


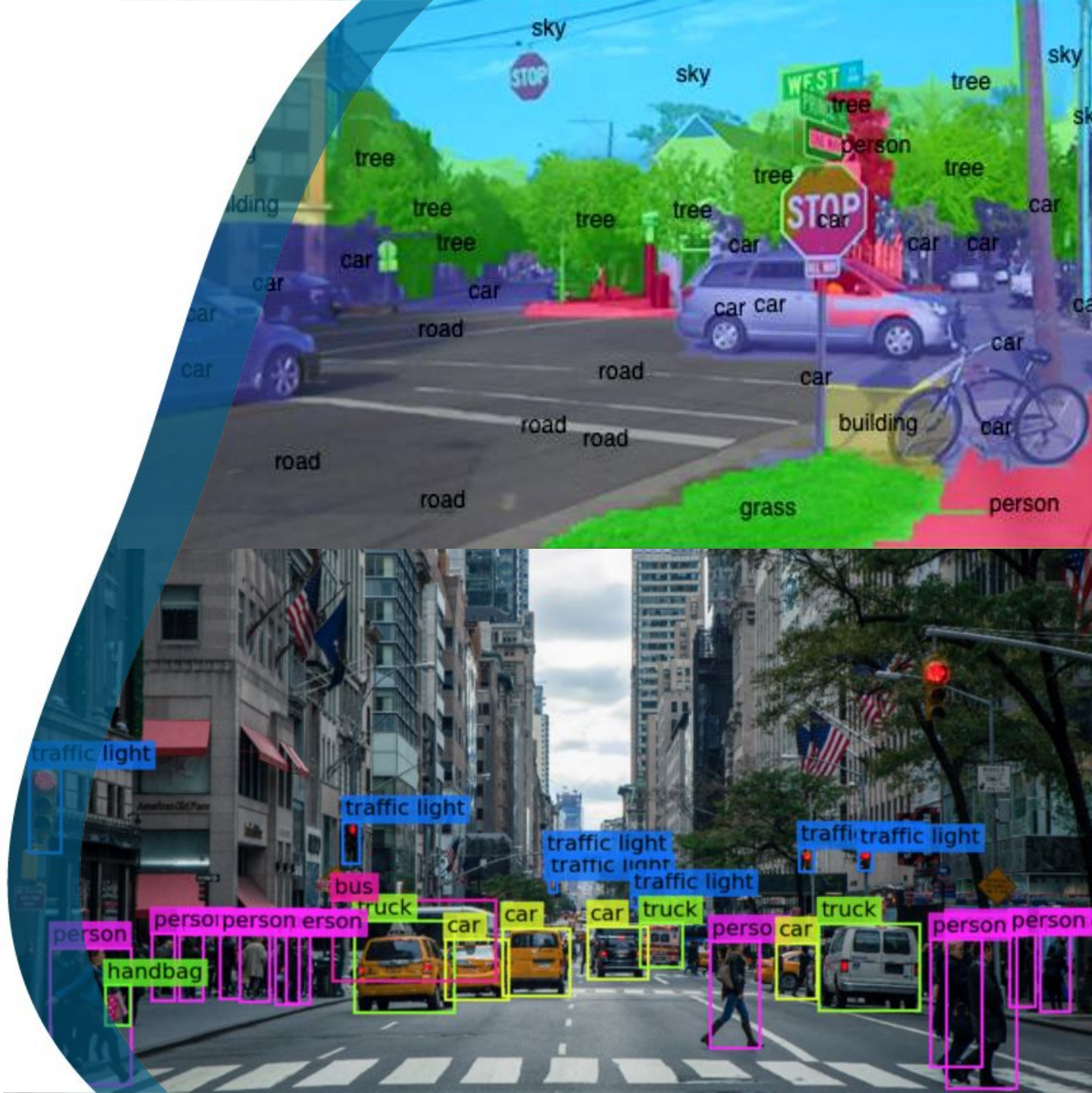
CCE5205: Computer Vision

Fundamentals of Computer Vision

 Reuben Farrugia

 reuben.farrugia@um.edu.mt

 <https://www.um.edu.mt/staff/reuben.farrugia>



Today's Agenda

During this lecture we will do the following

- Convolution of Images
- Image Gradients (Sobel and Laplacian)
- Image Pyramids (Gaussian, Laplacian)
- Linear Algebra
- Image Registration

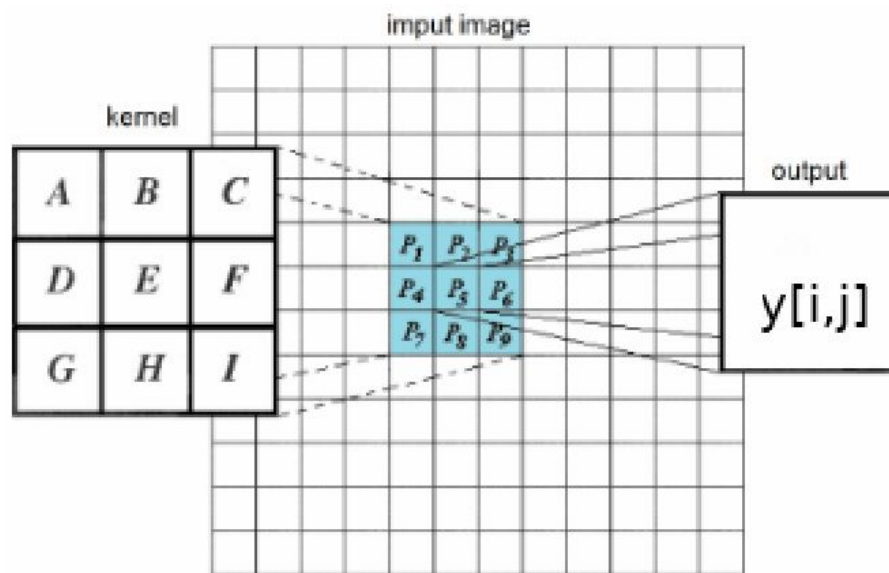


Convolution of Images

The convolution of 2D signals is computed using

$$y[m, n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x[i, j] h[m - i, n - j] = x[n] * h[n]$$

where $x[i, j]$ is the image and $h[i, j]$ is the filter (or kernel).



Example of kernels

1	1	1
1	1	1
1	1	1

3x3 Mean kernel

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

5x5 Gaussian kernel

Convolution of Images

1	2	3
4	5	6
7	8	9

Input

$n \backslash m$	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

Kernel

Note that the kernel is rotated around both m and n direction

1	2	1	
0	0	1	2
-1	-2	-1	5
	7	8	9

Step 1

-13

1	2	1
0 1	0 2	0 3
-1 4	-2 5	-1 6
7	8	9

Step 2

-20

	1	2	1
1	0	0	0
4	-1	-2	-1
7			

Step 3

-17

Convolution of Images

1	2	3
4	5	6
7	8	9

Input

m	-1	0	1
n	-1	-2	-1
0	0	0	0
1	1	2	1

Kernel

Note that the kernel is rotated around both m and n direction

1	2	1		3
0	0	4	5	6
-1	-2	7	8	9

Step 1
-18

1	2	1	3
0	4	5	6
-1	7	8	9

Step 2
-24

1	2	3	1
4	5	6	0
7	8	9	-1

Step 3
-18

Convolution of Images

1	2	3
4	5	6
7	8	9

Input

	m	-1	0	1
n	-1	-1	-2	-1
0	0	0	0	0
1	1	2	1	

Kernel

-13	-20	-17
-18	-24	-18
13	20	17

Output

Note that the kernel is rotated around both m and n direction

		1	2	3
1	2	4	5	6
0	0	7	8	9
-1	-2	-1		

Step 1

30

	1	2	3
1	2	5	6
0	0	8	9
-1	-2	-1	

Step 2

20

1	2	3
4	5	6
7	8	9
	-1	-2

Step 3

17

Convolution of Images

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

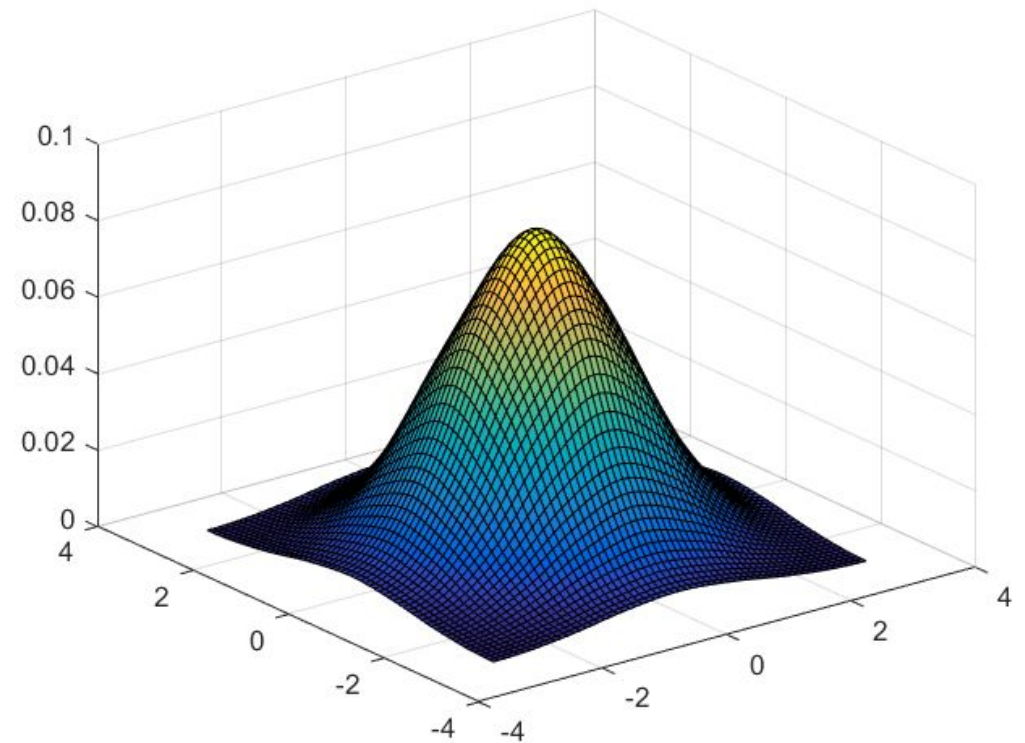
4		

Convolution of Images: Gaussian Kernel

The kernel of a **Gaussian filter** is given by

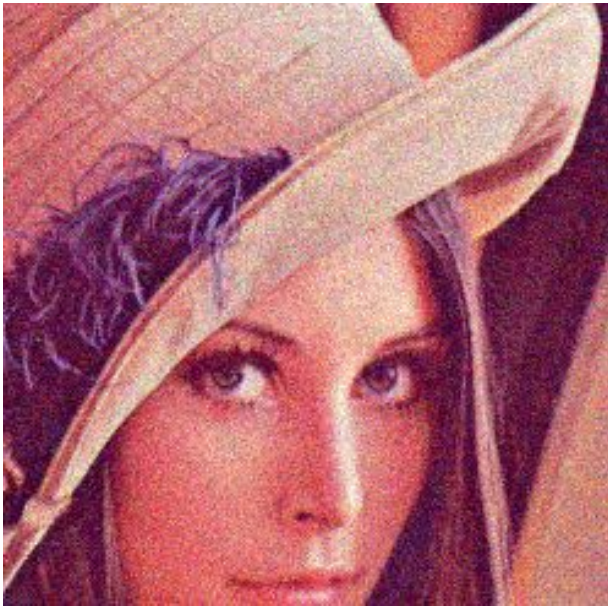
$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

0.0008	0.0030	0.0065	0.0084	0.0065	0.0030	0.0008
0.0030	0.0108	0.0232	0.0299	0.0232	0.0108	0.0030
0.0065	0.0232	0.0498	0.0643	0.0498	0.0232	0.0065
0.0084	0.0299	0.0643	0.0830	0.0643	0.0299	0.0084
0.0065	0.0232	0.0498	0.0643	0.0498	0.0232	0.0065
0.0030	0.0108	0.0232	0.0299	0.0232	0.0108	0.0030
0.0008	0.0030	0.0065	0.0084	0.0065	0.0030	0.0008

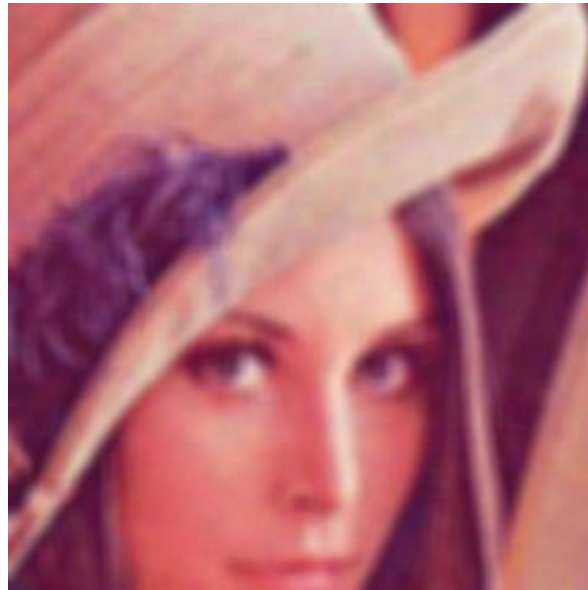


Convolution of Images: Gaussian Kernel

Comparing the Mean and Gaussian filters



Noisy image



Mean filter



Gaussian filter ($\sigma=2$)

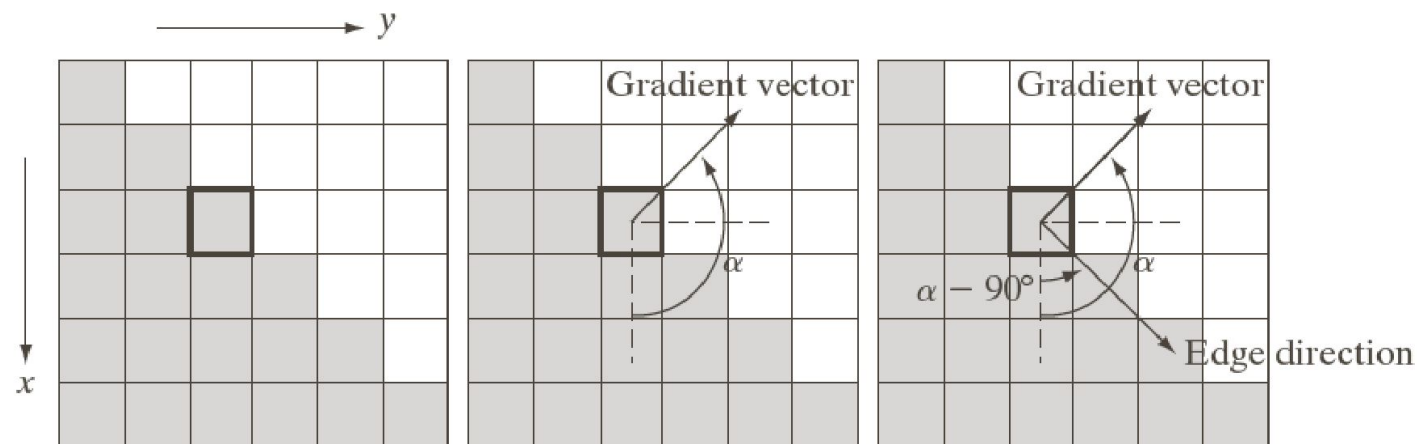
Convolution of Images: Sobel Kernel

Computing the Image Gradient using the Sobel Kernel (1st Order Derivative)

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$|\nabla f(x, y)| = \sqrt{g_x(x, y)^2 + g_y(x, y)^2}$$

$$\alpha(x, y) = \text{atan} \left(\frac{g_y(x, y)}{g_x(x, y)} \right)$$



Convolution of Images: Sobel Kernel

The Sobel kernel has two kernels: one for the vertical gradients and another for the horizontal gradients. These gradients can be computed using

$$G_x = k_x * I$$

$$G_y = k_y * I$$

-1	-2	-1
0	0	0
1	2	1

k_x

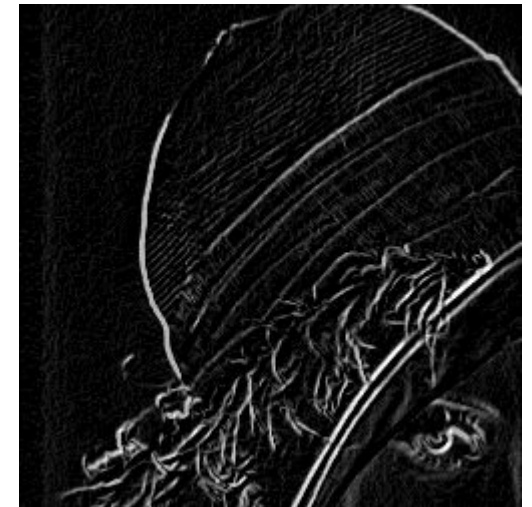
-1	0	1
-2	0	2
-1	0	1

k_y

$$|\nabla f(x, y)| = \sqrt{g_x(x, y)^2 + g_y(x, y)^2}$$



Original Image



Sobel

Convolution of Images: Laplacian Kernel

The 2nd order derivative (also known as Laplacian) is derived using

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

In digital form this is defined as

$$\frac{\partial^2 f}{\partial x^2} = f(x-1, y) + f(x+1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y-1) + f(x, y+1) - 2f(x, y)$$

$$\nabla^2 f = f(x-1, y) + f(x+1, y) + f(x, y-1) + f(x, y+1) - 4f(x, y)$$

Convolution of Images: Laplacian Kernel

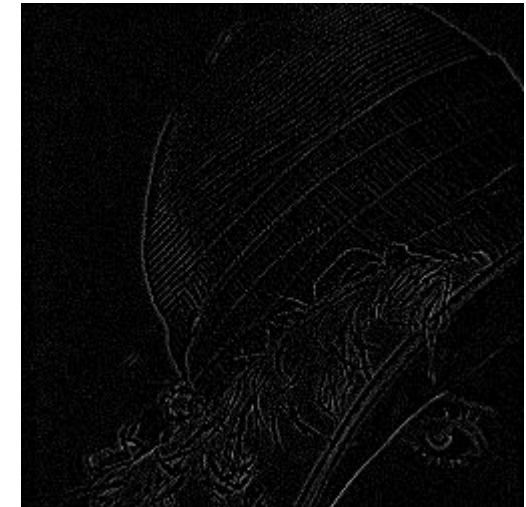
$$\nabla^2 f = f(x - 1, y) + f(x + 1, y) + f(x, y - 1) + f(x, y + 1) - 4f(x, y)$$

0	1	0
1	-4	1
0	1	0

Laplacian Kernel



Original Image



Laplacian

Image Pyramids

- Each level of the **Gaussian Pyramid** is obtained by blurring and downsampling (Reduce) the image from the previous level by a factor of 2. The Gaussian Pyramid is very useful for coarse-to-fine optimizations.
- The **Laplacian Pyramid** is more useful to detect difference in texture across different scales. This is obtained by expanding (upsampling) the lower level (L-1) image by a factor of 2 and subtracting it from the image within the Gaussian Pyramid at level L. This is useful in data compression.

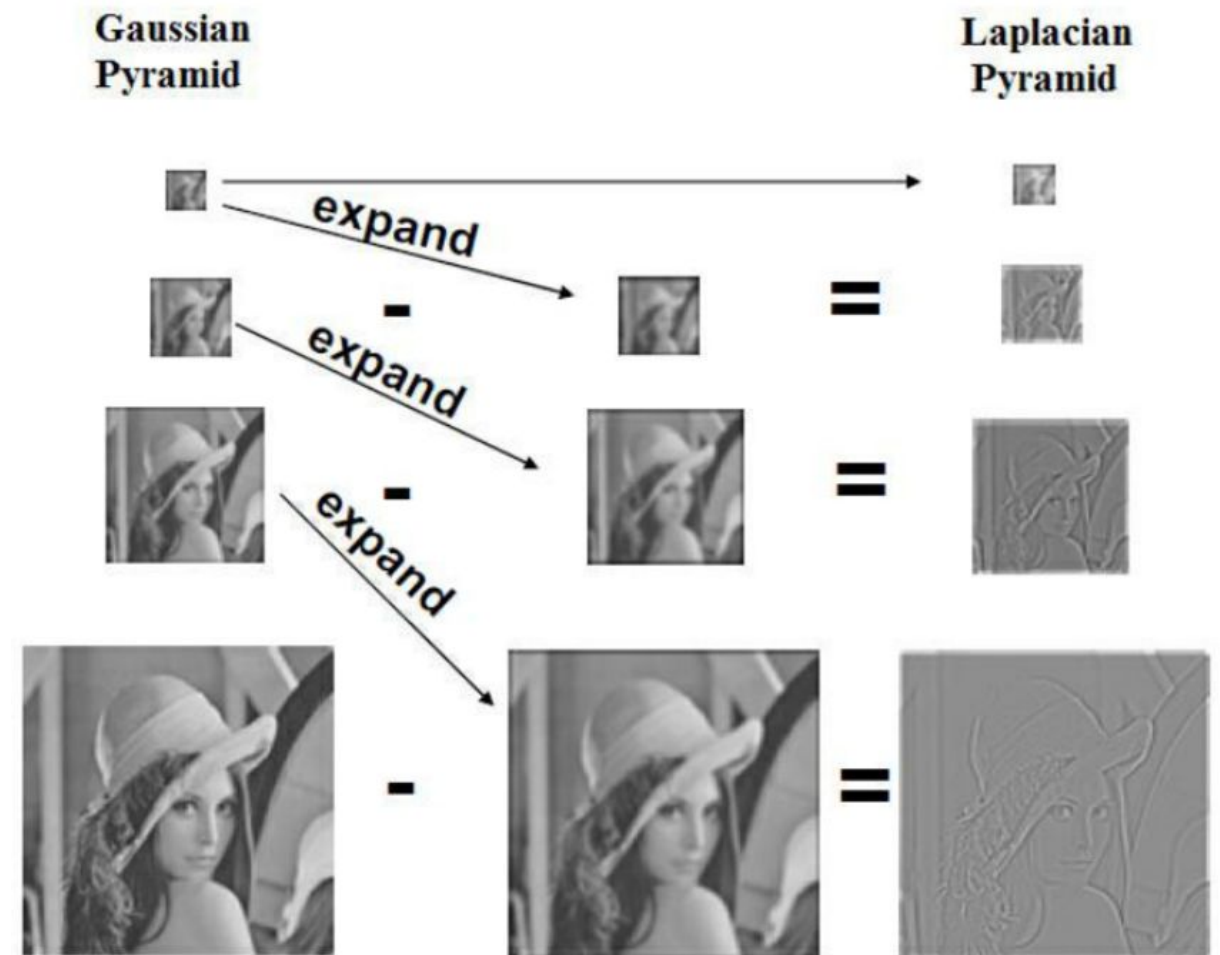
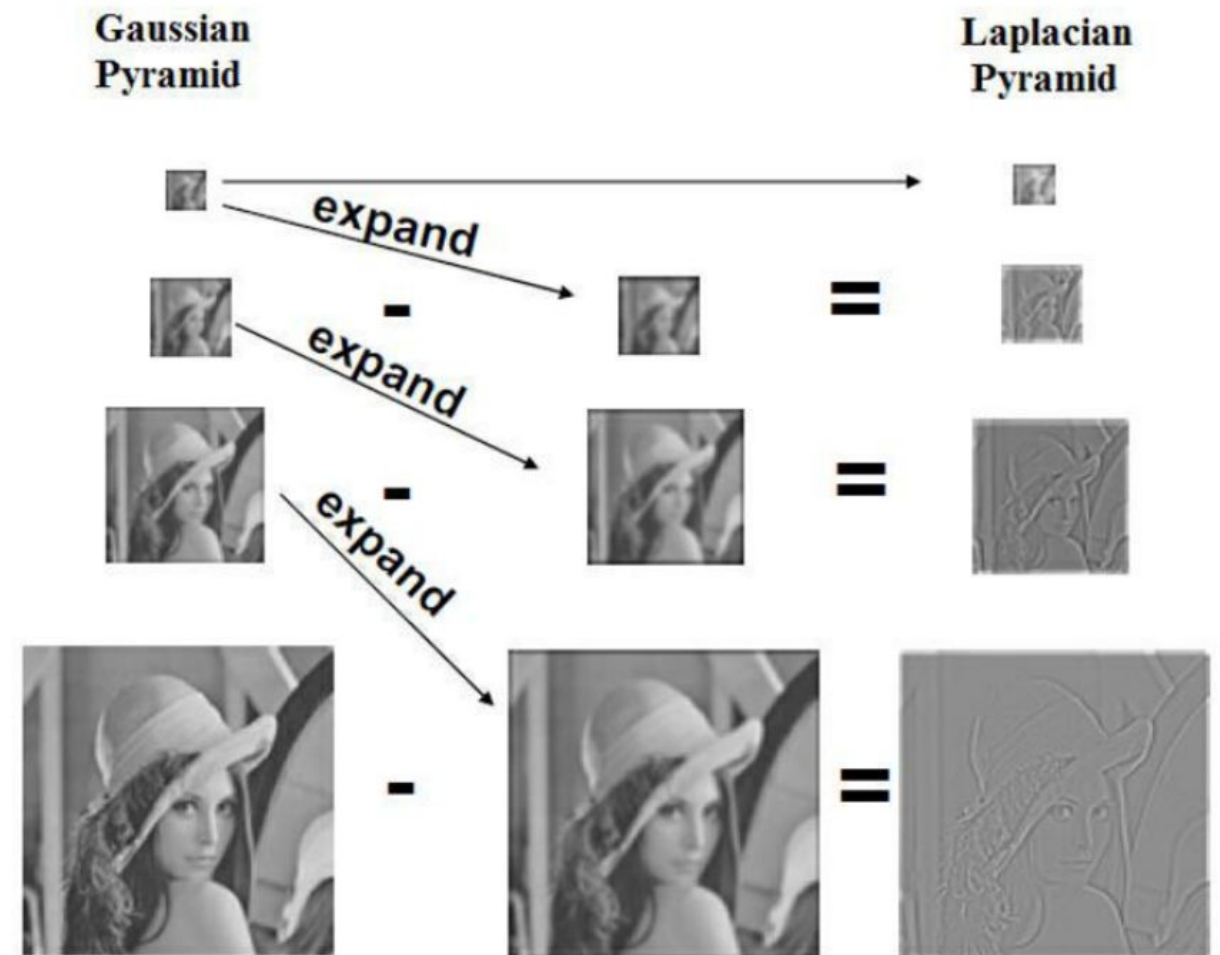


Image Pyramids

- Each level of the **Gaussian Pyramid** is obtained by blurring and downsampling (Reduce) the image from the previous level by a factor of 2. The Gaussian Pyramid is very useful for coarse-to-fine optimizations.
- The **Laplacian Pyramid** is more useful to detect difference in texture across different scales. This is obtained by expanding (upsampling) the lower level (L-1) image by a factor of 2 and subtracting it from the image within the Gaussian Pyramid at level L. This is useful in data compression.



Linear Algebra: Least Squares

The least squares problem is of the form

$$Ax = b$$

The least squares problem can be formulated using

$$\operatorname{argmin}_x \|Ax - b\|^2$$

This can be derived using calculus by computing the gradient and set it equal to zero i.e.

$$f(x) = (Ax - b)^2$$

$$\nabla f(x) = 2A^T(Ax - b) = 0$$

From this it implies that

$$x = (A^T A)^{-1} A^T b$$

Linear Algebra: Least Squares

Derive the least squares solution of the following problem

$$Ax = b$$

where

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 3 \\ 3 \\ 5 \end{bmatrix}$$

Solution: $x = \begin{bmatrix} 0 \\ 6 \\ \frac{5}{5} \end{bmatrix}$

Linear Algebra: Inverse of a 3x3 Matrix

Calculate the inverse of the following 3x3 matrix

$$A = \begin{bmatrix} 13 & 18 & 14 \\ 9 & 15 & 10 \\ 12 & 11 & 7 \end{bmatrix}$$

$$\text{Solution: } A^{-1} = \begin{bmatrix} 0.0067 & -0.1319 & 0.175 \\ -0.0767 & 0.1036 & 0.0054 \\ 0.109 & 0.0633 & -0.1655 \end{bmatrix}$$

Linear Algebra: Ridge Regression

The Least Squares problem can be derived when the square matrix $A^T A$ is invertible. $A^T A$ is invertible if and only if $A^T A$ is of full rank. Now, let's consider the previous example

$$A^T A = \begin{bmatrix} 2 & -2 & 5 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

Linear Algebra: Ridge Regression

The Least Squares problem can be derived when the square matrix $A^T A$ is invertible. $A^T A$ is invertible if and only if $A^T A$ is of full rank. Now, let's consider the previous example

$$A^T A = \begin{bmatrix} 2 & -2 & 5 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

The matrix is a rank-3 matrix and thus is a full-rank matrix. This means that the rows and columns are linearly independent. This can be computed using the numpy library `np.linalg.matrix_rank()`. This matrix is thus invertible.

Linear Algebra: Ridge Regression

The Least Squares problem can be derived when the square matrix $A^T A$ is invertible. $A^T A$ is invertible if and only if $A^T A$ is of full rank. Now, let's consider the previous example

$$A^T A = \begin{bmatrix} 2 & -2 & 5 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

The matrix is a rank-3 matrix and thus is a full-rank matrix. This means that the rows and columns are linearly independent. This can be computed using the numpy library `np.linalg.matrix_rank()`. This matrix is thus invertible.

The inverse of this matrix exists and can be computed using the `np.linalg.inv()` function.

$$(A^T A)^{-1} = \begin{bmatrix} -2 & 5 & -4 \\ 0 & 0 & 1 \\ 1 & -2 & 2 \end{bmatrix}$$

Linear Algebra: Ridge Regression

Now, let's consider the following matrix

$$(A^T A)^{-1} = \begin{bmatrix} 2 & -2 & 4 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

Linear Algebra: Ridge Regression

Now, let's consider the following matrix

$(A^T A)^{-1} = \begin{bmatrix} 2 & -2 & 4 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$ The matrix is a rank-2 matrix and thus is not a full-rank matrix. This means that the rows and columns are not linearly independent. This can be computed using the numpy library `np.linalg.matrix_rank()`. This matrix is thus non-invertible.

Linear Algebra: Ridge Regression

Now, let's consider the following matrix

$(A^T A)^{-1} = \begin{bmatrix} 2 & -2 & 4 \\ 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$ The matrix is a rank-2 matrix and thus is not a full-rank matrix. This means that the rows and columns are not linearly independent. This can be computed using the numpy library `np.linalg.matrix_rank()`. This matrix is thus non-invertible.

This can be solved using the following constrained minimization which enforces the values of the least squares solution to have small values i.e.

$$\operatorname{argmin}_x ||Ax - b||^2 + \lambda ||x||^2$$

This has a closed form solution that is given by

$$x = (A^T A + \lambda I)^{-1} A^T b$$

where I is the identity matrix and λ is a small value (1E-6).

Linear Algebra: Homogeneous Systems of Equations

Find the non-trivial solution to the system of homogeneous linear equations of the form

$$Ax = 0$$

where

$$A = \begin{bmatrix} 1 & 2 & 3 & 2 \\ 1 & 3 & 5 & 5 \\ 2 & 4 & 7 & 1 \\ -1 & -2 & -6 & 7 \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

Solution: $x = \begin{bmatrix} 7k \\ -9k \\ 3k \\ k \end{bmatrix}$

Linear Algebra: Homogeneous Systems of Equations

The numerically best way to solve this problem is to perform Singular Value Decomposition (SVD) on the matrix A. Singular Vector Decomposition (SVD) factors the matrix into a diagonal matrix D and two orthogonal matrices U, V such that

$$A = UDV^T$$

The diagonal entries of D are related to eigenvalues of $A^T A$.

$$D = \begin{bmatrix} 12.5441 & 0.0 & 0.0 & 0.0 \\ 0.0 & 8.9586 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.6229 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

This means that matrix A has 3 linearly-independent equations and thus is a rank-3 matrix. The solution to this problem can be obtained from the last column of V that corresponds to the column with 0 eigenvalue.

$$V = \begin{bmatrix} -0.2068 & 0.0202 & -0.7790 & 0.5916 \\ -0.4516 & 0.118 & -0.4551 & -0.7606 \\ -0.8678 & -0.0725 & 0.4211 & 0.2536 \\ -0.0128 & 0.9920 & 0.0934 & 0.0845 \end{bmatrix}$$

Linear Algebra: Homogeneous Systems of Equations

The numerically best way to solve this problem is to perform Singular Value Decomposition (SVD) on the matrix A. Singular Vector Decomposition (SVD) factors the matrix into a diagonal matrix D and two orthogonal matrices U, V such that

$$A = UDV^T$$

The diagonal entries of D are related to eigenvalues of $A^T A$.

$$D = \begin{bmatrix} 12.5441 & 0.0 & 0.0 & 0.0 \\ 0.0 & 8.9586 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.6229 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

This means that matrix A has 3 linearly-independent equations and thus is a rank-3 matrix. The solution to this problem can be obtained from the last column of V that corresponds to the column with 0 eigenvalue.

$$V = \begin{bmatrix} -0.2068 & 0.0202 & -0.7790 & 0.5916 \\ -0.4516 & 0.118 & -0.4551 & -0.7606 \\ -0.8678 & -0.0725 & 0.4211 & 0.2536 \\ -0.0128 & 0.9920 & 0.0934 & 0.0845 \end{bmatrix} \quad \text{Previous Solution: } x = \begin{bmatrix} 7k \\ -9k \\ 3k \\ k \end{bmatrix}$$

Linear Algebra: Homogeneous Systems of Equations

The numerically best way to solve this problem is to perform Singular Value Decomposition (SVD) on the matrix A. Singular Vector Decomposition (SVD) factors the matrix into a diagonal matrix D and two orthogonal matrices U, V such that

$$A = UDV^T$$

The diagonal entries of D are related to eigenvalues of $A^T A$.

$$D = \begin{bmatrix} 12.5441 & 0.0 & 0.0 & 0.0 \\ 0.0 & 8.9586 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.6229 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

This means that matrix A has 3 linearly-independent equations and thus is a rank-3 matrix. The solution to this problem can be obtained from the last column of V that corresponds to the column with 0 eigenvalue.

$$V = \begin{bmatrix} -0.2068 & 0.0202 & -0.7790 & 0.5916 \\ -0.4516 & 0.118 & -0.4551 & -0.7606 \\ -0.8678 & -0.0725 & 0.4211 & 0.2536 \\ -0.0128 & 0.9920 & 0.0934 & 0.0845 \end{bmatrix} \quad \text{Previous Solution: } x = \begin{bmatrix} 7k \\ -9k \\ 3k \\ k \end{bmatrix} \quad \text{If } k = 0.00845 \quad x = \begin{bmatrix} 0.5916 \\ -0.7606 \\ 0.2536 \\ 0.0845 \end{bmatrix}$$

Image Registration

Image registration is the process of transforming different sets of data (pixels from different images) into one coordinate system.



Image Registration

Image registration is the process of transforming different sets of data (pixels from different images) into one coordinate system.



All the landmark points occur roughly at the same location -> they share the same coordinate system.

Image Registration: Affine Transformation



Image Registration: Affine Transformation

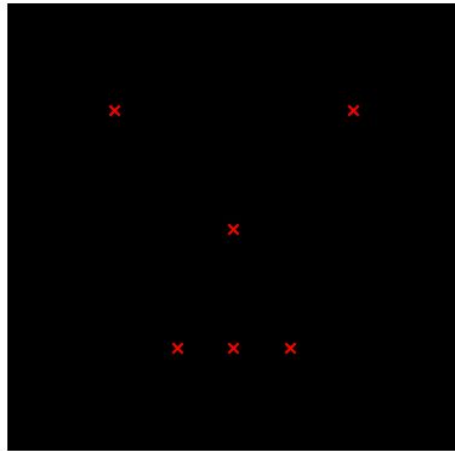


Image Registration: Affine Transformation

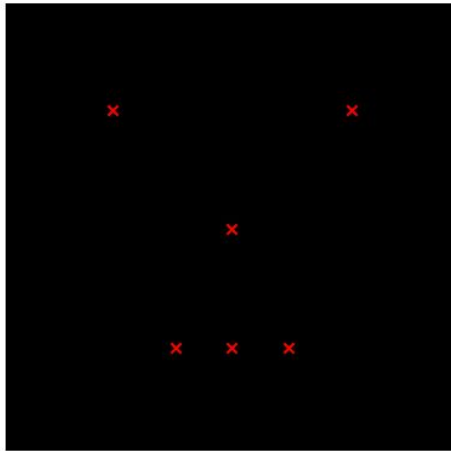


Image Registration: Affine Transformation

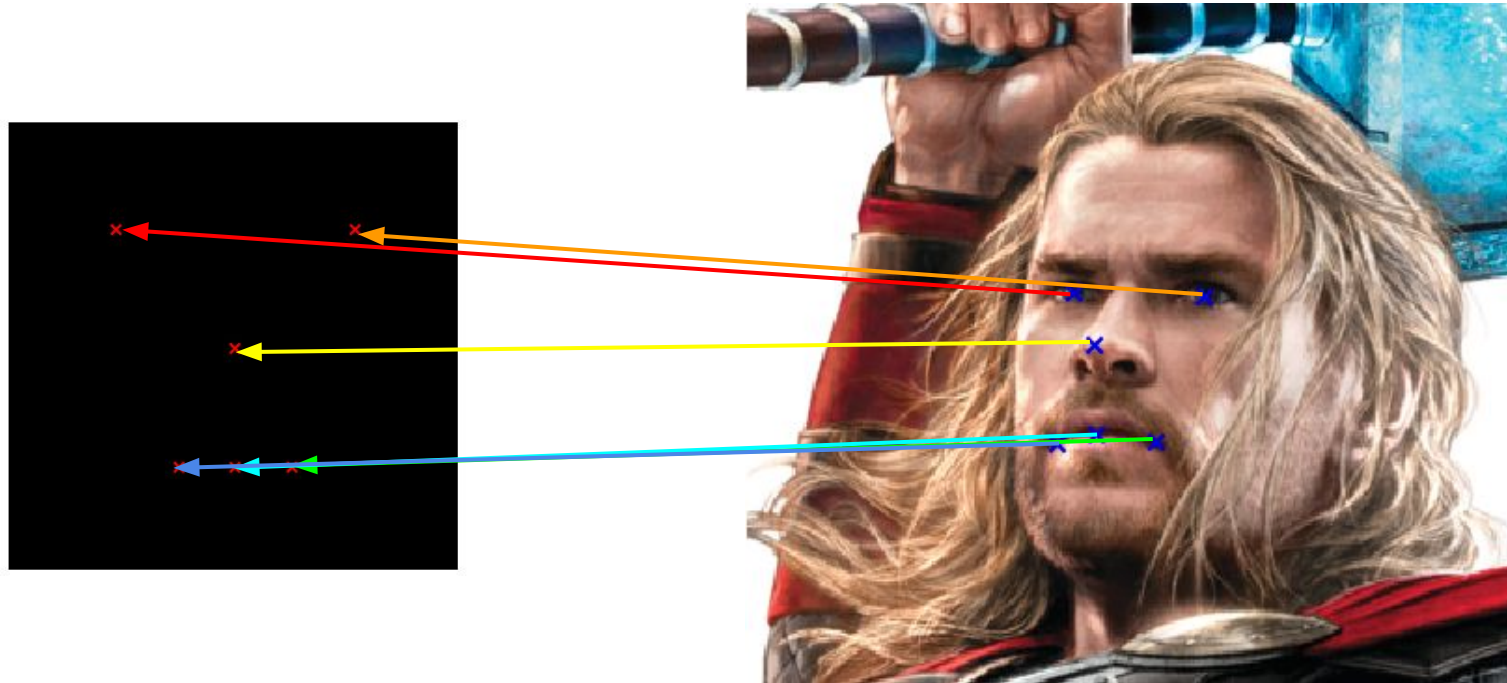
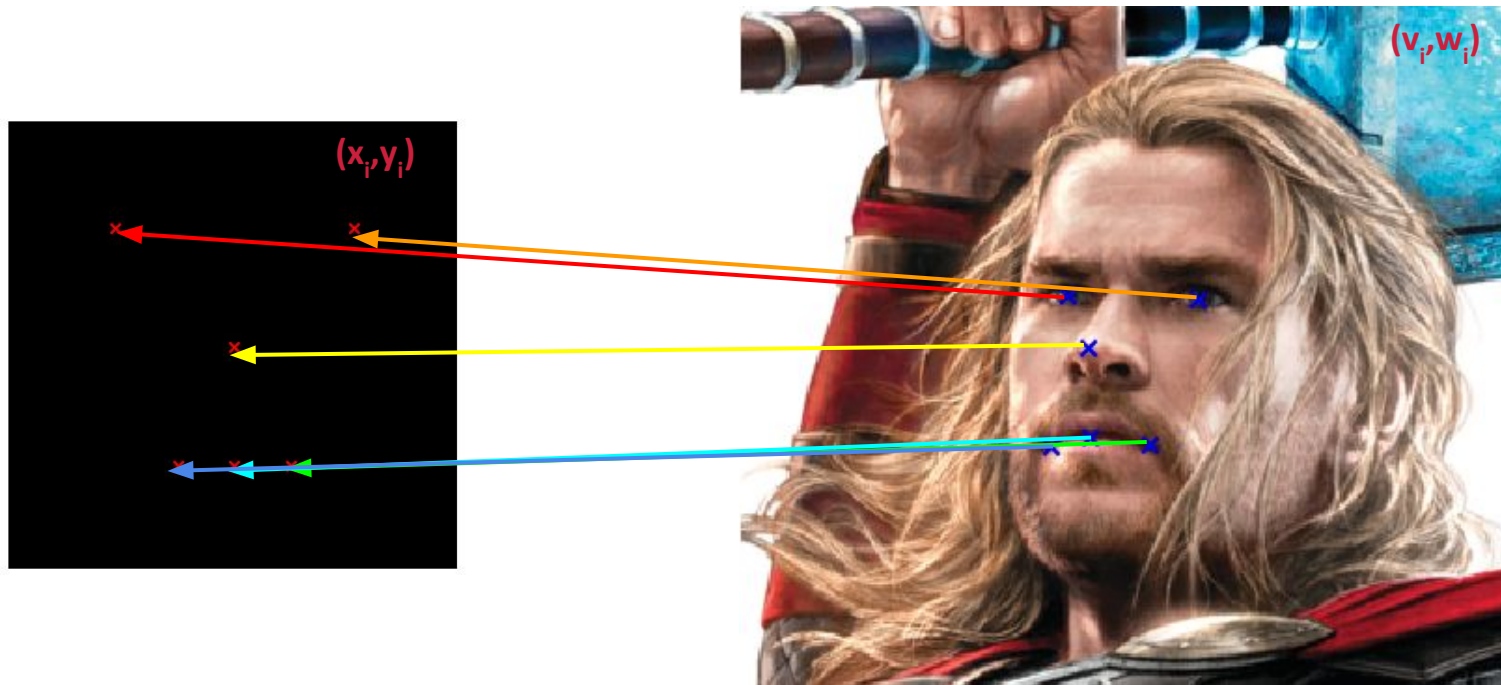


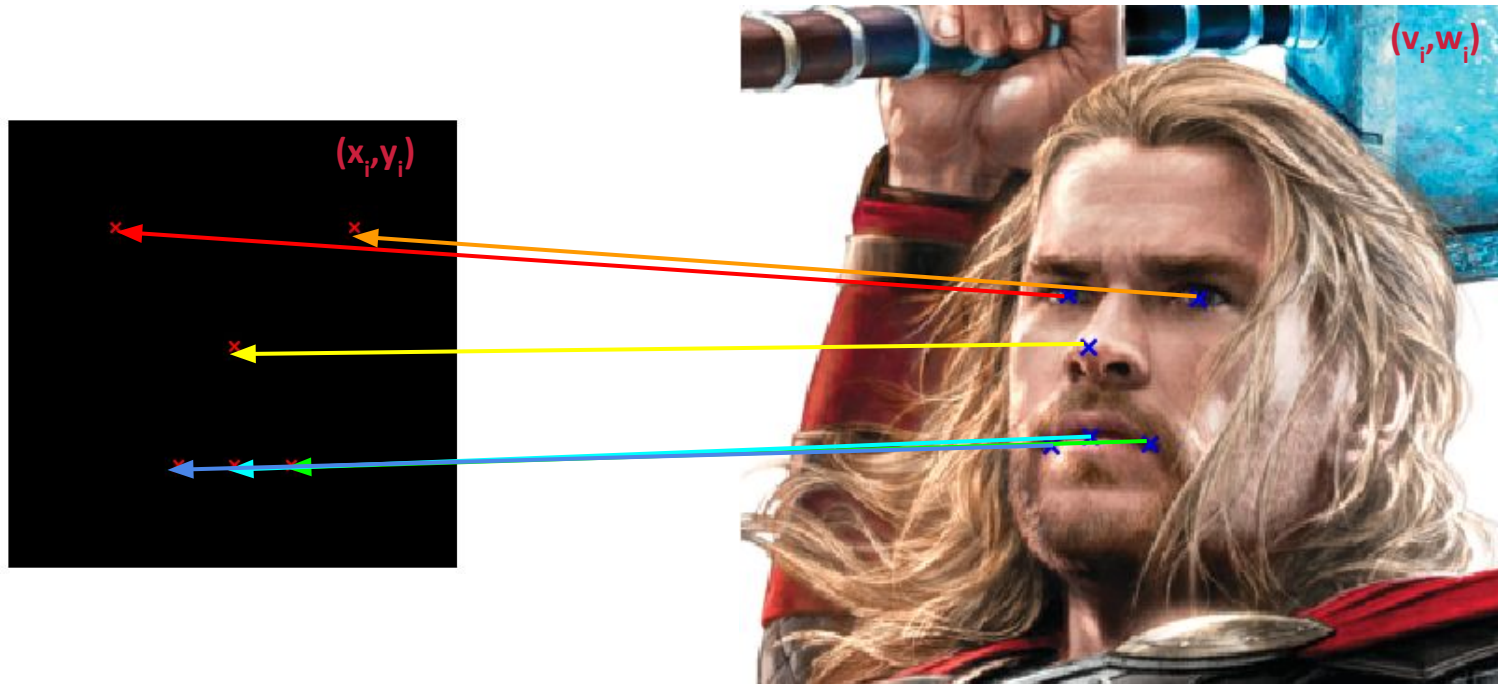
Image Registration: Affine Transformation



Mathematically, this can be formulated as

$$[x_i, y_i, 1] = [v_i, w_i, 1] \begin{bmatrix} t_{1,1} & t_{1,2} & 0 \\ t_{2,1} & t_{2,2} & 0 \\ t_{3,1} & t_{3,2} & 1 \end{bmatrix}$$

Image Registration: Affine Transformation



Now, in this case we have six points and therefore the equation needs to be rewritten as

$$[x_i, y_i, 1] = [v_i, w_i, 1] \begin{bmatrix} t_{1,1} & t_{1,2} & 0 \\ t_{2,1} & t_{2,2} & 0 \\ t_{3,1} & t_{3,2} & 1 \end{bmatrix} \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \\ x_5 & y_5 & 1 \\ x_6 & y_6 & 1 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} & t_{1,2} & 0 \\ t_{2,1} & t_{2,2} & 0 \\ t_{3,1} & t_{3,2} & 1 \end{bmatrix}$$

Image Registration: Affine Transformation

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \\ x_5 & y_5 & 1 \\ x_6 & y_6 & 1 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} & t_{1,2} & 0 \\ t_{2,1} & t_{2,2} & 0 \\ t_{3,1} & t_{3,2} & 1 \end{bmatrix}$$

Given that the pixel coordinates for the six points are known at both source and reference image, the only parameters that are unknown are $t_{1,1}$, $t_{1,2}$, $t_{1,3}$, $t_{2,1}$, $t_{2,2}$, $t_{2,3}$.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} \\ t_{2,1} \\ t_{3,1} \end{bmatrix} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,2} \\ t_{2,2} \\ t_{3,2} \end{bmatrix}$$

Image Registration: Affine Transformation

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \\ x_5 & y_5 & 1 \\ x_6 & y_6 & 1 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} & t_{1,2} & 0 \\ t_{2,1} & t_{2,2} & 0 \\ t_{3,1} & t_{3,2} & 1 \end{bmatrix}$$

Given that the pixel coordinates for the six points are known at both source and reference image, the only parameters that are unknown are $t_{1,1}$, $t_{1,2}$, $t_{1,3}$, $t_{2,1}$, $t_{2,2}$, $t_{2,3}$.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \overset{\text{A}}{\begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix}} \begin{bmatrix} t_{1,1} \\ t_{2,1} \\ t_{3,1} \end{bmatrix} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \overset{\text{A}}{\begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix}} \begin{bmatrix} t_{1,2} \\ t_{2,2} \\ t_{3,2} \end{bmatrix}$$

Image Registration: Affine Transformation

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \\ x_5 & y_5 & 1 \\ x_6 & y_6 & 1 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} & t_{1,2} & 0 \\ t_{2,1} & t_{2,2} & 0 \\ t_{3,1} & t_{3,2} & 1 \end{bmatrix}$$

Given that the pixel coordinates for the six points are known at both source and reference image, the only parameters that are unknown are $t_{1,1}$, $t_{1,2}$, $t_{1,3}$, $t_{2,1}$, $t_{2,2}$, $t_{2,3}$.

b

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} \\ t_{2,1} \\ t_{3,1} \end{bmatrix}$$

c

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,2} \\ t_{2,2} \\ t_{3,2} \end{bmatrix}$$

Image Registration: Affine Transformation

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \\ x_5 & y_5 & 1 \\ x_6 & y_6 & 1 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} & t_{1,2} & 0 \\ t_{2,1} & t_{2,2} & 0 \\ t_{3,1} & t_{3,2} & 1 \end{bmatrix}$$

Given that the pixel coordinates for the six points are known at both source and reference image, the only parameters that are unknown are $t_{1,1}$, $t_{1,2}$, $t_{1,3}$, $t_{2,1}$, $t_{2,2}$, $t_{2,3}$.

$$\begin{array}{ccc} \text{b} & \text{A} & \text{x} \\ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} & = & \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} \\ t_{2,1} \\ t_{3,1} \end{bmatrix} \end{array} \quad \begin{array}{ccc} \text{c} & \text{A} & \text{y} \\ \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} & = & \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,2} \\ t_{2,2} \\ t_{3,2} \end{bmatrix} \end{array}$$

Image Registration: Affine Transformation

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \\ x_5 & y_5 & 1 \\ x_6 & y_6 & 1 \end{bmatrix} = \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} & t_{1,2} & 0 \\ t_{2,1} & t_{2,2} & 0 \\ t_{3,1} & t_{3,2} & 1 \end{bmatrix}$$

Given that the pixel coordinates for the six points are known at both source and reference image, the only parameters that are unknown are $t_{1,1}$, $t_{1,2}$, $t_{1,3}$, $t_{2,1}$, $t_{2,2}$, $t_{2,3}$.


$$\begin{array}{ccc} \text{b} & \text{A} & \text{x} \\ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} & = & \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,1} \\ t_{2,1} \\ t_{3,1} \end{bmatrix} \end{array} \quad \begin{array}{ccc} \text{c} & \text{A} & \text{y} \\ \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} & = & \begin{bmatrix} v_1 & w_1 & 1 \\ v_2 & w_2 & 1 \\ v_3 & w_3 & 1 \\ v_4 & w_4 & 1 \\ v_5 & w_5 & 1 \\ v_6 & w_6 & 1 \end{bmatrix} \begin{bmatrix} t_{1,2} \\ t_{2,2} \\ t_{3,2} \end{bmatrix} \end{array}$$


Solution using Least Squares

$$\begin{aligned} x &= (A^T A)^{-1} A^T b \\ y &= (A^T A)^{-1} A^T c \end{aligned}$$

Thank you for your attention.

If you have any questions, now is the right time to ask!
Contact me on the details below for further info:

 Reuben Farrugia

 (+356) 2340 3088

 <https://www.um.edu.mt/staff/reuben.farrugia>

 reuben.farrugia@um.edu.mt