Lecture 2: SVD, Pseudo-Inverse, and Projection

Mark Hasegawa-Johnson

University of Illinois

ECE 417: Multimedia Signal Processing, Fall 2023

Application: Animating a still image

Singular Value Decomposition

Pseudo-Inverse

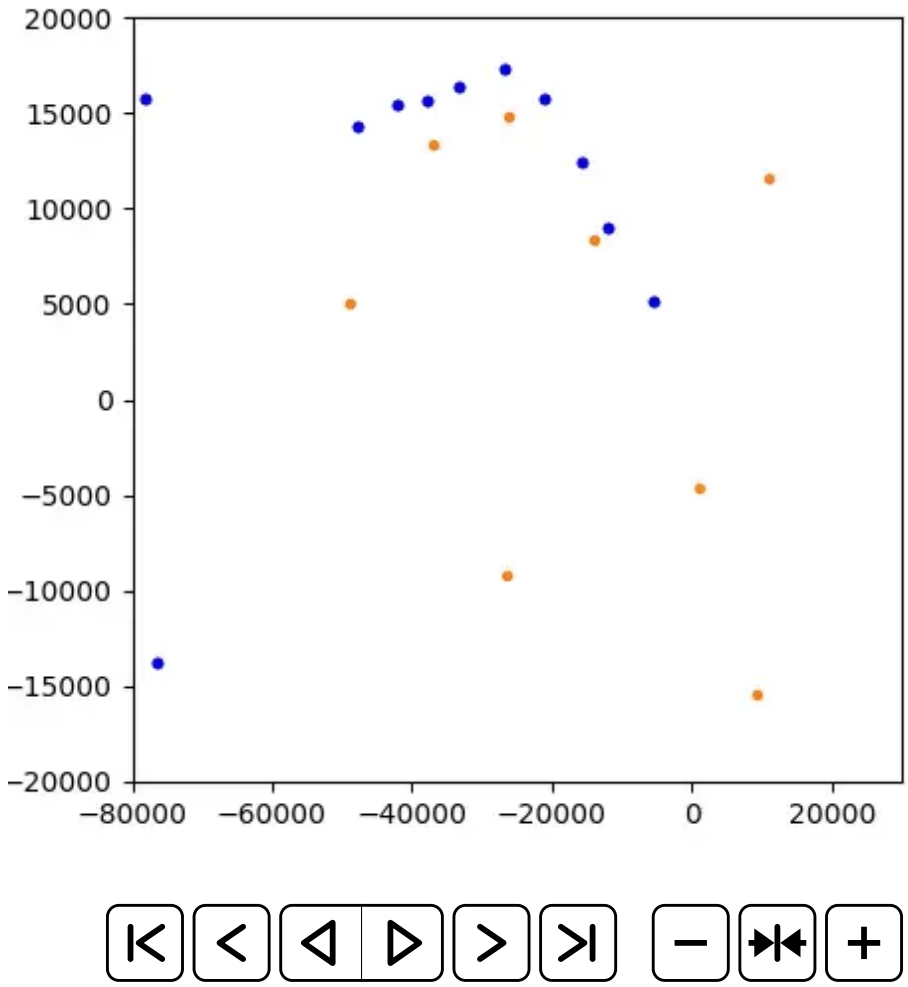
Affine Transformations

MMSE Estimation of an Affine Transform

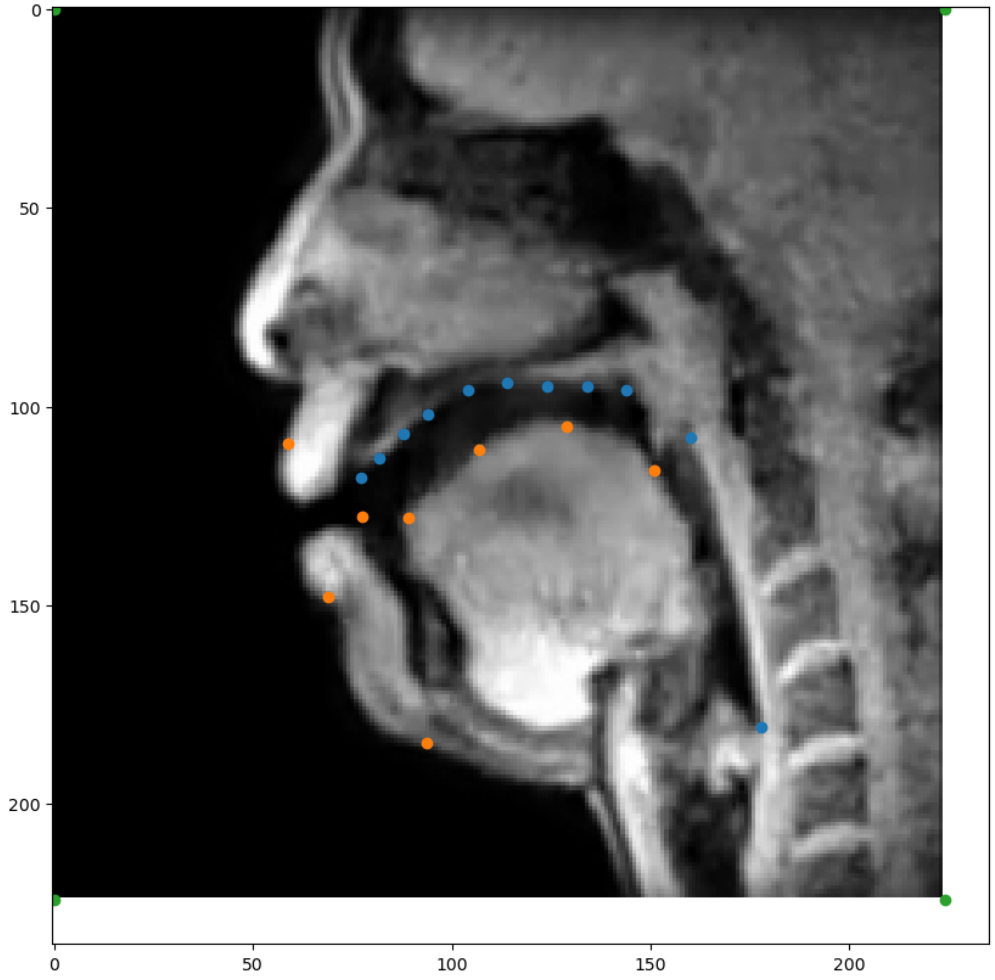
Conclusions

Outline

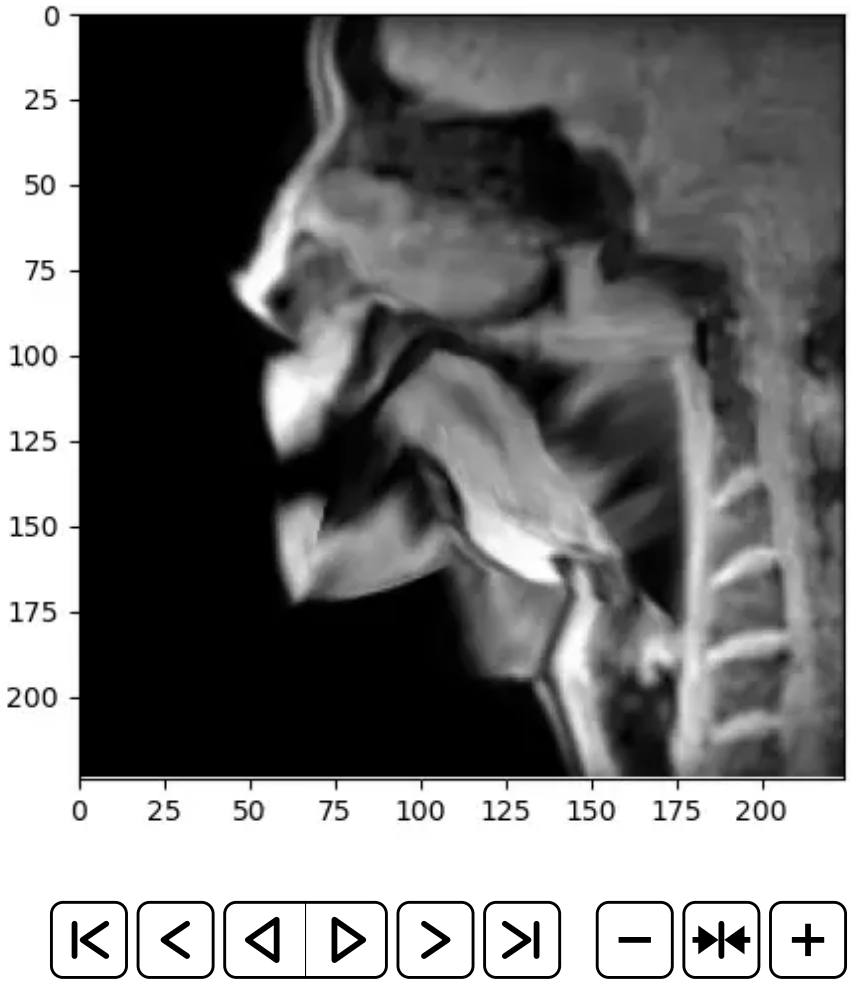
Input #1: X-Ray Microbeam Data



Input #2: A Still Image Acquired Using MRI



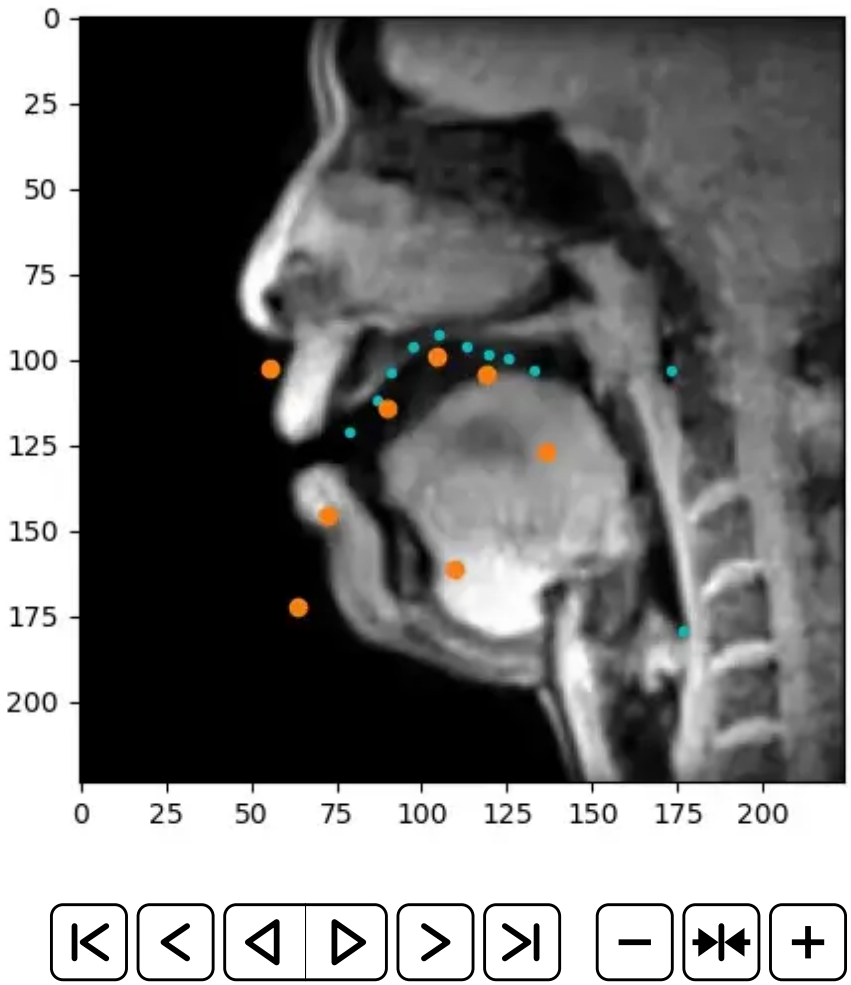
Desired Output: Animated Image



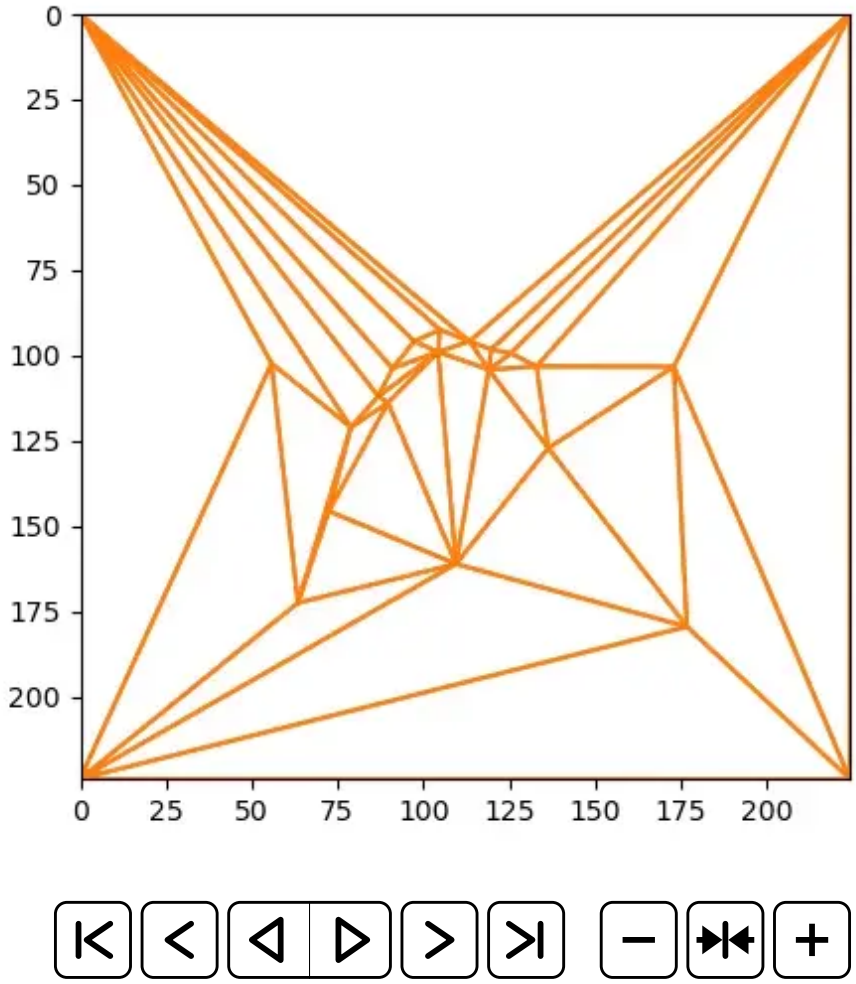
Strategy

1. Use affine projection to rotate, scale, and shear the XRMB points so that they match the shape of the MRI as well as possible.
2. Draw triangles on the MRI so that every pixel is inside a triangle.
3. Move the triangles.

Step 1 (Today): Use affine projection to map XRMB to MRI



Step 2 (Next time): Draw triangles on the MRI



Outline

Summary: Purpose of this section

Any real-valued matrix of any dimension, $M \in \Re^{m \times n}$, can be written as

$$M = U \Sigma V^T$$

. . . where Σ is a diagonal matrix, and U and V are orthonormal matrices.

Proof Step #1: Eigenvectors of a Gram Matrix

The gram matrix, $G = M^T M \in \Re^{n \times n}$, is the matrix whose elements are inner products between the **columns** of M. G is square, symmetric, and positive-semidefinite, therefore $G = V \Lambda V^T$, where

$$V = [v_1, \ldots, v_n], \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

- The rows and columns of V are orthonormal, $VV^T = V^T V = I$
- $\lambda_i \geq 0$ are the eigenvalues of G

Proof Step #2: Singular Values

The eigenvalues are non-negative, so we can take their square roots:

$$\Sigma = \Lambda^{1/2}$$
$$= \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma_n \end{bmatrix}$$

where $\sigma_i = \sqrt{\lambda_i}$. Therefore, we can write

$$G = V\Lambda V^T = V\Sigma\Sigma V^T$$

Proof Step #3: the Sum-of-Squares Matrix

The sum-of-squares matrix, $S = MM^T \in \Re^{m \times m}$, is the matrix whose elements are inner products between the **rows** of M. S is square, symmetric, and positive-semidefinite, therefore $S = U \Lambda U^T$, where

$$U = [u_1, \ldots, u_m], \quad \Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \lambda_m \end{bmatrix}$$

- The eigenvalues of S are the same as the eigenvalues of G, except that if $m > n$, the extra eigenvalues are all zero: $\lambda_{n+1} = \cdots = \lambda_m = 0$.
- The rows and columns of U are orthonormal, $UU^T = U^T U = I$

Proof Step #4: Replace I by $U^T U$

Starting with $G = VSSV^T$, we can introduce an identity matrix into the middle to get

$$G = V\Sigma I\Sigma V^T = V\Sigma U^T U\Sigma V^T$$

But by the same logic, we can write

$$S = U\Sigma I\Sigma U^T = U\Sigma V^T V\Sigma U^T$$

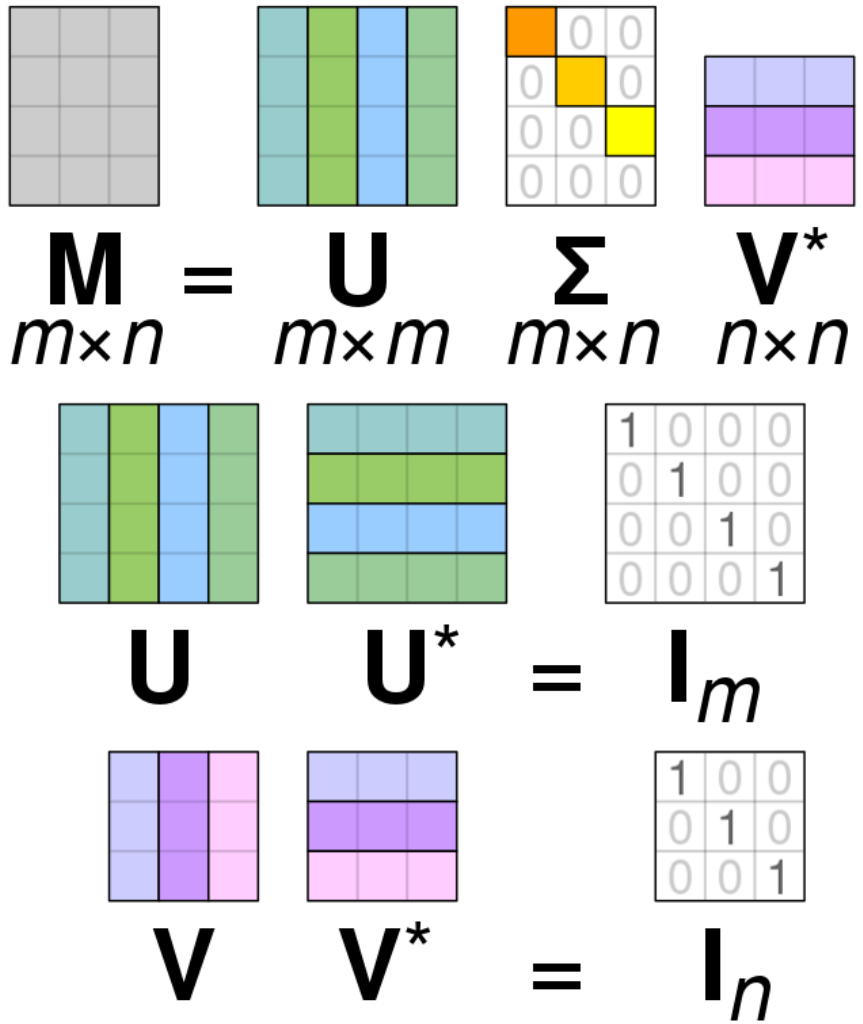
The only way these can both be true is if the original matrix was

$$M = U\Sigma V^T$$

Singular Value Decomposition

Any real matrix $M \in \Re^{m \times n}$ can be decomposed as

$$M = U \Sigma V^T$$



CC-SA 4.0, Cmglee

Outline

Summary: Purpose of this section

Any arbitrary real matrix $M = U\Sigma V^T \in \Re^{m \times n}$ has a pseudo-inverse, defined as

$$M^\dagger = U_r \Sigma_r^{-1} V_r^T$$

where U_r, Σ_r, and V_r are the "compact singular value decomposition" of M, defined as:

- $\Sigma_r \in \Re^{r \times r}$ contains only the nonzero singular values,
- $U_r \in \Re^{m \times r}$ contains the corresponding columns of U,
- $V_r \in \Re^{n \times r}$ contains the corresponding columns of V.

Why it makes sense: Eigenvalue definition of matrix inverse

First, suppose that G is a square positive-definite matrix, i.e., all of its eigenvalues are greater than zero. Then we can write G-inverse as

$$G^{-1} = V\Lambda^{-1}V^T$$

Proof:

$$\begin{aligned} G^{-1}G &= \left(V\Lambda^{-1}V^T\right)\left(V\Lambda V^T\right) \\ &= V\Lambda^{-1}\Lambda V^T \\ &= VV^T \\ &= I \end{aligned}$$

Algebraic Forms

- If $m < n$ and M has full row rank (all rows linearly independent), then $MM^T \in \Re^{m \times m}$ is invertible, and

$$M^\dagger = M^T (MM^T)^{-1}$$

- If $m > n$ and M has full column rank (all columns linearly independent), then $M^T M \in \Re^{n \times n}$ is invertible, and

$$M^\dagger = (M^T M)^{-1} M^T$$

The homework will explore the properties of these two very useful special cases.

Proof that the Algebraic Forms are Pseudo-Inverse

Suppose that $M = U_r \Sigma_r V_r^T$, and $M^\dagger = U_r \Sigma_r^{-1} V_r^T$. Then

$$\begin{aligned}
M^T (M M^T)^{-1} &= V_r \Sigma_r U_r^T (U_r \Sigma_r V_r^T V_r \Sigma_r U_r^T)^{-1} \\
&= V_r \Sigma_r U_r^T (U_r \Sigma_r \Sigma_r U_r^T)^{-1} \\
&= V_r \Sigma_r U_r^T (U_r \Lambda_r U_r^T)^{-1} \\
&= V_r \Sigma_r U_r^T U_r \Lambda_r^{-1} U_r^T \\
&= V_r \Sigma_r \Lambda_r^{-1} U_r^T \\
&= V_r \Sigma_r^{-1} U_r^T \\
&= M^\dagger
\end{aligned}$$

Outline

Affine Transformation

Given an input pixel location $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \Re^2$, the goal is to rotate, scale, shift and shear the image to find an output pixel location $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \in \Re^2$.

An **affine transform** is a linear transform plus a shift:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

. . . or we could just write. . .

$$u = Ax + b$$

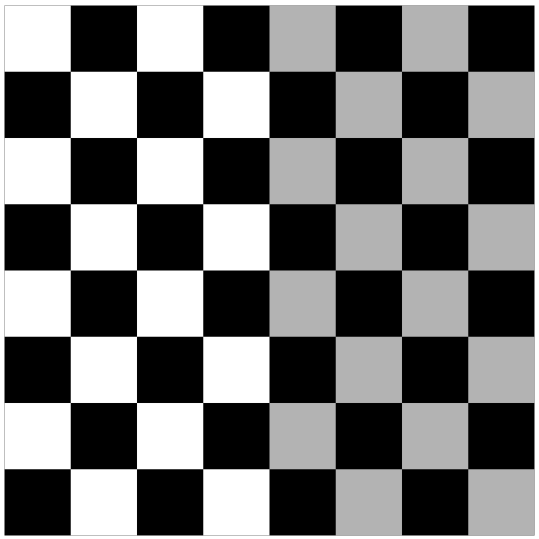
Affine Transforms

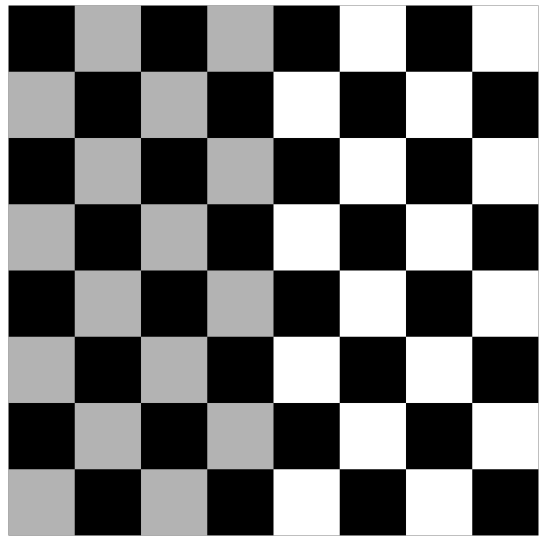
Notice that the affine transformation has 6 degrees of freedom: $(a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}, b_1, b_2)$. Therefore, you can accmplish 6 different types of transformation:

- Shift the image left$\leftrightarrow$right (using b_1)
- Shift the image up$\leftrightarrow$down (using b_2)
- Scale the image horizontally (using $a_{1,1}$)
- Scale the image vertically (using $a_{2,2}$)
- Rotate the image (using $a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}$)
- Shear the image horizontally (using $a_{1,2}$)

Vertical shear (using $a_{2,1}$) is a combination of horizontal shear + rotation.

Example: Reflection

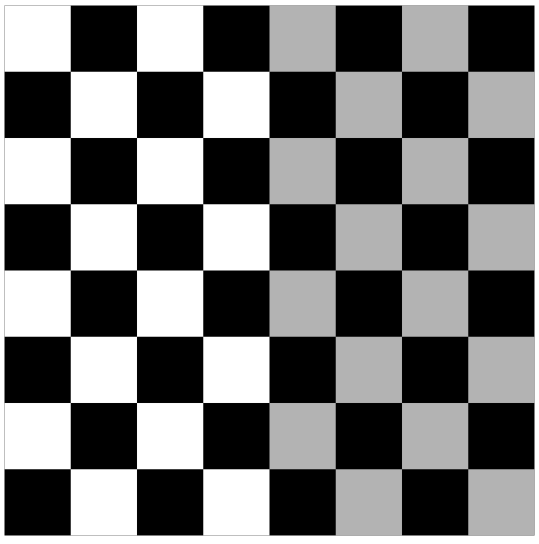


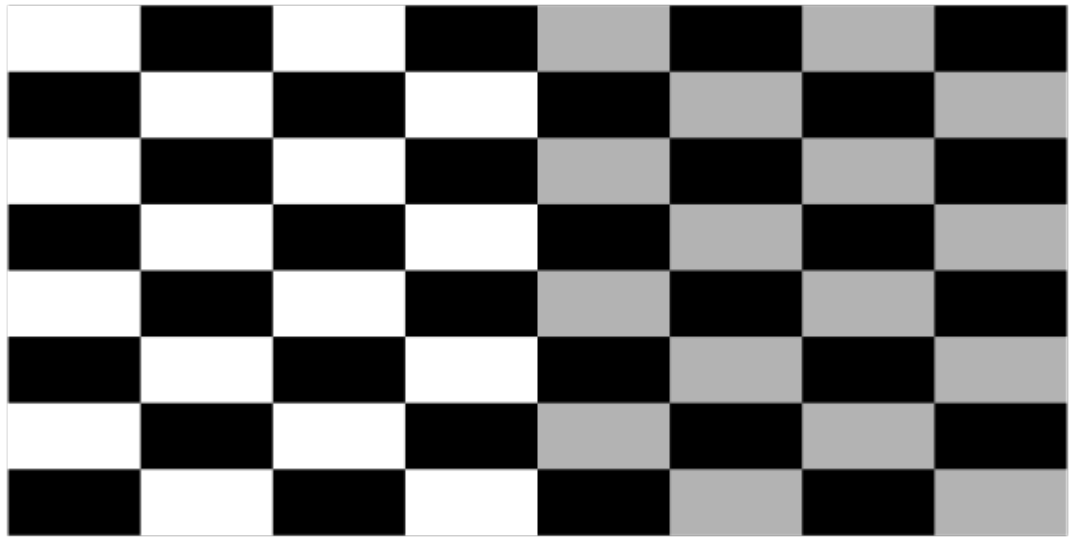


CC-SA 4.0, `https://en.wikipedia.org/wiki/Affine_transformation`

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example: Scale

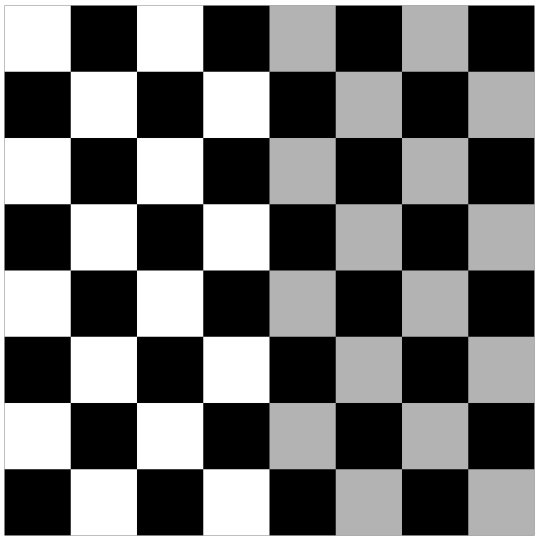


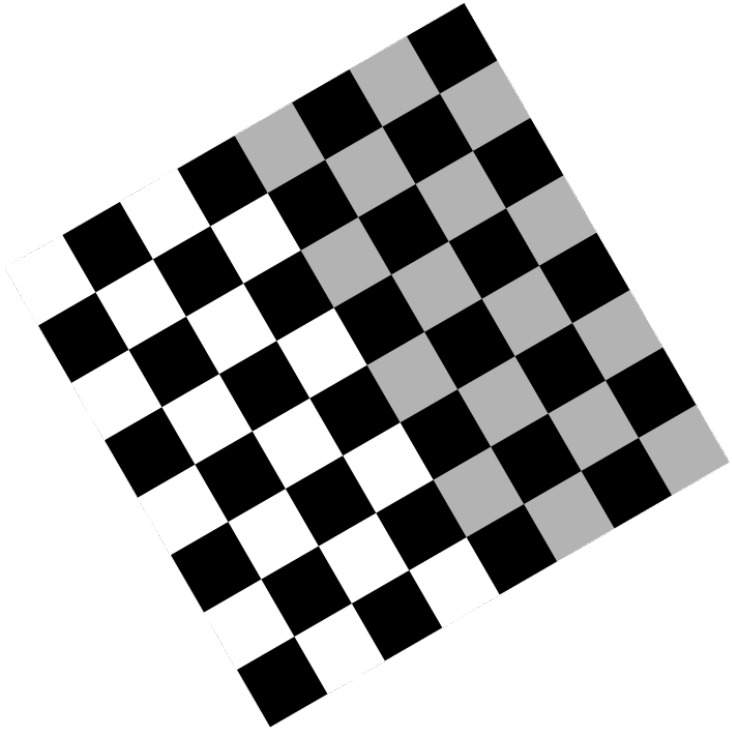


CC-SA 4.0, https://en.wikipedia.org/wiki/Affine_transformation

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example: Rotation

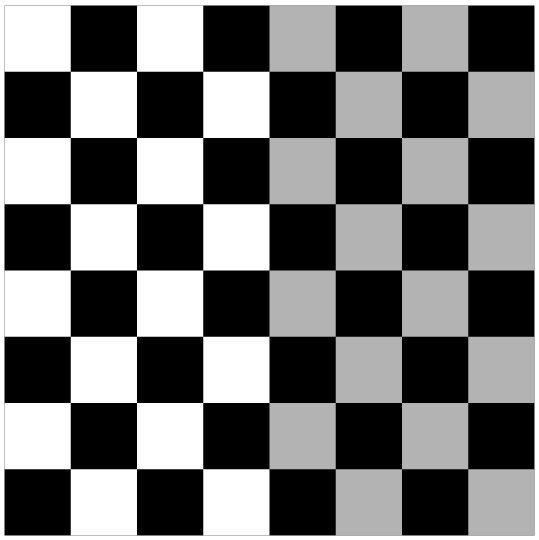


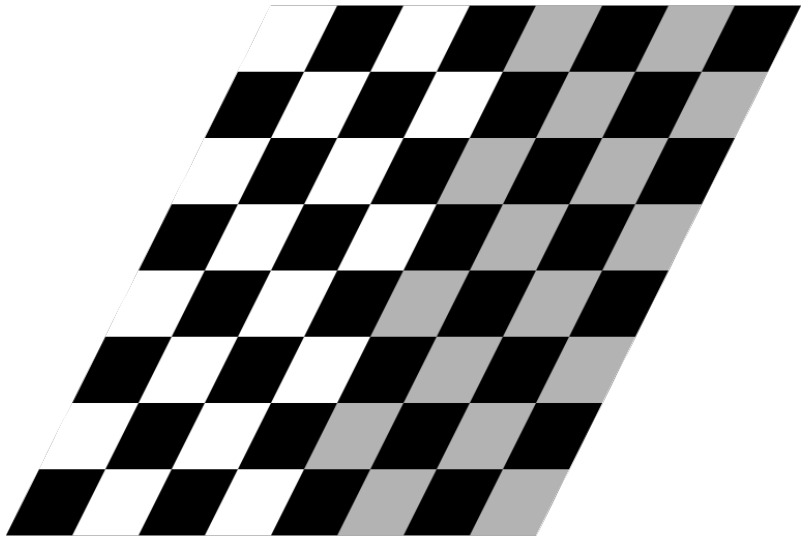


CC-SA 4.0, `https://en.wikipedia.org/wiki/Affine_transformation`

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example: Shear





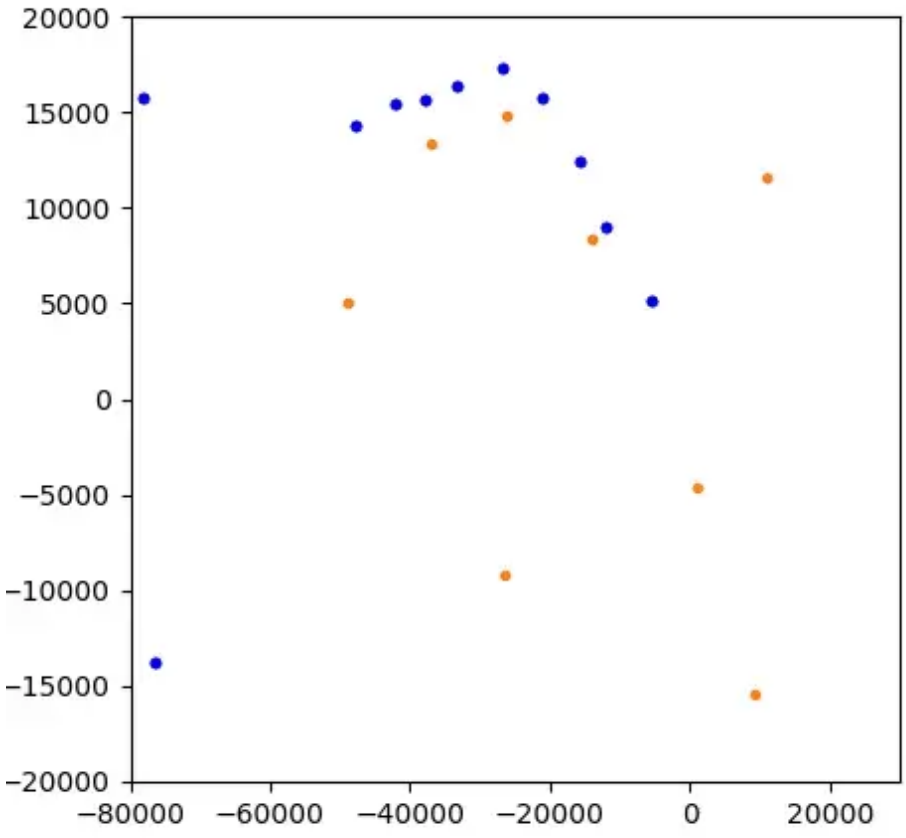
CC-SA 4.0, https://en.wikipedia.org/wiki/Affine_transformation

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Outline

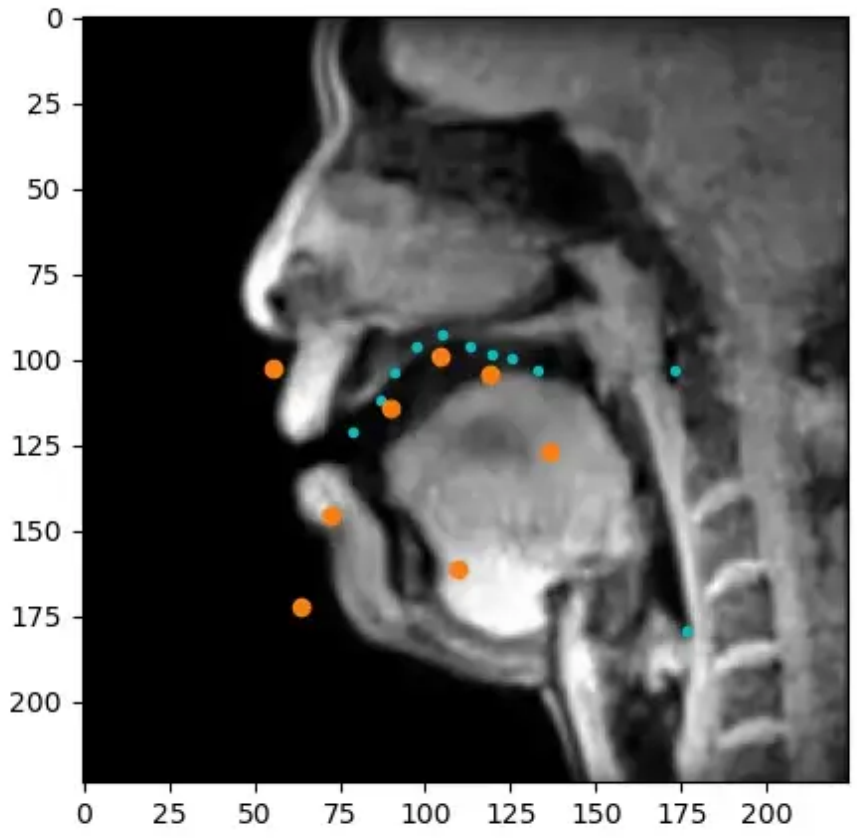
Input

Suppose we have a set of points like this:



Output

...we want to rotate, reflect, shift and scale so it looks like this:



Estimating an Affine Transform

Our goal is to find a 3×2 affine transform matrix, A, such that

$$Y \approx XA$$

where $x_i = [x_{i,1}, x_{i,2}, 1]$ is a 1-augmented X-ray microbeam landmark point, $y_i = [y_{i,1}, y_{i,2}]$ is the corresponding MRI landmark point, and

$$Y = \begin{bmatrix} y_{1,1} & y_{2,1} \\ \vdots & \vdots \\ y_{n,1} & y_{n,1} \end{bmatrix}, \quad X = \begin{bmatrix} x_{1,1} & x_{2,1} & 1 \\ \vdots & \vdots & \vdots \\ x_{n,1} & x_{n,1} & 1 \end{bmatrix}, \quad A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ b_1 & b_2 \end{bmatrix}$$

MMSE Estimation of Affine Transform

In particular, suppose we want to minimize the mean-squared error:

$$\epsilon = \sum_{i=1}^{n} \|x_i A - y_i\|^2 = \sum_{i=1}^{n} (x_i A - y_i)(x_i A - y_i)^T$$

Notice that this is just squaring the samples of $XA - Y$, and adding across the rows. That's called the "Frobenius norm" of $XA - Y$, and there a few different ways it can be expanded:

$$\begin{aligned}
\epsilon &= \|XA - Y\|_F^2 \\
&= \text{trace}\left((XA - Y)(XA - Y)^T\right) \\
&= \text{trace}\left((XA - Y)^T(XA - Y)\right)
\end{aligned}$$

MMSE Estimation of Affine Transform

Since MMSE is quadratic, we can find the minimum by just differentiating with respect to the elements of the matrix. Differentiating is easy if we choose the version of ϵ, from the previous slide, that puts A on the outside:

$$\begin{aligned}\frac{d\epsilon}{dA} &\equiv \begin{bmatrix} \frac{d\epsilon}{da_{1,1}} & \frac{d\epsilon}{da_{1,2}} \\ \frac{d\epsilon}{da_{2,1}} & \frac{d\epsilon}{da_{2,2}} \\ \frac{d\epsilon}{db_1} & \frac{d\epsilon}{db_2} \end{bmatrix} \\ &= \frac{d}{dA}\text{trace}\left(A^T X^T X A - Y^T X A - A^T X^T Y + Y^T Y\right) \\ &= A^T X^T X + (X^T X A)^T - (X^T Y)^T - Y^T X\end{aligned}$$

Setting that to zero and re-arranging gives $X^T X A = X^T Y$, and therefore

$$A_{\text{MMSE}} = (X^T X)^{-1} X^T Y = X^{\dagger} Y$$

MMSE Estimation of Affine Transform

In case you don't like matrix differentiations, here is the derivative from the previous slide done element-by-element:

$$\epsilon = \|XA - Y\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{2}\left(\sum_{k=1}^{3} x_{i,k}a_{k,j} - y_{i,j}\right)^2$$

$$\frac{\partial \epsilon}{\partial a_{p,q}} = 2\sum_{i=1}^{m}\sum_{j=1}^{2}\left(\sum_{k=1}^{3} x_{i,k}a_{k,j} - y_{i,j}\right)\left(\frac{\partial \sum_{k=1}^{3} x_{i,k}a_{k,j}}{\partial a_{p,q}}\right)$$

$$= 2\sum_{i=1}^{m}\left(\sum_{k=1}^{3} x_{i,k}a_{k,q} - y_{i,q}\right)x_{i,p}$$

If we stack up the last line into a matrix where p is the row number and q is the column number, we get

$$\frac{d\epsilon}{dA} = 2X^T XA - 2X^T Y$$

Interpretation of MMSE as Projection

Pay attention to what's happening here.
$A_{\text{MMSE}} = (X^T X)^{-1} X^T Y$, where

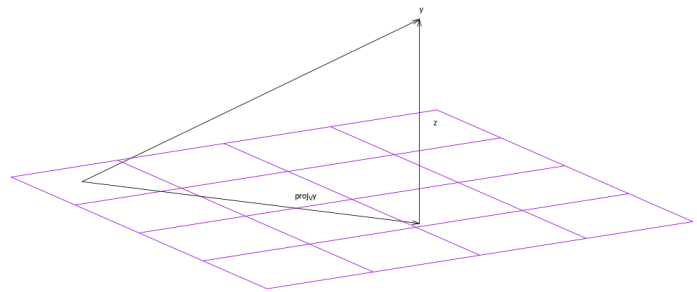
$$X^T Y = \begin{bmatrix} x_{1,1} & x_{2,1} & \cdots & x_{n,1} \\ x_{1,2} & x_{2,2} & \cdots & x_{n,2} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} y_{1,1} & y_{2,1} \\ \vdots & \vdots \\ y_{n,1} & y_{n,1} \end{bmatrix}$$

So A_{MMSE} projects each column of Y onto the columns of X, then normalizes by $(X^T X)^{-1}$.

Interpreting MMSE as Projection

If X is tall and thin, columns of Y are projected onto columns of X as

$$\begin{aligned}\hat{Y} &= X(X^T X)^{-1} X^T Y \\ &= XX^\dagger Y\end{aligned}$$



If X is short and fat, columns of Y are projected onto **rows** of X as

$$\begin{aligned}\hat{Y} &= X^T (XX^T)^{-1} XY \\ &= X^\dagger XY\end{aligned}$$

Outline

Conclusions

- Singular value decomposition:

$$M = U\Sigma V^T = U_r \Sigma_r V_r^T$$

- Pseudo-inverse:

$$\begin{aligned} M^\dagger &= U_r \Sigma_r^{-1} V_r^T \\ &= (M^T M)^{-1} M^T \quad \text{if } M \text{ tall \& thin} \\ &= M^T (M M^T)^{-1} \quad \text{if } M \text{ short \& fat} \end{aligned}$$

- MMSE Projection:

$$A_{\text{MMSE}} = (X^T X)^{-1} X^T Y$$