# Computing Methods for Experimental Physics and Data Analysis

## Data Analysis in Medical Physics

Lecture 5a (Hands-on): Defining functions, Code vectorization; interpolation methods for image transformation and resizing

Alessandra Retico

alessandra.retico@pi.infn.it

INFN - Pisa

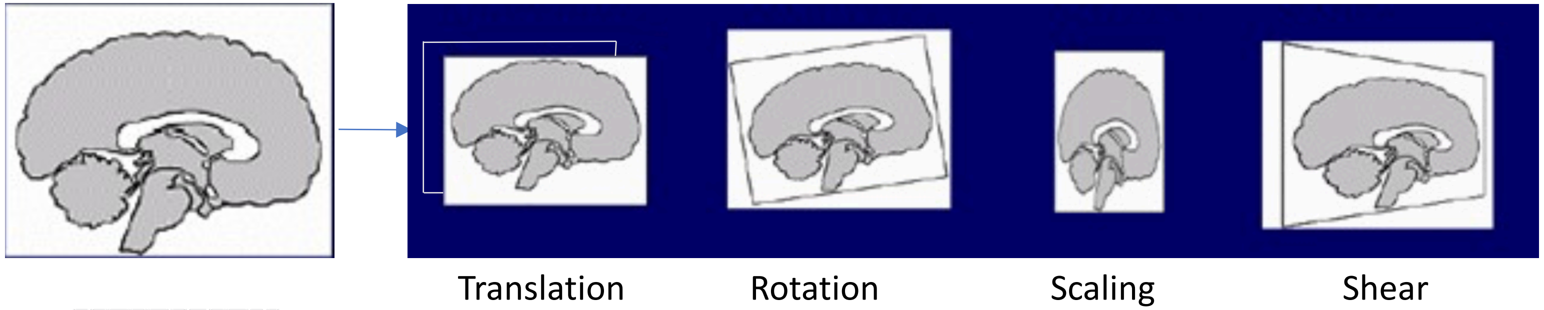
# Performance issues in Matlab

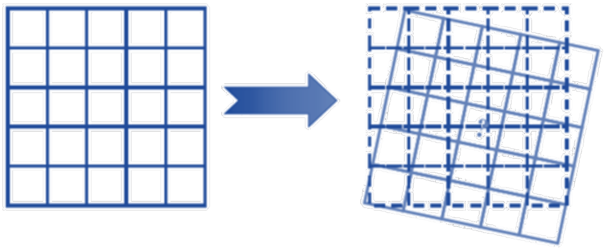
- MATLAB is:
  - very fast on vector and matrix operations
  - correspondingly slow with loops
- MATLAB is a matrix-based language. Avoiding for loops, and using matrices is useful:
  - sometimes for speed
  - sometimes to improve code readability and easy maintenance
- Thus:
  - Try to avoid loops
  - Try to vectorize your code

See demo code:
- show_diamond.m
  - diamond.m
  - (diamond_bad.m)

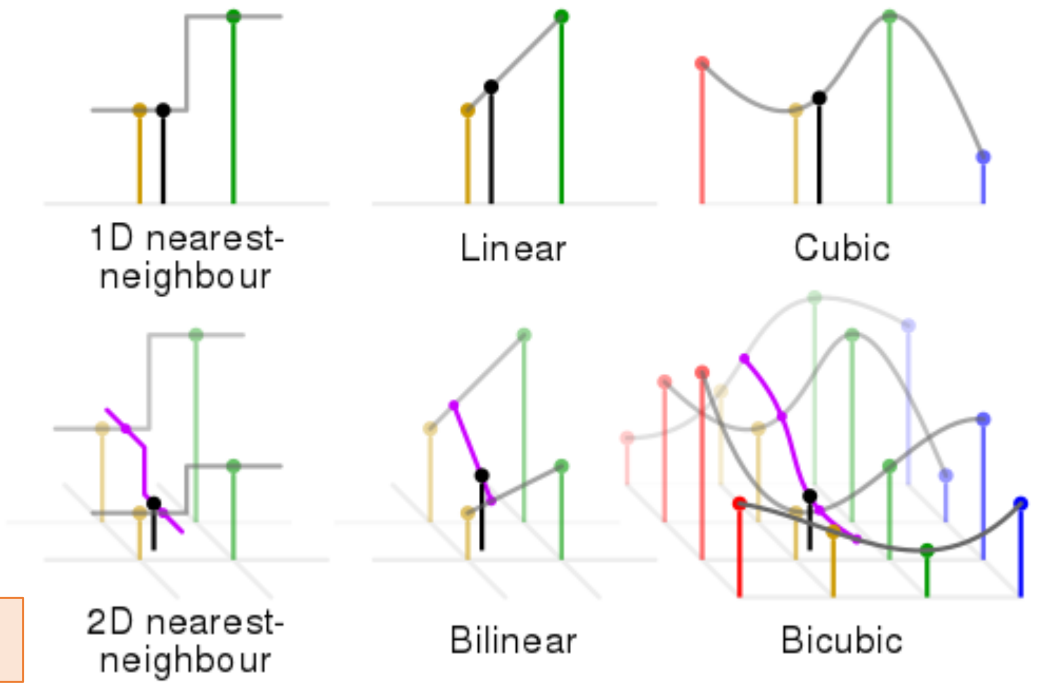
# Image transformations

- Geometric transformations: translation, rotation, scaling, shear



- Transformed images often need resampling; an interpolation method should be specified, e.g.:
  - Nearest neighbor
  - Bilinear interpolation
  - Bicubic interpolation




Exercise: Lecture5_exercise.m

# Affine transformation

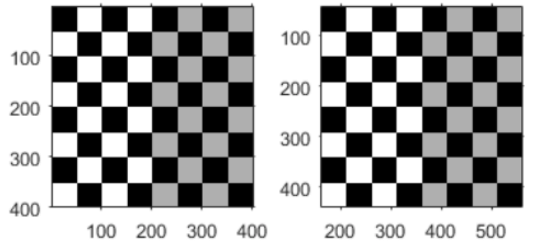
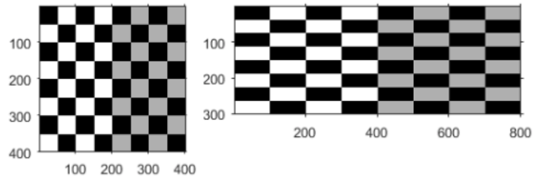
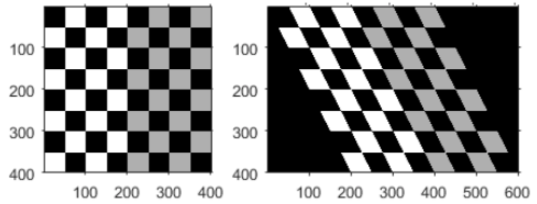
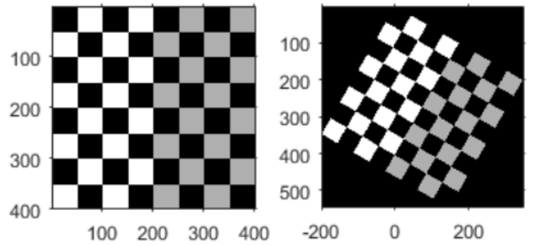
- In Euclidean geometry, an **affine transformation** is a geometric transformation that preserves lines and parallelism (but not necessarily distances and angles).
- An affine map is the composition of two functions: a linear map (multiplication by a matrix **A**) and a translation (addition of a vector ***b***).

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b}.$$

- Using an augmented matrix and an augmented vector, it is possible to represent both the translation and the linear map using a single matrix multiplication.

$$\begin{bmatrix} \vec{y} \\ 1 \end{bmatrix} = \left[ \begin{array}{ccc|c} & A & & \vec{b} \\ 0 & \ldots & 0 & 1 \end{array} \right] \begin{bmatrix} \vec{x} \\ 1 \end{bmatrix}$$

# 2D affine transformations

| 2-D Affine Transformation | Example (Original and Transformed Image) | Transformation Matrix | |
|---|---|---|---|
| Translation |  | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$ | $t_x$ specifies the displacement along the *x* axis<br>$t_y$ specifies the displacement along the *y* axis.<br>For more information about pixel coordinates, see Image Coordinate Systems. |
| Scale |  | $\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $s_x$ specifies the scale factor along the *x* axis<br>$s_y$ specifies the scale factor along the *y* axis. |
| Shear |  | $\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $sh_x$ specifies the shear factor along the *x* axis<br>$sh_y$ specifies the shear factor along the *y* axis. |
| Rotation |  | $\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | q specifies the angle of rotation about the origin. |

Exercise:
- Lecture5_exercise.m