Image Analysis

Rasmus R. Paulsen
Tim B. Dyrby
DTU Compute

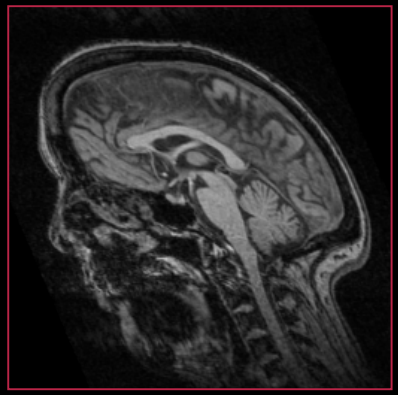
rapa@dtu.dk

http://www.compute.dtu.dk/courses/02502

Plenty of slides adapted from Thomas Moeslunds lectures

Lecture 8 – Geometric Transformation



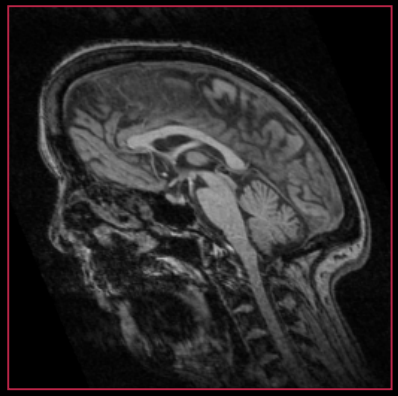
What can you do after today?

- Compute the translation of a point
- Compute the rotation of a point
- Compute the scaling of a point
- Compute the shearing of a point
- Construct a translation, rotation, scaling, and shearing transformation matrix
- Use transformation matrices to perform point transformations
- Describe the difference between forward and backward mapping
- Use bilinear interpolation to compute the interpolated value at a point
- Transform an image using backward mapping and bilinear interpolation
- Compute a line profile using bilinear interpolation
- Describe the homography
- Describe how the coefficients of a homography can be computed using four corresponding points

Geometric transformation

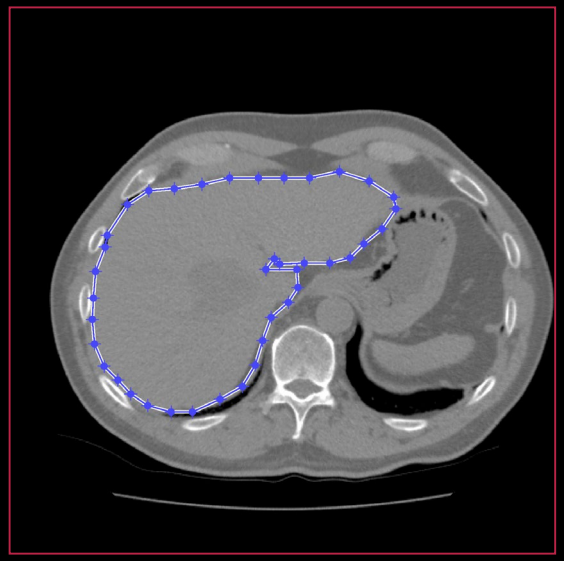
- Moving and changing the dimensions of images
- Why do we need it?



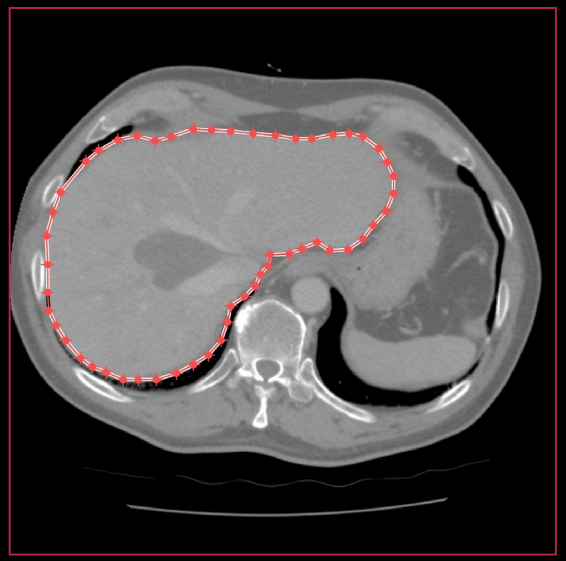



Change detection

- Patient imaged before and after surgery
- What are the changes in the operated organ?
- Patient can not be placed in the exact same position in the scanner



Before surgery

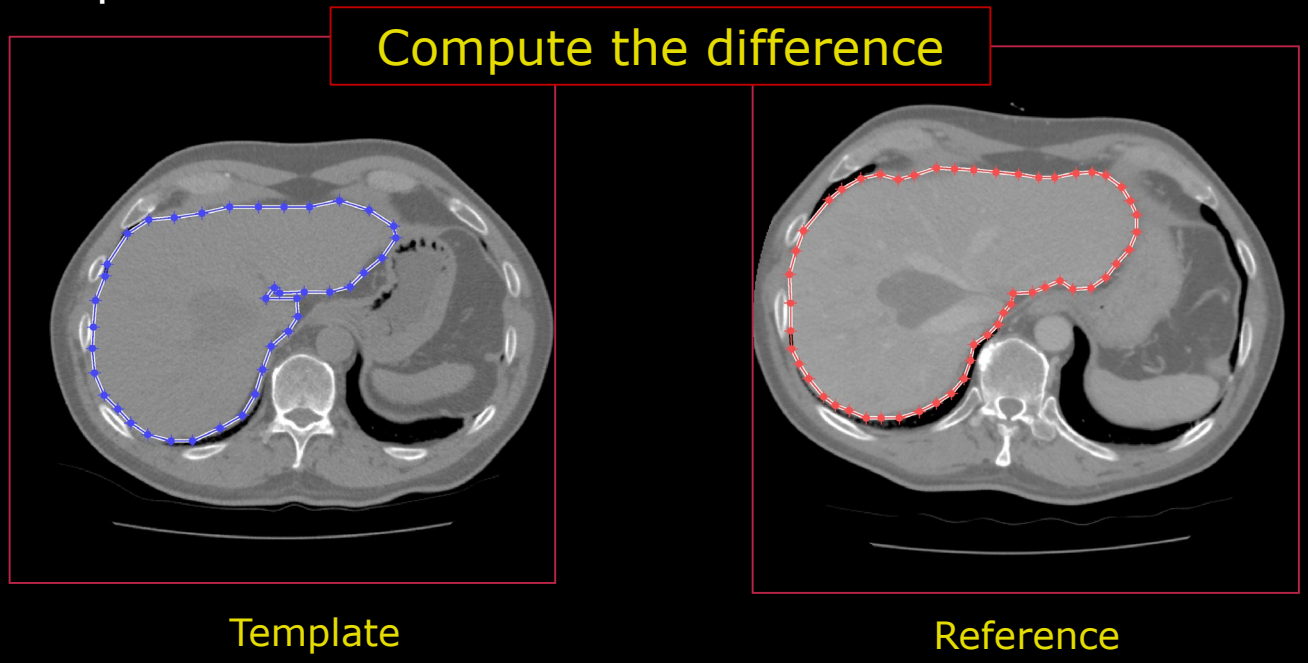


After surgery

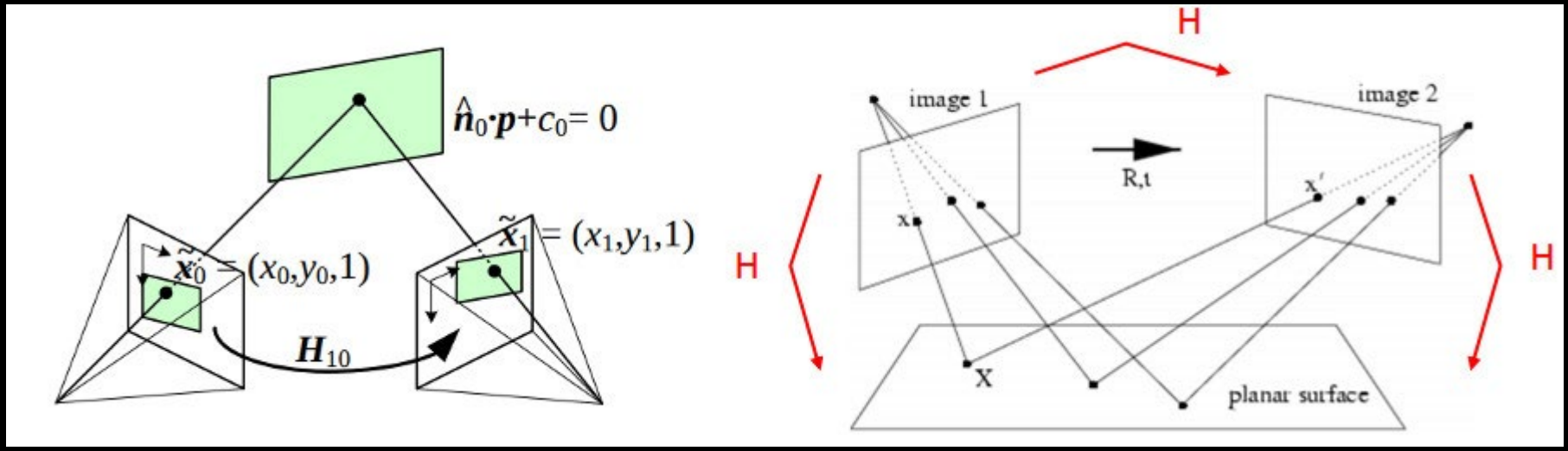
Bachelor project: Image Guided Surgery Planning

Image Registration

- Change one of the images so it fits with the other
- Formally
  - Template image
  - Reference image
  - Template is moved to fit the reference



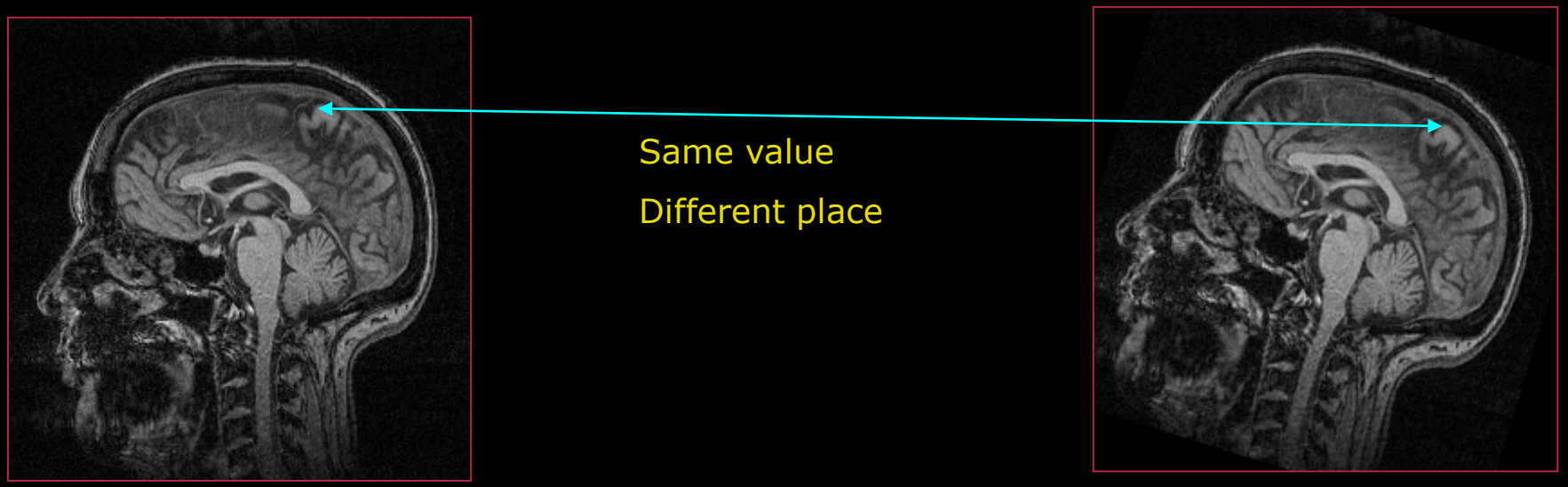
Camera Calibration



2D projective transformations (homographies).
Christiano Gava, Gabriele Bleser

Geometric Transform

- The pixel intensities are not changed
- The "pixel values" just change positions



Different transformations

- Translation
- Rotation
- Scaling
- Shearing

- Homographies
- Advanced transformations

Translation

- The image is shifted – both vertically and horizontally

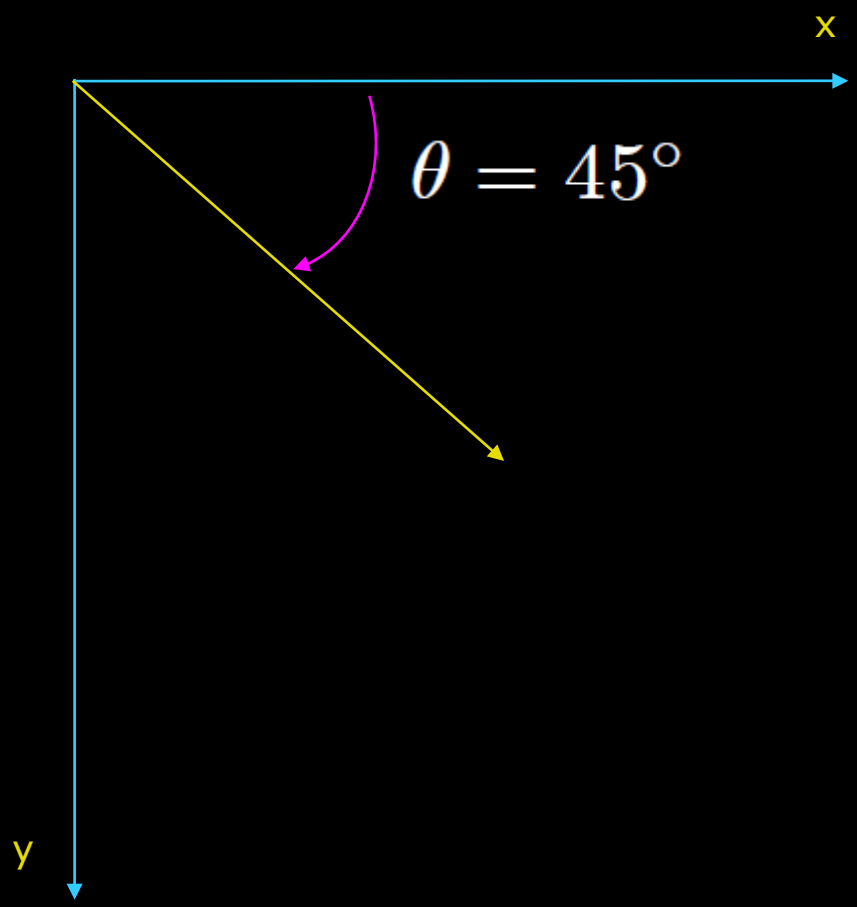
$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} 60 \\ 20 \end{bmatrix}$$

Rotation

- The image is rotated around the centre or the upper left corner
- Remember to use degrees and radians correctly
  - Matlab uses radians
  - Degrees easier for us humans

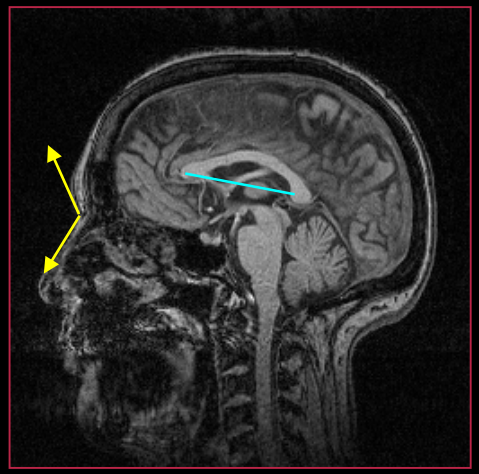
$$\theta = 15°$$

Rotation coordinate system



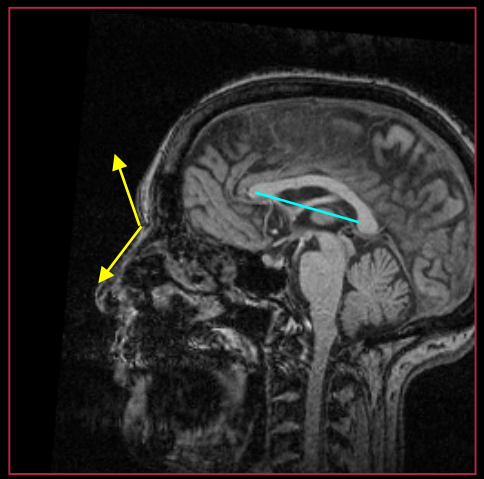
Rigid body transformation

- Translation and rotation
- Rigid body = stift legeme
- Angles and distances are kept



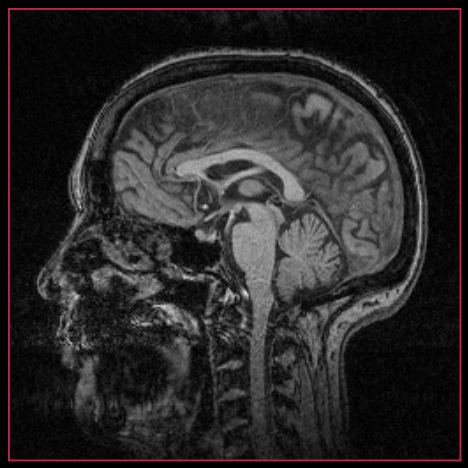
$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} 60 \\ 20 \end{bmatrix}$$

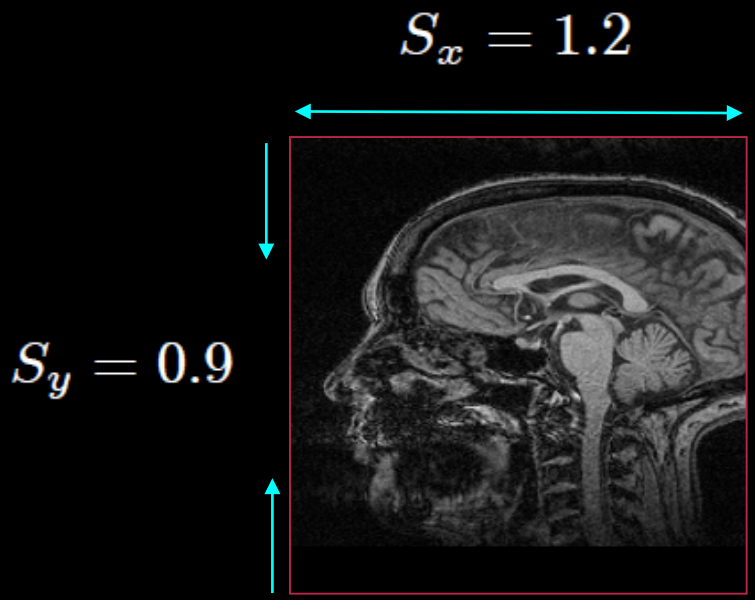
$$\theta = 5°$$



Scaling

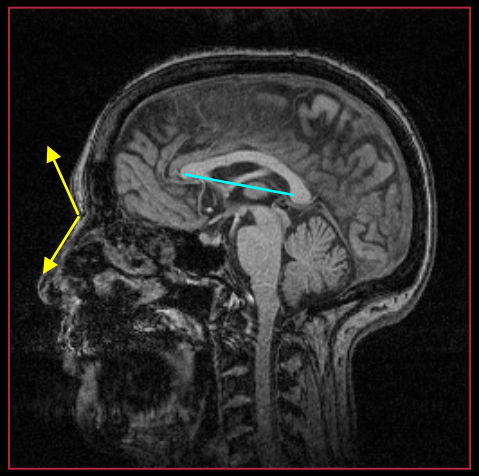
- The size of the image is changed
- Scale factors
  - X-scale factor S_x
  - Y-scale factor S_y
- Uniform scaling: $S_x = S_y$

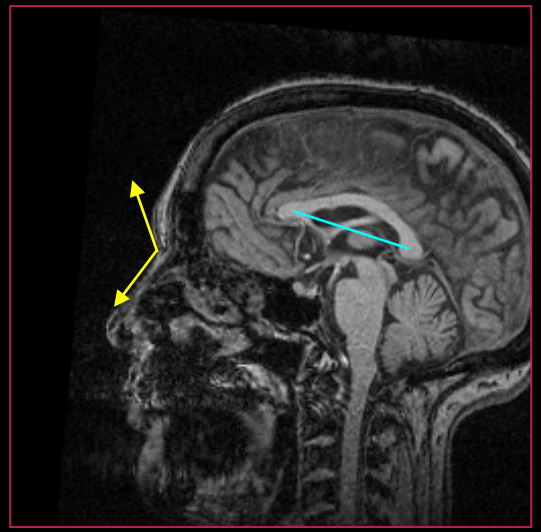




Similarity transformation

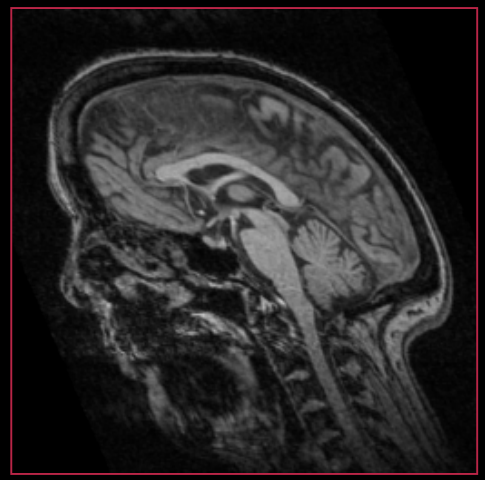
- Translation, rotation, and uniform scaling
- Angles are kept
- Distances change





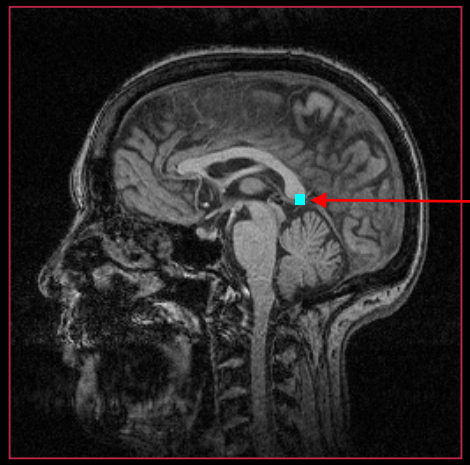
Shearing

- Pixel shifted horizontally or/and vertically
- Shearing factors
  - X-shear factor B_x
  - Y-shear factor B_y
- Is less used than translation, rotation, and scaling

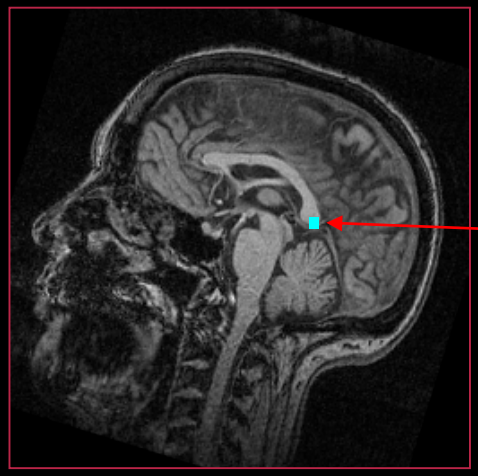


Transformation Mathematics

f



g



- Transformation of *positions*
- Structure found at position (x,y) in the input image f
- Now at position (x',y') in output image g
- A *mapping function* is needed

$$x' = A_x(x, y)$$
$$y' = A_y(x, y)$$

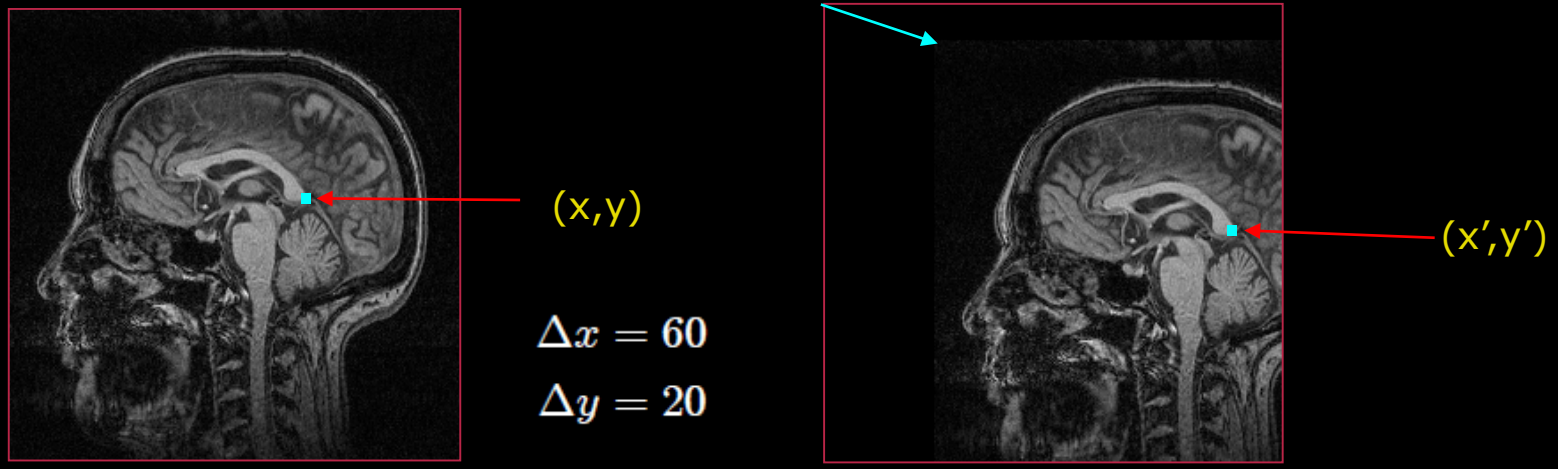
Depends on both x and y!

Translation mathematics

- The image is shifted – both vertically and horizontally

$$x' = x + \Delta x$$

$$y' = y + \Delta y$$



Matrix notation

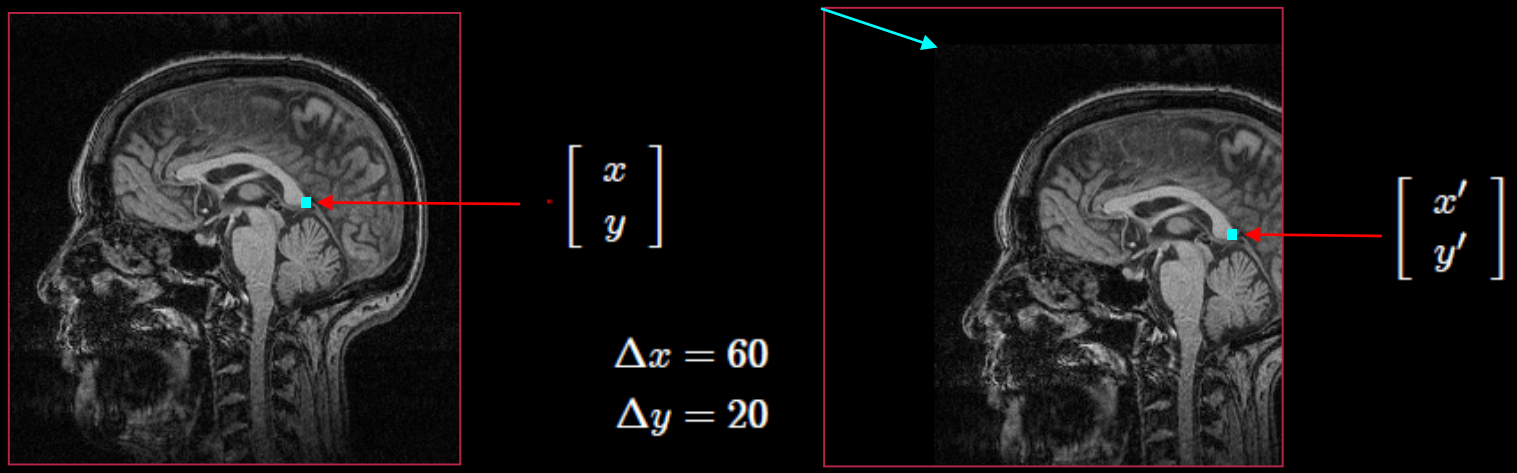
- Coordinates in column matrix format

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

Translation mathematics in matrix notation

- The image is shifted – both vertically and horizontally

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

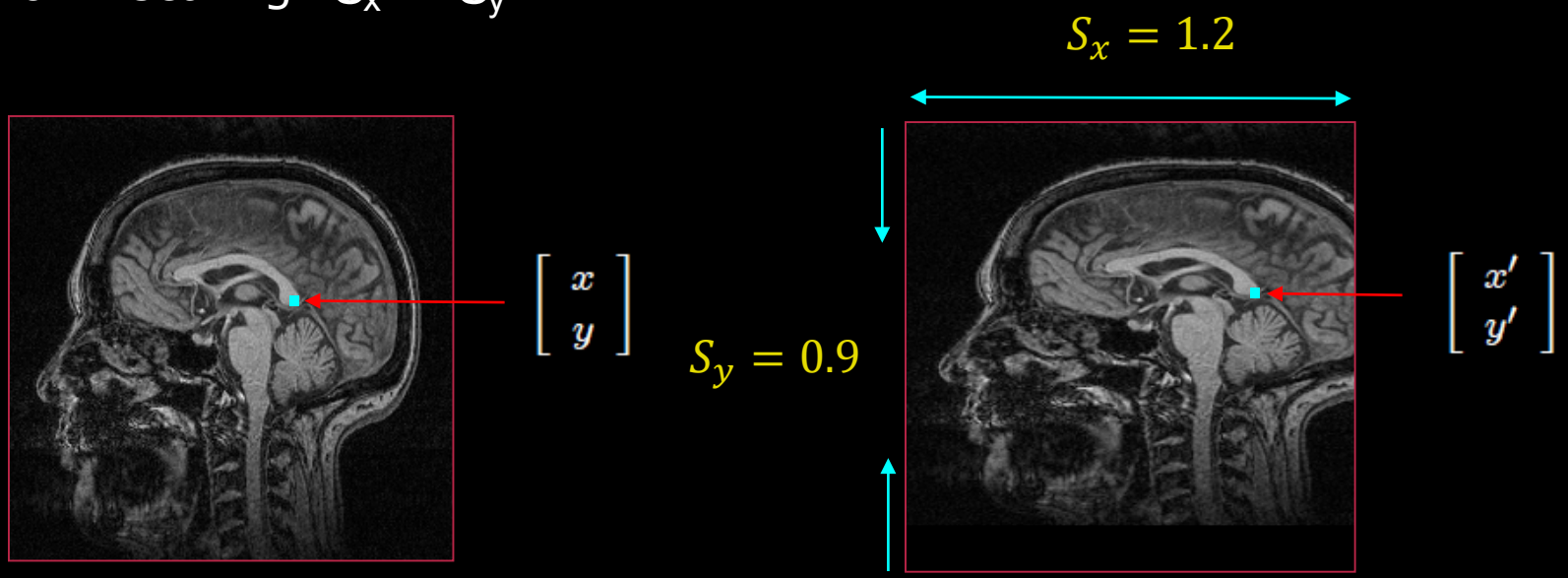


Scaling

- The size of the image is changed
- Scale factors
  - X-scale factor S_x
  - Y-scale factor S_y

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

- Uniform scaling: $S_x = S_y$



Matrix multiplication details

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Is equal to:

$$x' = x \cdot S_x$$
$$y' = y \cdot S_y$$

Transformation matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Can be written as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

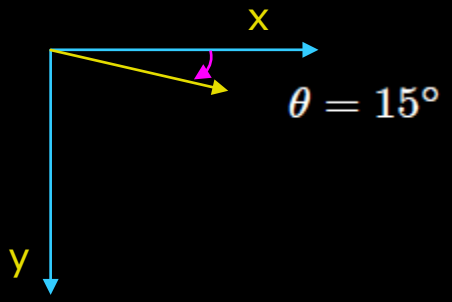
Where

$$\mathbf{A} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

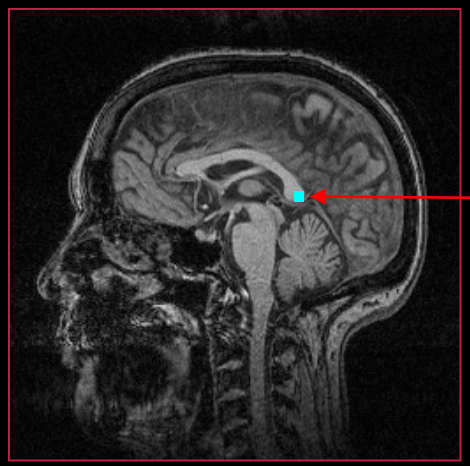
is a *transformation matrix*

Rotation

- A rotation matrix is used

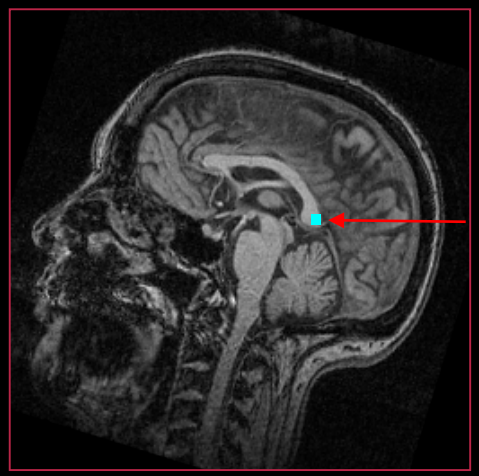


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$



$$\begin{bmatrix} x \\ y \end{bmatrix}$$

$$\theta = 15°$$



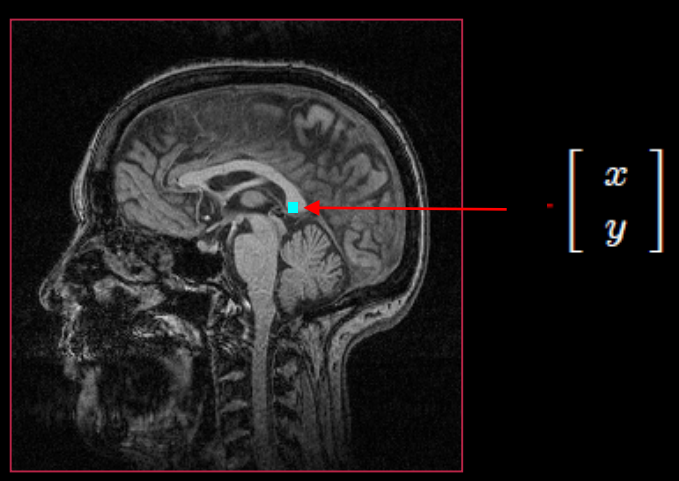
$$\begin{bmatrix} x' \\ y' \end{bmatrix}$$

Shearing

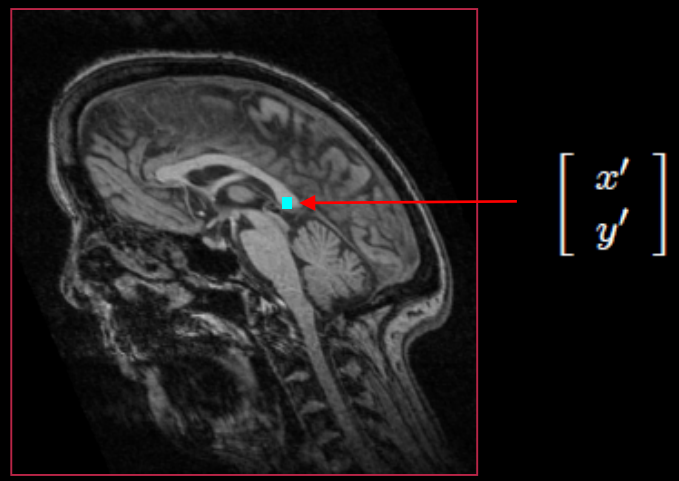
- Pixel shifted horizontally or/and vertically

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & B_x \\ B_Y & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

New x value depends on x and y



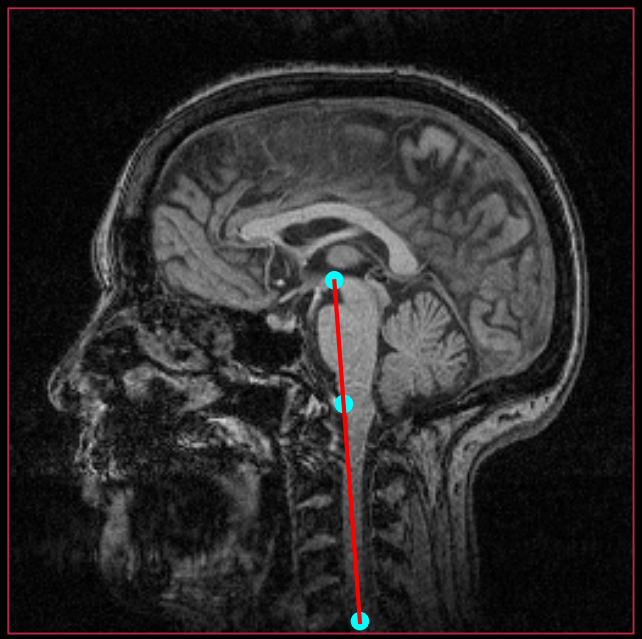
$$\cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

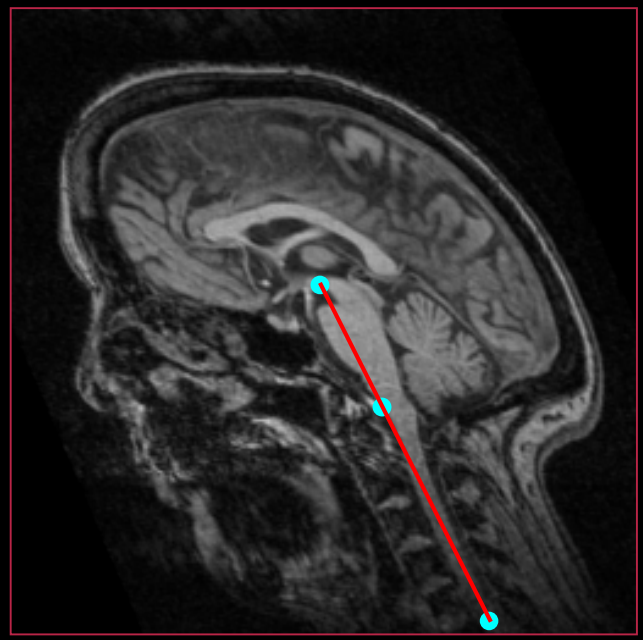


$$\begin{bmatrix} x' \\ y' \end{bmatrix}$$

Affine transformation

- The collinearity relation between points, i.e., three points which lie on a line continue to be collinear after the transformation





Combining transformations

Scaling $S_x = S_y = 1.10$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation $\theta = 5°$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

- Suppose you first want to rotate by 5 degrees and then scale by 10%

How do we combine the transformations?

Combining transformations

- Combination is done by matrix multiplication

Scaling

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Combined

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Combining transformations

- Compact notation

Scaling

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{A}_S \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

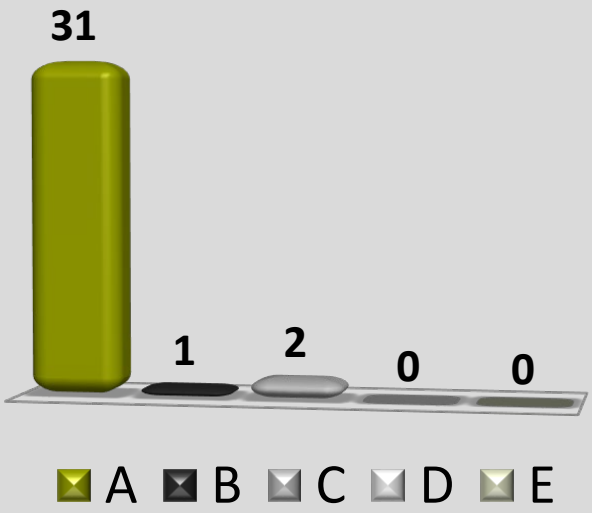
Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{A}_R \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Combined

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{A}_S \cdot \mathbf{A}_R \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Combining transforms



The point (x,y)=(5,6) is transformed. First with:

$$\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \tag{2}$$

and then with:

$$\begin{bmatrix} 0.9239 & 0.3827 \\ -0.3827 & 0.9239 \end{bmatrix} \tag{3}$$

The result is:

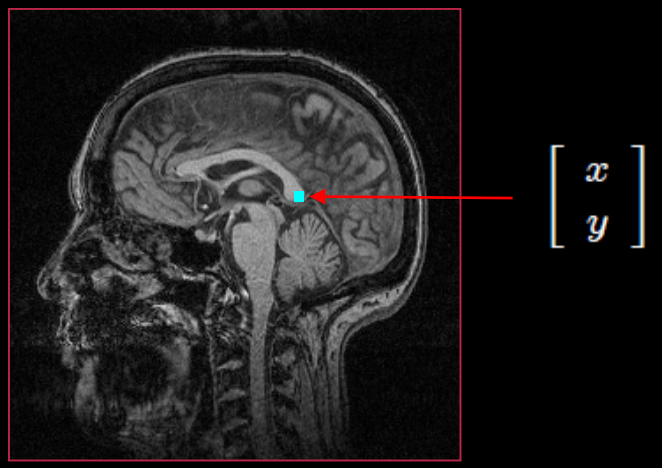
1. (16.12, 12.8)
2. (2.35, 20.46)
3. (11.3, 1.21)
4. (-1.2, 3.13)
5. (-30.8, 24.21)
6. Ved ikke

What do we have now?

- We can pick a position in the input image f and find it in the output image g

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

We can transfer one pixel – what about the whole image?



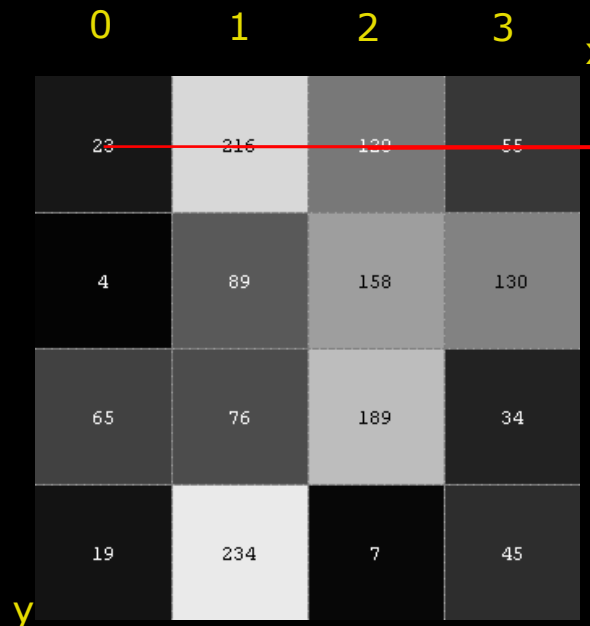
f

g

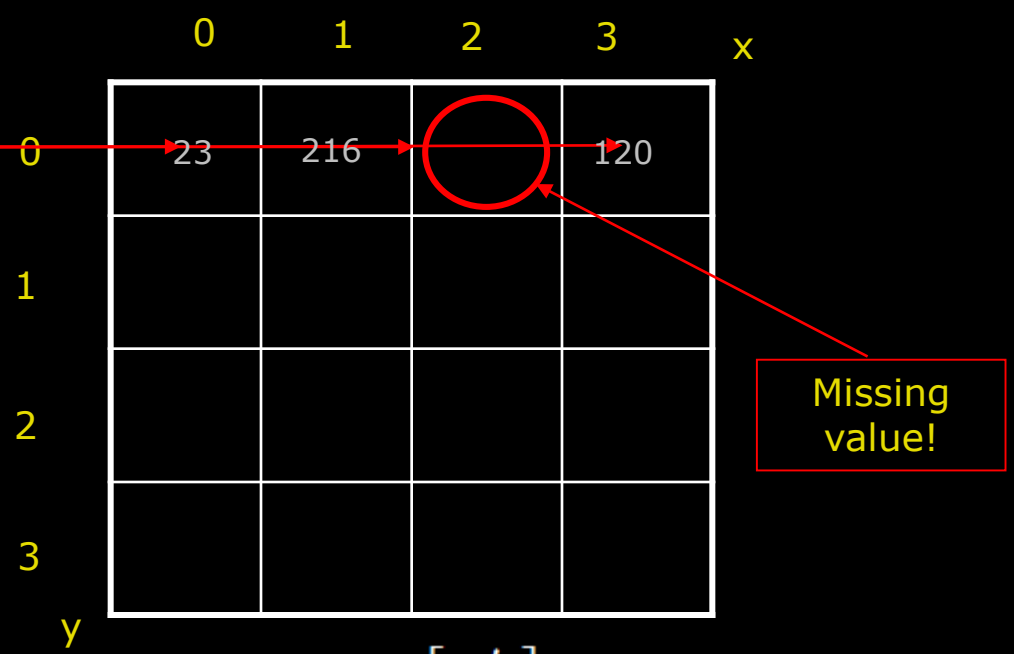
Solution 1 : Input-to-output

- Run through all pixel in input image
- Find position in output image and set output pixel value

Scaling example $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1.5 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$



f $\begin{bmatrix} x \\ y \end{bmatrix}$



g $\begin{bmatrix} x' \\ y' \end{bmatrix}$

Input-to-Output

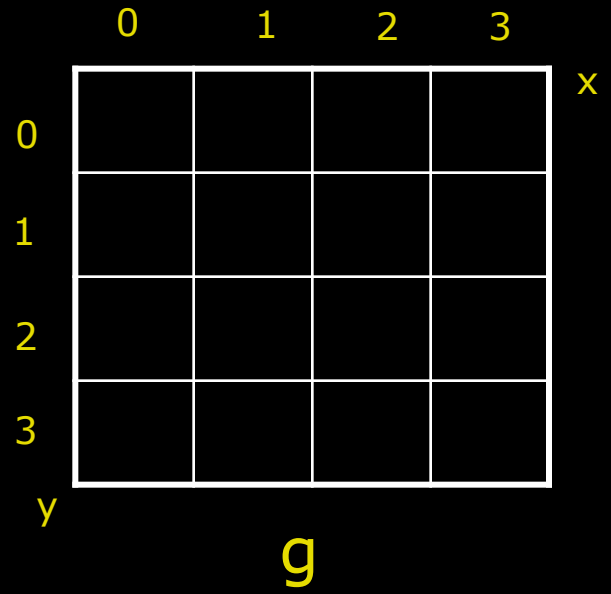
- The input to output transform is not good!
- It creates holes and other nasty looking stuff
- What do we do now?

Some observations

- We want to fill all the pixels in the output image
  - Not just the pixels that are "hit" by the pixels in the input image
- Run through all pixels in the output image?
  - Pick the relevant pixels in the input image?

We need to go "backwards"

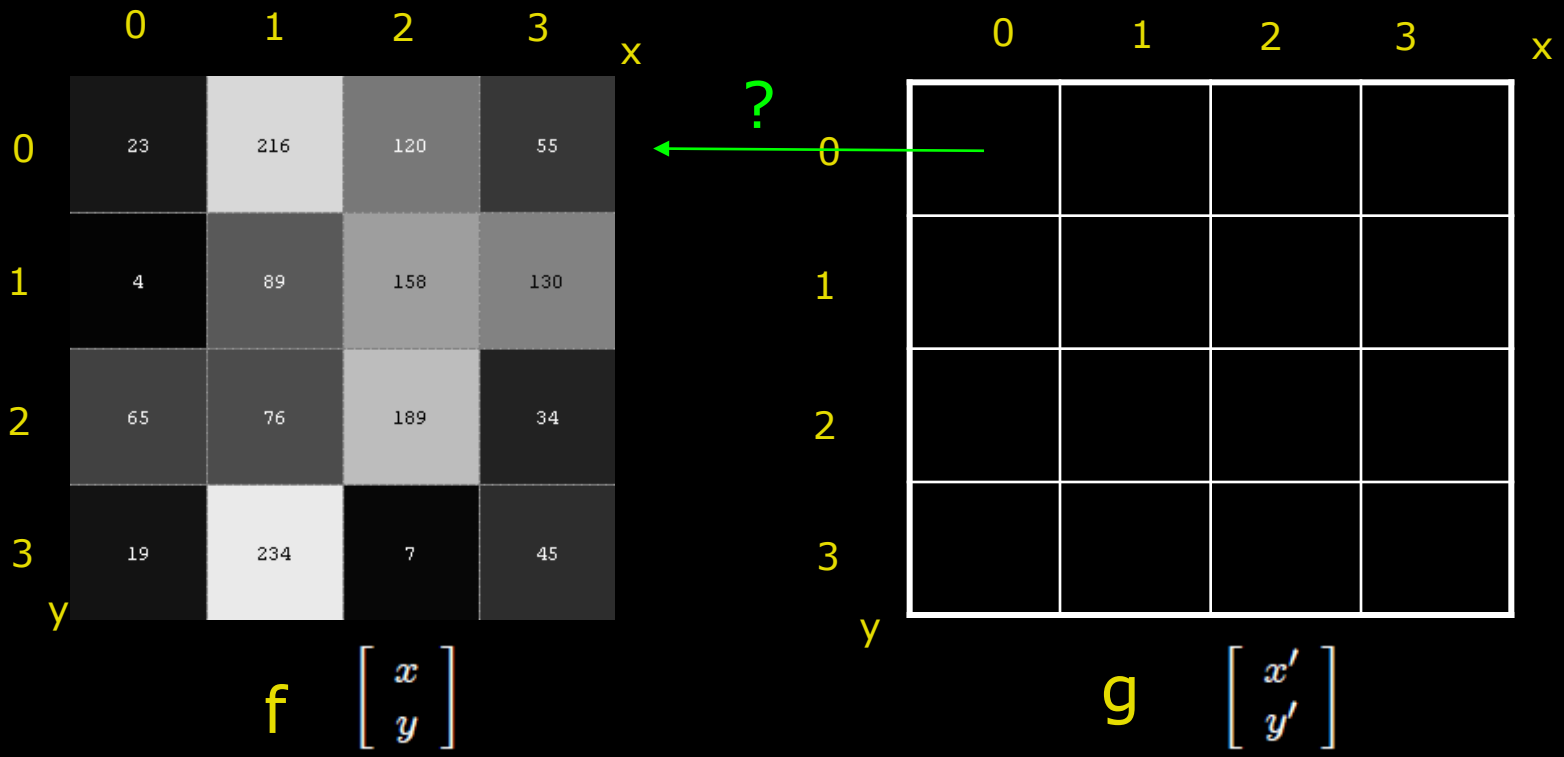
From the output to the input



Inverse transformation

- We want to go from the output to the input

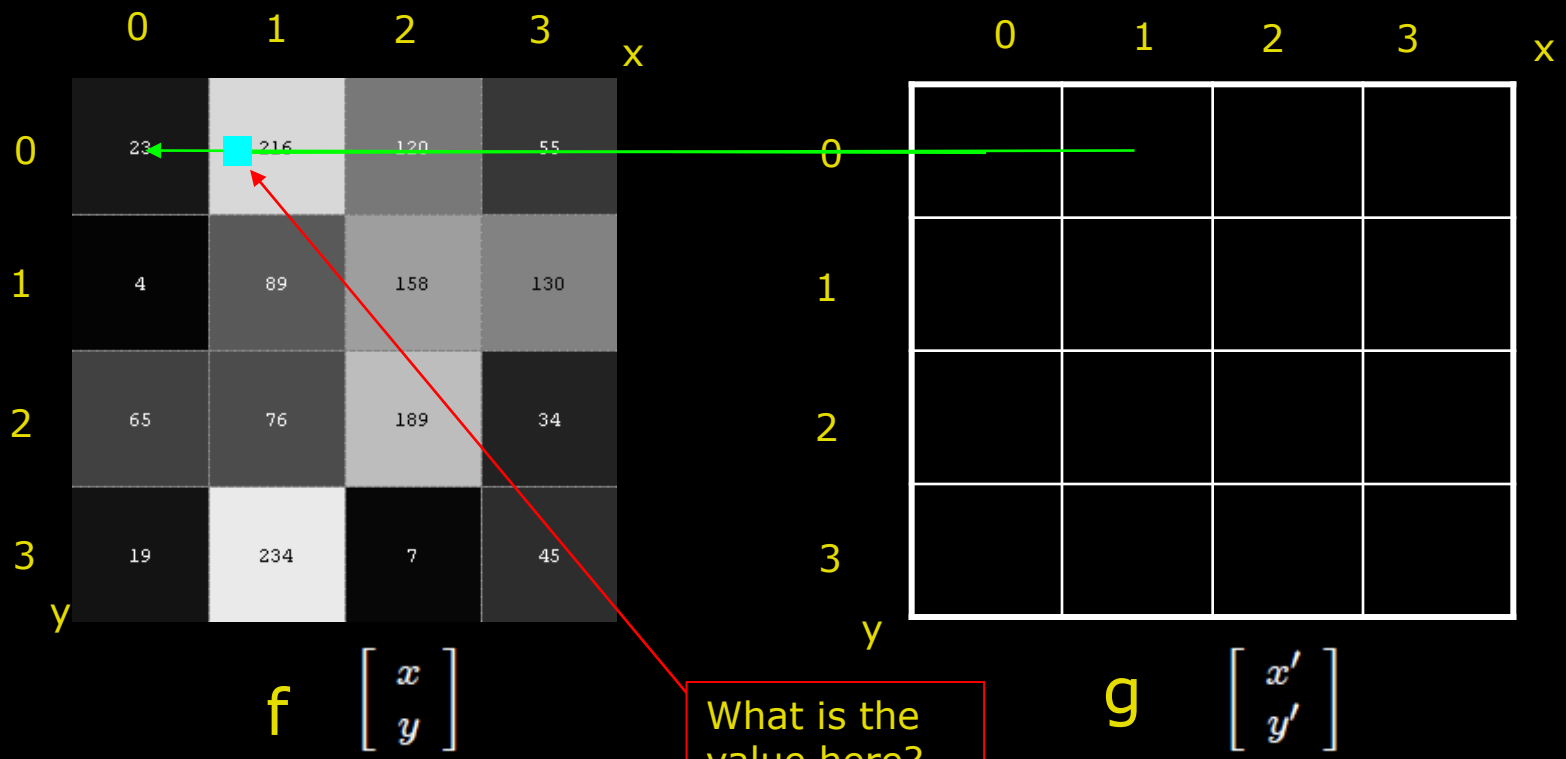
Scaling example $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1.5 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$ inverse $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/1.5 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \end{bmatrix}$



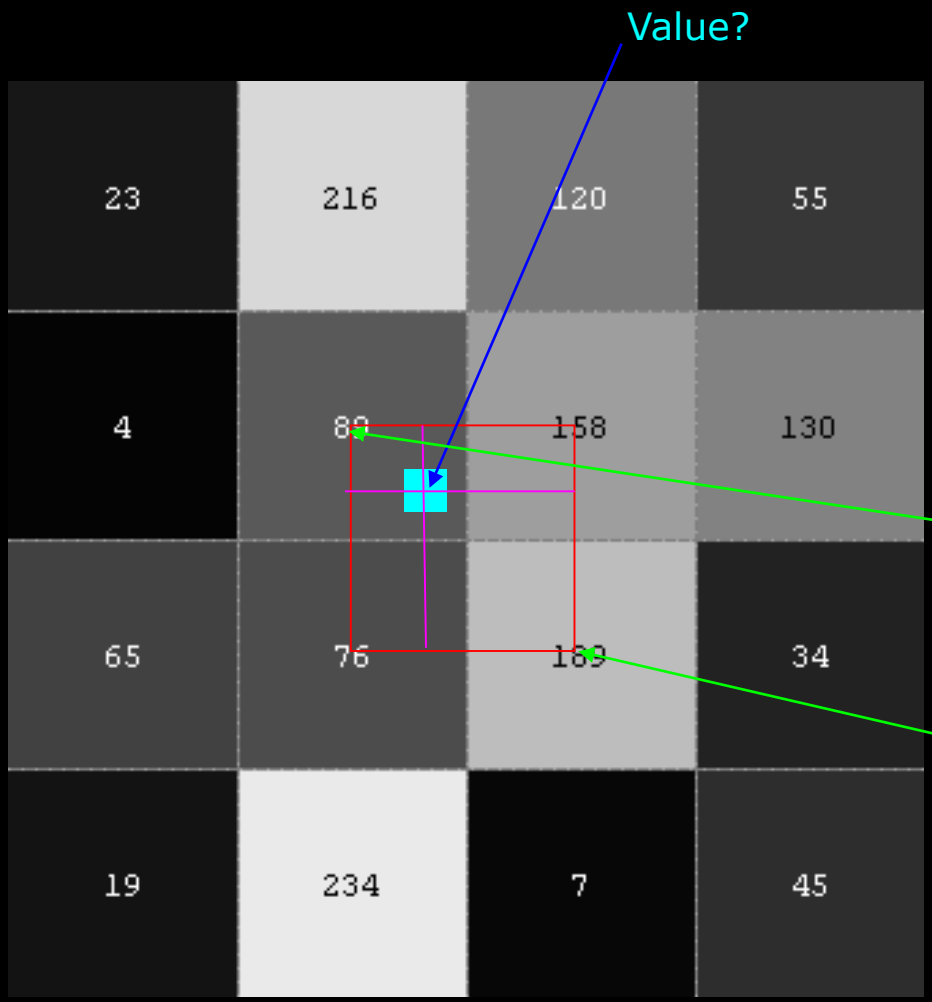
Output-to-input transformation
Backward mapping

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/1.5 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \end{bmatrix}$$

- Run through all pixel in output image
- Find position in input image and *get the value*



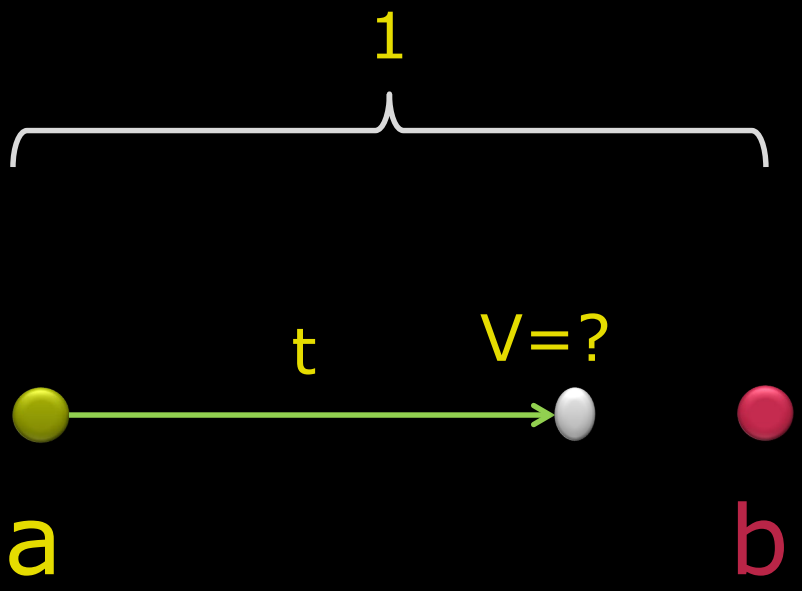
Bilinear Interpolation



- The value is calculated from 4 neighbours
- The value is based on the distance to the neighbours

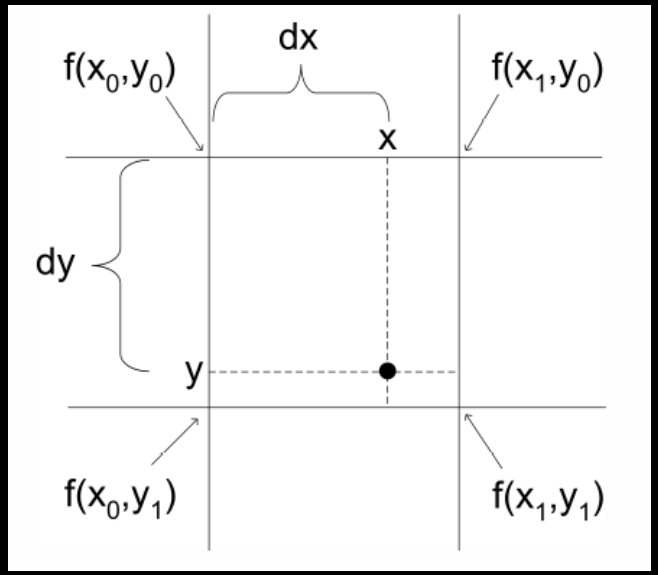
Linear Interpolation

$$v = tb + (1 - t)a$$

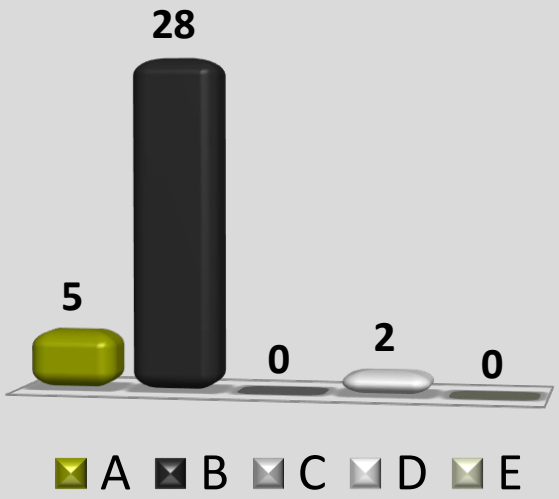


Bilinear interpolation

$$
\begin{aligned}
g(x', y') \quad = \quad & f(x_0, y_0) \cdot (1 - dx)(1 - dy) + \\
& f(x_1, y_0) \cdot (dx)(1 - dy) + \\
& f(x_0, y_1) \cdot (1 - dx)(dy) + \\
& f(x_1, y_1) \cdot (dx \cdot dy) \ ,
\end{aligned}
$$



Bilinear interpolation



Bilinear interpolation is used to create a line profile from an image. In a given point (x,y) = (173.1, 57.8), the four nearest pixels are:

x	y	værdi
173	57	110
174	57	140
173	58	156
174	58	101

What is the interpolated value in the point:

1. 131
2. 143
3. 128
4. 151
5. 139
6. Ved ikke

Output-to-input transformation
Backward mapping

- Run through all the pixel in the output image
- Use the inverse transformation to find the position in the input image
- Use bilinear interpolation to calculate the value
- Put the value in the output image

Inverse transformation

Scaling

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1.5 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

- We can calculate the inverse transformation for the scaling
- What about the others?

Inverse

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/1.5 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \end{bmatrix}$$

General inverse transformation

Affine transformation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

- The transformation is expressed as a transformation matrix A
- The *matrix inverse* of A gives the inverse transformation

Inverse transformation

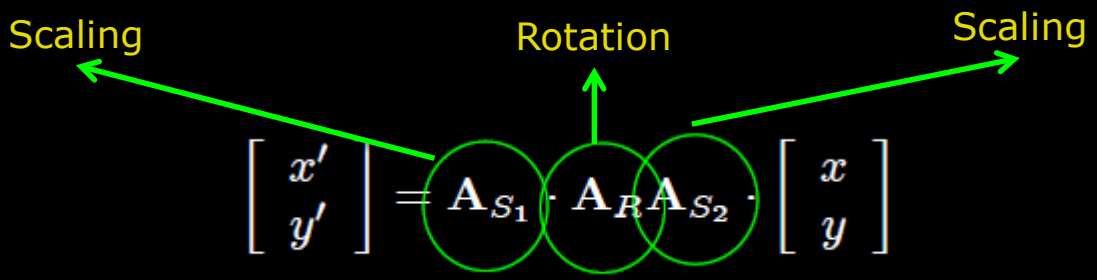
$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{A}^{-1} \cdot \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Where

$$\mathbf{A}^{-1} \cdot \mathbf{A} = \mathbf{I}$$

General inverse transformation

You can create an arbitrary sequence of rotation, scaling, and shearing



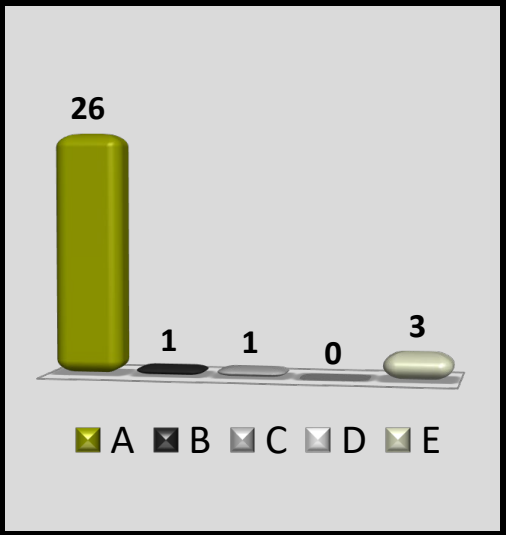
Combined:

$$\mathbf{A} = \mathbf{A}_{S_1} \cdot \mathbf{A}_R \mathbf{A}_{S_2}$$

Inverse:

$$\mathbf{A}^{-1}$$

Transformation



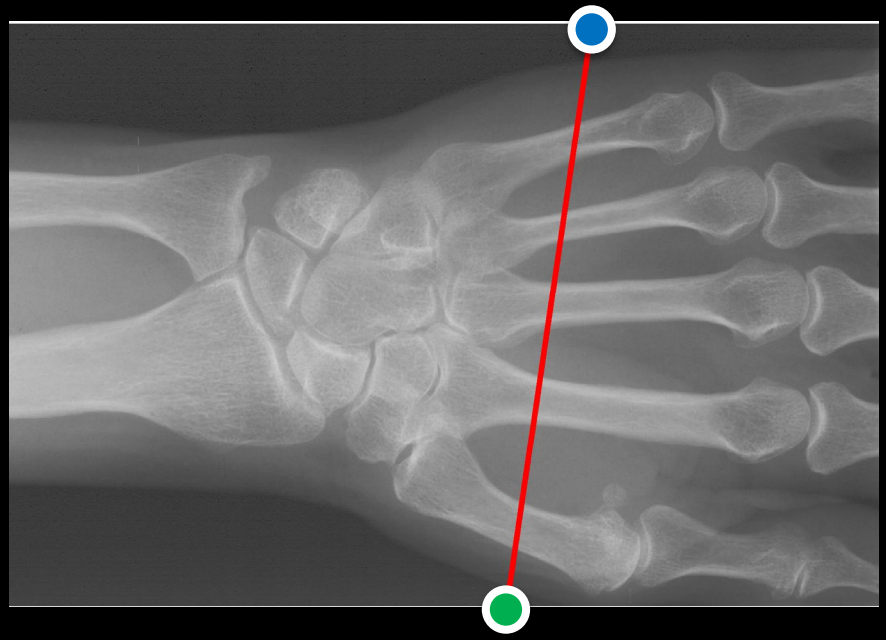
The point (x,y) = (45, 23) is transformed using:

$$\begin{bmatrix} 0.5 & 2 \\ 2 & 0.8 \end{bmatrix} \quad (1)$$

And the result is translated with (-15, 20). The result is:

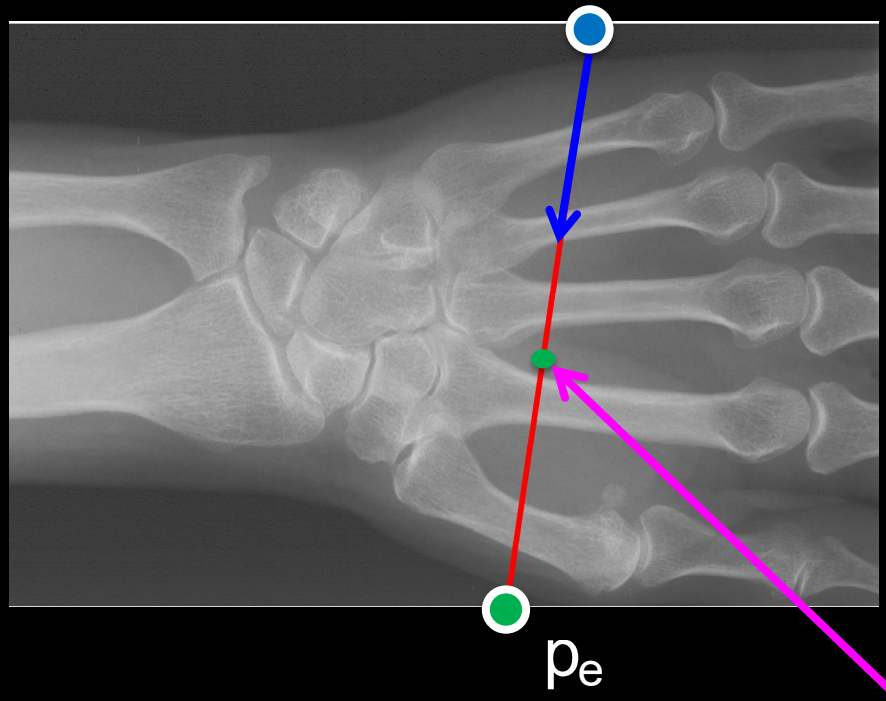
1. (53.5, 128.4)
2. (3.4, -10.3)
3. (45.3, 80.2)
4. (150.8, 32.4)
5. (-20.5, 22.6)
6. Ved ikke

Profile Analysis



- Sample the pixel values under the profile
- From start point to end point

Profile Analysis



- Length of profile

$$l = \|p_e - p_s\|$$

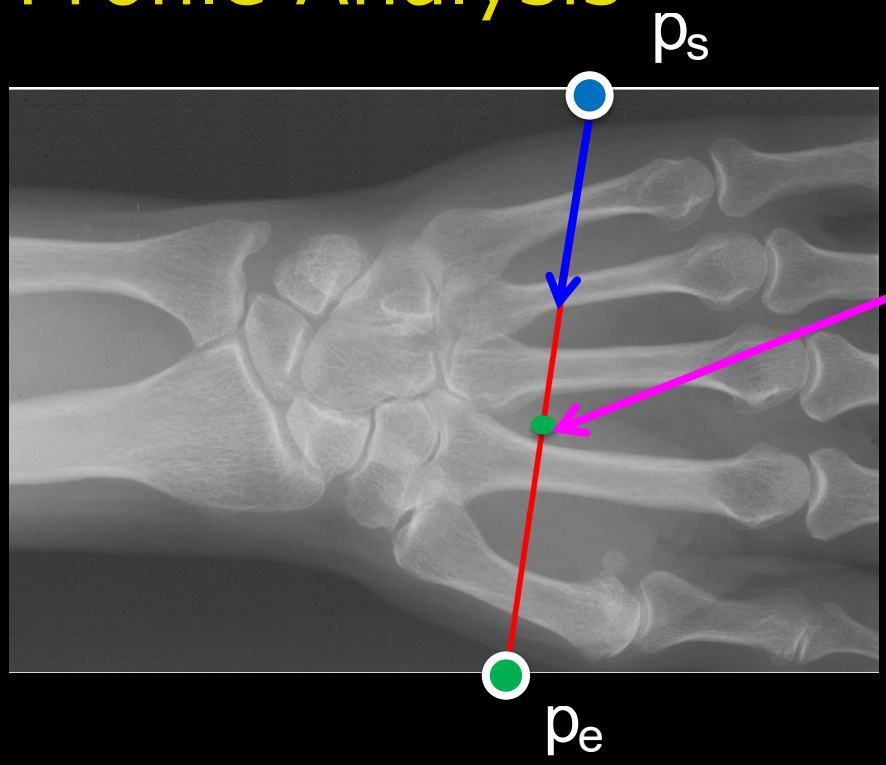
- Unit vector

$$\vec{u} = \frac{p_e - p_s}{l}$$

- Point on profile

$$p = p_s + i \cdot \vec{u}$$

Profile Analysis

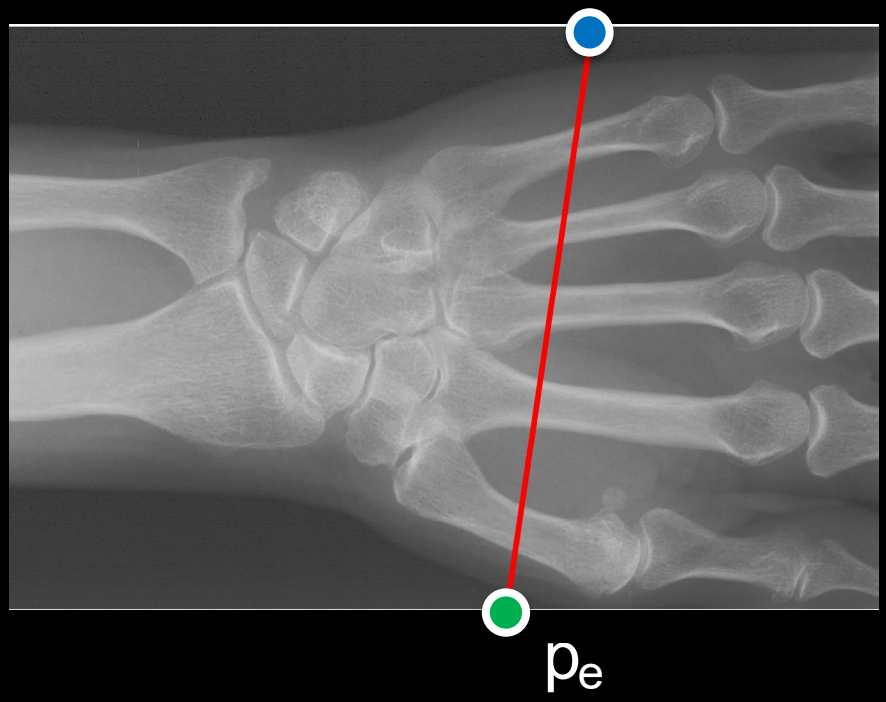


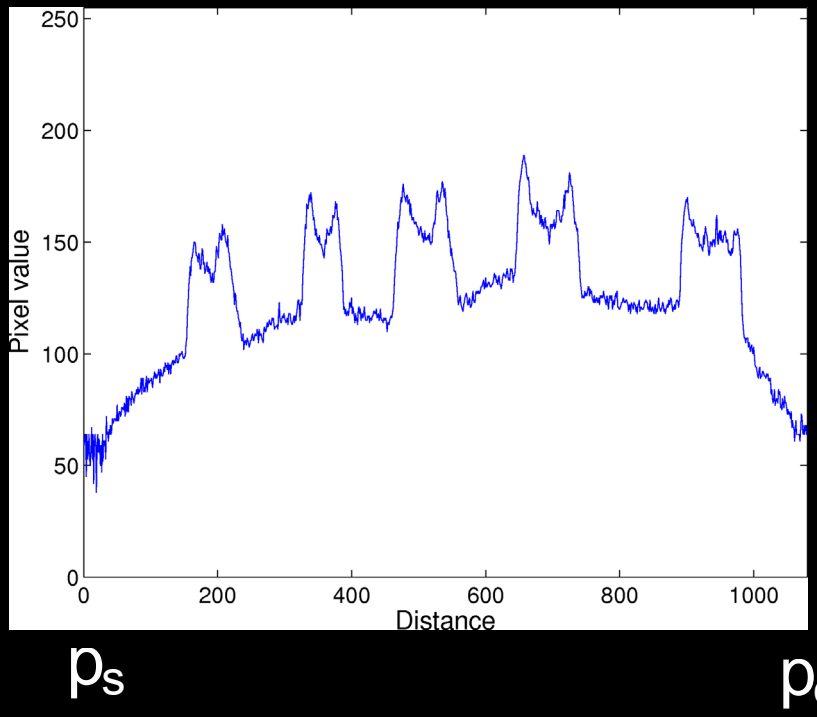
- Point on profile

$$p = p_s + i \cdot \vec{u}$$

- Sample all points on profile using bi-linear interpolation

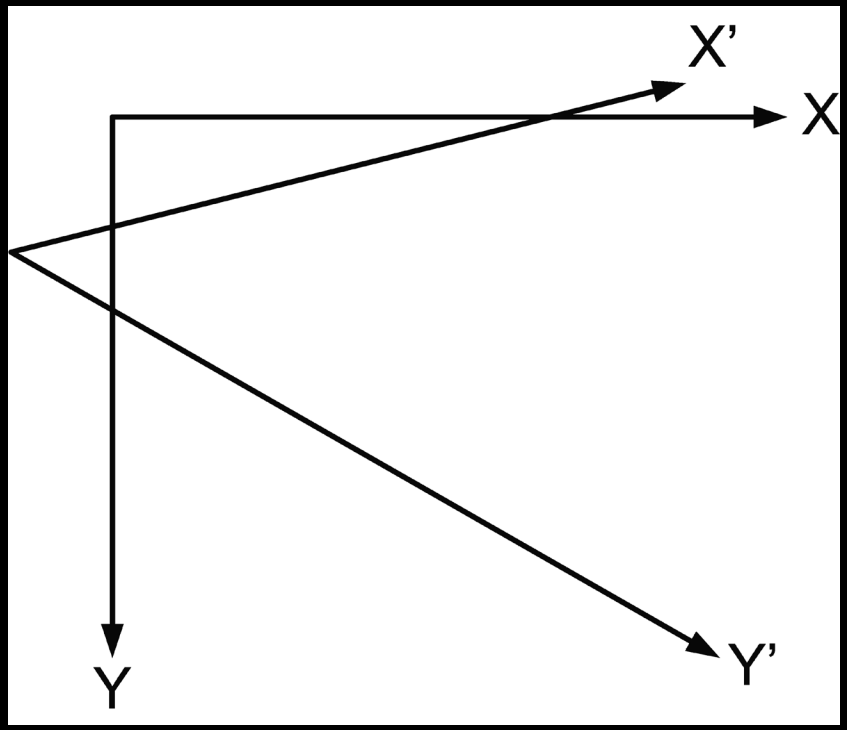
Profile Analysis





Homography

Homography



Mapping from coordinates (x,y) to (x', y')

- Maps from one plane to another plane
- Projective transformation

$$\begin{bmatrix} h \cdot x' \\ h \cdot y' \\ h \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Virtual Billboard Example

https://www.learnopencv.com/homography-examples-using-opencv-python-c/

Virtual Billboard Example

$$\begin{bmatrix} h \cdot x' \\ h \cdot y' \\ h \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

https://www.learnopencv.com/homography-examples-using-opencv-python-c/

Virtual Billboard Example

One point in each image gives:

$$\begin{bmatrix} h \cdot x' \\ h \cdot y' \\ h \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\frac{h \cdot x'}{h} = x' = \frac{a_1 \cdot x + a_2 \cdot y + a_3}{c_1 \cdot x + c_2 \cdot y + 1}$$
$$\frac{h \cdot y'}{h} = y' = \frac{b_1 \cdot x + b_2 \cdot y + b_3}{c_1 \cdot x + c_2 \cdot y + 1} .$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x \cdot x' & -y \cdot x' \\ 0 & 0 & 0 & x & y & 1 & -x \cdot y' & -y \cdot y' \end{bmatrix} \cdot \vec{d} ,$$

We want to estimate:

$$\vec{d} = [a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2,]^T .$$

https://www.learnopencv.com/homography-examples-using-opencv-python-c/

Virtual Billboard Example

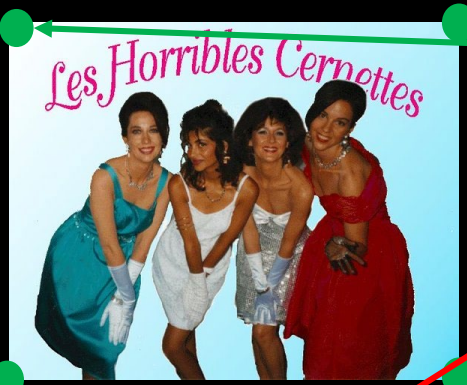
Eight unknowns:

$$\vec{d} = [a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2,]^T.$$

Requires four corresponding points

https://www.learnopencv.com/homography-examples-using-opencv-python-c/

Virtual Billboard Example



$$\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 \cdot x'_1 & -y_1 \cdot x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 \cdot y'_1 & -y_1 \cdot y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 \cdot x'_2 & -y_2 \cdot x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2 \cdot y'_2 & -y_2 \cdot y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3 \cdot x'_3 & -y_3 \cdot x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3 \cdot y'_3 & -y_3 \cdot y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4 \cdot x'_4 & -y_4 \cdot x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4 \cdot y'_4 & -y_4 \cdot y'_4 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \\ c_1 \\ c_2 \end{bmatrix}$$

Equation system:

$$\vec{e} = \mathbf{K}\vec{d}:$$

Can be solved using linear algebra (SVD):

$$\vec{d} = \mathbf{K}^{-1}\vec{e},$$

https://www.learnopencv.com/homography-examples-using-opencv-python-c/

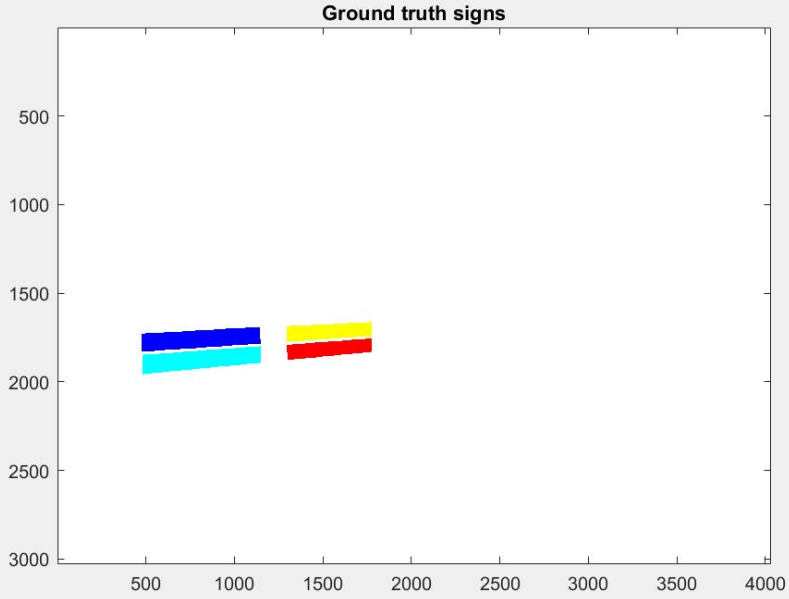
Virtual Billboard Example

- For each pixel in the output image
  - Compute the position in the billboard image using the homography
  - If position is within the billboard
    - Get the billboard pixel value and put it in the output image

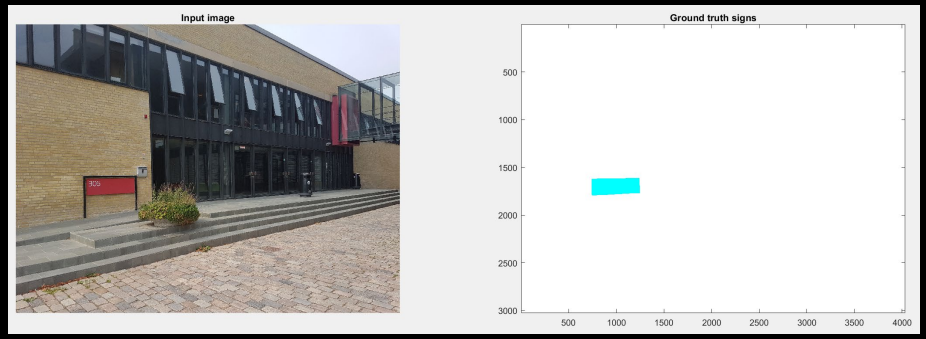
$$\begin{bmatrix} h \cdot x' \\ h \cdot y' \\ h \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

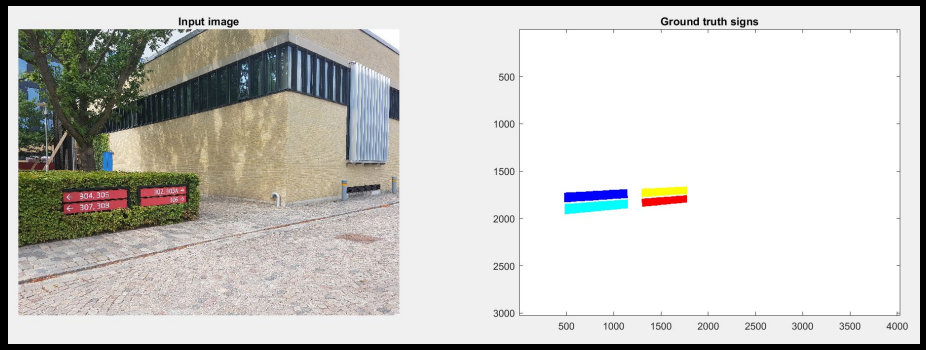
https://www.learnopencv.com/homography-examples-using-opencv-python-c/

DTU Sign Challenge – exercise 8



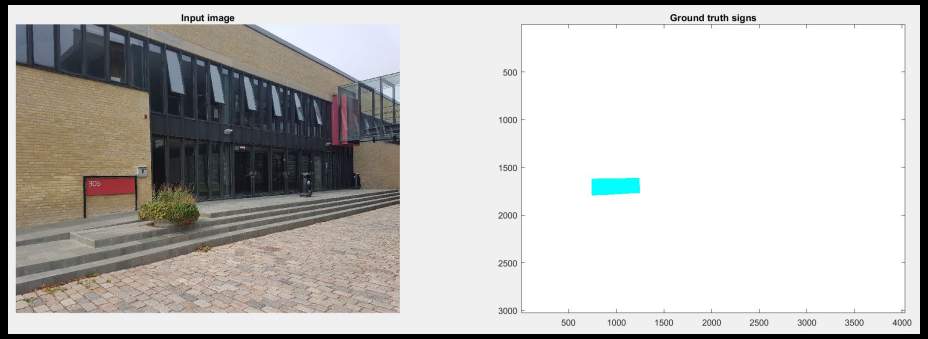
Data

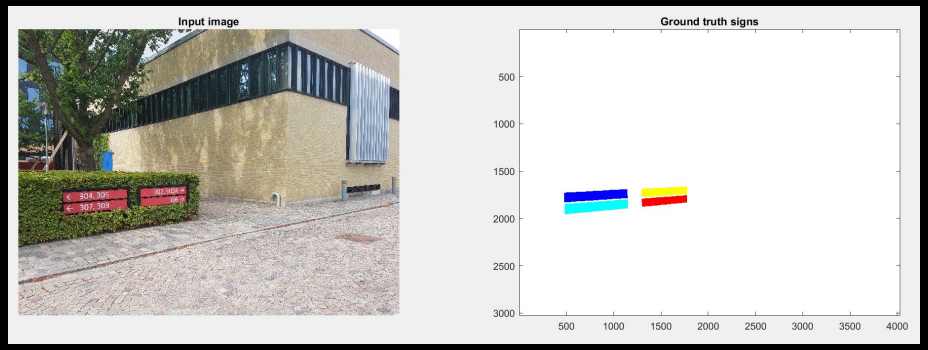




- 34 photos with ground truth (GT)

Goal





- To create a function that can locate signs
- Should create a label image
  - Label 0: Background
  - Label 1: Sign 1
  - Label 2: Sign 2
  - …

Supplied scripts and functions

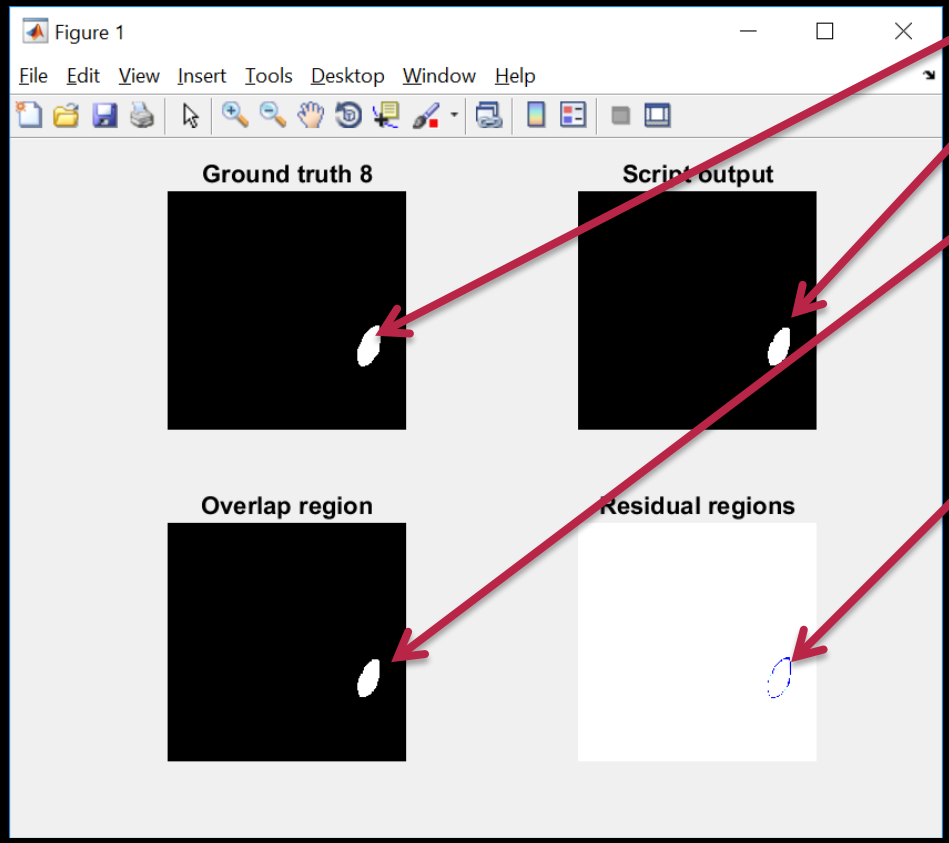
- Some scripts and functions supplied
- MyDTUSignFinder.m
  - Should be renamed into a new cool name

- You should also make a commercial for your amazing method!
  - One or two powerpoint slides!

How do we measure performance?

DICE Scores

$$DICE = \frac{2(A \cap B)}{A + B}$$

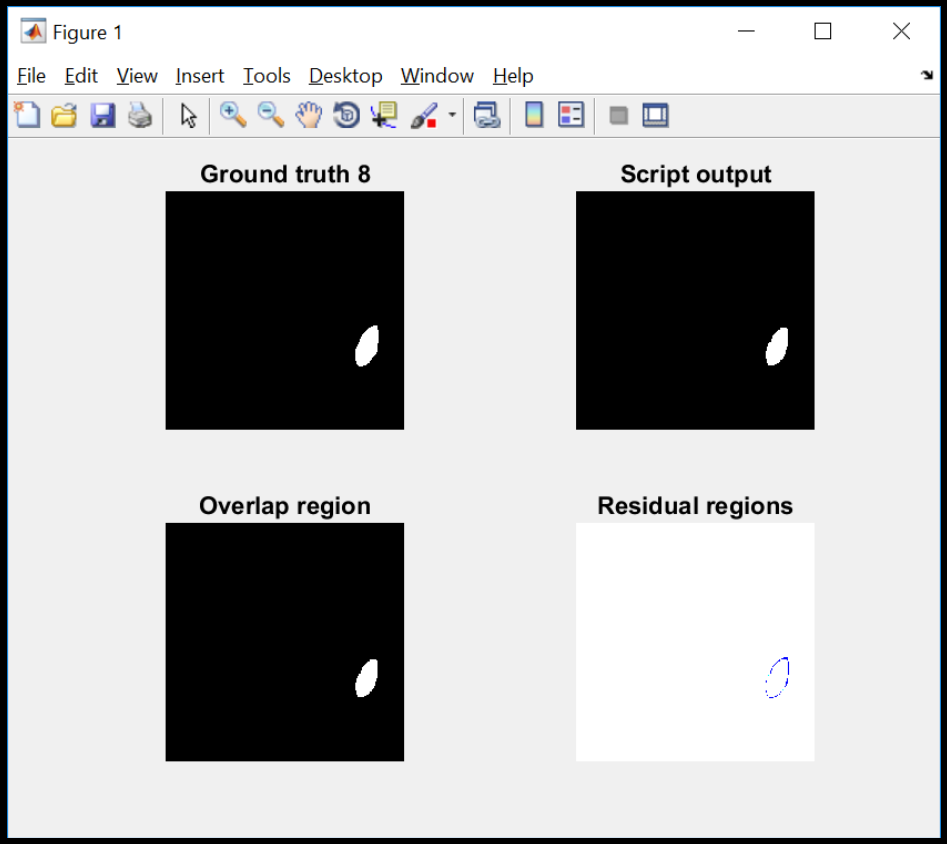


- A: area of ground truth
- B: area of your result
- $A \cap B$: area of intersection/overlap

- Residuals is the difference between ground truth and your result

DICE Scores

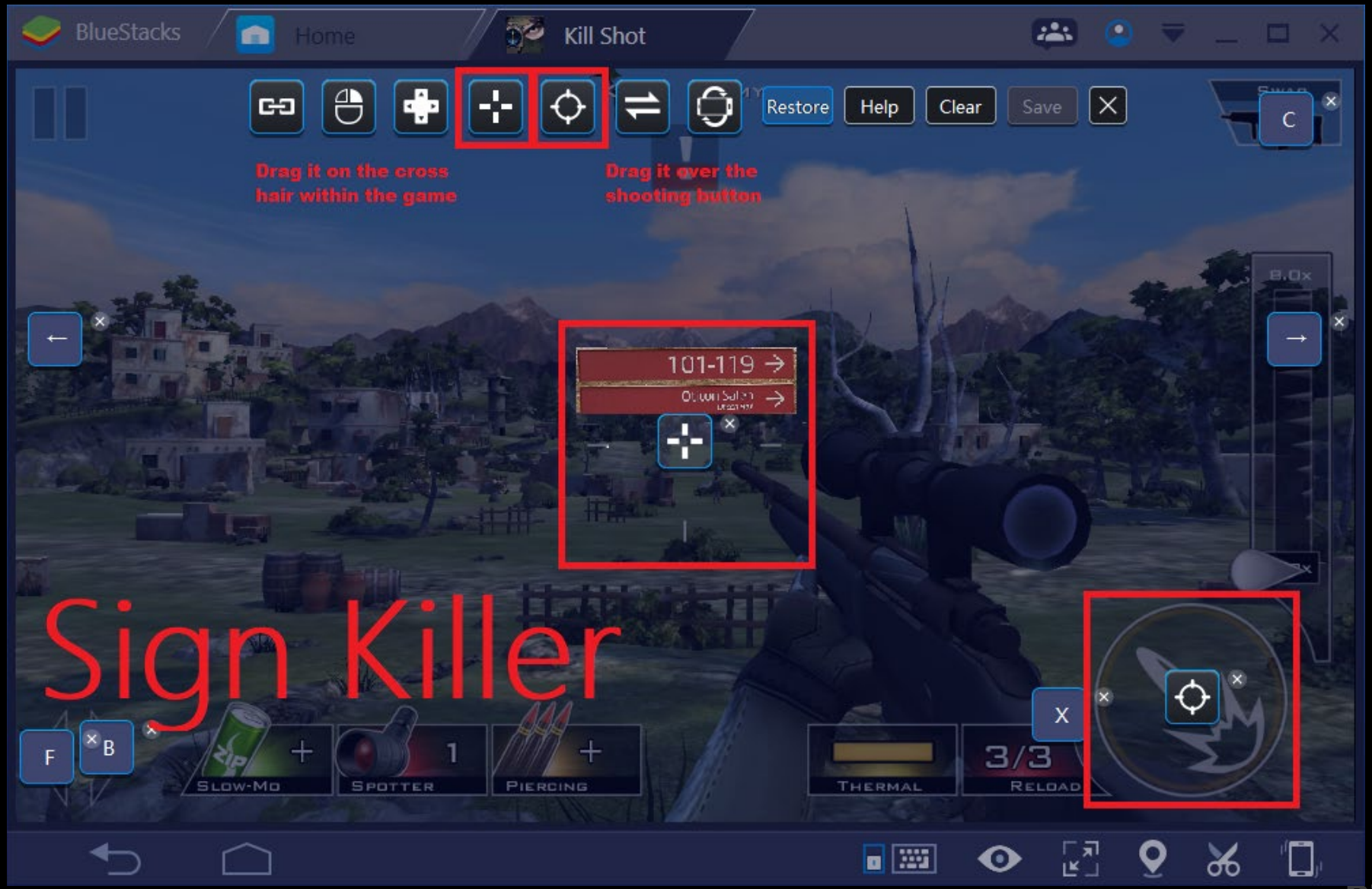
$$DICE = \frac{2(A \cap B)}{A + B}$$



- 1 is perfect
  - No residuals
- 0 is really bad
  - Only residuals

Multiple DICE scores

- Average DICE score over found matches
- Penalises
  - Too many sign candidates
  - Too few sign candidates


BlueStacks
Home
Kill Shot
Restore
Help
Clear
Save
Drag it on the cross hair within the game
Drag it over the shooting button
101-119
Sign Killer
F
B
SLOW-MO
SPOTTER
PIERCING
THERMAL
3/3
RELOAD
X
C

signeTheSignFinder

It doensn't matter if it is inside or outside of an -elephant or a –rhino's ass.

- "*We will find your sign*"

"Signs are more important than knowledge."
Albert, Einstein - German theoretical physicist (1879-1955)

"I never dreamed about signs, I worked for it."
Estée Lauder - American founder of the cosmetic company (1908-2004)

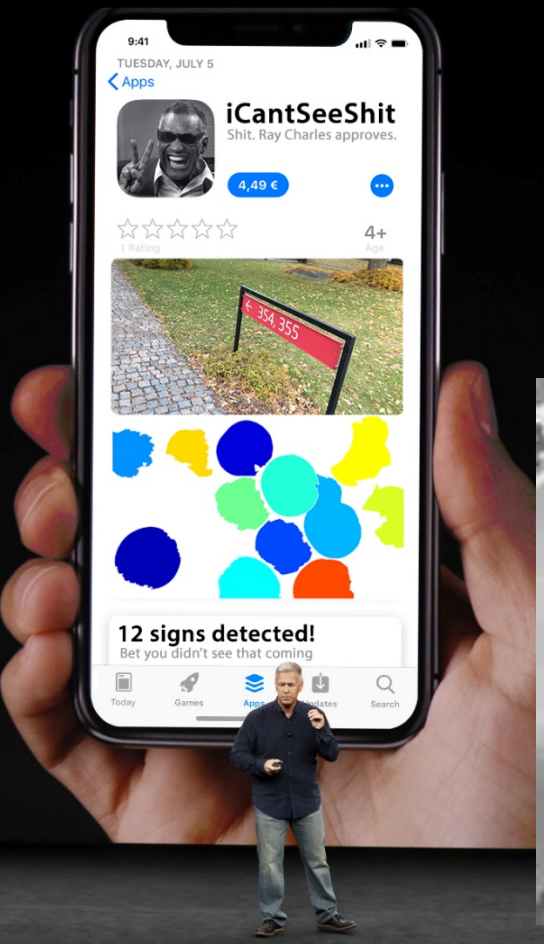
"There is nothing more powerful in the world than the sign that came in time."
Victor Hugo - French poet, prose writer, dramatist, essayist and politician (1802-1855)

"If I have seen further it is by standing on the shoulders of signs".
Isaac Newton – Physicist, mathematician, astronomer, alchemist, inventor, theologian and natural Philosopher (1634-1727)

Disclaimer: We do not promise that the current program is capable of finding a sign inside a rhino's ass.

We care about all of our users.
Introducing iCantSeeShit®

*30% reduction for the visually impaired.



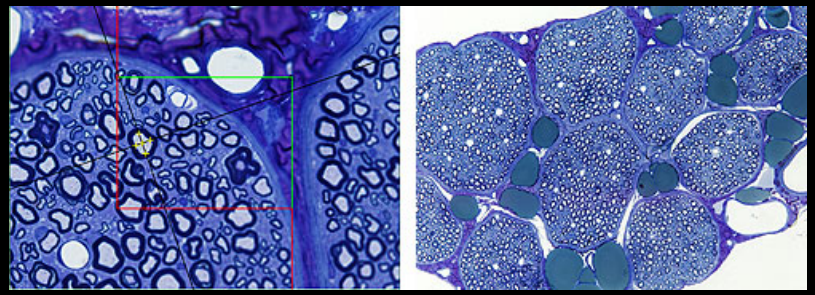
2019


It seems the ground truth shows a different result — (Batman slaps Robin and frowns disappointedly)
Robin your twat, use thresholding, opening, closing, erosion, dilation, edge dedetection, B-features, LSD, backtracking, neighbourhood processing and pixel classification !!!


It's a Match!
Matlab and DTU Sign Challenge have liked each other
MATLAB
DTU

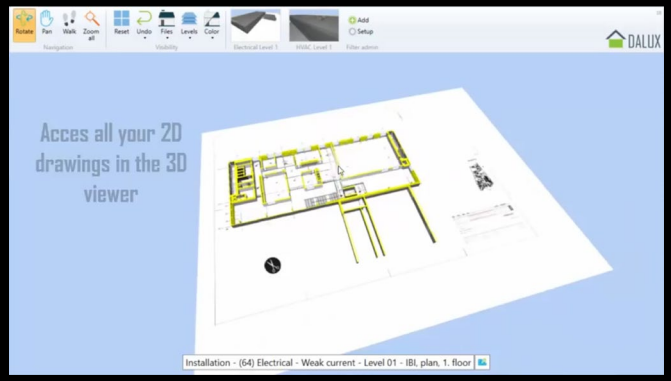
Next week – Company presentations

Presentations start at 9:30!



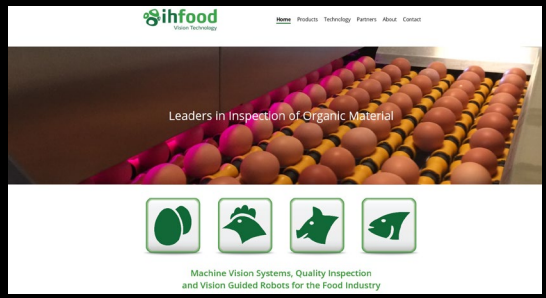
Visiopharm

JLI Vision



Dalux

Videometer



IH Food