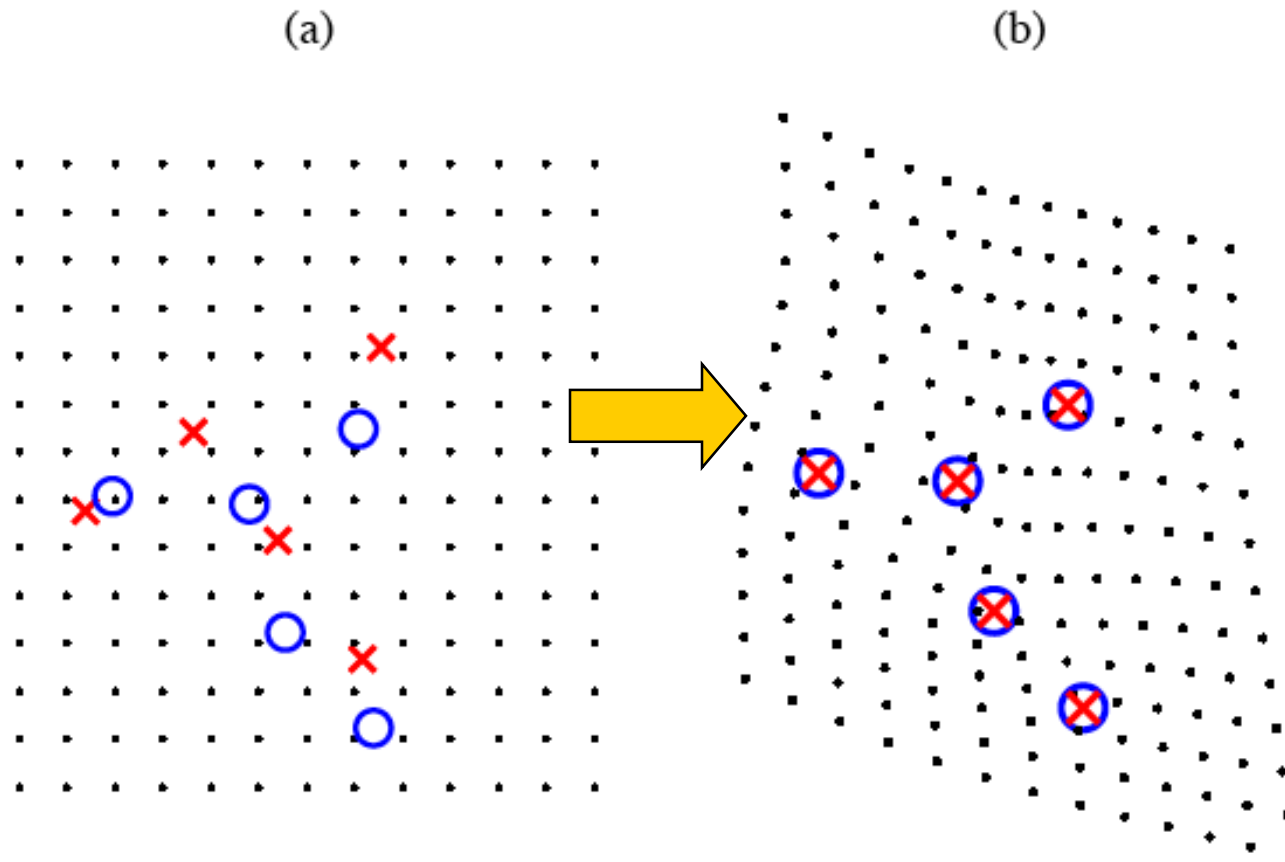
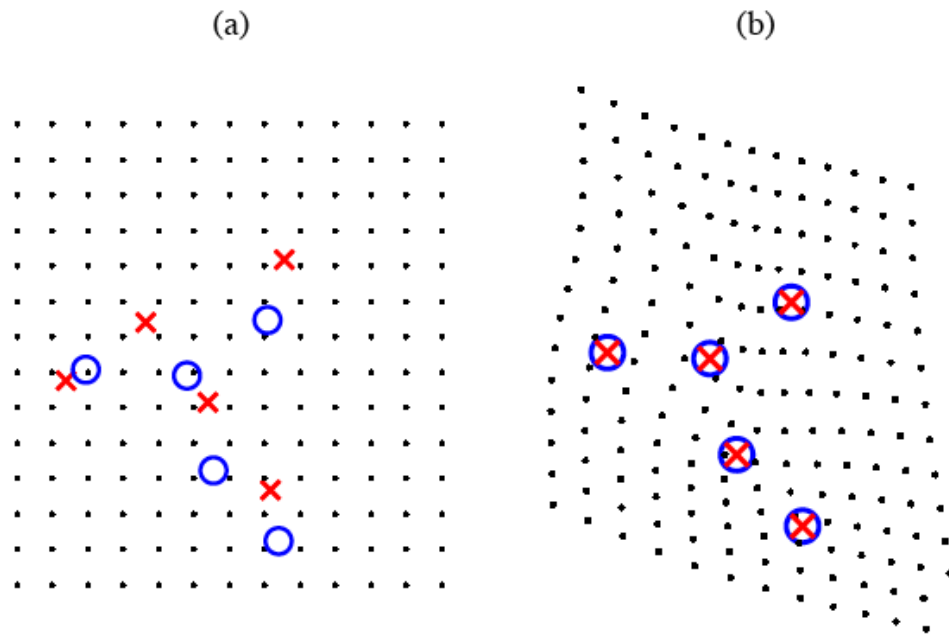


Thin-plate spline TPS



Sparse and irregular positioned feature points, and smooth interpolation

Simple example of coordinate transformation using TPS



Let's consider two sets of points for which we assume the correspondences to be known (a). The TPS warping allows a perfect alignment of the points and the bending of the grid shows the deformation needed to bring the two sets on top of each other (b). Note that in the case of TPS applied to coordinate transformation we actually use two splines, one for the displacement in the x direction and one for the displacement in the y direction. The displacement in each direction is considered as a height map for the points and a spline is fit as in the case of scattered points in 3D space. And finally the two resulting transformations are combined into a single mapping.

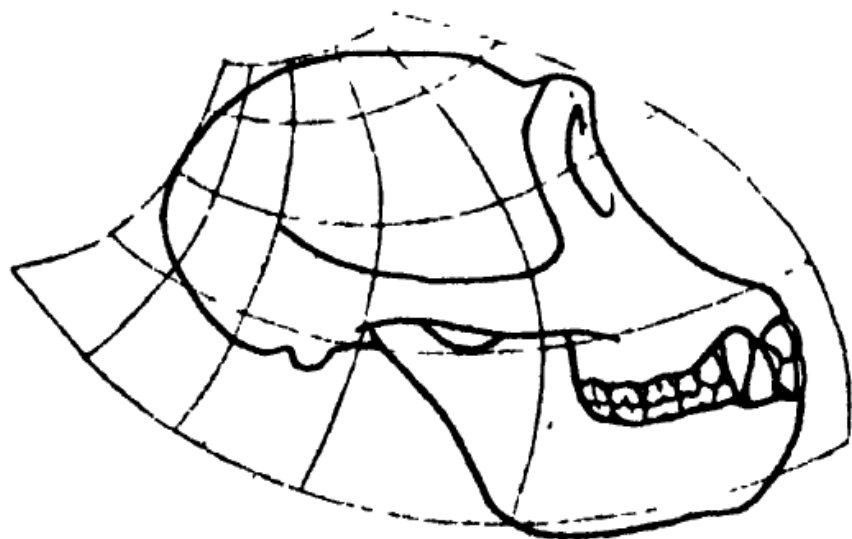


Fig. 550. Skull of chimpanzee.

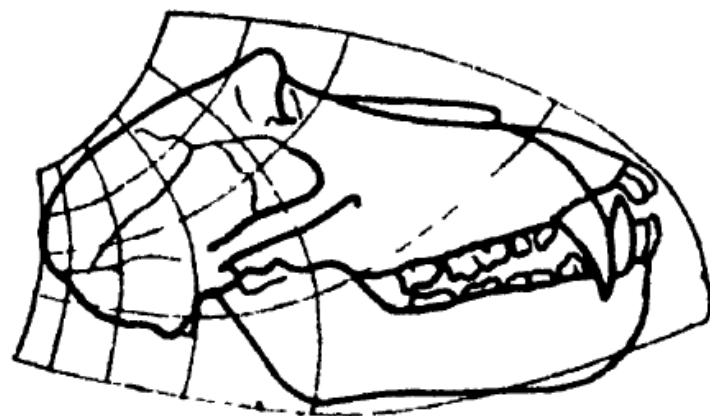


Fig. 551. Skull of baboon.

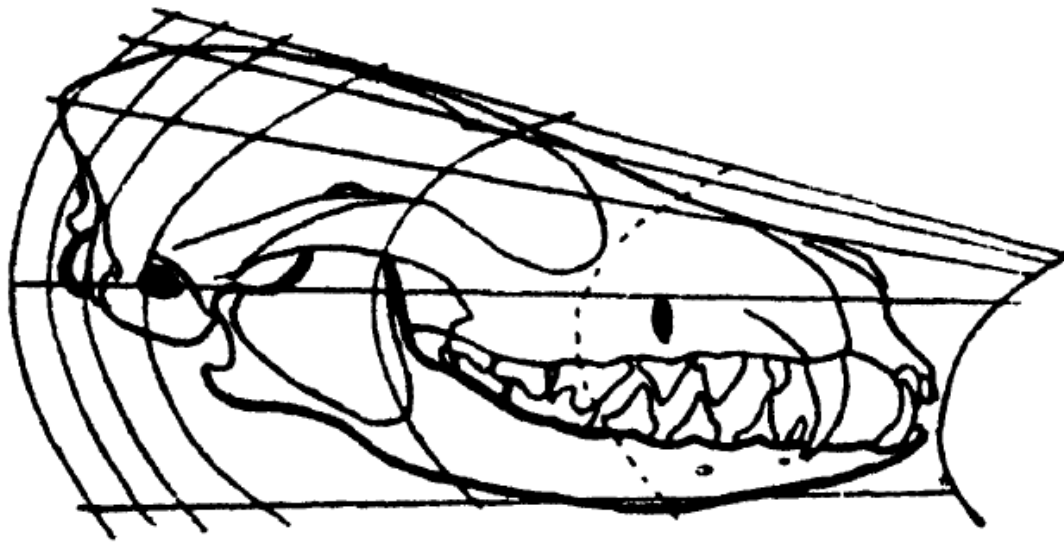


Fig. 552. Skull of dog, compared with the human skull of Fig. 548.

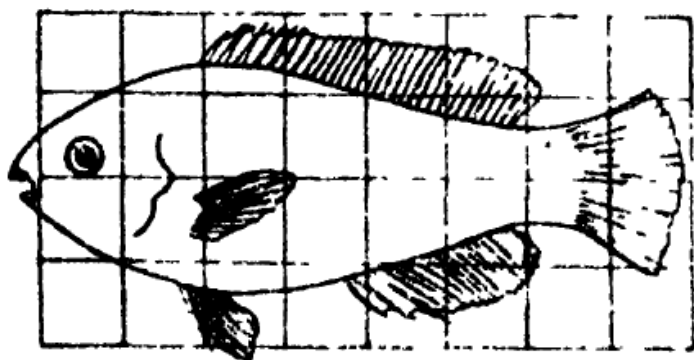


Fig. 519. *Scarus* sp.

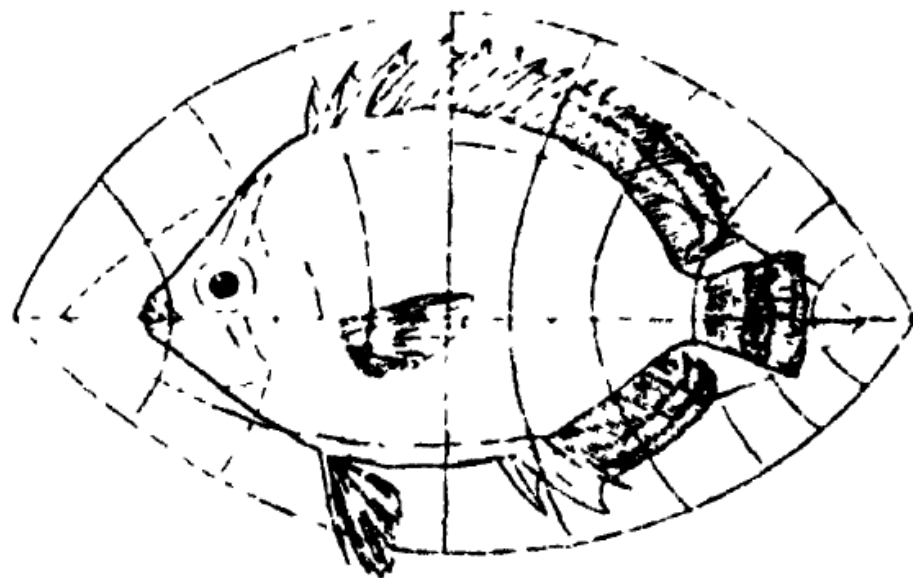


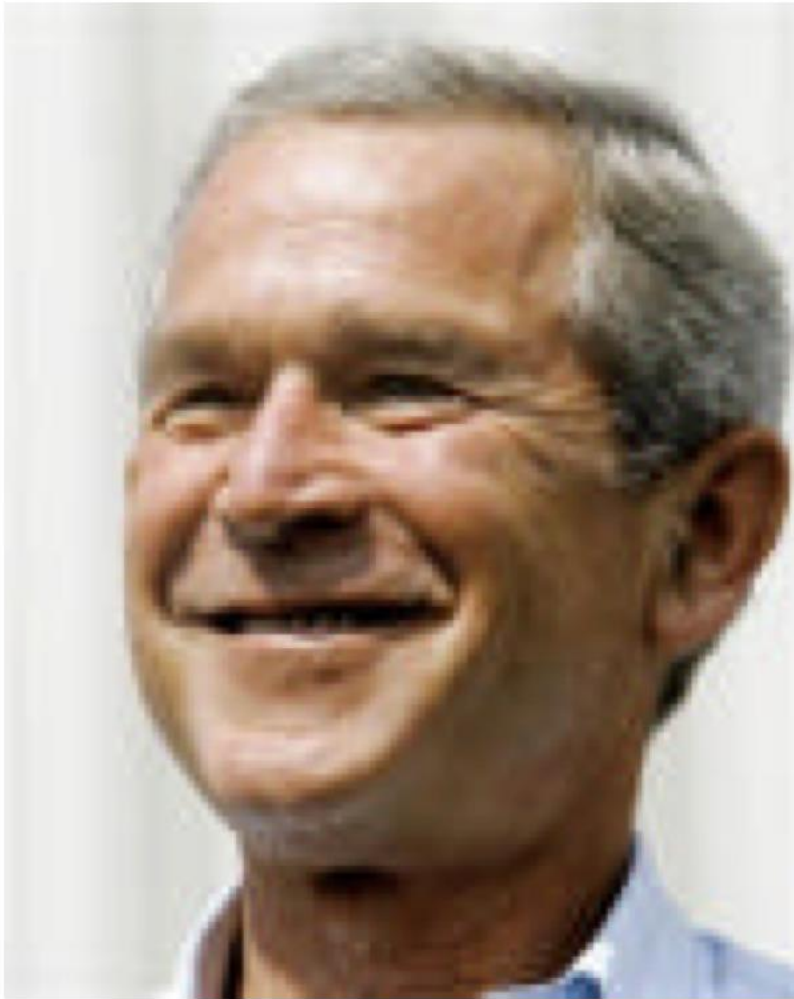
Fig. 520. *Pomacanthus*.



Original



Thin-Plate Splines



Original



Thin-Plate Splines

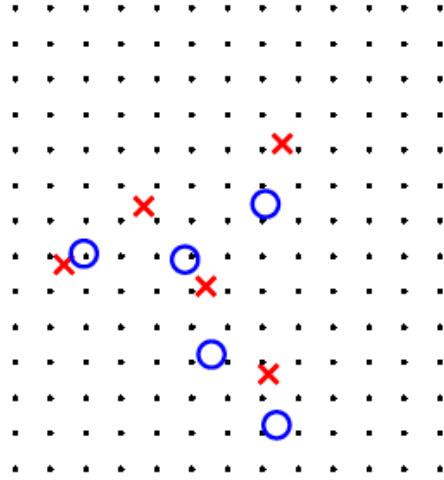


Original

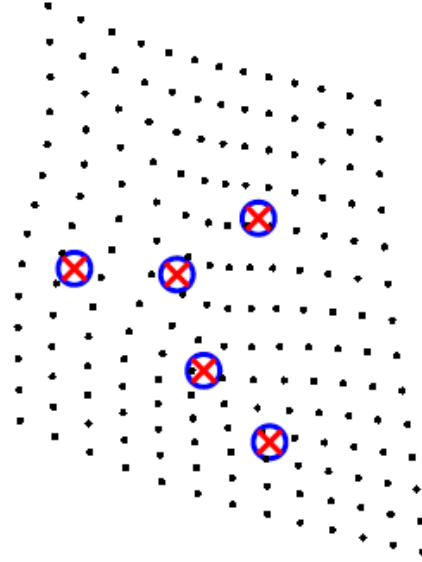


Thin-Plate Splines

(a)



(b)

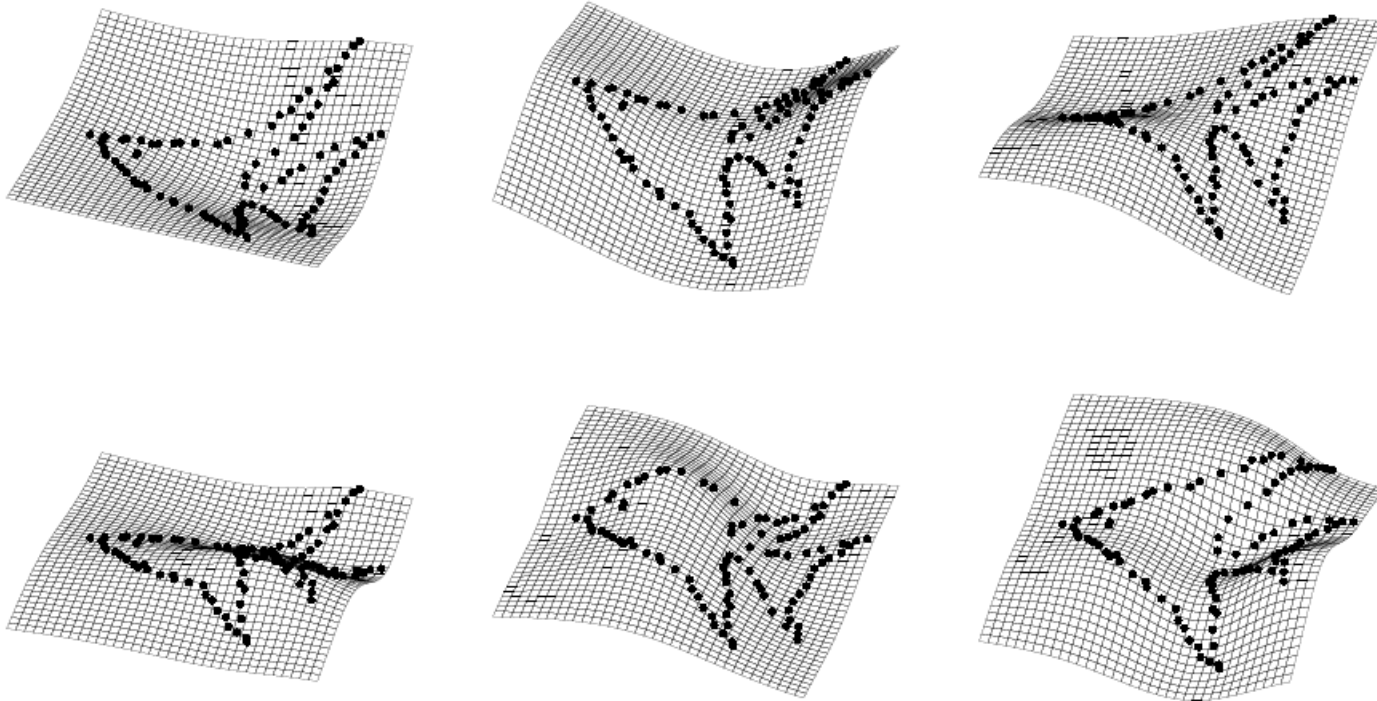


$$\left\{ \begin{array}{l} \forall i \ f_x(x_i, y_i) = x'_i \\ f_x = \operatorname{argmin}_g \left\{ I_g = \int \int_{\mathbb{R}^2} \left(\frac{\partial^2 g}{\partial x^2} \right)^2 + \left(\frac{\partial^2 g}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 g}{\partial y^2} \right)^2 \right\} \\ f_x(x, y) = v + v_x x + v_y y + \sum_{i=1}^n w_i U(\|(x_i, y_i) - (x, y)\|) \end{array} \right. \quad (17)$$

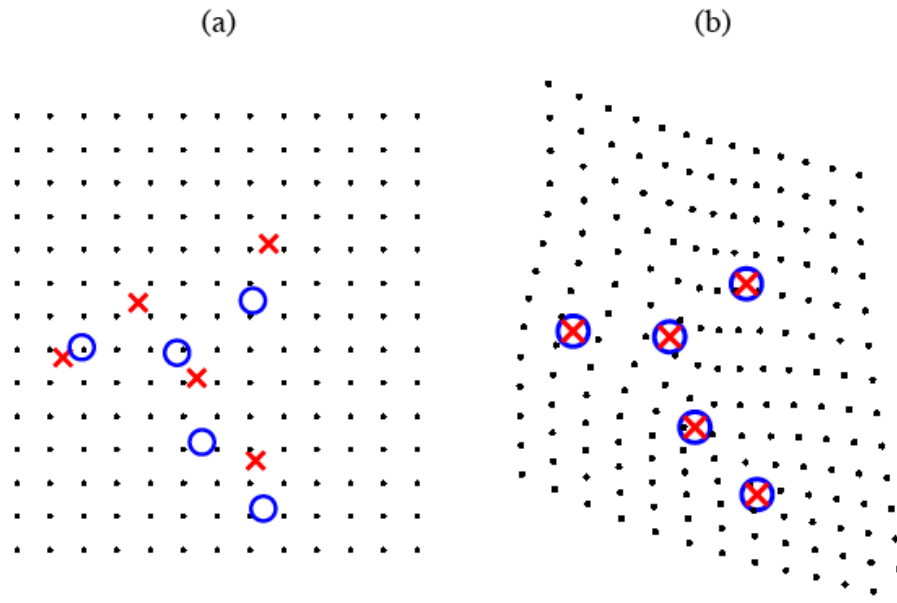
TPS model:

First the equation:

$$f(x, y) = a_1 + a_x x + a_y y + \sum_{i=1}^p w_i U(\|(x_i, y_i) - (x, y)\|)$$



how do we estimate the TPS parameters?



Say the blue circles are the source image features, and red crosses are the target image features, how do we “back-warp” all the image features in the target image back to the source image?

We could compute a TPS model that maps “red” to “blue”, and apply it to the rest of the target image pixels.

$$f(x, y) = a_1 + a_x x + a_y y + \sum_{i=1}^p w_i U(\|(x_i, y_i) - (x, y)\|)$$

how do we estimate the TPS parameters?

We would need two functions:

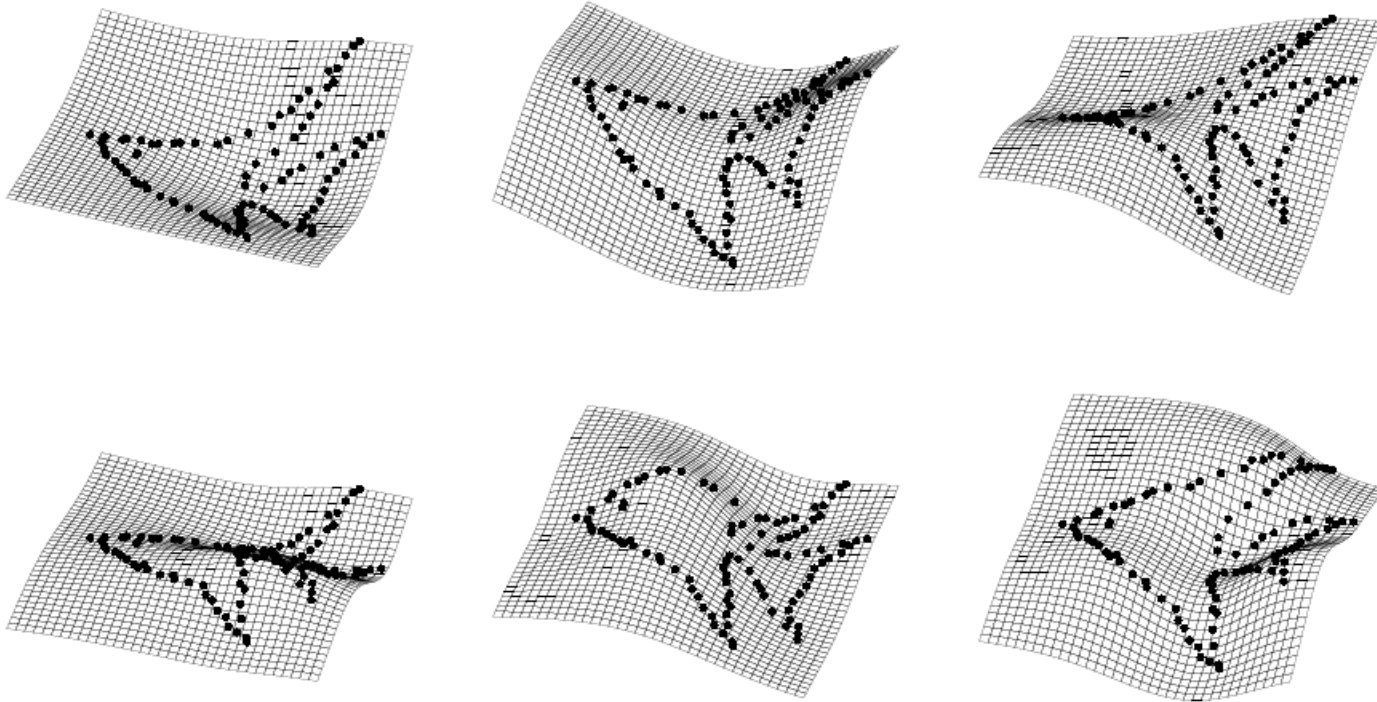
- 1) `tps_model = est_tps(source_pts, target_pts);`
- 2) `morphed_im = morph(im_source, tps_model);`

tps_model = (a1, ax, ay, w1, ..., wp)

$$f(x, y) = \boxed{a_1} + \boxed{a_x}x + \boxed{a_y}y + \sum_{i=1}^p \boxed{w_i} U(\|(x_i, y_i) - (x, y)\|)$$

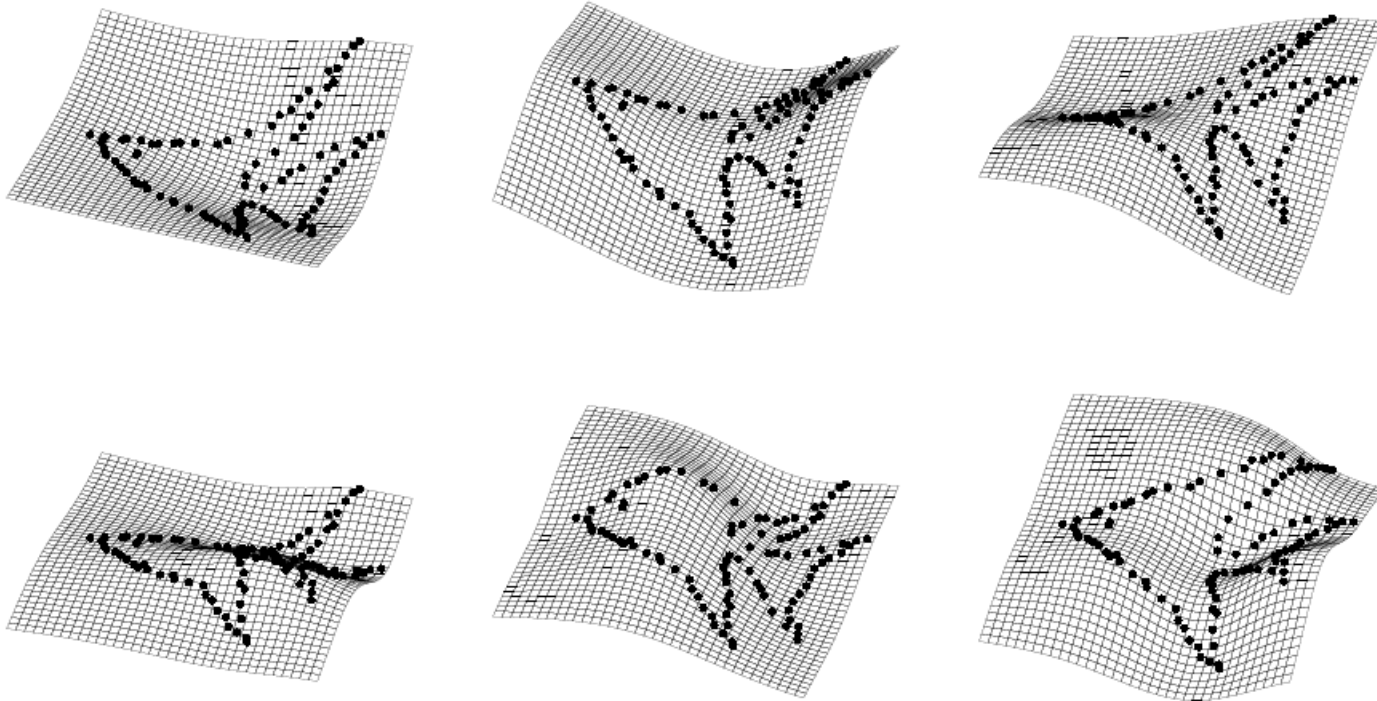
TPS model, special case 1, translation only

$$f(x, y) = \boxed{a_1} + a_x x + a_y y + \sum_{i=1}^p w_i U(\|(x_i, y_i) - (x, y)\|)$$



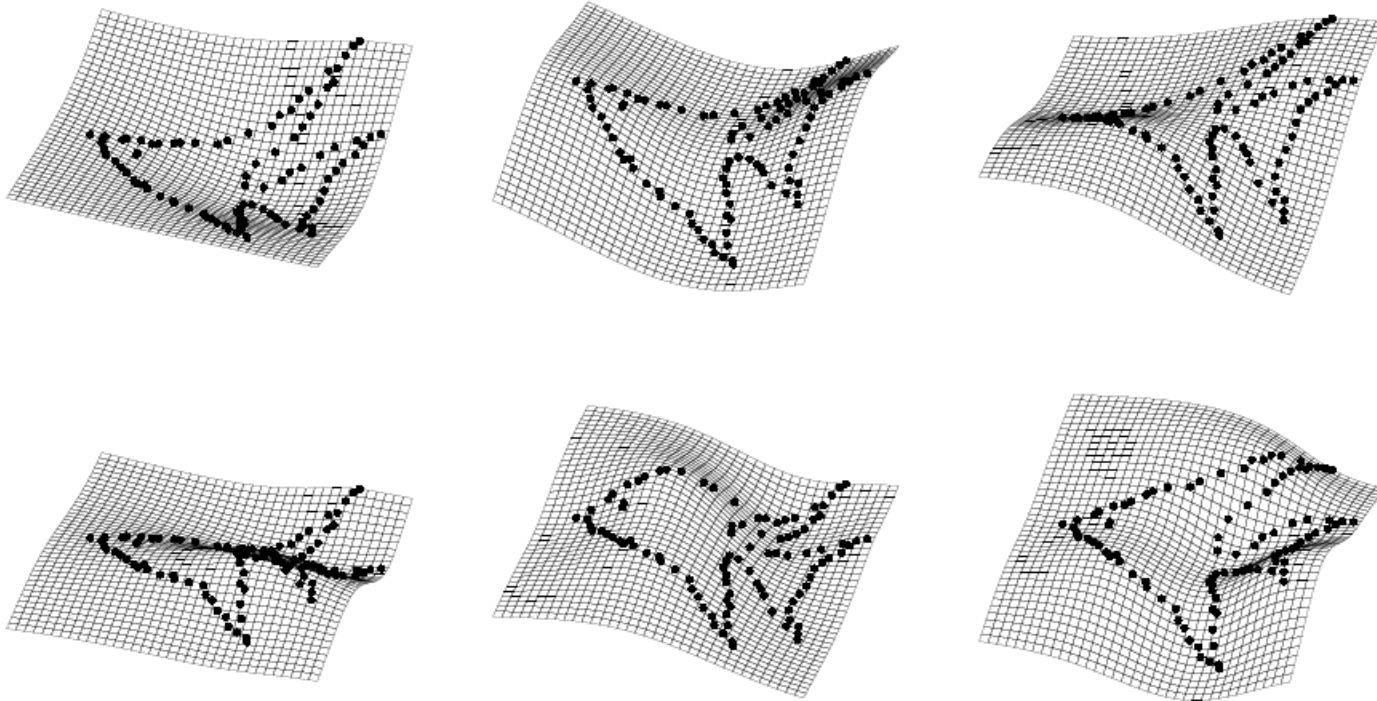
TPS model, case 2: Affine parameters only

$$f(x, y) = \boxed{a_1 + a_x x + a_y y} + \sum_{i=1}^p w_i U(\|(x_i, y_i) - (x, y)\|)$$

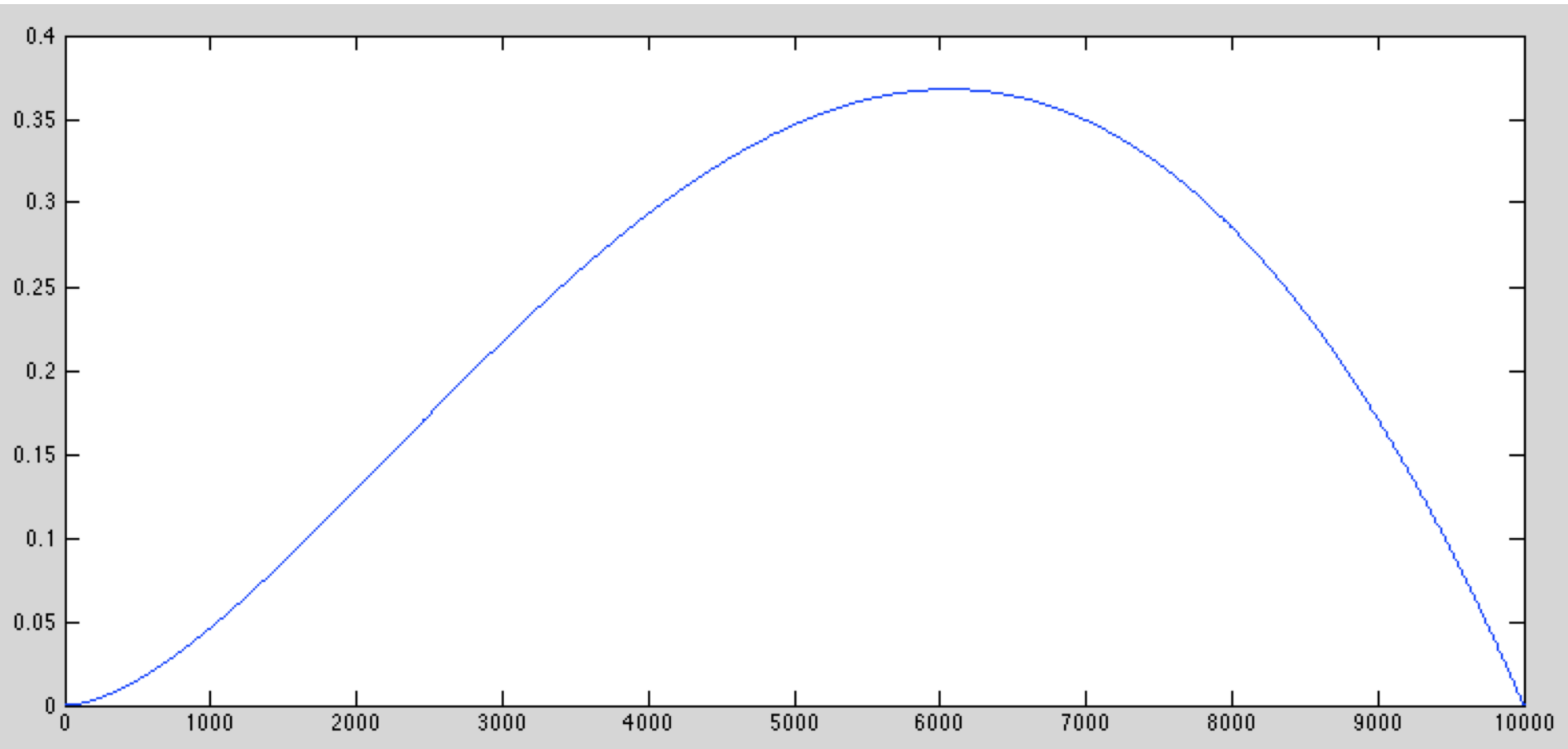


TPS model: Full general case:

$$f(x, y) = a_1 + a_x x + a_y y + \sum_{i=1}^p w_i U(\|(x_i, y_i) - (x, y)\|)$$



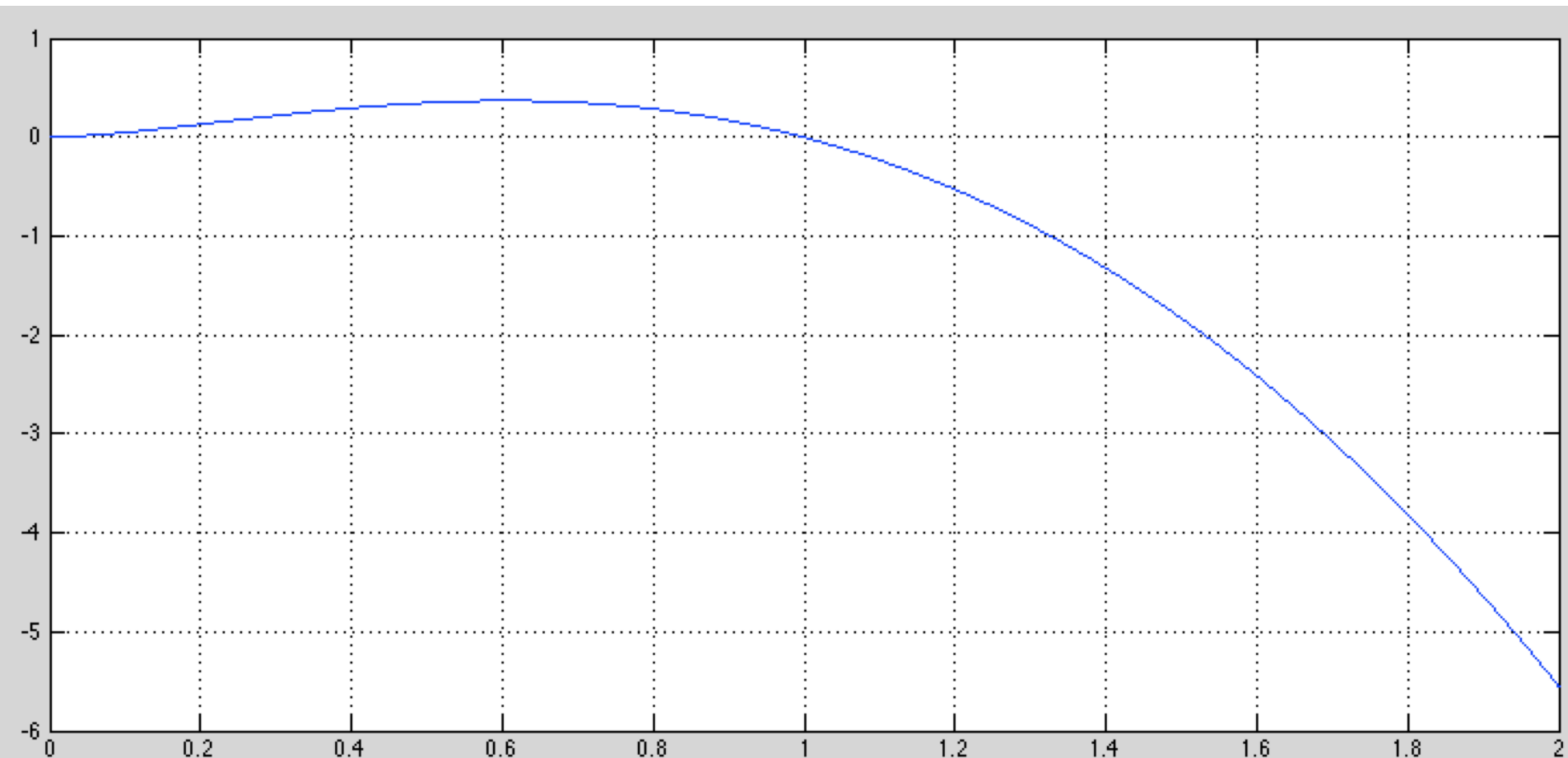
$$U(r) = -r^2 \log(r^2)$$



$r = [0, 1]$

Max at $r = 1/\sqrt{e} = 0.607$

$$U(r) = -r^2 \log(r^2)$$



$$r = [0, 2]$$

TPS model:

$$f(x, y) = a_1 + a_x x + a_y y + \sum_{i=1}^p w_i U(\|(x_i, y_i) - (x, y)\|)$$

$$K_{ij} = U(\|(x_i, y_i) - (x_j, y_j)\|),$$

i th row of P is $(1, x_i, y_i)$,

$$\begin{bmatrix} K & P \\ P^T & O \end{bmatrix} \begin{bmatrix} w \\ a \end{bmatrix} = \begin{bmatrix} v \\ o \end{bmatrix}$$

Compute TPS Coefficients (est_tps)

$$f(x, y) = a_1 + a_x \cdot x + a_y \cdot y + \sum_{i=1}^P W_i U(\|(x_i, y_i) - (x, y)\|),$$

$$U(r) = r^2 \log(r^2)$$

- Compute two sets of coefficients for x and y:
 - $x' = f_x(x, y)$
 - $y' = f_y(x, y)$

- Note: Take care of $r = 0$.

Compute TPS Coefficients cont.

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

where

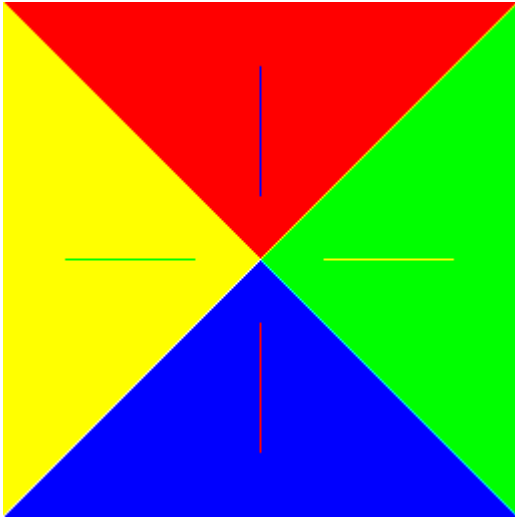
$$K_{ij} = U(\|(x_i, y_i) - (x_j, y_j)\|),$$

$$\begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \text{inv}\left(\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} + \lambda * I(n+3, n+3) \right) \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

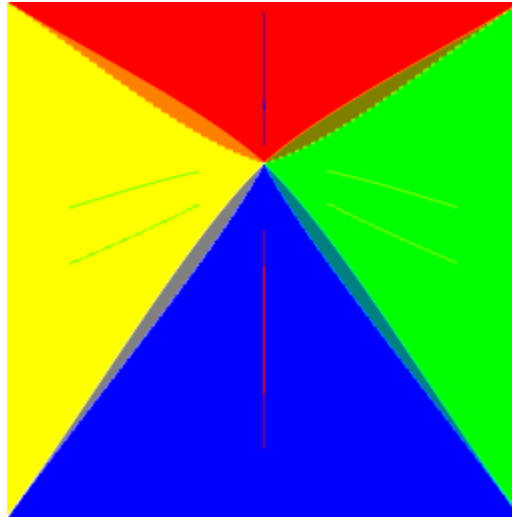
- Note: λ should be very small.

Example

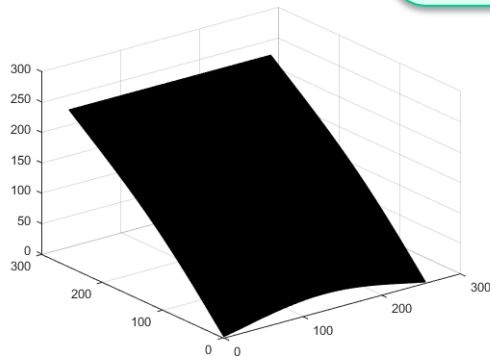
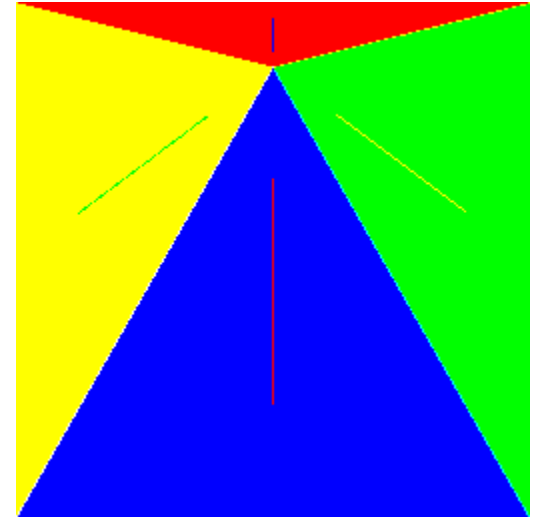
Frac = 0



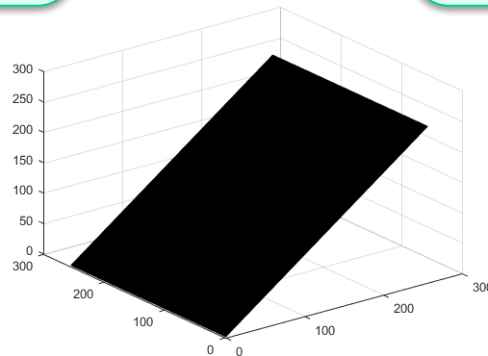
Frac = 0.5



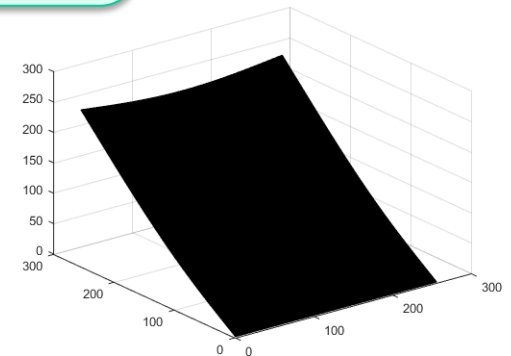
Frac = 1



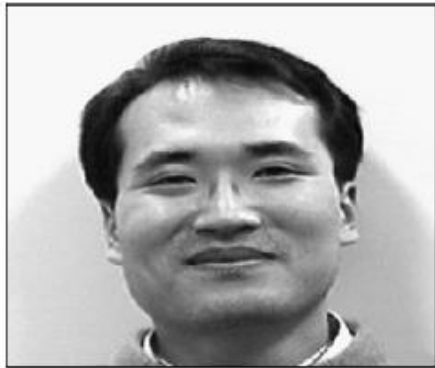
X surface



Y unchanged



X surface



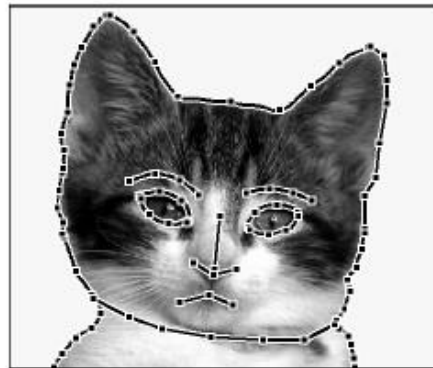
(a)



(b)



(c)



(d)

