Evolve Thyself:
Algorithm Construction for Image Analysis Using Genetic Programming

Mark Roberts and Ela Claridge

University of Birmingham
School of Computer Science

Introduction

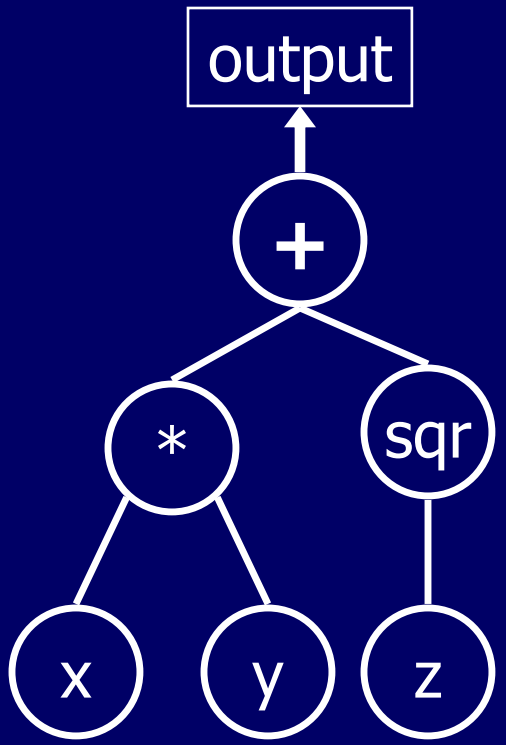
- Traditional image processing systems
  - Usually created by image processing experts
  - Need to capture domain expert's knowledge
  - Task specific
- A better approach would be where
  - System learns by example from domain experts
  - Image processing expertise not necessary
- This work shows some approaches using genetic programming (GP)

GP Introduction

- Genetic programming is a method of evolving computer programs
- A population of programs is used (initially random)
- At each generation, the programs are evaluated and assigned a fitness ...
- ... which is used to determine which individuals are selected for ...
- ... reproduction, usually using crossover or mutation

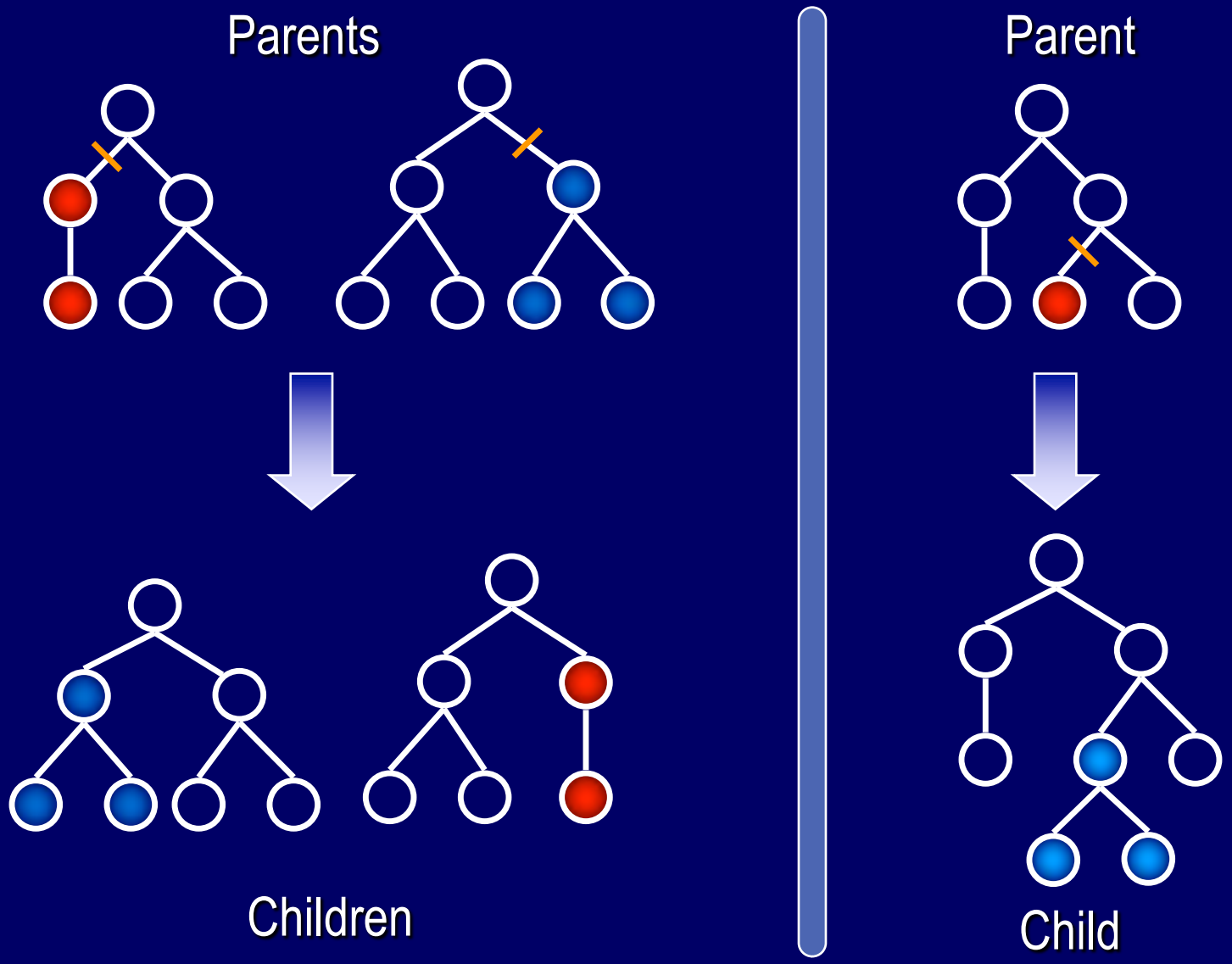
GP Representation

output = (x*y)+sqr(z);



- GP normally uses a tree structure to represent program expressions
  - There are other, better, representations
- Trees are made up from function nodes and terminal nodes
- This sort of representation makes them easy to modify

GP Operators



GP for Imaging

- GP is a proven method for the automatic production of programs
- Many human-competitive results
  - And many human beating results
- Has not been applied to imaging very much
  - Very computationally expensive
  - Population based - we need to do hundreds of thousands of evaluations
  - Image processing algorithms are already very expensive

Three Example Projects

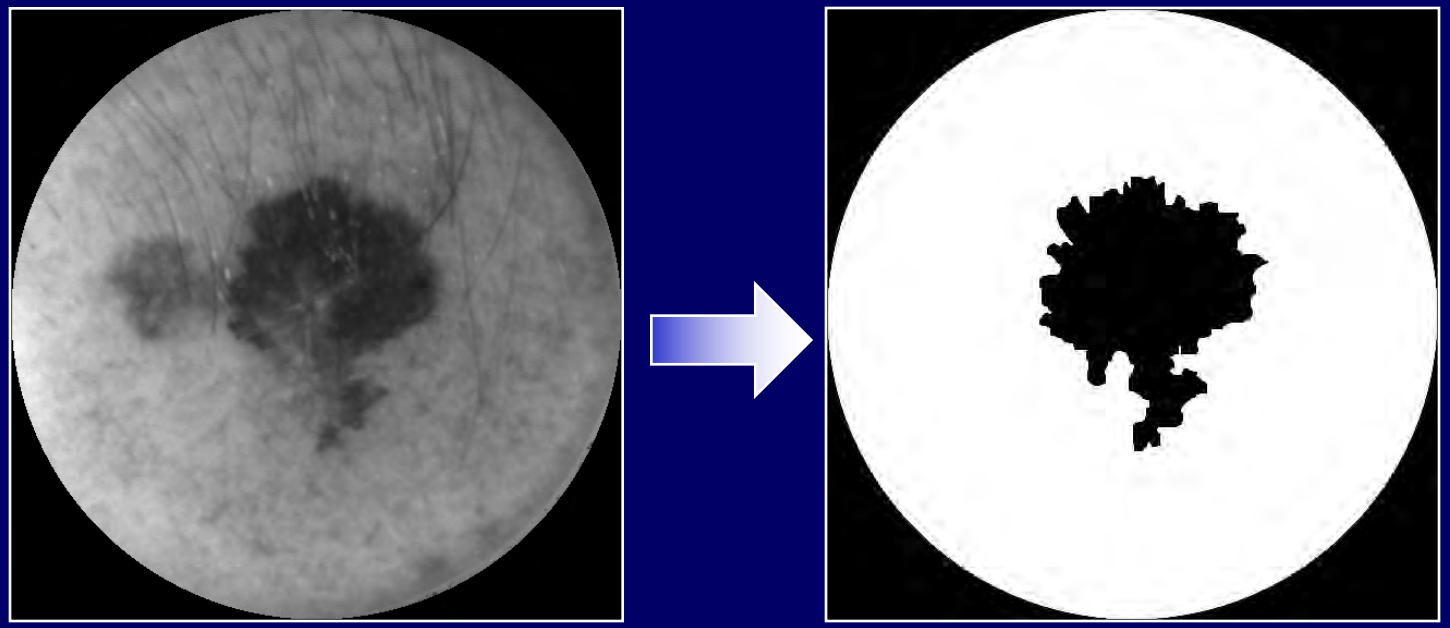
Automatic Segmentation Using Genetic Programming

Object Detection Using Coevolutionary Genetic Programming

A Multistage Approach to Object Detection Using Coevolutionary Genetic Programming

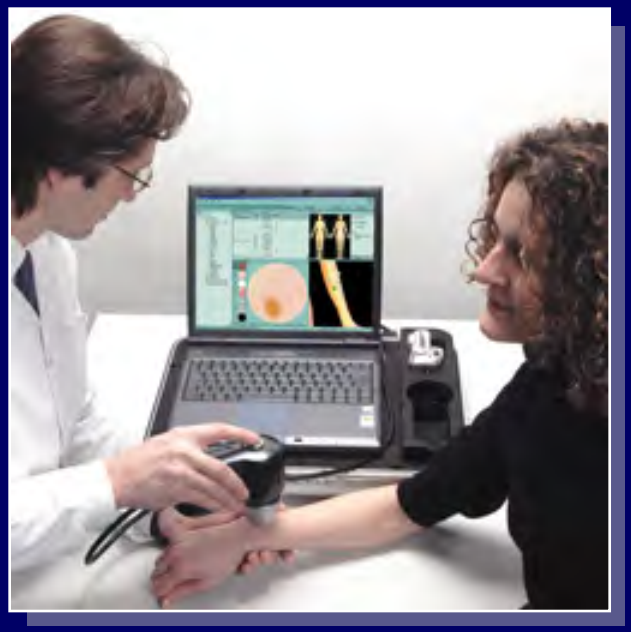
Skin Lesion Segmentation

- Malignant melanoma a growing problem, but well suited to non-invasive screening
- Any attempt at automated diagnosis first requires segmentation



Lesion Dataset

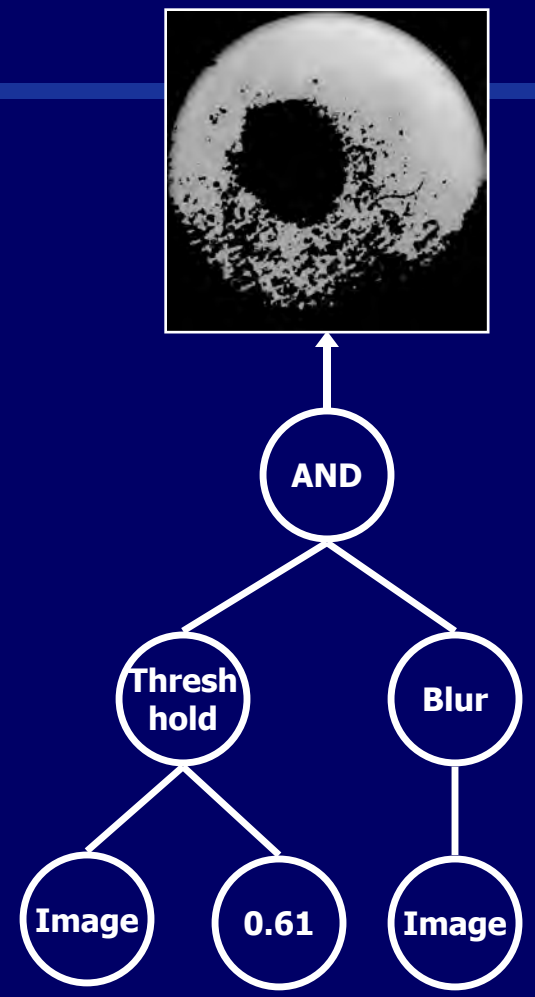
- Set of 100 images used
- Images acquired using a SIAscope™
- Each images has 560 pixel diameter at 40 microns per pixel

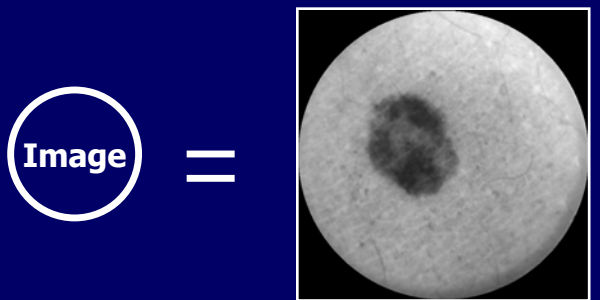


- Ground truth set drawn manually by specialist from Addenbrooke's Hospital, Cambridge
- Divided into training set (8 images) and unseen testing set (92 images)

Image based GP

- Genetic programming used to evolve image processing algorithms
- Tree nodes are
  - Functions - image processing operations
  - Terminals – the input image or constants
- Programs are run, and assigned a fitness based on how well they perform



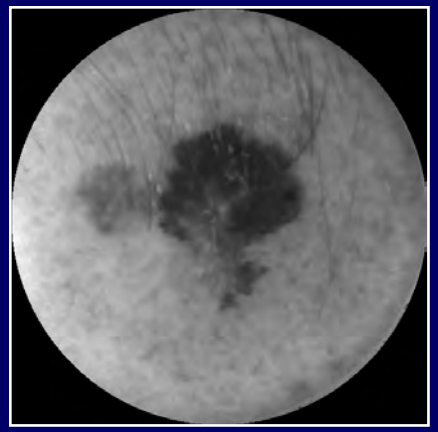


Method (1)

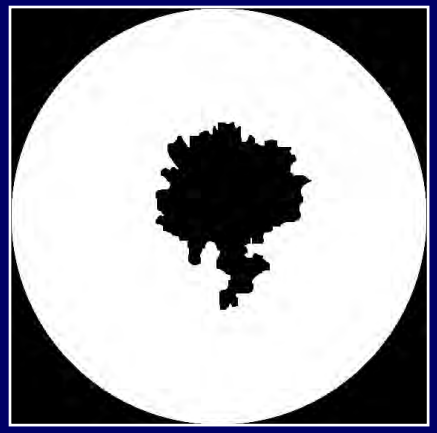
- Random population (5000) of programs created
- Every generation, each segmentation program is run on every training image
- The quality of the segmentation is assessed and a fitness value is assigned based on how well it did
- Population adapted (bred) to create new generation
- At the end, a single fittest program comes out. This is our final solution

Method (2)

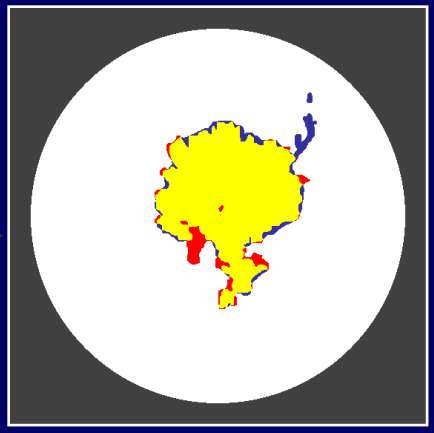
Input Image



Ground Truth



Program Output



TP	26398
FP	2645
TN	168992
FN	2222
Sensitivity	923%
Specificity	985%

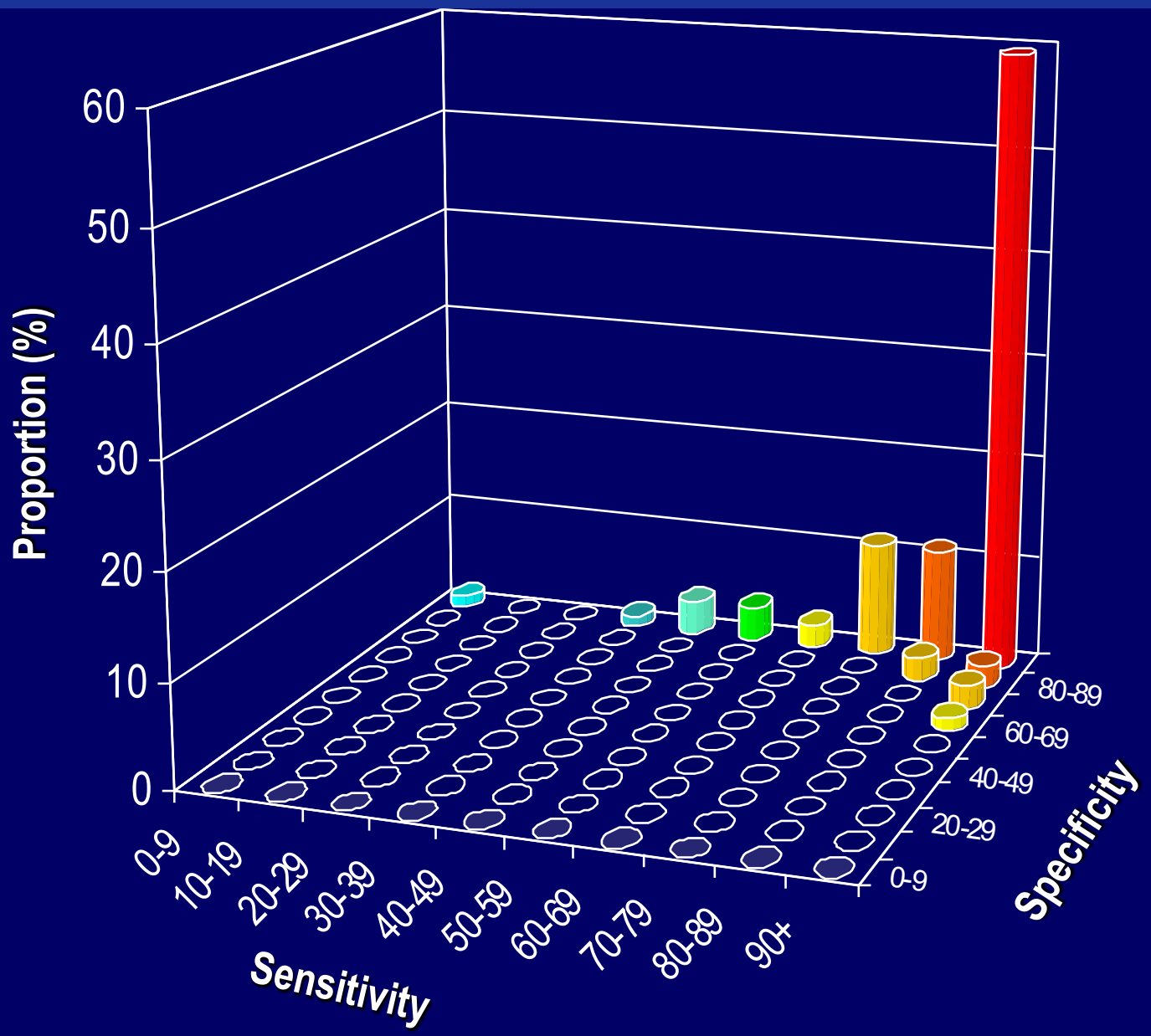
Fitness Function

$$f = \sum_{i=1}^{N} \frac{FP_i}{FP_i + TN_i} + \frac{FN_i}{TP_i + FN_i} e^{\left(10\left(\frac{FN_i}{TP_i + FN_i} - \alpha\right)\right)}$$

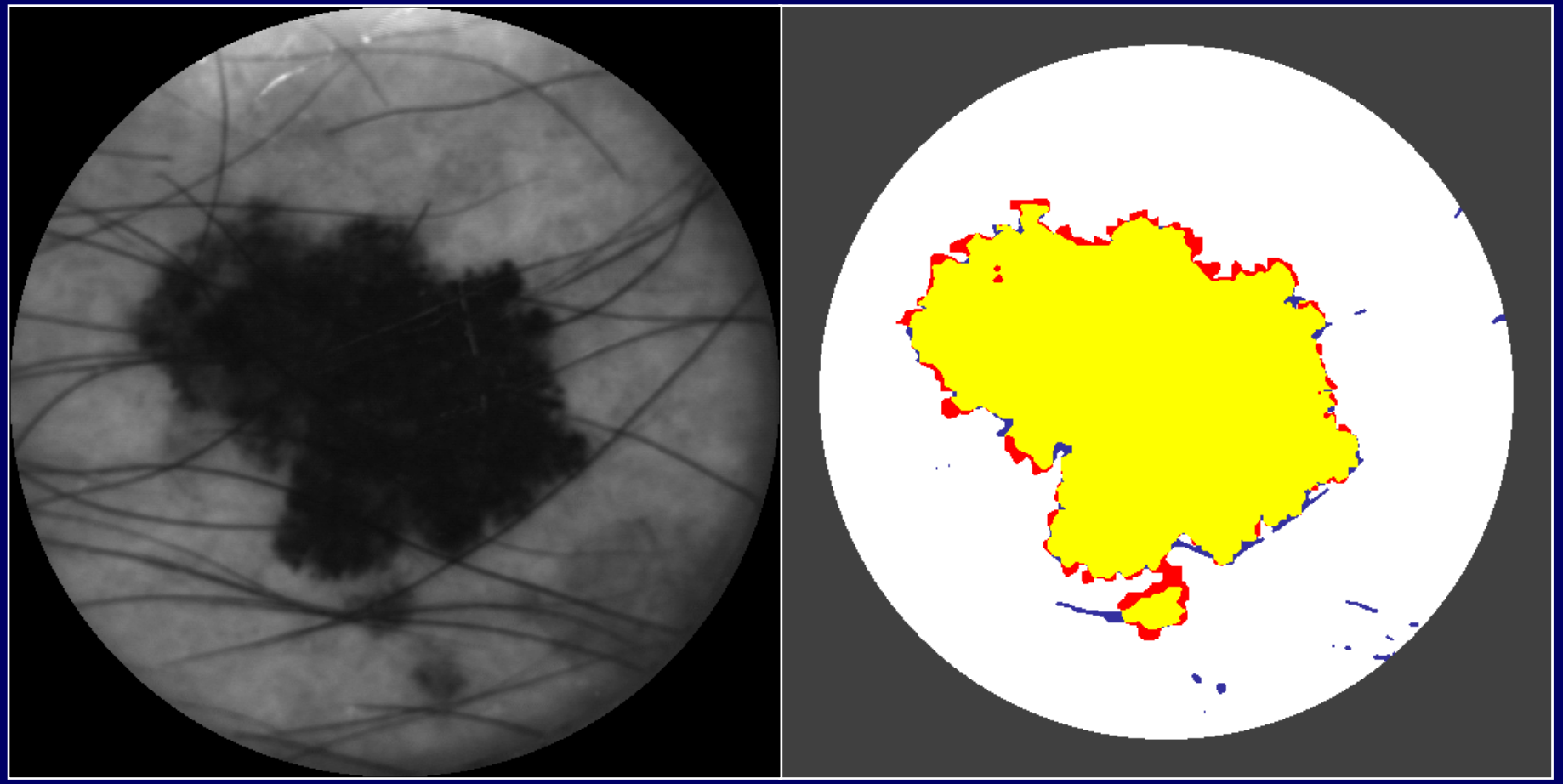
- Quantifies how good an evolved solution is in a single value
- Determines which programs are selected for adaptation
- Provides good balance between sensitivity and specificity
  - Balance changed using α
  - 0.4 in this application

Results

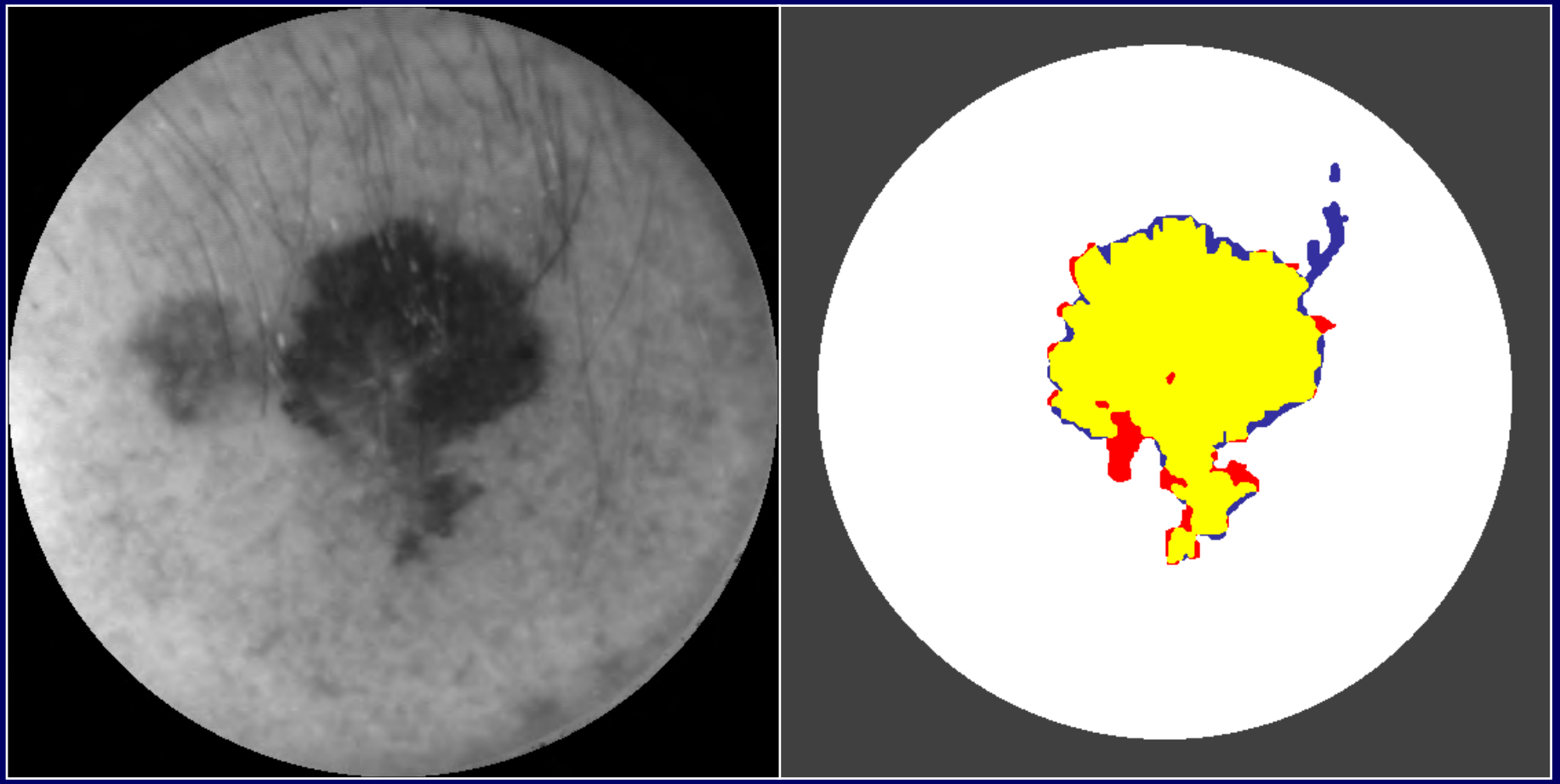
- Average sensitivity 88%
- Average specificity 96%



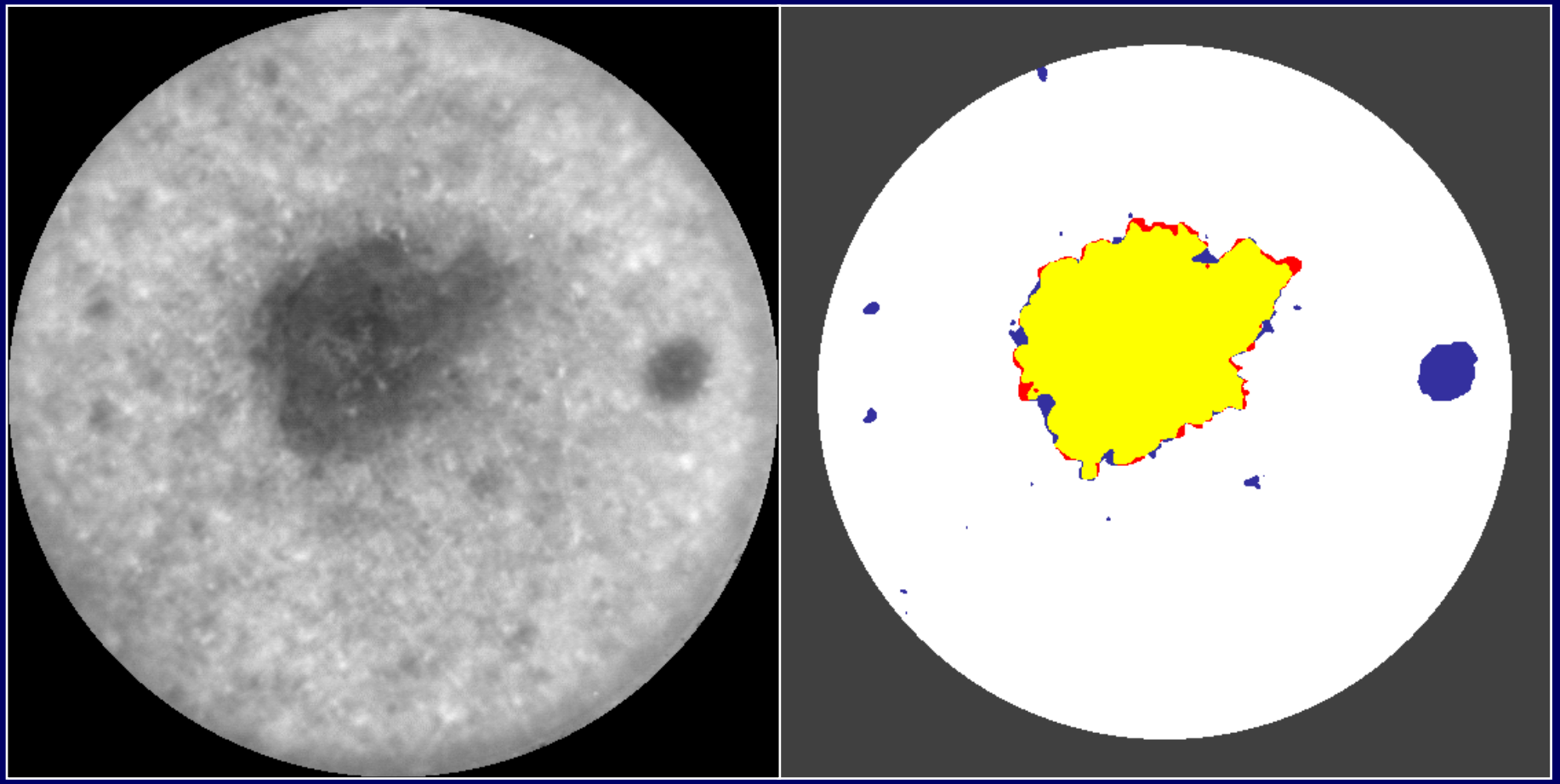
Example Results



Example Results



Example Results



Issues

- Computation time
  - Every program tree is a fairly complicated image processing algorithm. We need to run hundreds of thousands of these in training
  - Took 40 computers 20 hours to train!
  - But only a one-off expense
  - Final algorithm runs in a fraction of a second
- Analysis of programs is difficult
  - But is still possible, unlike other methods (e.g. NNs)
- Intra/inter expert variability
  - Confuses the learning process

Segmentation: Conclusions

- Generally very good segmentations considering ambiguity in training data
- Technique learns to “see” the feature by example
- Not limited to skin lesions
  - No changes needed to evolve algorithms for other imaging problems

Three Example Projects

Automatic Segmentation Using Genetic Programming

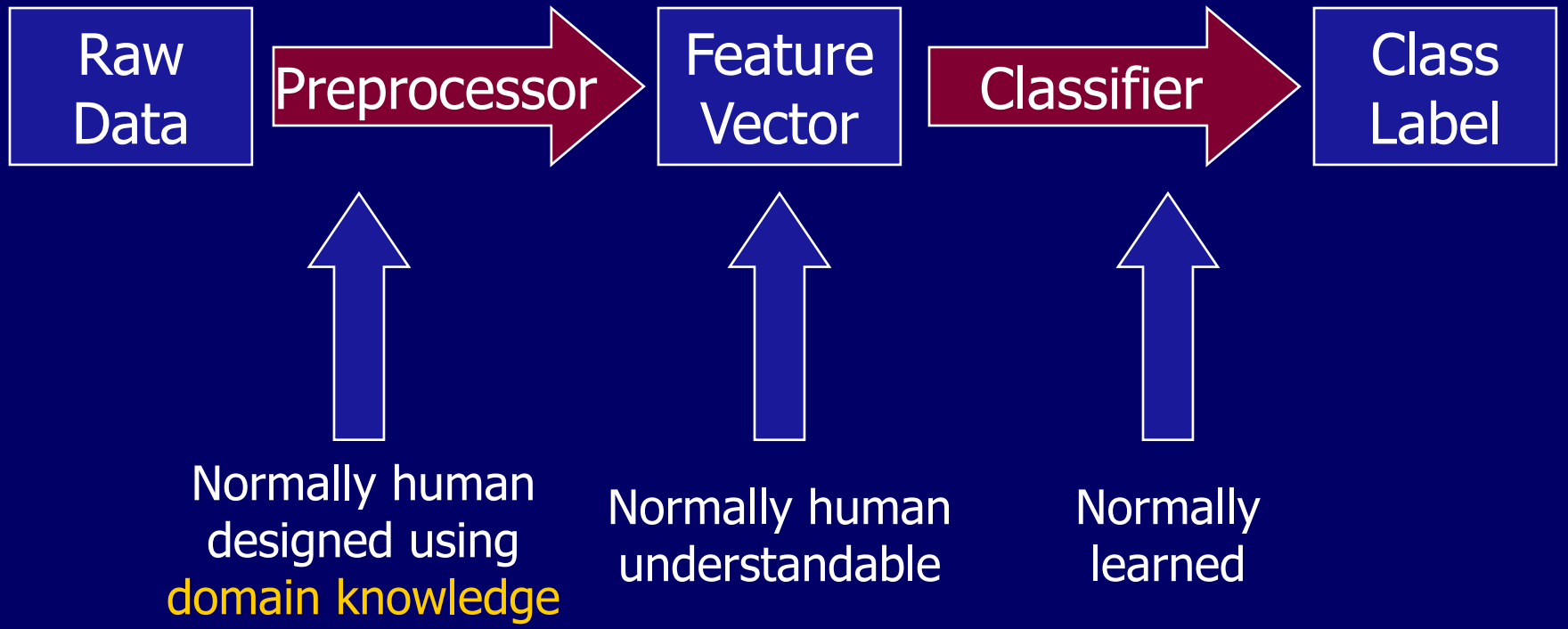
Object Detection Using Coevolutionary Genetic Programming

A Multistage Approach to Object Detection Using Coevolutionary Genetic Programming

Object Detection Problems

- In an object detection problem, we are not interested in the exact boundary
- We just want to find all the objects with as few FPs as possible

Standard Classification Pipeline



- Learning is useful
- But tt operates in a restricted domain because we bias most of the pipeline with our conventional ideas at the feature selection stage

Co-evolutionary GP

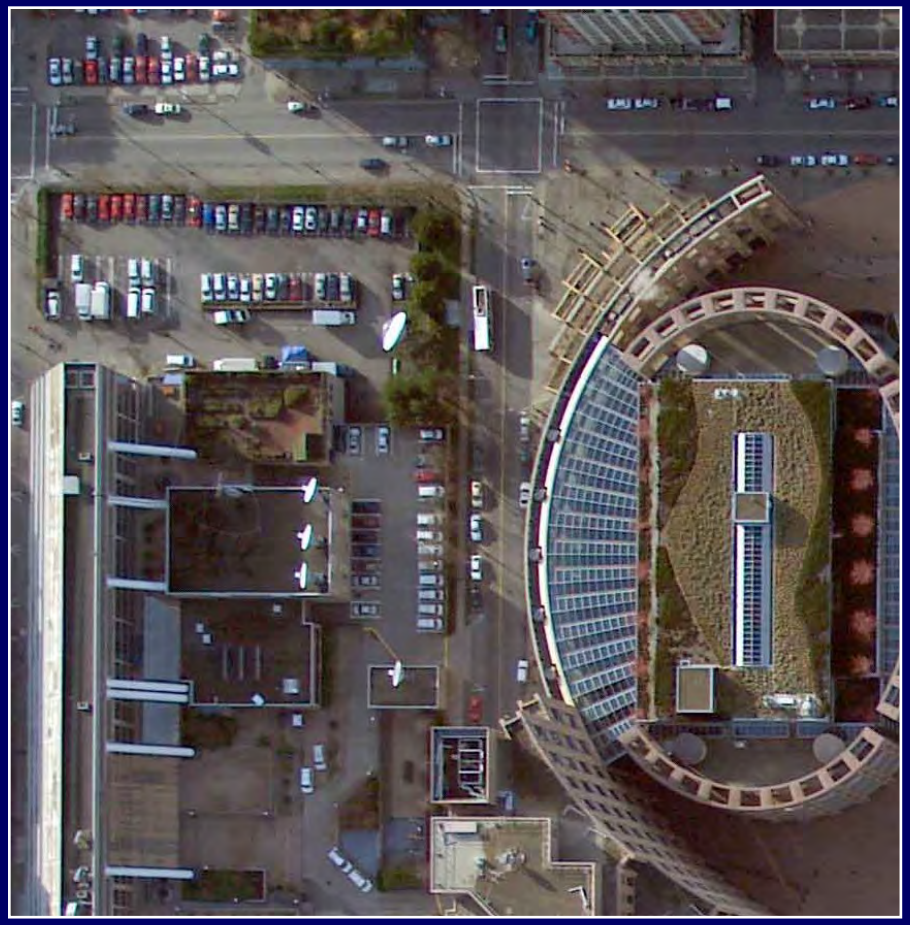
- A solution: evolve a classifier AND the extraction of the features that the classifier uses simultaneously
- We first describe problems with current GP based methods
- Then we show a novel approach using cooperative coevolution
- This method needs no domain knowledge

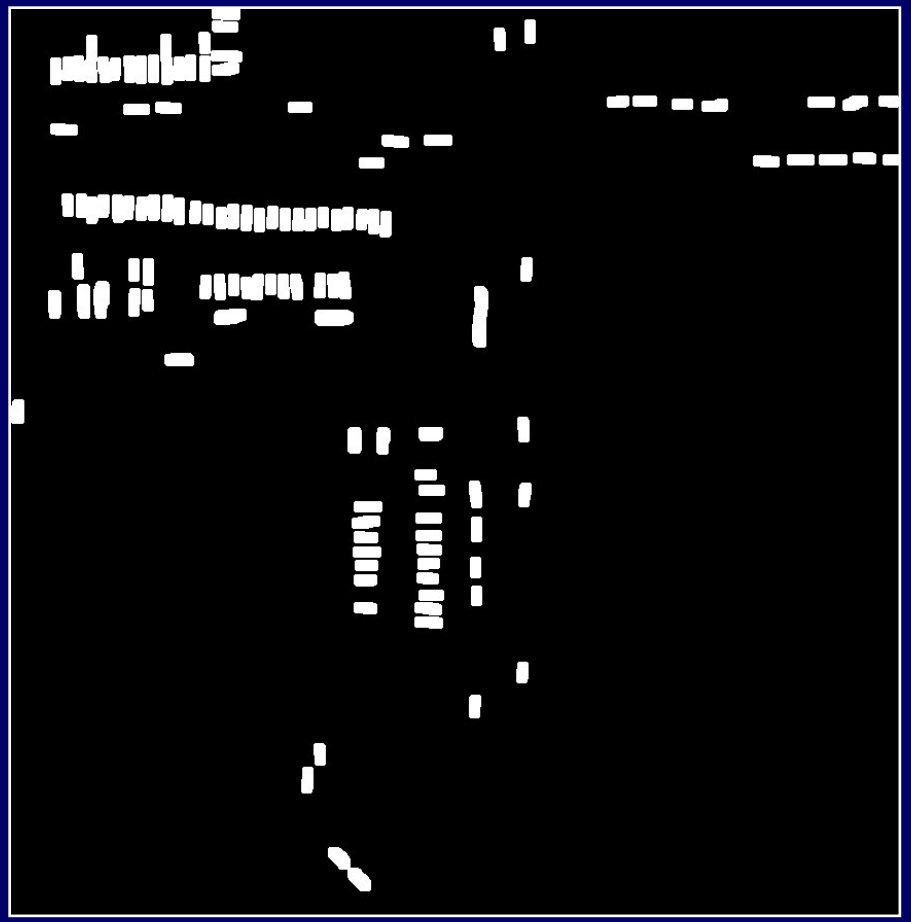
Standard Method

- When using GP or other classifiers a common approach to object detection is
  - At every pixel, extract a number of local features (normally simple statistics) around that pixel
  - Use these features as input to the classifier
  - Output of the classifier represents the “guess”:
    - Object
    - Not object
  - Training is a simple supervised learning problem

Car detection problem: Input Data

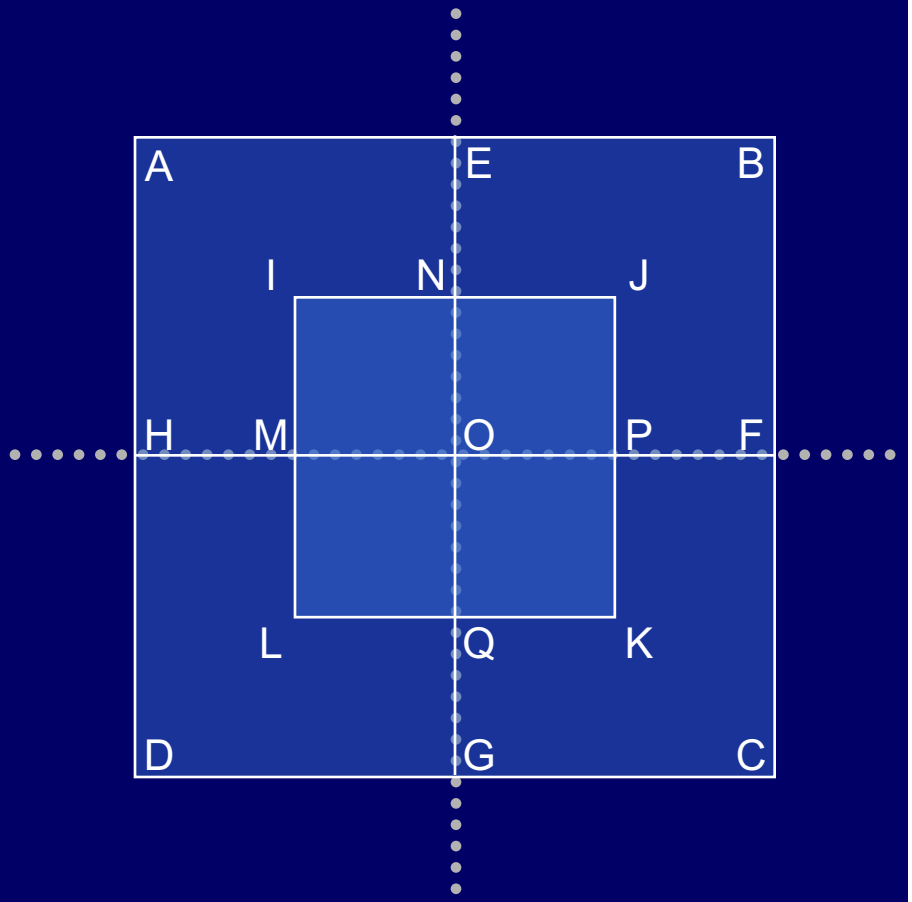
- Raw image and binary mask of targets



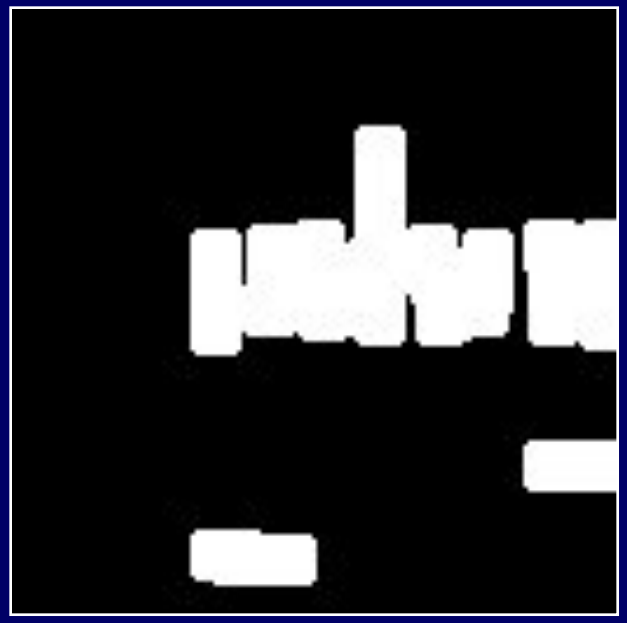


Example Feature Set

Feature set = {F0, F1, F20}



Feature Zone	Mean	Std Dev
Big square A, B, C, D	F0	F1
Small square I, J, K, L	F2	F3
Upper left square A, E, O, H	F4	F5
Upper right square E, B, F, O	F6	F7
Lower right square F, C, G, O	F8	F9
Lower left square H, O, G, D	F10	F11
Line H, F	F12	F13
Line E, F	F14	F15
Line M, P	F16	F17
Line N, Q	F18	F19



x	y	F0	F1	…	Class
16	16	083	025	…	0
17	16	086	026	…	0
18	16	082	028	…	0
19	16	076	038	…	0
20	16	06	035	…	0
21	16	055	04	…	0
22	16	051	038	…	0
23	16	048	037	…	0
24	16	044	036	…	0

Basic Procedure

- Extract features
- Use genetic programming to evolve classifiers that use the features as inputs
- Run the classifier at every pixel, producing an output image
- Calculate fitness of the resulting image, usually in terms of TP, FP, and FN
- Keep evolving until some termination criteria are met

- This technique works quite well. Many successful examples on hard problems in the literature since 1992

Problems

- We do not know the optimal feature set
- The chosen set too general (usual outcome)
  - May not be even be able to solve the problem with the available features
  - Extracts useless features (time consuming)
- The chosen set too specific
  - Designed for the exact problem
  - Needs redesigning for each new instance of the problem
- Advantageous to find an optimal feature set

Solution

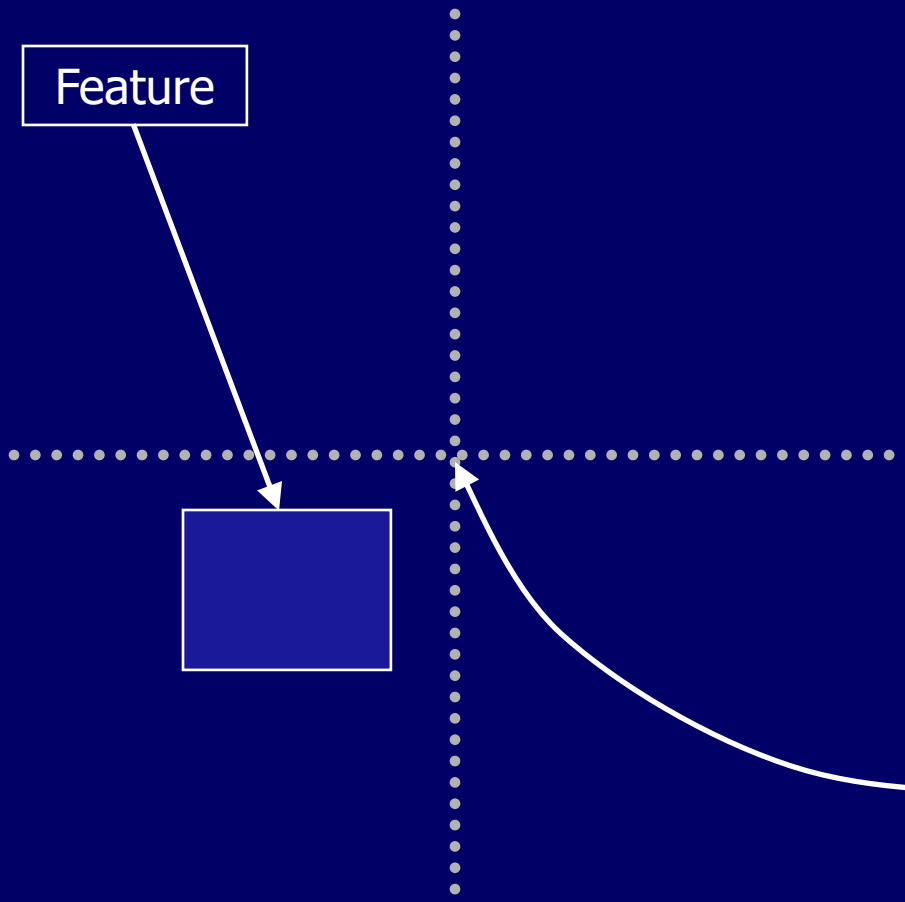
- Would it be great if you just supplied training data, and both feature extraction AND classification was learned?
- Ideally, classifiers AND the features they use would both adapt to each other and to the data
  - Difficult – like learning to type while the keyboard layout is changing
- Cooperative coevolution allows us to do this

Cooperative Coevolution

- In cooperative (as opposed to competitive) coevolution –
  - Several populations evolve different parts of a solution
  - Individuals from two or more populations are selected
  - These are combined to form a complete solution
  - The fitness of an entire solution is calculated
  - Fitness is assigned back to the constituent parts of the solution

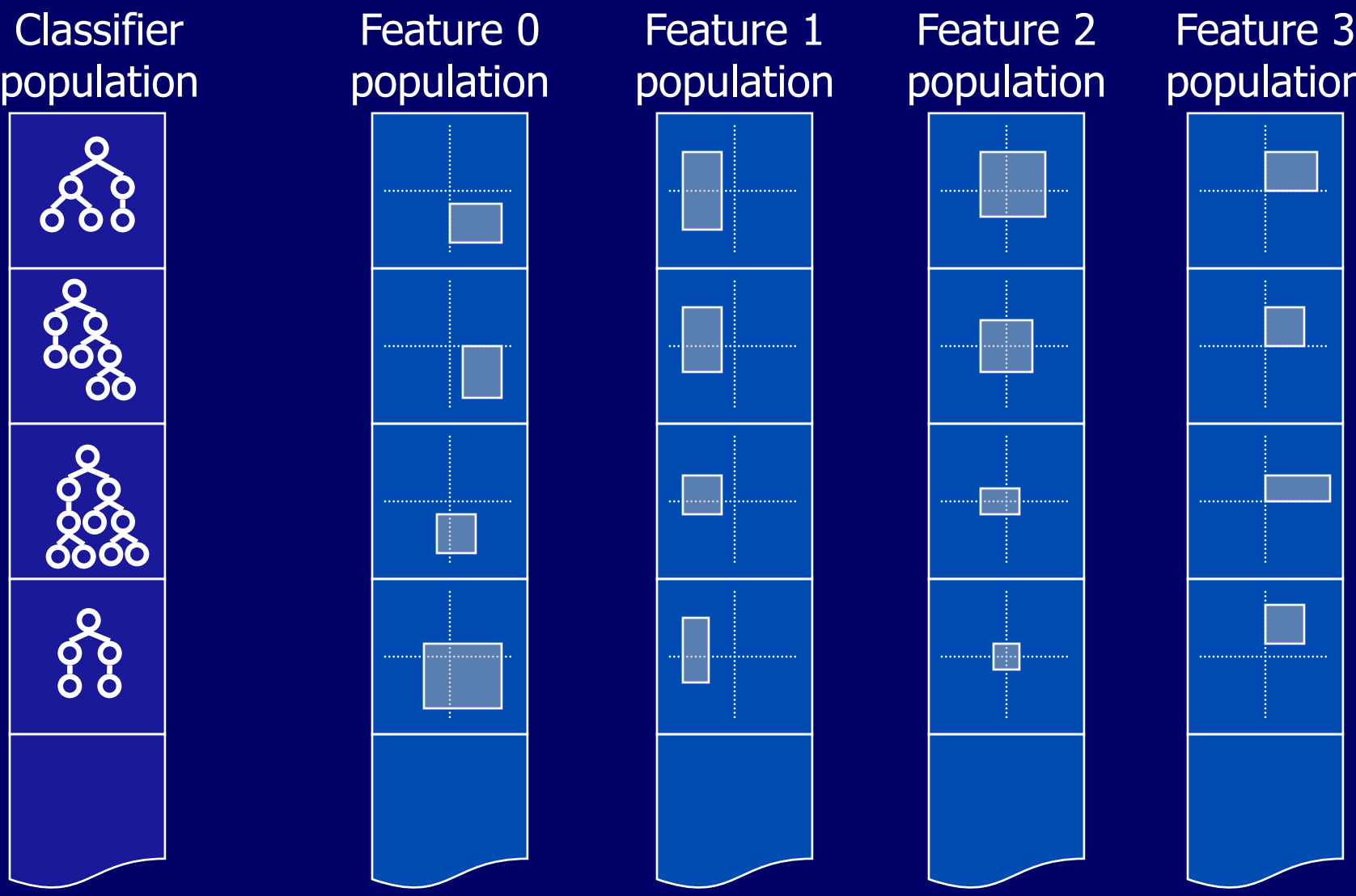
Feature Extractor Representation

- We use cooperative coevolution to evolve feature extractors and classifiers

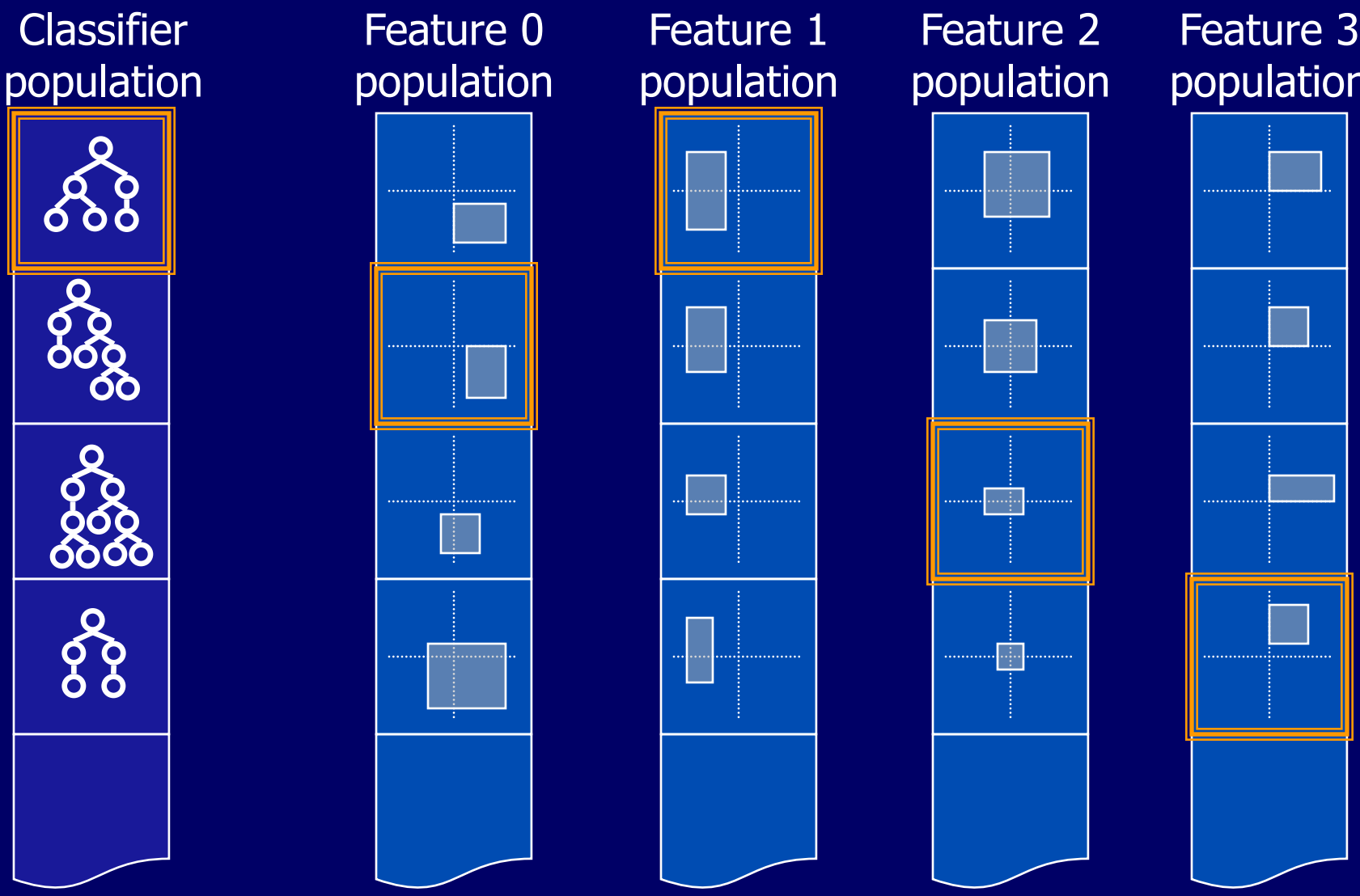


- Features extractors are represented as rectangular zones, relative to the current pixel
- For each we store
  - Position, size, statistic, shape

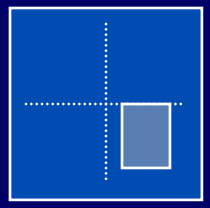
Population Structure

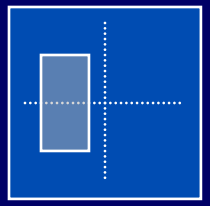


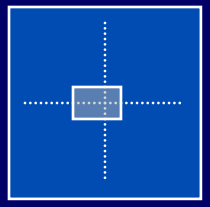
Population Structure

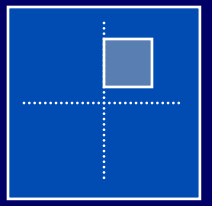


Complete Solution

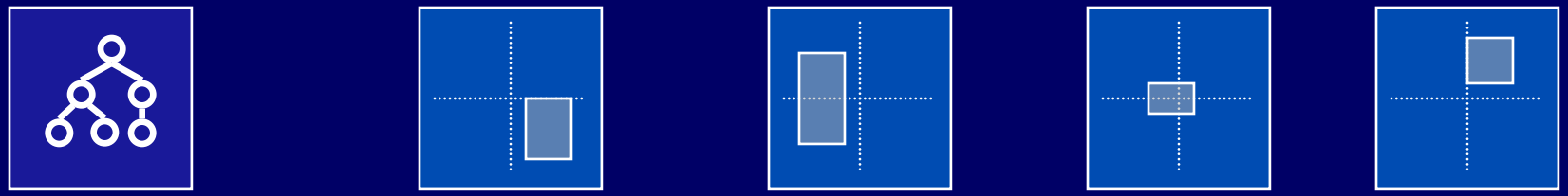


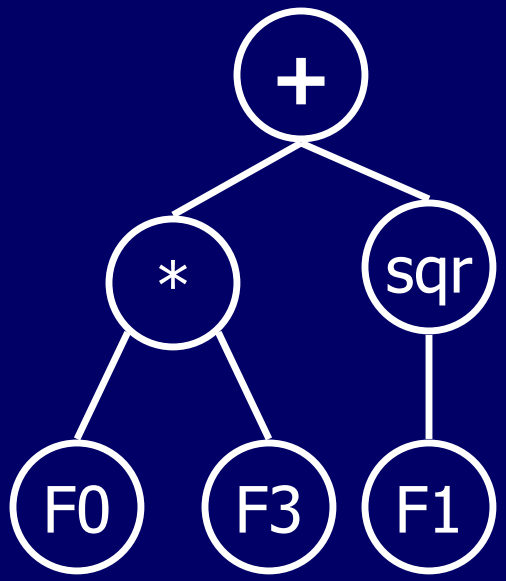


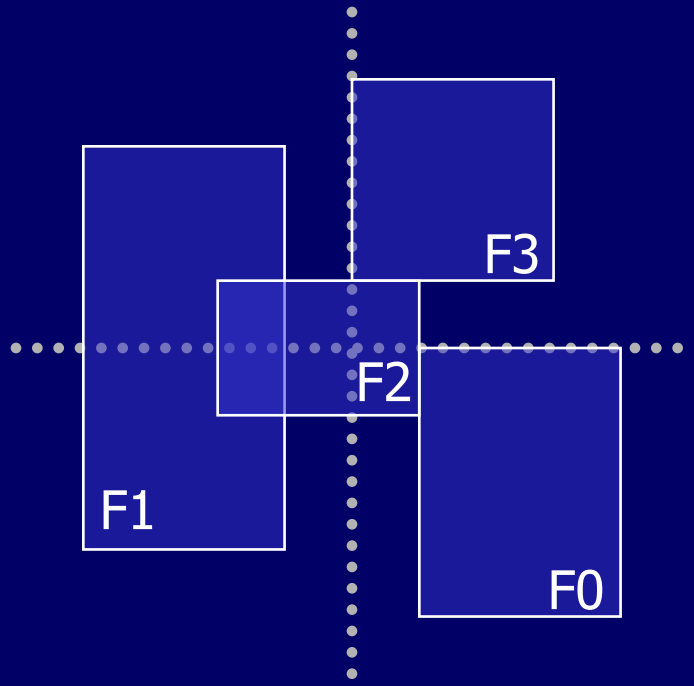




Complete Solution



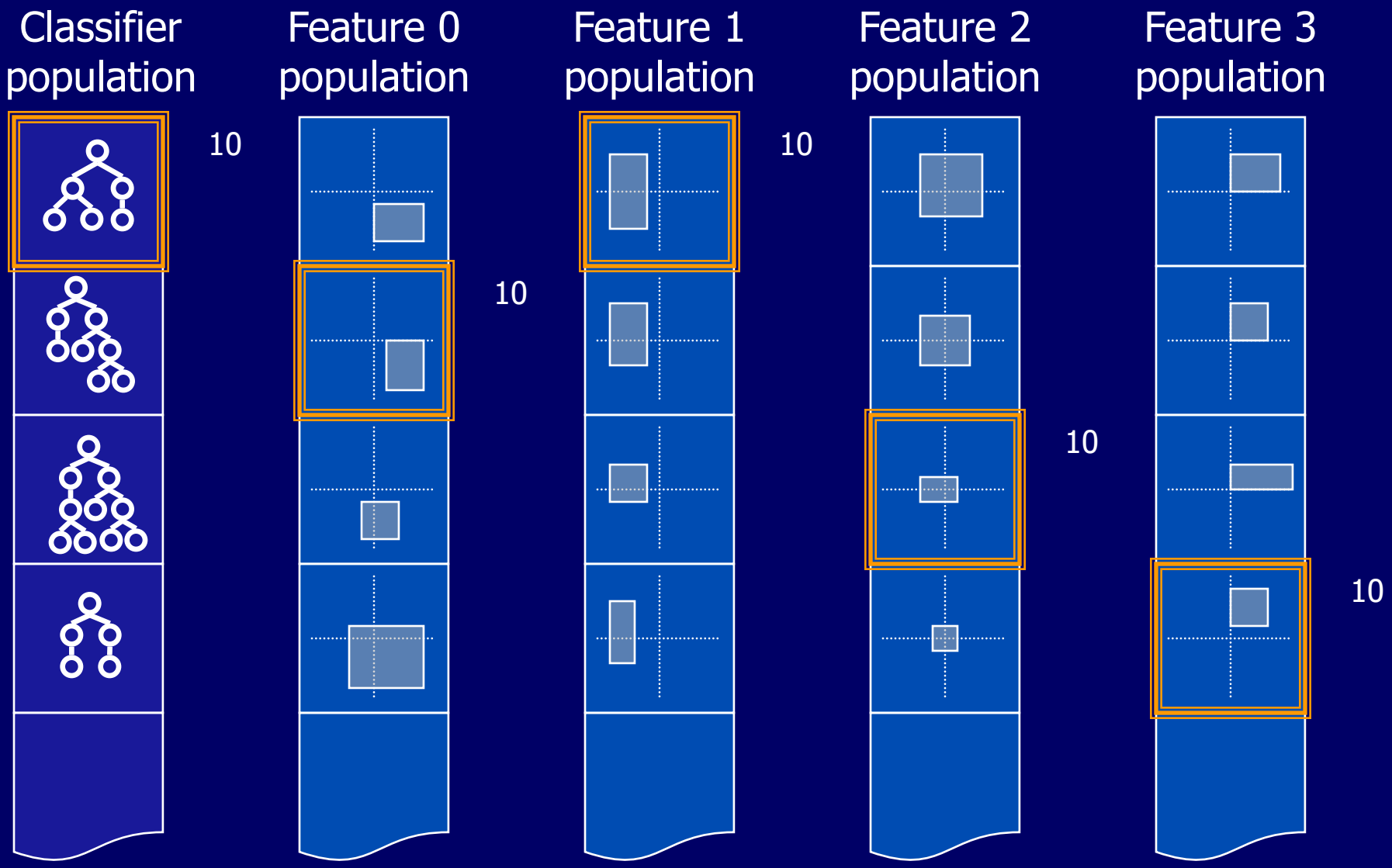




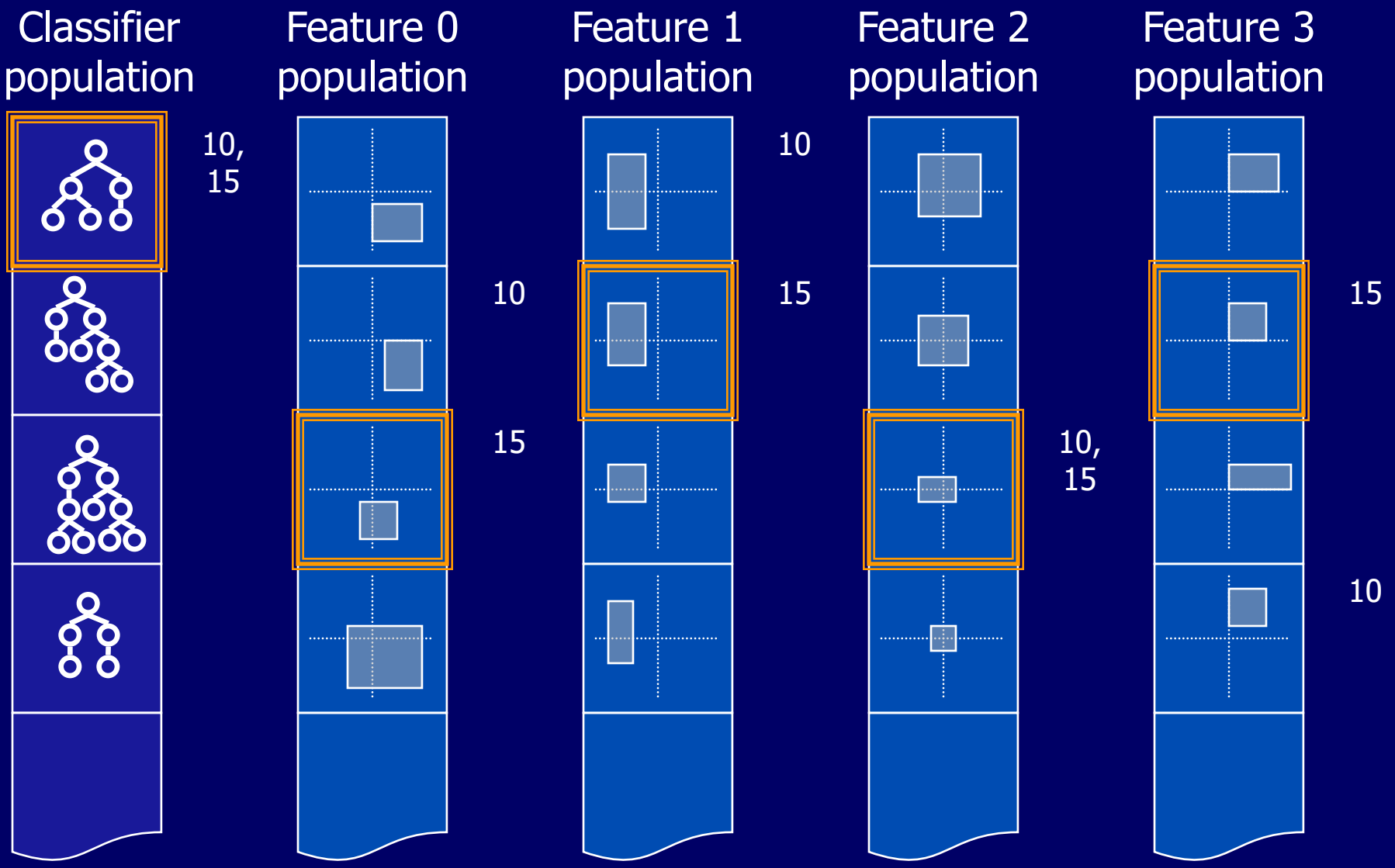
Cooperative Coevolution algorithm

- Randomly select collaborations to form a complete solution
  - i.e. a classifier and a set of feature extractors
- Evaluate the solution's fitness
- Store the fitness with each constituent part
- Do all of this several times per detector
- At the end of the generation, assign a final fitness back to each individual
- Breed each population independently

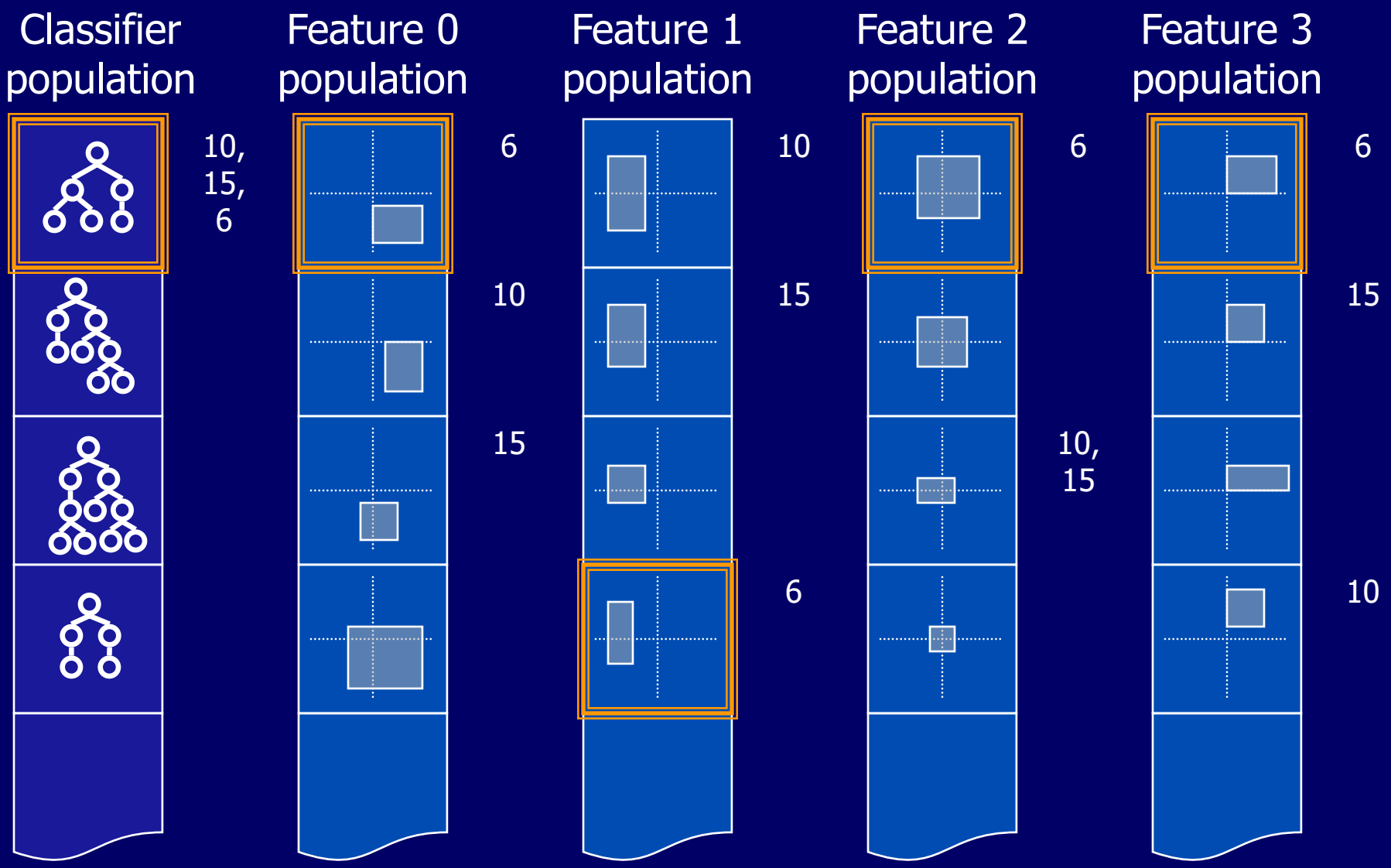
Population Structure



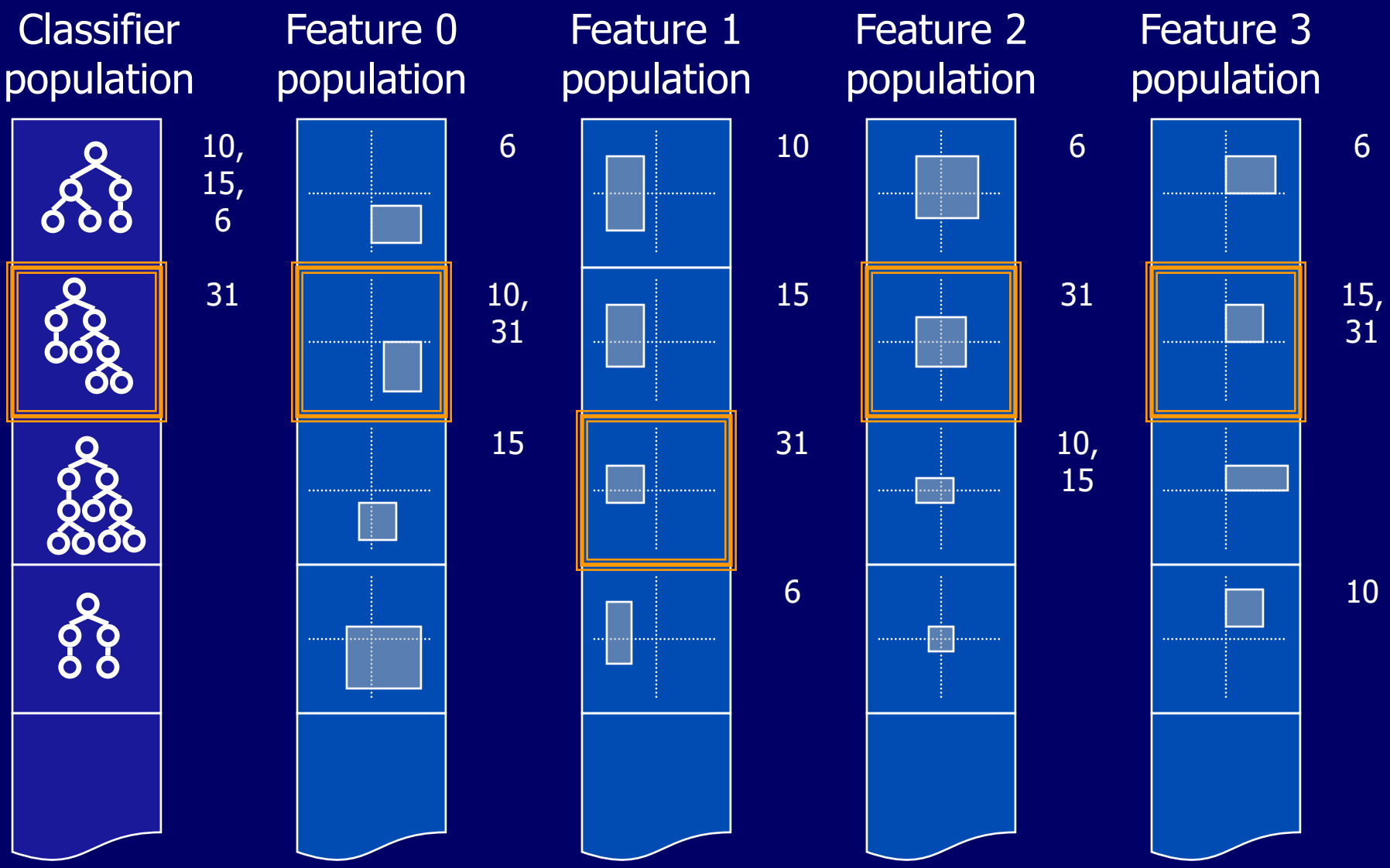
Population Structure



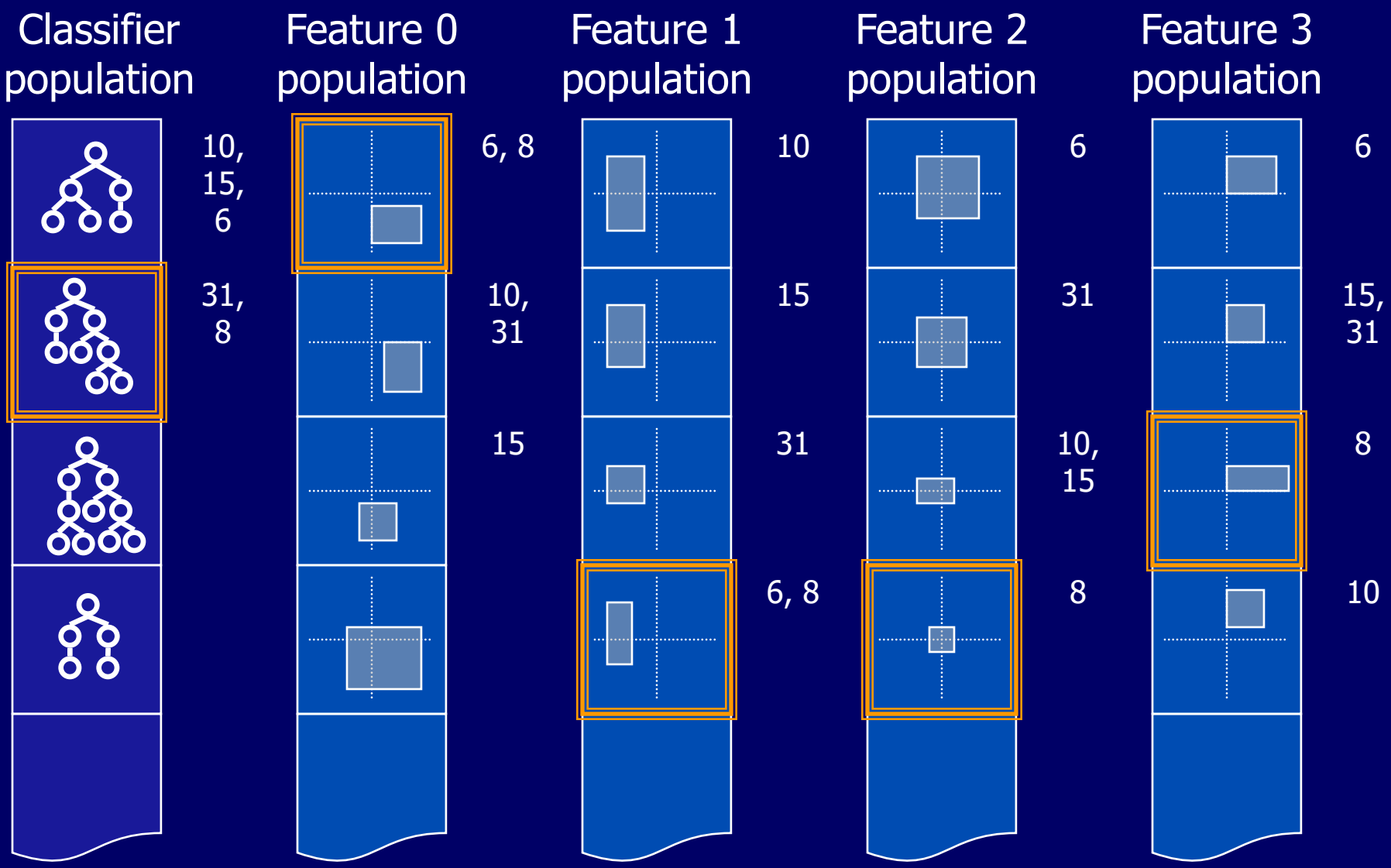
Population Structure



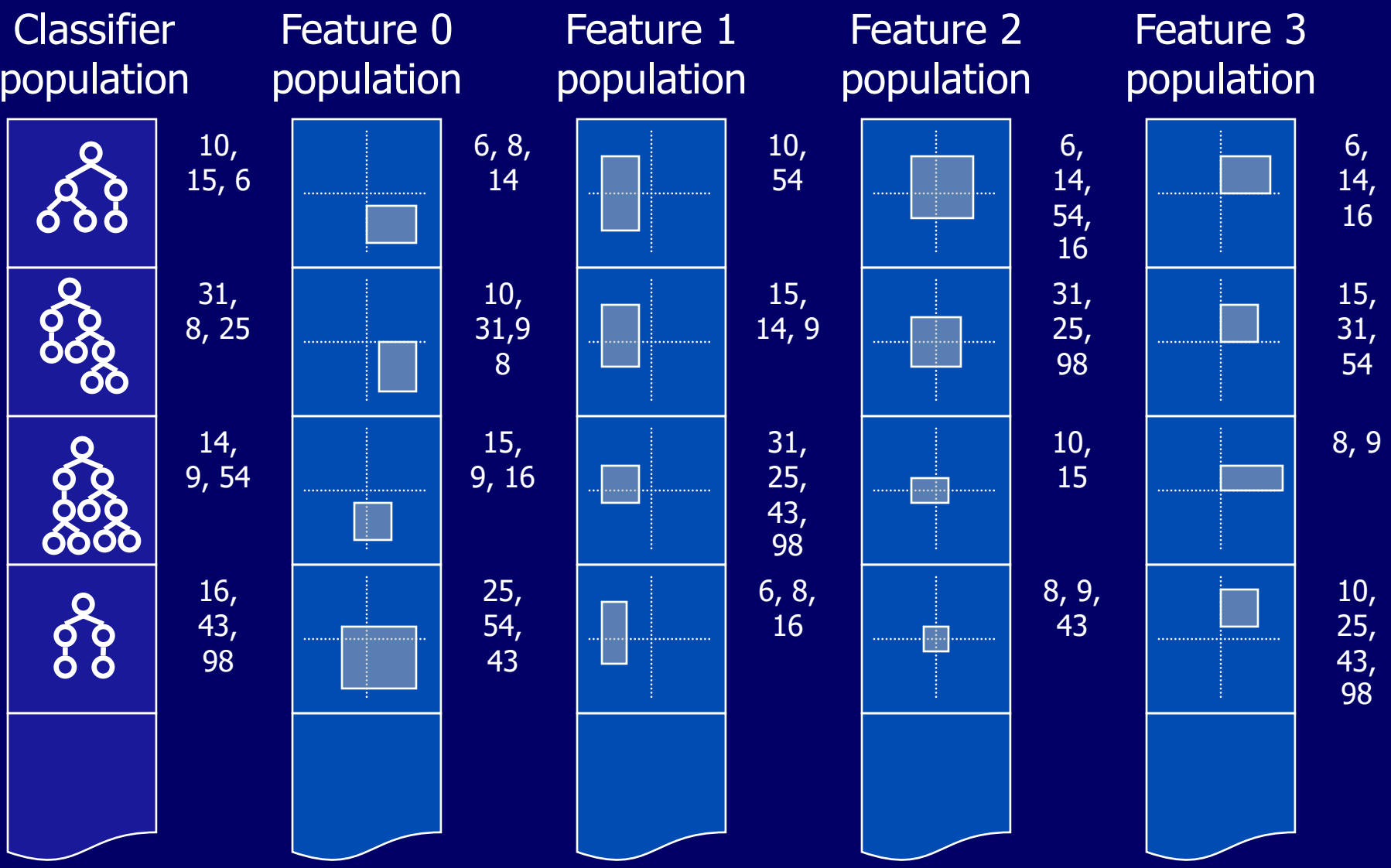
Population Structure



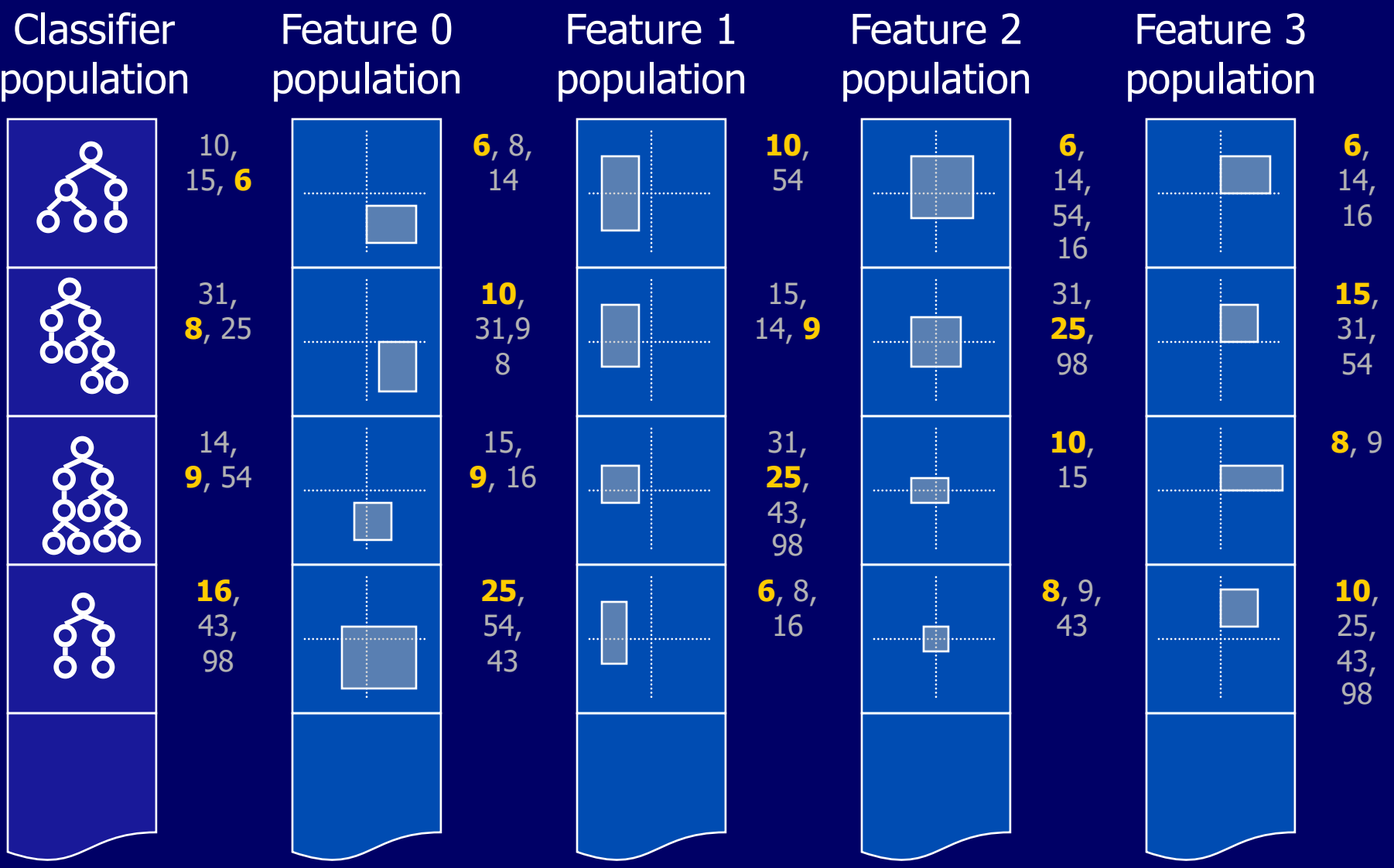
Population Structure



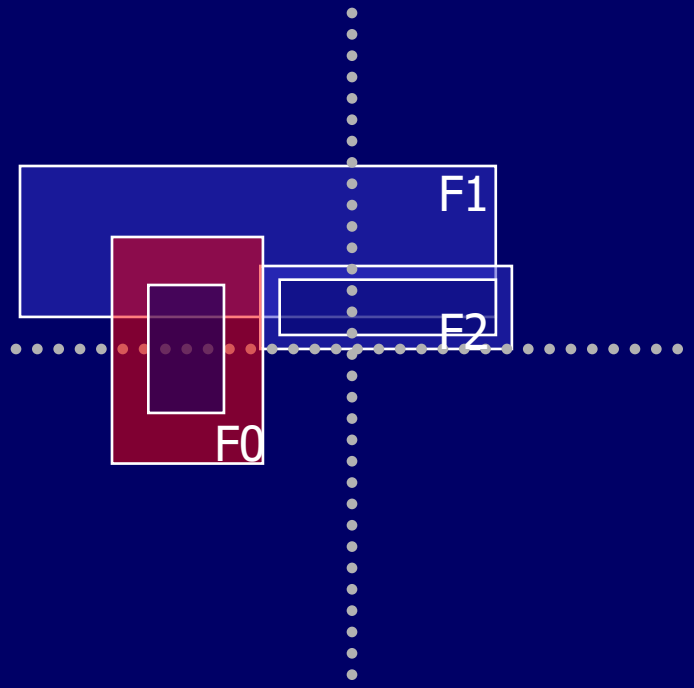
Population Structure

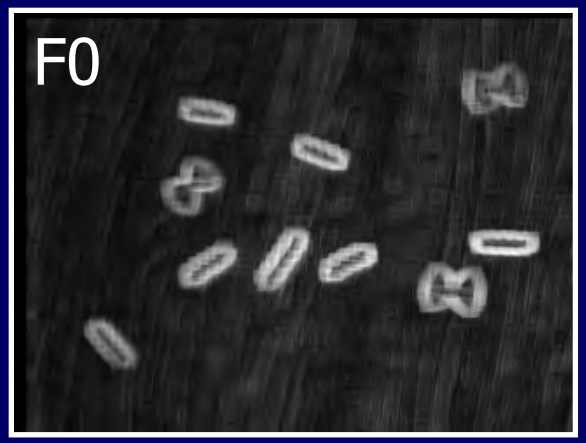


Population Structure



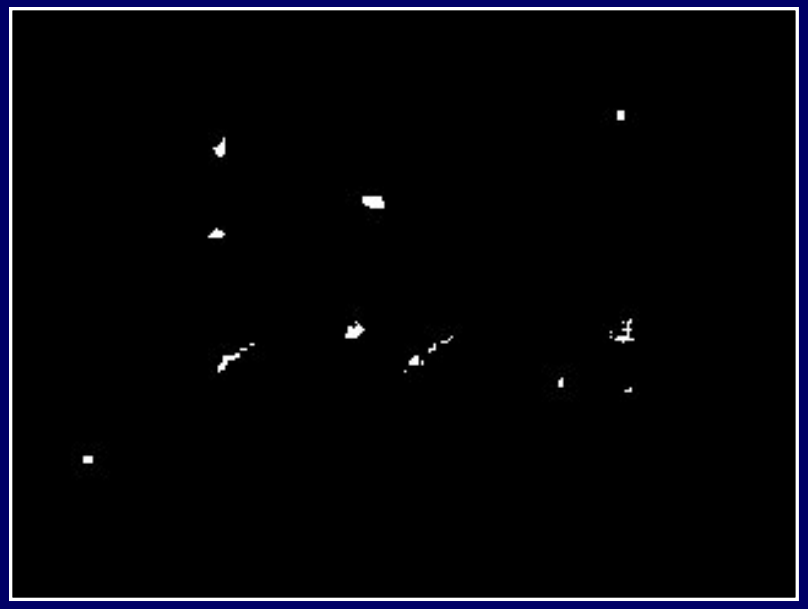
Feature Extraction



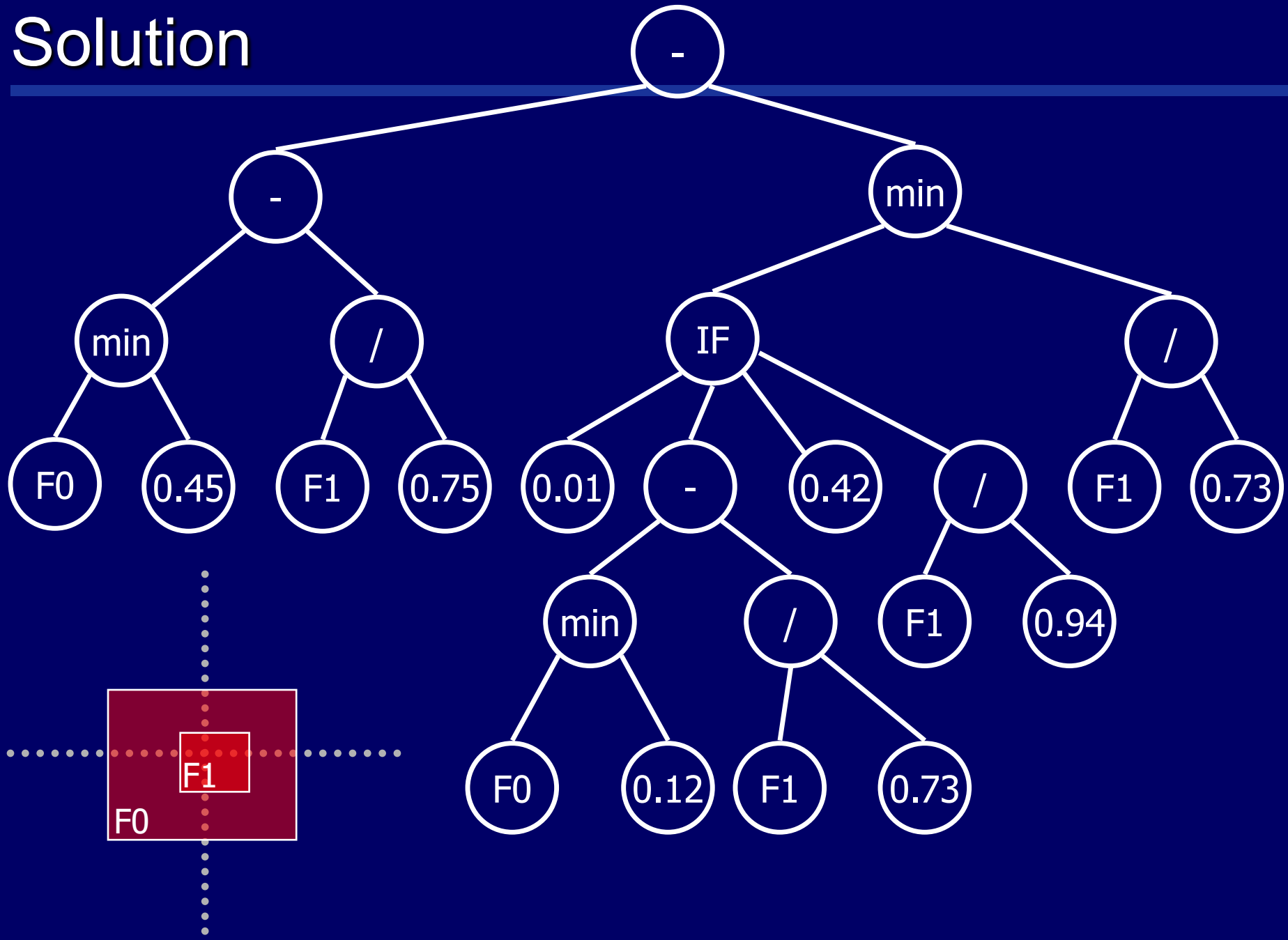


Evaluation

- Run the GP tree at every pixel, threshold at 0
- Extract the resulting regions, each one is a “guess”
  - We use a fitness function which tries to minimise the distance between a guess and a target, and make sure there is only one guess per target



Solution


-
-
min
min
/
IF
/
F0
0.45
F1
0.75
0.01
-
0.42
/
F1
0.73
min
/
F1
0.94
F0
0.12
F1
0.73
F1
F0

Results

- This approach works [1], and produces solutions which have less than 2% error on 50 unseen images, but ...
- For a typical run
  - 50 (640x480) training images, 4 feature populations of size 100, detector population of 2000, collaborator pool of size 10, 50 generations
- We have to do 153 trillion tree evaluations, and 614 trillion feature extractions!
- Roughly 2000 CPU hours

A Different Approach

- That's too much computation. Doesn't scale well either
- We can use a sampling approach
  - Create a set of all of the target pixels from each image and a random(ish) sample of the non-target points
  - Quickly train a basic classifier on this set
  - Run the best solution on the entire image
    - Will produce a lot of FPs, but not many relative to the size of the image
  - Train a new classifier to distinguish between the targets and these FPs

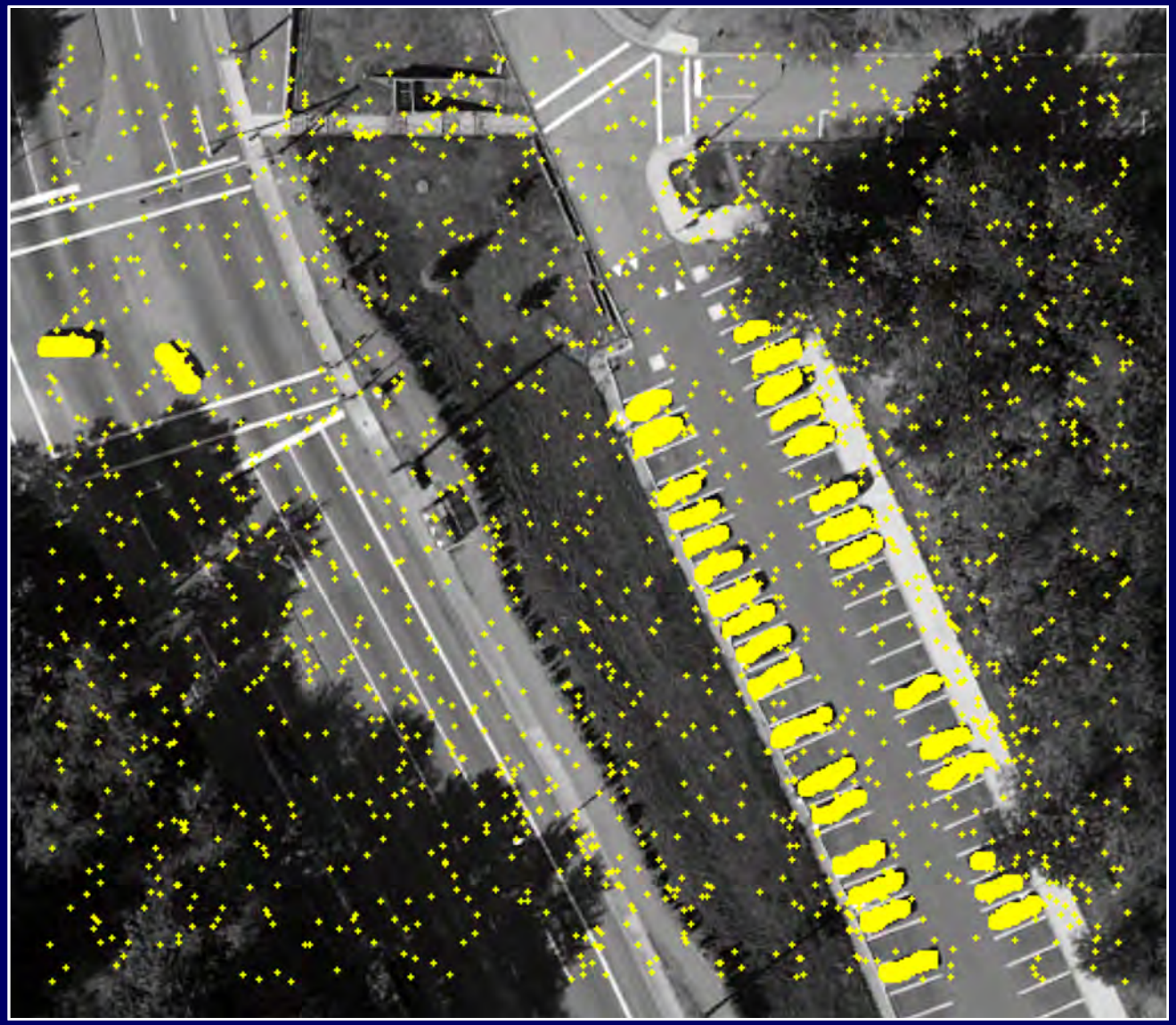
Three Example Projects

Automatic Segmentation Using Genetic Programming

Object Detection Using Coevolutionary Genetic Programming

A Multistage Approach to Object Detection Using Coevolutionary Genetic Programming

Multistage Approach

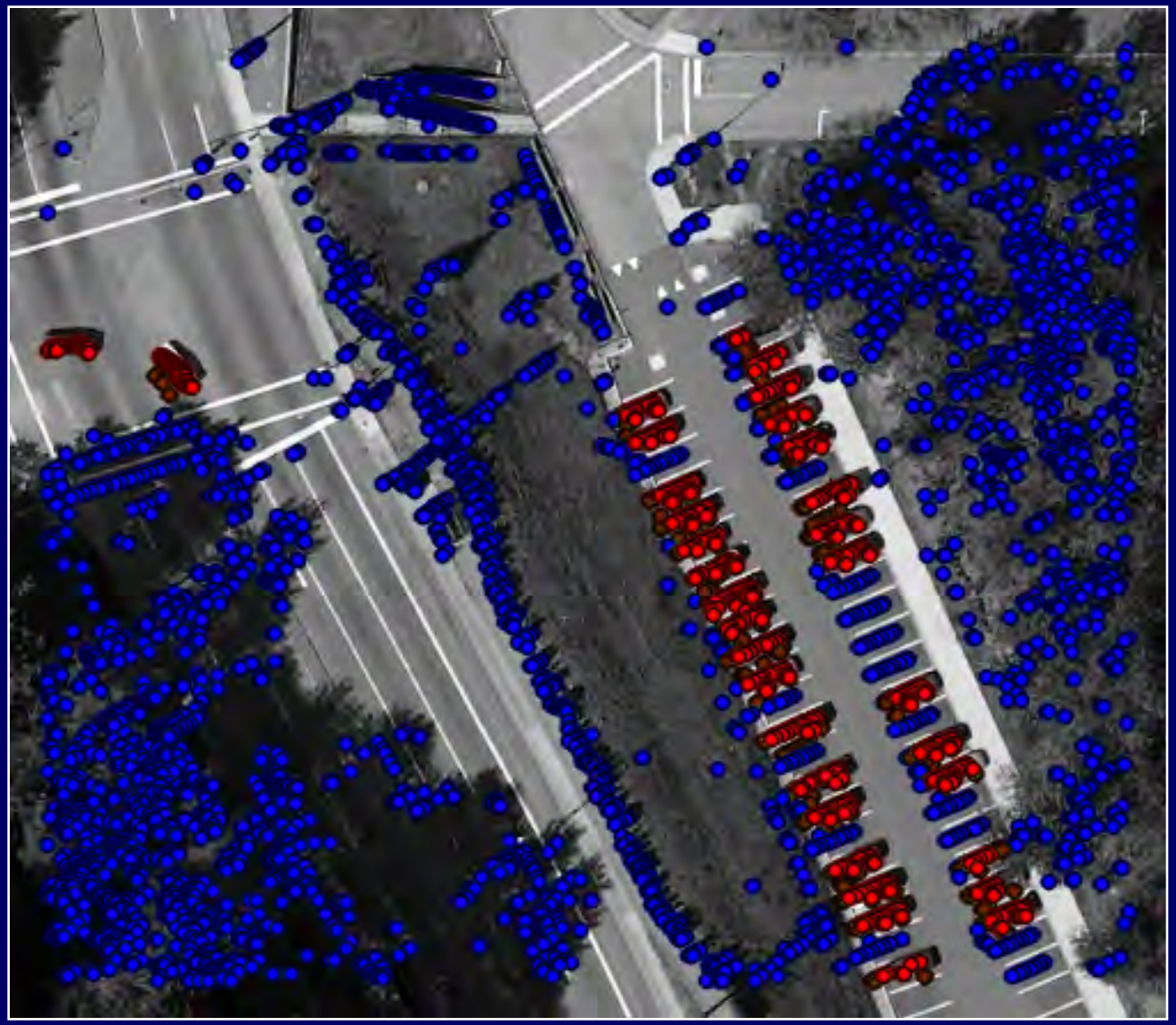


2240 target points

1770 non-target points

We train with a sample of only 4010 points instead of all 221,760

Multistage Approach



- After training we run the solution on every pixel
- Hits most targets, produces a lot of FPs
- We now have a new training set
- 220 target points, 3974 non-target points

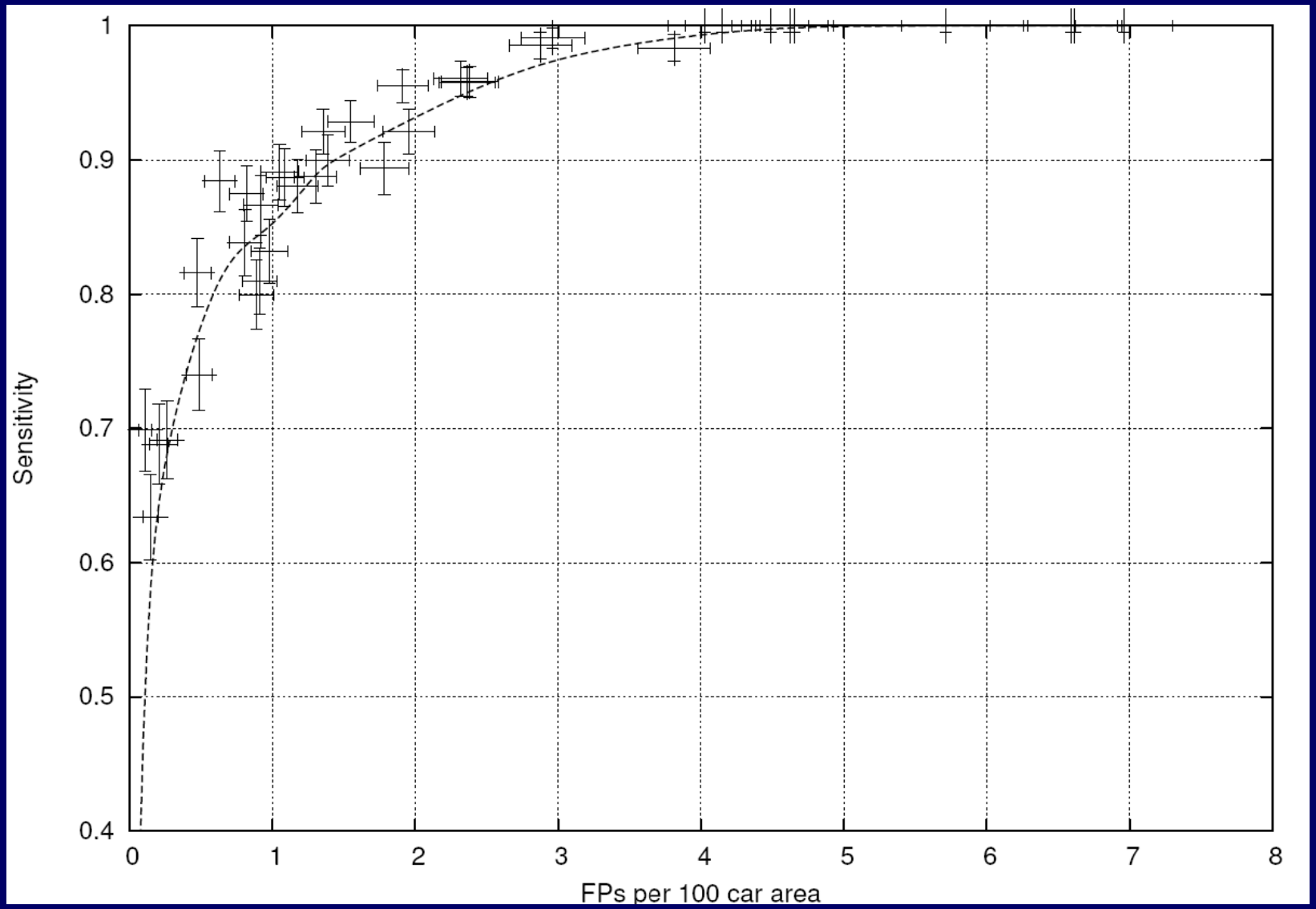
Multistage Approach

- We train again on this new set
- Anything that is hit twice is a “final guess”
- Run several of these second stages
- After 3 second stages we can spot most targets with a relatively low number of FPs

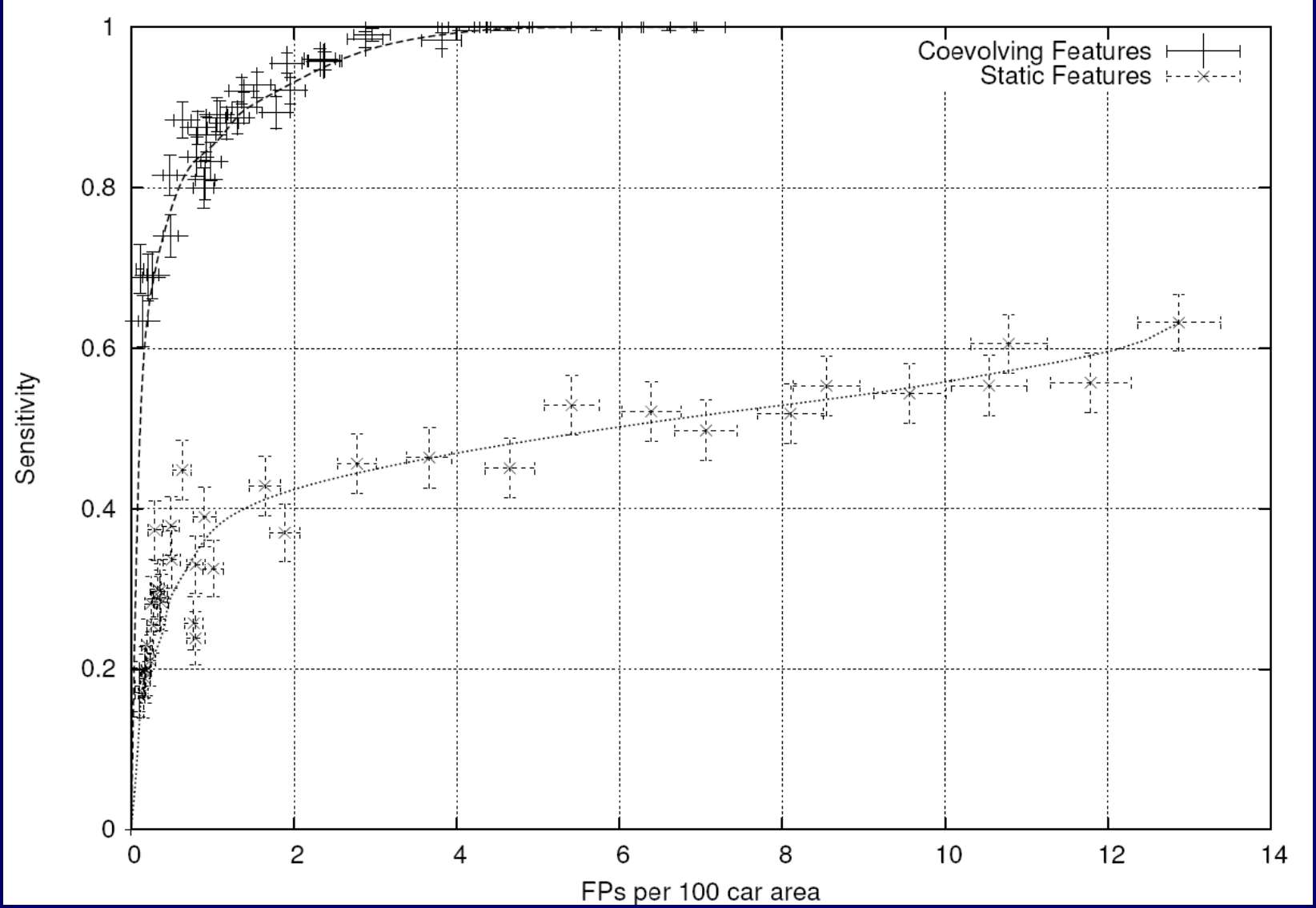
Performance

- Training set of 15 images (220 cars in total)
- Test set of 9 images (108 cars)
- Solution has 4 stages
- Each uses 3 features and a tree of size < 30
- Result analysis using FROC

Performance



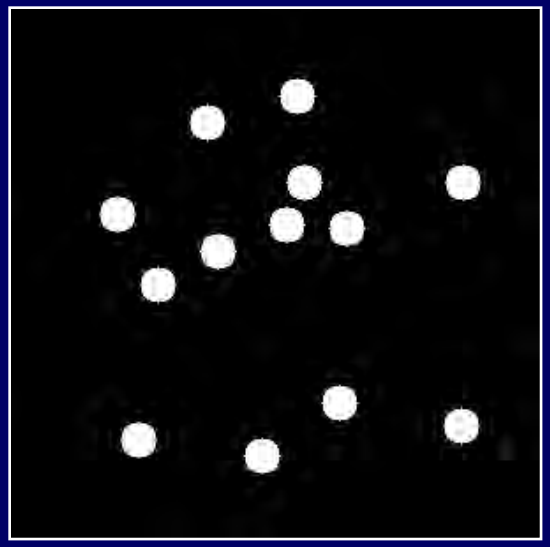
Performance

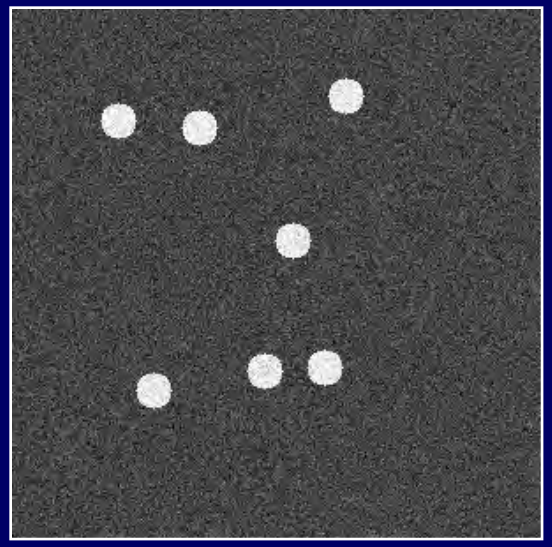


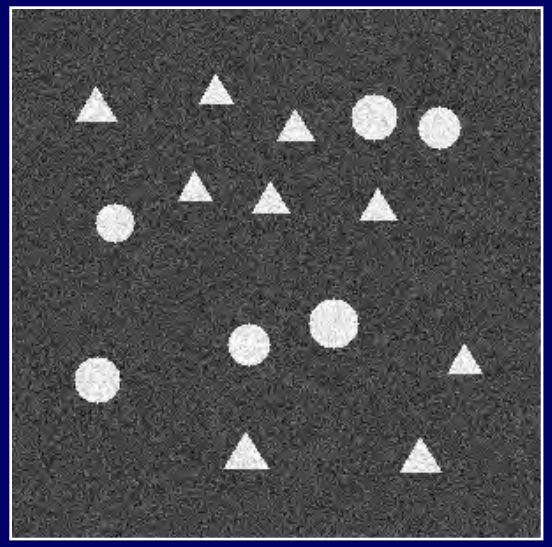
Multistage Approach

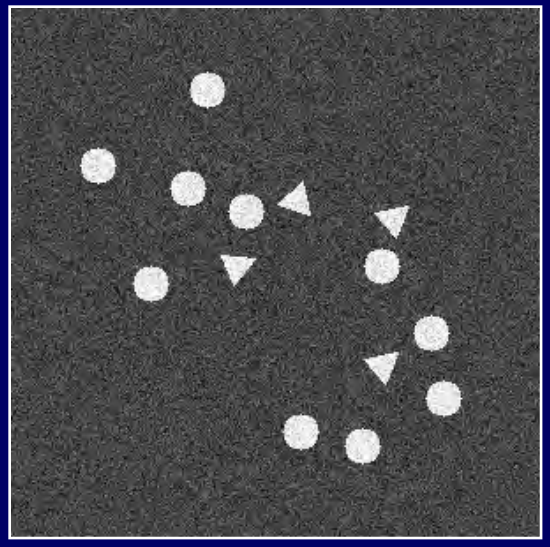
- This approach works well [2]
- Uses significantly less computation
  - About 1000 times less!
- Training time is a only few hours
  - Allows more experimentation with coevolutionary feature extraction
- Produces similar results to previous approach
- Run-time of solutions around 0.5 seconds. Would be real-time if compiled

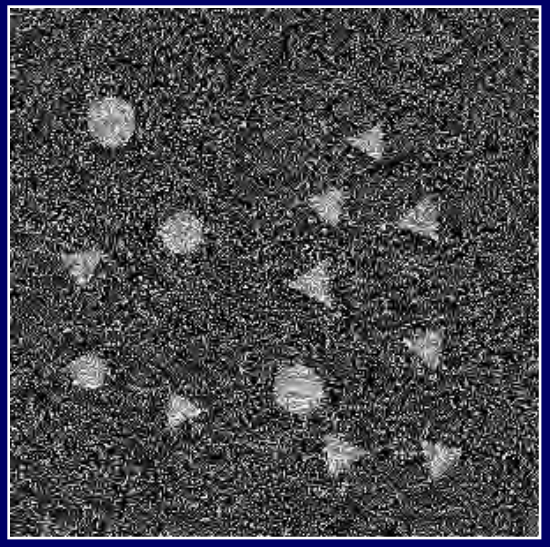
Example Datasets



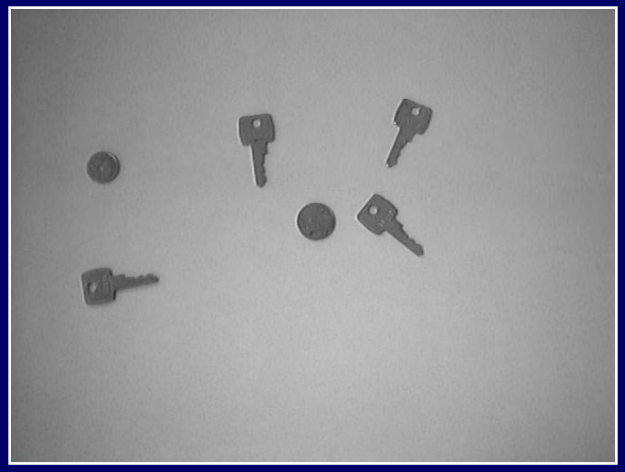








Example Datasets



Object Detection Summary

- Cooperative coevolution can be used to train visual object detectors without ANY domain knowledge
- The synergy between the stages allows us to evolve solutions which
  - Use less features – less computation
  - Use smaller classifiers
- If we extract features that are suited to the data, we can use less of them

General Conclusions (1)

- The methods shown here require no domain knowledge
- This is true automatic programming
  - We just describe the behaviour we want…
  - …and the system produces an algorithm to do it (approximate it at least)
- Cooperative coevolution means we can remove yet another human-designed stage from the pipeline

General Conclusions (2)

- Increases in computational power, and accelerated graphics hardware, mean we can start to use GP for developing image analysis programs
- GP does not suffer the “black box” curse of other learning systems (e.g. neural nets)
  - Program’s can be analysed, scrutinised, optimised, etc
  - More importantly, we can learn from the evolved programs

References

1. Roberts ME, Claridge E, *Cooperative Coevolution of Image Feature Construction and Object Detection* In et al, XY, ed: Parallel Problem Solving from Nature - PPSN VIII Volume 3242 of LNCS, Birmingham, UK, Springer-Verlag (2004) 899–908

2. Roberts ME, Claridge E, *A Multistage Approach To Cooperatively Coevolving Feature Construction and Object Detection* In Franz Rothlauf et al, eds: Proceedings of the 7th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP) LNCS, Lausanne, Switzerland, Springer-Verlag (March 2005)

3. Mark Roberts and Ela Claridge *An artificailly evolved vision system for segmenting skin lesion images* In Randy E Ellis and Terry M Peters, editors, Proceedings of the 6th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), volume 2878 of LNCS, pages 655 662, Montreal, Canada, 15-19 2003 Springer-Verlag