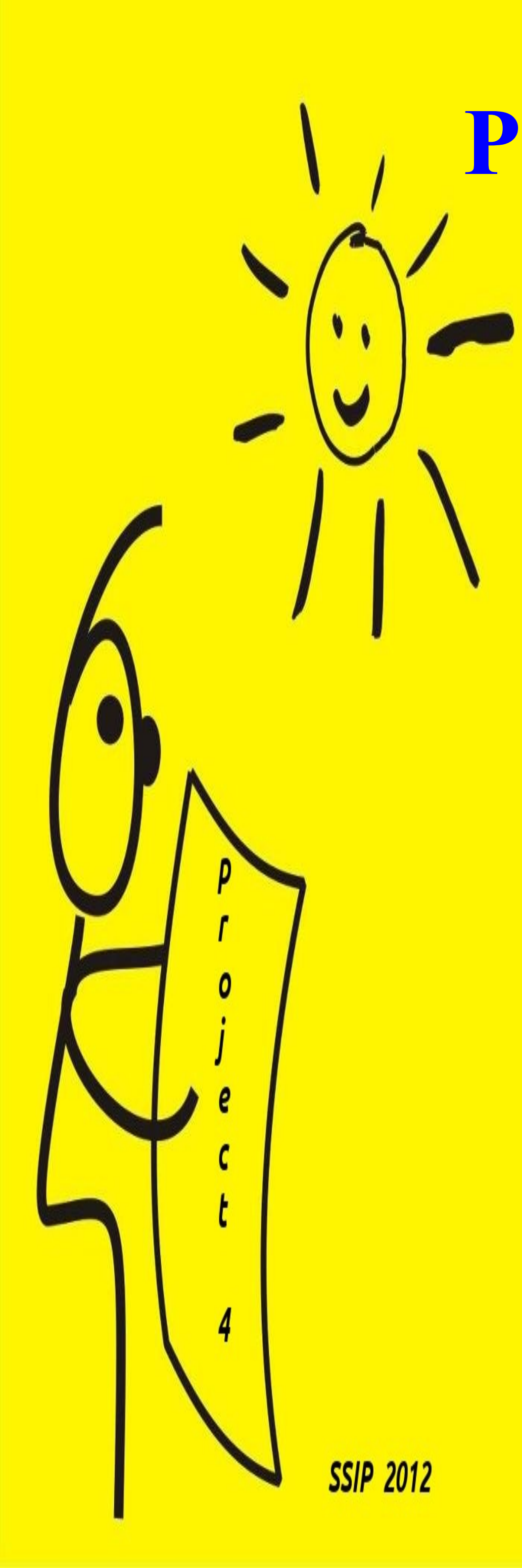


Particle Filter Segmentation

Vienna, Austria, July 2012

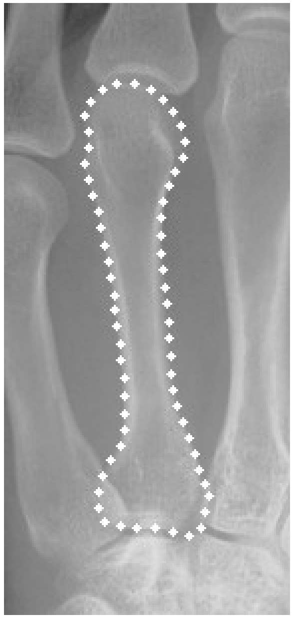
Zoltán Bárdosi

Donatella Granata

Gabor Lugos

Ahmad Pahlavan Tafti

Sanjay Saxena

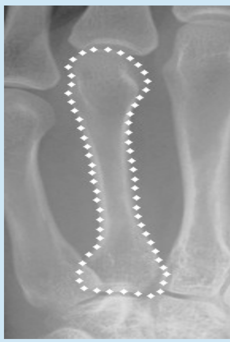


Outline

- Introduction
- Model
  - Generative Shape model
- Training
  - Random forest
- Fitting
  - Cost Function
- Results

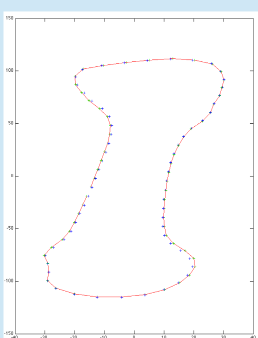
Introduction

Shape model



Annotated Shape Contour

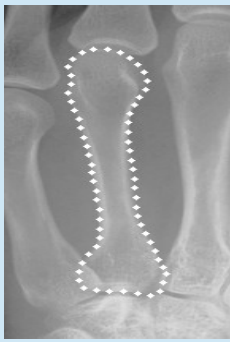
PCA



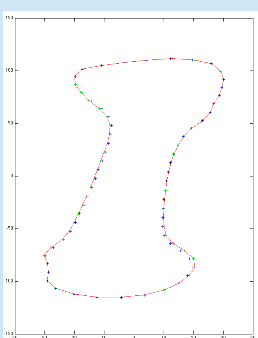
Generative shape model

Introduction

S h a p e m o d e l

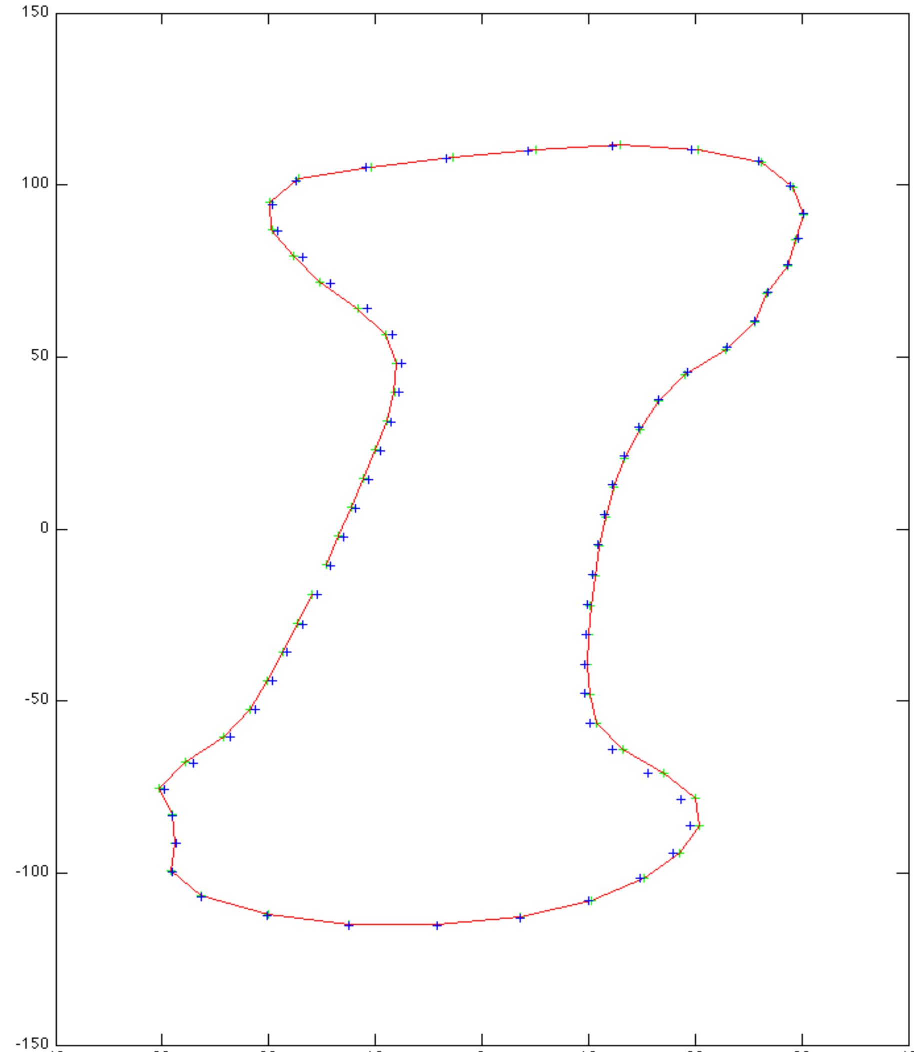


Annotated
Shape Contour

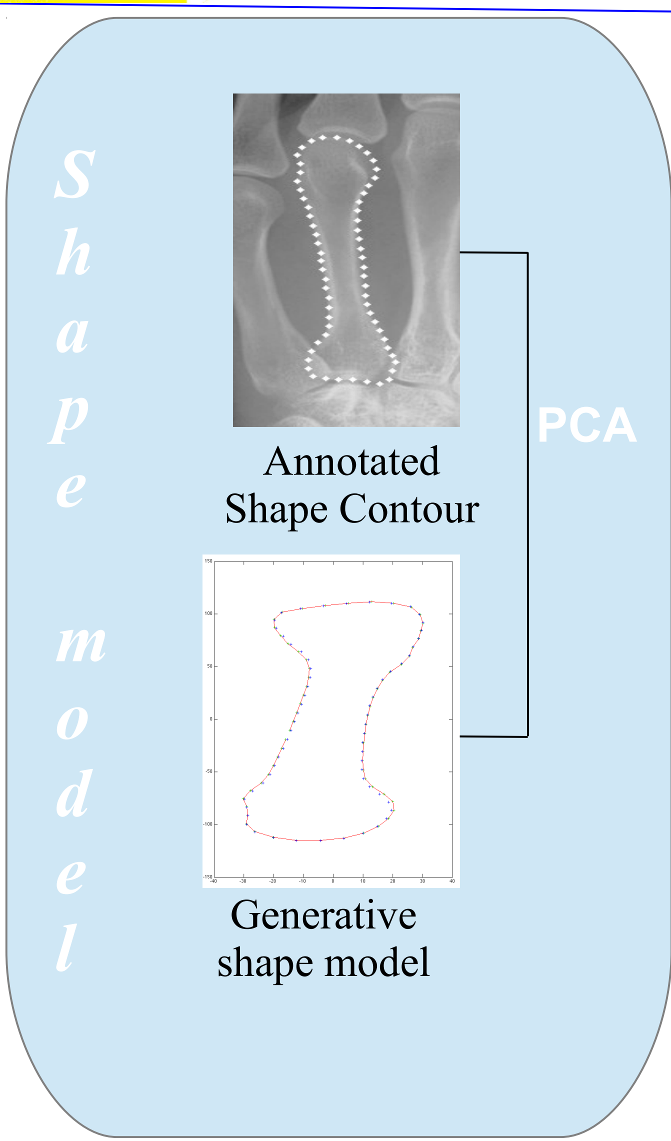


Generative
shape model

PCA

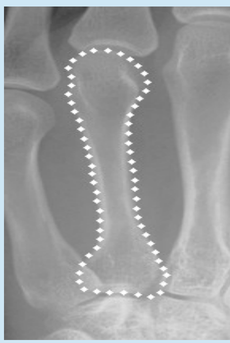


Introduction

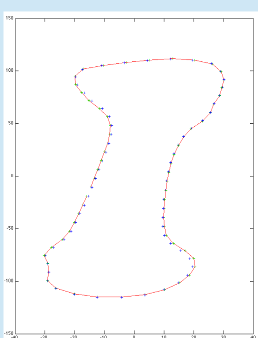


Introduction

Shape model



Annotated Shape Contour

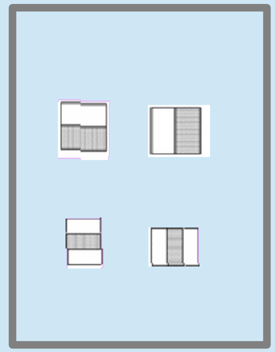


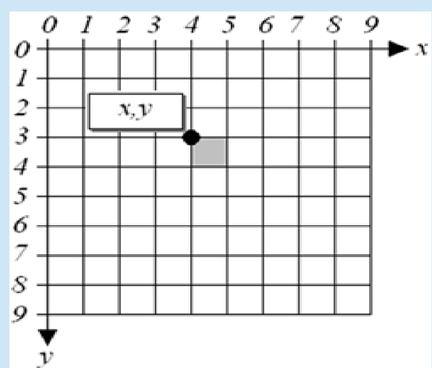
Generative shape model

PCA

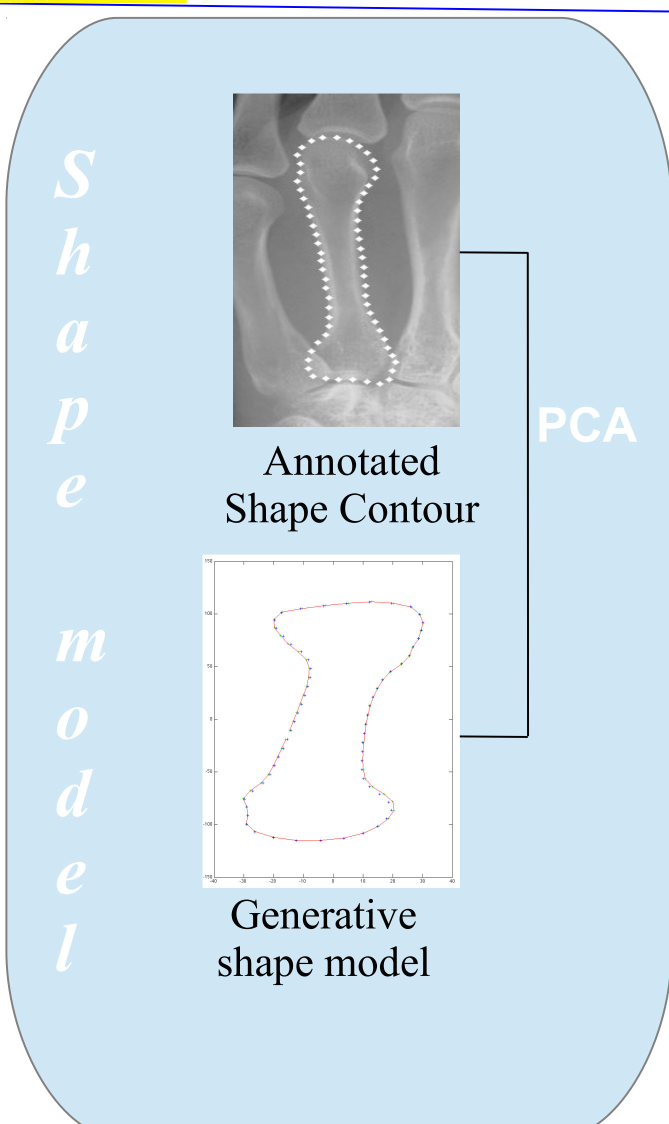
Boundary features

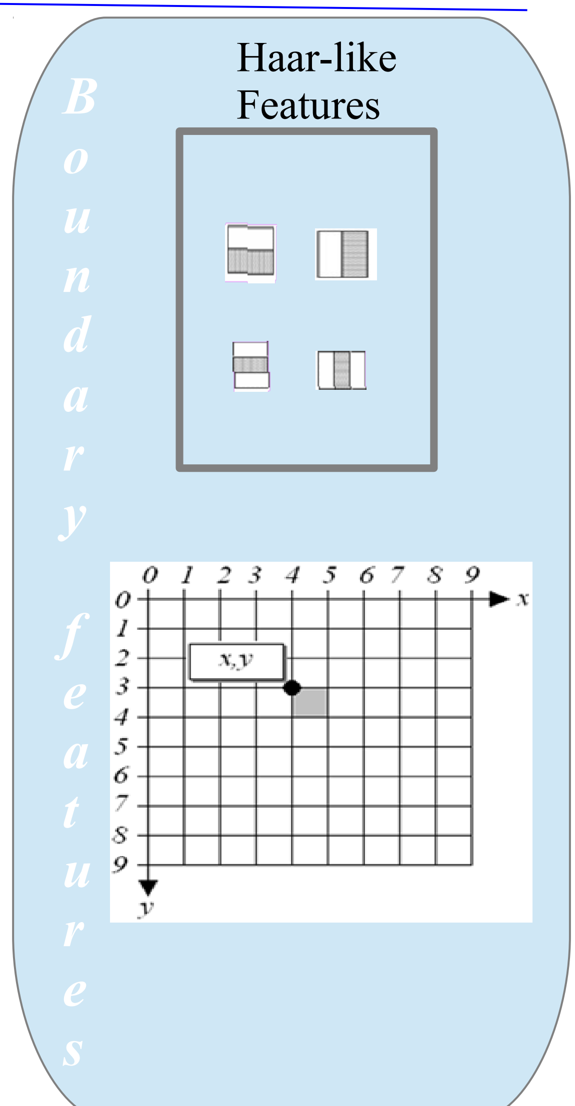
Haar-like Features

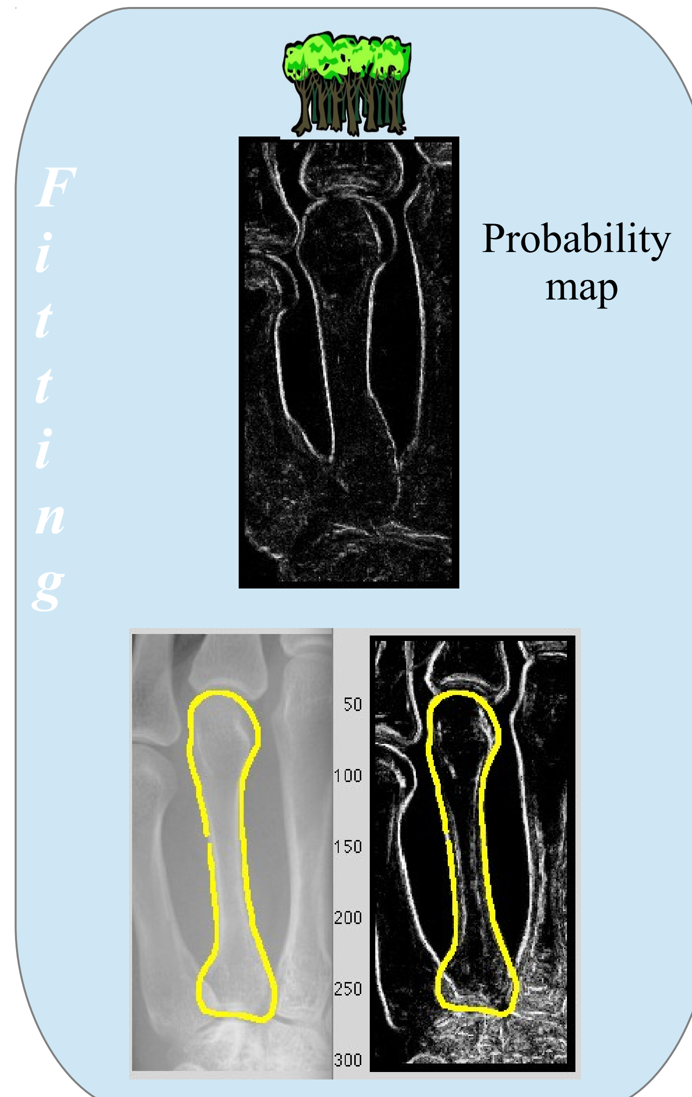




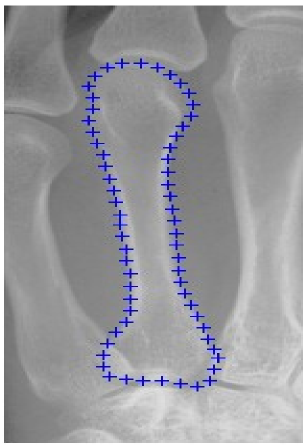
Introduction







Shape Model



Landmarks are derived from this manually annotated image contours.

$\forall\, i \in \{1, \ldots, N_s\}$ annotated shapes

for d-dimensional landmarks

$$vi = \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1d} \\ l_{21} & & & l_{2n} \\ \vdots & & & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nd} \end{bmatrix}$$

The training set is then aligned using Procrustes Analysis, which minimizes

$\sum \left(v_i - \bar{v}\right)^2$ where $\bar{v}$ is the mean of all vectors, i. e. the mean of the shape

Principal Component Analysis

$$\begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_m \end{bmatrix} = \overbrace{\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{nd} \\ A_{21} & & & A_{nd} \\ \vdots & & & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{m(nd)} \end{bmatrix}}^{\text{A}}$$

m=number of samples (i.e 50)

nd=nd denotes the number of landmarks
(i.e. n=64 and d=2)

$\forall\, i \in \{1, \ldots nd\}$ normalized data set (with zero mean)

$A_i = A_i - \bar{A}_i$ where $\bar{A}_i$ is the mean of each dimension or column

Principal Component Analysis

$$\begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_m \end{bmatrix} = \overbrace{\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{nd} \\ A_{21} & & & A_{nd} \\ \vdots & & & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{m(nd)} \end{bmatrix}}^{\text{A}}$$

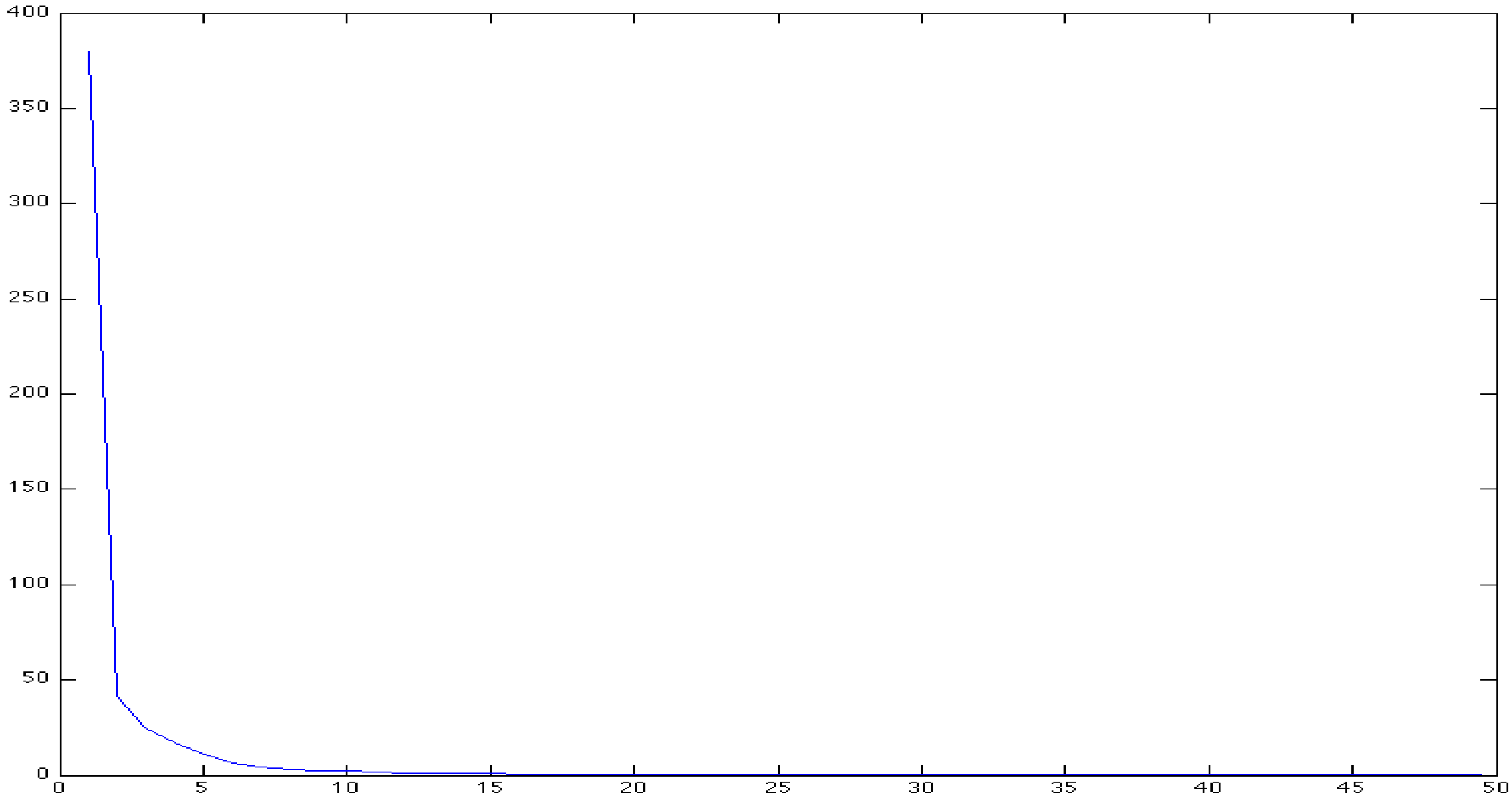
Normalization is crucial for PCA
to assure that the data is centered
around the origin, otherwise
a wrong direction for the eigenvectors
is calculated.

$\forall\, i \in \{1, \ldots nd\}$ normalized data set (with zero mean)

$A_i = A_i - \bar{A}_i$ where $\bar{A}_i$ is the mean of each dimension or column

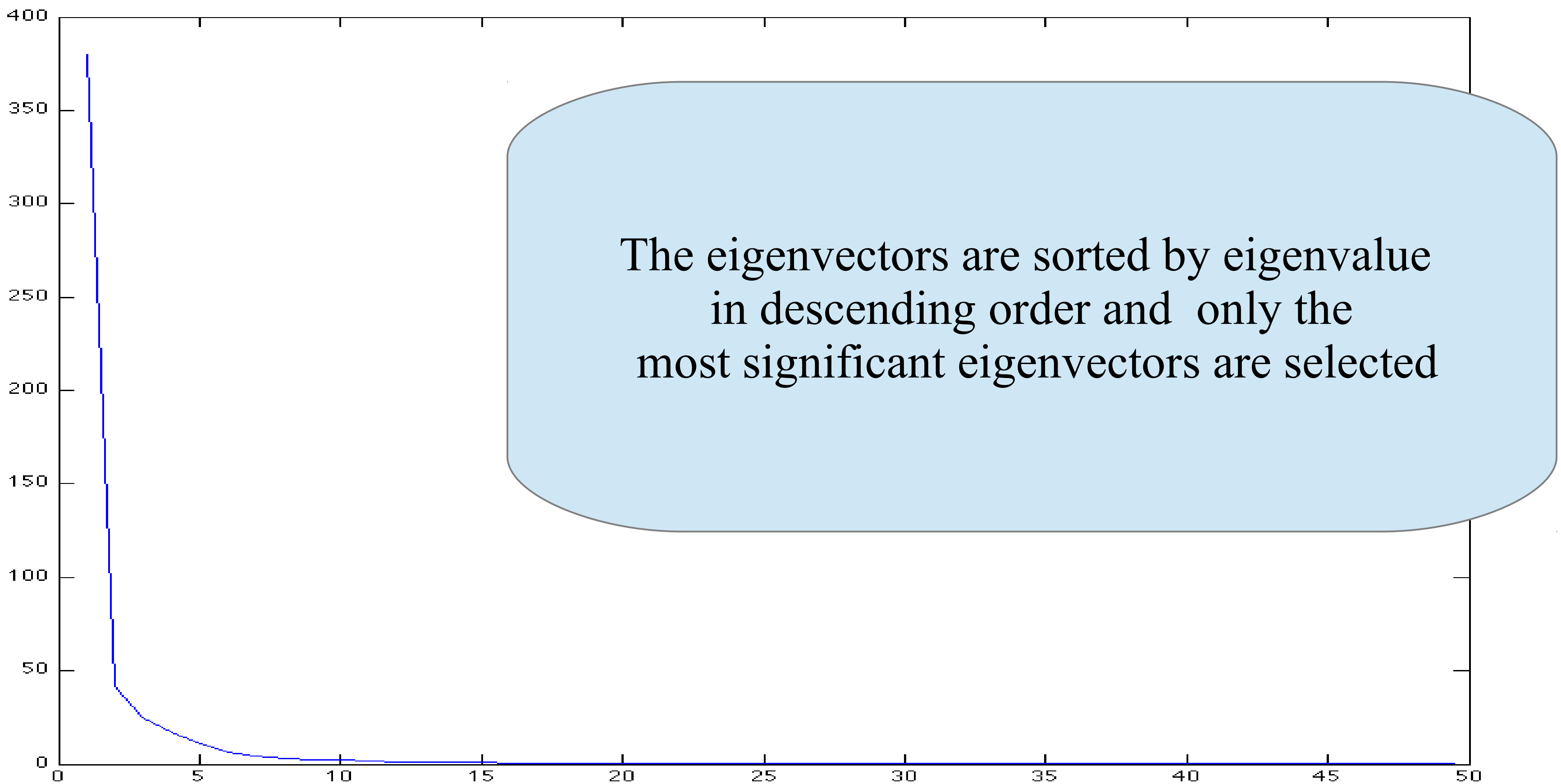
Principal Component Analysis

The eigenvector with the largest eigenvalue of the covariance matrix of the data is the data set's principle component.

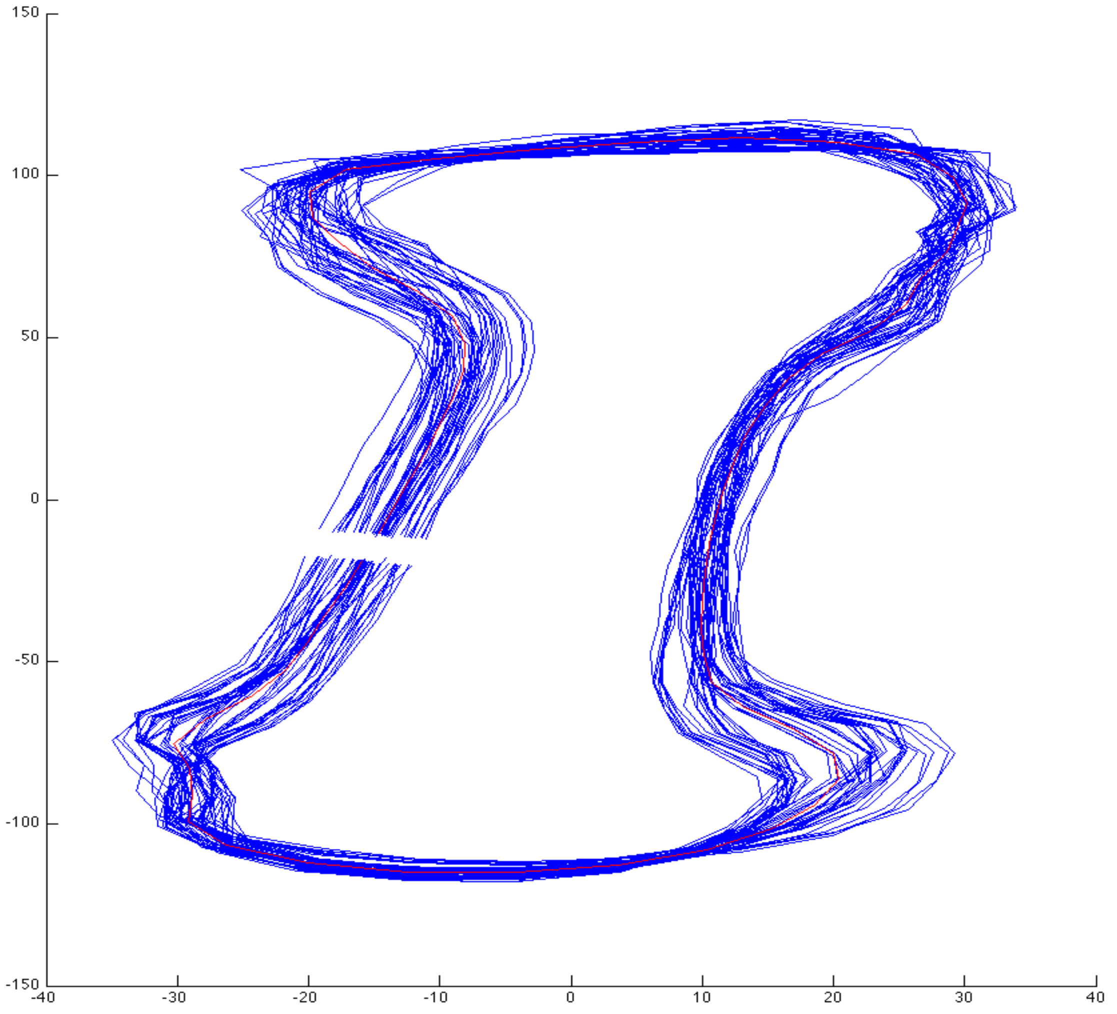


Principal Component Analysis

The eigenvector with the largest eigenvalue of the covariance matrix of the data is the data set's principle component.



Input and mean shapes



Generative Shape Model

To generate new shapes v.

The actual point distribution model is built by applying PCA.

on $V=(V_{1,}\dots,V_{N_s})$ yielding eigenvectors $e_{1,}\dots,e_{\hat{e}}$ with $e_{\hat{e}}=min(nd,N_s)$

Excluding the modes with small variance $e<\hat{e}$

Any shape v within the subspace spanned by the training set can be represented by:

$$v\approx\bar{v}+Eb$$

Generative Shape Model

To generate new shapes v.

The actual point distribution model is built by applying PCA.

on $V=(V_{1,} \ldots, V_{N_s})$ yielding eigenvectors $e_{1,} \ldots, e_{\hat{e}}$ with $e_{\hat{e}} = min(nd, N_s)$

Excluding the modes with small variance $e < \hat{e}$

Any shape v within the subspace spanned by the training set can be represented by:

$$v \approx \bar{v} + Eb$$

shape within subspace

Generative Shape Model

To generate new shapes v.

The actual point distribution model is built by applying PCA.

on $V = (V_{1}, \ldots, V_{N_s})$ yielding eigenvectors $e_{1}, \ldots, e_{\hat{e}}$ with $e_{\hat{e}} = min(nd, N_s)$

Excluding the modes with small variance $e < \hat{e}$

Any shape v within the subspace spanned by the training set can be represented by:

$$v \approx \bar{v} + Eb$$

shape within subspace

basis of the eigenspace

Generative Shape Model

To generate new shapes v.

The actual point distribution model is built by applying PCA.

on $V=(V_{1},\ldots,V_{N_s})$ yielding eigenvectors $e_{1},\ldots,e_{\hat{e}}$ with $e_{\hat{e}}=min(nd,N_s)$

Excluding the modes with small variance $e<\hat{e}$

Any shape v within the subspace spanned by the training set can be represented by:

$$v\approx\bar{v}+Eb$$

shape within subspace

basis of the eigenspace

defines the parameters for the deformable model

Generative Shape Model

To generate new shapes v.

The actual point distribution model

Each element of b controls
one mode of shape variation, the first
modes being responsible for the highest variation,
in descending order

on $V=(V_{1,}\dots,V_{N_s})$ yielding $N_s)$

Excluding the modes with small

Any shape v within the subspace spanne training set can be represented by:

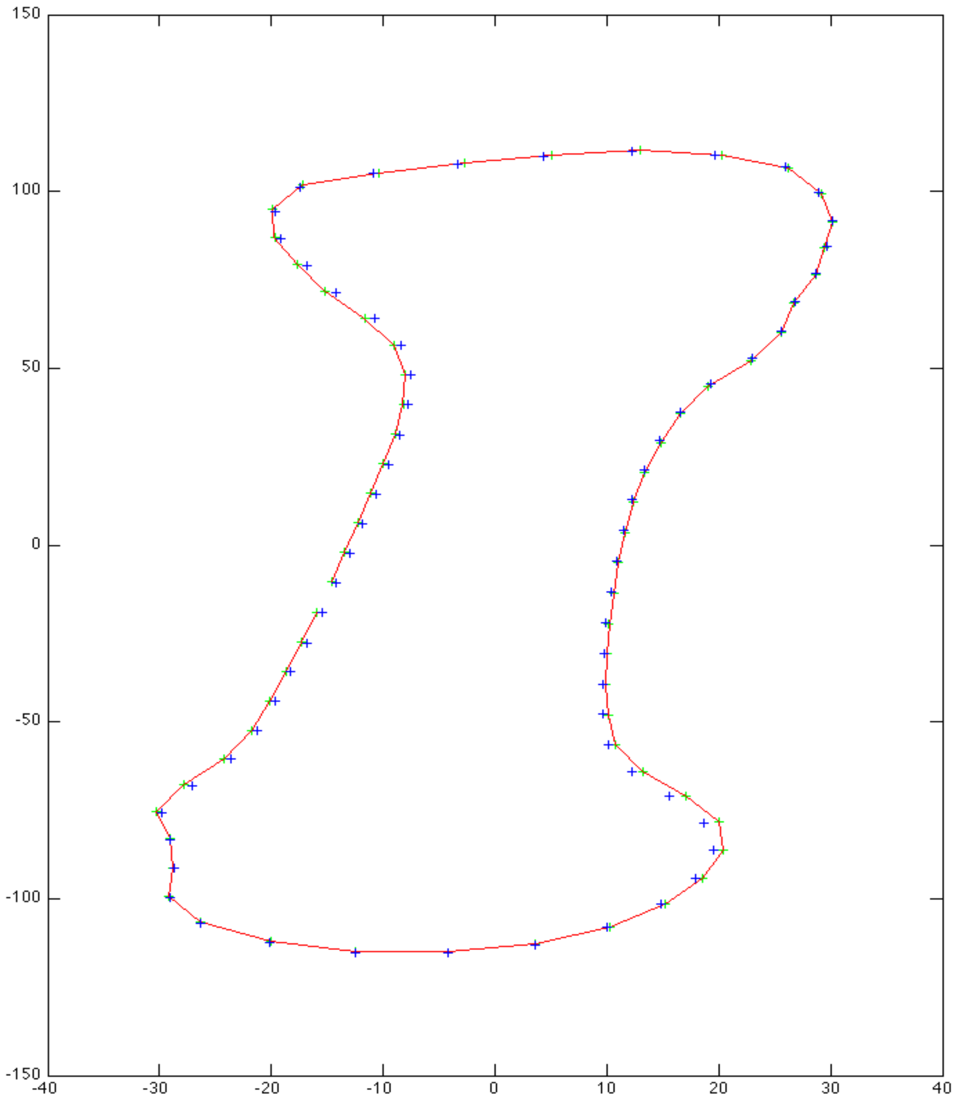
$$v \approx \bar{v} + Eb$$

shape within subspace

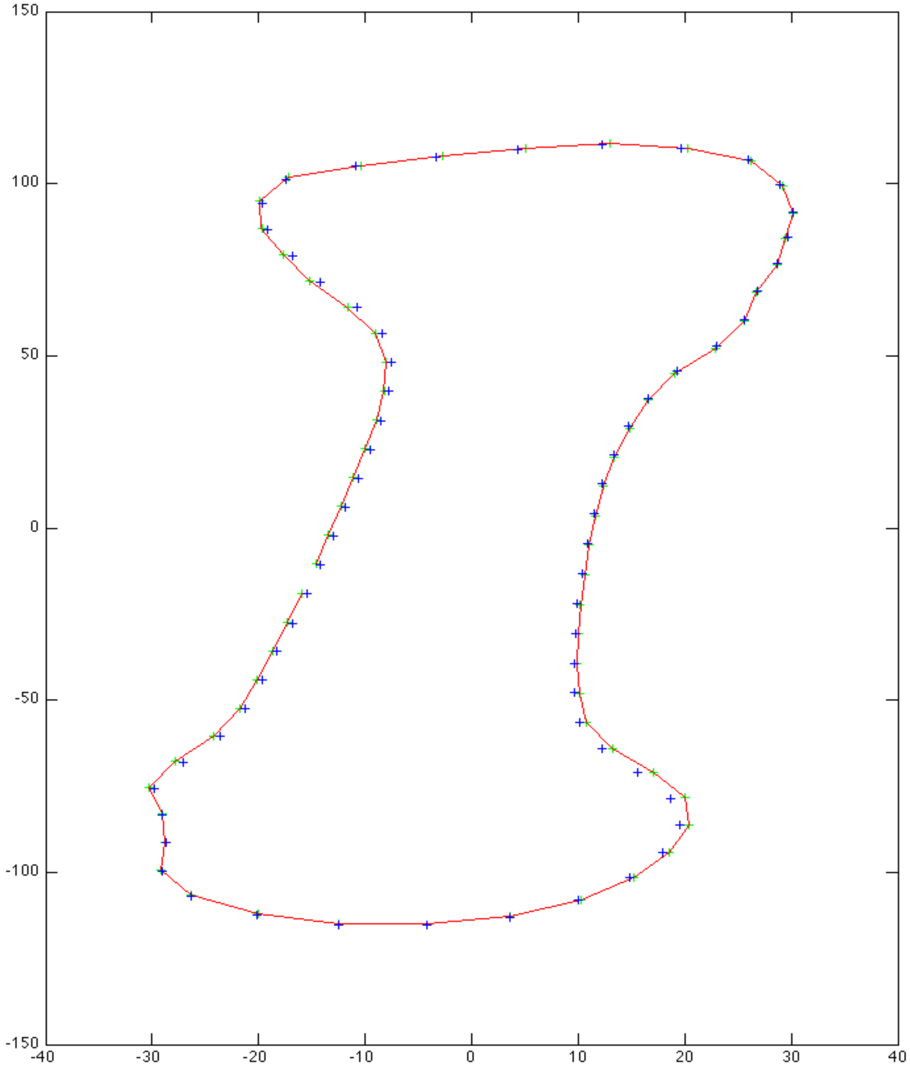
basis of the eigenspace

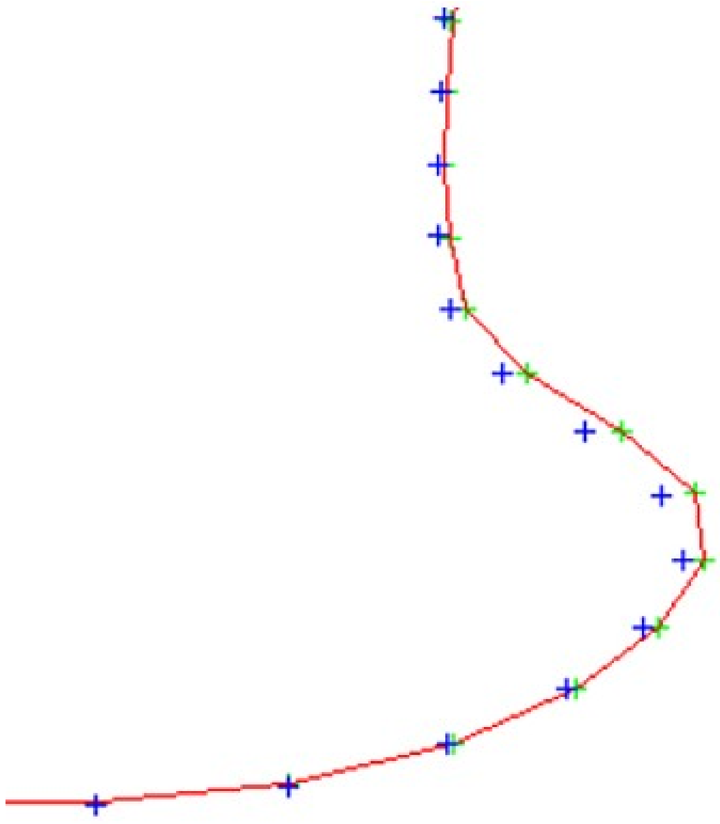
defines the parameters
for the deformable model

New shape example

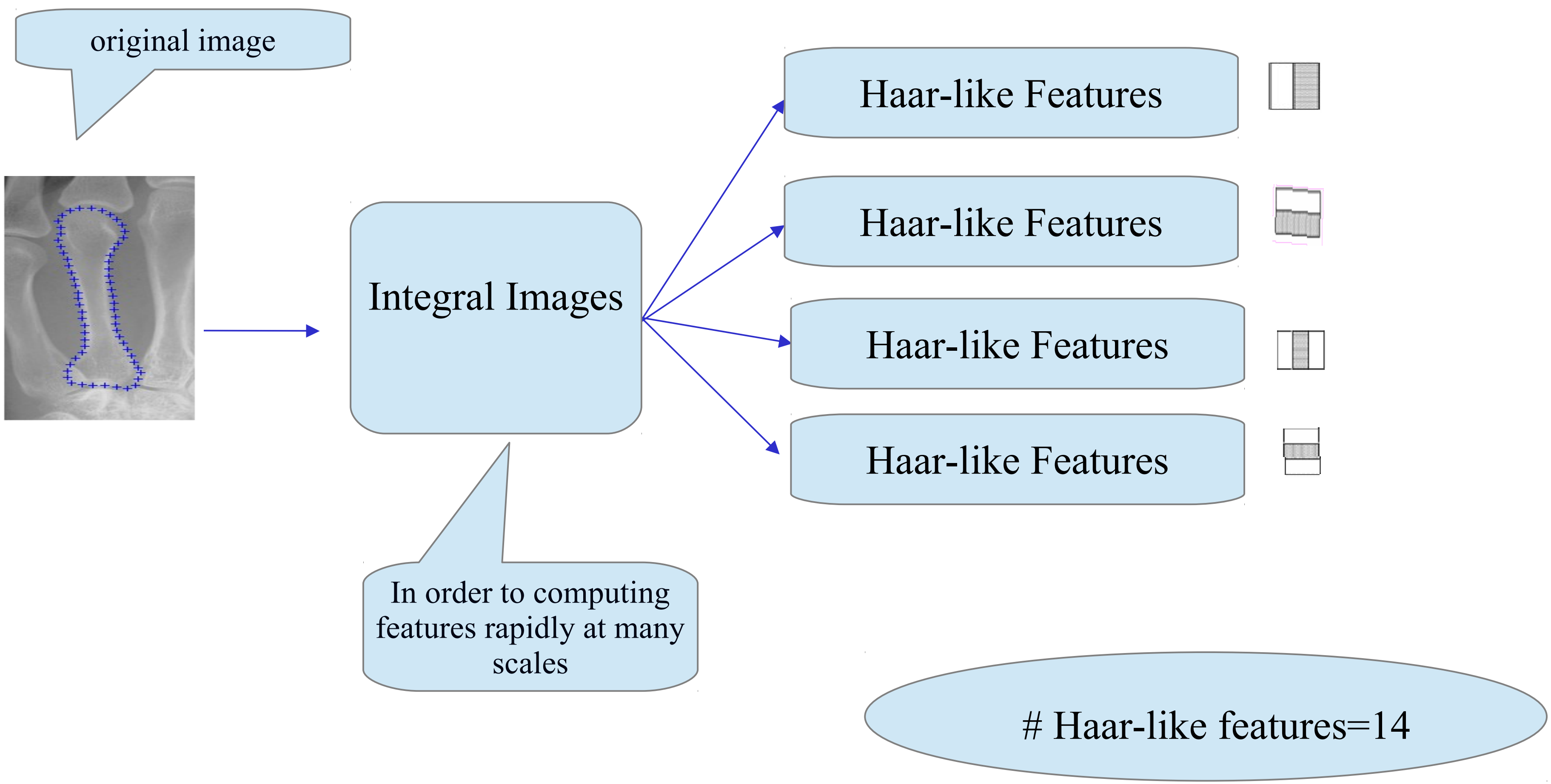


New shape example



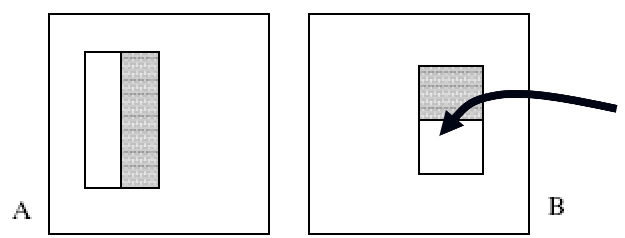


Haar-like features



Haar-like features: two rectangle

The difference between the sum of pixels within two rectangular regions



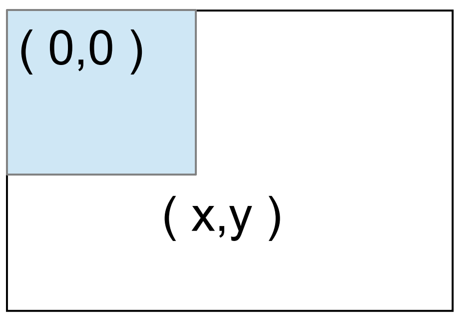
The region have the same size and shape
And are horizontally or vertically adjacent

Integral Image

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y'),$$

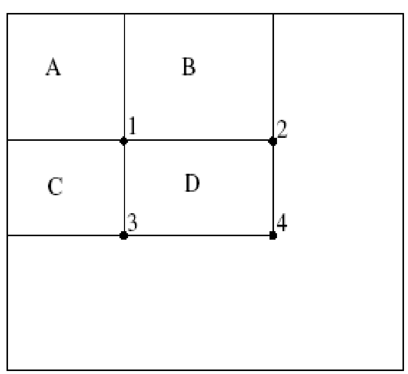
original image

integral image



$$\begin{aligned} s(x,y) &= s(x,y-1) + i(x,y) \\ ii(x,y) &= ii(x-1,y) + s(x,y) \end{aligned}$$

Calculating any rectangle sum with integral image



1 ➡ A
2 ➡ A + B
3 ➡ A + C
4 ➡ A + B + C + D

Rectangle Sum
D = 4 - 3 - 2 + 1

Random Forest Classification

- Build many decision trees
- Train each tree on a different subset of training data (bagging)
- At each tree node, split the data based on a different subset of features.
- Evaluate each tree using “out-of-bag” data (i.e., data not used to train tree).
- Aggregate results over all trees

Random Forest Classification

```
>> t= treBagger(32, X, Y);
```

predictors

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_{12} \\ P_{13} \\ P_{14} \\ \vdots \\ P_{30} \\ P_{31} \\ P_{32} \end{bmatrix}$$

```
>>[Y_pred, allpred]=predict(t,X_new);
```

response

$\overline{p}$

Cost function

2 translation coeffs

2 scaling coeffs

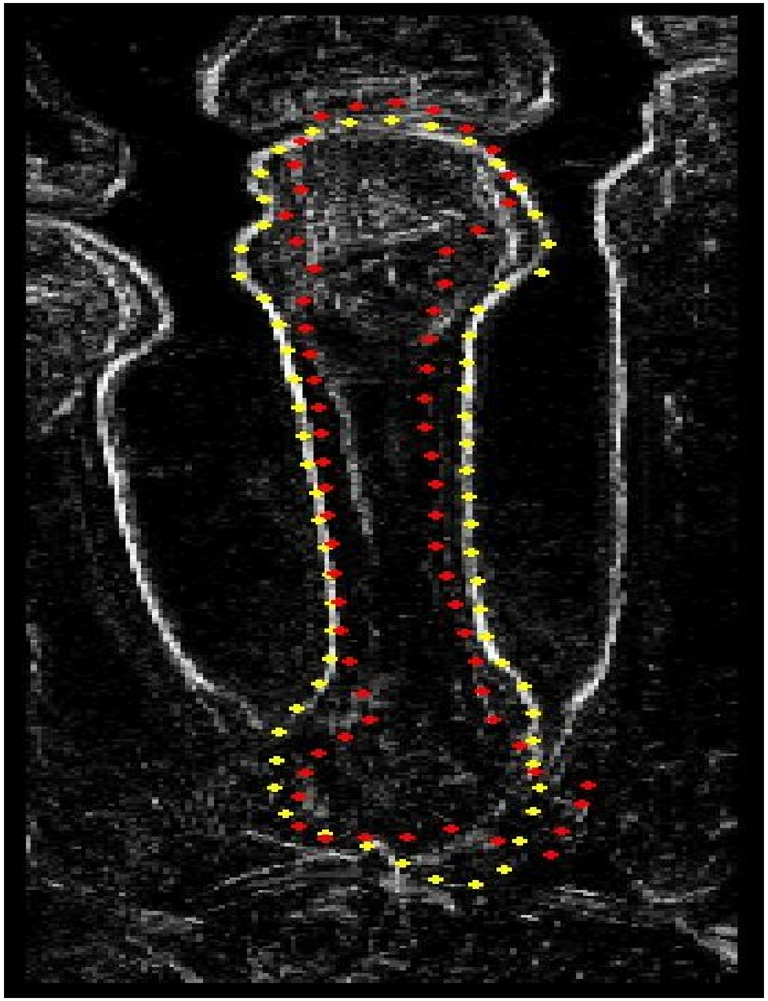
$$x = \left[b_{1,} \dots b_{8,} t_x, t_y, r, s_{1,} s_2 \right] \quad x \in R^{13}$$

8 PCA comps

1 rotation param

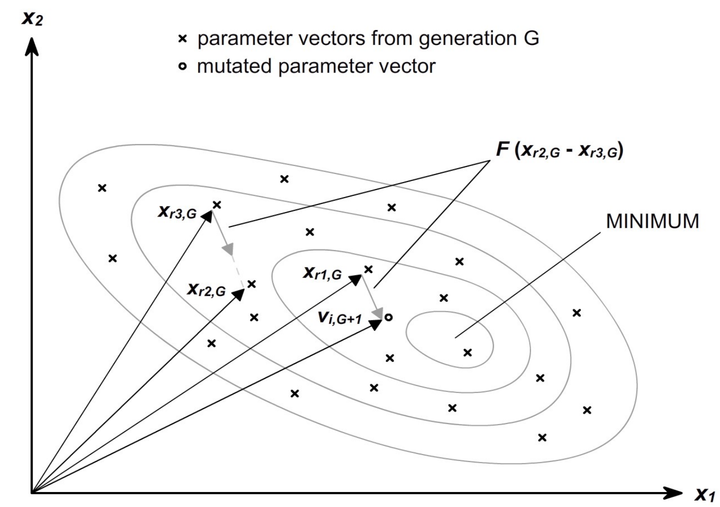
$$f(x) = 1 - \frac{\sum_{p_i \in shape(x)}^{n} p_i}{n}$$

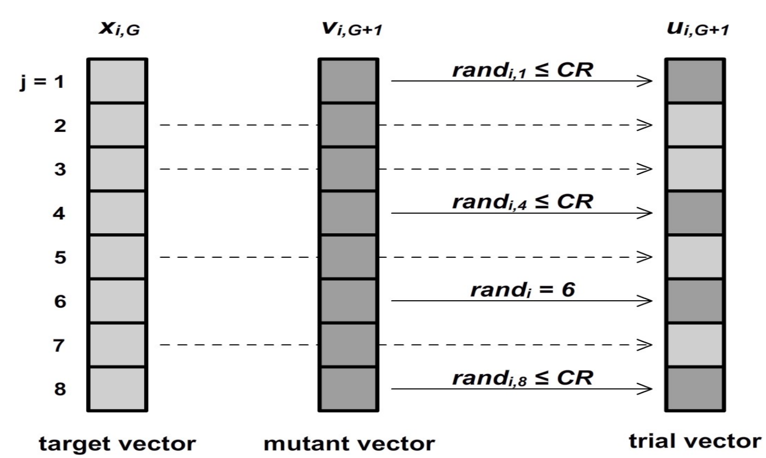
$$\min f(x)$$
$$x \in R^{13}$$



Differential Evolution

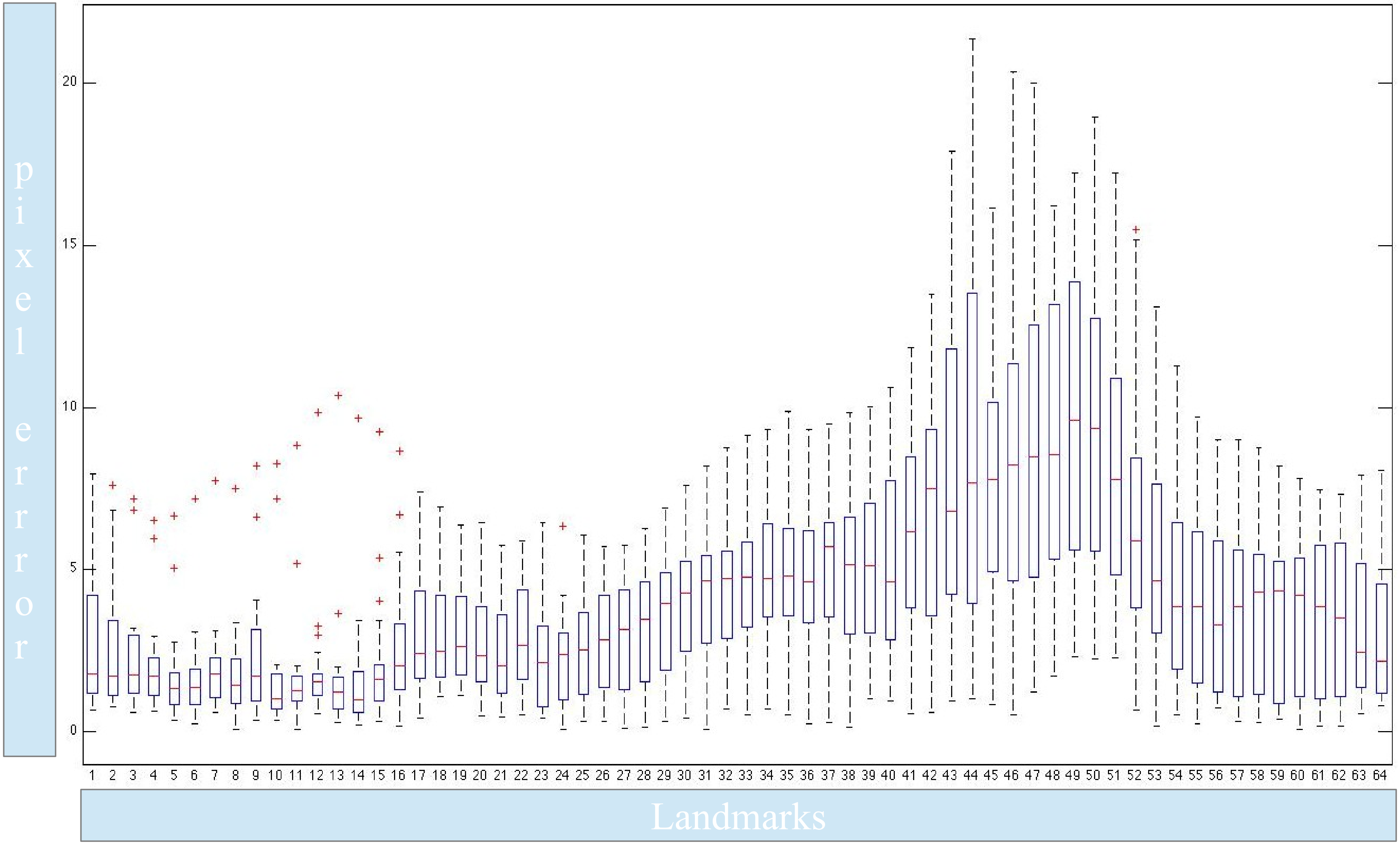
- DE is an extention of classical
  GA algorithms
- It works with a set of candidate solutions (population).
- It uses a nature inspired search operator.
  - -Mutation
  - -Crossover
  - -Selection
- It is dependent on the problem that we need to solve.



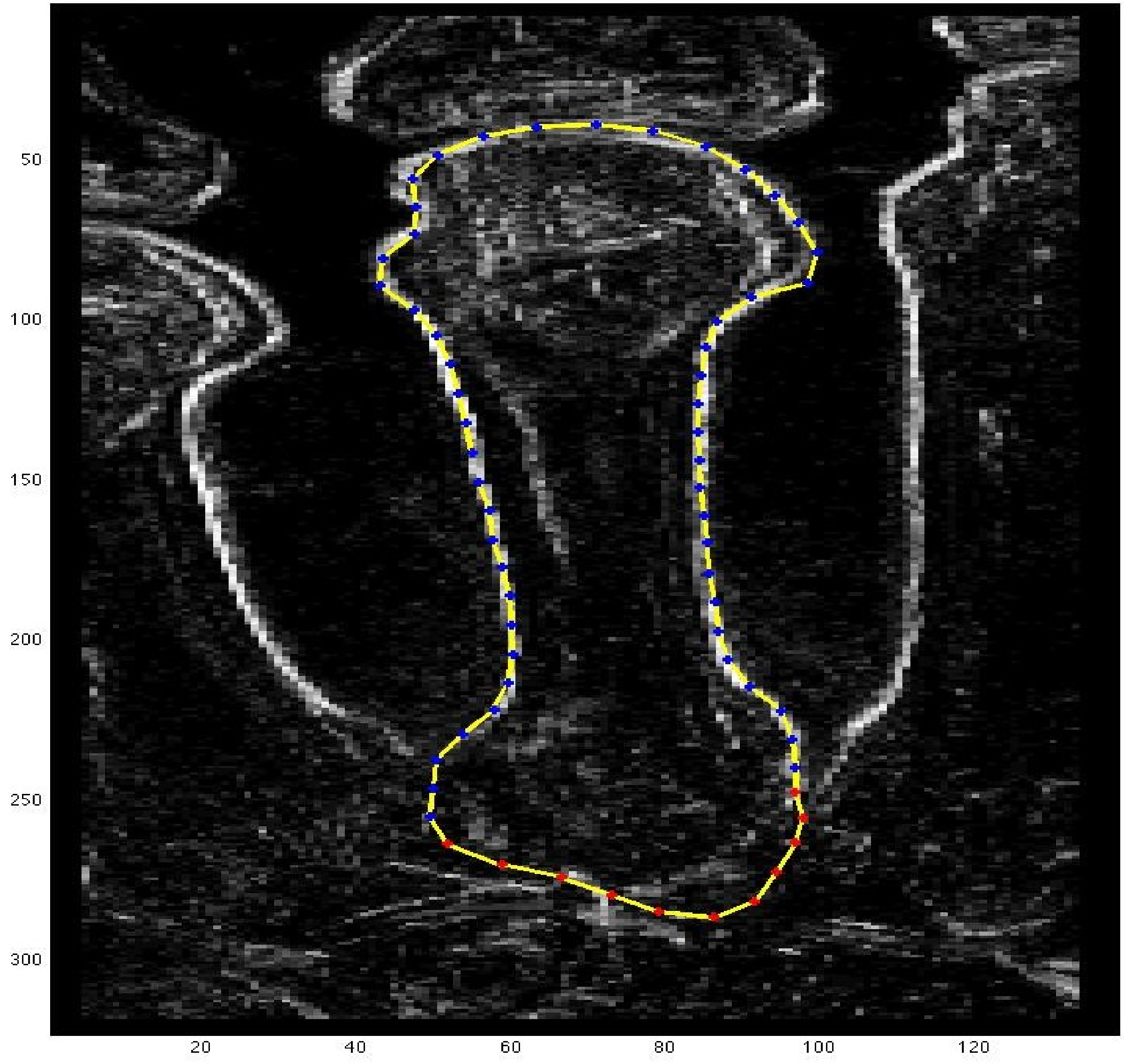


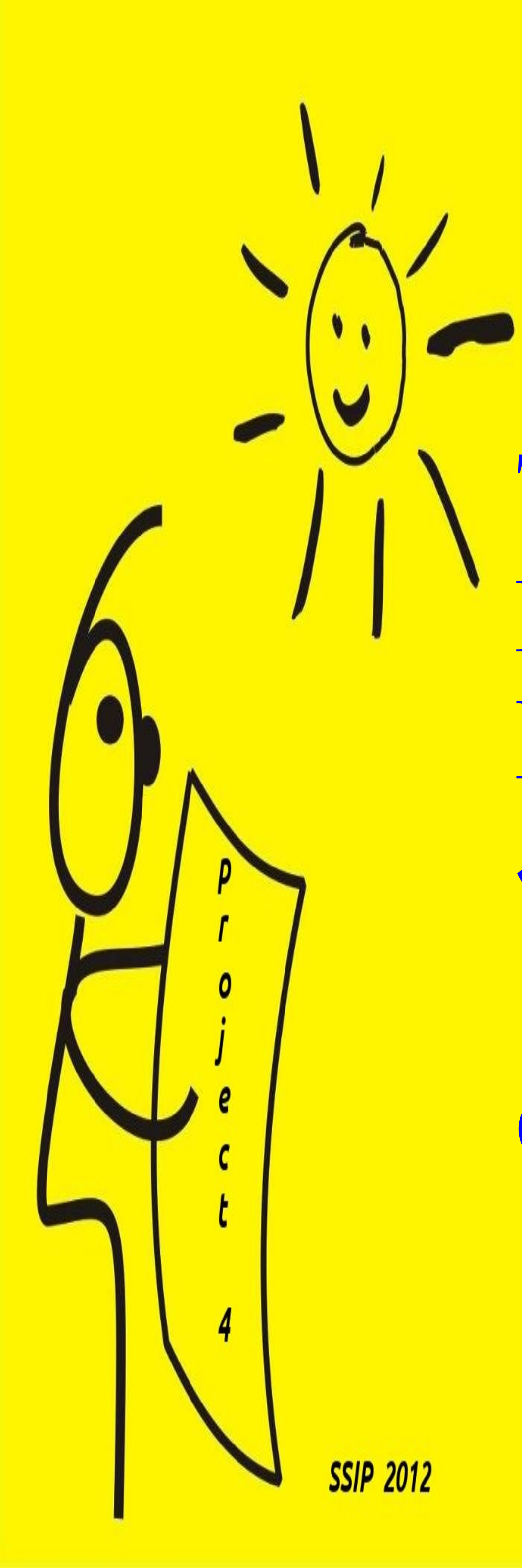
Results:

Results: Landmarks MSE



Results: Landmarks MSE





Thanks for your attention!!

Danke für Ihre Aufmerksamkeit!!

Köszönjük a figyelmet!!

आपका ध्यान के लिए धन्यवाद !!

با تشکر از توجه شما !!

Grazie per l'attenzione !!