithms 67

Result after 2 generations



27-aug-18 Genetic Algorithms 68

Result after 10 generations



27-aug-18 Genetic Algorithms 69

Example: Optimising Neural Network selecting features

In this example the neural network with feature vector for recognizing the OCR with feature vector will be optimized from the chapter about Neural Networks

The fitness criteria is chosen as:

- **Number of false classifications * 1000 +**
- **Number of low confidences +**
- **Error in output layer**

Parameters to optimize:

- **Which features to select**
  If param >= 1 then feature is selected

27-aug-18 Genetic Algorithms 70

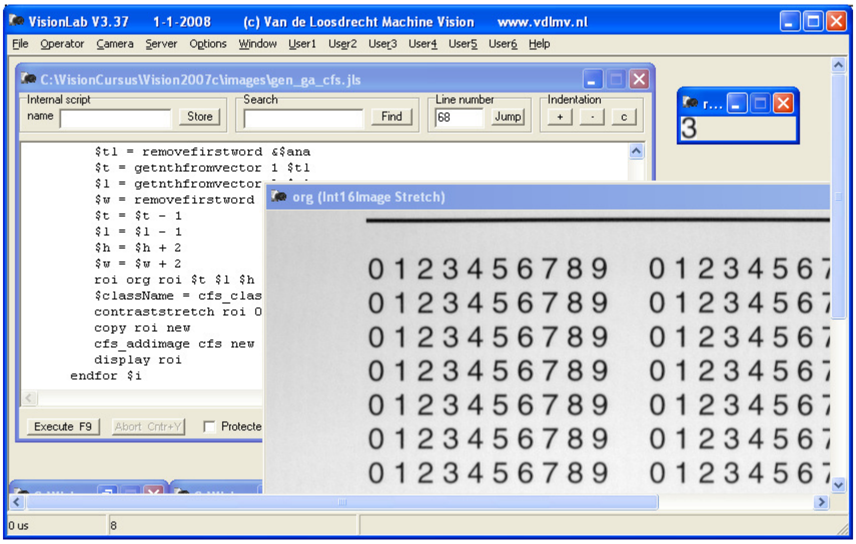
Demonstration Optimising Neural Network selecting features

- Open scripts:
  - gen_ga_cfs.jls — script for generating CFS
  - ga_features.jls: — the main program
  - ga_featuresCostFunc.jls: — calculation the fitness of a solution

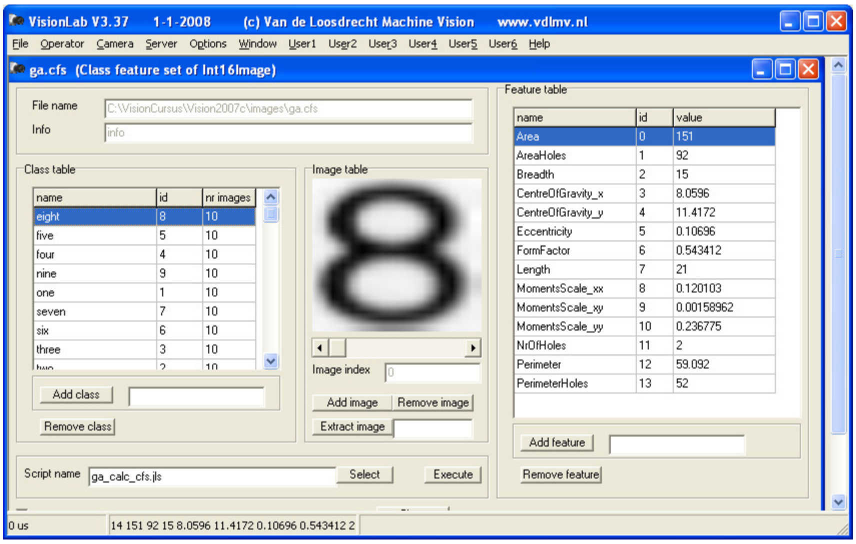
- Run script gen_ga_cfs.jls to generate ga.cfs
- Open variable window
- Run script ga_features.jls

- Note 1: this is time consuming,
  for real optimisation $nrEpochs and population size should be greater
- Note 2: if calculations at server side takes much time the client will generated a timeout exception. The timeout value can be changed in the Server Options form in the Server menu.

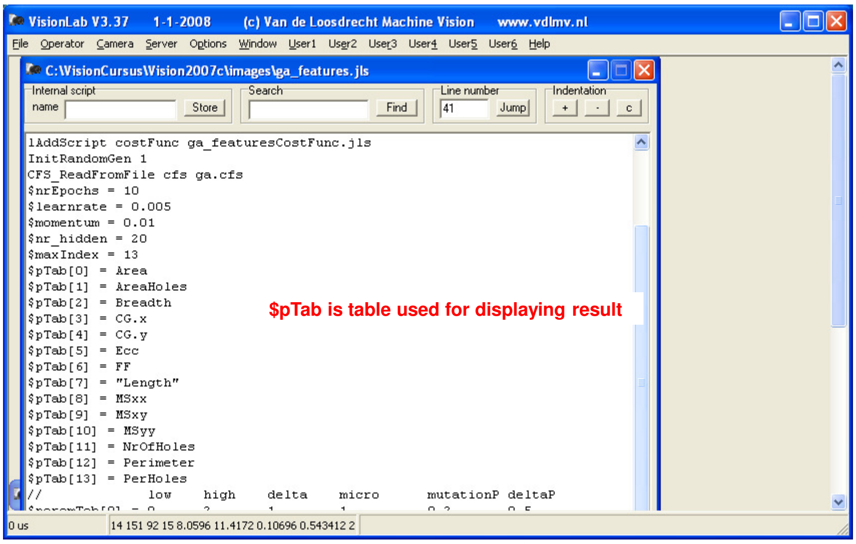
gen_ga_cfs.jls: generate ga.cfs



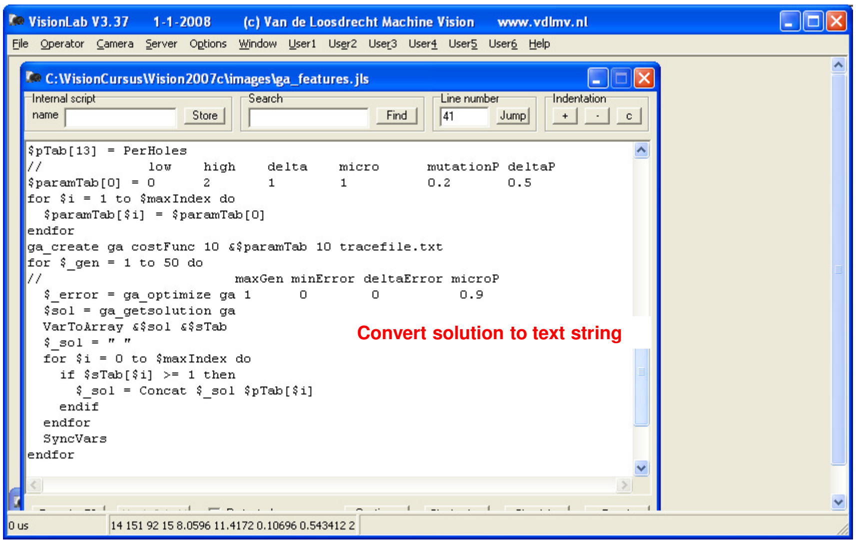
Class Feature Set ga.cfs



ga_features.jls

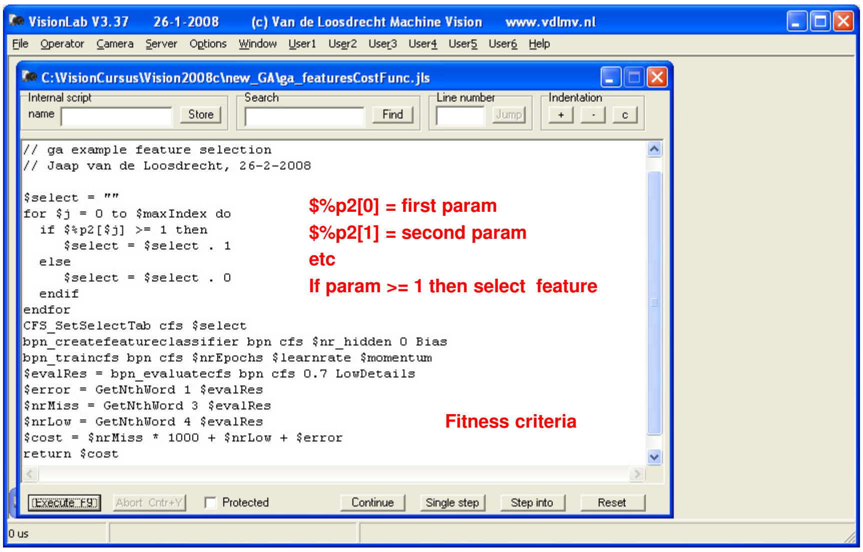


ga_features.jls



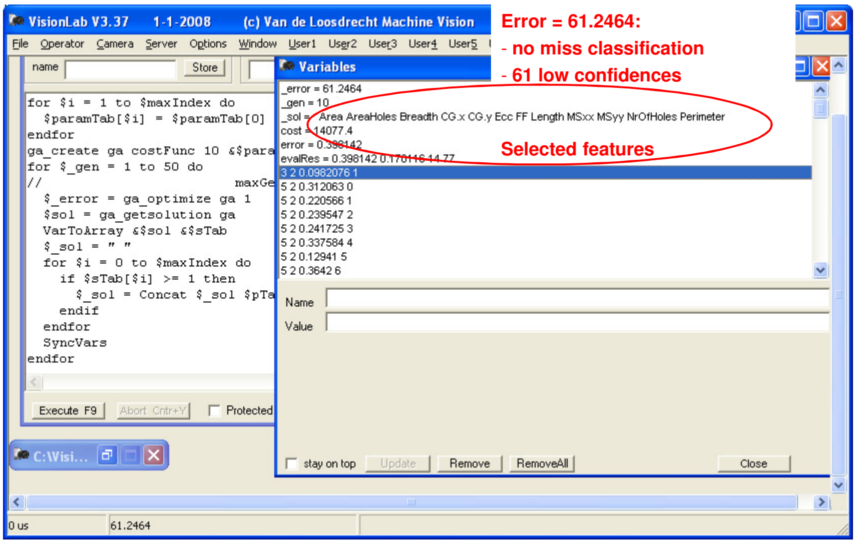
27-aug-18 Genetic Algorithms 75

ga_featuresCostFunc.jls



27-aug-18 Genetic Algorithms 76

Result after 10 generations



27-aug-18 Genetic Algorithms 77