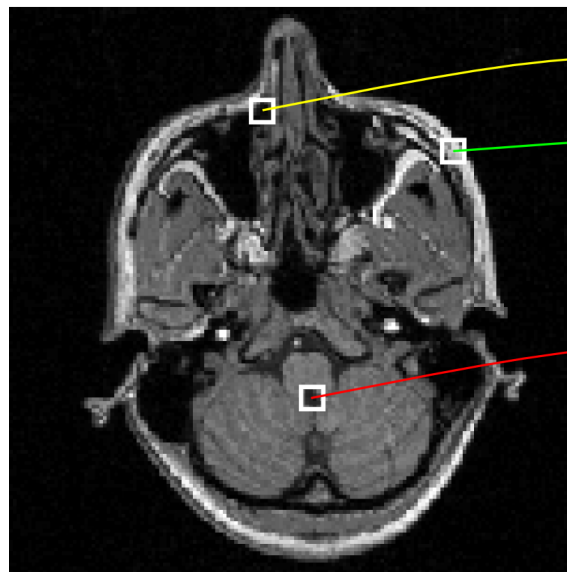
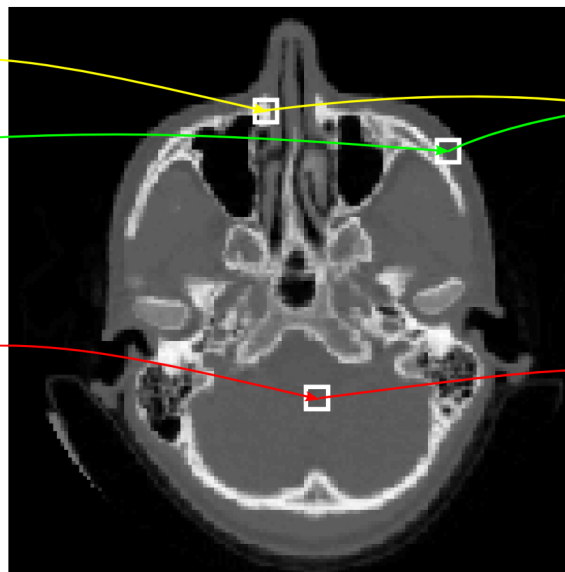


# Chapter 4

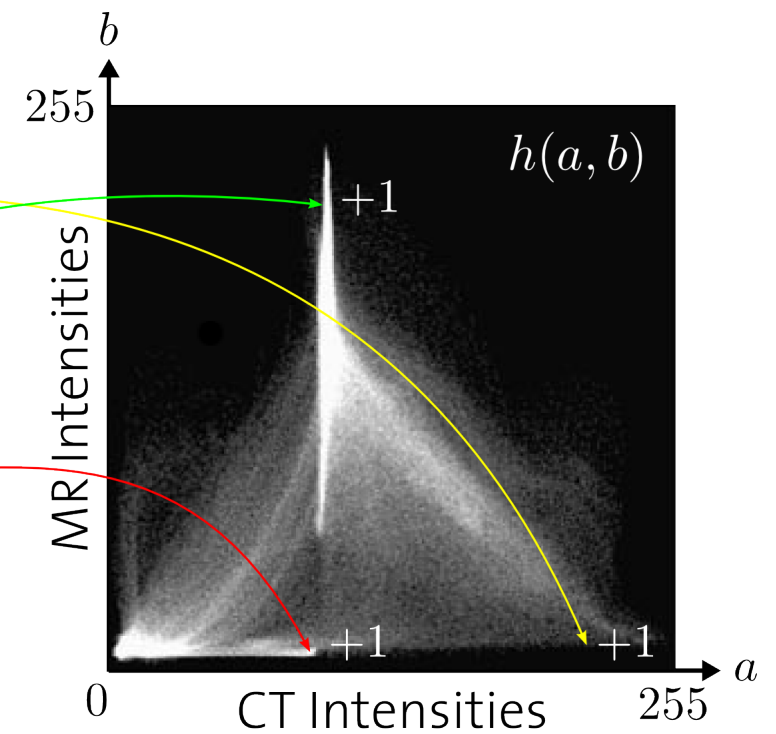
## Optimization without Gradients



MR Image



CT Image



# Optimization without Gradients

- Optimization **with gradient information**: steepest descent, conjugate gradients, Newton etc. (will be covered in the Numerical Analysis course)
- Sometimes **direct methods** without gradient information are needed:
  - function is **not differentiable**,
  - gradients are **difficult to compute**,
  - gradient-based optimization problematic due to **many local optima**.
- Example: Image registration (i.e. spatial alignment of images)
- Proposed method: Downhill-Simplex (a.k.a. Nelder-Mead) method

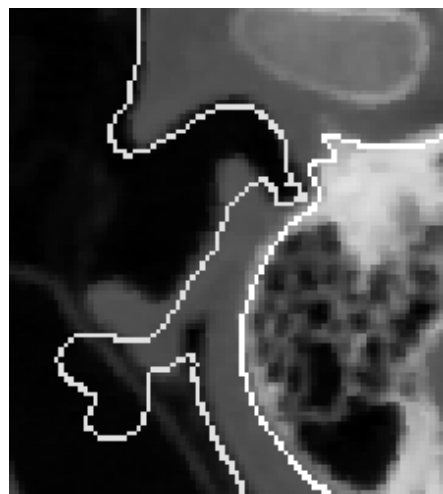
# Example: Multi-modal Image Registration (ear)

Magnetic resonance imaging (MRI): atomic nuclei oriented in external magnetic field, absorption of RF energy  $\rightsquigarrow$  spin polarization  $\rightsquigarrow$  RF signal in detector. Basically measures local proton density. Potential problem: Spatial distortions due to in-homogeneity of magnetic field.

Computed tomography (CT): Measures local absorption coefficients for X-rays from external source.



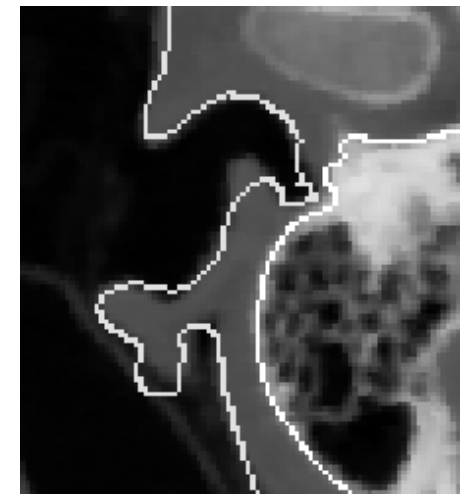
Original MR



Original CT with  
MR contour



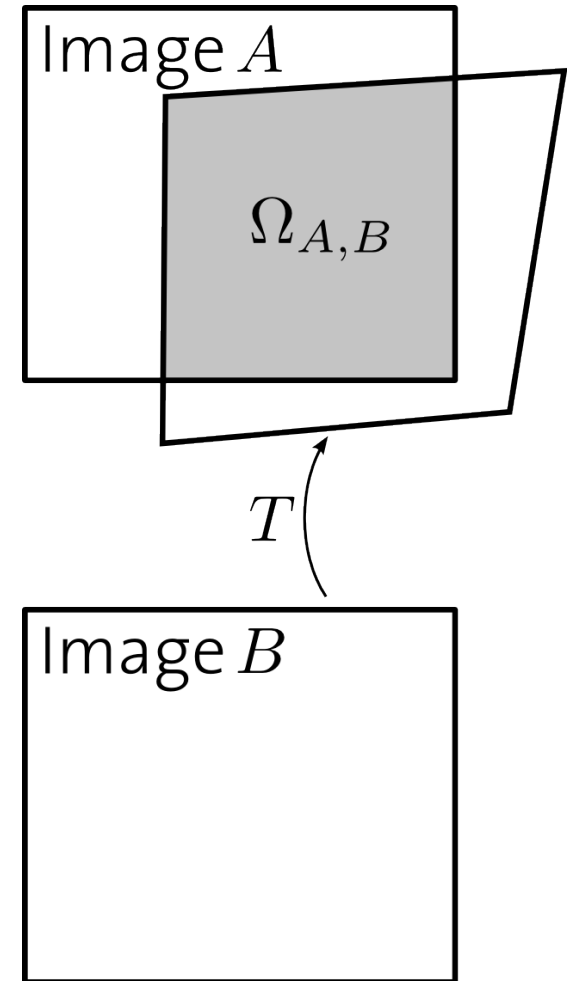
Registered MR



CT with registered  
MR contour

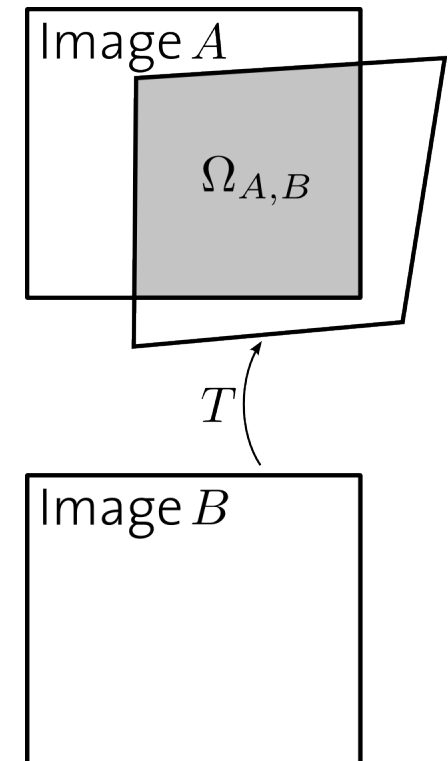
# Problem Definition

- Given: Target or **reference image**  $A$  and the **floating image**  $B$ .
- Task: Find a **reasonable transformation**  $T$ , such that the transformed image  $T(B)$  is **similar** to  $A$ ,
- where **reasonable transformations** are ensured through a **regularization** and the **similarity** is defined by a **similarity measure**  $C$



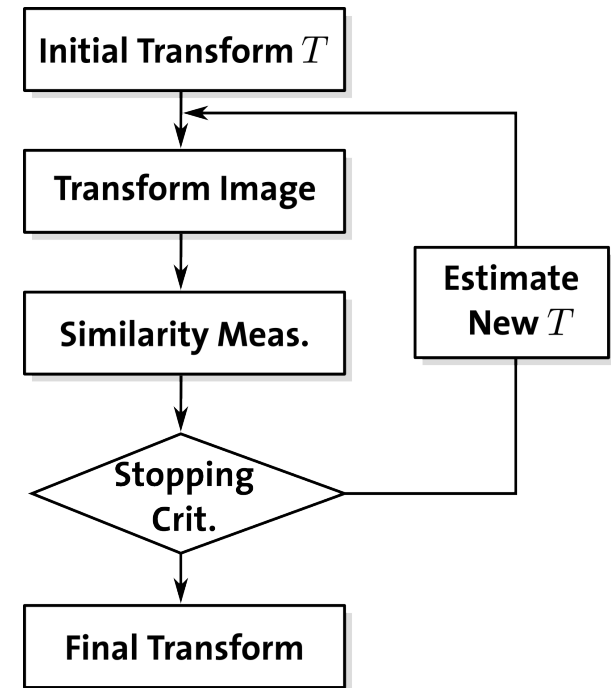
# Terminology

- **Reference image  $A$** : kept unchanged and used as the reference
- **Floating image  $B$** : spatially warped to align with the reference image
- **Transformation  $T()$** : class of allowed transformations to warp the floating image onto the reference image.
- **Similarity measure  $C$** : metric used to **quantify the registration success**.
- **Overlap Domain  $\Omega_{A,B}$**



# Rigid Registration Algorithm

- **Rigid registration:** compensate for the **global rigid transformation** between the images. Applicability is **limited to special cases**.
- Select the **initial transform**  $T$
- **Transform** the floating image
- Calculate the **quality of the fit** using the **similarity measure**.
- Verify the **stopping criterion:** if the fit is still not good enough estimate a **new**  $T$ . Otherwise transform the floating image to its **final position**.

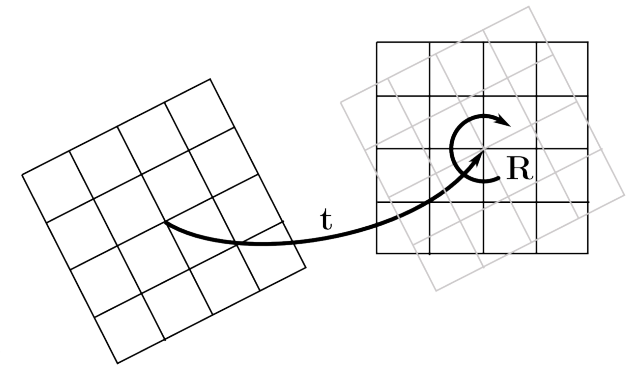


# Rigid Transformation Model

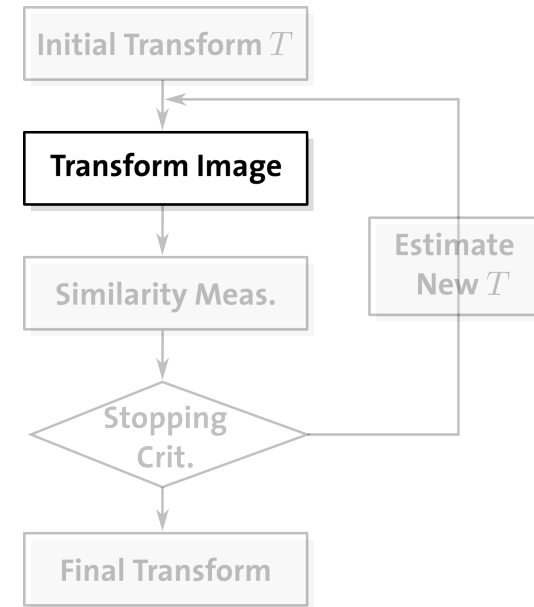
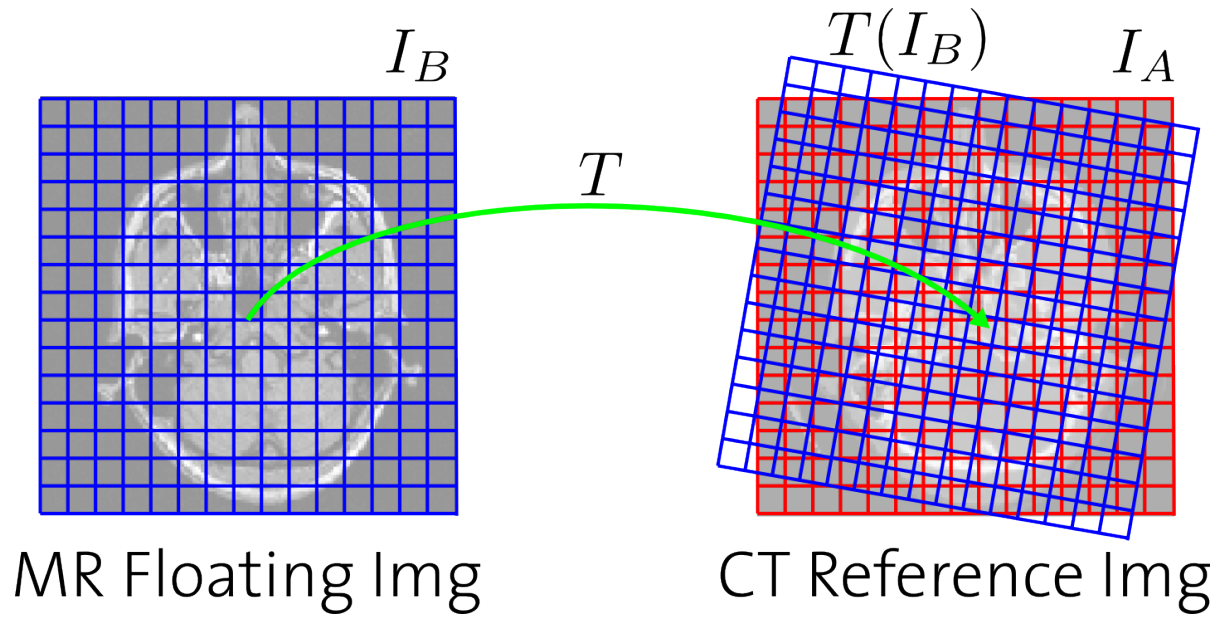
As the rigid transformation model **preserves Euclidean distances** it is also known as **isometry** (from iso = same, metric = measure).

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \epsilon \cos \theta & -\sin \theta & t_x \\ \epsilon \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

where  $\epsilon = \pm 1$ . If  $\epsilon = 1$  then the isometry is **orientation-preserving** and is composed of a **translation and rotation**. If  $\epsilon = -1$  then the isometry reverses orientation (not useful here).



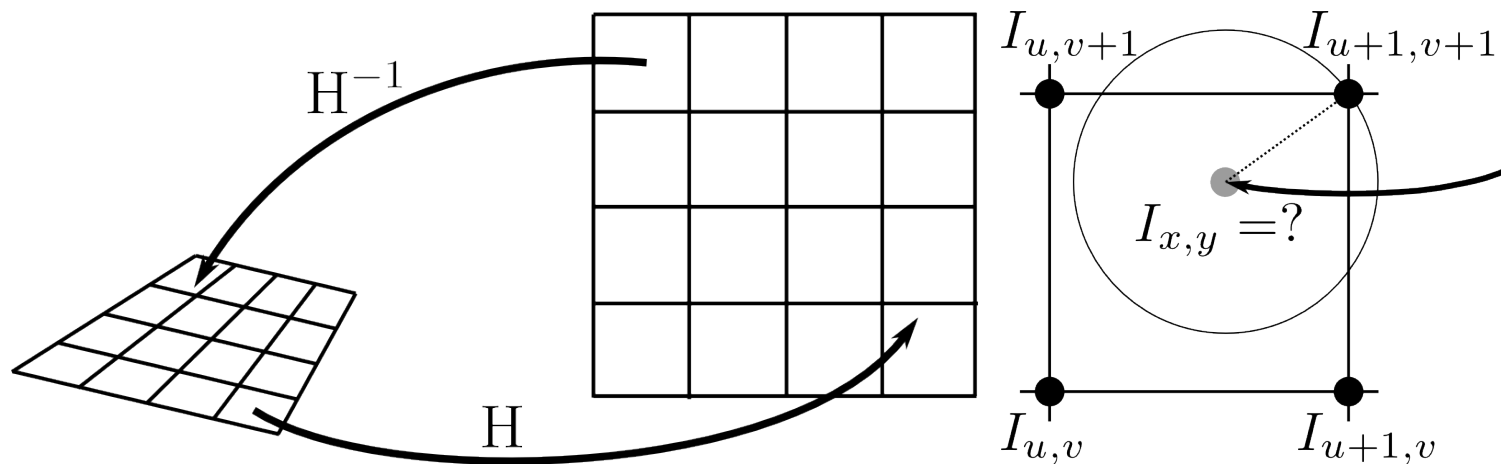
# Pixel Interpolation



# Spatial Transformations

Given the **spatial transformation**  $H$ , the floating image has to be mapped into the output image. **Two approaches** are common:

- Forward mapping:  $x' = Hx$
- Backward mapping:  $x = H^{-1}x'$



**Problem with both approaches:** The pixel coordinate  $x$  generally **does not fall onto an exact pixel location**  $\Rightarrow$  **interpolation needed.**

# Bilinear Interpolation

**Idea:** compute weighted average of the **four closest pixels**:

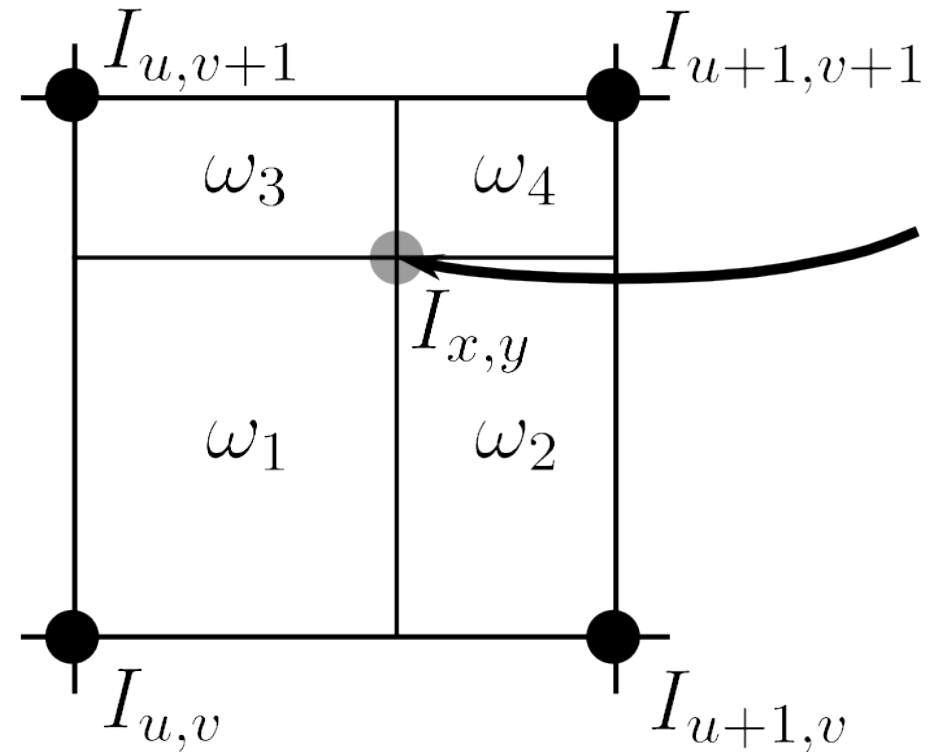
$$I_{x,y} = \omega_4 I_{u,v} + \omega_2 I_{u,v+1} + \omega_3 I_{u+1,v} + \omega_1 I_{u+1,v+1},$$

$$\omega_1 = (x - u)(y - v)$$

$$\omega_2 = (u + 1 - x)(y - v)$$

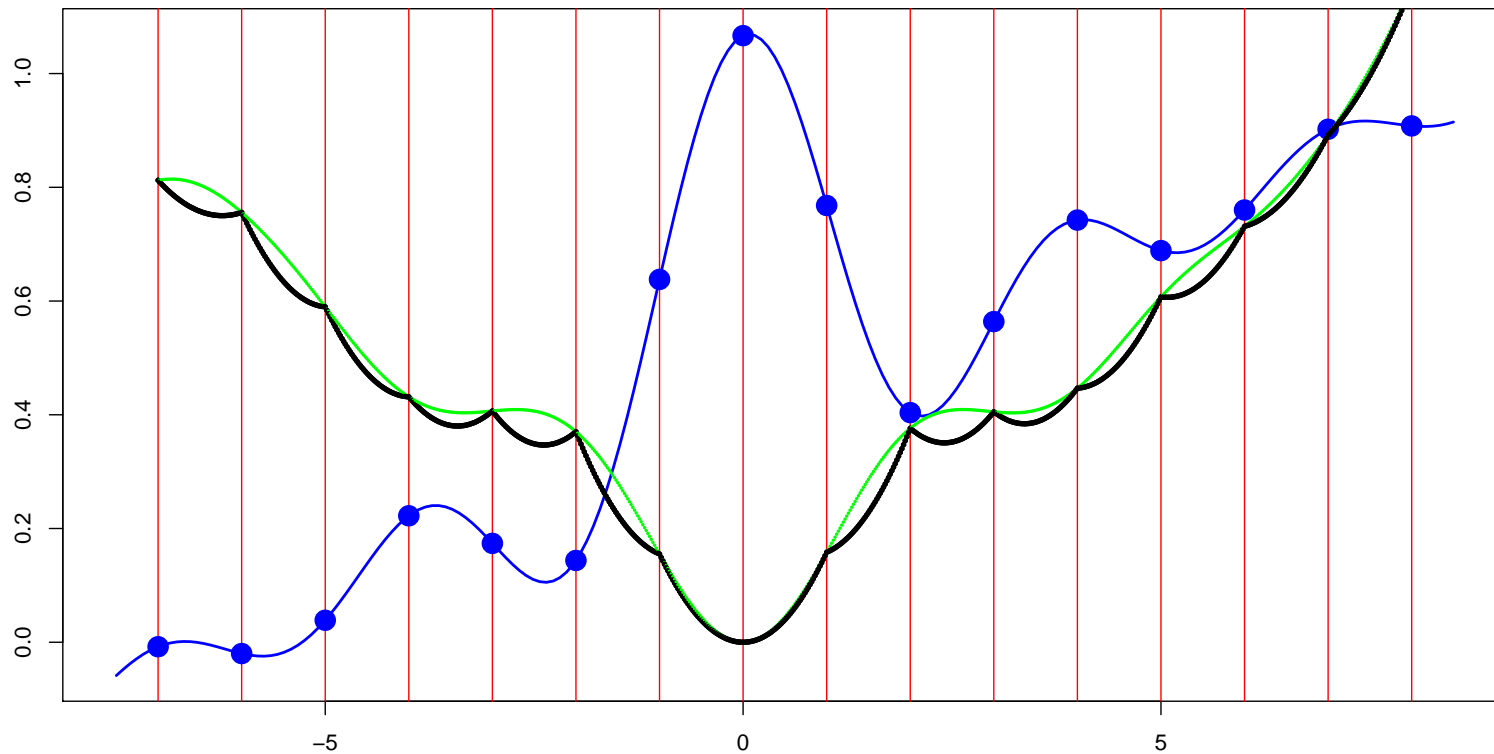
$$\omega_3 = (x - u)(v + 1 - y)$$

$$\omega_4 = (u + 1 - x)(v + 1 - y)$$



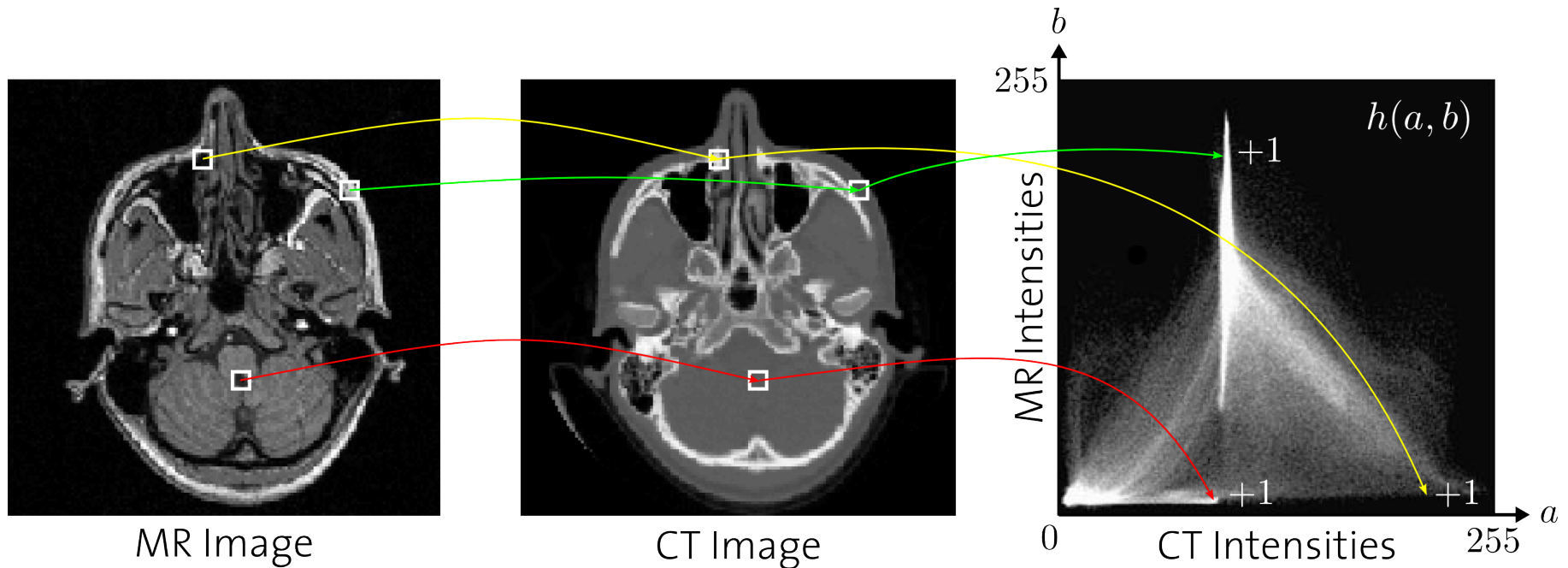
# Artifacts of Interpolation in Similarity Measures

1D-Example: approximate  $f(x) = ax + b \sin(x)/x + c$  on a grid, consider translation  $f_t(x) = f(x + \Delta)$ , use linear interpolation and measure similarity as a function of  $\Delta$  by the sum of squared differences:



# Similarity Measures based on the Joint Histogram

The **Joint- or 2D Histogram** forms the basis of most **similarity measures** in **multi-modal registration**.



The value at position  $a, b$  is the **number of pixels with value  $a$  in one modality and value  $b$  at the same location in the other modality**.

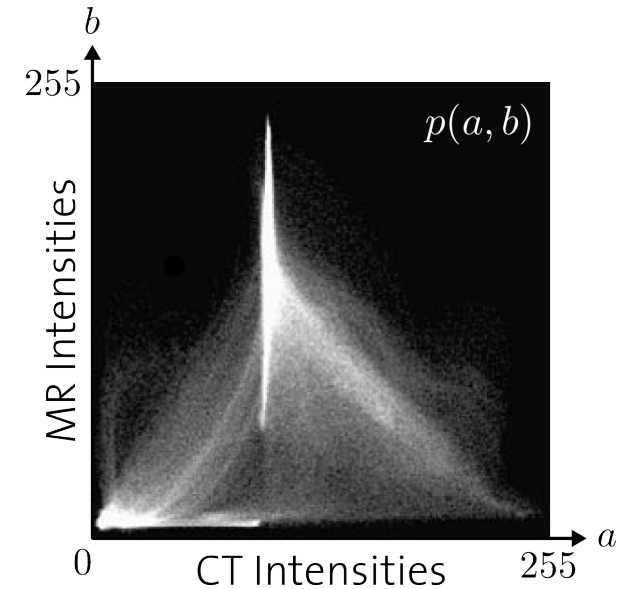
As the intensities are only related by their **co-occurrence** and not by their value, the similarity measure can handle **multi-modal images**.

## Joint Histogram (2)

Scaling the joint histogram with the total number of pixel pairs  $N$  yields an approximation of the **joint probability**

$$p(a, b) = \frac{1}{N}h(a, b)$$

$p(a, b)$  represents the **probability of the pixel pair with intensities  $a$  and  $b$  to occur in the two images.**

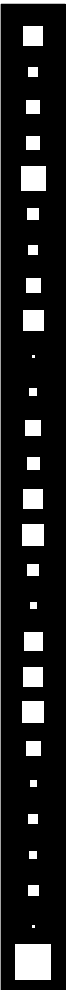


# Review: Joint and Conditional Probabilities

Probability distribution over the 27 outcomes for a randomly selected letter in an English language document (estimated from *The Frequently Asked Questions Manual for Linux* ).

The picture shows the probabilities by the areas of white squares.

$i$	$a_i$	$p_i$	
1	a	0.0575	a
2	b	0.0128	b
3	c	0.0263	c
4	d	0.0285	d
5	e	0.0913	e
6	f	0.0173	f
7	g	0.0133	g
8	h	0.0313	h
9	i	0.0599	i
10	j	0.0006	j
11	k	0.0084	k
12	l	0.0335	l
13	m	0.0235	m
14	n	0.0596	n
15	o	0.0689	o
16	p	0.0192	p
17	q	0.0008	q
18	r	0.0508	r
19	s	0.0567	s
20	t	0.0706	t
21	u	0.0334	u
22	v	0.0069	v
23	w	0.0119	w
24	x	0.0073	x
25	y	0.0164	y
26	z	0.0007	z
27	–	0.1928	–



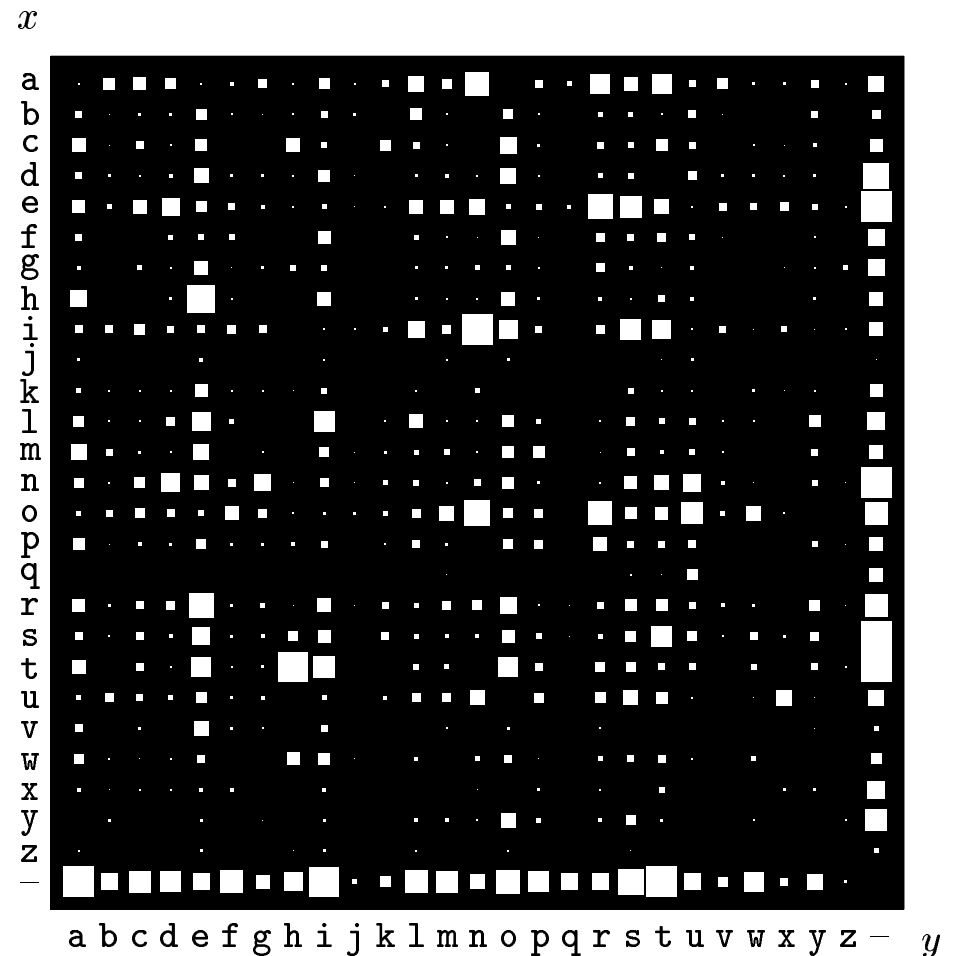
David J.C. MacKay, Cambridge University Press, 2003.

# Review: Joint and Conditional Probabilities

The probability distribution over the  $27 \times 27$  possible **bigrams**  $xy$  in *The Frequently Asked Questions Manual for Linux*.

Relation to marginals:

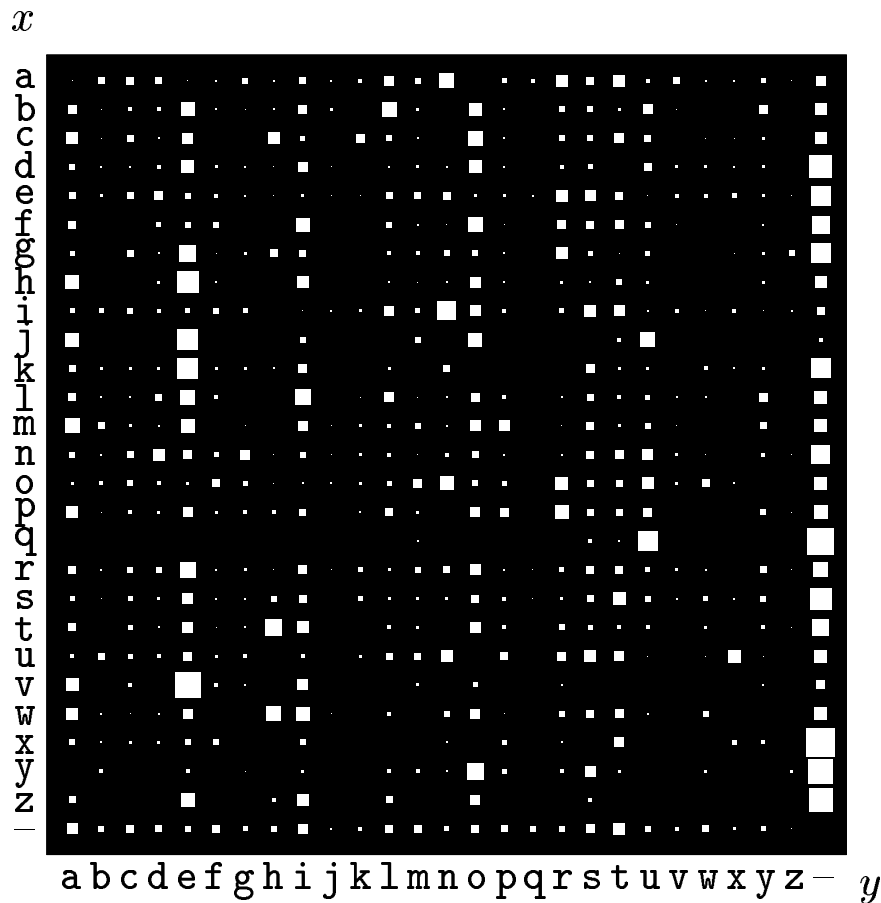
$$p(x) = \sum_{y \in \mathcal{Y}} p(x, y).$$



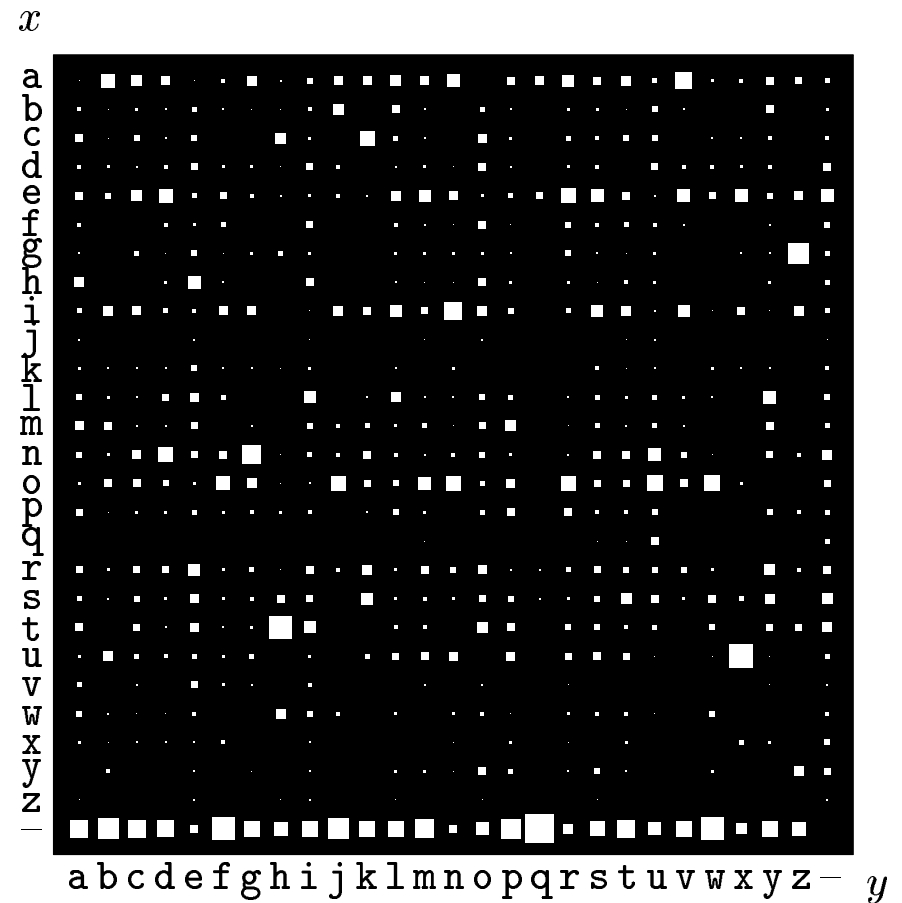
Information Theory, Inference, and Learning Algorithms,  
David J.C. MacKay, Cambridge University Press, 2003.

# Review: Joint and Conditional Probabilities

(a)  $p(y|x)$ : Each row shows the conditional distribution of the second letter,  $y$ , given the first letter,  $x$ , in a bigram  $xy$ . (b) vice versa.



(a)  $P(y | x)$



(b)  $P(x | y)$

# Excursion to Information Theory

- Image registration: maximizing the amount of information shared by the two images  $\rightsquigarrow$  suggests the use of a measure of information.
- The most commonly used: **Shannon-Wiener entropy**  $H$

$$H = - \sum_{i=1}^n p_i \log p_i = - \sum_{x \in \mathcal{X}} p(x) \log p(x)$$

- Entropy  $H$  will have a
  - **maximum** if all symbols have equal probability  $p_i = 1/n, \forall i$
  - **minimum** of zero if the probability of one symbol is 1 (all others 0).  
Note that  $0 \log 0 = 0$ .

# Interpretation of Entropy

- Logarithms of base 2  $\rightsquigarrow$  entropy measured in **bits**.
- Entropy is a measure of the average uncertainty in a RV:  
**number of bits on the average required to describe the RV.**
- **Example:** uniform distribution over 32 outcomes  
 $\rightsquigarrow$  for identifying an outcome we need a label that takes 32 different values  
 $\rightsquigarrow$  5-bit strings suffice.

$$H(X) = - \sum_{i=1}^{32} \frac{1}{32} \log \frac{1}{32} = - \log \frac{1}{32} = \log 32 = 5 \text{ bits}$$

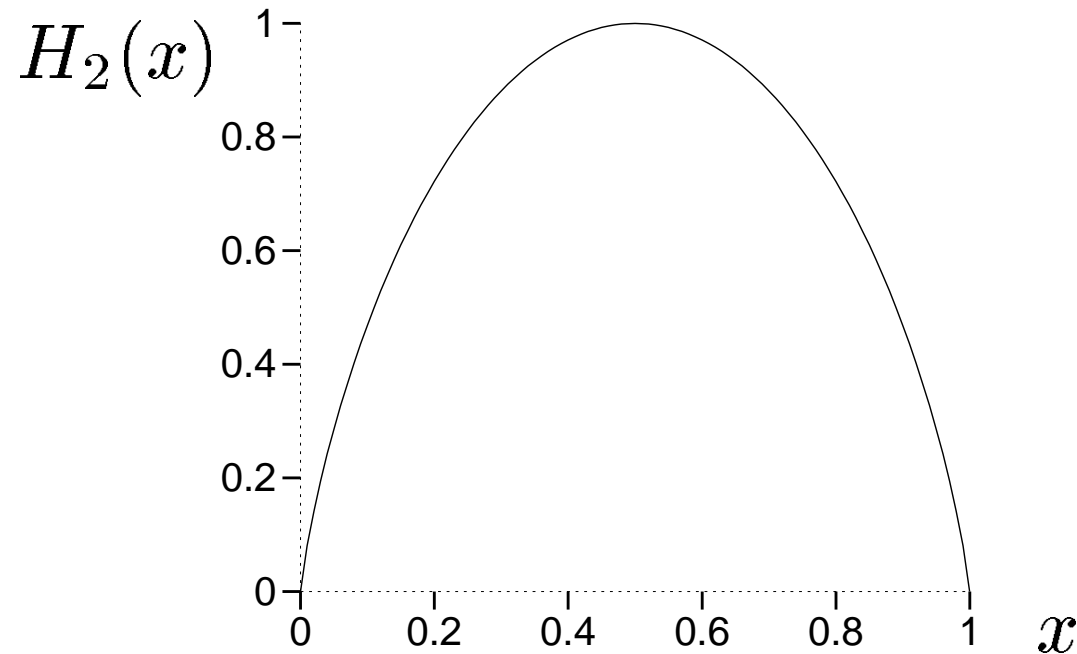
# Interpretation of Entropy (2)

**Example:**

$$X = \begin{cases} 1 & \text{with prob. } p \\ 0 & \text{with prob. } 1 - p. \end{cases}$$

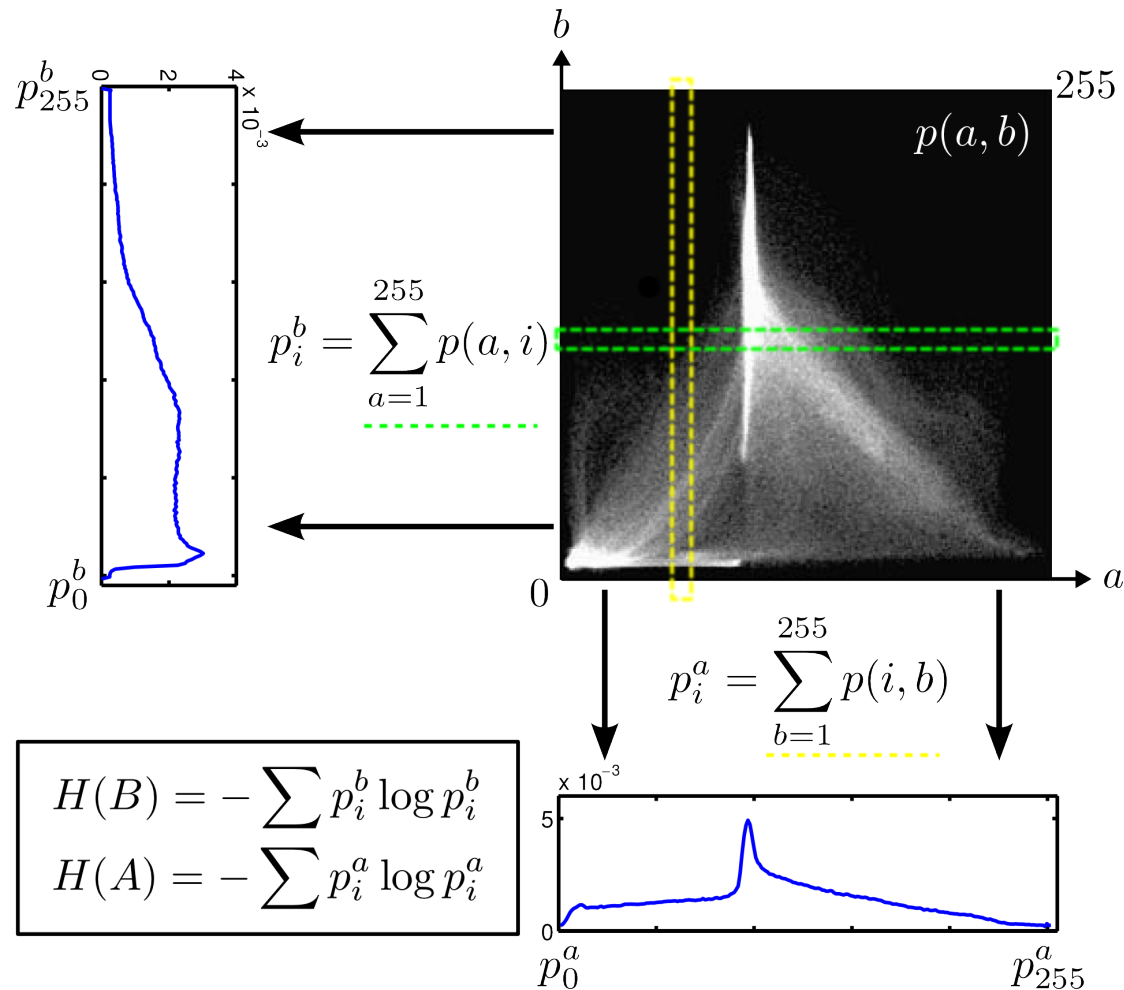
**Entropy:**  $H(X) = -p \log p - (1 - p) \log(1 - p)$

**Special cases:**  $p = 1/2 \Rightarrow H(X) = 1$ ,  $p = 0$  or  $1 \Rightarrow H(X) = 0$



# Entropy of Images

The entropy of two registered images  $A$  and  $B$  can be determined from the joint probability  $p(A, B)$  – estimated by the joint histogram on the overlap domain  $\Omega_{A,B}$  – via marginalization:



# Joint Entropy

Joint entropy measures uncertainty in combined RVs  $(X, Y)$ :

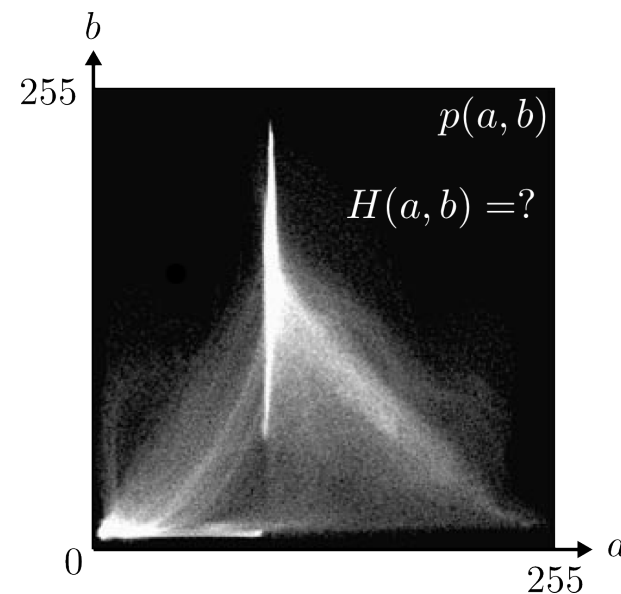
$$H(X, Y) = - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y)$$

If  $X, Y$  are independent, the joint entropy is the sum of the individual entropies

$$H(X, Y) = H(X) + H(Y)$$

The less independent ( the more “similar” )  $X$  and  $Y$  are, the lower the joint entropy compared to the sum of the individual entropies

$$H(X, Y) \leq H(X) + H(Y)$$



# Conditional Entropy

Conditional Entropy = entropy of one RV given another  
= expected value of entropies of conditional distributions,  
averaged over the conditioning variable:

$$H(Y|X) = \sum_{x \in \mathcal{X}} p(x) H(Y|X = x)$$

Let the **combined system** determined by two random variables  $X, Y$  have **joint entropy**  $H(X, Y)$

↪ we need  $H(X, Y)$  bits of information to describe its exact state.

**Observing**  $X$  gives us  $H(X)$  bits of information

↪ we only need  $H(X, Y) - H(X)$  bits.

This quantity is  $H(Y|X)$  ↪ **chain rule of conditional entropy:**

$$H(X, Y) = H(X) + H(Y|X)$$

# Conditional Entropy

$$\begin{aligned} H(Y|X) &\equiv \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \\ &= - \sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x) \\ &= - \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log p(y|x) \\ &= - \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)}. \end{aligned}$$

$H(Y|X) = 0$  if and only if the value of  $Y$  is completely determined by the value of  $X$  (then,  $p(y|x)$  is a degenerate  $(0, 1)$  probability, and  $0 \log 0 = 0 = 1 \log 1$ )

Conversely,  $H(Y|X) = H(Y)$  if and only if  $Y$  and  $X$  are independent.

## Conditional Entropy: Chain rule

The chain rule follows from the above definition of conditional entropy:

$$\begin{aligned} H(Y|X) &= - \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \left( \frac{p(x, y)}{p(x)} \right) \\ &= - \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log(p(x, y)) + \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log(p(x)) \\ &= H(X, Y) + \sum_{x \in \mathcal{X}} p(x) \log(p(x)) \\ &= H(X, Y) - H(X). \end{aligned}$$

# Mutual Information

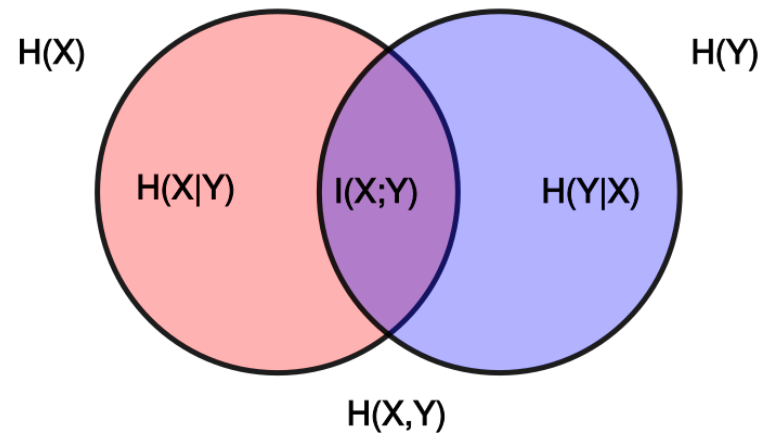
**Mutual Information**  $I(X; Y)$  = reduction in uncertainty of  $X$  due to knowledge of  $Y$  (and vice versa):

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Since  $H(X, Y) = H(X) + H(Y|X)$ , it follows that

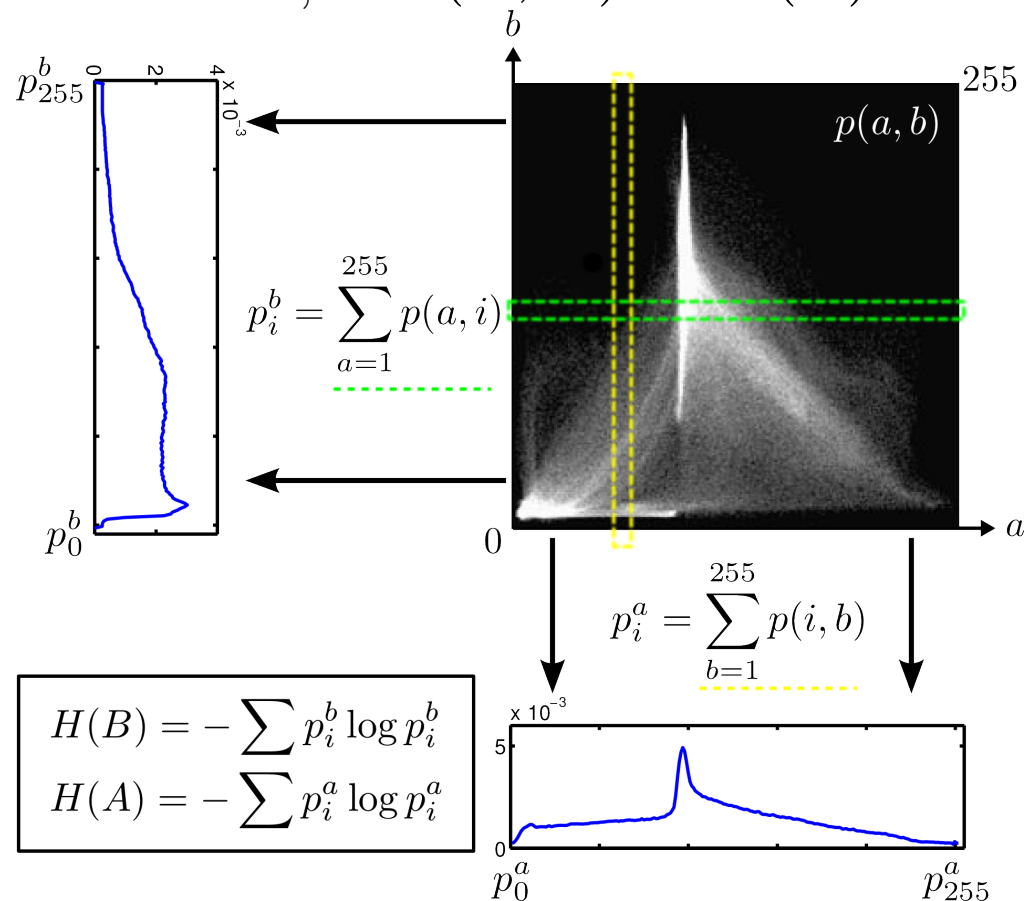
$$I(X; Y) = H(X) + H(Y) - H(X, Y).$$

If  $X$  and  $Y$  independent:  $\Rightarrow H(X, Y) = H(X) + H(Y) \Rightarrow I(X, Y) = 0$ .



# MI and image registration

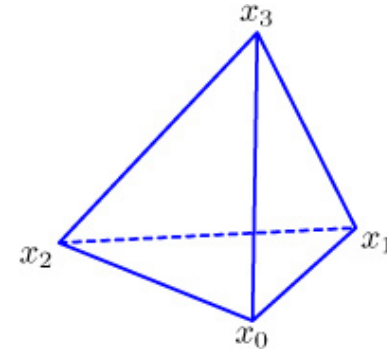
Identify RV  $X$  with image  $A$  and  $Y$  with image  $B$ . Consider only information contained in  $\Omega_{A,B}$ .  $I(A, B) = H(A) + H(B) - H(A, B)$ .



Maximizing MI: Find registrations with high marginal entropies and low joint entropy. MI is maximum if images  $A$  and  $B$  are **properly aligned**.

# The Downhill Simplex (or Nelder-Mead) Method

- A **simplex** is a simple geometric shape defined by the convex hull of  $n + 1$  vertices in  $n$ -dimensional space.



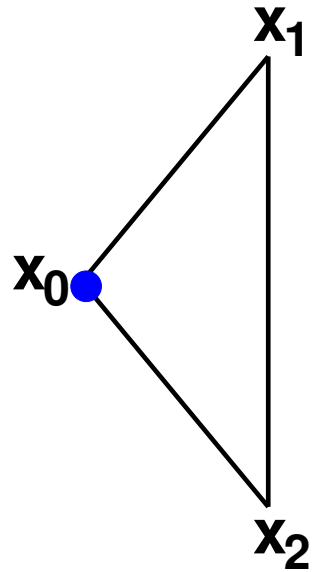
- 1D: **edge**; 2D: **triangle**; 3D: **tetrahedron**.
- If we're optimizing a function on  $n$  parameters, then we're searching in a  $n$ -dimensional parameter space, and our simplex has  $n + 1$  vertices.
- Calculate function values at simplex vertices
- Simplex “**crawls**”
  - Towards minimum
  - Away from maximum
- Probably the most widely used optimization method

# Simplex transformation algorithm

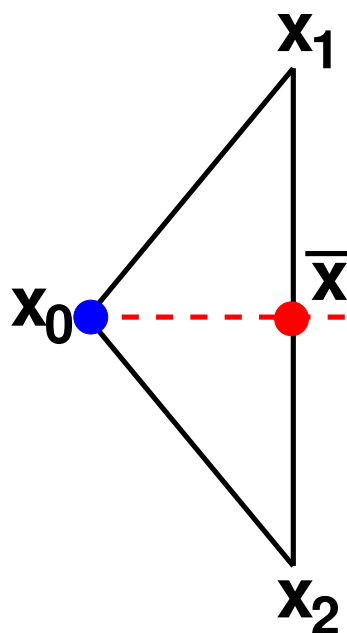
One iteration consists of the following three steps.

1. **Ordering:** Determine the indices  $\{0, 1, \dots, n - 1, n\}$ , of the worst, second worst, ..., best vertex in the current simplex  $S$ :  
$$f_0 = \max_j f_j > f_1 > \dots > f_n = \min_j f_j.$$
2. **Centroid:** Calculate the centroid  $\bar{\mathbf{x}}$  of the best side – the one opposite to the worst vertex  $\mathbf{x}_0$   
$$\bar{\mathbf{x}} := \frac{1}{n} \sum_{j=1}^n \mathbf{x}_j.$$
3. **Transformation:** Compute the new working simplex from the current one. First, try to replace only the worst vertex  $\mathbf{x}_0$  with a better point by using **reflection**, **expansion** or **contraction** with respect to the best side.

# A Simplex in Two Dimensions

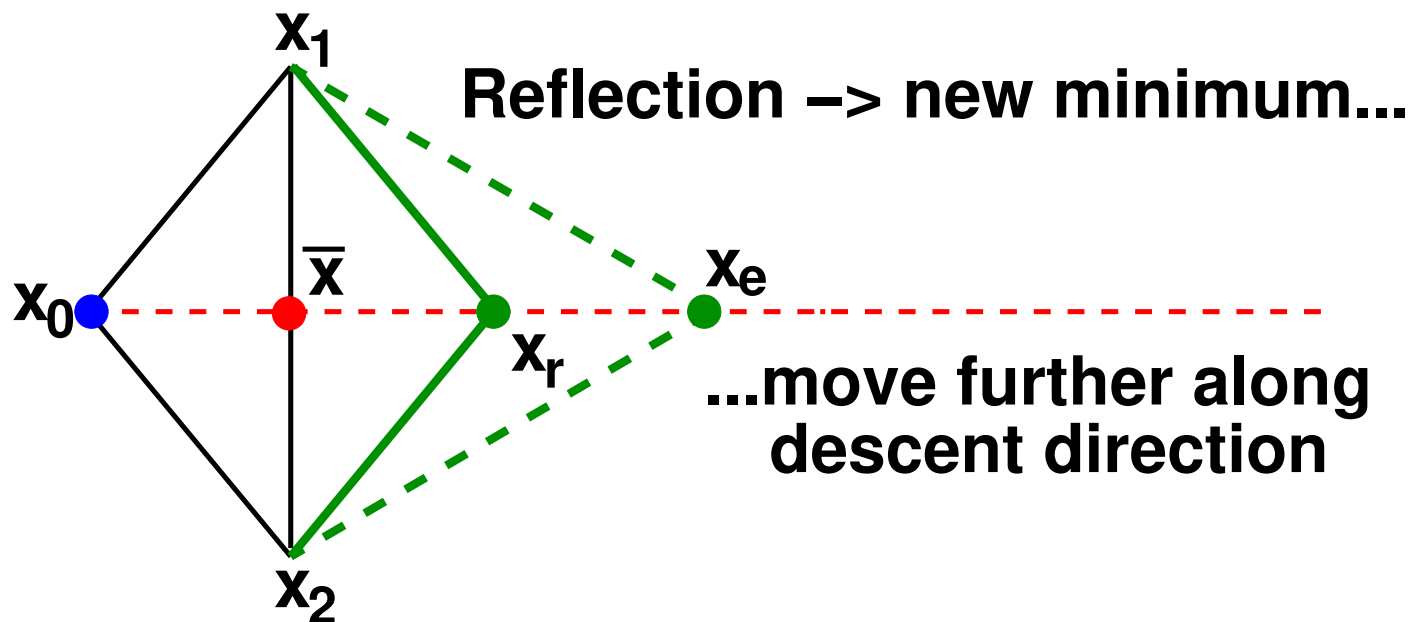
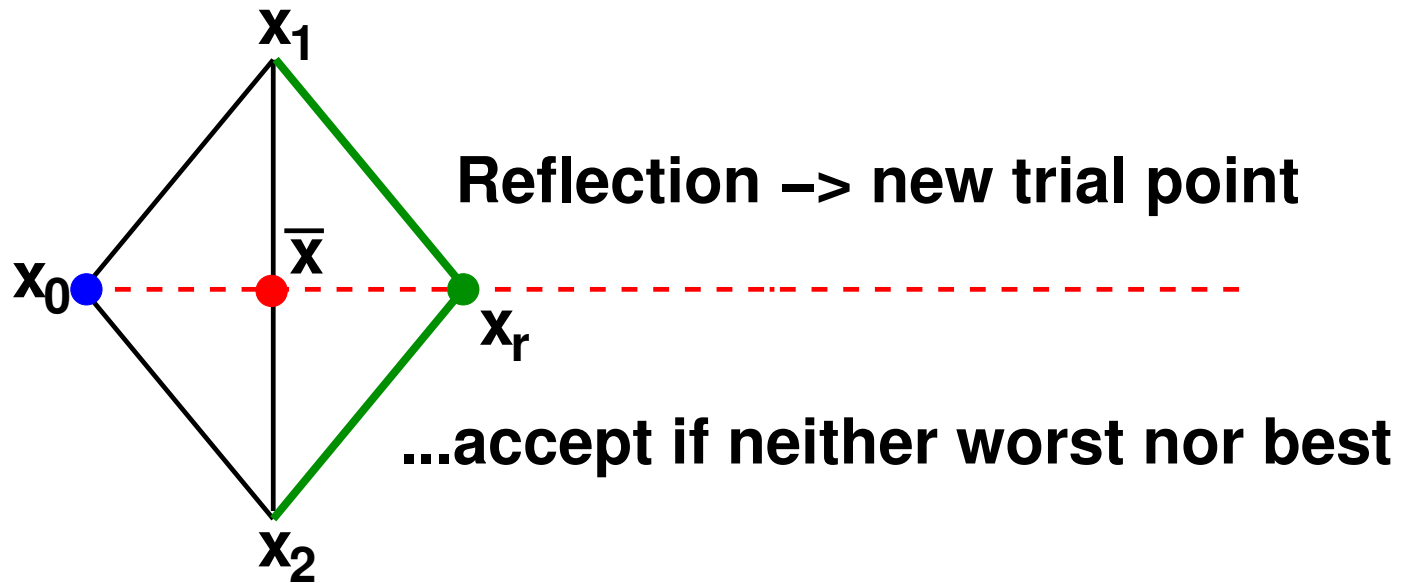


- Evaluate function at vertices
- Highest (worst) point:  $x_0$   
Next highest point:  $x_1$   
Lowest (best) point:  $x_2$
- Intuition: move away from high point, towards low point.

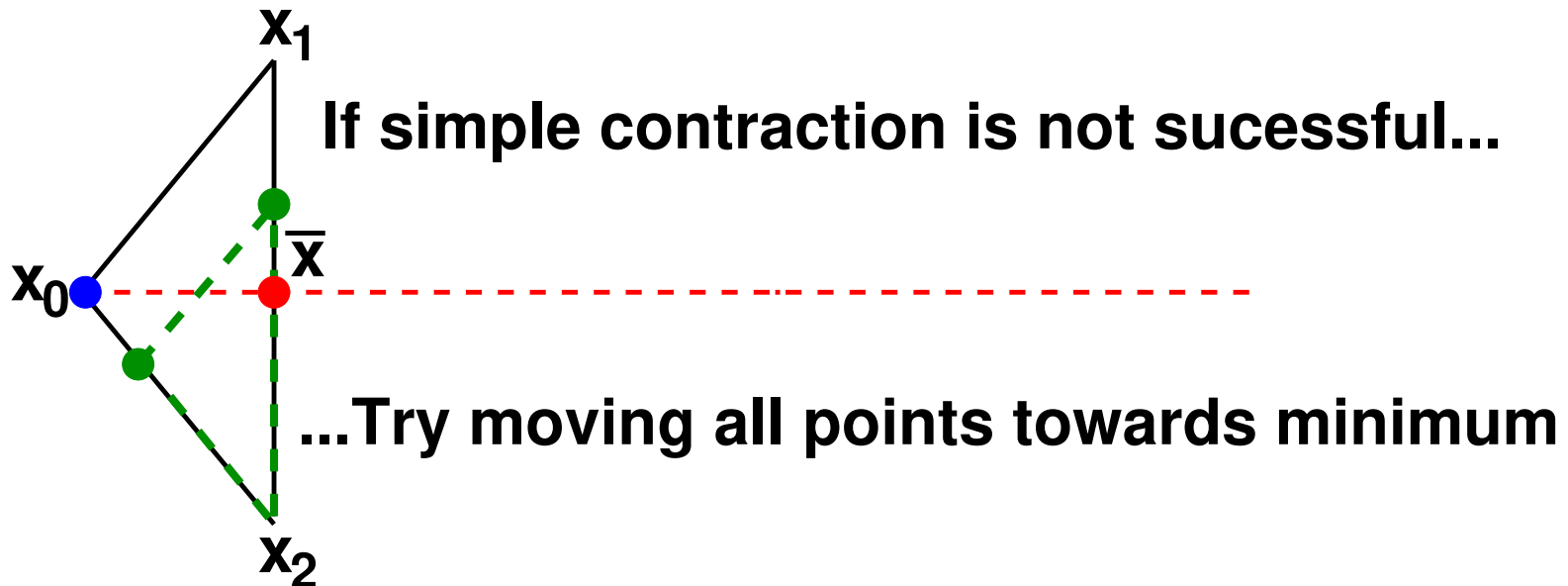
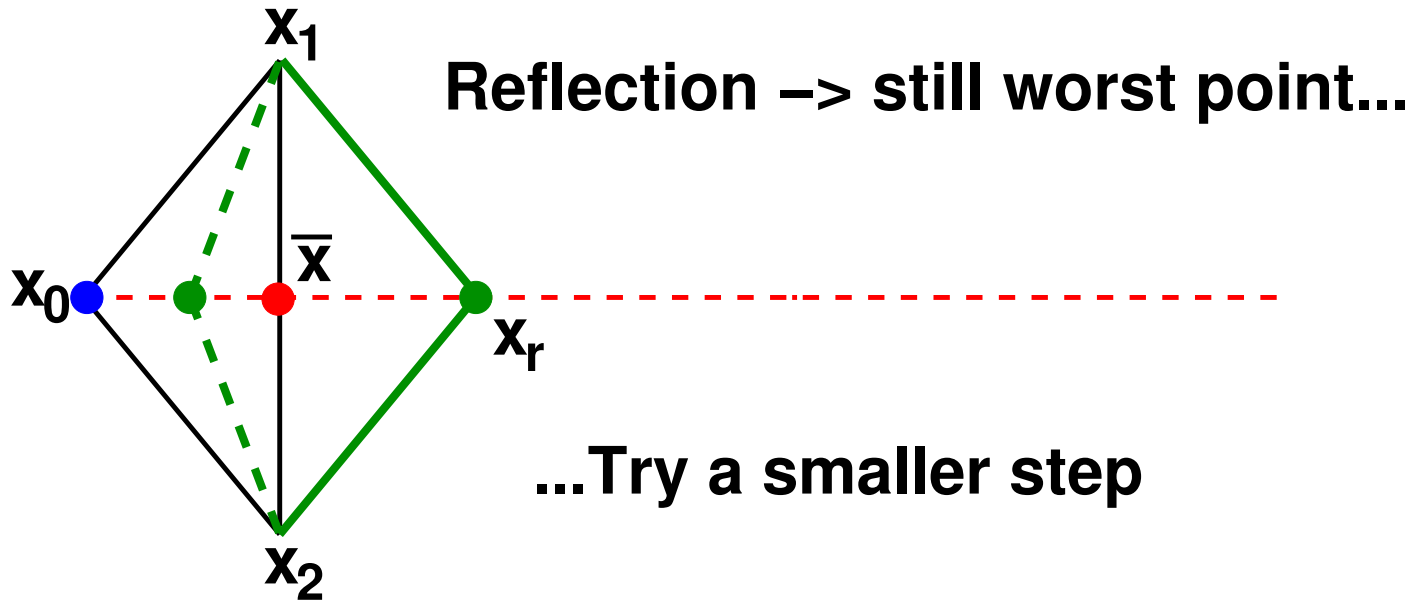


**Line through worst point ●  
and average of other points ●**

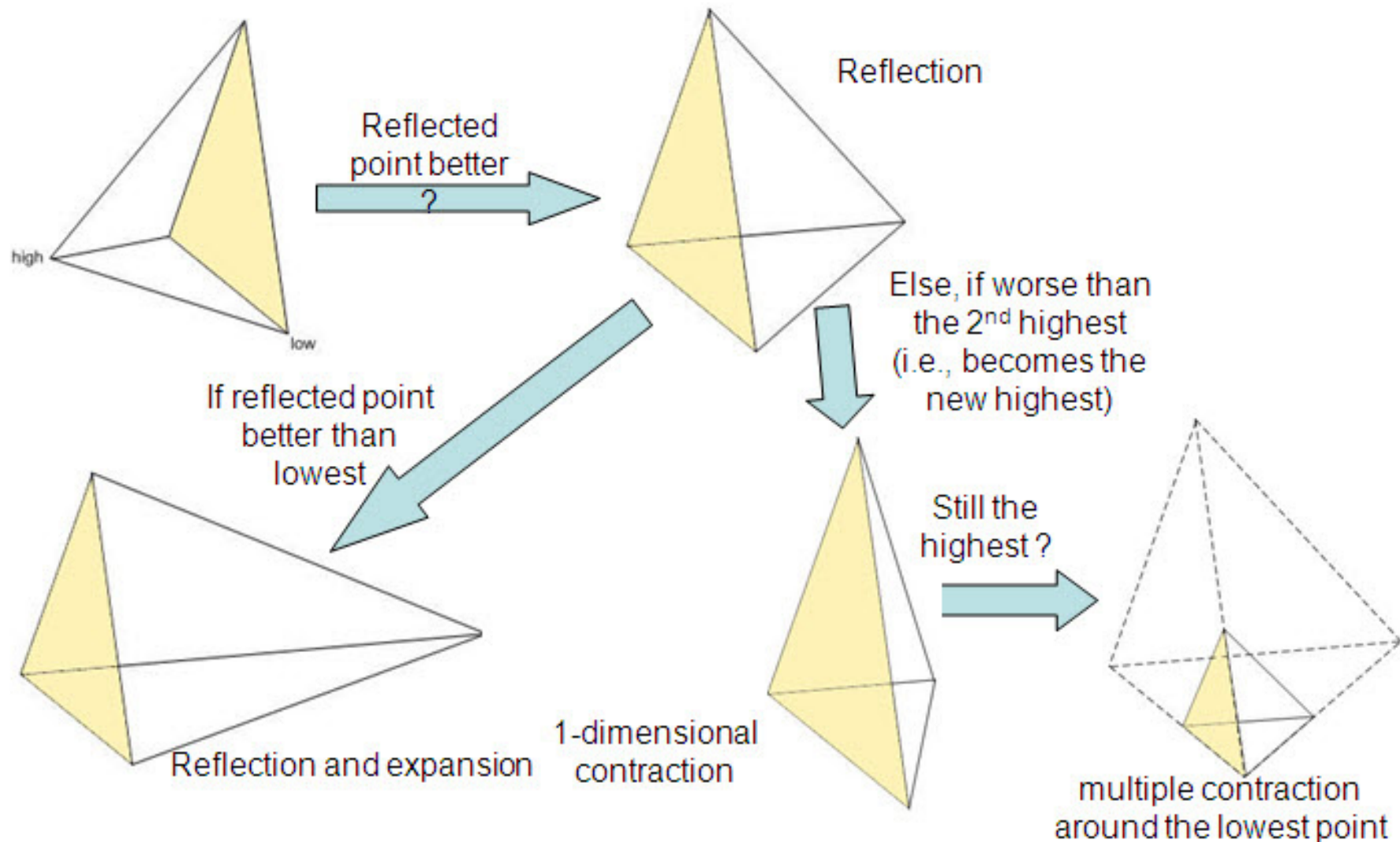
# Reflection and Expansion



# Contraction



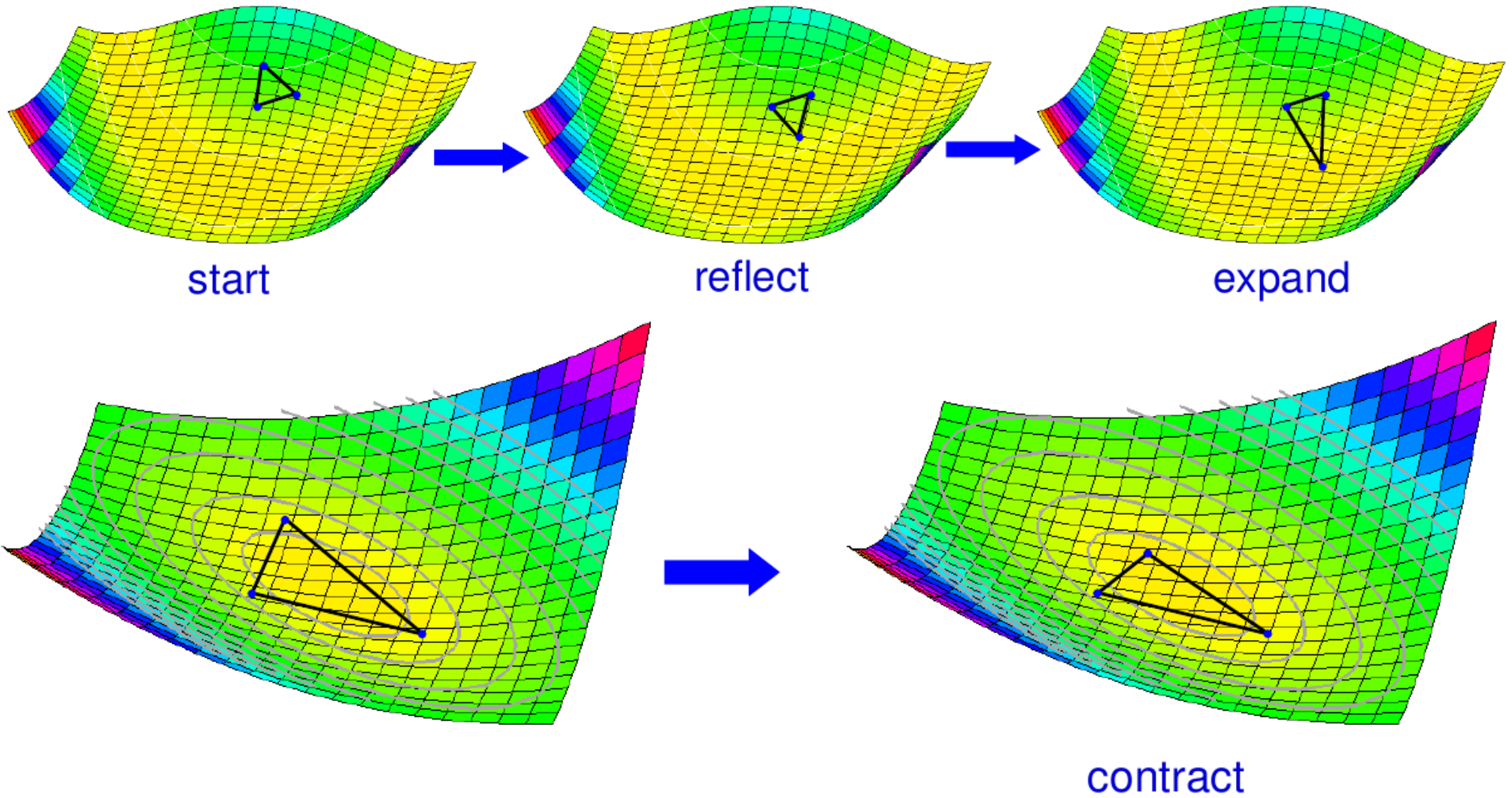
# Summary: The Downhill Simplex Method



Karimzadehgan, Maryam et al. (2011). A stochastic learning-to-rank algorithm and its application to contextual advertising.

10.1145/1963405.1963460.

# Summary: The Downhill Simplex Method

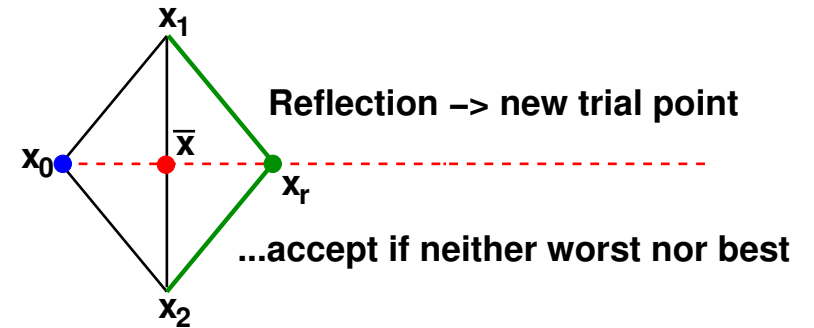


# Algorithm

- **Reorder points:**  $f(\mathbf{x}_0) > f(\mathbf{x}_1) > \dots > f(\mathbf{x}_n)$  ( $\mathbf{x}_0$  is worst point).

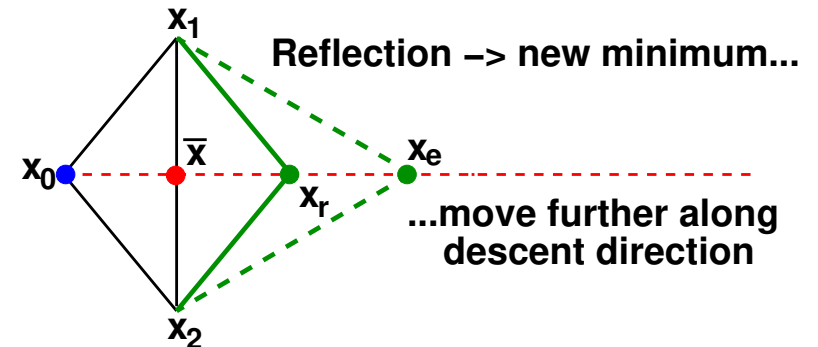
- **New trial point  $\mathbf{x}_r$  by reflection**

$$\mathbf{x}_r = \bar{\mathbf{x}} + \alpha(\bar{\mathbf{x}} - \mathbf{x}_0), \quad \bar{\mathbf{x}} := \frac{1}{n} \sum_{j \neq 0} \mathbf{x}_j.$$



Compute  $f(\mathbf{x}_r)$ , **3 possibilities:**

1.  $f(\mathbf{x}_n) < f(\mathbf{x}_r) < f(\mathbf{x}_0)$ ,  
**replace  $\mathbf{x}_0$  by  $\mathbf{x}_r$ .**
2.  $f(\mathbf{x}_r) < f(\mathbf{x}_2)$   
 $\rightsquigarrow$  direction of **reflection is good**  
 $\rightsquigarrow$  **expansion**  $\mathbf{x}_e = \mathbf{x}_r + \beta(\mathbf{x}_r - \bar{\mathbf{x}})$ .  
 If  $f(\mathbf{x}_e) < f(\mathbf{x}_r)$ , replace  $\mathbf{x}_0$  by  $\mathbf{x}_e$ .  
 Otherwise, **expansion has failed**,  
 replace  $\mathbf{x}_0$  by  $\mathbf{x}_r$ .



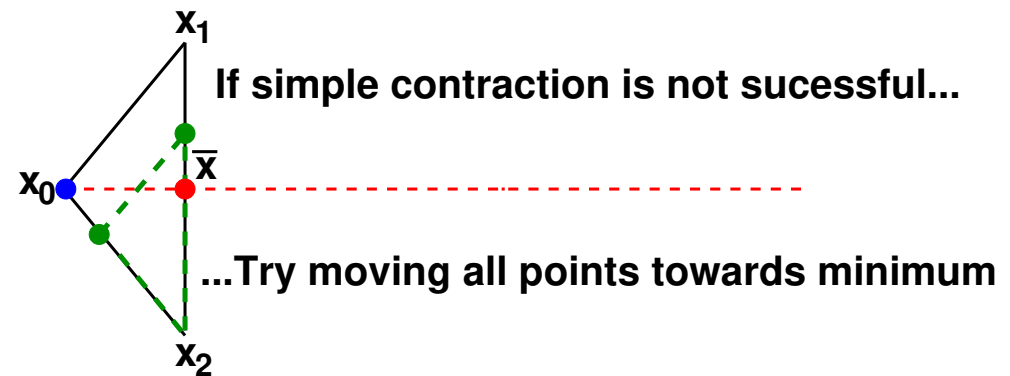
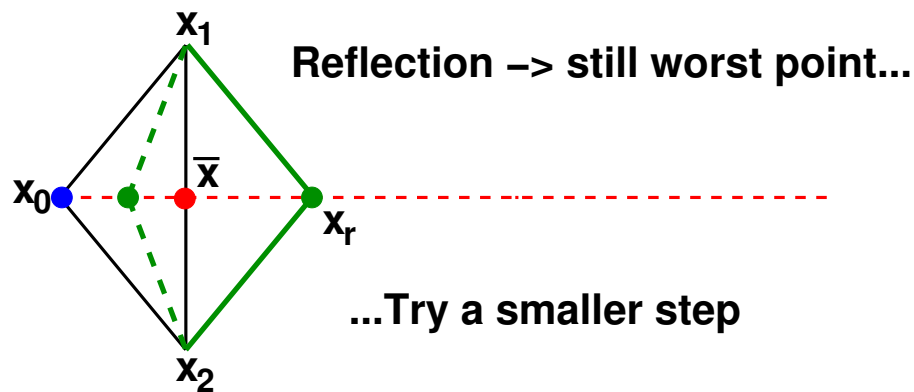
# Algorithm

3.  $f(\mathbf{x}_r) > f(\mathbf{x}_0)$

$\rightsquigarrow$  **polytope is too large**

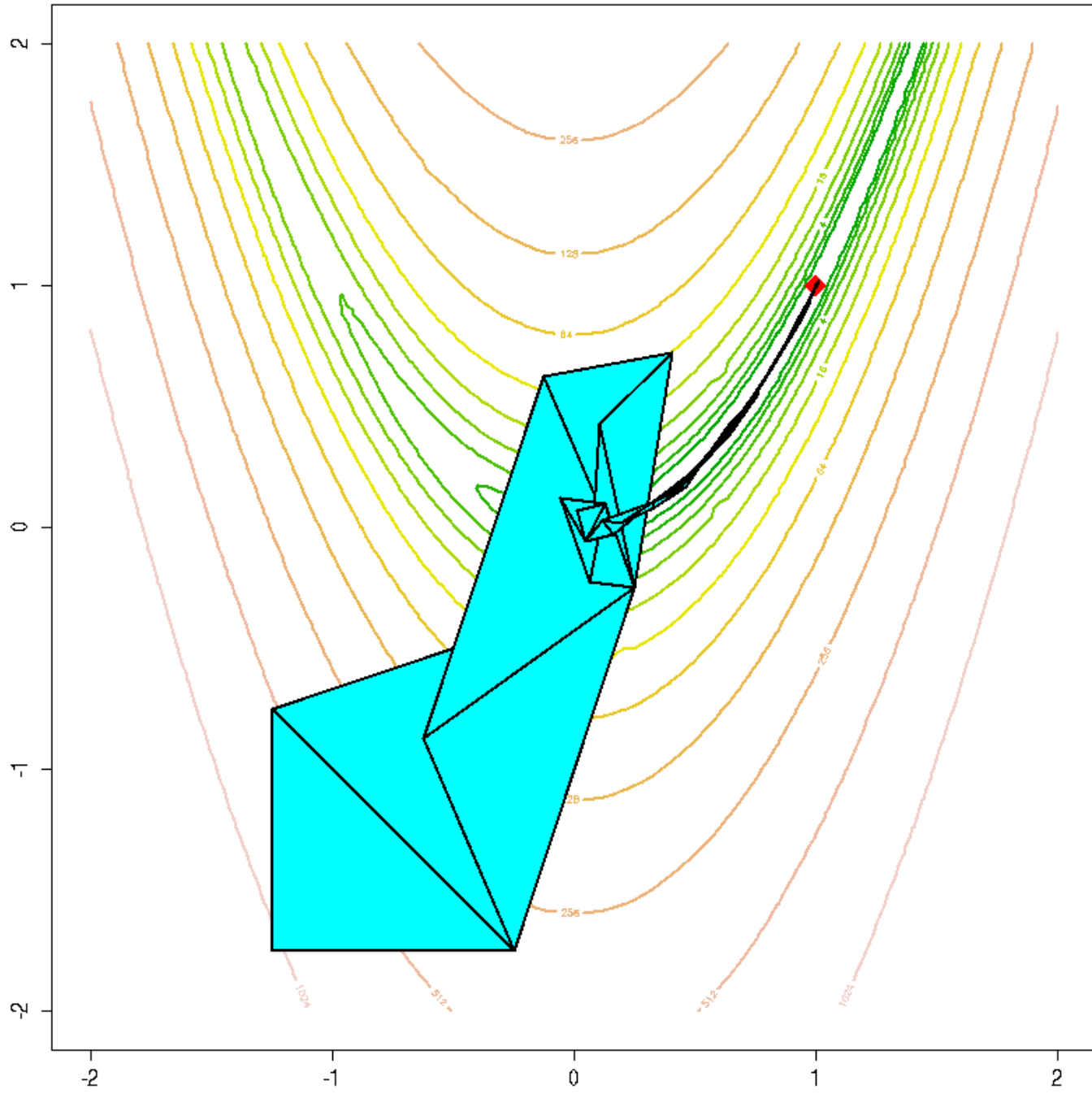
$\rightsquigarrow$  **contraction**  $\mathbf{x}_c = \bar{\mathbf{x}} + \gamma(\mathbf{x}_0 - \bar{\mathbf{x}})$  where  $(0 < \gamma < 1)$ .

If  $f(\mathbf{x}_c) < f(\mathbf{x}_0)$   $\rightsquigarrow$  **contraction succeeded**  $\rightsquigarrow$  replace  $\mathbf{x}_0$  by  $\mathbf{x}_c$ , otherwise **contract again**.

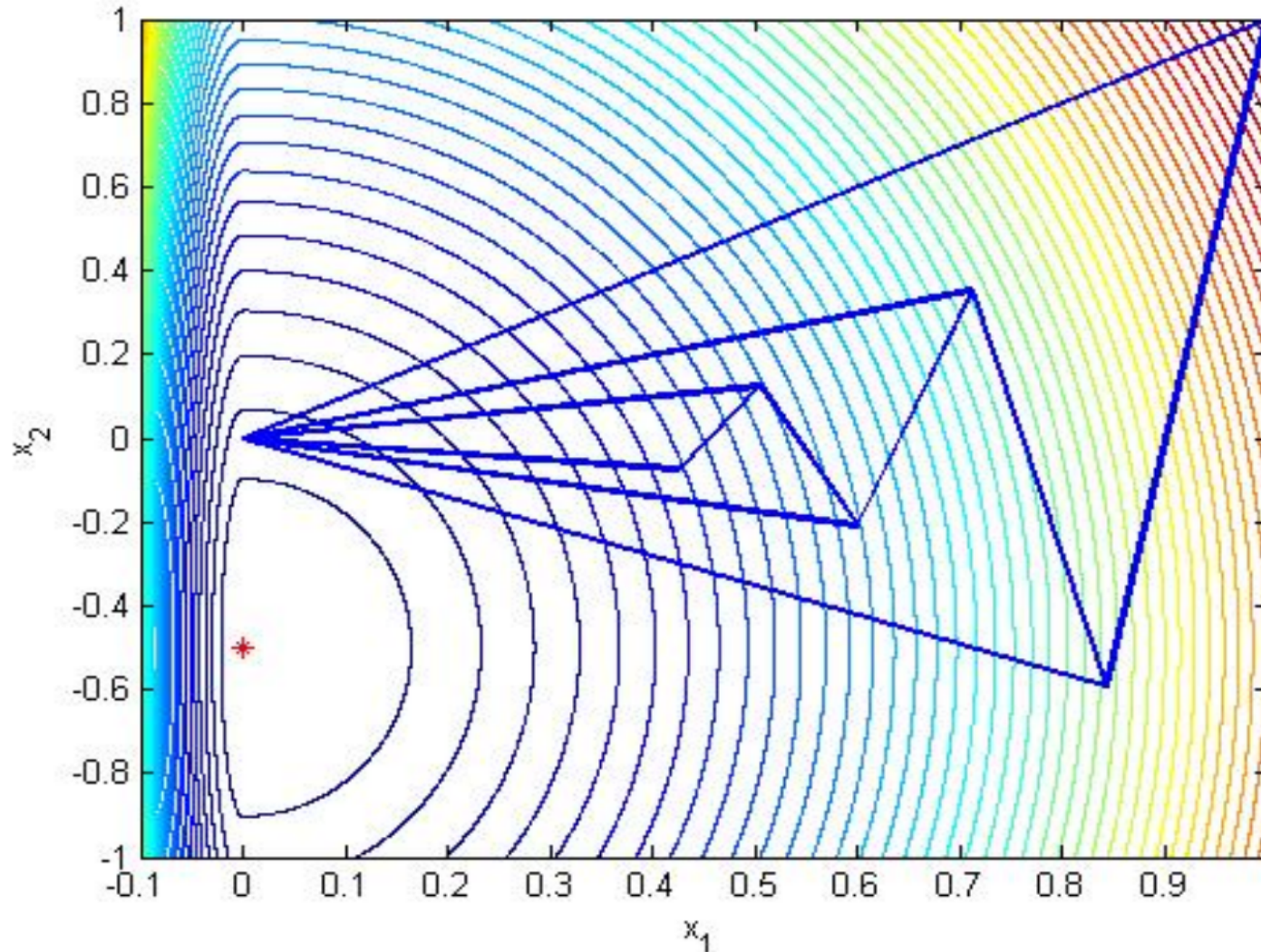


If simple contraction is not successful, replace all points except the best with  $\mathbf{x}_i = \mathbf{x}_n + \sigma(\mathbf{x}_i - \mathbf{x}_n)$ ,  $0 < \sigma < 1$ .





# A Counter Example due to McKinnon



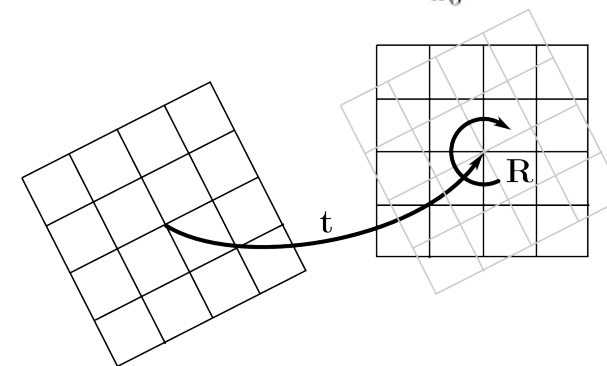
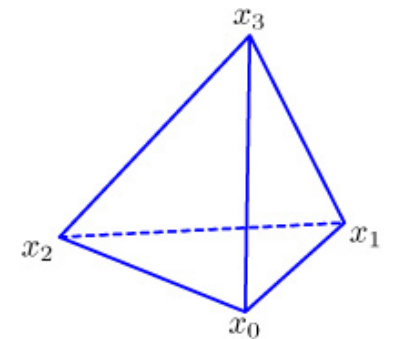
McKinnon, K.I.M., Convergence of the Nelder-Mead simplex method to a non-stationary point, SIAM Journal on Optimization 9 (1998), 148-158.

# Back to image registration

Use Downhill-Simplex for minimizing the negative mutual information over rigid transformations  $(\theta, t_x, t_y)$

$\rightsquigarrow$  4-dim simplex (tetrahedron)

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

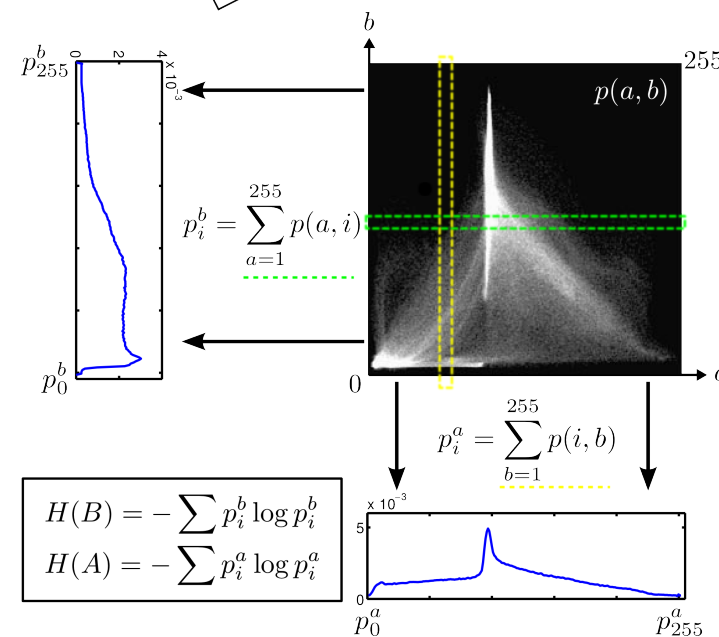


Function that computes  $MI(\theta, t_x, t_y)$ :

Transformation  $\rightsquigarrow$  interpolation

$\rightsquigarrow$  joint histogram on  $\Omega$

$\rightsquigarrow$  marginal and joint entropies  $\rightsquigarrow$  MI



$$H(B) = -\sum p_i^b \log p_i^b$$

$$H(A) = -\sum p_i^a \log p_i^a$$