Image Analysis 2

12-01-2015

APIE 2014/2015

Overview

- Orientation && Eccentricity
- Pattern Matching
  - Cross correlation
- Particle Velocimetry
  - Particle Image Velocimetry
  - Particle Tracking Velocimetry

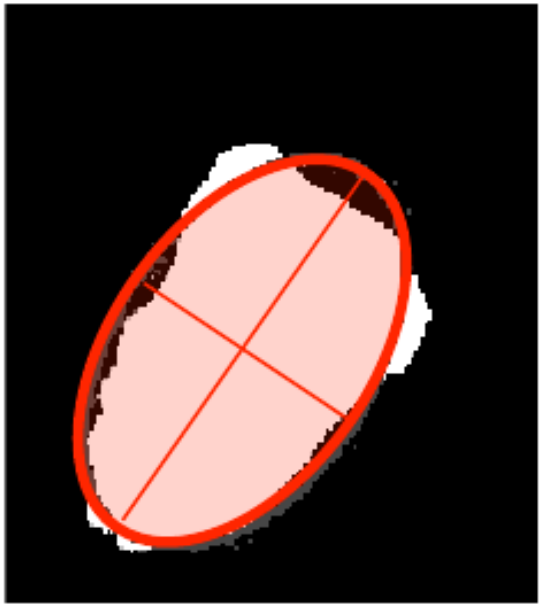
Region labeling - Moments

- Raw moments

$$M_{pq} = \sum_x \sum_y x^p y^q$$

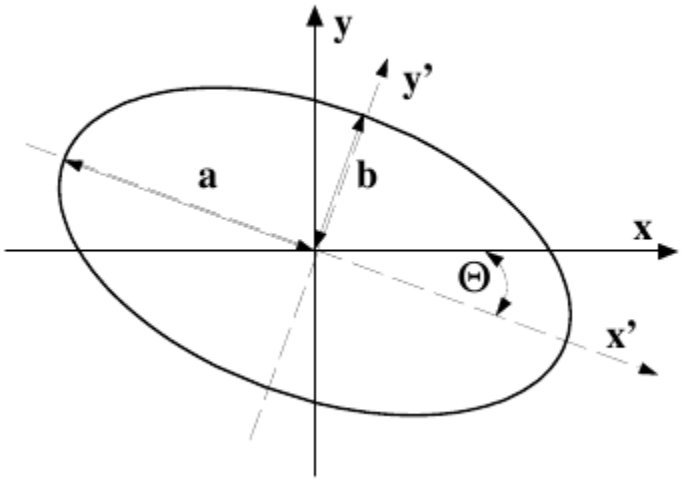
- M_{00} : area, #pixels
- M_{10} : sum over x
- M_{01} : sum over y
- Centroid coordinates: $\overline{x} = \frac{M_{10}}{M_{00}}, \overline{y} = \frac{M_{01}}{M_{00}}$

- Orientation and Eccentricity
  (details on next lecture)



Region labeling - Moments

Orientation:
The orientation of the object is defined as the tilt angle between the x-axes and the axis, around which the object can be rotated with minimal inertia (i.e. the direction of the major semi-axis a).



Eccentricity:
The eccentricity ε of an ellipse can be thought of as a measure of how much the conic section deviates from being circular. It can be directly derived from the semi-major and semi-minor axes a and b of the object:

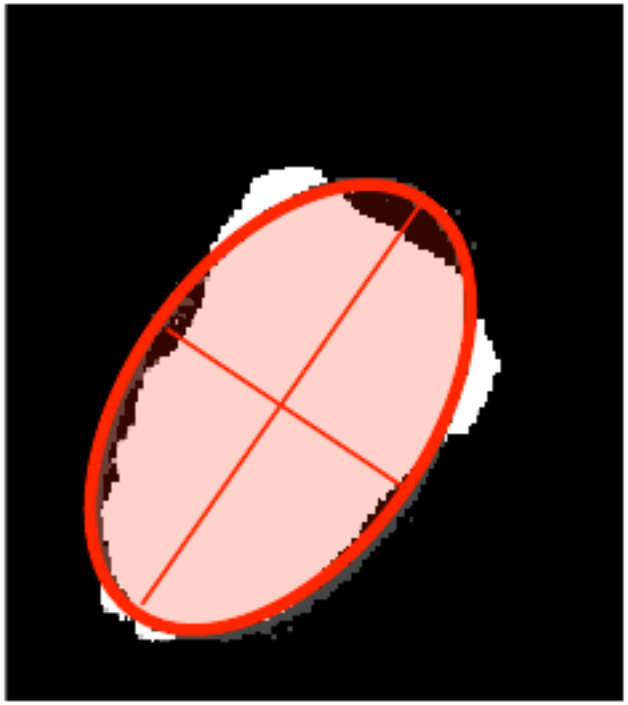
$$\varepsilon = \frac{\sqrt{a^2 - b^2}}{a}$$

The eccentricity can have values from 0 to 1. It is 0 with a perfectly round object and 1 with a line shaped object.

Region labeling – Moments

Raw moments:

$$M_{pq} = \sum_x \sum_y x^p y^q$$



Central moments:
From the raw moments the central moments can be derived by reducing the raw moments with the center of gravity ($\overline{x}; \overline{y}$) of the object, so all the central moments refer to the center of gravity of the object.

$$\mu_{p,q} = \int \int (x - \overline{x})^p (y - \overline{y})^q f(x,y)\ \mathrm{d}x\,\mathrm{d}y$$

The central moments of first or higher order can directly be derived from the raw moments by

$$\mu_{p,q} = \frac{M_{p,q}}{M_{0,0}} - \left(\frac{M_{1,0}}{M_{0,0}}\right)^p * \left(\frac{M_{0,1}}{M_{0,0}}\right)^q$$

Ref: [1]

Region labeling - Moments

Examples of
central moments:

$$\mu_{1,0} = \frac{m_{1,0}}{m_{0,0}} - \left(\frac{m_{1,0}}{m_{0,0}}\right) = 0$$

$$\mu_{0,1} = \frac{m_{0,1}}{m_{0,0}} - \left(\frac{m_{0,1}}{m_{0,0}}\right) = 0$$

$$\mu_{2,0} = \frac{m_{2,0}}{m_{0,0}} - \left(\frac{m_{1,0}}{m_{0,0}}\right)^2 = \frac{m_{2,0}}{m_{0,0}} - x_c^2$$

$$\mu_{0,2} = \frac{m_{0,2}}{m_{0,0}} - \left(\frac{m_{0,1}}{m_{0,0}}\right)^2 = \frac{m_{0,2}}{m_{0,0}} - y_c^2$$

$$\mu_{1,1} = \frac{m_{1,1}}{m_{0,0}} - \left(\frac{m_{1,0}}{m_{0,0}}\right) * \left(\frac{m_{0,1}}{m_{0,0}}\right) = \frac{m_{1,1}}{m_{0,0}} - x_c * y_c$$

Region labeling - Moments

- The three central moments of second order build the components of the inertial tensor of the rotation of the object about its center of gravity:

$$J_{\alpha\beta} = m_i \sum_i \left(\sum_\gamma \delta_{\alpha\beta} r_{i\gamma}^2 - r_{i\alpha} r_{i\beta} \right) = \begin{bmatrix} \mu_{0,2} & -\mu_{1,1} \\ -\mu_{1,1} & \mu_{2,0} \end{bmatrix}$$

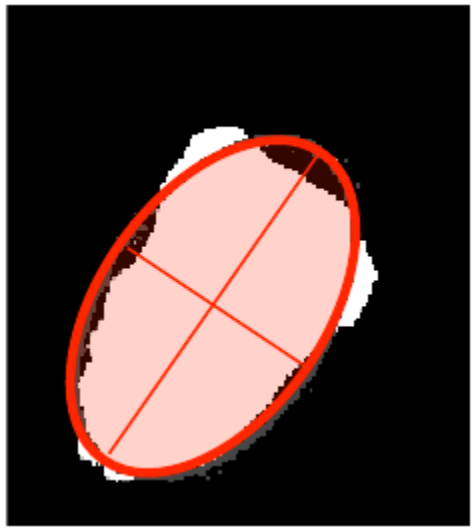
- **Semi-major and Semi-major axes a and b:**
  The main inertial axes of the object are in the direction of the eigenvectors of the moment of inertia tensor J. They can be derived by calculating the eigenvalues of the inertial tensor:

$$\lambda_{+,-} = \frac{1}{2}(\mu_{2,0} + \mu_{0,2}) \pm \sqrt{4\mu_{1,1}^2 + (\mu_{2,0} - \mu_{0,2})^2}$$

  the direction of the principal axes can than be found by solving:

$$J\omega_{\mathbf{a},\mathbf{b}} = \lambda_{-,+}\omega_{\mathbf{a},\mathbf{b}}$$

$$(J - \lambda_{-,+} I)\omega_{\mathbf{a},\mathbf{b}} = 0$$



Ref: [6]

Region labeling – Moments

The directions of the semi-major and -minor axes correspond to those of the eigenvectors ω_i. The lengths of the semi-principal axes are then found by:

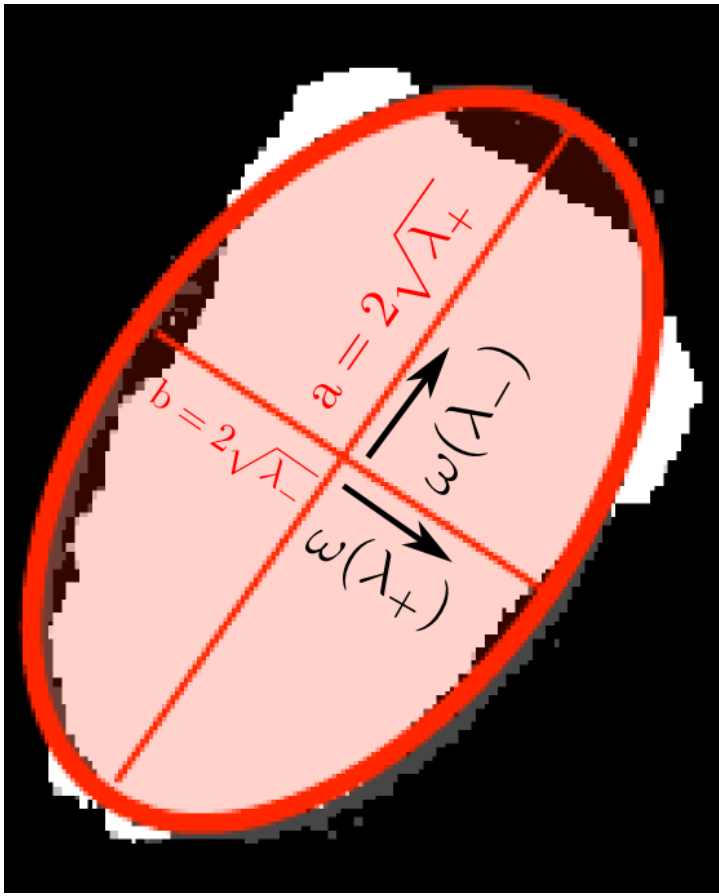
$$a = 2\sqrt{\lambda_+}\,,\; b = 2\sqrt{\lambda_-}$$

in which a and b are the semi-major and -minor axis, respectively.

Eccentricity revisited:
The eccentricity can also be expressed in terms of the eigenvalues:

$$\varepsilon = \frac{\sqrt{\mathrm{a}^2 - \mathrm{b}^2}}{a}$$

$$= \frac{\sqrt{\lambda_+ - \lambda_-}}{\sqrt{\lambda_+}} = \sqrt{1 - \frac{\lambda_-}{\lambda_+}}$$



Region labeling - Moments

- **Orientation:**
  The orientation of the object is defined as the tilt angle between the x-axis and the axis, around which the object can be rotated with minimal inertia (i.e. the direction of the semi-major axis a). This corresponds to the eigenvector with minimal eigenvalue. In this direction the object has its biggest extension. It is calculated as follows:

$$\theta = \frac{1}{2}\arctan\left(\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}\right)$$

$\mu_{2,0}-\mu_{0,2}$	$\mu_{1,1}$	θ	
Zero	Zero	0^o	
Zero	Positive	$+45^o$	
Zero	Negative	-45^o	
Positive	Zero	0^o	
Negative	Zero	-90^o	
Positive	Positive	$\frac{1}{2}\arctan\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}$	$0^o < \theta < 45^o$
Positive	Negative	$\frac{1}{2}\arctan\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}$	$-45^o < \theta < 0^o$
Negative	Positive	$\frac{1}{2}\arctan\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}} + 90^o$	$45^o < \theta < 90^o$
Negative	Negative	$\frac{1}{2}\arctan\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}} - 90^o$	$-90^o < \theta < -45^o$

Table 1: Tilt angle θ (orientation) of an image ellipsis

Ref: [1]

NB: note that in a MATLAB matrix, x is down and y is left. To get the tilt angle relative to the *horizontal* axis, you can use $\theta = \text{atand}\left(-\omega_a(1)/\omega_a(2)\right)$, which gives an angle $-90^\circ \leq \theta < 90^\circ$.

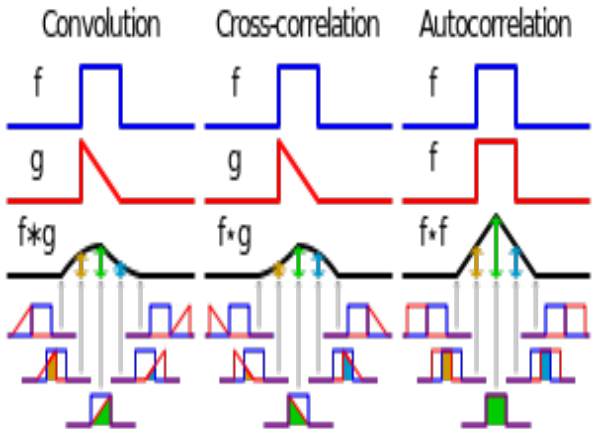
Pattern Matching – cross correlation

- In signal processing, **cross-correlation** is a measure of similarity of two waveforms as a function of a time-lag applied to one of them.
- Definition:
  - For continuous signal:

$$(f \star g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f^*(\tau)\ g(t+\tau)\, d\tau,$$

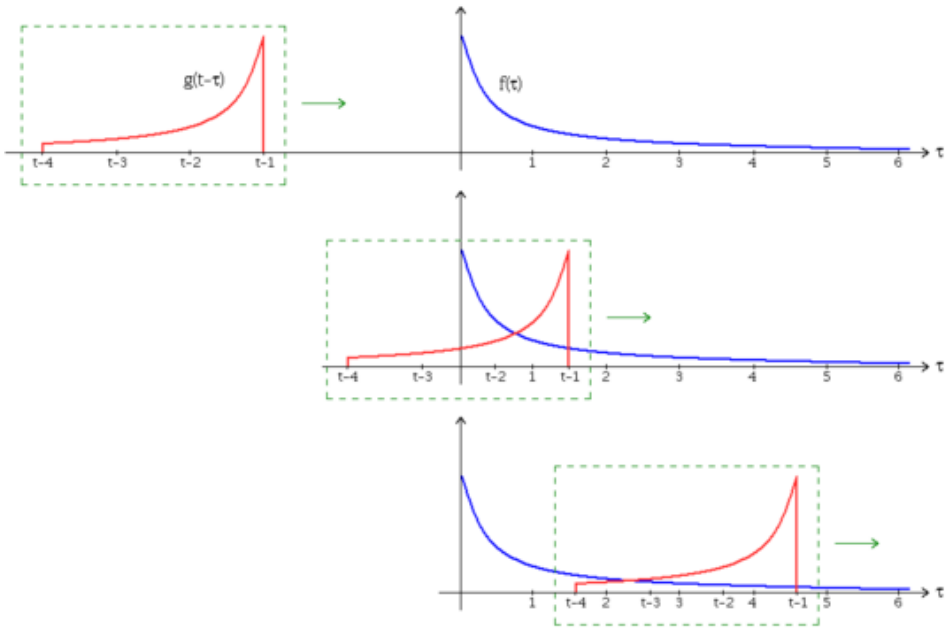
  - For digital signal:

$$(f \star g)[n] \stackrel{\text{def}}{=} \sum_{m=-\infty}^{\infty} f^*[m]\ g[n+m].$$



Explanation:
As an example, consider two real valued functions **f** and **g** differing only by an unknown shift along the x-axis. One can use the cross-correlation to find how much **g** must be shifted along the x-axis to make it identical to **f**. The formula essentially slides **g** the function along the x-axis, calculating the integral of their product at each position. When the functions match, the value of **(f*g)** is maximized.

Ref: *[2]*, *[3]*



Pattern Matching – normalized cross correlation

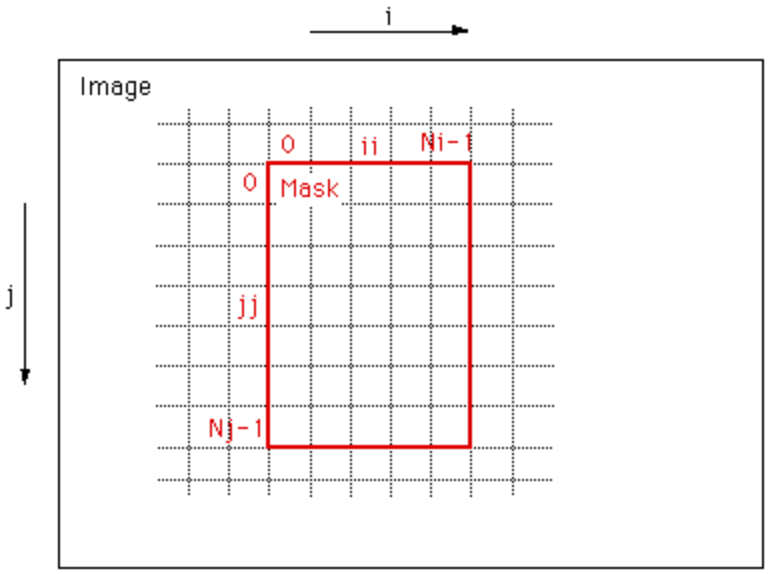
- For image-processing applications in which the brightness of the image and template can vary due to lighting and exposure conditions, the images can be first normalized. This is typically done at every step by subtracting the mean and dividing by the standard deviation.

$$\frac{1}{n}\sum_{x,y}\frac{(f(x,y)-\overline{f})(t(x,y)-\overline{t})}{\sigma_f\sigma_t}$$

- $f(x,y)$: sub-image f
- $t(x,y)$: template t
- $\overline{f}, \overline{t}$: average of f and t
- σ_f, σ_t : standard deviation of f and t
- n : number of pixels

Pattern Matching

One approach to identifying a pattern within an image uses cross correlation of the image with a suitable *mask*. Where the mask and the pattern being sought are similar the cross correlation will be high. The mask is itself an image which needs to have the same functional appearance as the pattern to be found.
Consider the image to the right in black and the mask shown in red. The mask is centered at every pixel in the image and the cross correlation calculated, this forms a 2D array of correlation coefficients.



The un-normalized **correlation coefficient** at position (i,j) on the image is given by:

$$r[i][j] = \sum_{jj=-Nj/2}^{jj<Nj/2} \sum_{ii=-Ni/2}^{ii<Ni/2} (mask[ii+Ni/2][jj+Nj/2] - \overline{mask})\,(image[i+ii][j+jj] - \overline{image})$$

where $\overline{mask}$ is the mean of the masks pixels and $\overline{image}$ is the mean of the image pixels under (covered by) the mask.

Ref: *[2]*

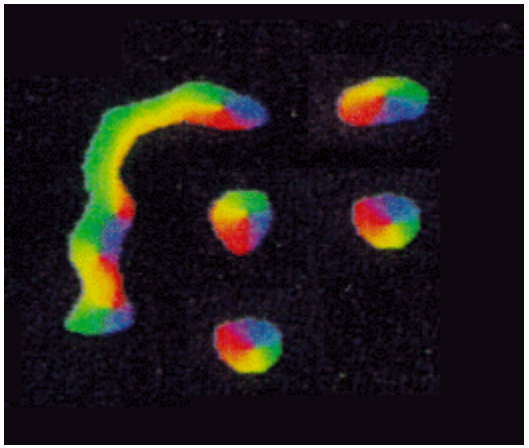
Pattern Matching

To consider

- The process can be extremely time consuming, the 2D cross correlation function needs to be computed for every point in the image. Calculation of the cross correlation function is itself a N^2 operation. Ideally the mask should be chosen as small as practicable.
- In many image identification processes the mask may need to be rotated and/or scaled at each position.

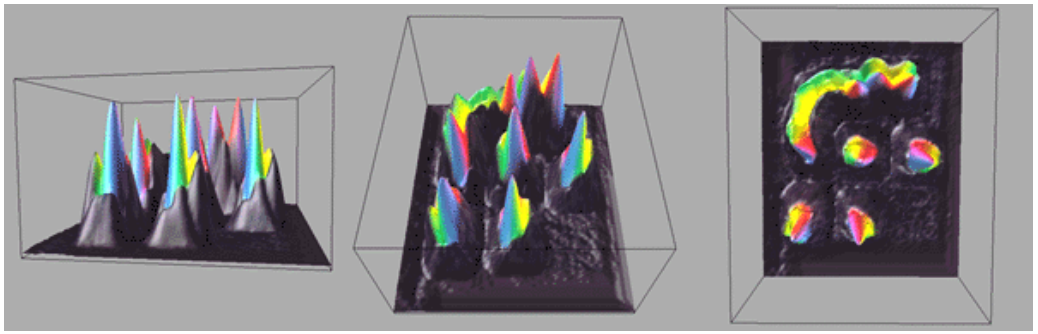
Example

Input



Mask

Output



Ref: *[2]*

Particle Velocimetry

- **Particle image velocimetry** (**PIV**) is an optical method of flow visualization used in education and research. It is used to obtain instantaneous velocity measurements and related properties in fluids. The fluid is seeded with tracer particles which, for sufficiently small particles, are assumed to faithfully follow the flow dynamics (the degree to which the particles faithfully follow the flow is represented by the Stokes number). The fluid with entrained particles is illuminated so that particles are visible. The motion of the seeding particles is used to calculate speed and direction (the velocity field) of the flow being studied.

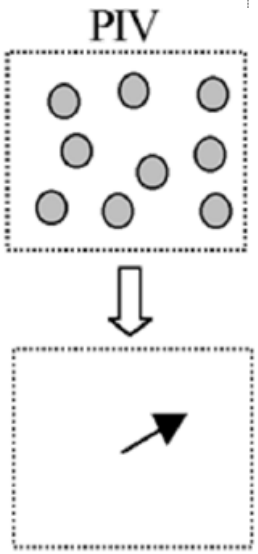
- **Assumptions:**
  - Seeded the flow with small particles (~ μm in size)
  - the particle tracers move with the same velocity as local flow velocity!

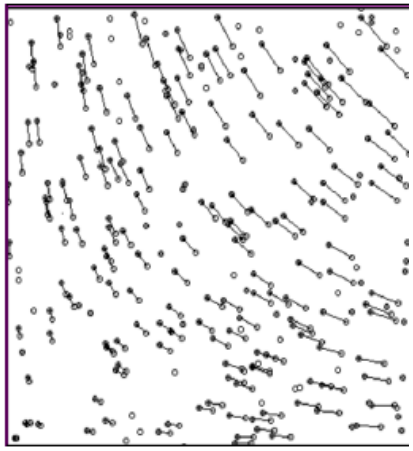
Ref: *[4]*

Particle Velocimetry - methods

PIV – Particle Image Velocimetry

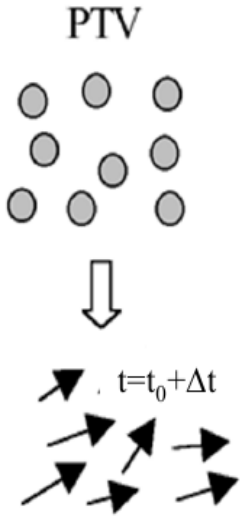
- Tracking *a group of particles*
- Applicable to high particle image density
- Resolution limited by the size of the interrogation window
- Velocity vector can be at regular grid points

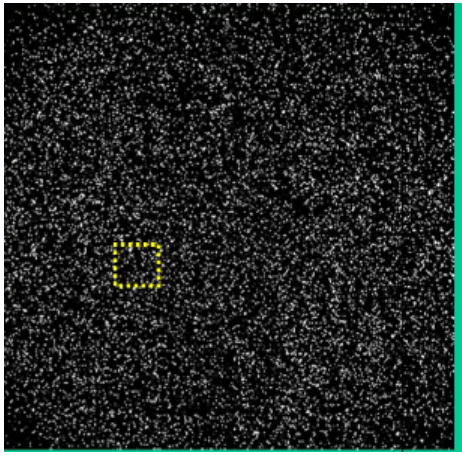


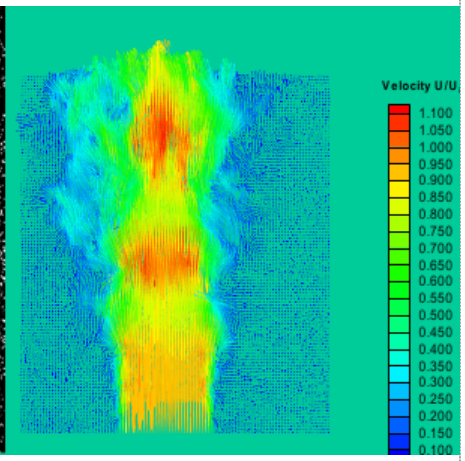


PTV – Particle Tracking Velocimetry

- Tracking *individual particles*
- Limited to low particle image density
- Resolution limited by the number of tracer particles
- Velocity wherever tracer particles exist

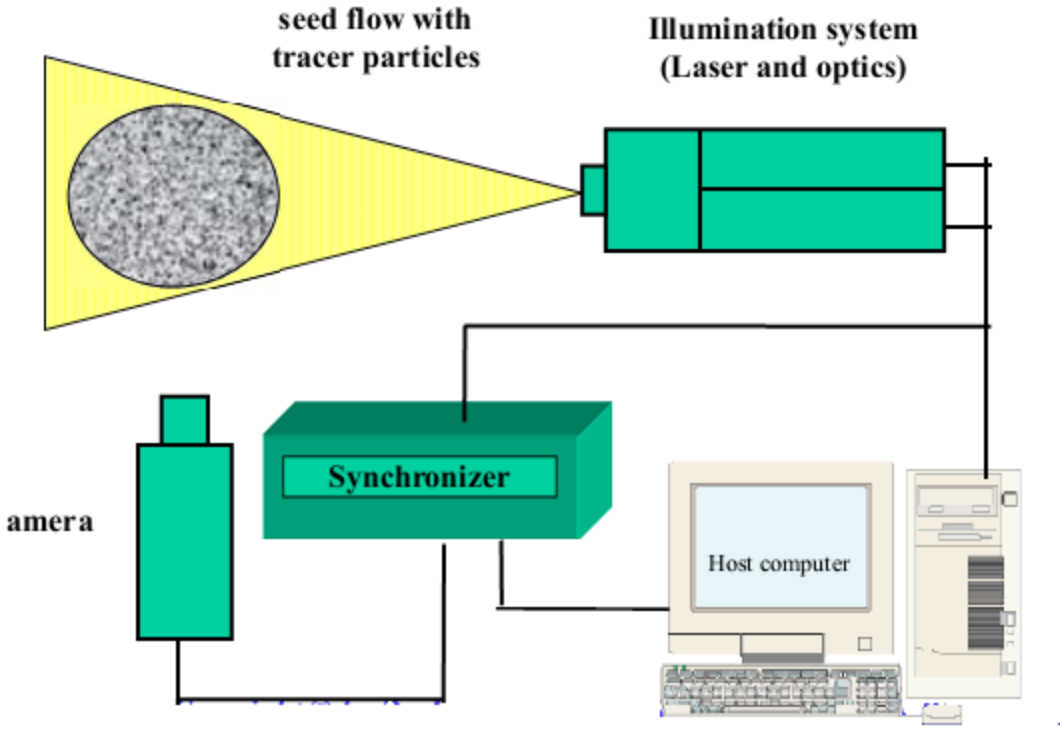




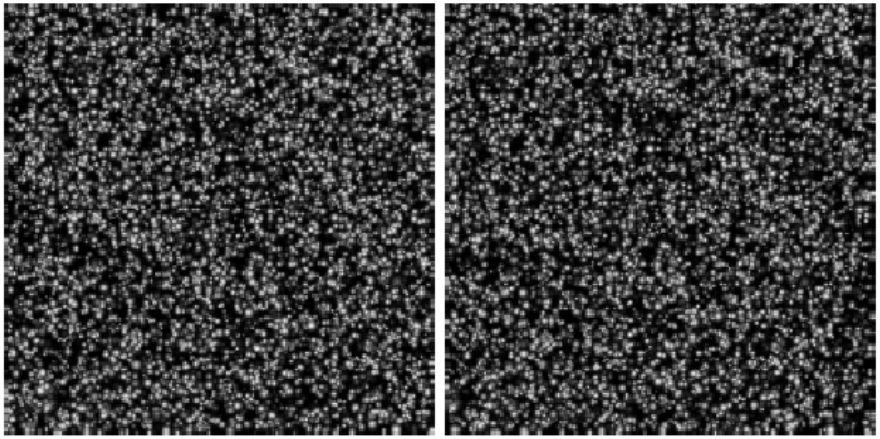


PV – system setup

Particle tracers: to track the fluid movement.
Illumination system: to illuminate the flow field in the interest region.
Camera: to capture the images of the particle tracers.
Synchronizer: the control the timing of the laser illumination and camera acquisition.
Host computer: to store the particle images and conduct image processing.

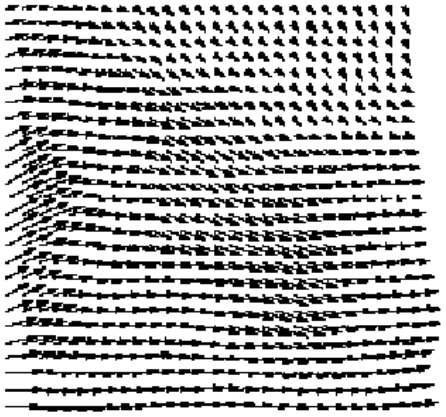


Particle Image Velocimetry

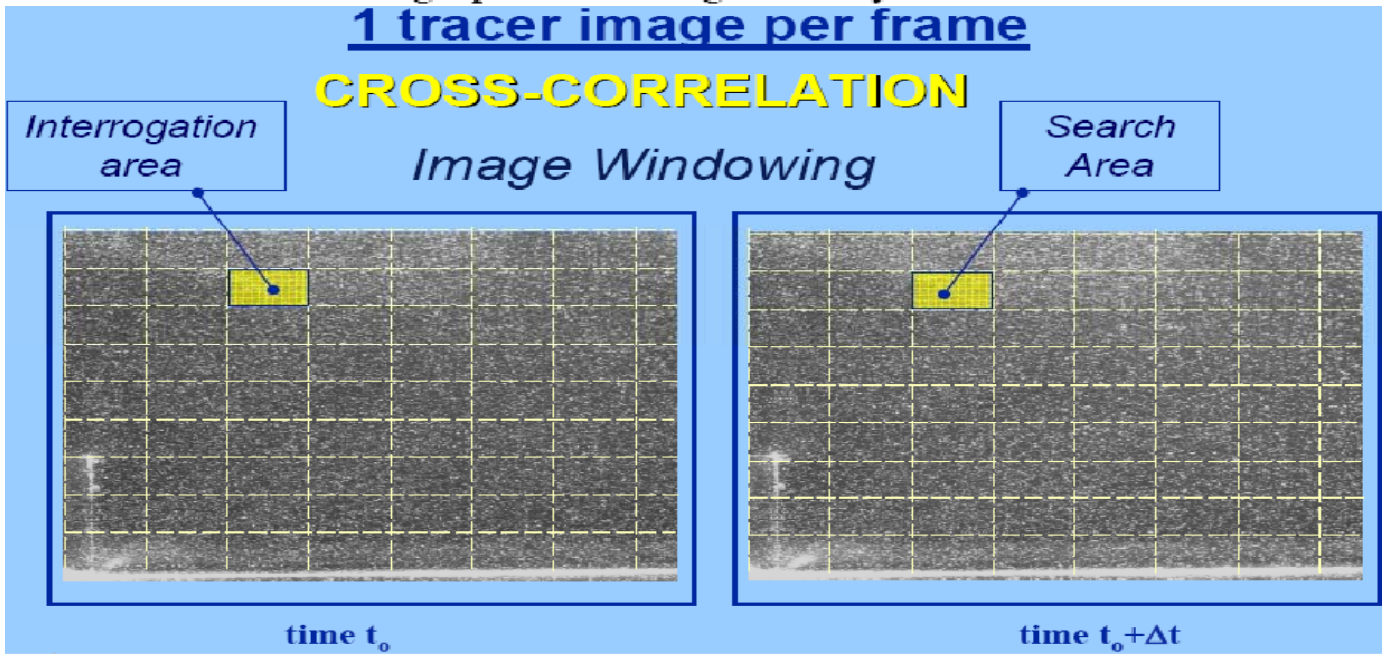


$t=t_0$ $t=t_0+\Delta t$

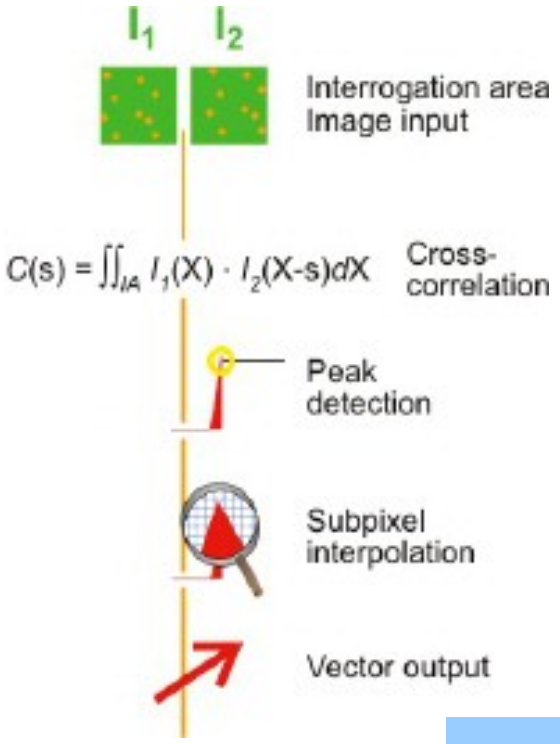
high particle-image density



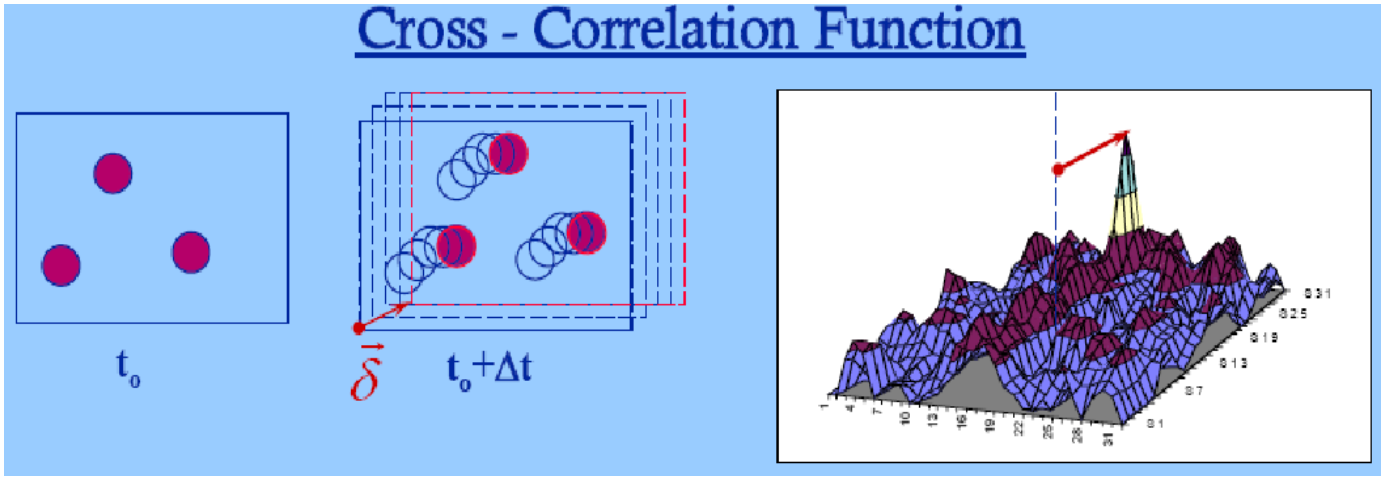
Corresponding flow velocity field



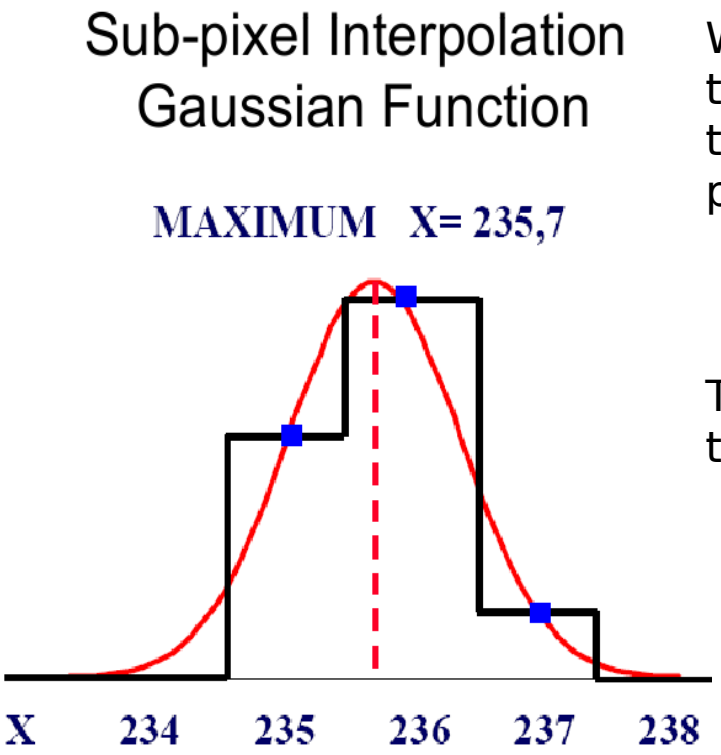
Particle Image Velocimetry-displacement



1) Each of the images (t and $t + \Delta t$) are divided into small subsections called interrogation areas (IA).
2) The interrogation areas from each image frame, I_1 and I_2, are cross-correlated with each other, pixel by pixel.
3) The correlation produces a signal peak, identifying the common particle displacement.
4) An accurate measure of the displacement - and thus also the velocity - is achieved with sub-pixel interpolation.



Particle Image Velocimetry – sub-pixel interpolation



When the maximum peak has been detected at [i, j], the neighbouring values are used to fit a function to the peak. In the case of a Gaussian peak fit, the peak is assumed to have the shape

$$f(x) = Ce^{[-(x_0-x)^2/k]}$$

The coordinates of the interpolated maximum are then found by:

$$x_0 = i + \frac{\ln R_{(i-1,j)} - \ln R_{(i+1,j)}}{2\ln R_{(i-1,j)} - 4\ln R_{(i,j)} + 2\ln R_{(i+1,j)}}$$

$$y_0 = i + \frac{\ln R_{(i,j-1)} - \ln R_{(i,j+1)}}{2\ln R_{(i,j-1)} - 4\ln R_{(i,j)} + 2\ln R_{(i,j+1)}}$$

Particle Image Velocimetry –example

- Interrogate area: $I(t) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$ $I(t + \Delta t) = \begin{bmatrix} 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$
- Cross-Correlation function:

  def cross_correlation$(a = I(t), b = (I + \Delta t))$:

  $a = a - average(a)$

  $b = b - average(b)$

  $c = correlate2d(a, b)$

  $return\ c$
- $c_{out} =$

```
[[ 0.16 -0.08 -0.32 -0.56 -0.4  -0.56 -0.32 -0.08  0.16]
 [ 0.32 -0.56 -0.84 -0.72  0.6  -0.72 -0.84 -0.56  0.32]
 [ 0.48 -1.04 -1.36  1.12  3.6   1.12 -1.36 -1.04  0.48]
 [ 0.64 -1.12 -0.68  2.56  7.2   2.56 -0.68 -1.12  0.64]
 [ 0.8  -0.8  -1.2   1.2   4.    1.2  -1.2  -0.8   0.8 ]
 [ 0.64 -0.72 -1.88 -2.24 -0.6  -2.24 -1.88 -0.72  0.64]
 [ 0.48 -0.24 -1.36 -2.08 -1.6  -2.08 -1.36 -0.24  0.48]
 [ 0.32  0.24  0.16  0.08  0.4   0.08  0.16  0.24  0.32]
 [ 0.16  0.32  0.48  0.64  0.8   0.64  0.48  0.32  0.16]]
```

Particle Image Velocimetry – example

$c_{out} =$

```
[[ 0.16 -0.08 -0.32 -0.56 -0.4  -0.56 -0.32 -0.08  0.16]
 [ 0.32 -0.56 -0.84 -0.72  0.6  -0.72 -0.84 -0.56  0.32]
 [ 0.48 -1.04 -1.36  1.12  3.6   1.12 -1.36 -1.04  0.48]
 [ 0.64 -1.12 -0.68  2.56  7.2   2.56 -0.68 -1.12  0.64]
 [ 0.8  -0.8  -1.2   1.2   4.    1.2  -1.2  -0.8   0.8 ]
 [ 0.64 -0.72 -1.88 -2.24 -0.6  -2.24 -1.88 -0.72  0.64]
 [ 0.48 -0.24 -1.36 -2.08 -1.6  -2.08 -1.36 -0.24  0.48]
 [ 0.32  0.24  0.16  0.08  0.4   0.08  0.16  0.24  0.32]
 [ 0.16  0.32  0.48  0.64  0.8   0.64  0.48  0.32  0.16]]
```

- Peak = 7.2 (3,4), Origin coordinates = (4,4) //[row, column], index start from 0.
- *def gaussian_sub_pixel_estimator():*

  f0 = log(c[x,y])

  f1 = log(c[x-1,y])

  f2 = log(c[x+1,y])

  *peakx = x + (f1-f2)/(2*f1-4*f0+2*f2)*

  f0 = log(c[x, y])

  f1 = log(c[x, y-1])

  f2 = log(c[x, y+1])

  *peaky = y + (f1-f2)/(2*f1-4*f0+2*f2)*

  return peakx,peaky

Particle Image Velocimetry – example

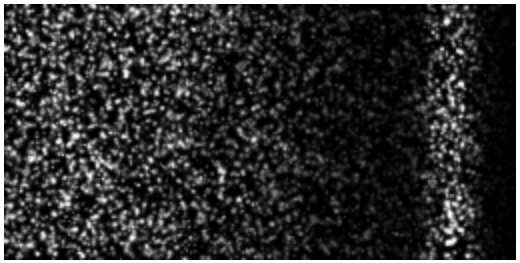
- peakx = 3.041, peaky = 4.0,
- origin coordinates (ocx,ocy)= (4,4),
- displacement(x_d, y_d):

$$x_d = \mathrm{ocy} - \mathrm{peaky} = 0.0$$

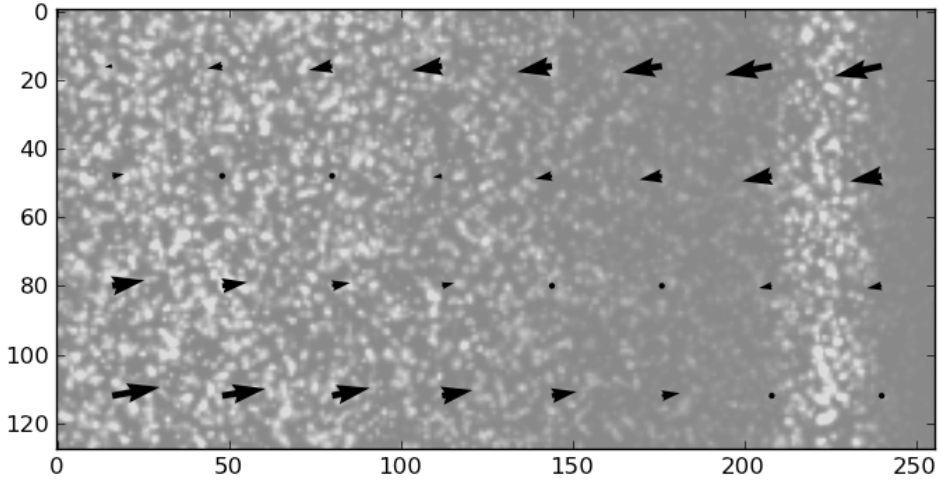
$$y_d = \mathrm{peakx} - \mathrm{ocx} = -0.959$$

Particle Image Velocimetry – example

- Python toolbox: http://sourceforge.net/projects/pypiv/ *[5]*
- Command line: $ python pyPIV.py images2 32 32 2 1 1 10 [0,0,0,0]
- Input data – shear sliding:



- Output data:



References

[1] Kilian, J., 2001: '*Simple Image Analysis By Moments v. 0.2*' (PDF)

[2] http://paulbourke.net/miscellaneous/correlate/

[3] http://en.wikipedia.org/wiki/Cross-correlation

[4] http://en.wikipedia.org/wiki/Particle_image_velocimetry

[5] http://sourceforge.net/projects/pypiv/

[6] Goldstein, H., 2000: '*Classical Mechanics*', 3rd ed.