Transformations

Reference: All the learning notes are based on the course "Signal and Image Processing" delivered by the UCPH

Introduction

What is the shape of an object?

Object shape can simply be a curve that represents the **outline of the object boundary**.

Object shape can also be a **collection of curves and points** that are particular for the project.

Mathematical representation: A collection of 2D points

$$x = [x_1, y_1, x_2, y_2, \ldots, x_N, y_N]$$

Definition of **object shape**:

Shape is all the geometrical information that remains after location, scale and rotation effects have been filtered out from an object.

Working with shapes represents with point sets, we need to be able to **remove dependecies on location, scale and rotation in order to compare shapes**.

我的理解是首先物体的形状可以由代表边缘的曲线所表示，同时也可以是一些点集和曲线集合。

同时，如果我们如果想要在点集合表示的物体形状做一些变形，我们就一定要移除点集合里的这些点之间的依赖性，以此来达到变形之后还是代表原物体，因此才能进行比较物体。

Geometric transformations of point sets in R_n

If $\Omega \subset R^n$ is a set of points on R^n, a geometric transformation of any point in Ω is a mapping (a transformation)

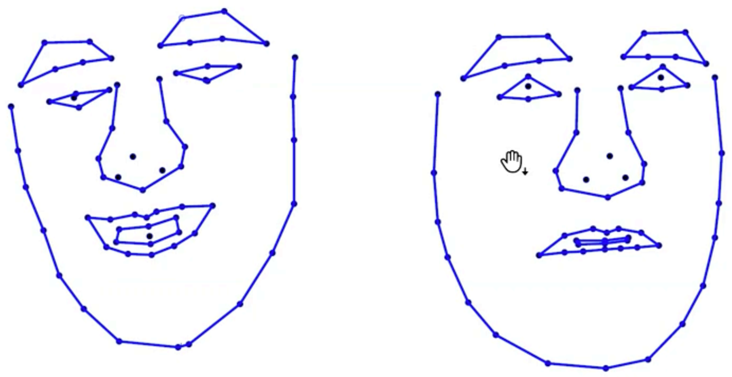
$$\phi : \Omega \to R^n,$$

which is usually assumed to be (at least) injective: every point in Ω is transformed to a unique point in R^n by ϕ.

Depending on application, one may want $\phi(\Omega) = \Omega$.

Important special case: when Ω is a finite set of points.

So you get from this face into this face and not introducing extra things.



Geometric transformations of images

Two "new" problems - boundary conditions

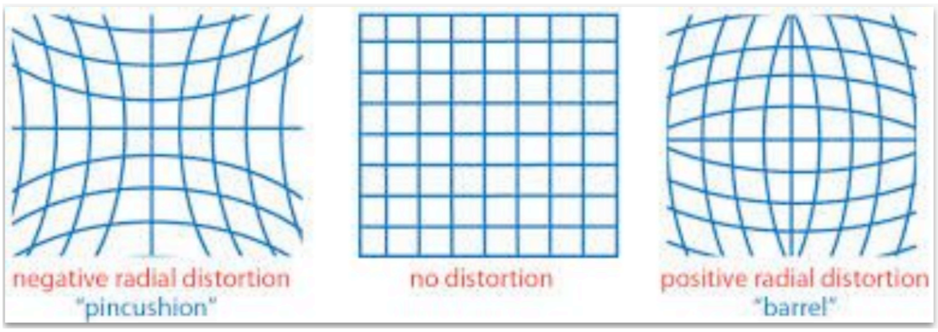
How to discretize? => As the geometric transformations are normally in continuous.

How to handle "outside the image" => Just like what we have mentioned before in the filtering.

Applications

Correction of lens distortion

Cheap or wide-angle lenses causes **geometric lens distortion**. The most common distortion is (barrel) radical distortion.



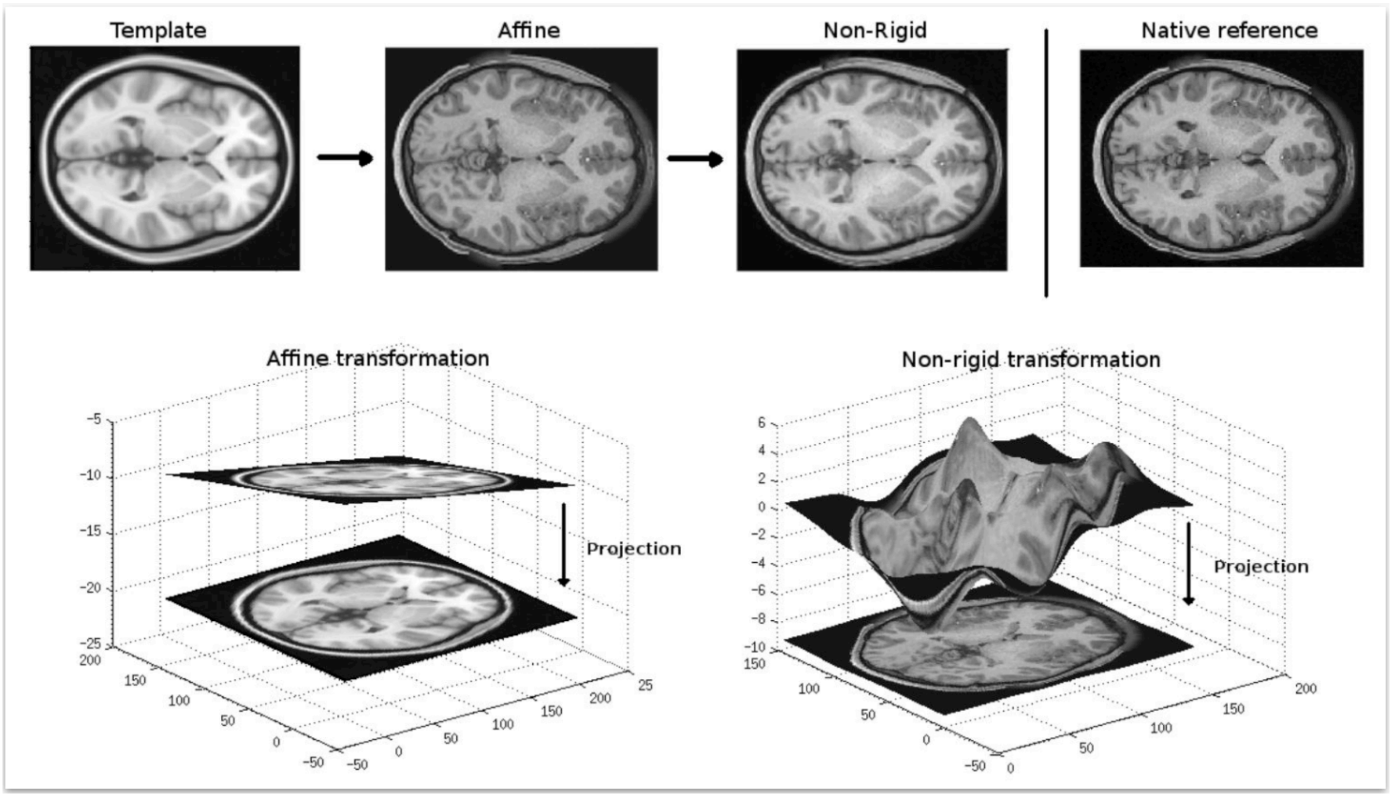
What we really want is in the middle here which is **no distortion**.

So, to comply with the assumptions of the pin-hole camera model, it may be necessary to **correct for lens distrotion** before doing anythong else.

Below is the example of lens distortion. **A poor webcam with a barrel distortion**

Image registration

Then, there is an image registration (matching): **Brain MRI**



In medical imaging, the **native reference** refers to the **original coordinate system of an image**, which is defined by the **scanner** or imaging modality that acquired the image. The native reference typically includes information such as the position, orientation, and voxel size of the image.

In the context of image registration, **the template is a reference image that is used as a target to which other images are aligned**.

Affine registration is a type of registration that involves **applying a linear transformation to the moving image to bring it into alignment with the template.** This transformation typically includes scaling, rotation, and translation. Affine registration is a useful method for aligning images that have undergone similar transformations, such as images from the same modality or images acquired in the same scanning session.

Non-rigid registration, on the other hand, is a type of registration that **allows for more flexible transformations of the moving image**. Non-rigid registration can be used to align images that have undergone more complex deformations, such as images acquired from different modalities or images acquired at different time points. Non-rigid registration typically involves a more complex transformation, such as a free-form deformation, which allows for local deformations of the moving image.

在医学图像处理中，本地参考是指图像的原始坐标系，由扫描仪或成像模态定义。本地参考通常包括位置、方向和像素大小等信息。

在图像配准的背景下，模板是一个参考图像，用于将其他图像对齐到该图像。仿射配准是一种配准方法，涉及将移动图像应用线性变换，使其与模板对齐。此变换通常包括缩放、旋转和平移。仿射配准是一种有用的方法，用于对齐经历了类似变换的图像，例如来自同一模态的图像或在同一扫描会话中获取的图像。

而非刚性配准则允许更灵活的移动图像变换。非刚性配准可用于对齐经历了更复杂变形的图像，例如来自不同模态或在不同时间点获取的图像。非刚性配准通常涉及更复杂的变换，例如自由形变，它允许移动图像的局部变形。

这些配准方法之间的关系在于它们都旨在将一个图像对准到另一个图像，通常是一个模板。仿射配准是一种更简单的配准方法，适用于对齐经历了类似变换的图像，而非刚性配准则是一种更灵活的配准方法，允许更复杂的变形。这两种方法都是医学图像分析中的重要工具，可用于各种应用，例如图像分割、图像融合和图像引导干预。

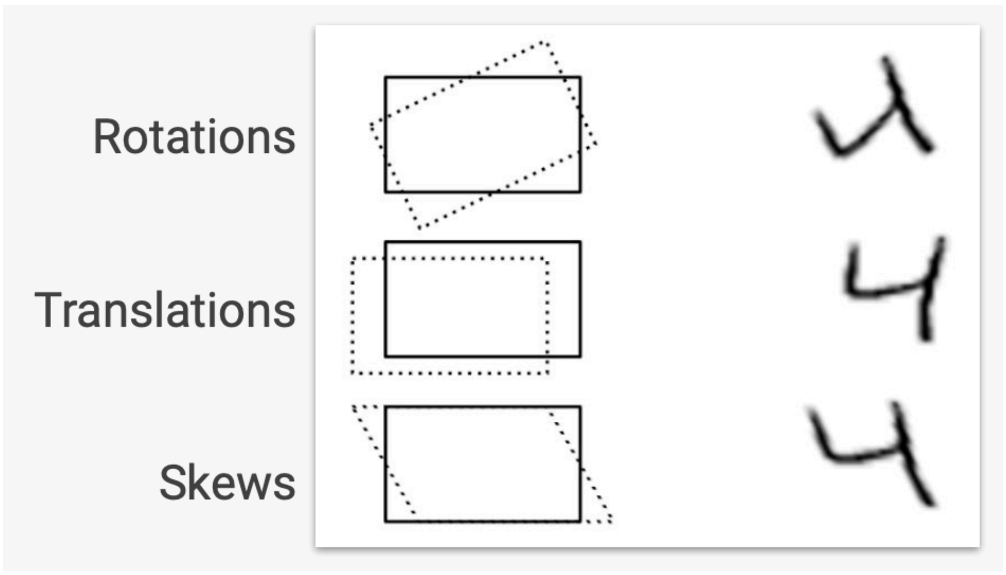
Data augmentation

Machine learning methods needs many examples to learn from.

Additional examples can be generated from existing examples by geometrical transformations => New data is created by deforming the old data.

- Transformation **types** are typically chosen and **parameters** sampled from a range of allowed values.
- Choices of transformations can be used to teach methods invariances.

Example deformations (for image data) include:



数据增强（Data augmentation）是一种在机器学习和深度学习中经常使用的技术，其目的是通过对原始数据集进行各种转换来扩充数据集规模，以增加模型的鲁棒性和泛化能力。

对于手写数字识别问题，通过对原始图像进行旋转、平移或倾斜等操作可以生成一系列新的图像，这些新图像包含了原始图像的一些变化，从而增加了数据集的规模。**通过增加数据集的规模，可以帮助机器学习模型更好地学习到数据的不同特征，提高其分类性能和鲁棒性。**

例如，通过对数字4进行旋转、平移或倾斜等操作，可以生成多个不同的图像，如横向或纵向倾斜的4、略微旋转的4、左右平移的4等。**这些新的图像可以用来扩充原始数据集，增加数据的多样性和数量，从而提高模型的性能**。

因此，旋转、平移或倾斜等操作被视为数据增强技术的一部分，因为它们可以通过增加数据集的规模和多样性来提高机器学习模型的性能。

Common for these applications is that **we need to be able to apply transformations to sets of points in images**.

Either we only **transform the point set** or we **transform the image intensities** based on the point set.

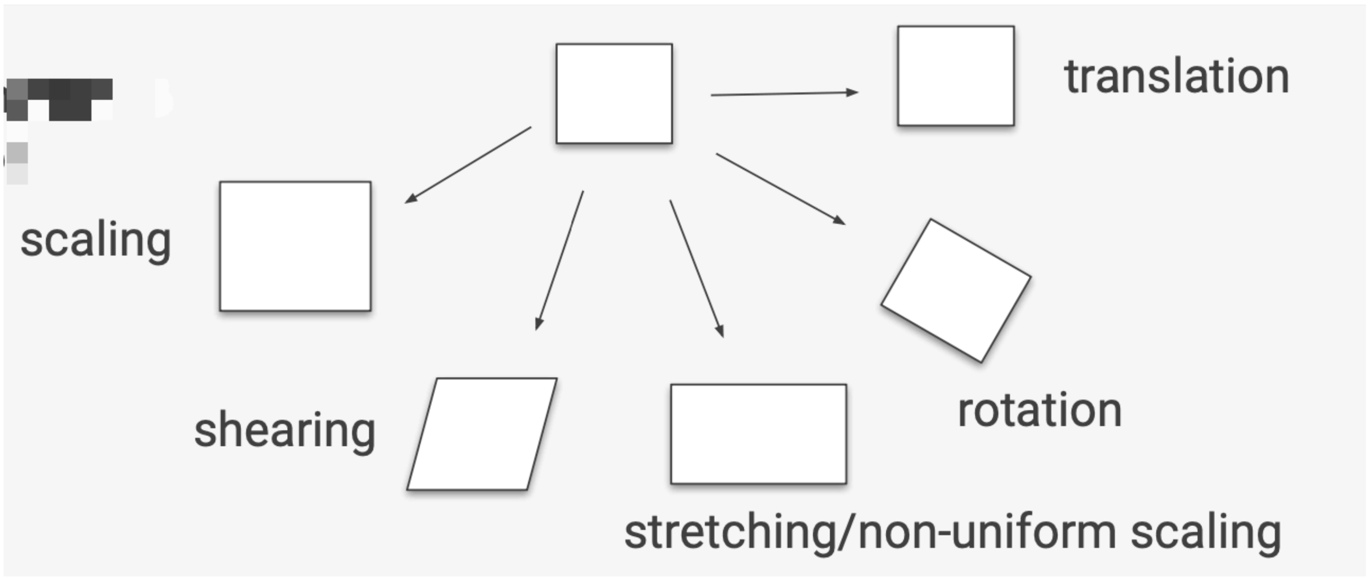
Types of transformations

Linear

Affine transformation

We start by the most simple one which is an affine transformation.

An affine transformation is one that consists of **following**, so just moving things around basically.



So, the affine transformation can be written as, $R^2 \rightarrow R^2$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

这个公式是描述 2D Affine 变换的数学模型。假设有一个点 (x, y)，经过 Affine 变换之后的坐标为 (x', y')，那么这个公式告诉我们如何计算 x' 和 y'。

公式中的矩阵 $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ 表示线性变换部分，它对应了平移、旋转和缩放等操作。矩阵中的 a_{11} 和 a_{22} 分别对应了水平和垂直方向的缩放因子，a_{12} 和 a_{21} 分别对应了旋转和剪切因子。

公式中的向量 $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$ 表示平移部分，它对应了在水平和垂直方向上的平移量。

当一个点 (x, y) 经过 Affine 变换后，它的新坐标 (x', y') 可以通过将原始坐标 (x, y) 进行线性变换，再加上平移部分来计算得到。具体来说，公式中的 [x\y] 表示原始点的坐标，将其乘以矩阵 $[a_{11} \quad a_{12}\, a_{21} \quad a_{22}]$ 可以得到一个新的向量，表示线性变换后的坐标。再将这个向量加上平移向量 $[t_x\ t_y]$，就得到了最终的坐标 $[x'\ y']$。

总之，这个公式提供了一种数学描述方法，可以将 Affine 变换转换为矩阵和向量的形式，方便计算机程序实现。

What is this?

1

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \implies \mathcal{Translation}$$

This is only the translation as the previous term is the **identity matrix**.

2

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \implies \mathit{Scaling\ around\ origin}\ (0,0)$$

And **it is uniform** if $a_{11} = a_{22}$. x scaling with a_{11} and y scaling with a_{22}.

3

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \implies \mathit{Counter - clockwise\ rotation\ around\ origin}$$

4

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a_{12} \\ a_{21} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \implies \mathit{Shearing\ around\ origin}$$

Affine transformations in homogeneous coordinates

As the translation part of the affine transformation **<u>does not keep the origin fixed</u>**, affine transformation is **not a linear transformation** and cannot be written as a single 2×2 multiplication.

It can be written as a single 3×3 matrix multiplication in ***homogeneous coordinates***:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

在计算机视觉和计算机图形学中，Affine 变换通常使用齐次坐标（Homogeneous Coordinates）来表示。

齐次坐标是一种增加了一个维度的坐标表示方法，它可以将 Euclidean 空间中的点表示为一个更高维度的向量。例如，在二维平面中，一个点 (x, y) 可以表示为一个三维向量 $(x, y, 1)$。在三维空间中，一个点 (x, y, z) 可以表示为一个四维向量 $(x, y, z, 1)$。

使用齐次坐标，Affine 变换可以表示为一个矩阵乘法形式。例如，在二维平面中，Affine 变换可以表示为以下形式：

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

其中，$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$ 表示 Affine 变换的矩阵形式，$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ 表示原始点的齐次坐标，$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ 表示变换后的点的齐次坐标。

使用齐次坐标可以简化 Affine 变换的数学表示和计算，同时可以方便地将多个变换组合起来进行复合变换。因此，在计算机视觉和计算机图形学中，齐次坐标经常被用来表示 Affine 变换。

When we have the resulting transformation here, we can just remove the one and plot x and y coordinates.

How many **degrees of freedom** does an affine transformation have? How many points needed to **uniquely determine a mapping**?

自由度是指可以在一个系统中独立变化的参数数量。在计算机视觉和计算机图形学中，自由度通常用于描述某种变换所需要的最小参数数量。

对于一个 Affine 变换，它可以表示为一个 2×2 的旋转、缩放和错切矩阵和一个 2×1 的平移向量的组合。因此，它总共有 $4 + 2 = 6$ 个自由度。

另一方面，为了唯一地确定一个 Affine 变换，需要至少知道三个点及其对应的映射点。因为 Affine 变换包含了平移变换，所以需要至少三个点来解决平移的影响。每个点提供了两个方程（对应于 x 和 y 方向的变换），因此需要至少三个点来提供 6 个方程以确定 Affine 变换的 6 个自由度。因此，我们需要至少知道三个点来唯一地确定一个 Affine 变换。 => Three points ensure an unique affine function (mapping).

A 2D point (x, y) has, for any $w \neq 0$, a homogenous coordinate representation $[wx, wy, w]^\top$.

For $w = 1$, we get $[x, y, 1]^\top$.

If $[h_1, h_2, h_3]^\top$ is a homogeneous coordinate representation of (x, y), then

$$x = h_1/h_3, y = h_2/h_3$$

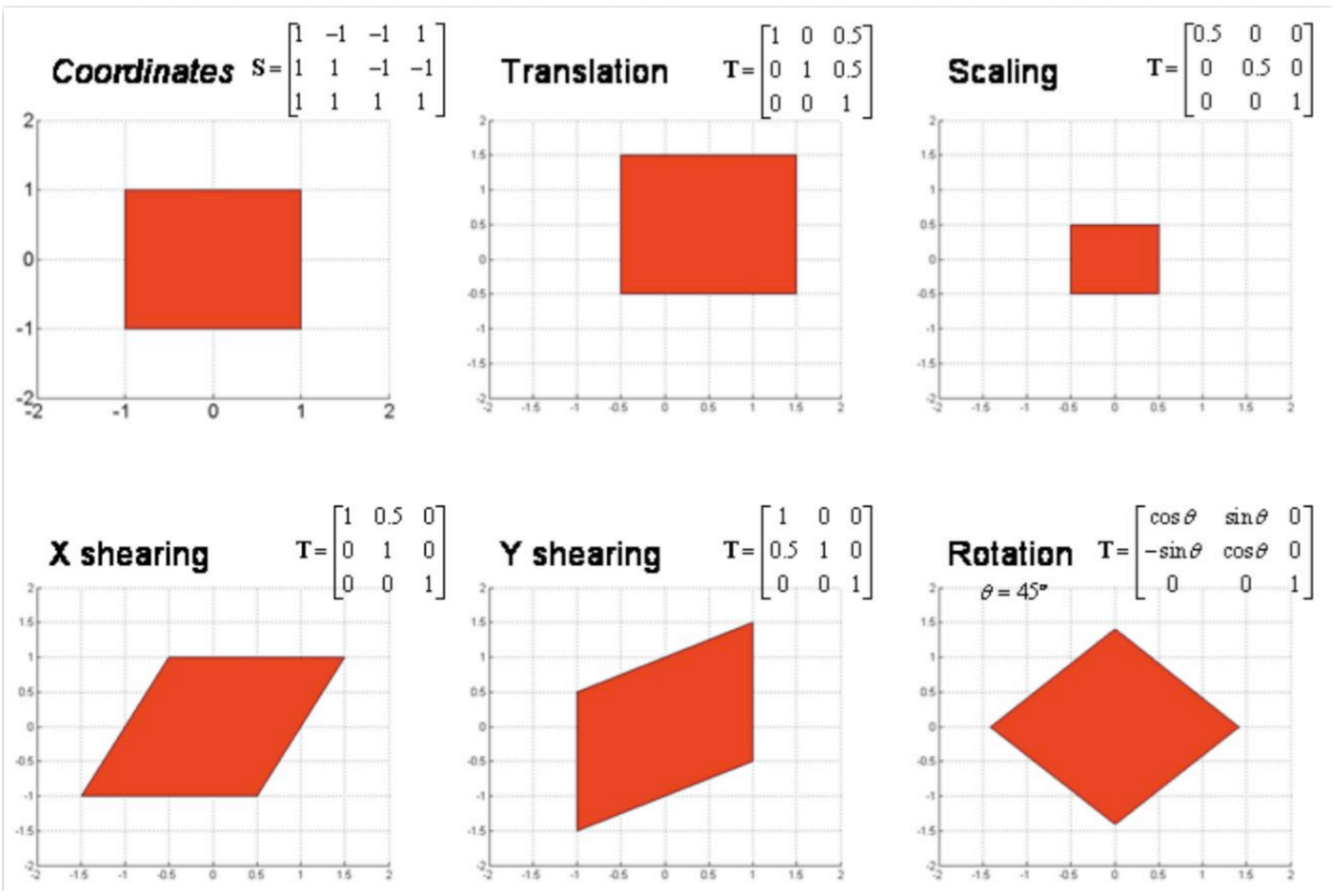
Any **linear transormation** can be written in homogeneous coordinates as,

$$\begin{bmatrix} wx' \\ wy' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Any **affine transformation** can be written as a single 3×3 matrix multiplication in homogeneous coordinates as,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Examples



Remember the **shape of an object** is loosely defined **as the properties of the object** that are invariant under scaling, translation and rotation.

Affine transformations do not generally preserve shape.

Rigid transformations

An affine transformation matrix $T = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$ with submatrix $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ defines a rigid transformation ***if A is orthonormal***, that is, if for the vectors $v = [a_{11}, a_{12}]^\top$ and $w = [a_{21}, a_{22}]^\top$, we have,

$$||v|| = ||w|| = 1 \; and \; v^\top w = 0$$

What does this mean geometrically?

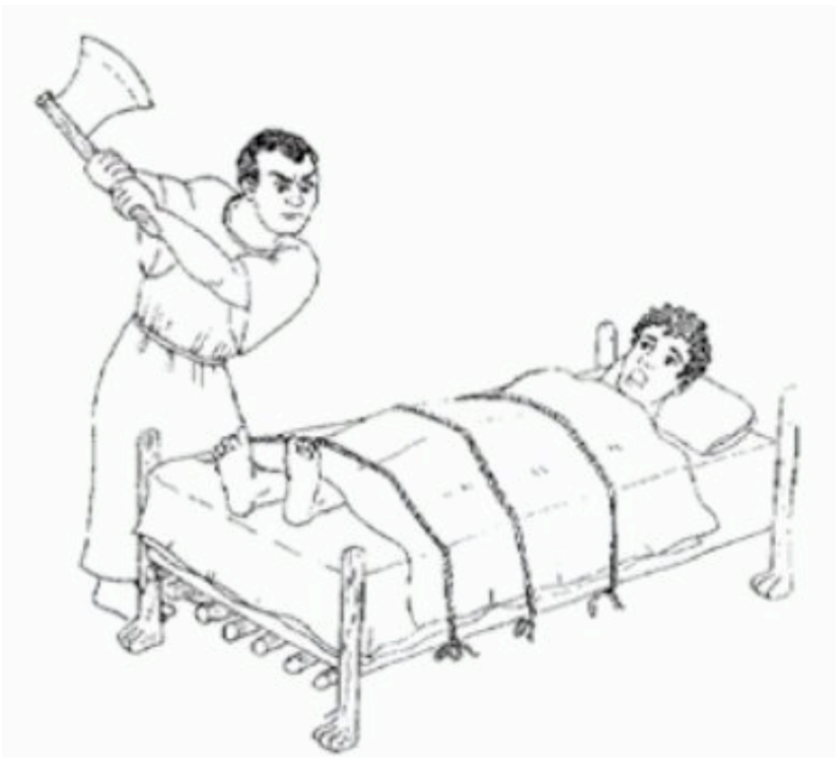
Rigid transformations are composed of **translation**, **rotation** and **reflection** only.

Preserve **straight lines, parallel lines, lengths and angles**.

Levaing out reflection, how many degrees of freedom? => 3 => *as we have a translation and rotation. translation is 2 and rotation is only 1*

Do rigid transformations preserve shape? => *Yes, but there are more that do. What is missing?*

Procrustes



So, here we have the famous ***procrustes***.

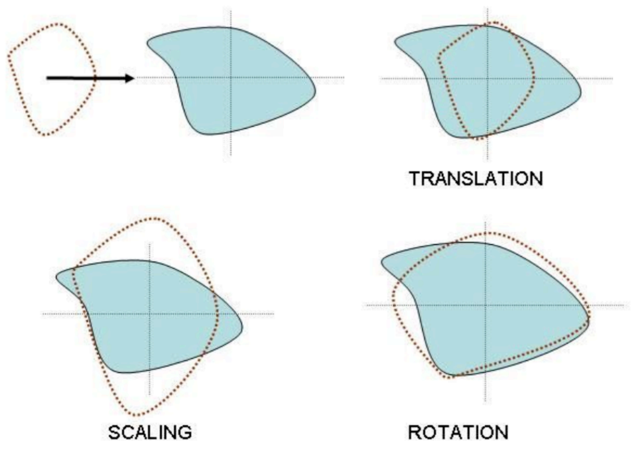
Procrustes（普洛克鲁斯特斯）是一个古希腊神话中的人物，据说他有一张铁床，他会将旅行者按照铁床的大小拉伸或压缩来让他们适应床的大小。

在计算机视觉和数学中，Procrustes问题指的是将两组点进行配准的问题。具体来说，**给定两组点集，我们需要将它们进行平移、旋转和缩放等变换，以使得它们的形状和位置尽可能相似**。

Procrustes问题在计算机视觉中有着广泛的应用，比如在人脸识别中，可以使用 Procrustes 算法将两张人脸图像进行对齐，以便进行特征提取和比较。此外，Procrustes问题还有许多其他的应用，比如在形态分析、信号处理、语音识别等领域中。

So, procrustes if we have these two shapes.

Example



Above is what procrustes do.

Procrustes transformations in homogeneous coordinates are given by

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & -\gamma & \lambda_1 \\ \gamma & \alpha & \lambda_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

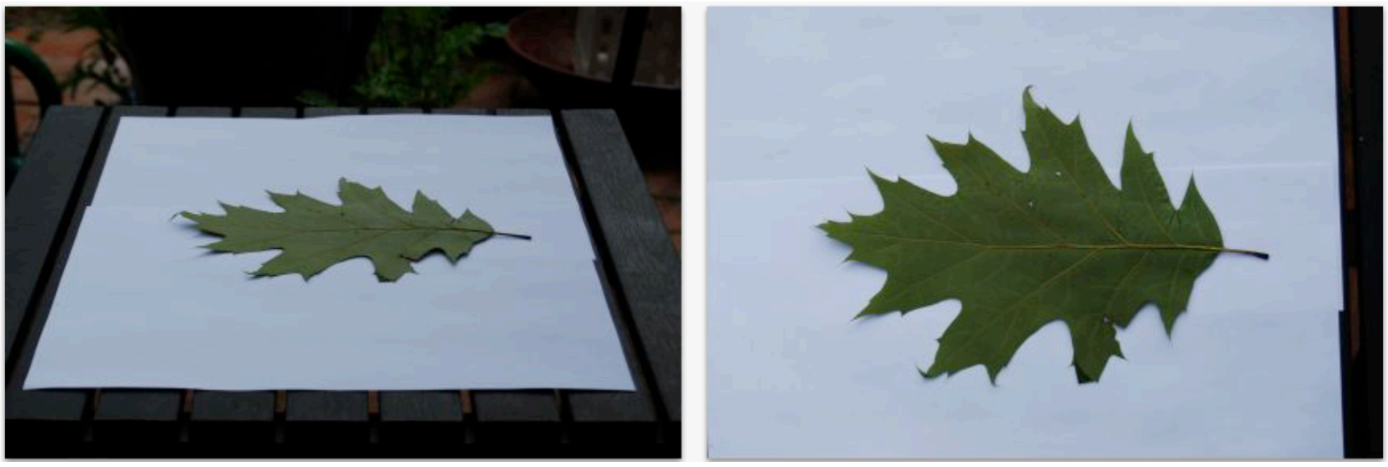
Notice only 4 degrees of freedom (DOF) compared to the 6 DOF of affine transformations.

Procrustes $\in$ ***Rigid Transformation***

Perspective transformation

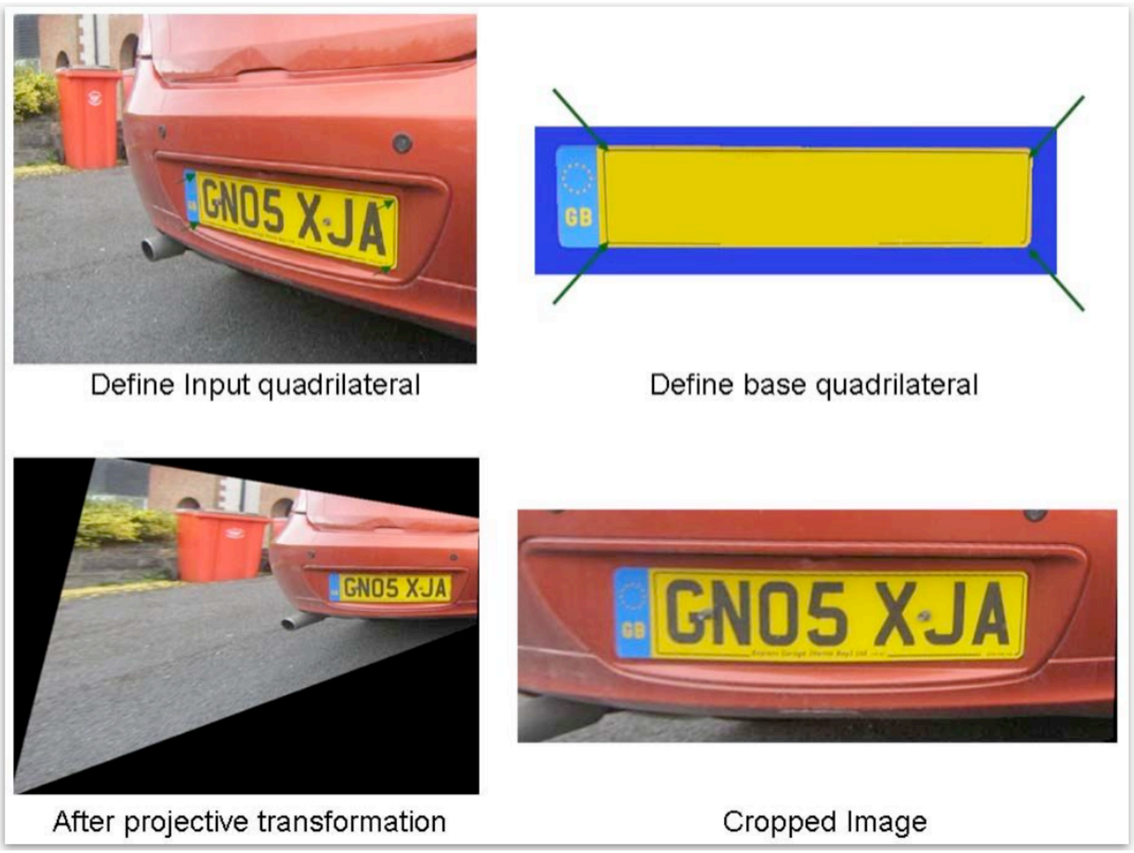
Describe transformation between two images of perspective projections of 3D planar scenes

Map lines to lines and map rectangles to quadrilaterals.



How many DOF => **8** and how mnay points needed to dertermine one => **4**

Examples usage of perspective transformation


Define Input quadrilateral
Define base quadrilateral
After projective transformation
Cropped Image
GN05 XJA
GB

Composition of transformations

Transformations can be composed

$$\phi_1 \circ (\phi_2 \circ \phi_3) = (\phi_1 \circ \phi_2) \circ \phi_3$$

In general the order matters

$$\phi_1 \circ \phi_2 \neq \phi_2 \circ \phi_1$$

Nonlinear

Diffeomorphisms

Often represented via (smooth) vector fields.

Commonly used for warping of images

- deformable image registration
- data augmentation in ML

- Motion models
- etc.

The most common type of nonlinear transformation is the diffeomorphism, which is:

- A $1-1$ map (so there exists an inverse)
- Smooth
- Has a smooth inverse

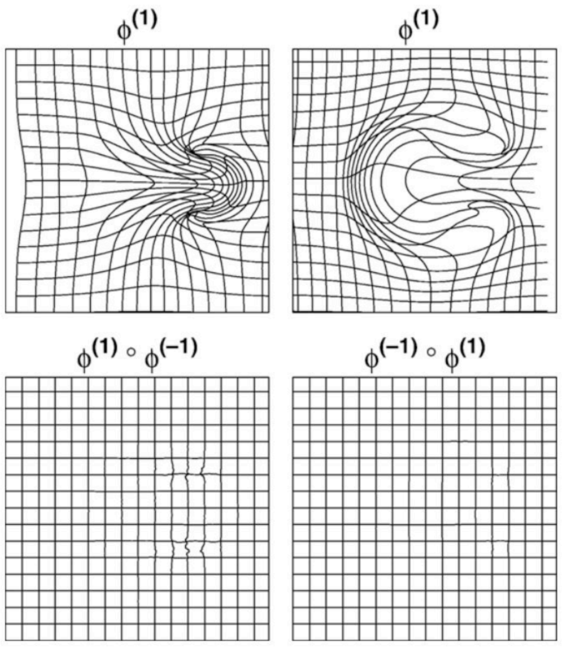
Diffeomorphisms are usually represented via their tangential vector fields

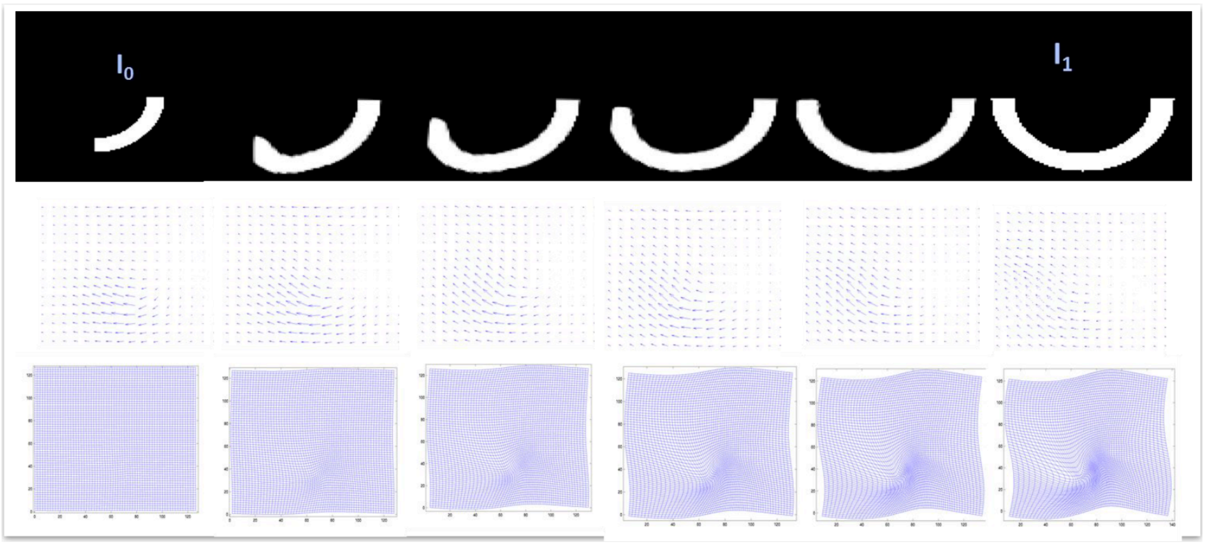
非线性变换中最常见的类型是微分同胚（diffeomorphism），其定义如下：

- 一个一一对应的映射（即每个点都有唯一的映射）
- 具有光滑性质
- 具有光滑的反函数

微分同胚通常通过其切向量场来表示。切向量场指的是在空间中每个点上定义的向量场，每个向量描述了在该点上的变换效果。微分同胚的切向量场也具有光滑性质。

换句话说，微分同胚是一种光滑的、双向可逆的变换，可以用它的切向量场来表示。它的双向可逆性保证了变换的唯一性，光滑性质则保证了变换的平滑性和连续性。微分同胚在图像处理和计算机视觉领域中有广泛的应用，例如图像配准、形变建模、形状匹配等方面。


$\phi^{(1)}$
$\phi^{(1)}$
$\phi^{(1)} \circ \phi^{(-1)}$
$\phi^{(-1)} \circ \phi^{(1)}$



Common issues

Not all vector fields correspond to diffeomorphisms.

Numerically found diffeomorphisms are not inverse of their numercially found inverses.

Do you think diffeomorphisms are sufficiently general to cover all needs for geometric transformations? => Ofc Not.

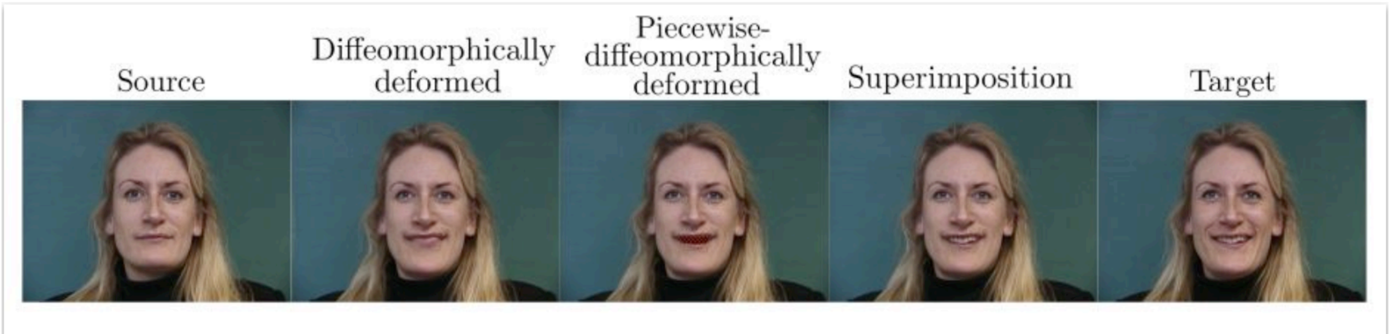

Source
Diffeomorphically deformed
Piecewise-diffeomorphically deformed
Superimposition
Target

Figure: Kok Nielsen, Darkner, Feragen, 2019.

For instance, the source and the target.

The diffeomorphism is not able to open the mouth as it's smooth.

Simply cut it to open, superimpose the missing data.

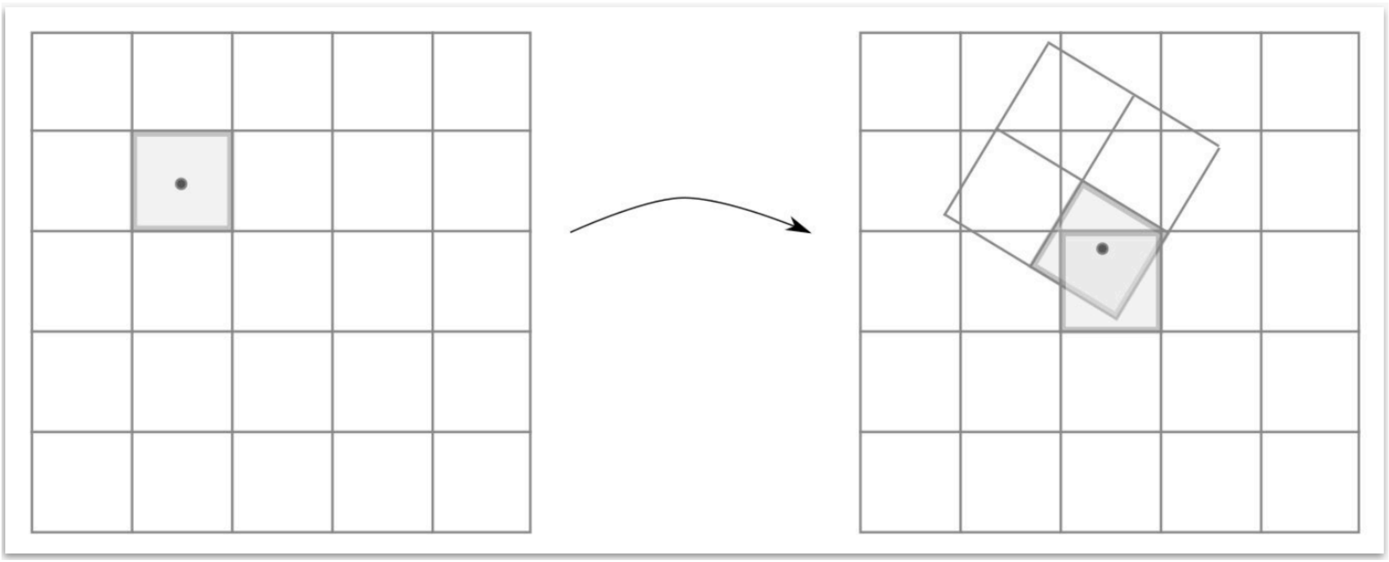
Transformations of images versus point sets

Let's say $\phi : R^2 \rightarrow R^2$ is a transformation.

We know how to apply ϕ to a set of points $S \in R^2$: We obtain $S' = \{\phi(s) | s \in S\} = \phi(S)$.

How do I apply the transformation ϕ to the image? What problems do I face?

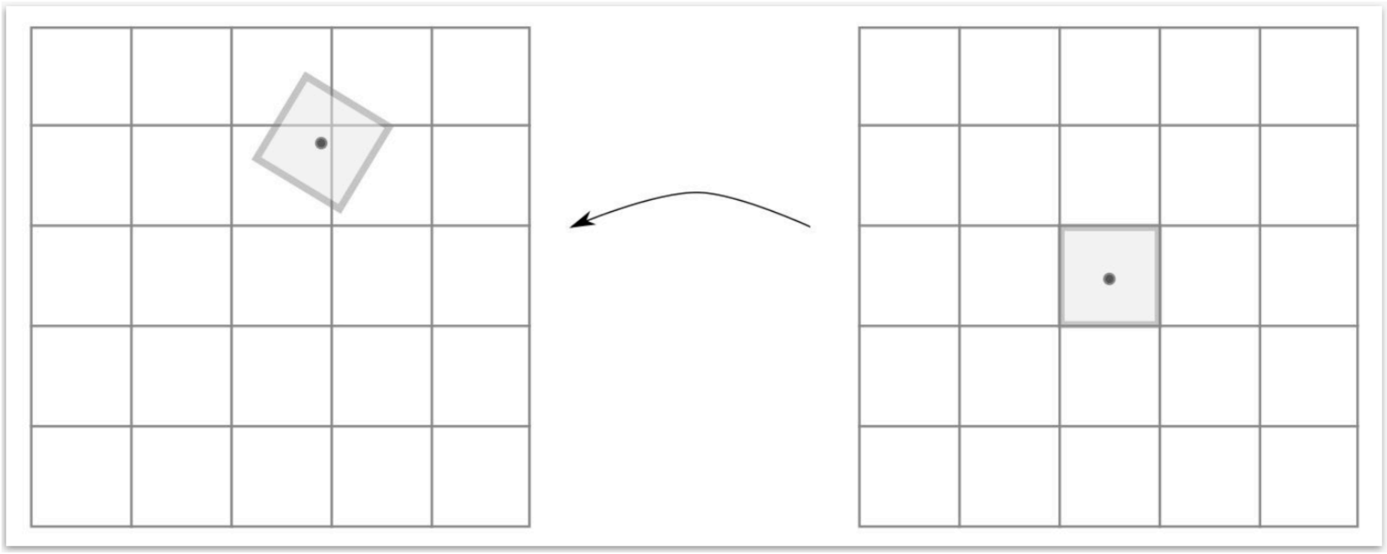
For instance, I have a **Forward mapping** here.



Are there potential problems of this?

- Mapping multiple pixels onto one => The original pixel currently overlap 4 pixels
- Holes (pixels not being hit)

Backward mapping (inverse warp)

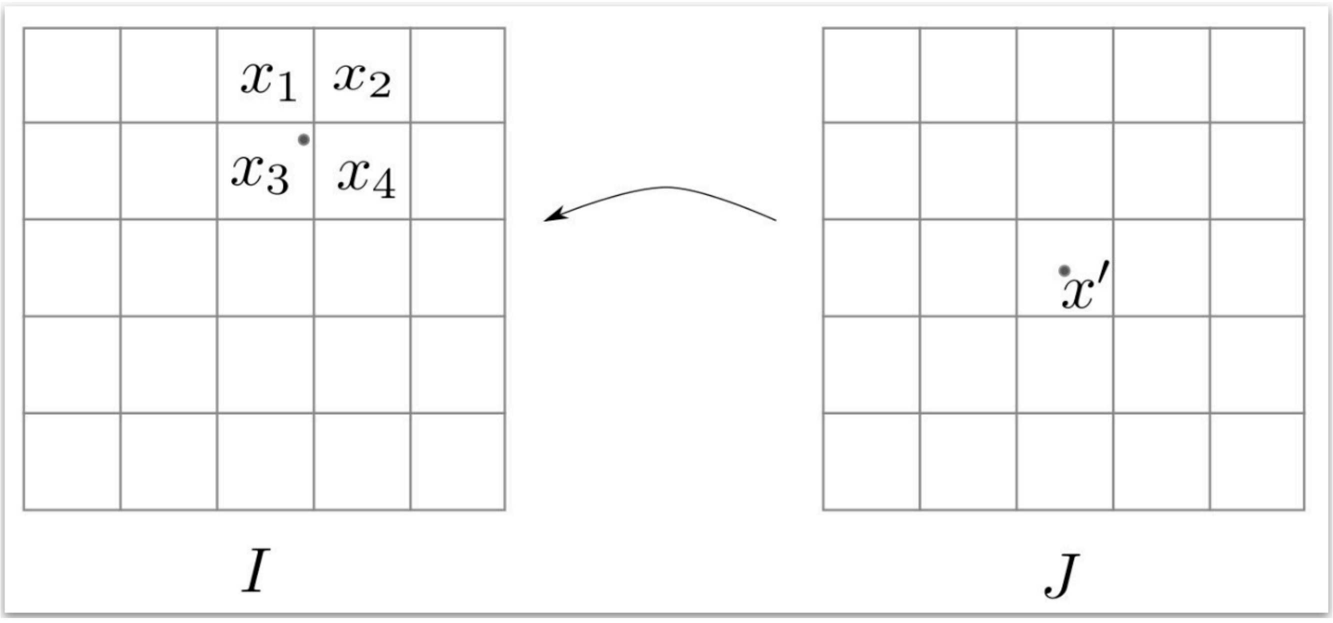


Are there potential problems of this?

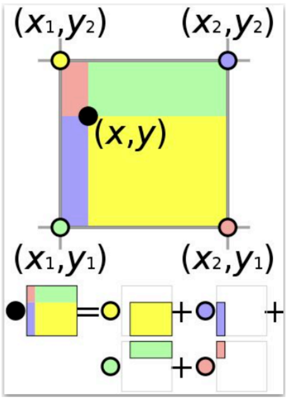
- Don't hit pixel centres - need to interpolate.
- Don't always hit the original image - need boundary conditions

Common interpolation methods

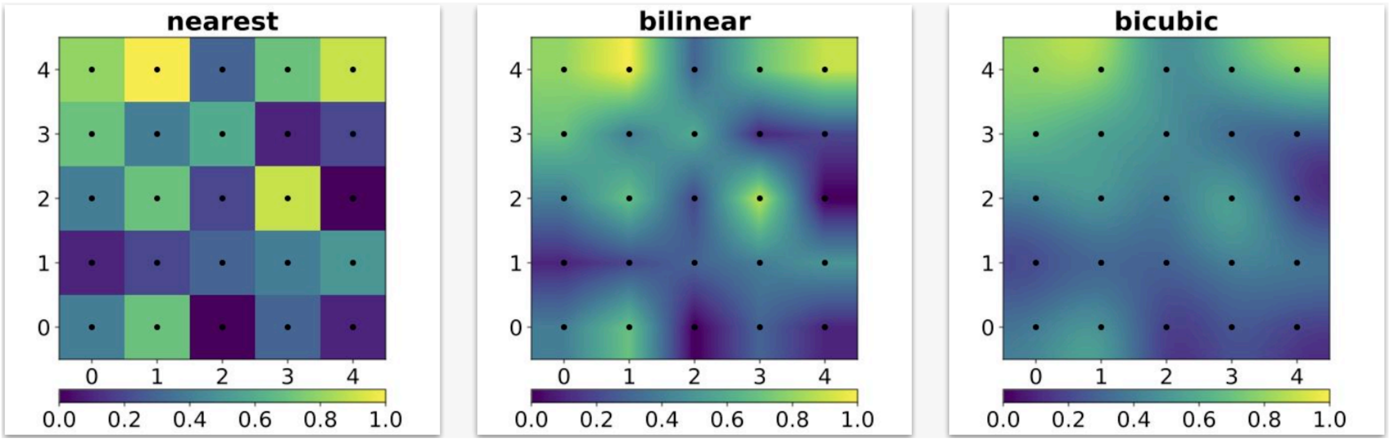
Nearest neighbor (pick closest pixel): $J(x') = I(x_3)$



Or **bilinear**, $J(x') = (1-b)(1-a)(I(x_1) + aI(x_2)) + b((1-a)I(x_3) + aI(x_4))$.



Other: Polynomial, spline, or in the Fourier domain.



Common boundary conditions

- Zero padding: set pixels outside 1 to 0
- Symmetric mirrorin: Mirror duplicate pixels at the border

Dictator: Ying Liu, March 12, 2023