Dept. of Math. University of Oslo
Research Report in Mechanics
ISSN 0801–9940 May 2016

Getting started with HydrolabPIV v1.1*

Jostein Kolaas[1]

[1]Department of Mathematics, University of Oslo

February 1, 2017

1 Short introduction to Particle image velocimetry (PIV)

There are many ways to measure fluid motion, one of which is particle image velocimetry (PIV). This is a non-intrusive optical measurement method, which gives velocity fields resolved in both time and space. PIV and other similar methods like particle tracking velocimetry (PTV) are based on three principles (Sveen and Cowen, 2004)

1. A visual representation of fluid motion.
2. An imaging system for capturing the visual representation.
3. A method for processing the images to quantify the fluid motion.

To visualize the flow, natural or added tracer particles are usually suspended in the fluid. While the use of tracer particles is the most common, it is also possible to use dye patterns (Raffel et al, 2007). If the particles are small and/or neutrally buoyant the particles are deemed passive and will follow the fluid motion. The imaging system usually consist of a light source, a camera and a lens. The light source is most commonly a pulsed solid state laser (ND:YAG/ND:YLF), but white light can also be used (Raffel et al, 2007). Using light sheet optics the laser beams is spread to illuminated the field of view, which is captured by a high speed digital camera synchronized with the laser, resulting in a sequence of frames I_0, I_1, I_2 and I_3, see figure 1.

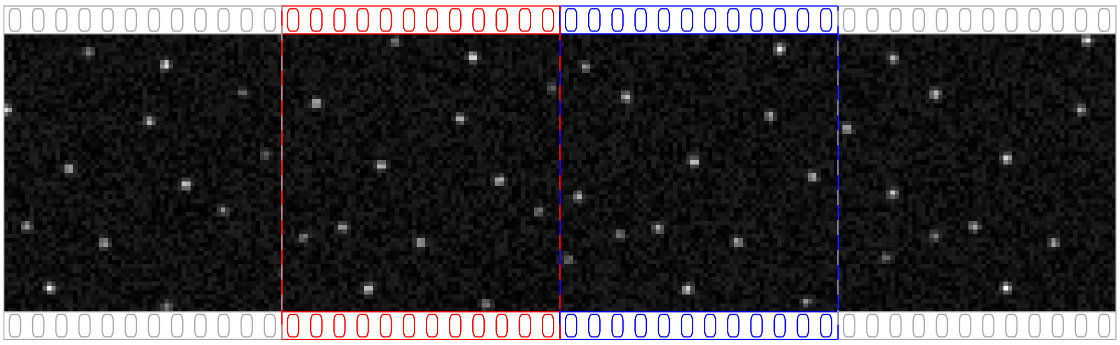


Figure 1: Raw image sequence

*This work was financed by the research project DOMT - Developments in Optical Measurement Technologies (project number 231491) funded by the Research Council of Norway.

Given two images we now want to find an Eulerian description of the velocity field. We start by dividing images into a regular grid of subwindows, usually with overlap. For each subwindow in the first frame, I_1 (see figure 2a), we try to match its pattern with a similar pattern in the second frame , I_2. Dividing the optimal match displacement of the pattern by Δt we find an ensemble averaged velocity. For this we need a

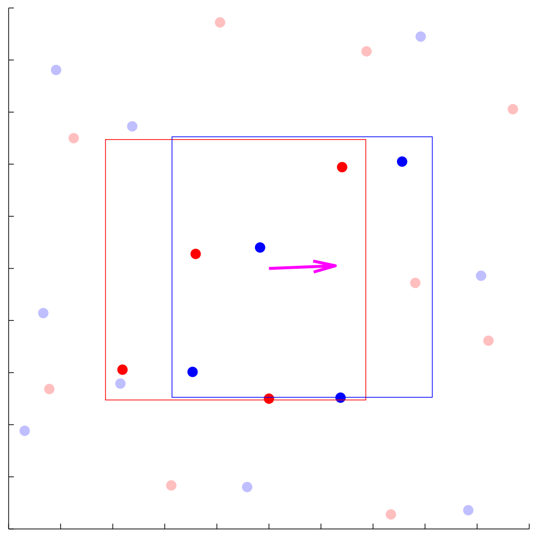


Figure 2: In PIV the image pattern created by the particles in frame I_1 and I_2 within the subwindow (rectangles) is matched to find an ensemble average velocity vector.

metric to tell how different the pattern is, a common choice is cross-correlation (Willert and Gharib, 1991)

$$R_{cc} = \sum I_1 I_2 \qquad\qquad R_{ncc} = \frac{\sum I_1 I_2 - \frac{\sum I_1 \sum I_2}{N}}{\sqrt{\left(\sum I_1{}^2 - \frac{(\sum I_1)^2}{N}\right)\left(\sum I_2{}^2 - \frac{(\sum I_2)^2}{N}\right)}}, \tag{1}$$

shown with (R_{ncc}) and without normalization (R_{cc}). To speed up processing, fast Fourier transform (FFT) is often used to calculate the cross-correlation. Padfield (2012) showed that the normalized version also can be done with FFT. Alternatively, if the image is double exposed, one can auto-correlate the image with itself, see Adrian (1991). This enables slower cameras to be used for measurement, but with reduced quality and there is an ambiguity in the flow direction so it can only be used for unidirectional flow.

To improve the resolution, PIV uses subpixel interpolation to estimate the peak of the difference measure. This can be done by fitting a quadratic or bi-quadratic polynomial to a 3x3 pixel block around the peak (Nobach and Honkanen, 2005; Dalziel, 2012). A downside to subpixel interpolation is that it causes peak-locking, an aliasing effect towards whole pixels (Christensen, 2004). It is important to test the experimental setup before measurements to keep this bias error under control.

After the velocity are found, the result can be made more robust by checking the quality of the image and difference measure, and further validation of the vectors using outlier detection (Westerweel, 1993; Westerweel and Scarano, 2005). In more advanced PIV algorithms, multipass methods including distorted passes (Huang et al, 1993a,b; Scarano, 2002), are often used to improve resolution and accuracy.

2 Setup of HydrolabPIV

To run HydrolabPIV it needs to be added to the Matlab path:

```
addpath(genpath('/mn/kadingir/domt_143903/HydrolabPIV/src'));
addpath('/mn/kadingir/domt_143903/HydrolabPIV/images');
javaaddpath('/mn/kadingir/domt_143903/HydrolabPIV/src/measures');
javaaddpath('/mn/kadingir/domt_143903/HydrolabPIV/src/interp');
```

If necessary replace the folder name with where you installed HydrolabPIV. If one often uses HydrolabPIV these lines can be added to startup.m in the MatLab folder in your home area, which runs every time MatLab starts. If this file does not exist you can create it.

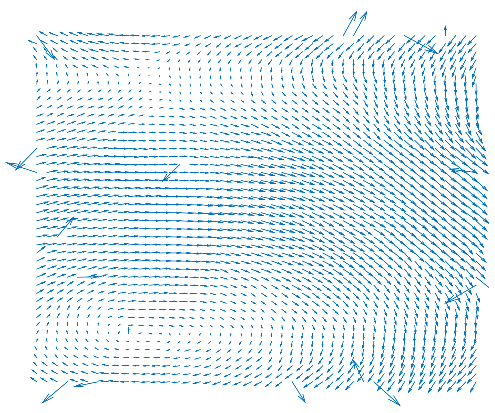
3 Single pass PIV

After we have read an image pair, we will setup PIV to use 24x24 pixel subwindows with 50% overlap. This will give us a velocity vector for every 12 pixels. The search range should normally be less than half the subwindow size, in this case we set it to go from -8 to 8 pixels (a third of the subwindow) in both x and y direction. Other options can also be set using the *setpivopt(...)* function. The single pass PIV is done with *normalpass(...)* with the images and options as arguments. (The empty argument will be explained later). The result is returned using a structure, which among other things contains the positions (x,y) and the velocities (U,V).

Example 1: Single pass PIV

```
im1 = imread('imA.png');
im2 = imread('imB.png');

opt = setpivopt('range',[-8 8 -8 8],'subwindow',24,24,.50);
piv = normalpass([],im1,[],im2,[],opt);

figure;
scale = 5;
quiver(piv.x,piv.y,scale*piv.U,scale*piv.V,0);
```

Note that when plotting the vector field it is useful, especially if comparing vector fields, to set the scale parameter in *quiver(...)* to zero (avoiding autoscaling).


(a) Example 1

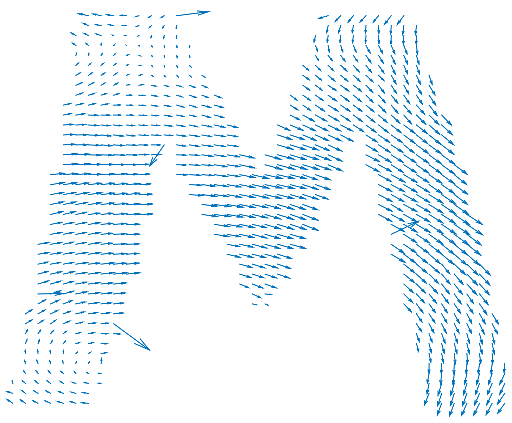

(b) Example 2

Figure 3: Single pass piv with and without masking.

In some cases it is useful to mask part of the image, for instance when the image contains a dynamic free surface. The mask is provided as arguments to *normalpass(. . .)* as another image pair, where 0 is masked and 1 is umasked. (It is also possible to give a number between 0 and 1 to give less weight to certain pixels). You can try this out by selecting a few points with *impoly()*, double-click for last point.

Example 2: Single pass PIV with Masking

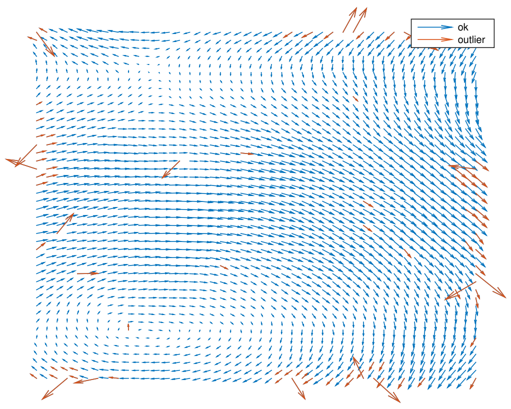
```
% create a polygonal mask
figure;
imagesc(im1);
set(gca,'Ydir','normal')
h = impoly();
mask = h.createMask();

opt = setpivopt('range',[-8 8 -8 8],'subwindow',24,24,.50);
piv = normalpass([],im1,mask,im2,mask,opt);

figure;
scale = 5;
idm = interp2(double(mask),piv.x,piv.y)==1;
quiver(piv.x(idm),piv.y(idm),scale*piv.U(idm),scale*piv.V(idm),0);
```

An important step of the PIV measurement process is to check the quality. One way is to inspect the signal-to-noise ratio *piv.snr*, where the noise level is estimated as the median absolute deviation of the correlation/diffrence measure (taken over the whole measure or within the search range) and the signal strength is the difference between the noise and the peak value. If the signal-to-noise ratio has a blocky apperance, it would indicate that the particle concentration is to low. To compansate for this it may be possible to use larger subwindows.

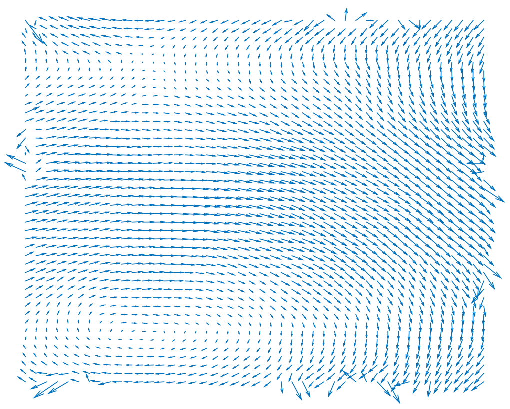
Another step in improving the quality of the PIV result is to detect and replace outliers. For this HydrolabPIV uses 3x3 normalized local median filter (Westerweel and Scarano, 2005), the residuals from the filter is stored in *piv.localres*. If part of the 3x3 is masked, a masked 5x5 local median filter is used instead. In addition a cubic B-spline is fitted to the velocity field using an iterative weighted least squares fit. The local residuals are used with the biweight function in the first iteration. The fit is then iterated a few times where the residuals normalized by 9x9 local median of the residuals, is used to update the weights in the least squares fit. The normalized residuals form the last iteration are stored in *piv.globalres*, which together with the residuals from the local median filter are used to mark the outliers in *piv.out*. These outlier can be replaced with *replaceoutliers(. . .)*, where the missing vectors are evaluated using the fitted B-spline.

Example 3: Outlier detection and replacement

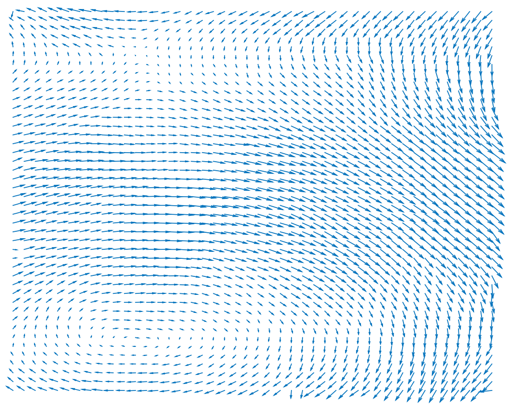
```
im1 = imread('imA.png');
im2 = imread('imB.png');

opt = setpivopt('range',[-8 8 -8 8],'subwindow',24,24,.50);
piv = normalpass([],im1,[],im2,[],opt);
[U,V,x,y] = replaceoutliers(piv);

figure;
scale = 5;
quiver(piv.x,piv.y,scale*piv.U,scale*piv.V,0)
hold on;
idx = not(piv.out);
quiver(piv.x(idx),piv.y(idx),scale*piv.U(idx),scale*piv.V(idx),0)
hold off;
legend(' ok ',' outlier ');
title('Outlier detection');

figure;
quiver(x,y,scale*U,scale*V,0)
```



(a) Outlier detection



(b) After replacement of outliers (50% overlap)



(c) After replacement of outliers (75% overlap)

Figure 4: Outlier detection and replacement (Example 3)

```
title('Outlier replacement (50% overlap)');

opt2 = setpivopt('range',[-8 8 -8 8],'subwindow',24,24,.75);
piv2 = normalpass([],im1,[],im2,[],opt2);
[U2,V2,x2,y2] = replaceoutliers(piv2);

figure;
idx = 1:2:size(U2,2);
quiver(x2(idx,idx),y2(idx,idx),scale*U2(idx,idx),scale*V2(idx,idx),0)
title('Outlier replacement (75% overlap)');
```

Note that there are more outliers on the edge, since approximately 50% of the subwindow is masked. While for single pass piv vectors on the edge is often ignored, it is important to have an estimate of velocity on the edge to get high quality displacement when using distorted passes. Since the amount of least squares fit in the B-spline is set to coincide with the subwindow overlap, a 0% overlap would result in interpolation of the velocity field and no outlier would be replaced. Increasing the overlap from 50% to 75% gives 4 times as many vectors resulting in higher computational cost, but will in general improve the outlier detection and replacement as is shown in figure 4.

4 Multipass PIV

Further improvements in accuracy and resolution can be obtained by using multiple PIV passes, using for instance window shifting or window distortion.

4.1 Window shifting

For a given difference measure the search range is usually maximum half of the subwindow size, limiting the maximum displacement. Larger subwindows enables larger maximal displacement/velocity, but at reduced resolution. A way to improve the resolution while retaining the maximum displacement is to first use a pass with large subwindows, followed by a pass with smaller subwindows shifted by the displacement found in the previous pass as shown in the next example.

Example 4: Shallow water wave

```
% Read in wave images
im1 = imread('wave1.png');
mask1 = imread('wave1mask.png');
im2 = imread('wave2.png');
mask2 = imread('wave2mask.png');

% First pass with masks
opt1 = setpivopt('range',[-8 8 -8 8],'subwindow',64,64,.75);
piv1 = normalpass([],im1,mask1,im2,mask2,opt1);
[U1,V1] = replaceoutliers(piv1);

%Shifted pass using smaller subwindows
opt2 = setpivopt('range',[-4 4 -4 4],'subwindow',32,32,.75);
piv2 = normalpass(piv1,im1,mask1,im2,mask2,opt2);
[U2,V2] = replaceoutliers(piv2);
```

4.2 Window distortion

Subwindow based methods like PIV with correlation/difference measure only gives translation, assuming the velocity field within the subwindow is uniform. To improve accuracy for shear flows, subwindow distortion (Scarano, 2002) can be used. HydrolabPIV finds the distortion of the image pair by integrating the B-splines fitted to the displacements using Runge-Kutta. The displacement is then used to interpolate a new image pair tending toward zero/uniform displacement. Processing this image pair in the regular way gives a correction to the displacements.

Example 5: Shallow water wave (continued)

```
% Distorted passes
opt3 = setpivopt('range',[-4 4 -4 4],'subwindow',32,32,.75);
piv3 = distortedpass(piv2,im1,mask1,im2,mask2,opt3);
[U3,V3] = replaceoutliers(piv3);

opt4 = setpivopt('range',[-4 4 -4 4],'subwindow',32,32,.75);
piv4 = distortedpass(piv3,im1,mask1,im2,mask2,opt4);
[U4,V4,x,y] = replaceoutliers(piv4);
```

To get the best accuracy 3-4 distorted passes are recommended. The default option for estimating distortion uses the standard RK4, but any other explicit schemes can be used by specifing the Butcher tableau. It is also possible to only partially undistort the image pair by setting $\alpha < 1$.

```
% partial distort images (half-way) using Heun's method
alpha = 0.5;
opt = setpivopt('tableau',[0 0 0; 1 1 0; 0 1/2 1/2],alpha)
```

An important choice that greatly effects the error is interpolation of the image pair.

```
opt = setpivopt('iminterp',@iminterp2matlab,'linear') % Linear interpolation using matlab
opt = setpivopt('iminterp',@iminterp2matlab,'spline') % Bspline using matlab
opt = setpivopt('iminterp',@iminterp2bsplinej,[])     % Bspline using Java
% Lanczos kernel with 2n-1 lobes using Java
n = 3;
opt = setpivopt('iminterp',@iminterp2lanczosj,n)
```

Lanczos resampling works better for smaller particles d_p = 2-4 pixels while cubic b-spline works better for particls $d_p > 5$ pixels. The two Java methods allows for setting the number of computational threads N using *Bspline2(N)* or *Lanczos2(N)*.

4.3 Init pass

If the experiment is time resolved it is possible to save computation time by using previous found velocity fields as an initial guess to the next image pair using *initpass(...)*.

Example 6: Shallow water wave

```
% Read in more wave images
im3 = imread('wave3.png');
mask3 = imread('wave3mask.png');

% Use result from previous image pair as initial guess
piv1 = initpass(piv2.Uspline,piv2.Vspline);

%Shifted pass using smaller subwindows
opt2 = setpivopt('range',[-4 4 -4 4],'subwindow',32,32,.75);
piv2 = normalpass(piv1,im2,mask2,im3,mask3,opt2);
[U2,V2] = replaceoutliers(piv2);
```

5 Coordinate transformations

To get the physical quanteties we need convert the from the camera coordinate system x in pixel to the real world coordinates $\tilde{x}$. For this we need a coordinate transformation, the most basic being linear transformation

$$\tilde{\boldsymbol{x}} = \begin{bmatrix}\tilde{x}\\ \tilde{y}\\ 1\end{bmatrix} = \begin{bmatrix}t_{11} & t_{12} & x_0\\ t_{21} & t_{22} & y_0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ 1\end{bmatrix} = T\boldsymbol{x}, \tag{2}$$

which include translation, scaling and rotation. While this transfrom is used for the positions, it becomes necessary to calcuate the Jabobian J to transform the velocity vectors

$$\tilde{\boldsymbol{u}} = \begin{bmatrix}\tilde{u}\\ \tilde{v}\end{bmatrix} = \begin{bmatrix}\frac{\partial\tilde{x}}{\partial x} & \frac{\partial\tilde{x}}{\partial y}\\ \frac{\partial\tilde{y}}{\partial x} & \frac{\partial\tilde{y}}{\partial y}\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix} = J\boldsymbol{u} \tag{3}$$

In the linear case this is rather simple and the Jacobian becomes

$$J = \begin{bmatrix}t_{11} & t_{12}\\ t_{21} & t_{22}\end{bmatrix}. \tag{4}$$

Given a set of control points in pixel and world coordinates

$$X' = \begin{bmatrix}x_1 & y_1 & 1\\ \vdots & \vdots & \vdots\\ x_n & y_n & 1\end{bmatrix} \qquad \tilde{X}' = \begin{bmatrix}\tilde{x}_1 & \tilde{y}_1 & 1\\ \vdots & \vdots & \vdots\\ \tilde{x}_n & \tilde{y}_n & 1\end{bmatrix} \tag{5}$$

inserting into equation 2

$$\tilde{X} = TX, \tag{6}$$

which can be solved with respect to transformation T

$$T = \Big[(X')^{-1}(\tilde{X}')\Big]'. \tag{7}$$

If there are more controlpoints than is needed, we can solve for T in a least squares fashion

$$T = \Big[(XX')^{-1}(X\tilde{X}')\Big]'. \tag{8}$$

Sometimes a linear transform is not enough to describe the mapping, for instance if camera is slight tilted up- or downward we get projective transfrom. This can be approximated by Cubic-transformation

$$\tilde{\boldsymbol{x}} = \begin{bmatrix}\tilde{x}\\ \tilde{y}\\ 1\end{bmatrix} = \begin{bmatrix}t_{11} & t_{12} & t_{13} & t_{14} & t_{15} & t_{16} & t_{17} & t_{18} & t_{19} & x_0\\ t_{21} & t_{22} & t_{23} & t_{24} & t_{25} & t_{26} & t_{27} & t_{28} & t_{29} & y_0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x^3\\ x^2y\\ xy^2\\ y^3\\ x^2\\ xy\\ y^2\\ x\\ y\\ 1\end{bmatrix} = T\boldsymbol{x}. \tag{9}$$

Using the control point matrix

$$X' = \begin{bmatrix} x_1^3 & x_1^2 y_1 & x_1 y_1^2 & y_1^3 & x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^3 & x_n^2 y_n & x_n y_n^2 & y_n^3 & x_n^2 & x_n y_n & y_n^2 & x_n & y_n & 1 \end{bmatrix}, \tag{10}$$

we can solve for T in a simular fashion as for the linear case. Note that since this transform is non-linear, (T is not invertible). Also the Jabobian for this transform is slightly more complicated as it depends on the position.

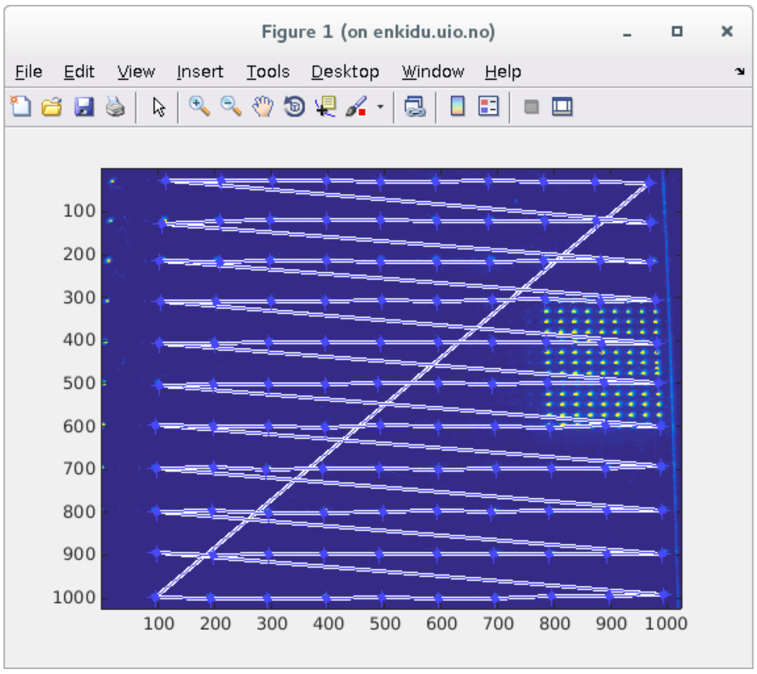


Figure 1 (on enkidu.uio.no)
File Edit View Insert Tools Desktop Window Help
100
200
300
400
500
600
700
800
900
1000
100 200 300 400 500 600 700 800 900 1000

Figure 5: Selecting pixel reference points (Example 7)

In the Shallow water wave example the camera is slighty tilted upwards to get a better measurement of the surface elevation, so we will use a cubic coordinate transformation. If we use *ndgrid(. . .)* to define a grid of world control points, it is convenient to select the pixel control point from right to left, and then upwards, see figure 5. The last control point is selected with a double-click.

Example 7: Creating a coordinate system

```
coord = imread('wavecoord.png');
imagesc(coord)

% Select reference points in pixel coordinate
h=impoly;
pixel = h.getPosition;

% Refine pixel positions (optional)
c = graythresh(coord);
bw = im2bw(coord);
cc = bwconncomp(bw);
stats = regionprops(cc,'Centroid');
xc = vertcat(stats.Centroid);
idx = knnsearch(xc,pixel);
```

```
pixel = xc(idx,:);

% Define matching reference points in world coordinate
[wx,wy] = ndgrid((9:-1:0)*0.02 + 4.33,(-10:0)*0.02 + 0.103);
world = [wx(:) wy(:)];

% Create coordinate transformation
[tform,err,errinv] = createcoordsystem(pixel,world,'cubic')
```

Using this coordinate transformation we can look at the shallow water stokes wave using real world units as seen in figure 6.

Example 8: Using the coordinate transform

```
dt = 1/375;
[U4,V4,x,y] = replaceoutliers(piv4,mask1&mask2);
[Uw,Vw,xw,yw] = pixel2world(tform,U4,V4,x,y,dt);

% Surface
[idx1,eta1] = max(mask1);
[idx2,eta2] = max(mask2);
[etax,etay] = tformfwd(tform,1:1024,(eta1+eta2)/2);

figure;
idx = 1:4:129;
scale = 1/50;
quiver(xw(idx,idx),yw(idx,idx),scale*Uw(idx,idx),scale*Vw(idx,idx),0);
hold on;
plot(etax,etay,'r');
hold off;
xlabel(' x [ m ] ');
ylabel(' y [ m ] ');
```

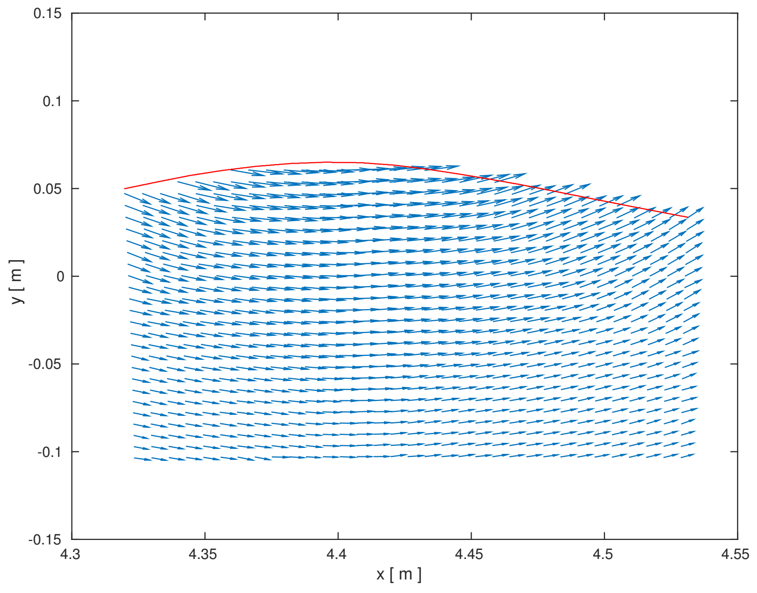
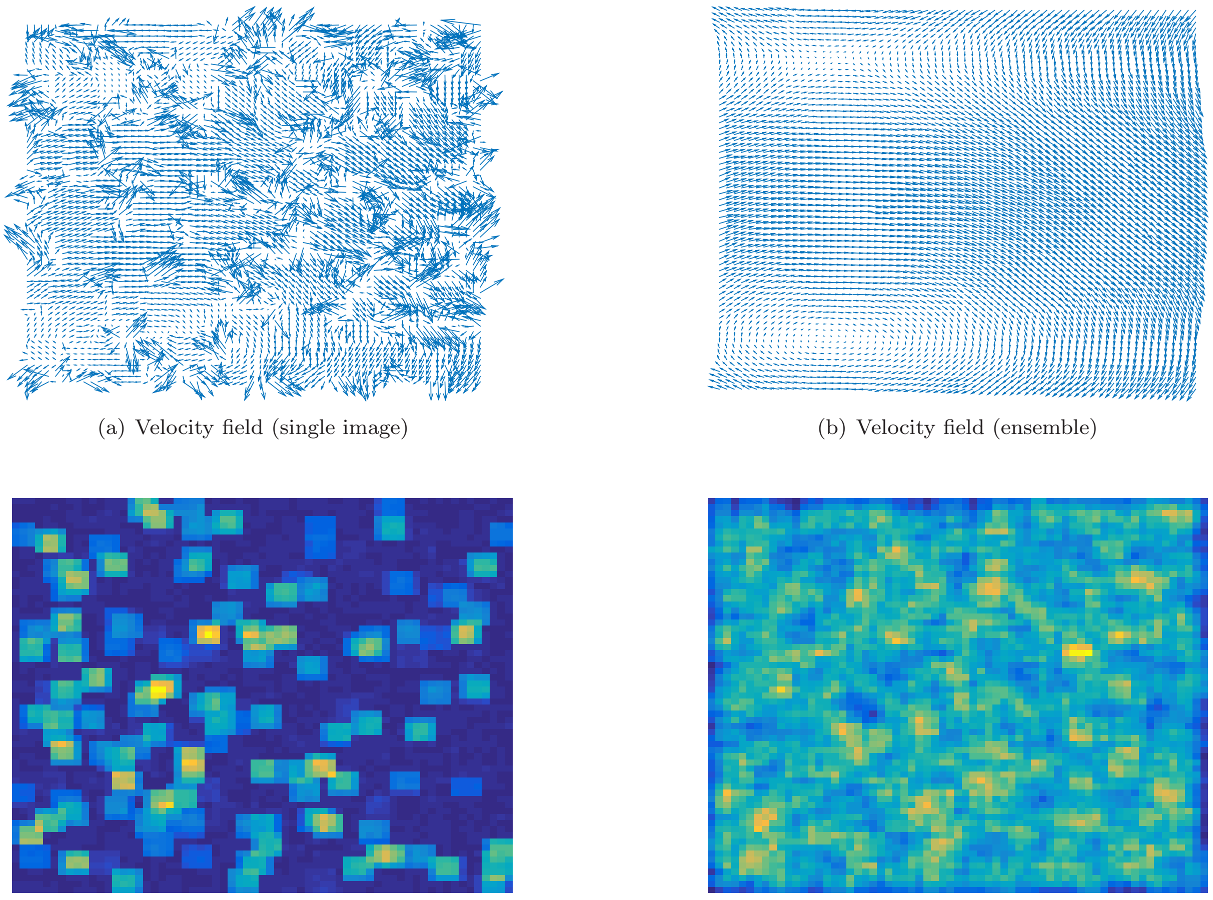


Figure 6: Top of shallow water Stokes wave from Example 8

6 Ensemble PIV

In MicroPIV volume illumination is often used, which results in low seeding concentration due to visibility problem (Olsen and Adrian, 2000). PIV on these type of images has a poor signal-to-noise ratio, and the velocity from PIV becomes noisy, see figure 6a,c. To overcome this Wereley and Meinhart (2005) suggest taking an ensemble average of the correlation peaks in time taken over many images, improving the signal-to-noise ratio and providing an average velocity field, see figure 6b,d. HydrolabPIV has two ways to perform


(a) Velocity field (single image) (b) Velocity field (ensemble)

(c) SNR (single image) (d) SNR (ensemble)

Figure 7: Ensemble PIV

ensemble PIV. The first way is to provide the images series as a 3D matrix.

Example 9: Ensemble PIV

```
% Load images
I=512; J=512; K=25;
im1 = zeros(I,J,K);
im2 = zeros(I,J,K);
for k=1:K
  im1(:,:,k) = imread(sprintf('sparse%02dA.png',k));
  im2(:,:,k) = imread(sprintf('sparse%02dB.png',k));
end

opt = setpivopt('ensemble',@nanmedian);
```

```
piv = normalpass([],im1,[],im2,[],opt);
[U,V,x,y] = replaceoutliers(piv);

figure;
quiver(x,y,U,V);
```

This has the advantage that it can be made more robust by setting median as the ensemble function. Also note that color images will be seen as an ensemble average over the color channels. The second way is to read in the image pairs sequentially with multiple calls to normalspass. This requires less memory, but is limited to using mean as ensemble function.

Example 10: Ensemble PIV (alternative)

```
opt1 = setpivopt('savepeaks',true);
for k=1:25
  im1 = imread(sprintf('sparse%02dA.png',k));
  im2 = imread(sprintf('sparse%02dB.png',k));

  if(k==1)
    piv1 = normalpass([],im1,[],im2,[],opt1);
  else
    piv1 = normalpass([],im1,[],im2,[],piv1);
  end
end
[U1,V1,x1,y1] = replaceoutliers(piv1);

figure;
quiver(x1,y1,U1,V1);
```

To make a compromise between memory and robustness, both of the methods can be combined. Increasing the accuracy is also possible using multipass ensemble PIV, including distorted passes.

Example 11: Multipass ensemble PIV

```
opt2 = setpivopt('savepeaks',true,'savedisplacements',true);
for k=1:25
  im1 = imread(sprintf('sparse%02dA.png',k));
  im2 = imread(sprintf('sparse%02dB.png',k));

  if(k==1)
    piv2 = distortedpass(piv1,im1,[],im2,[],opt2);
  else
    piv2 = distortedpass(piv1,im1,[],im2,[],piv2);
  end
end
[U2,V2,x2,y2] = replaceoutliers(piv2);
```

7 Measure functions

To match the pattern in a subwindow pair a measure function is needed, in PIV this is most often cross-correlation, but other options like minimum quadratic difference are also available. The measure function can be set in HydrolabPIV using

```
opt = setpivopt(..., 'measure', @mfun, mparam, @prefun, ...)
```

where $mfun$ can be one of several measures included with HydrolabPIV or a custom function:

```
[dm,mm] = mfun(f1,m1,f2,m2,idh,idw,mparam)
```

Given two subwindows with masks f_1/m_1 and f_2/m_2 with sizes (i,j) and (k,l) respectively, $@mfun$ should return a measure, dm, with zero displacement peak located in the center and the masked fraction, mm, both of size $(i+k-1)\times(j+l-1)$. The indices (idh, idw) show which part of the difference measure is needed. $mparam$ provides an additional argument to the $@mfun$ function, see specific $mfun$ function for it usage. The function $@prefun$ is applied to the measure before subpixel interpolation. For cross-correlation this can be for instance $@log$, practically assuming the measure peak is Gaussian when using quadratic fit in subpixel interpolation

7.1 Cross-correlation

The most common measure used in PIV is cross-correlation.

$$cc(r,s) = \frac{1}{IJ}\sum_{i,j} f_{1(i+r,j+s)} \cdot f_{2(i,j)},$$

often calculated with fast Fourier transform

$$cc = \frac{1}{IJ}\mathcal{F}^{-1}\left(\mathcal{F}(f_1)\cdot\mathcal{F}(f_2^*)\right).$$

To improves the bias error for partially masked subwindows (including the edges of the image), HydrolabPIV uses masked/weighted cross-correlation

$$mcc(r,s) = \frac{\sum_{i,j}\left(m_{1(i+r,j+s)}\cdot f_{1(i+r,j+s)}\right)\cdot\left(m_{2(i,j)}\cdot f_{2(i,j)}\right)}{\sum_{i,j} m_{1(i+r,j+s)}\cdot m_{2(i,j)}},$$

which also can be calculate with fast Fourier transform

$$mcc = \frac{\mathcal{F}^{-1}\left(\mathcal{F}(m_1\cdot f_1)\cdot\mathcal{F}(m_2^*\cdot f_2^*)\right)}{\mathcal{F}^{-1}\left(\mathcal{F}(m_1)\cdot\mathcal{F}(m_2^*)\right)}.$$

To achieve higher accuracy it is possible to use normalized cross-correlation

$$ncc(r,s) = \frac{num(r,s)}{\sqrt{den_1(r,s)\cdot den_2}},$$

where

$$num(r,s) = \sum_{i,j} f_{1(i+r,j+s)}\cdot f_{2(i,j)} - \frac{\sum_{i,j} f_{1(i+r,j+s)}\sum_{i,j} f_{2(i,j)}}{IJ},$$

$$den_1(r,s) = \sum_{i,j} f_{1(i+r,j+s)}^2 - \frac{\left(\sum_{i,j} f_{1(i+r,j+s)}\right)^2}{IJ},$$

$$den_2 = \sum_{i,j} f_{2(i,j)}^2 - \frac{\left(\sum_{i,j} f_{2(i,j)}\right)^2}{IJ}.$$

Note that den_2 is constant and is sometimes left out of the calculation as it is only a scaling factor. In a similar way the normalized cross-correlation can be generalized to a masked/weighted normalized cross-correlation

$$mncc(r,s) = \frac{num(r,s)}{\sqrt{den_1(r,s)\cdot den_2(r,s)}}$$

where

$$
\begin{aligned}
num(r,s) = &\sum_{i,j}\left(m_{1(i+r,j+s)}\cdot f_{1(i+r,j+s)}\right)\cdot\left(m_{2(i,j)}\cdot f_{2(i,j)}\right)\\
&-\frac{\sum_{i,j} m_{2(i,j)}\cdot\left(m_{1(i+r,j+s)}\cdot f_{1(i+r,j+s)}\right)\sum_{i,j} m_{1(i+r,j+s)}\cdot\left(m_{2(i,j)}\cdot f_{2(i,j)}\right)}{\sum_{i,j} m_{1(i+r,j+s)}\cdot m_{2(i,j)}}\\
den_1(r,s) = &\sum_{i,j} m_{2(i,j)}\cdot\left(m_{1(i+r,j+s)}\cdot f_{1(i+r,j+s)}^{2}\right)-\frac{\left(\sum_{i,j} m_{2(i,j)}\cdot\left(m_{1(i+r,j+s)}\cdot f_{1(i+r,j+s)}\right)\right)^2}{\sum_{i,j} m_{1(i+r,j+s)}\cdot m_{2(i,j)}}\\
den_2(r,s) = &\sum_{i,j} m_{1(i+r,j+s)}\cdot\left(m_{2(i,j)}\cdot f_{2(i,j)}^{2}\right)-\frac{\left(\sum_{i,j} m_{1(i+r,j+s)}\cdot\left(m_{2(i,j)}\cdot f_{2(i,j)}\right)\right)^2}{\sum_{i,j} m_{1(i+r,j+s)}\cdot m_{2(i,j)}}
\end{aligned}
$$

Note that when a mask is present den_2 is not constant with respect to r,s anymore and should be included. Using this form also enables calculating the masked/weighted normalized cross-correlation with fast Fourier transform (Padfield, 2012)

$$mncc = \frac{num}{\sqrt{den_1\cdot den_2}},$$

where

$$
\begin{aligned}
num &= \mathcal{F}^{-1}(F_1\cdot F_2)-\frac{\mathcal{F}^{-1}(M_1\cdot F_2)\mathcal{F}^{-1}(M_2\cdot F_1)}{\mathcal{F}^{-1}(M_1\cdot M_2)},\\
den_1 &= \mathcal{F}^{-1}(M_2\cdot\mathcal{F}(m_1\cdot(f_1)^2))-\frac{\left(\mathcal{F}^{-1}(M_2\cdot F_1)\right)^2}{\mathcal{F}^{-1}(M_1\cdot M_2)},\\
den_2 &= \mathcal{F}^{-1}(M_1\cdot\mathcal{F}(m_2\cdot(f_2^*)^2))-\frac{\left(\mathcal{F}^{-1}(M_1\cdot F_2)\right)^2}{\mathcal{F}^{-1}(M_1\cdot M_2)},
\end{aligned}
$$

$F_1 = \mathcal{F}(m_1\cdot f_1)$, $M_1 = \mathcal{F}(m_1)$, $F_2 = \mathcal{F}(m_2^*\cdot f_2^*)$ and $M_2 = \mathcal{F}(m_2^*)$. When all the masks values are one, this can be used to calculate the regular normalized cross-correlation as well. The fast Fourier transform based cross-corrections methods can be used by setting

```
opt = setpivopt('measure',@regularcc,true,@log);
opt = setpivopt('measure',@maskedcc,true,@log);
opt = setpivopt('measure',@maskedncc,true,@(x) x);
```

The subwindows are padded before FFT evaluation, but can be turn off by using mparam (increased speed, less accuracy). It is possible to apply the log function to the difference measure before the subpixel interpolation, but this is not recommended for normalized cross-correlation, see subsection 7.3 for more information. Under some condition, calculation direct sums is faster, which can be done by setting

```
opt = setpivopt('measure',@maskednccj,[],@(x) x);
opt = setpivopt('measure',@maskedccj,[],@(x) x);
```

Mparam is not used in these cases.

7.2 Minkowski differences

Other measures have been used for PIV, for instance Gui et al (1998) used Minimum quadratic difference, which is a special case of the Minkowski differences

$$md_{(r,s;\alpha)} = \left(\frac{1}{IJ} \sum_{i,j} \left| f_{1\,(i+r,j+s)} - f_{2\,(i,j)} \right|^{\alpha} \right)^{1/\alpha} .$$

These also exist in a masked/weighted version

$$mmd_{(r,s;\alpha)} = \left[\frac{\sum_{i,j} w \left| f_{1\,(i+r,j+s)} - f_{2\,(i,j)} \right|^{\alpha}}{\sum_{i,j} w} \right]^{1/\alpha} ,$$

where $w = m_{1\,(i+r,j+s)} \cdot m_{2\,(i,j)}$, and a normalized version

$$mnmd_{(r,s;\alpha)} = \left[\frac{\sum_{i,j} w \left| f_{1\,(i+r,j+s)} - f_{2\,(i,j)} \right|^{\alpha}}{\sqrt{\left(\sum_{i,j} w \left| f_{1\,(i+r,j+s)} \right|^{\alpha} \right) \cdot \left(\sum_{i,j} w \left| f_{2\,(i,j)} \right|^{\alpha} \right)}} \right]^{1/\alpha} ,$$

introduced in DigiFlow by Dalziel (2012). The masked/weighted or normalized Minkowski differences can be used by setting:

```
opt = setpivopt('measure',@maskedmdj,alpha,@(x) -x);
opt = setpivopt('measure',@maskednmdj,alpha,@(x) -x);
```

With the exception of some of the special cases this is a rather slow measure due to the usage of power function. Note the use of $-x$ to find the minimum value rather than a maximum peak value as for cross-correlation.

7.2.1 Geometric difference ($\alpha \to 0$)

The first special case is when $\alpha \to 0$

$$md_{(r,s;\alpha\to 0)} = \left[\prod_{i,j} \left| f_{1\,(i+r,j+s)} - f_{2\,(i,j)} \right| \right]^{1/IJ}$$

and is mostly provided for academic interest. Nevertheless it gives some insight into what happens for smaller values of α. Due to the product nature of the measure, it is sensitive to broadbanded noise in the image. It is recommended convert integer images to floating point and add small amount of noise, since the measure fails if $f_{1\,(i+r,j+s)} = f_{2\,(i,j)}$ for any i and j. The masked/weighted of the geometric difference measure is

$$mmd_{(r,s;\alpha\to 0)} = \left[\prod_{i,j} \left| f_{1\,(i+r,j+s)} - f_{2\,(i,j)} \right|^{w} \right]^{1/\sum_{i,j} w}$$

and its normalized version

$$mnmd_{(r,s;\alpha\to 0)} = \left[\frac{\prod_{i,j} \left| f_{1\,(i+r,j+s)} - f_{2\,(i,j)} \right|^{w}}{\sqrt{\left(\prod_{i,j} \left| f_{1\,(i+r,j+s)} \right|^{w} \right) \cdot \left(\prod_{i,j} \left| f_{2\,(i,j)} \right|^{w} \right)}} \right]^{1/\sum_{i,j} w} .$$

The Normalized also fails if $f_{1(i+r,j+s)} = 0$ or $f_{2(i,j)} = 0$ for any i and j, but this is easily fixed by adding some noise or a small constant. Note that the geometric difference measure can also be evaluated as a sum rather than a product using the log function.

7.2.2 Minimum absolute difference ($\alpha = 1$)

Minimum absolute difference used for PIV by Gui and Merzkirch (1997) and is the only Minkowski difference measure with some claim to robustness, which might be useful when out-of-plane loss of particles is present.

$$md_{(r,s;\alpha=1)} = \frac{1}{IJ}\sum_{i,j}\left|f_{1(i+r,j+s)} - f_{2(i,j)}\right|$$

It also has the least number of operation in its direct form, but practically the speed difference is neglectable. The masked/weighted version is

$$mmd_{(r,s;\alpha=1)} = \frac{\sum_{i,j} w\left|f_{1(i+r,j+s)} - f_{2(i,j)}\right|}{\sum_{i,j} w},$$

and its normalized form is

$$mnmd_{(r,s;\alpha=1)} = \frac{\sum_{i,j} w\left|f_{1(i+r,j+s)} - f_{2(i,j)}\right|}{\sqrt{\left(\sum_{i,j} w\left|f_{1(i+r,j+s)}\right|\right)\cdot\left(\sum_{i,j} w\left|f_{2(i,j)}\right|\right)}}.$$

7.2.3 Minimum quadratic difference ($\alpha = 2$)

Minimum quadratic difference

$$md_{(r,s;\alpha=2)} = \sqrt{\frac{1}{IJ}\sum_{i,j}\left(f_{1(i+r,j+s)} - f_{2(i,j)}\right)^2}$$

was used with PIV by Gui et al (1998). Its masked/weighted version is

$$mmd_{(r,s;\alpha=2)} = \sqrt{\frac{\sum_{i,j} w\left(f_{1(i+r,j+s)} - f_{2(i,j)}\right)^2}{\sum_{i,j} w}}.$$

This can be rewritten as

$$mmd_{(r,s;\alpha=2)} = \sqrt{\frac{\sum_{i,j} w\cdot f_{1(i+r,j+s)}^2 - 2w\cdot f_{1(i+r,j+s)}\cdot f_{2(i,j)} + w\cdot f_{2(i,j)}^2}{\sum_{i,j} w}}$$

which can be implemented with FFT using the same methods as for normalized cross-corrections,

$$mmd_{(\alpha=2)} = \sqrt{\frac{\mathcal{F}^{-1}(\mathcal{F}(m_1\cdot(f_1)^2)\cdot M_2) - 2\mathcal{F}^{-1}(F_1\cdot F_2) + \mathcal{F}^{-1}(M_1\cdot\mathcal{F}(m_2^*\cdot(f_2^*)^2))}{\mathcal{F}^{-1}\left(M_1\cdot M_2\right)}}$$

$F_1 = \mathcal{F}(m_1\cdot f_1)$, $M_1 = \mathcal{F}(m_1)$, $F_2 = \mathcal{F}(m_2^*\cdot f_2^*)$ and $M_2 = \mathcal{F}(m_2^*)$. Note that the middle term is essentially is the cross-correlation. The Normalized minimum quadratic difference is

$$mnmd_{(r,s;\alpha=2)} = \sqrt{\frac{\sum_{i,j} w\left(f_{1(i+r,j+s)} - f_{2(i,j)}\right)^2}{\sqrt{\left(\sum_{i,j} w\cdot f_{1(i+r,j+s)}^2\right)\cdot\left(\sum_{i,j} w\cdot f_{2(i,j)}^2\right)}}},$$

which can also implemented using FFT

$$mnmd_{(\alpha=2)} = \sqrt{\frac{\mathcal{F}^{-1}(\mathcal{F}(m_1 \cdot (f_1)^2) \cdot M_2) - 2\mathcal{F}^{-1}(F_1 \cdot F_2) + \mathcal{F}^{-1}(M_1 \cdot \mathcal{F}(m_2^* \cdot (f_2^*)^2))}{\sqrt{\mathcal{F}^{-1}(\mathcal{F}(m_1 \cdot (f_1)^2) \cdot M_2) \cdot \mathcal{F}^{-1}(M_1 \cdot \mathcal{F}(m_2^* \cdot (f_2^*)^2))}}}$$

Note that it does not require any more FFTs for the normalization, requiring fewer FFTs than normalized cross-corrections, making it the fastest of the normalized measures. The FFT based versions can be used in HydrolabPIV by setting:

```
opt = setpivopt('measure',@maskedmqd,true,@(x) -x);
opt = setpivopt('measure',@maskednmqd,true,@(x) -x);
```

7.2.4 Minimum maximum difference ($\alpha \to \infty$)

The last special case is the minimum maximum difference

$$md_{(r,s;\alpha\to\infty)} = \max_{i,j} \left| f_{1(i+r,j+s)} - f_{2(i,j)} \right|,$$

which works well with noisy images, but is sensitive to outliers in the image, for instance hotpixels or loss of particles due to both in-plane and out-of-plane motion. As allways it exists in a masked/weighted

$$mmd_{(r,s;\alpha\to\infty)} = \max_{i,j} \left(\text{sign}(w) \left| f_{1(i+r,j+s)} - f_{2(i,j)} \right| \right),$$

and normalized form

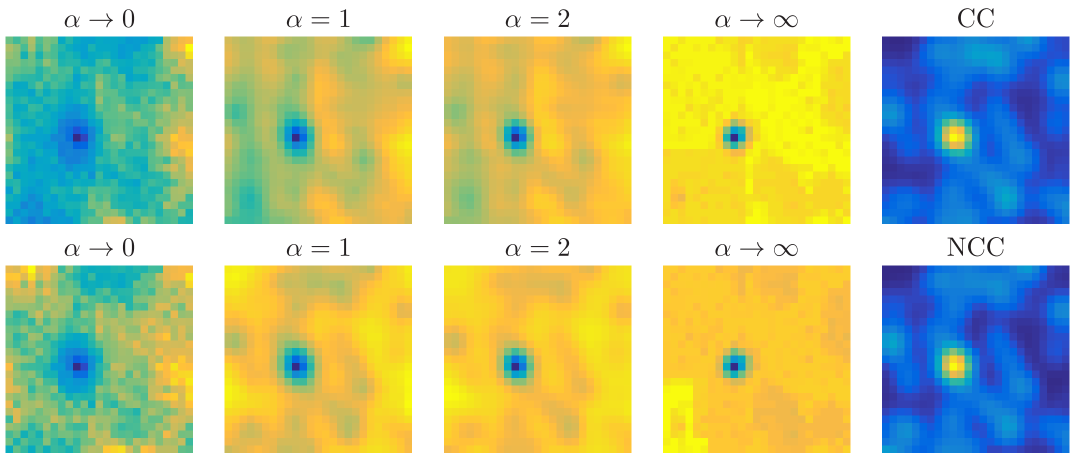
$$mnmd_{(r,s;\alpha\to\infty)} = \frac{\max_{i,j} \left(\text{sign}(w) \left| f_{1(i+r,j+s)} - f_{2(i,j)} \right| \right)}{\sqrt{\left(\max_{i,j} \left(\text{sign}(w) \left| f_{1(i+r,j+s)} \right| \right) \right) \cdot \left(\max_{i,j} \left(\text{sign}(w) \left| f_{2(i,j)} \right| \right) \right)}}.$$

7.3 Subpixel interpolation

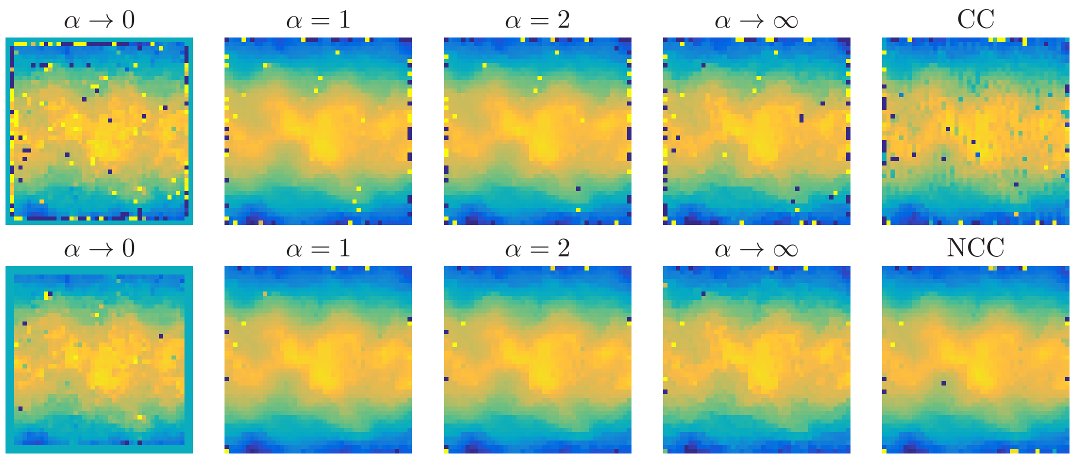
To gain increased accuracy PIV uses subpixel interpolation, based on assumptions of how small particles are imaged. The first PIV codes used two 1D quadratic fit using 3 points Raffel et al (2007), one in each velocity component. These subpixel interpolation are based on Newtons method with one iteration for finding the peak. To increase the accuracy of the peak position further, one can apply a preconditioner to accelerate the convergence, for instance if the peak is assumed to be Gaussian, the Newton method should converge in one iteration if the *log* functions is applied to the measures before subpixel interpolation (Willert and Gharib, 1991). Note that when peak is iteratively found using multipass with distorted passes, the use of *log* is not that important. While 1D subpixel does work, the idea should be generalized to 2D using a 3x3 bi-quadratic fit or 3x3 quadratic least squares fit to reduce bias errors (Nobach and Honkanen, 2005). As an alternative to using the *log* function, a nonlinear least squares fit to a Gaussian using Levenberg-Marquardt is also provided. There is also 5x5 version of the quadratic least squares and nonlinear least squares fit, these often give less outliers but with little or no increase in accuracy. Outliers caused by the subpixel interpolation are usually due to the Newton iteration diverging and are easily detected and replaced by the outlier filter. The subpixel interpolation in HydrolabPIV is set using:

```
 opt = setpivopt(..., 'subpixel', @subpixelfun, ...)
```

for example using these functions



(a) Measures



(b) U component of velocity, single pass with no outlier replacements

Figure 8: Minkowkski difference measures and cross-correlation. Normalized version below.

```
opt = setpivopt('subpixel',@subpixel3x2)
opt = setpivopt('subpixel',@subpixel3x3)
opt = setpivopt('subpixel',@subpixel3x3ls)
opt = setpivopt('subpixel',@subpixel3x3lm)
opt = setpivopt('subpixel',@subpixel5x5ls)
opt = setpivopt('subpixel',@subpixel5x5lm)
opt = setpivopt('subpixel',@subpixelnone)
```

As with the measure function, the subpixel interpolation can be set using a custom function

```
  [x0,delta,out] = subpixelfun(F)
```

where the $subpixelfun$ should return the integer position from the top corner, $x0$, the subpixel correction, $delta$, and an indication whether the method believes the result is valid ($out = 1$) or an outlier ($out = 0$).

References

Adrian RJ (1991) Particle-imaging techniques for experimental fluid mechanics. Annu Rev Fluid Mech 23:261–304

Christensen KT (2004) The influence of peak-locking errors on turbulence statistics computed from piv ensembles. Experiments in Fluids 36:484–497

Dalziel S (2012) DigiFlow user guide. Dalziel research partners, 3.4 edn, http://www.dalzielresearch.com/digiflow/digiflow.pdf

Gui LC, Merzkirch W (1997) A fast mask technique for the phase-separated evaluation of two phase piv recordings. In: Procedings of the 7th international conference on laser anemometry advances and applications, pp 447–454

Gui LC, Merzkirch W, Lindken R (1998) An advanced mqd tracking algorithm for dpiv. In: The 9th International Symposium on "Application of laser technique to fluid mechanics"

Huang HT, Fiedler HE, Wang JJ (1993a) Limitation and improvement of piv. part i: Limitation of conventional technique due to deformation of particle image patterns. Experiments in Fluids 15:168–174

Huang HT, Fiedler HE, Wang JJ (1993b) Limitation and improvement of piv. part ii: Particle image distortion, a novel technique. Experiments in Fluids 15:263–273

Kolaas J (2014) Optimization of optical measurement techniques in fluid mechanics with application to micro fluidics, multiphase flow and water waves. PhD thesis, University of Oslo

Nobach H, Honkanen M (2005) Two-dimensional gaussian regression for sub-pixel displacement estimation in particle image velocimetry or particle position estimation in particle tracking velocimetry. Experiments in Fluids 38:511–515

Olsen MG, Adrian RJ (2000) Out-of-focus effects on particle image visibility and correlation in microscopic particle image velocimetry. Experiments in Fluids 29:S166–S174

Padfield D (2012) Masked object registration in the fourier domain. IEEE Transactions on image processing 21(5):2706–2718

Raffel M, Willert CE, Wereley ST, Kompenhans J (2007) Particle image velocimetry: a practical guide, 2nd edn. Springer Berlin Heidelberg New York

Scarano F (2002) Iterative image deformation methods in piv. Measurement Science and Technology 13:R1–R19

Sveen JK, Cowen EA (2004) Quantitative imaging techniques and their application to wavy flows. In: Grue J, Liu PLF, Pedersen GK (eds) PIV and Water Waves, World Scientific Pub. Co, chap 1, pp 1–49

Wereley ST, Meinhart CD (2005) Micron-resolution particle image velocimetry. In: Breuer KS (ed) Microscale diagnostic techniques, Springer Berlin Heidelberg New York, chap 2, pp 51–112

Westerweel J (1993) Digital Particle Image Velocimetry -Theory and Application. Delft University Press

Westerweel J, Scarano F (2005) Universal outlier detection for piv data. Exp Fluids 39:1096–1100, DOI 10.1007/s00348-005-0016-6

Willert CE, Gharib M (1991) Digital particle image velcimetry. Experiments in Fluids 10:181–193