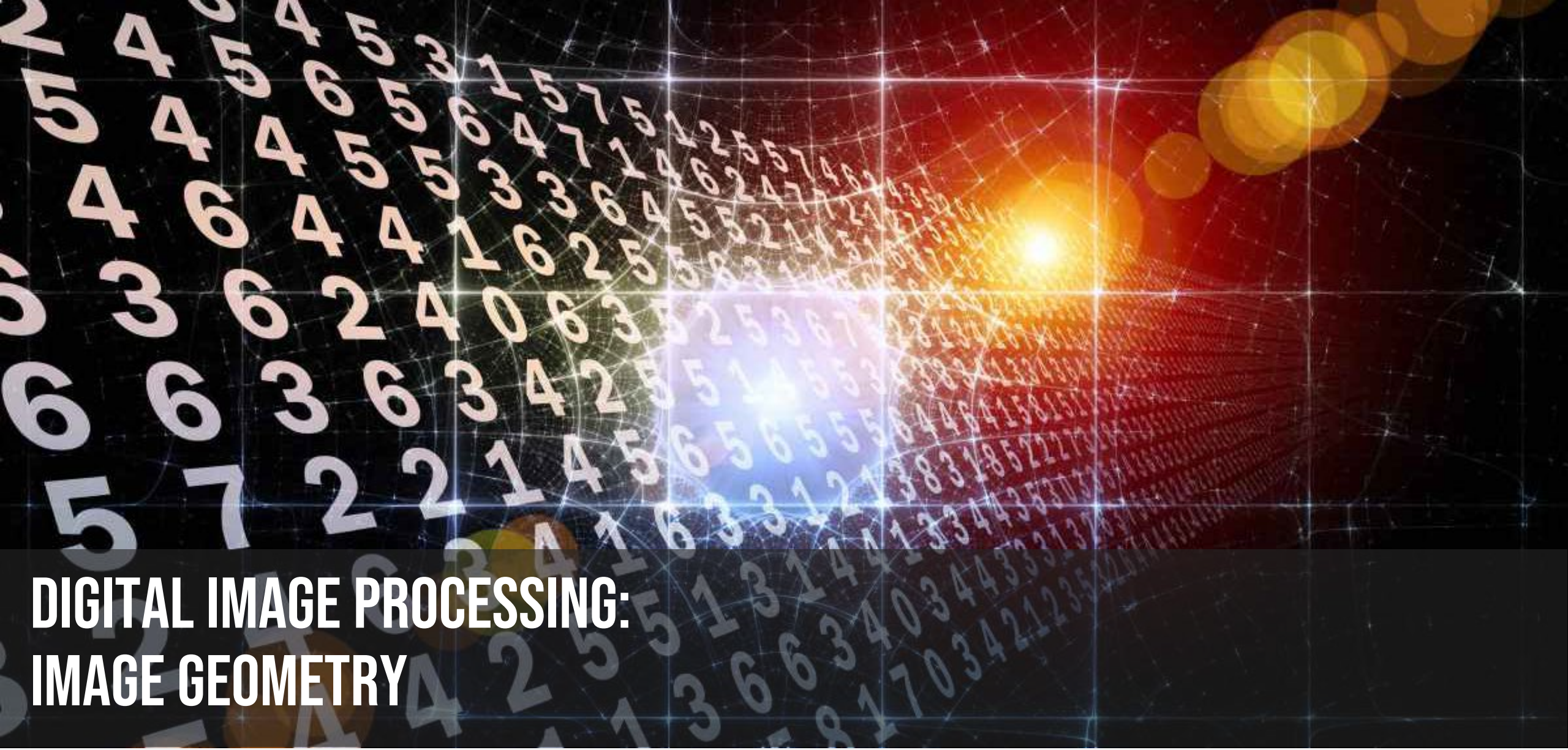
PENGOLAHAN SINYAL DIGITAL

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia



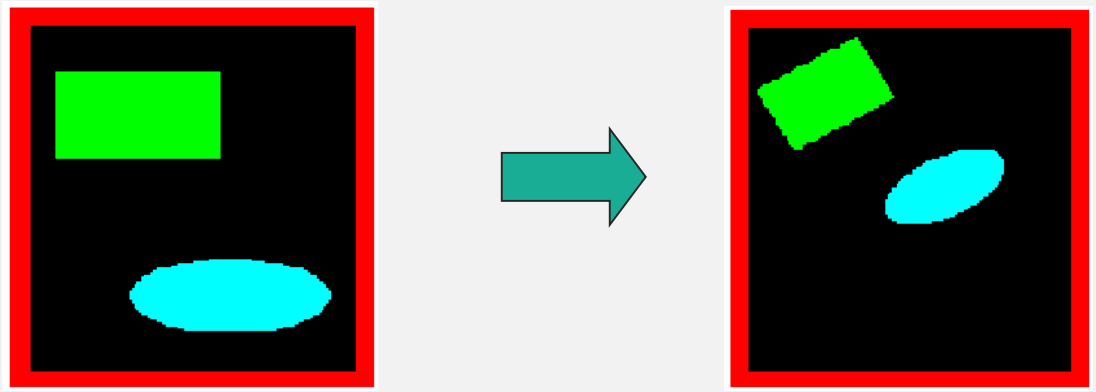
DIGITAL IMAGE PROCESSING: IMAGE GEOMETRY

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

GEOMETRIC OPERATIONS

- Scale - change image content size
- Rotate - change image content orientation
- Reflect - flip over image contents
- Translate - change image content position
- Affine Transformation - general image content linear geometric transformation

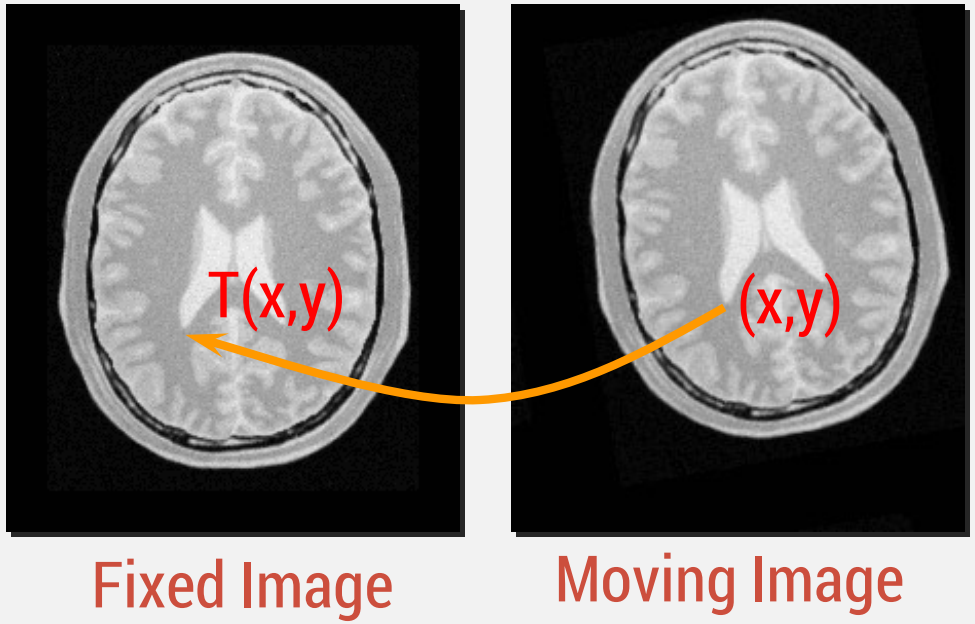


GEOMETRIC TRANSFORMATIONS

- Geometric transformations are common in computer graphics, and are often used in image analysis.
- Geometric transforms permit the elimination of geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed.
- Examples?

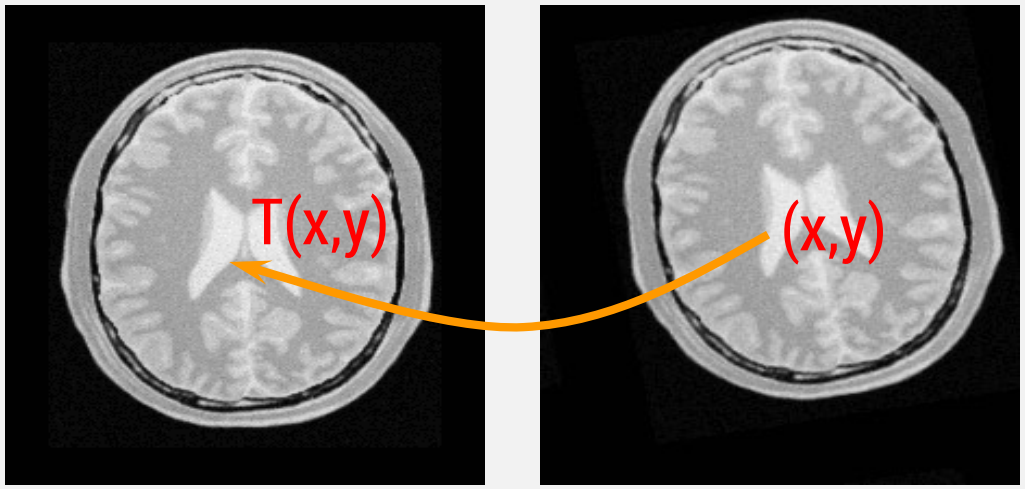
GEOMETRIC TRANSFORMATIONS

- A geometric transform consists of two basic steps ...
  - Step 1: determining the pixel co-ordinate transformation
    - mapping of the co-ordinates of the moving image pixel to the point in the fixed image.



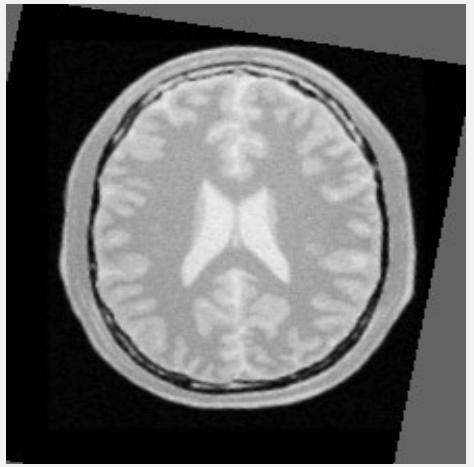
GEOMETRIC TRANSFORMATIONS

- Step 2: determining the brightness of the points in the digital grid of the transformed image.
  - brightness is usually computed as an interpolation of the brightness of several points in the neighborhood.



Fixed Image

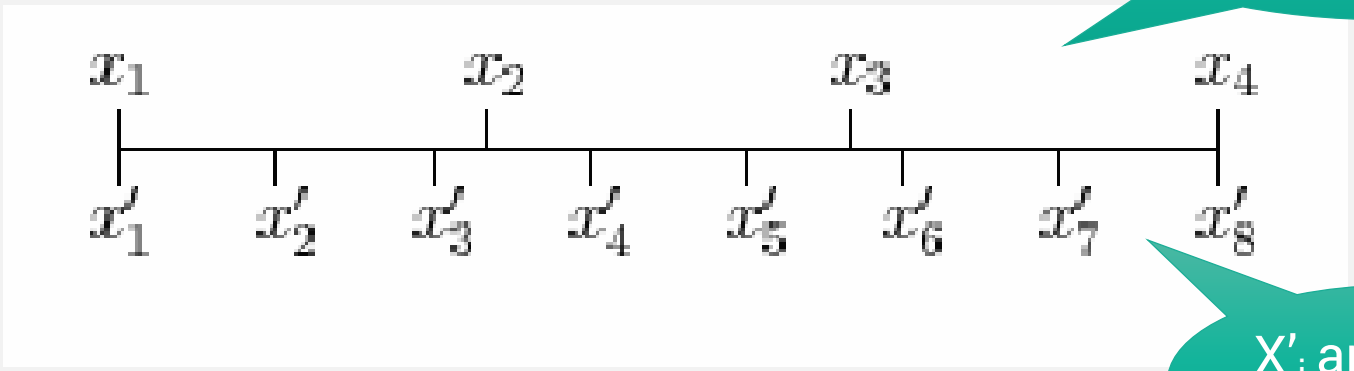
Moving Image



xformed Moving Image

INTERPOLATION OF DATA

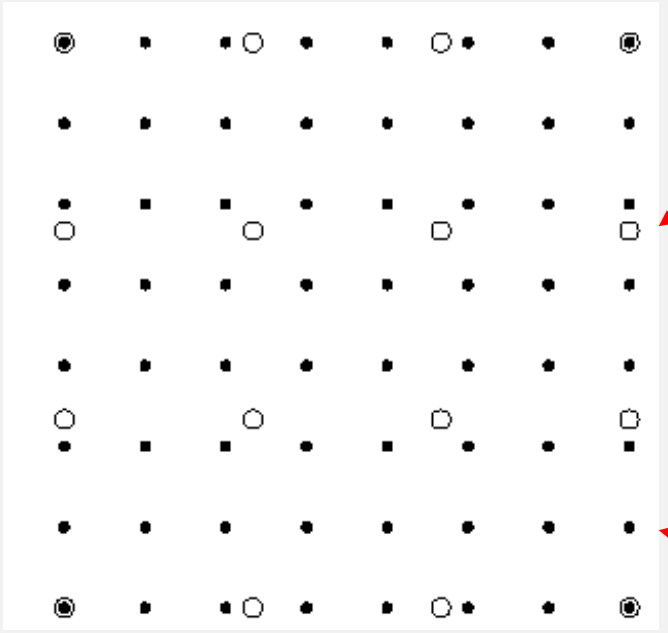
- Given a function f(x) at 4 points, how to “guess” values at other points?



- Guessing at the function values within the known range is called interpolation.
- Interpolation has great significance in general image/video processing.

ANOTHER EXAMPLE

- Interpolation on an image (4×4 → 8×8) after scaling

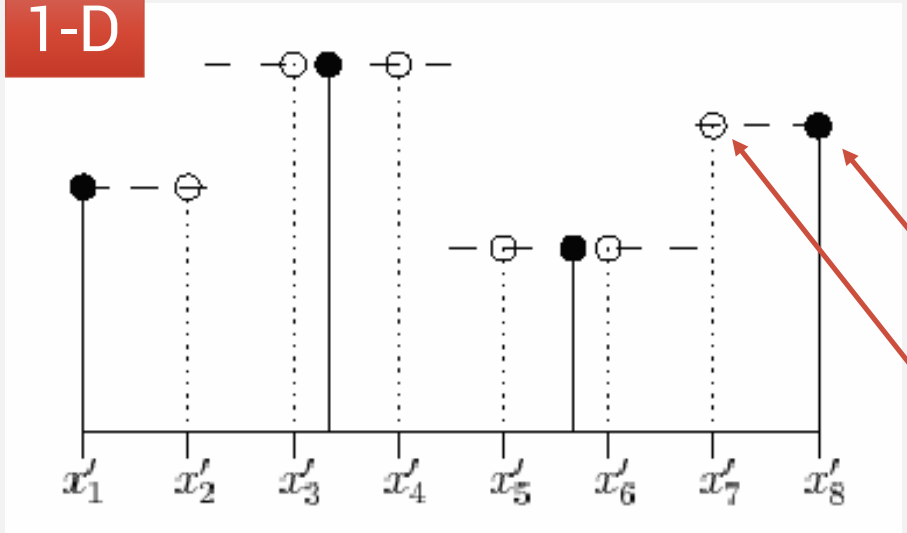


Open circle:
Original image pixel

Closed circle:
New pixels

INTERPOLATION: NEAREST NEIGHBOR

1-D



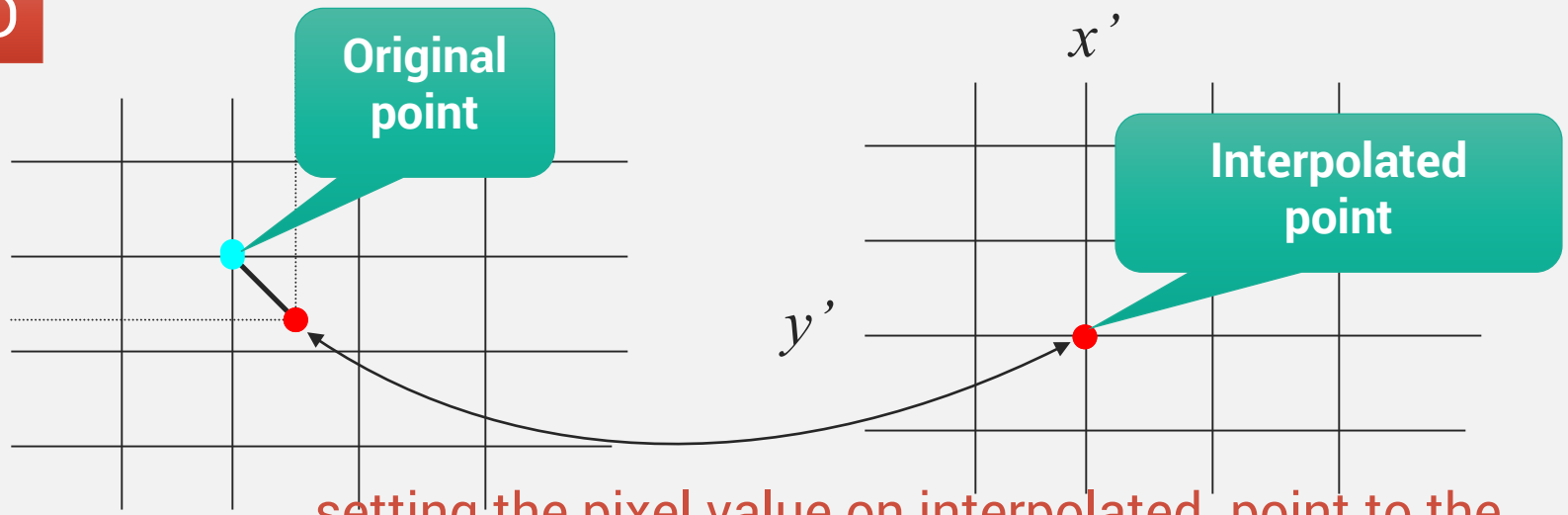
We assign $f(x_i') = f(x_j)$

x_j is the original point closest to x_i'

The original function values

The interpolated values

2-D



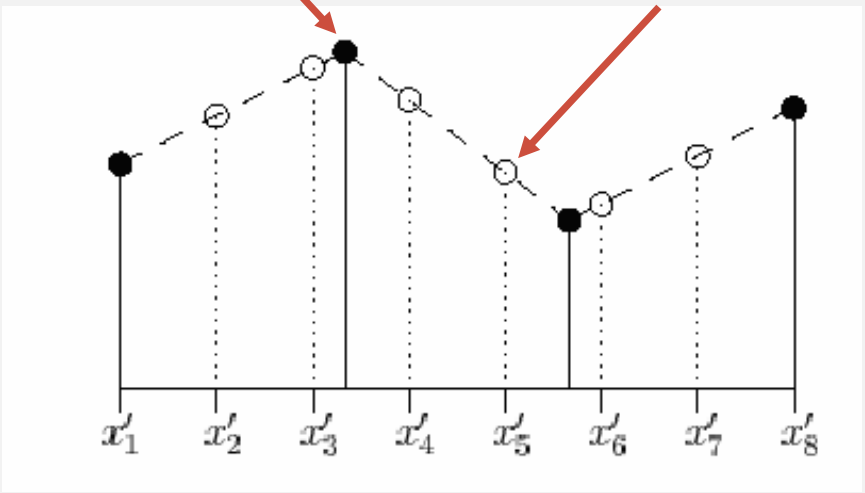
setting the pixel value on interpolated point to the pixel of closet image point

INTERPOLATION: LINEAR (1D)

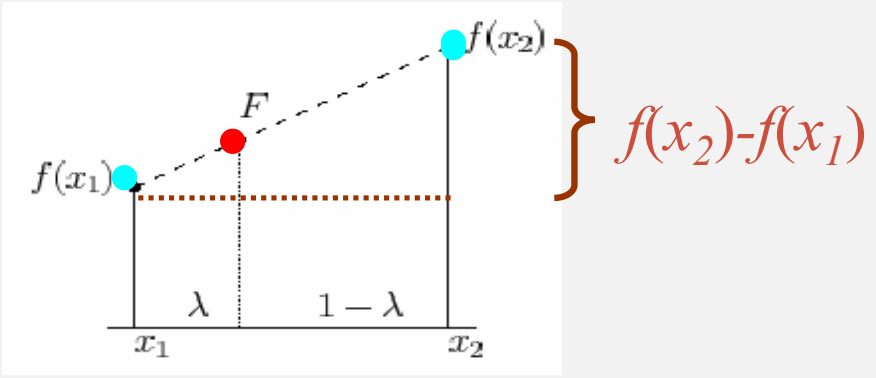
- General idea:

original function values

interpolated values



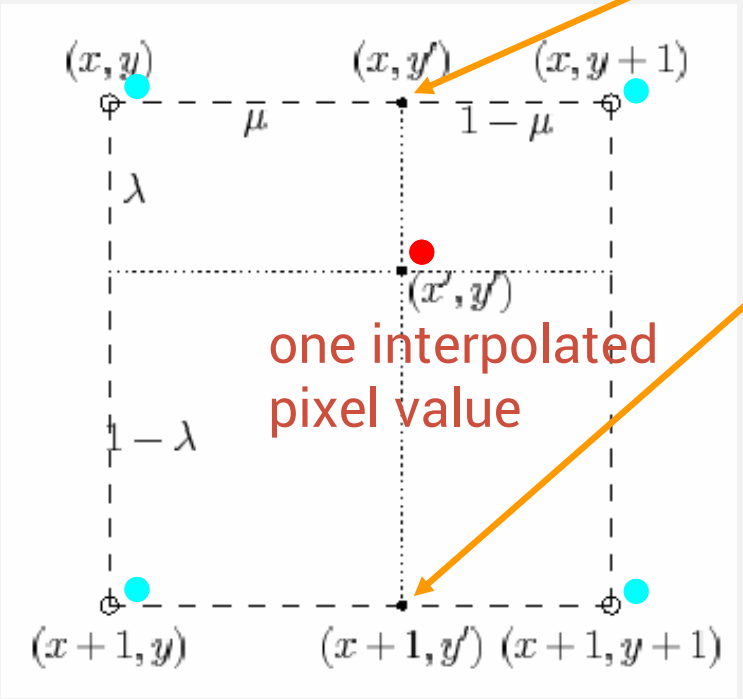
To calculate the interpolated values



$$\frac{F - f(x_1)}{\lambda} = \frac{f(x_2) - f(x_1)}{1}$$

INTERPOLATION: LINEAR (2D)

- How a 4x4 image would be interpolated to produce an 8x8 image?



$$f(x,y') = \mu f(x,y+1) + (1-\mu) f(x,y)$$

$$f(x+1,y') = \mu f(x+1,y+1) + (1-\mu) f(x+1,y)$$

Along the y' column we have

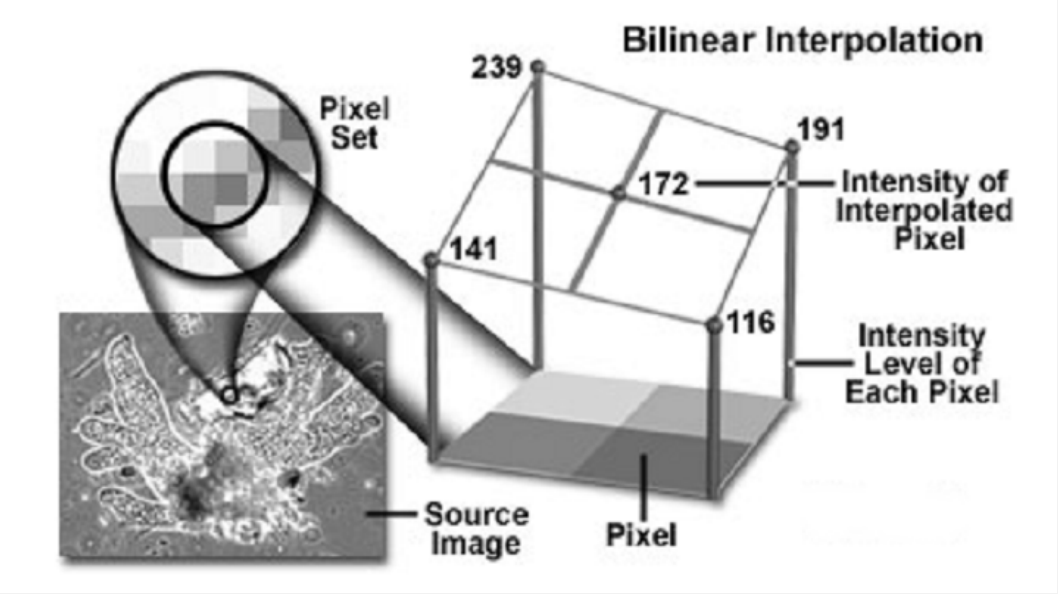
$$f(x',y') = \lambda f(x+1,y') + (1-\lambda) f(x,y')$$

BILINEAR INTERPOLATION

- Substituting with the values just obtained:

$$f(x',y') = \lambda\left(\mu f(x+1,y+1) + (1-\mu) f(x+1,y)\right) + (1-\lambda)\left(\mu f(x,y+1) + (1-\mu) f(x,y)\right)$$

- You can do the expansion as an exercise.
- This is the formulation for bilinear interpolation

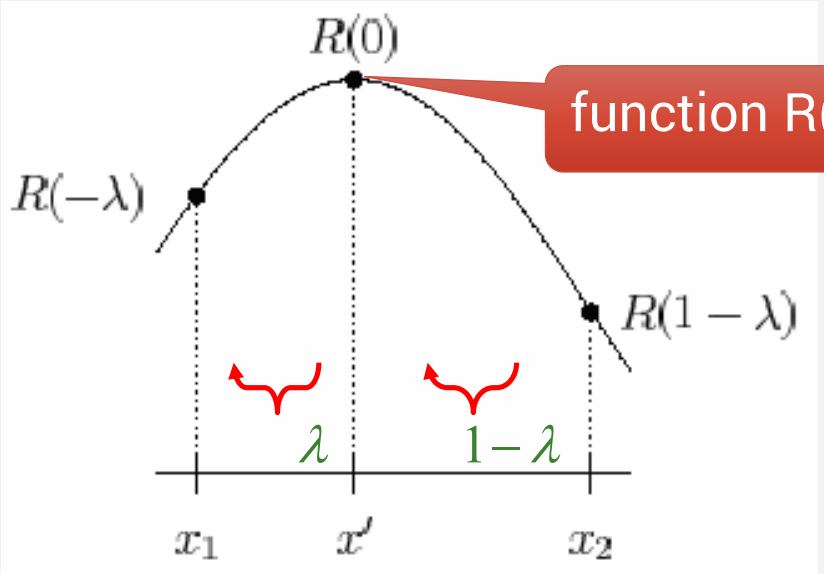


GENERAL INTERPOLATION

- We wish to interpolate a value f(x′) for $x_1 \le x' \le x_2$ and suppose $x' - x_1 = \lambda$
- We define an interpolated value R(u) and set

$$f(x') = R(-\lambda) f(x_1) + R(1-\lambda) f(x_2)$$

$0 \le \lambda \le 1$

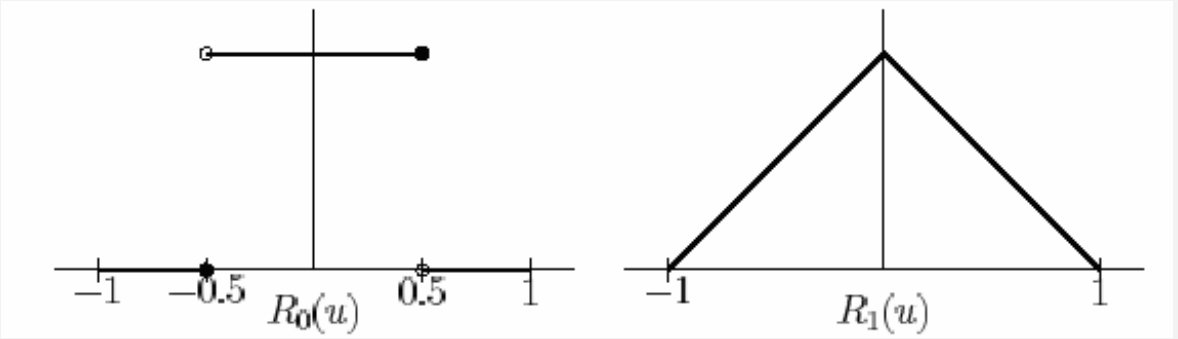


function R(u) is centered at x′

x_1 corresponds with u= - λ,
and x_2 with u= 1- λ

GENERAL INTERLOPLATION: 0TH AND 1ST ORDERS

- Consider 2 functions R0(u) and R1(u)



$$R_0(u) = \begin{cases} 0 & if\ u \le -0.5 \\ 1 & if\ -0.5 < u \le 0.5 \\ 0 & if\ u > 0.5 \end{cases}$$

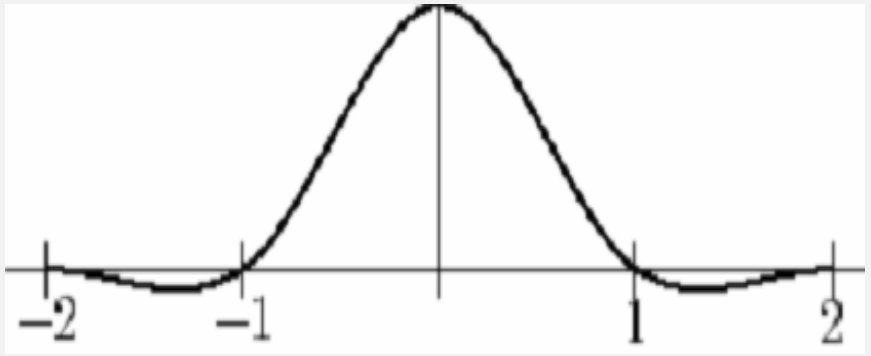
$$R_1(u) = \begin{cases} 1+u & if\ u \le 0 \\ 1-u & if\ u \ge 0 \end{cases}$$

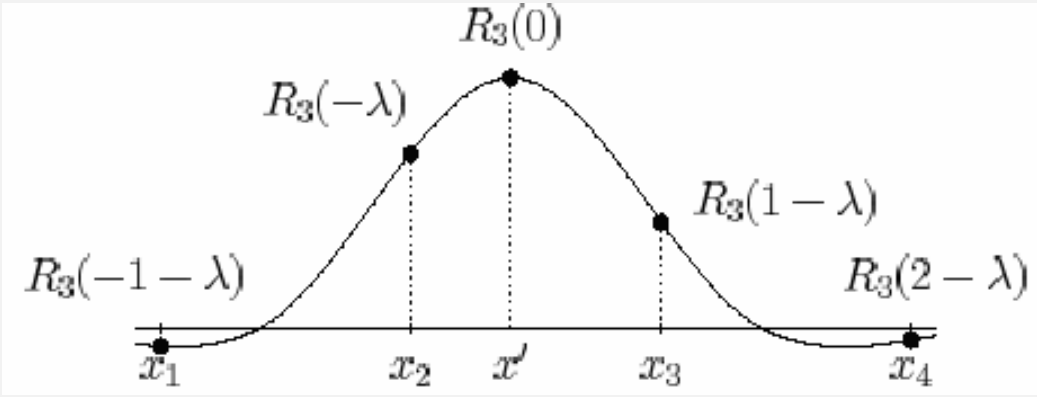
Substitute $R_0(u)$ for R(u) ➡ nearest neighbour interpolation.

Substitute $R_1(u)$ for R(u) ➡ linear interpolation.

GENERAL INTERLOPLATION: 3RD ORDER (CUBIC)

$$R_3(u) = \begin{cases} 1.5|u|^3 - 2.5|u|^2 + 1 & if\ |u| \le 1 \\ -0.5|u|^3 + 2.5|u|^2 - 4|u| + 2 & if\ 1 < |u| \le 2 \end{cases}$$

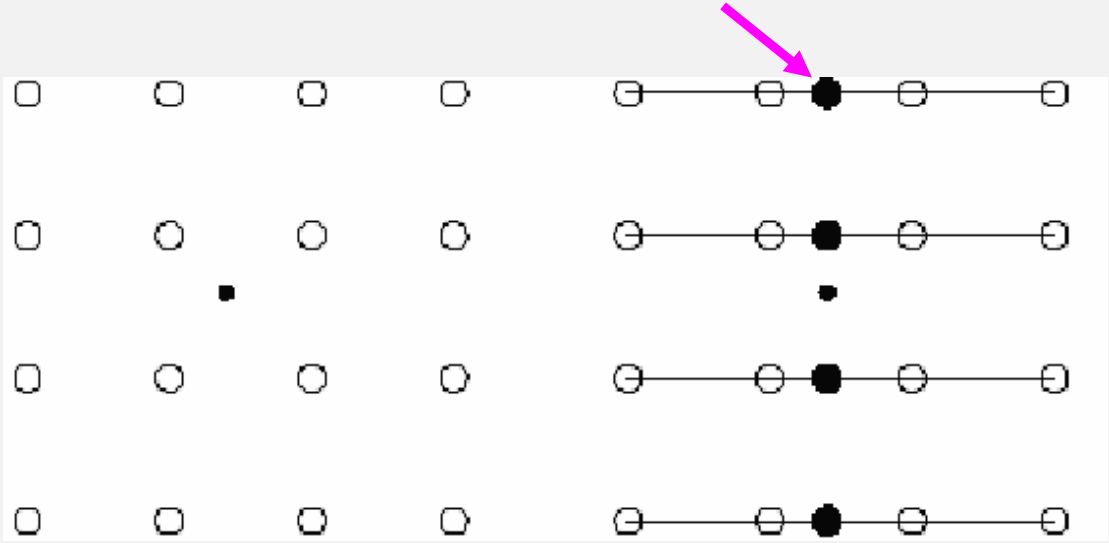




$$f(x') = R_3(-1-\lambda)f(x_1) + R_3(-\lambda)f(x_2) + R_3(1-\lambda)f(x_3) + R_3(2-\lambda)f(x_4)$$

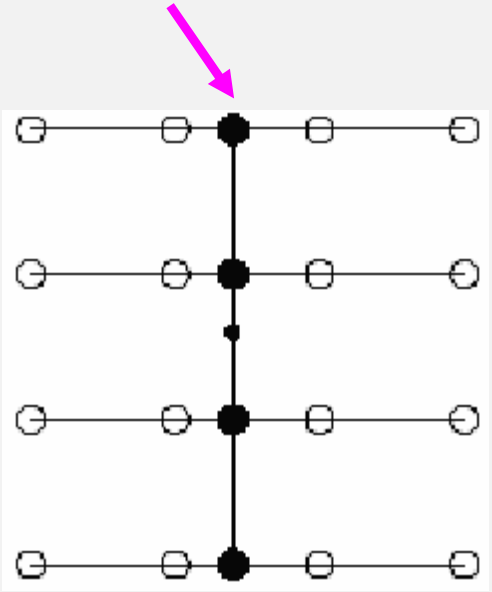
GENERAL INTERPOLATION: BICUBIC (2D)

- Bicubic interpolation fits a series of cubic polynomials to the brightness values contained in the 4 x 4 array of pixels surrounding the calculated address.
  - Step 1: four cubic polynomials F(i), i = 0, 1, 2, 3 are fit to the control points along the rows. The fractional part of the calculated pixel's address in the x-direction is used.



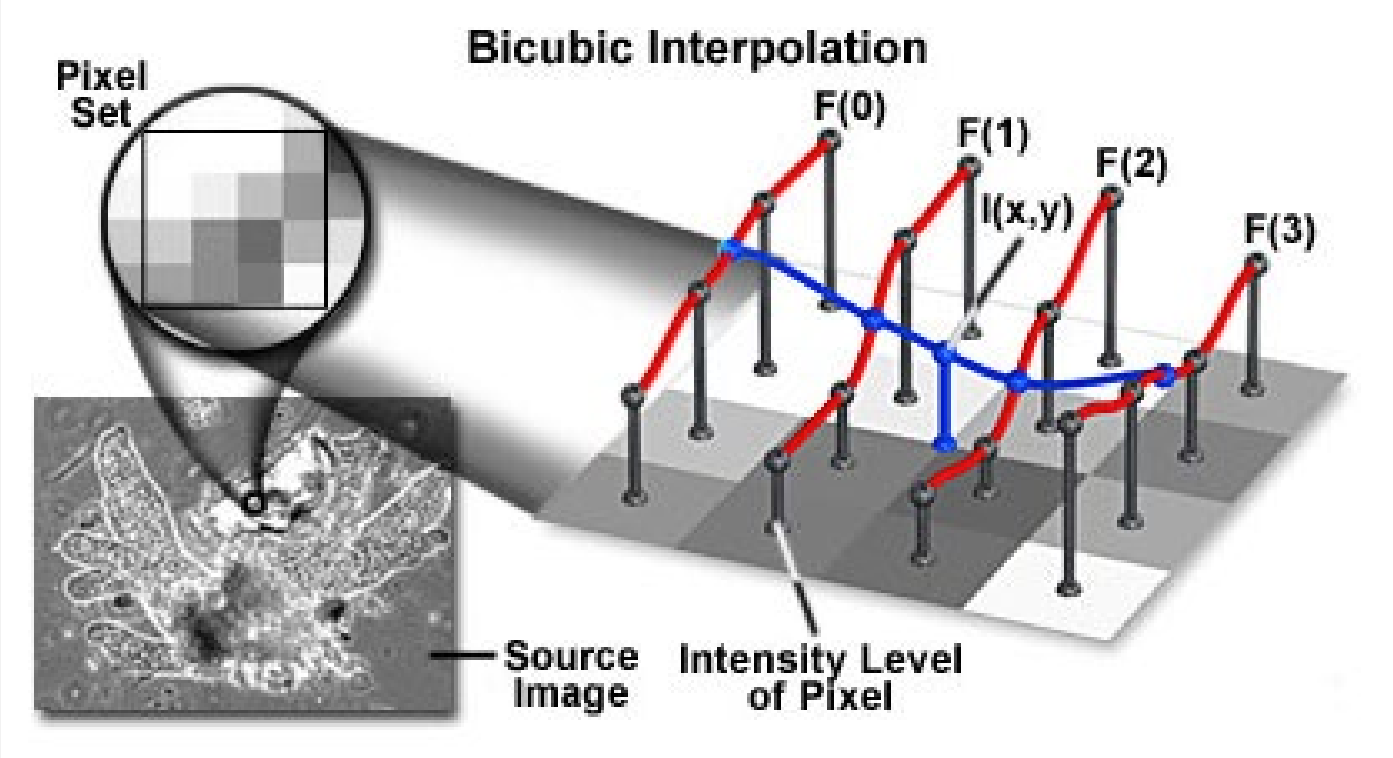
GENERAL INTERPOLATION: BICUBIC

- Step 2: the fractional part of the calculated pixel's address in the y-direction is used to fit another cubic polynomial down the column, based on the interpolated brightness values that lie on the curves F(i), i = 0, ..., 3.



GENERAL INTERPOLATION: BICUBIC

- Substituting the fractional part of the calculated pixel's address in the x-direction into the resulting cubic polynomial then yields the interpolated pixel's brightness value.



GENERAL INTERPOLATION: EXAMPLE

- Original detailed part of flower image (8bit, 75×75)
- Detailed part of super-resolution image (8bit, 300×300) :

NN Interpolation

Bilinear Interpolation

Bicubic Interpolation

```
im = imread('flower.jpg');
im2= imresize(im,[800,800],method);
```

GENERAL INTERPOLATION: SUMMARY

- For NN interpolation, the output pixel is assigned the value of the pixel that the point falls within. No other pixels are considered.
- For bilinear interpolation, the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood.
- For bicubic interpolation, the output pixel value is a weighted average of pixels in the nearest 4-by-4 neighborhood.
- Bilinear method takes longer than nearest neighbor interpolation, and the bicubic method takes longer than bilinear.
- The greater the number of pixels considered, the more accurate the computation is, so there is a trade-off between processing time and quality.

SCALING OPERATION

- To shrink or zoom the size of an image (or part of an image).
- To change the visual appearance of an image;
- To alter the quantity of information
- To use as a low-level pre-processor in multi-stage image processing chain which operates on features of a particular scale.
- Scaling is a special case of *affine transformation*.
- The matlab command is simply “imresize”.

ROTATION OPERATION

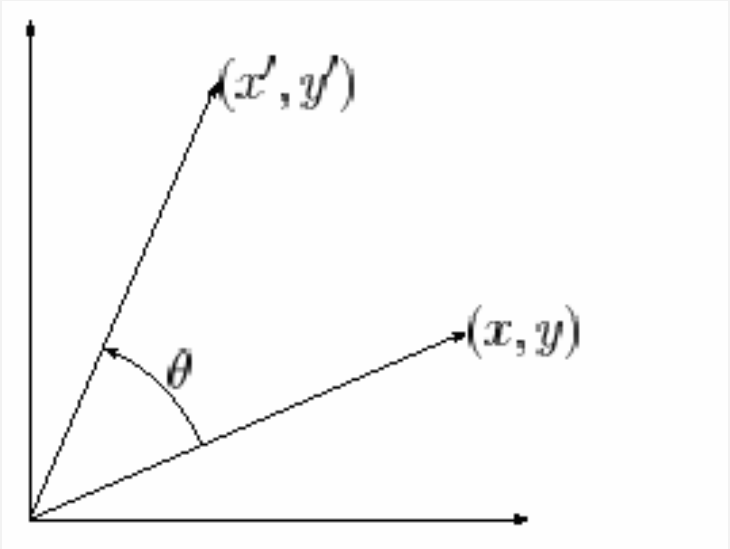
- A geometric transform which maps the position of a picture element in an input image onto a position in an output image by rotating it through an angle about an origin.
- Commonly used to improve the visual appearance of an image.
- Can also be useful as a pre-processor in applications where directional operators are involved.
- Rotation is a special case of affine transformation.

ROTATION OPERATION (CONT)

- Mapping of a point (*x*,*y*) to another (*x'*,*y'*) through a counter-clockwise rotation of θ

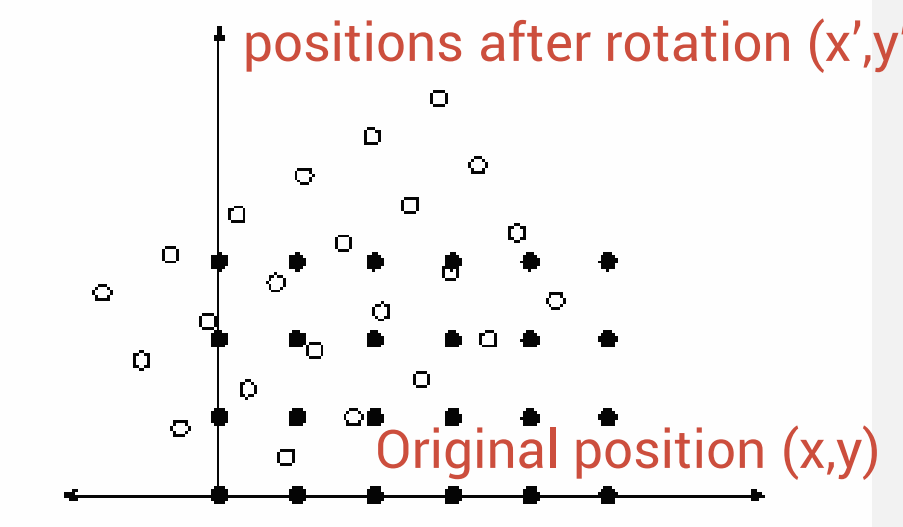
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

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



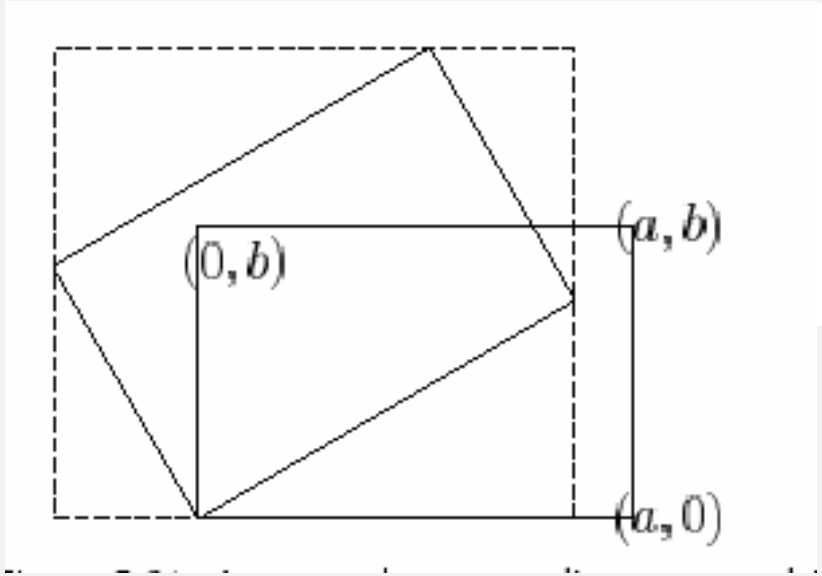
ROTATION OPERATION: PROBLEMS?

- In image space, when rotating a collection of points, what could go wrong?



ROTATION OPERATION: REMEDIES

- Problem 1: part of rotated image might fall out of valid image range.
- Problem 2: how to obtain the intensity values in the rotated image?



A rectangle surrounding a rotated image

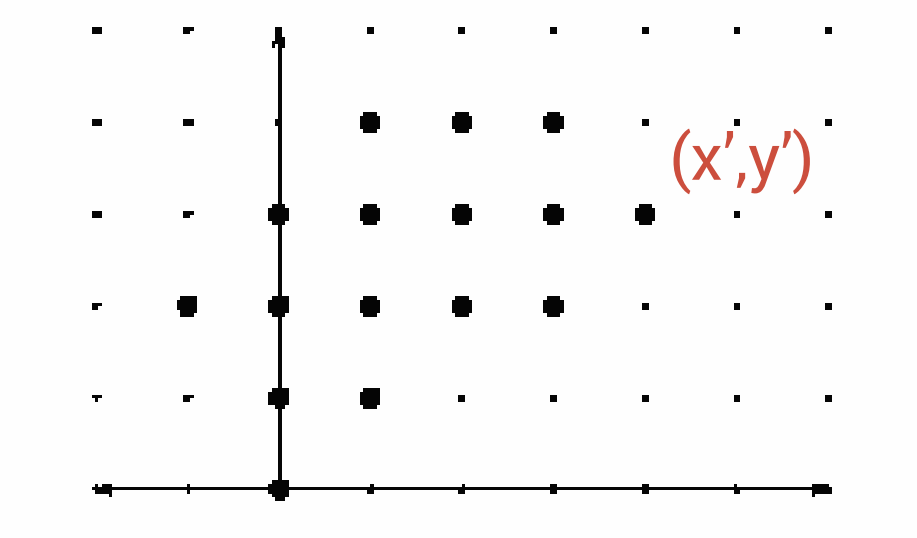
Consider all integer-valued points (x',y') in the dashed rectangle.

A point will be in the image if, when rotated back, it lies within the original image limits.

$$0 \le x'\cos\theta + y'\sin\theta \le a$$

$$0 \le -x'\sin\theta + y'\cos\theta \le b$$

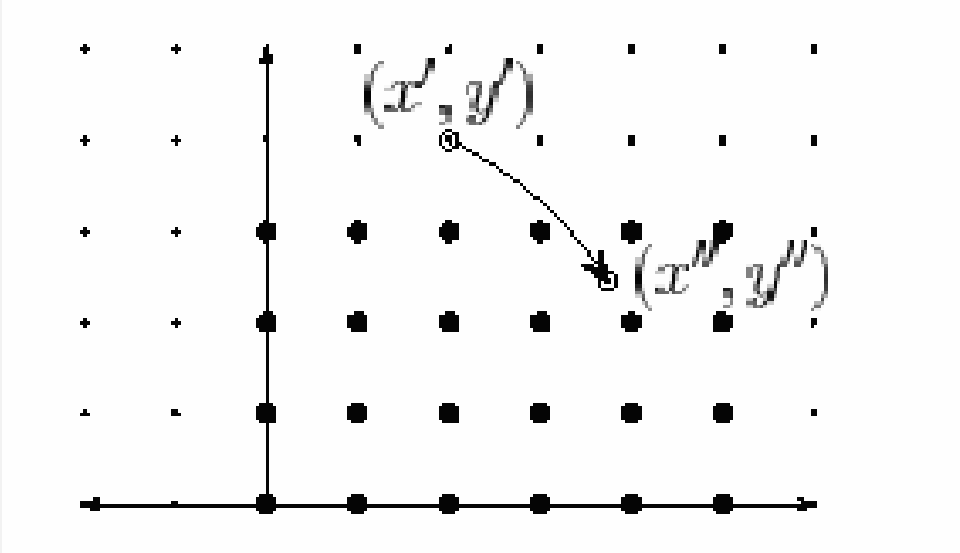
ROTATION OPERATION: REMEDIES (CON'D)



(x′,y′) in rotated image

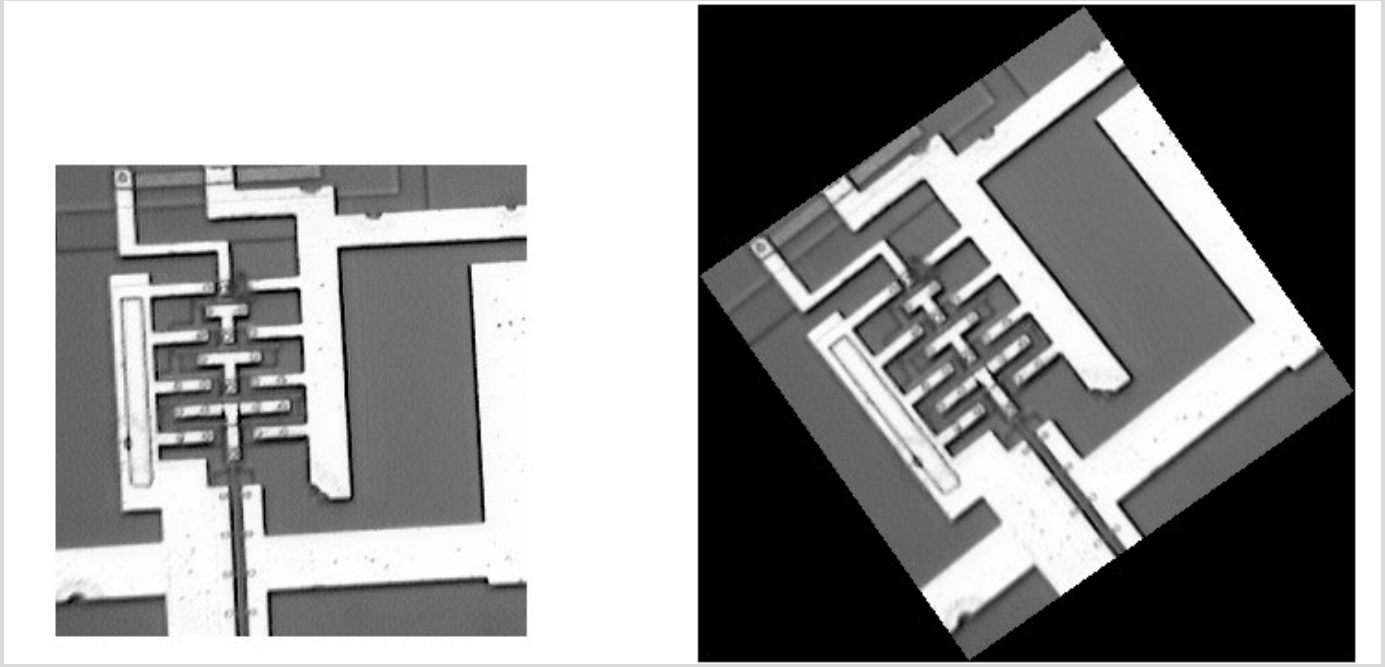
(x″,y″) is the rotated (x′,y′) back into the original image

- The grey value at (x″,y″) can be found by interpolation.
- This value is the grey value for the pixel at (x′,y′) in the rotated image.



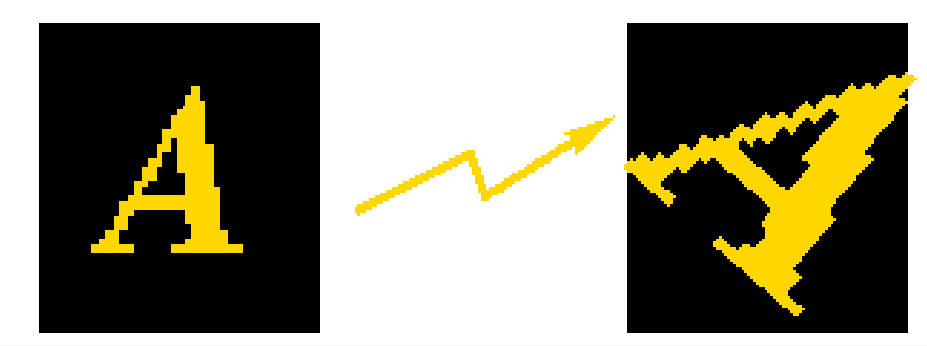
ROTATION OPERATION: EXAMPLE

```
I = imread('ic.tif');
J = imrotate(I,35,'bilinear');
imshow(I)
figure, imshow(J)
```



AFFINE TRANSFORMATION

- An affine transformation maps variables (e.g. pixel intensity values located at position in an input image) into new variables (e.g. in an output image) by applying a linear combination of translation, rotation, scaling operations.
- Significance: In some imaging systems, images are subject to geometric distortions. Applying an affine transformation to a uniformly distorted image can correct for a range of perspective distortions.



AFFINE TRANSFORMATION (CON'D)

- An affine transformation is equivalent to the composed effects of translation, rotation and scaling, and shearing.
- The general affine transformation is commonly expressed as below:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = A \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + B$$

1^{st} order coefficients

0^{th} order coefficients

AFFINE TRANSFORMATION (CON'D)

- By defining only the B matrix, this transformation can carry out pure translation:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

- Pure rotation uses the A matrix and is defined as (for positive angles being clockwise rotations):

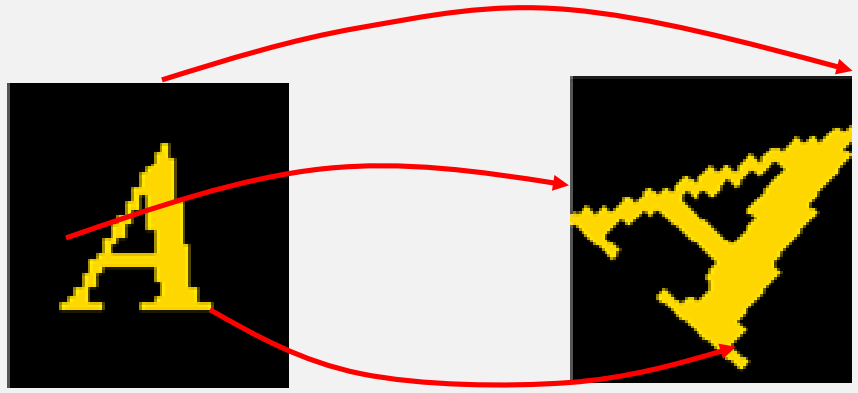
$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

AFFINE TRANSFORMATION (CON'D)

- Pure scaling is defined as

$$A = \begin{bmatrix} a_{11} & 0 \\ 0 & a_{22} \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Since the general affine transformation is defined by 6 constants, it is possible to define this transformation by specifying 3 corresponding point pairs (more in next class).



MATLAB IMTRANSFORM

- The imtransform function accepts two primary arguments:
  - The image to be transformed
  - A spatial transformation structure, called a TFORM, that specifies the type of transformation you want to perform

- Specify the type of transformation in a TFORM structure.
- Two ways to create a TFORM struct:
  - Using the maketform function
  - Using the cp2tform function

USING MAKETFORM

- When using the maketform function, you can specify the type of transformation, e.g
  - 'affine’
  - 'projective’
  - 'composite‘, et al
  - ‘custom' and 'composite' capabilities of maketform allow a virtually limitless variety of spatial transformations to be used
- Once you define the transformation in a TFORM struct, you canperform the transformation by calling imtransform.

EXAMPLE

```
I = imread('cman.tif');
tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
J = imtransform(I,tform);
imshow(I), figure, imshow(J)
```

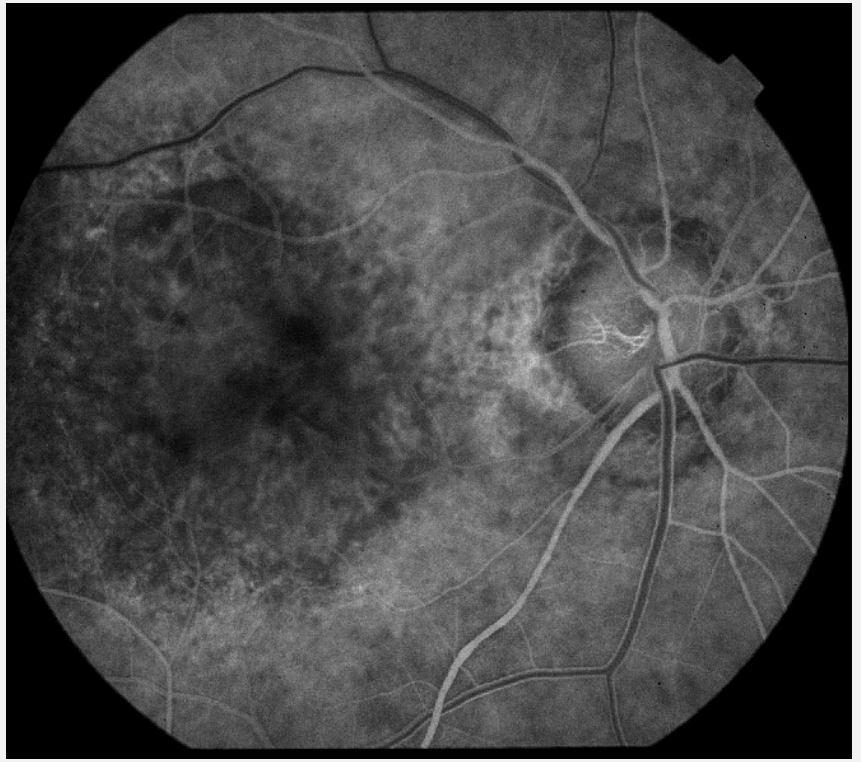
original

transform

REAL APPLICATION

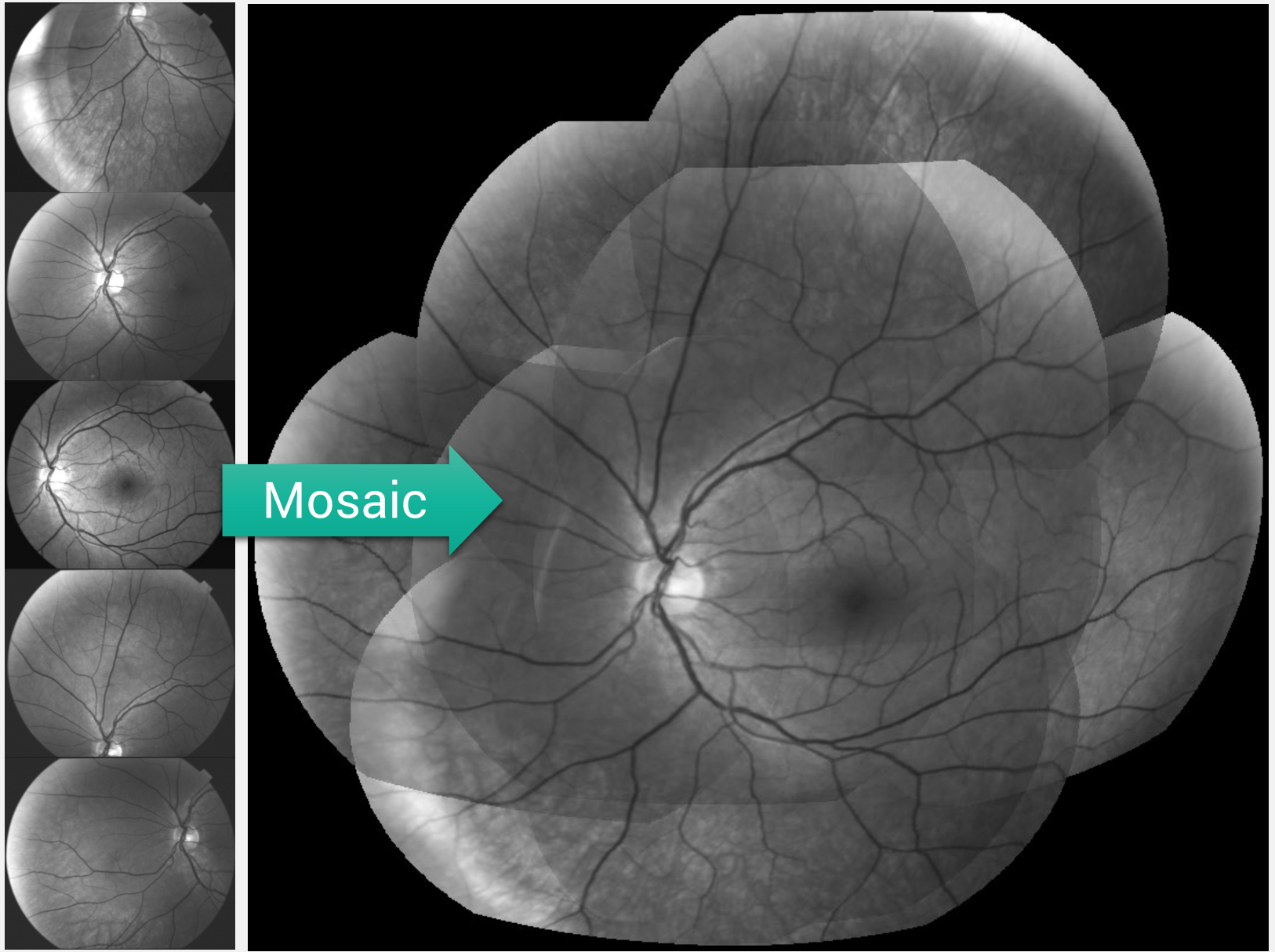
- Registration of retinal images of different modalities
- Importance?

Color slide

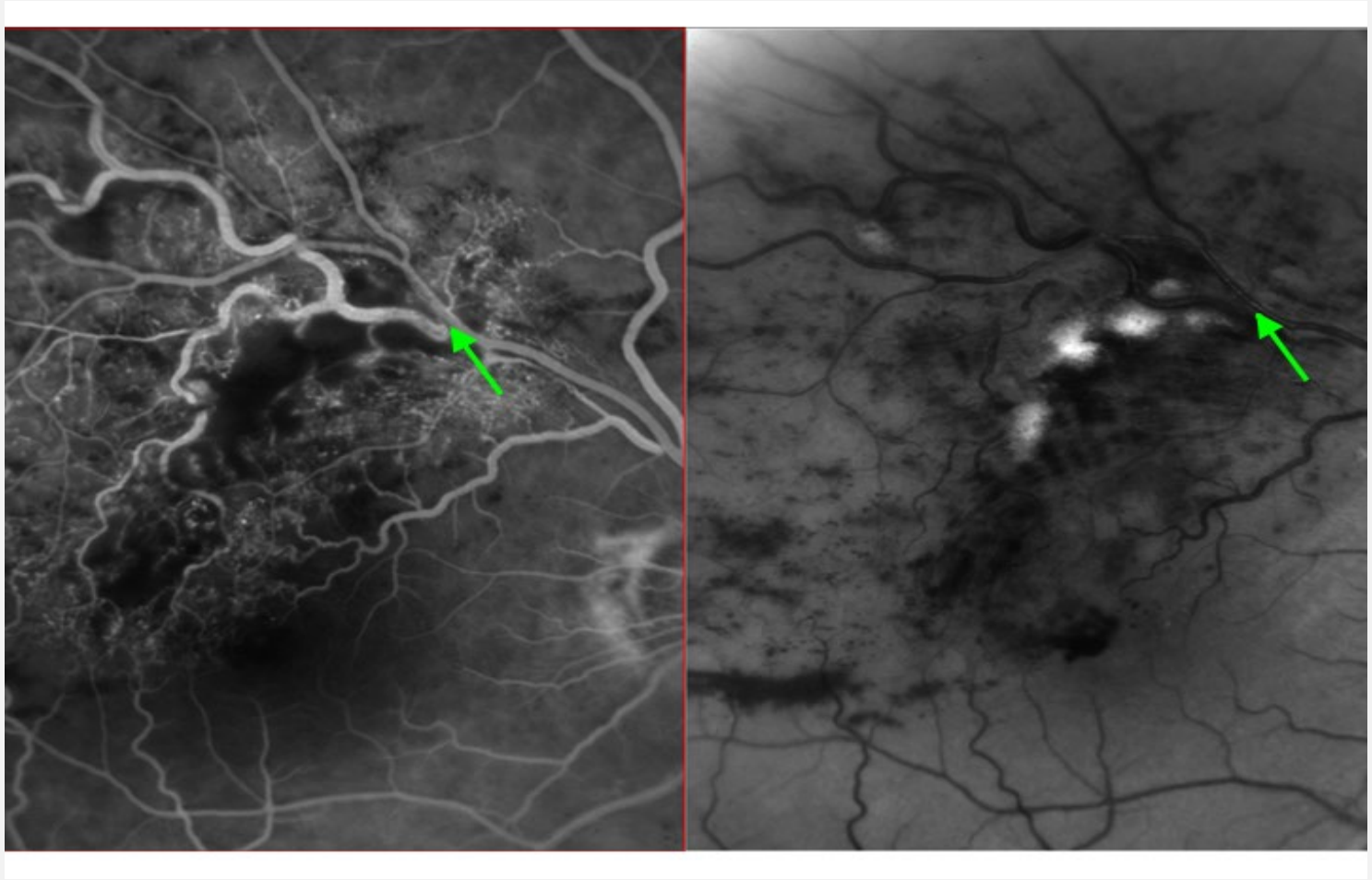


Fluorocein Angiogram

MOSAICING

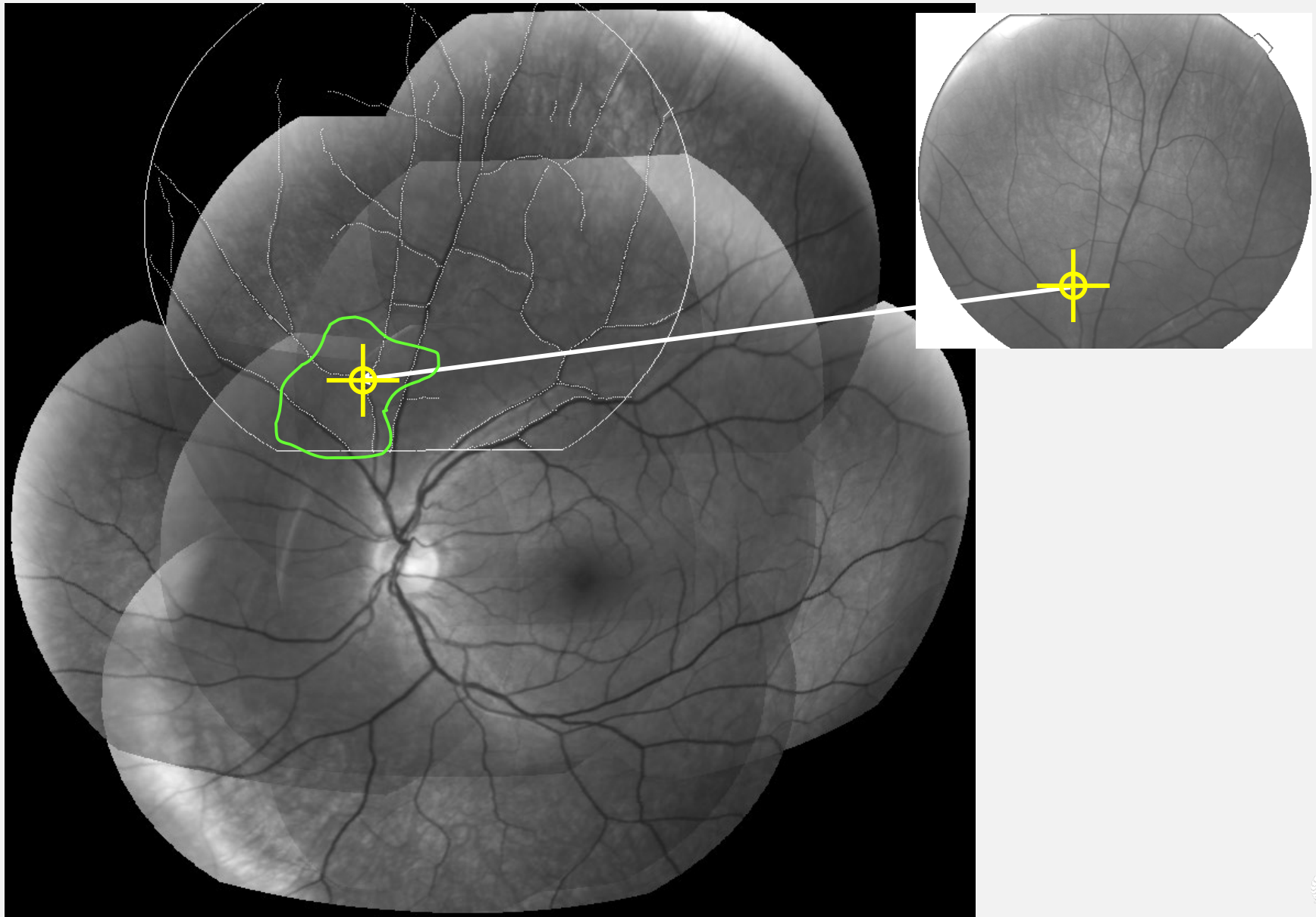


FUSION OF MEDICAL INFORMATION



CHANGE DETECTION

COMPUTER-ASSISTED SURGERY



WHAT IS NEEDED?

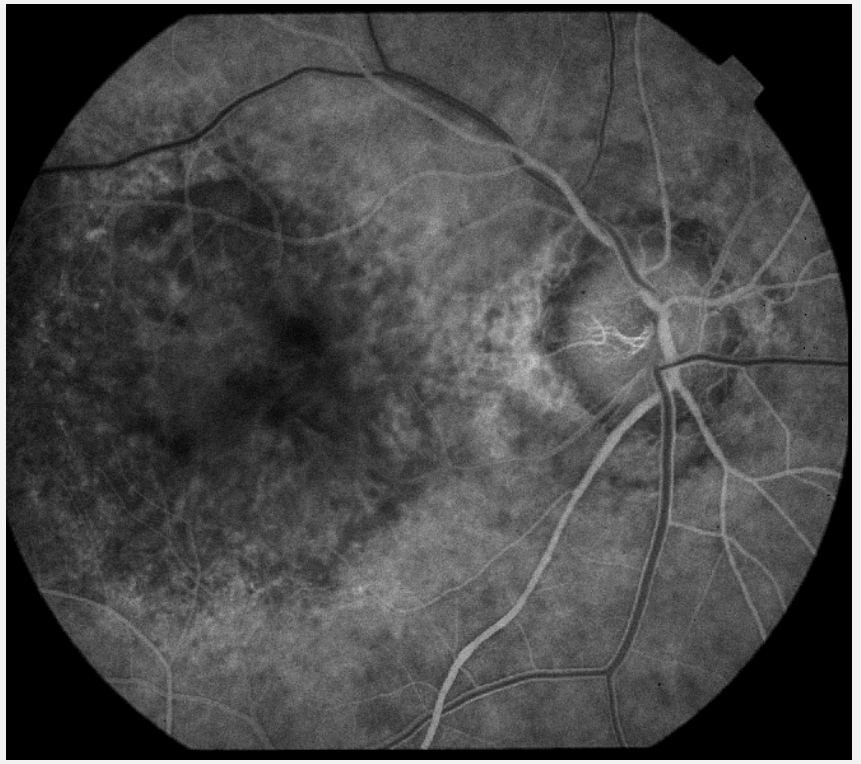
- A known transformation, which is less likely

OR

- Computing the transformation from features.
  - Feature extraction
  - Features in correspondence
  - Transformation models
  - Objective function

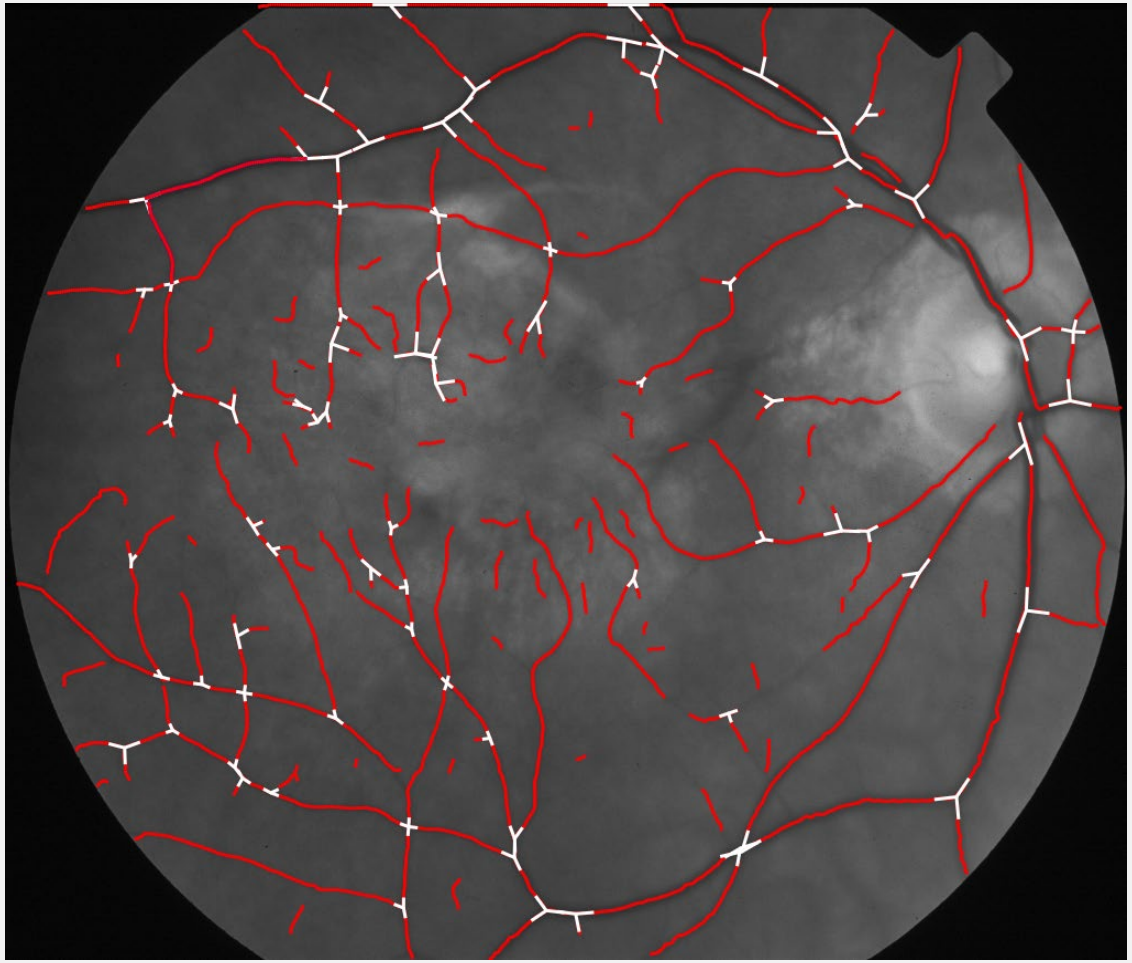
WHAT MIGHT BE A GOOD FEATURE?

Color slide

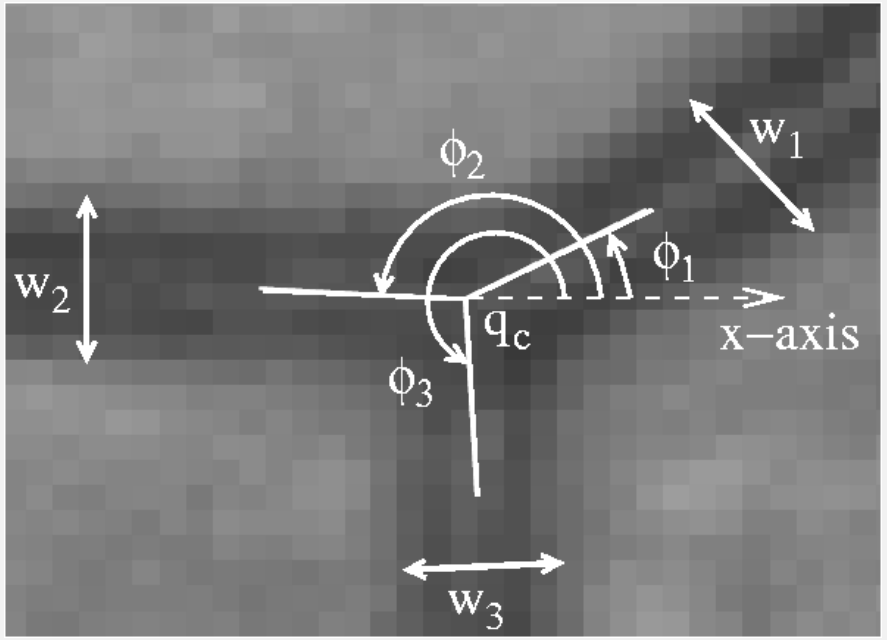


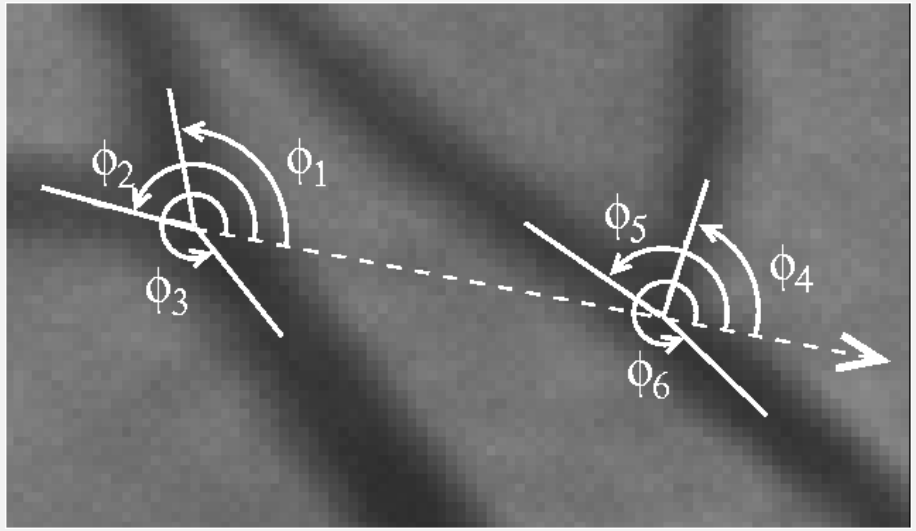
Fluorocein Angiogram

FEATURE EXTRACTION

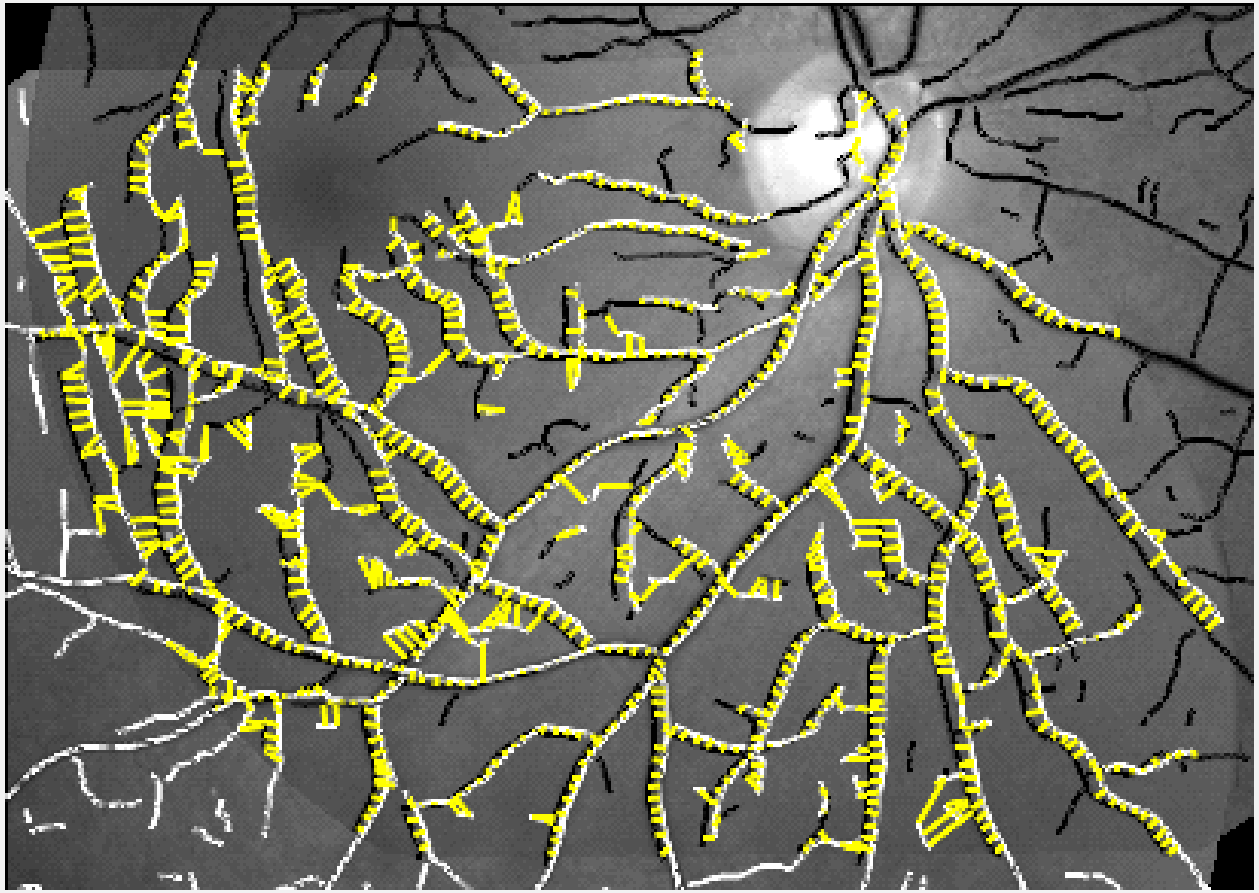


FEATURES IN CORRESPONDENCE (CROSSOVER & BRANCHING POINTS)

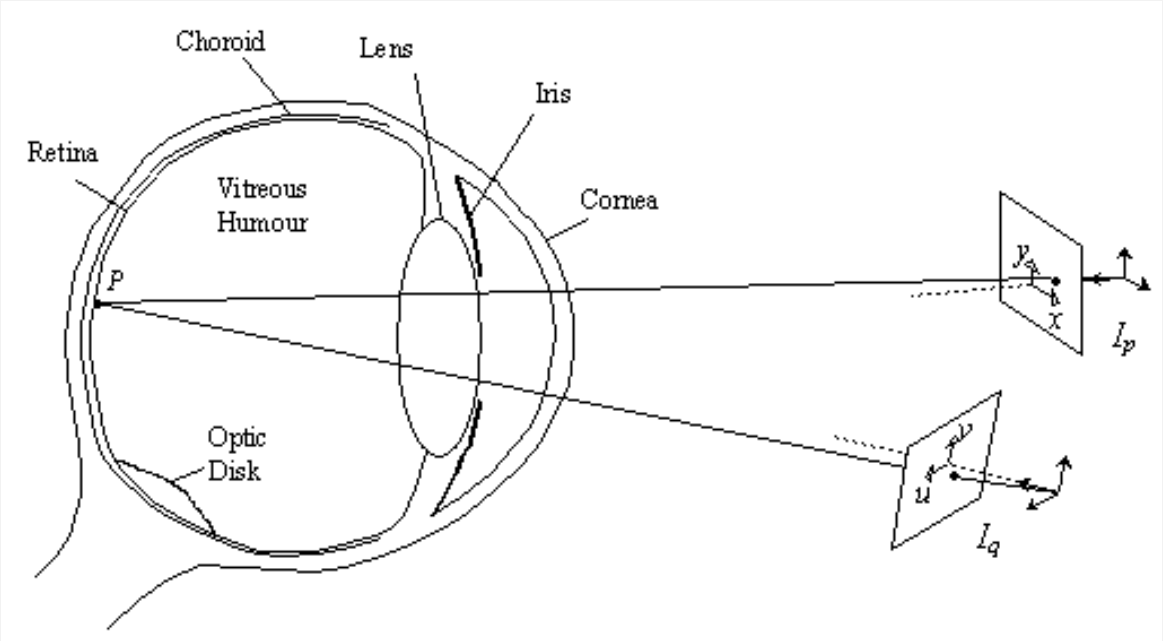


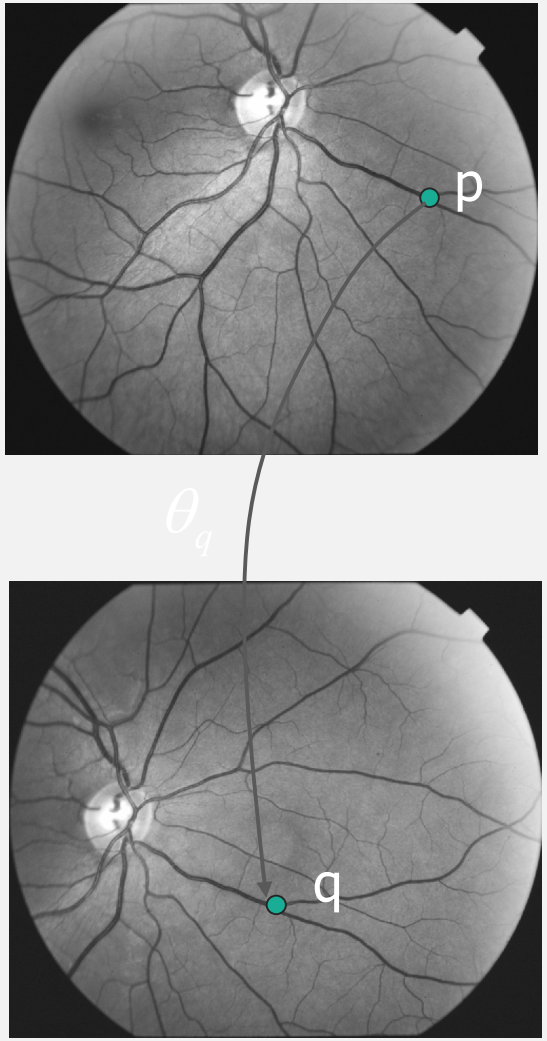


FEATURES IN CORRESPONDENCE (VESSEL CENTERLINE POINTS)



TRANSFORMATION MODELS





12-parameter quadratic transformation

$$\mathbf{q} = \mathrm{M}(\mathbf{p};\theta_q) = \theta_q \cdot \mathbf{X}(\mathbf{p})$$

$$\theta_q = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 & a_{10} & a_{11} & a_{12} \end{bmatrix}$$

$$\mathbf{X}(\mathbf{p}) = \begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix}^T \qquad \mathbf{p} = (x, y)^T$$

TRANSFORMATION MODEL HIERARCHY

$$q = M(p;\theta_s) = \begin{pmatrix} a & -b & t_x \\ b & a & t_y \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Similarity

$$q = M(p;\theta_a) = \begin{pmatrix} a & b & t_x \\ c & d & t_y \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Affine

$$q = M(p;\theta_r) = \begin{pmatrix} c & a & -b & t_x \\ d & b & a & t_y \end{pmatrix} \begin{pmatrix} x^2 + y^2 \\ x \\ y \\ 1 \end{pmatrix}$$

Reduced Quadratic

$$q = M(p;\theta_q)$$

Quadratic

OBJECTIVE FUNCTION

$$E(\theta;C) = \sum_{(p_i,q_i)\in C} \rho(d(M(p_i;\theta), q_i)/\sigma)$$

Transformation parameters

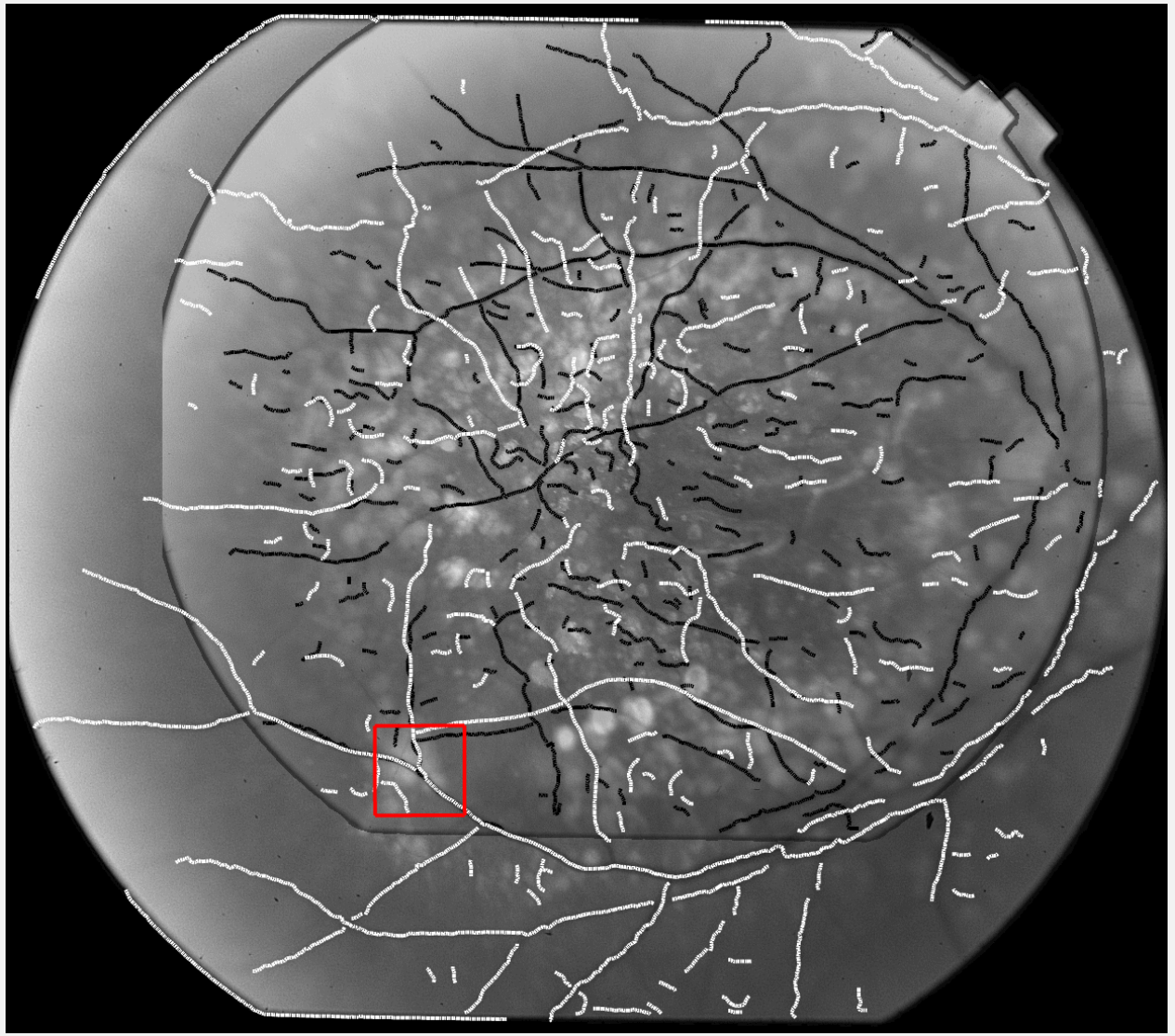
Set of correspondences $\neq \{\}$

Mapping of features from I_p to I_q based on θ

Feature point in image I_q

Feature point in image I_p

RESULT



SUMMARY

- Interpolation of intensity values on non-grid points:
  - Nearest Neibhgor (NN)
  - Bilinear
  - Bicubic
- Image transformation
  - Computation of intensity values of the transformed image
  - Discussed some instances of affine transformation
    - Translation
    - Rotation
    - Scaling

TERIMA KASIH

Adhi Harmoko Saputro

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia