PIVUQ

Release 0.4.1

Lento Manickathan

Jun 04, 2023

CONTENTS:

PIVUQ (*pivuq*) is Python library for PIV Uncertainty Quantification. Primary aim is to implement UQ algorithms for PIV techniques. Future goals include possible extensions to other domains including but not limited to optical flow and BOS.

Note: This project is still under active development. Proper documentation and DOI link coming soon.

CHAPTER

ONE

HOW TO CITE

Work in progress. In future, please cite the following paper:

Citation

Manickathan et al. (2022). PIVUQ: Uncertainty Quantification Toolkit for Quantitative Flow Visualization. *in prep*.

CHAPTER

TWO

USAGE

2.1 Installation

To use PIVUQ (*pivuq*), first install it using pip:

```
$ pip install pivuq
```

CHAPTER

THREE

EXAMPLES

3.1 Image matching

3.1.1 Description

Image matching using `skimage.transform.warp` [source].

3.1.2 Setup

Load packages

[1]:
```
%reload_ext autoreload
%autoreload 2
```

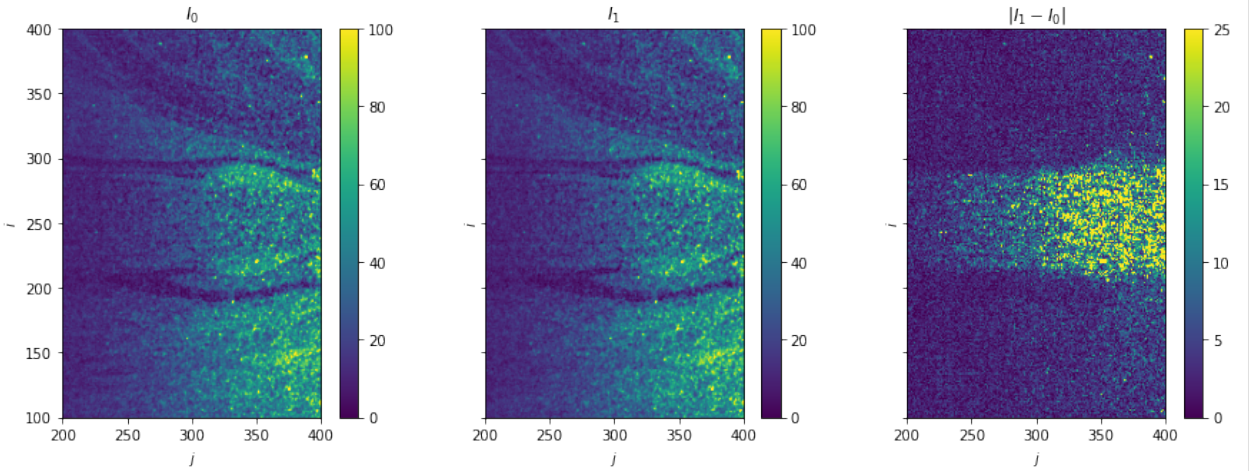
[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import os

import pivuq
```

Load images

[3]:
```
parent_path = "./data/particledisparity_code_testdata/"
image_pair = np.array(
    [plt.imread(os.path.join(parent_path + ipath)).astype("float") for ipath in ["B00010.
↪tif", "B00011.tif"]]
)
```

Plot: Raw image pairs

[4]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 5))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(image_pair[i], vmax=100)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$I_{i}$")

ax = axes[-1]
im = ax.imshow(np.abs(image_pair[1] - image_pair[0]), vmin=0, vmax=25)
fig.colorbar(im, ax=ax)
ax.set(title="$|I_1 - I_0|$")

for ax in axes:
    ax.set(xlim=(200, 400), ylim=(100, 400), xlabel="$j$", ylabel="$i$")
```

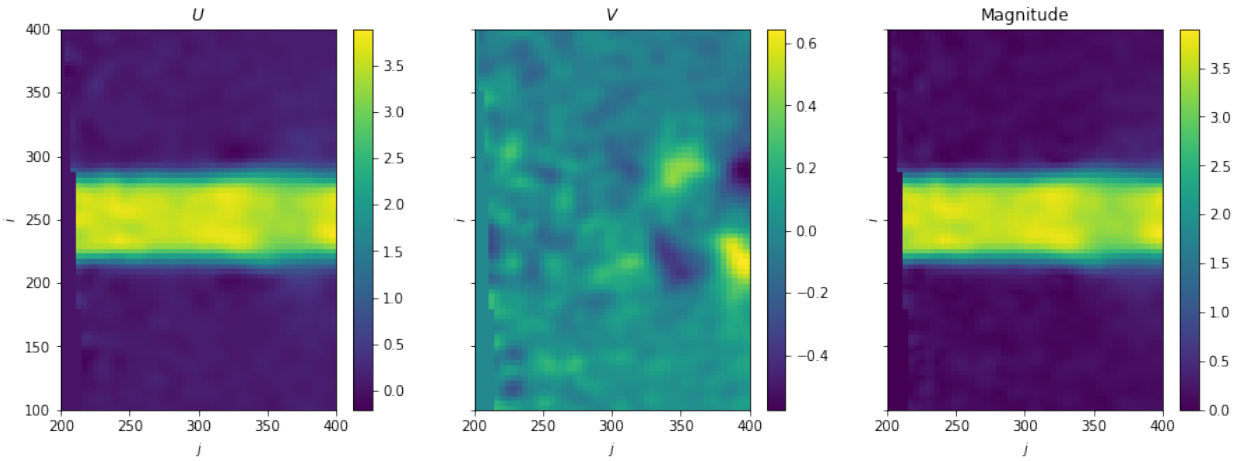


Load reference velocity

[5]:
```
data = np.loadtxt(os.path.join(parent_path + "B00010.dat"), skiprows=3).T

I, J = 128, 128
X = np.reshape(data[0], (I, J)) - 1  # zero-index
Y = np.reshape(data[1], (I, J)) - 1
U = np.stack((np.reshape(data[2], (I, J)), np.reshape(data[3], (I, J))))
```

Plot: Reference velocity fields

[6]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 5))

for i, (ax, var) in enumerate(zip(axes[:2], ["U", "V"])):
    im = ax.pcolormesh(X, Y, U[i])
    fig.colorbar(im, ax=ax)
    ax.set(title=f"${var}$")

ax = axes[-1]
im = ax.pcolormesh(X, Y, np.linalg.norm(U, axis=0))
fig.colorbar(im, ax=ax)
ax.set(title="Magnitude")

for ax in axes:
    ax.set(xlim=(200, 400), ylim=(100, 400), xlabel="$j$", ylabel="$i$")
```



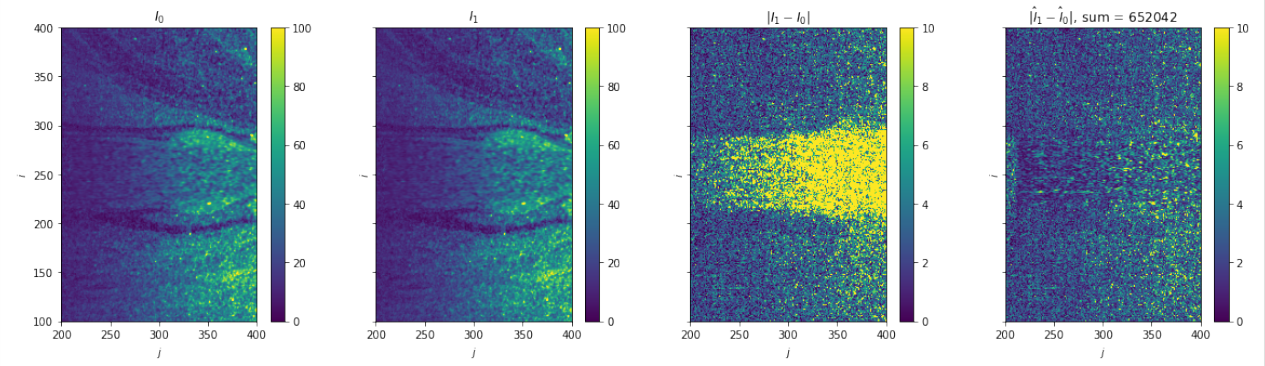
3.1.3 Image matching

[7]:
```
%%time
warped_image_pair = pivuq.warp(
    image_pair,
    U,
    velocity_upsample_kind="linear",
    direction="center",
    nsteps=5,
    order=1,
)
```

```
CPU times: user 219 ms, sys: 21.4 ms, total: 241 ms
Wall time: 241 ms
```

Plot: Warped image pairs

[8]:
```
fig, axes = plt.subplots(ncols=4, sharex=True, sharey=True, figsize=(20, 5))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(warped_image_pair[0], vmax=100)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$I_{i}$")

ax = axes[-2]
im = ax.imshow(np.abs(image_pair[1] - image_pair[0]), vmin=0, vmax=10)
fig.colorbar(im, ax=ax)
ax.set(title="$|I_1 - I_0|$")


ax = axes[-1]
im = ax.imshow(np.abs(warped_image_pair[1] - warped_image_pair[0]), vmin=0, vmax=10)
fig.colorbar(im, ax=ax)
ax.set(title=f"$|\hat{{I}}_1 - \hat{{I}}_0|$, sum = {np.abs(warped_image_pair[1] -␣
↪warped_image_pair[0]).sum():g}")

for ax in axes:
    ax.set(xlim=(200, 400), ylim=(100, 400), xlabel="$j$", ylabel="$i$")
```



3.2 Image matching - Cylinder wake

3.2.1 Description

Image matching using `skimage.transform.warp` [source].

3.2.2 Setup

Packages

[1]:
```
%reload_ext autoreload
%autoreload 2
```

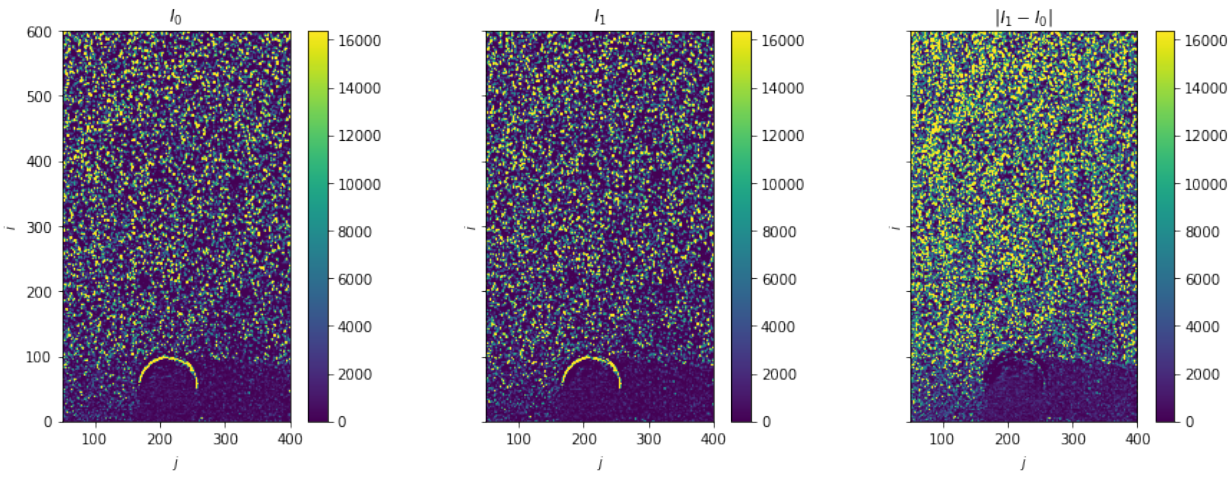
[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import os

import pivuq
```

Load images

[3]:
```
parent_path = "./data/cylinder_wake/"
image_pair = np.array(
    [plt.imread(os.path.join(parent_path + ipath)).astype("float") for ipath in ["frameA.
↪tif", "frameB.tif"]]
)
```

Plot raw images

[4]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 5))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(image_pair[i], vmax=2**14)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$I_{i}$")

ax = axes[-1]
im = ax.imshow(np.abs(image_pair[1] - image_pair[0]), vmin=0, vmax=2**14)
fig.colorbar(im, ax=ax)
ax.set(title="$|I_1 - I_0|$")

for ax in axes:
    ax.set(xlim=(50, 400), ylim=(0, 600), xlabel="$j$", ylabel="$i$")
```



3.2.3 Image Matching (PIV)

Load reference velocity

[5]:
```
data = np.loadtxt(os.path.join(parent_path + "vectors_PIV.dat"), skiprows=3).T

I, J = 75, 42
X = np.reshape(data[0], (I, J)) - 1  # zero-index
Y = np.reshape(data[1], (I, J)) - 1
U = np.stack((np.reshape(data[2], (I, J)), np.reshape(data[3], (I, J))))
```

Plot: reference fields

[6]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 5))

for i, (ax, var) in enumerate(zip(axes[:2], ["U", "V"])):
    im = ax.pcolormesh(X, Y, U[i])
    fig.colorbar(im, ax=ax)
    ax.set(title=f"${var}$")

ax = axes[-1]
im = ax.pcolormesh(X, Y, np.linalg.norm(U, axis=0), vmax=20)
fig.colorbar(im, ax=ax)
ax.set(title="Magnitude")

for ax in axes:
    ax.set(xlim=(0, 450), ylim=(0, 800), xlabel="$j$", ylabel="$i$")
```

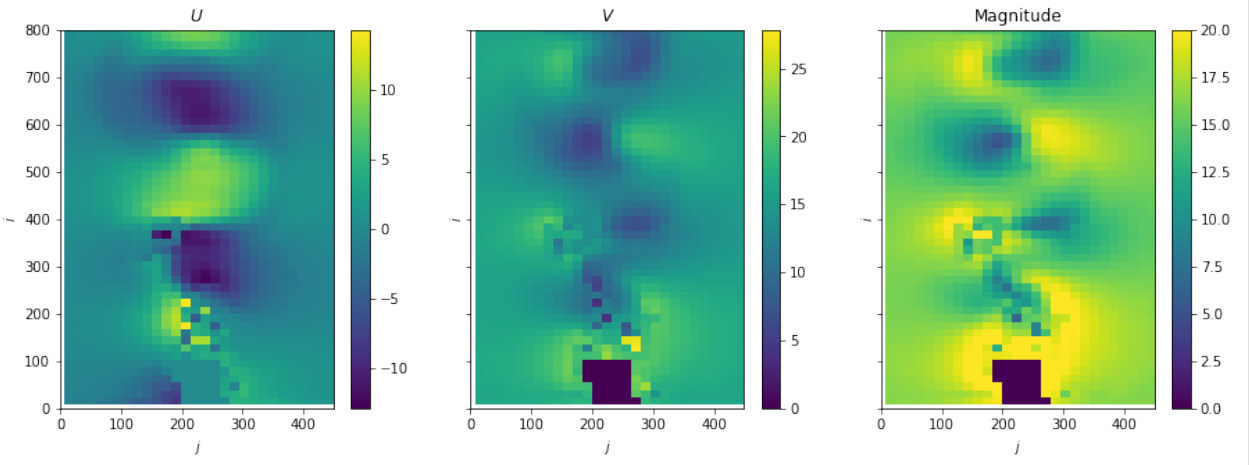


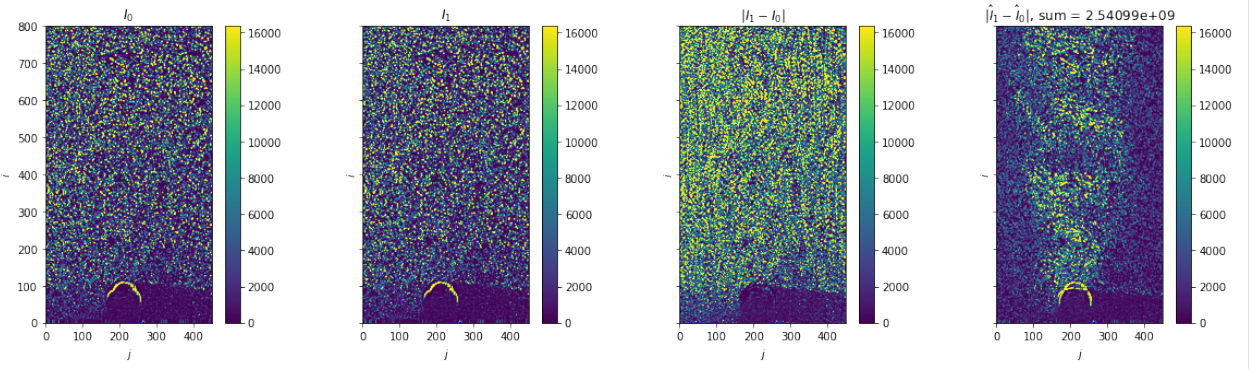
Image matching

[7]:
```
%%time
warped_image_pair = pivuq.warp(
    image_pair,
    U,
    velocity_upsample_kind="linear",
    direction="center",
    nsteps=1,
    order=1,
)
```

```
CPU times: user 201 ms, sys: 21.8 ms, total: 222 ms
Wall time: 223 ms
```

Plot: warped image pairs

[8]:
```
fig, axes = plt.subplots(ncols=4, sharex=True, sharey=True, figsize=(20, 5))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(warped_image_pair[0], vmax=2**14)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$I_{i}$")

ax = axes[-2]
im = ax.imshow(np.abs(image_pair[1] - image_pair[0]), vmax=2**14)
fig.colorbar(im, ax=ax)
ax.set(title="$|I_1 - I_0|$")


ax = axes[-1]
im = ax.imshow(np.abs(warped_image_pair[1] - warped_image_pair[0]), vmax=2**14)
fig.colorbar(im, ax=ax)
ax.set(title=f"$|\hat{{I}}_1 - \hat{{I}}_0|$, sum = {np.abs(warped_image_pair[1] -␣
↪warped_image_pair[0]).sum():g}")

for ax in axes:
    ax.set(xlim=(0, 450), ylim=(0, 800), xlabel="$j$", ylabel="$i$")
```



3.2.4 Image Matching (Optical Flow: pixel-level)

Load reference velocity

[9]:
```
data = np.load(os.path.join(parent_path + "vectors_OF.npz"))

X = data["X"]  # already zero-indexed
Y = data["Y"]
U = data["U"]  # u, v
```

Plot: reference fields

[10]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 5))

for i, (ax, var) in enumerate(zip(axes[:2], ["U", "V"])):
    im = ax.pcolormesh(X, Y, U[i])
    fig.colorbar(im, ax=ax)
    ax.set(title=f"${var}$")

ax = axes[-1]
im = ax.pcolormesh(X, Y, np.linalg.norm(U, axis=0), vmax=20)
fig.colorbar(im, ax=ax)
ax.set(title="Magnitude")

for ax in axes:
    ax.set(xlim=(0, 450), ylim=(0, 800), xlabel="$j$", ylabel="$i$")
```

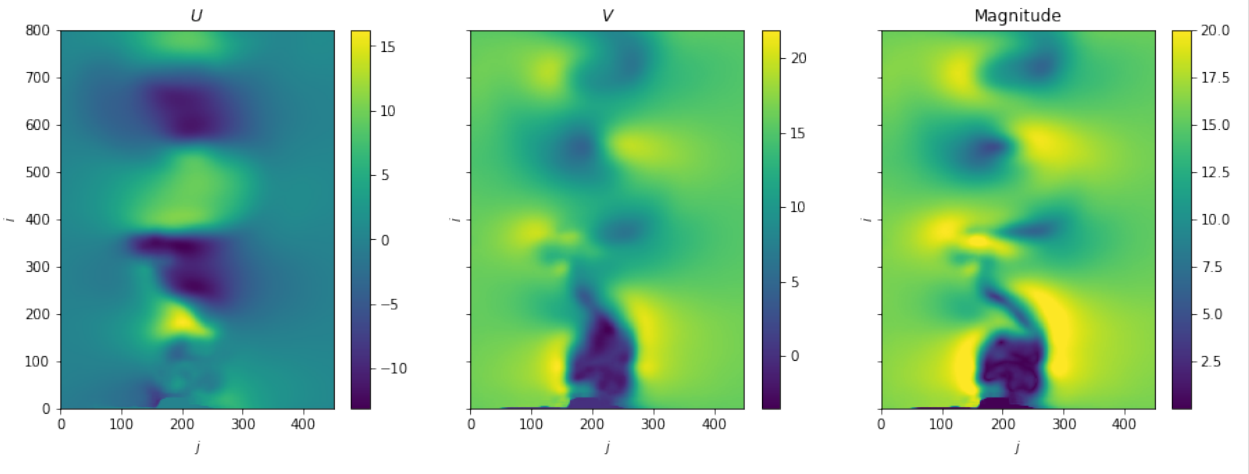


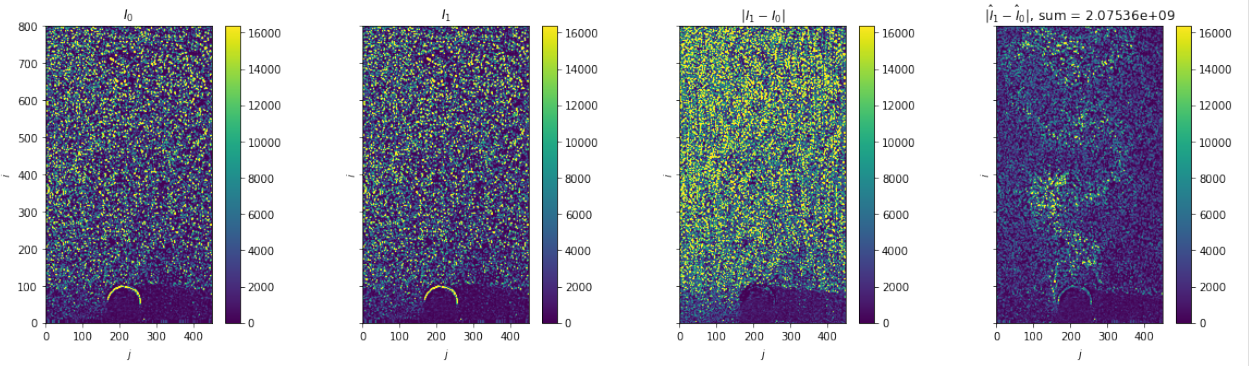
Image matching

[11]:
```
%%time
warped_image_pair = pivuq.warp(
    image_pair,
    U,
    velocity_upsample_kind="linear",
    direction="center",
    nsteps=1,
    order=1,
)
```

```
CPU times: user 57.8 ms, sys: 928 µs, total: 58.8 ms
Wall time: 58.5 ms
```

Plot: warped image pairs

[12]:
```
fig, axes = plt.subplots(ncols=4, sharex=True, sharey=True, figsize=(20, 5))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(warped_image_pair[0], vmax=2**14)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$I_{i}$")

ax = axes[-2]
im = ax.imshow(np.abs(image_pair[1] - image_pair[0]), vmax=2**14)
fig.colorbar(im, ax=ax)
ax.set(title="$|I_1 - I_0|$")


ax = axes[-1]
im = ax.imshow(np.abs(warped_image_pair[1] - warped_image_pair[0]), vmax=2**14)
fig.colorbar(im, ax=ax)
ax.set(title=f"$|\hat{{I}}_1 - \hat{{I}}_0|$, sum = {np.abs(warped_image_pair[1] -␣
↪warped_image_pair[0]).sum():g}")

for ax in axes:
    ax.set(xlim=(0, 450), ylim=(0, 800), xlabel="$j$", ylabel="$i$")
```



3.3 Disparity calculation using Iterative Lucas-Kanade

3.3.1 Description

Disparity calculation using optical flow registration technique, Iterative Lucas-Kanade [source].

3.3.2 Setup

Packages

[1]:
```
%reload_ext autoreload
%autoreload 2
```

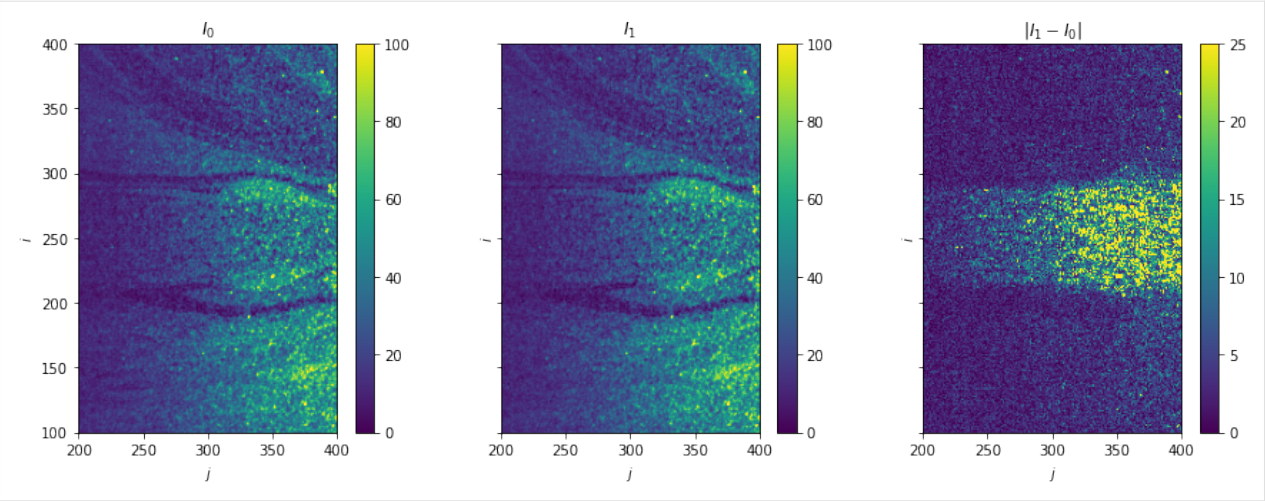
[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import os

import pivuq
```

Load images

[3]:
```
parent_path = "./data/particledisparity_code_testdata/"
image_pair = np.array(
    [plt.imread(os.path.join(parent_path + ipath)).astype("float") for ipath in ["B00010.
↪tif", "B00011.tif"]]
)
```

Plot raw data

[4]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 5))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(image_pair[i], vmax=100)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$I_{i}$")

ax = axes[-1]
im = ax.imshow(np.abs(image_pair[1] - image_pair[0]), vmin=0, vmax=25)
fig.colorbar(im, ax=ax)
ax.set(title="$|I_1 - I_0|$")

for ax in axes:
    ax.set(xlim=(200, 400), ylim=(100, 400), xlabel="$j$", ylabel="$i$")
```



Load reference velocity and disparity

[5]:
```
data = np.loadtxt(os.path.join(parent_path + "B00010_UQ.dat"), skiprows=3).T

I, J = 128, 128
X_ref = np.reshape(data[0], (I, J)) - 1  # zero-index
Y_ref = np.reshape(data[1], (I, J)) - 1
U_ref = np.stack((np.reshape(data[2], (I, J)), np.reshape(data[3], (I, J))))
e_ref = np.stack((np.reshape(data[4], (I, J)), np.reshape(data[5], (I, J))))
N_ref = np.reshape(data[6], (I, J))

window_size = X_ref[0][1] - X_ref[0][0]
```

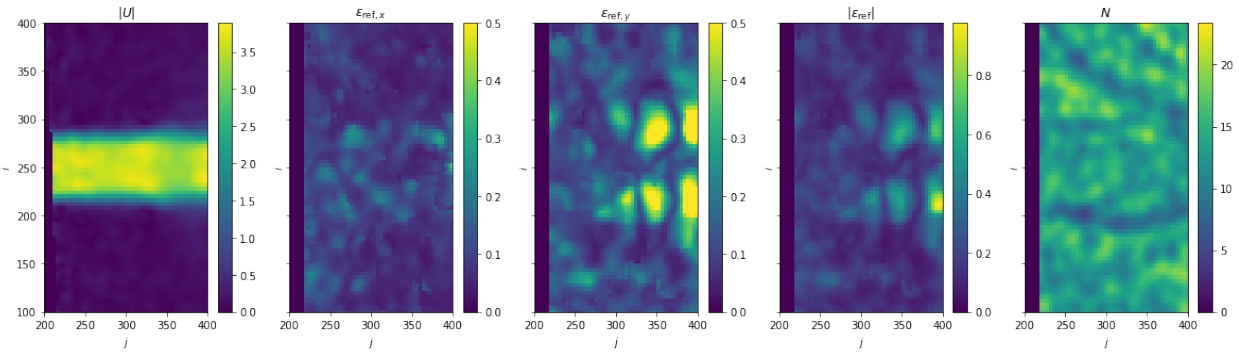
Plot reference velocity and disparity

[6]:
```
fig, axes = plt.subplots(ncols=5, sharex=True, sharey=True, figsize=(20, 5))

ax = axes[0]
im = ax.pcolormesh(X_ref, Y_ref, np.linalg.norm(U_ref, axis=0))
fig.colorbar(im, ax=ax)
ax.set(title=r"$|U|$")

for i, (ax, var) in enumerate(zip(axes[1:4], [r"\epsilon_{\mathrm{ref},x}", r"\epsilon_{\
↪mathrm{ref},y}"])):
    im = ax.pcolormesh(X_ref, Y_ref, e_ref[i], vmax=0.5)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"${var}$")

ax = axes[3]
im = ax.pcolormesh(X_ref, Y_ref, np.linalg.norm(e_ref, axis=0))
fig.colorbar(im, ax=ax)
ax.set(title=r"$|\epsilon_{\mathrm{ref}}|$")

ax = axes[4]
im = ax.pcolormesh(X_ref, Y_ref, N_ref)
fig.colorbar(im, ax=ax)
ax.set(title=r"$N$")
```

(continues on next page)

(continued from previous page)

```
for ax in axes.ravel():
    ax.set(xlim=(200, 400), ylim=(100, 400), xlabel="$j$", ylabel="$i$")
```



3.3.3 Disparity calculation

[7]:
```
%%time
X, Y, e = pivuq.disparity.ilk(
    image_pair,
    U_ref,
    window_size=16,
    window="gaussian",
    velocity_upsample_kind="linear",
    warp_direction="center",
    warp_order=1,
    warp_nsteps=1,
)
```

```
CPU times: user 2.31 s, sys: 50 ms, total: 2.36 s
Wall time: 2.37 s
```

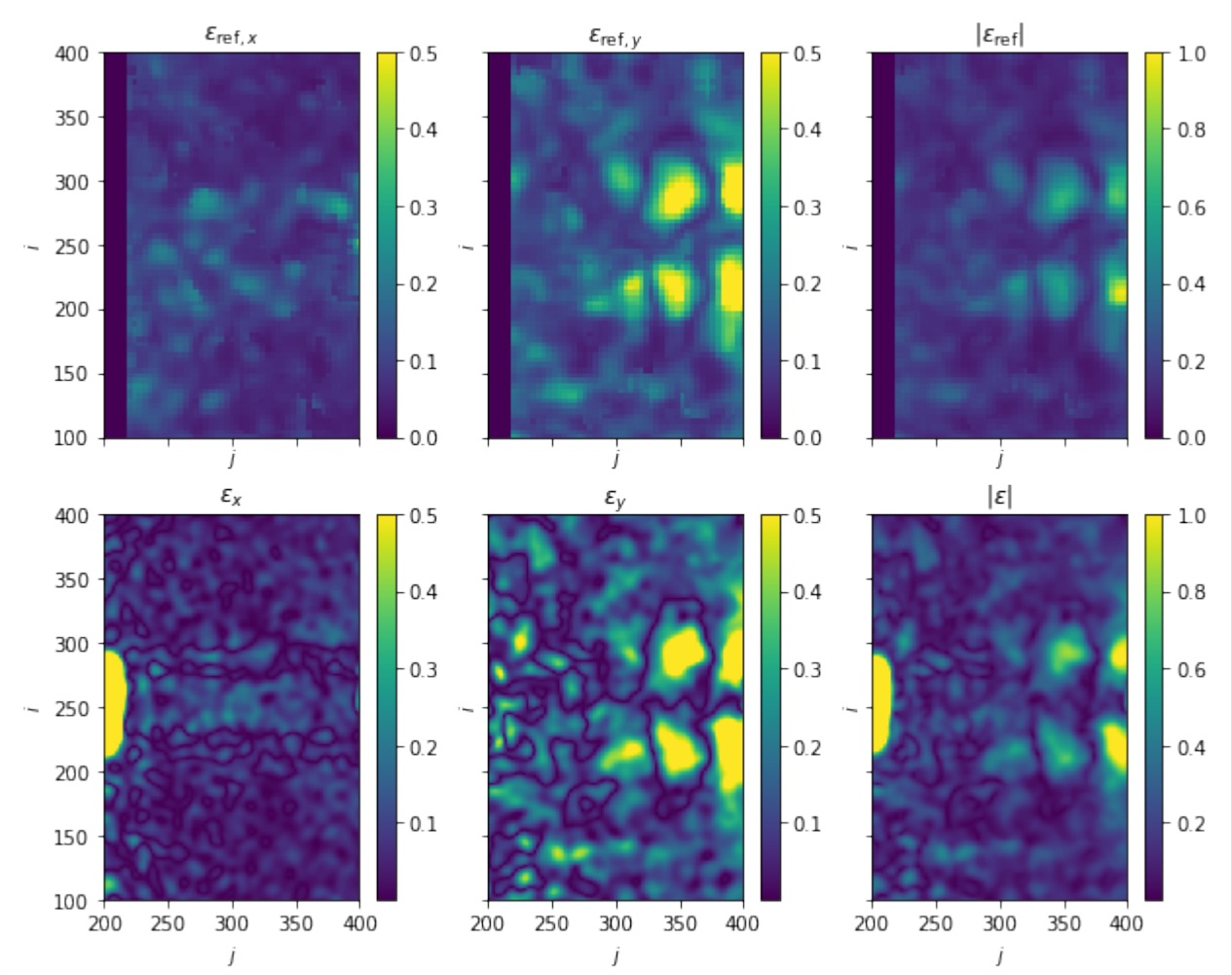
3.3.4 Plot: Disparity map - Reference vs. ILK

[8]:
```
fig, axes = plt.subplots(nrows=2, ncols=3, sharex=True, sharey=True, figsize=(10, 8))

for i, (ax, var) in enumerate(zip(axes[0, :3], [r"\epsilon_{\mathrm{ref},x}", r"\epsilon_
↪{\mathrm{ref},y}"])):
    im = ax.pcolormesh(X_ref, Y_ref, e_ref[i], vmax=0.5)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"${var}$")

ax = axes[0, -1]
im = ax.pcolormesh(X_ref, Y_ref, np.linalg.norm(e_ref, axis=0), vmax=1)
fig.colorbar(im, ax=ax)
ax.set(title=r"$|\epsilon_{\mathrm{ref}}|$")
```

(continues on next page)

(continued from previous page)

```
for i, (ax, var) in enumerate(zip(axes[1, :3], [r"\epsilon_x}", r"\epsilon_y}"])):
    im = ax.pcolormesh(X, Y, e[i], vmax=0.5)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"${var}$")

ax = axes[1, -1]
im = ax.pcolormesh(X, Y, np.linalg.norm(e, axis=0), vmax=1)
fig.colorbar(im, ax=ax)
ax.set(title=r"$|\epsilon|$")

for ax in axes.ravel():
    ax.set(xlim=(200, 400), ylim=(100, 400), xlabel="$j$", ylabel="$i$")
```



3.4 Disparity calculation using Iterative Lucas-Kanade - Cylinder wake

3.4.1 Description

Disparity calculation using optical flow registration technique, Iterative Lucas-Kanade [source].

3.4.2 Setup

Packages

[1]:
```
%reload_ext autoreload
%autoreload 2
```

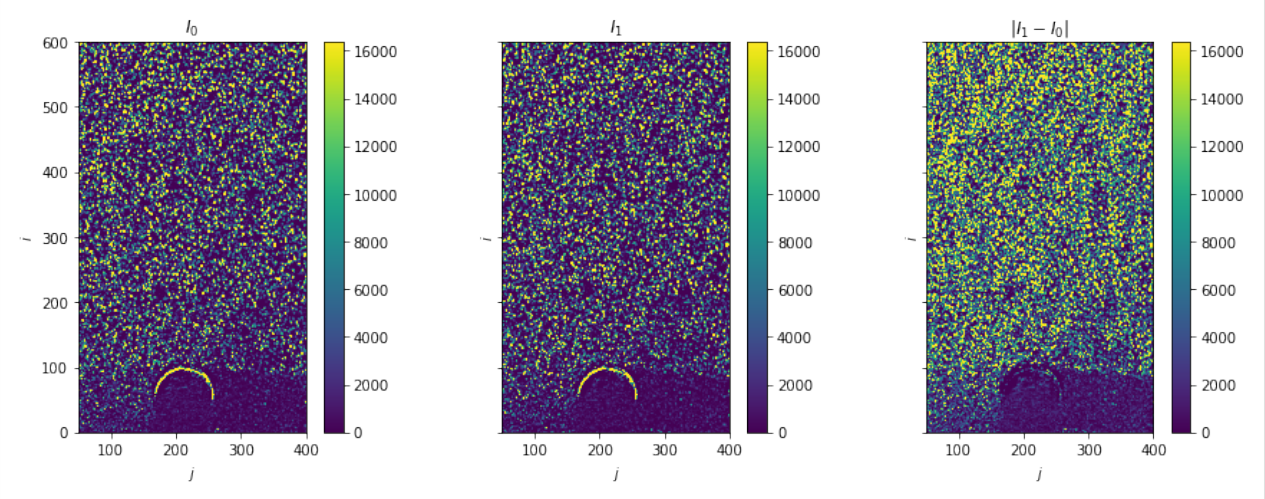
[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import os

import pivuq
```

Load images

[3]:
```
parent_path = "./data/cylinder_wake/"
image_pair = np.array(
    [plt.imread(os.path.join(parent_path + ipath)).astype("float") for ipath in ["frameA.
↪tif", "frameB.tif"]]
)
```

Plot raw images

[4]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 5))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(image_pair[i], vmax=2**14)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$I_{i}$")

ax = axes[-1]
im = ax.imshow(np.abs(image_pair[1] - image_pair[0]), vmin=0, vmax=2**14)
fig.colorbar(im, ax=ax)
ax.set(title="$|I_1 - I_0|$")

for ax in axes:
    ax.set(xlim=(50, 400), ylim=(0, 600), xlabel="$j$", ylabel="$i$")
```



Load reference data

[5]:
```
ref = {}
```

PIV

[6]:
```
data = np.loadtxt(os.path.join(parent_path + "vectors_PIV.dat"), skiprows=3).T

I, J = 75, 42
ref["PIV"] = {
    "X": np.reshape(data[0], (I, J)) - 1,  # zero-index,
    "Y": np.reshape(data[1], (I, J)) - 1,
    "U": np.stack((np.reshape(data[2], (I, J)), np.reshape(data[3], (I, J)))),
}
```

OF

[7]:
```
data = np.load(os.path.join(parent_path + "vectors_OF.npz"))

ref["OF"] = {
    "X": data["X"],  # already zero-indexed
    "Y": data["Y"],
    "U": data["U"],  # u, v
}
```

3.4.3 Disparity calculation

[8]:
```
err = {"PIV": {}, "OF": {}}
```

PIV

[9]:
```
%%time
err["PIV"]["X"], err["PIV"]["Y"], err["PIV"]["D"] = pivuq.disparity.ilk(
    image_pair,
```

(continues on next page)

(continued from previous page)

```
    ref["PIV"]["U"],
    window_size=16,
    window="gaussian",
    velocity_upsample_kind="linear",
    warp_direction="center",
    warp_order=1,
    warp_nsteps=1,
)
```

```
CPU times: user 6.48 s, sys: 80 ms, total: 6.56 s
Wall time: 6.58 s
```

OF

[10]:
```
%%time
err["OF"]["X"], err["OF"]["Y"], err["OF"]["D"] = pivuq.disparity.ilk(
    image_pair,
    ref["OF"]["U"],
    window_size=16,
    window="gaussian",
    velocity_upsample_kind="linear",
    warp_direction="center",
    warp_order=1,
    warp_nsteps=1,
)
```

```
CPU times: user 6.34 s, sys: 44.8 ms, total: 6.38 s
Wall time: 6.4 s
```

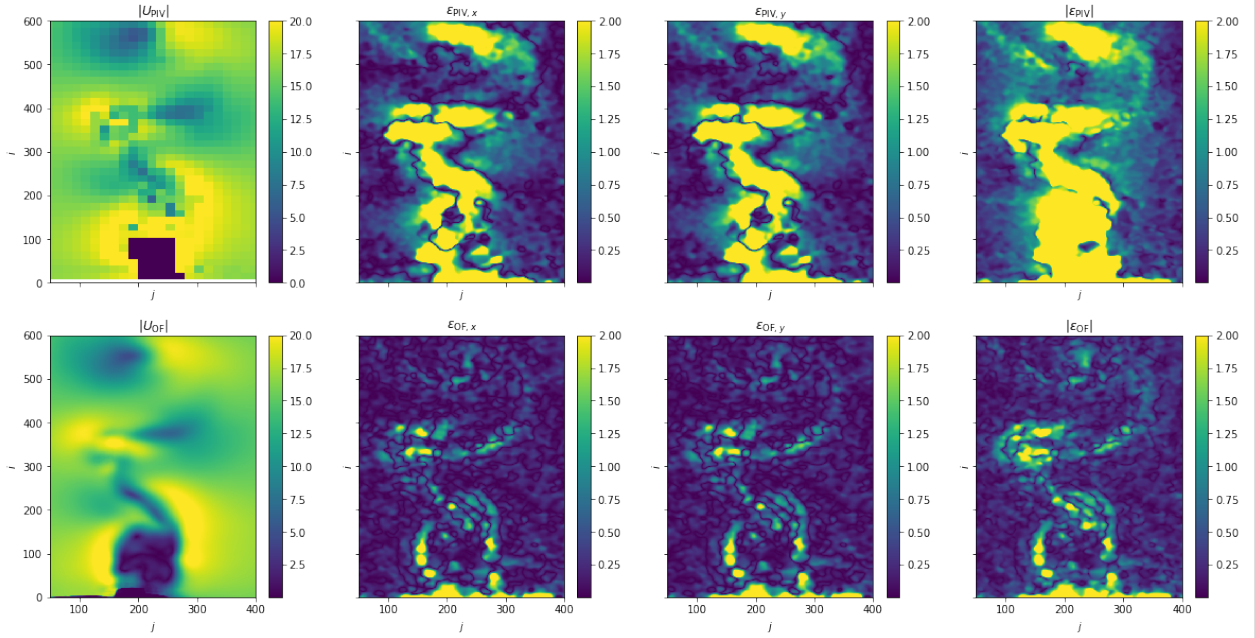
3.4.4 Plot: Disparity map - PIV vs. OF

[11]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, sharex=True, sharey=True, figsize=(20, 10))

for i, case in enumerate(["PIV", "OF"]):
    ax = axes[i, 0]
    im = ax.pcolormesh(ref[case]["X"], ref[case]["Y"], np.linalg.norm(ref[case]["U"],␣
↪axis=0), vmax=20)
    fig.colorbar(im, ax=ax)
    ax.set(title=rf"$|U_{{\mathrm{{{case}}}}}|$")

    for j, (ax, var) in enumerate(
        zip(
            axes[i, 1:4],
            [
                rf"\epsilon_{{\mathrm{{{case}}},x}}",
                rf"\epsilon_{{\mathrm{{{case}}},y}}",
            ],
        )
    ):
        im = ax.pcolormesh(err[case]["X"], err[case]["Y"], err[case]["D"][i], vmax=2)
        fig.colorbar(im, ax=ax)
```

(continues on next page)

(continued from previous page)

```
        ax.set(title=f"${var}$")

    ax = axes[i, 3]
    im = ax.pcolormesh(err[case]["X"], err[case]["Y"], np.linalg.norm(err[case]["D"],␣
↪axis=0), vmax=2)
    fig.colorbar(im, ax=ax)
    ax.set(title=rf"$|\epsilon_{{\mathrm{{{case}}}}}|$")

for ax in axes.ravel():
    ax.set(xlim=(50, 400), ylim=(0, 600), xlabel="$j$", ylabel="$i$")
```



3.5 Disparity vector computation - Cylinder wake

3.5.1 Description

Based on paper and source code: - Sciacchitano, A., Wieneke, B., & Scarano, F. (2013). PIV uncertainty quantification by image matching. *Measurement Science and Technology*, 24 (4). https://doi.org/10.1088/0957-0233/24/4/045302. - http://piv.de/uncertainty/?page_id=221

Step 1: Particle peak detection

As described by (Sciacchitano et al., 2013, Eq. 1), the image intensity product Π from image matching intensities is defined as:

$$\Pi = \hat{I}_1 \hat{I}_2$$

The peaks image φ is defined as:

$$\varphi(i,j) = \begin{cases} 1 & \text{if } \Pi(i,j) \text{ is a relative maximum} \\ 0 & \text{otherwise} \end{cases}$$

Step 2: Disparity vector computation

The sub-pixel peak position estimator adopted here is the standard 3-point Gaussian fit. The particle positions of times t_1 and t_2 are defined as $\boldsymbol{X}^1 = \{x_1^1, x_2^1, ..., x_N^1\}$ and $\boldsymbol{X}^2 = \{x_1^2, x_2^2, ..., x_N^2\}$. Discrete disparity vectors are defined as:

$$\boldsymbol{D} = \boldsymbol{X}^2 - \boldsymbol{X}^1$$

3.5.2 Setup

Packages

[1]:
```
%reload_ext autoreload
%autoreload 2
```

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import os

import pivuq
```

Load images

[3]:
```
parent_path = "./data/cylinder_wake/"
image_pair = np.array(
    [plt.imread(os.path.join(parent_path + ipath)).astype("float") for ipath in ["frameA.
↪tif", "frameB.tif"]]
)
```

Load reference velocity

[4]:
```
data = np.load(os.path.join(parent_path + "vectors_OF.npz"))

X_ref = data["X"]  # already zero-indexed
Y_ref = data["Y"]
U_ref = data["U"]  # u, v
```

3.5.3 Warp image pair

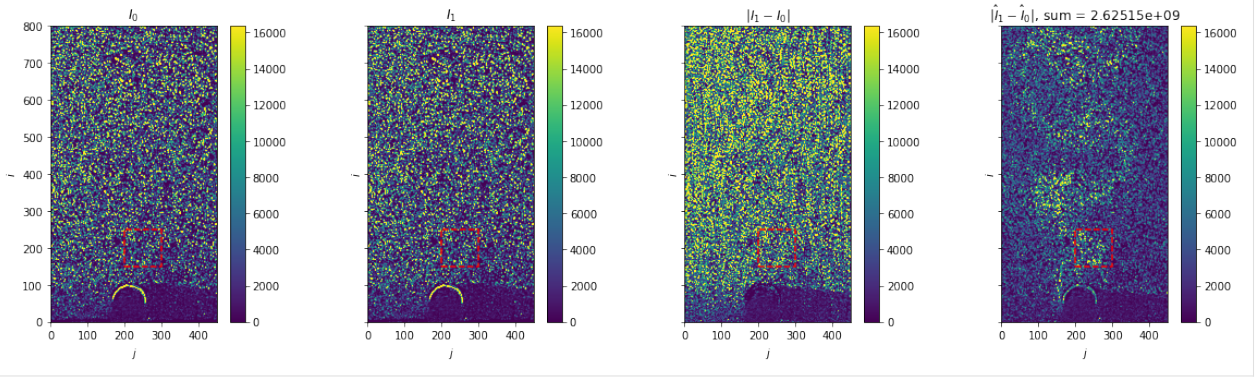
Image warping using Whittaker-shannon interpolation (`order=-1`)

[5]:
```
%%time
warped_image_pair = pivuq.warp(
    image_pair,
    U_ref,
    velocity_upsample_kind="linear",
    direction="center",
    nsteps=1,
    order=-1,
)
```

```
CPU times: user 1.45 s, sys: 44.8 ms, total: 1.49 s
Wall time: 541 ms
```

Plot warped images

[6]:
```
fig, axes = plt.subplots(ncols=4, sharex=True, sharey=True, figsize=(20, 5))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(warped_image_pair[0], vmax=2**14)
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$I_{i}$")

ax = axes[-2]
im = ax.imshow(np.abs(image_pair[1] - image_pair[0]), vmax=2**14)
fig.colorbar(im, ax=ax)
ax.set(title="$|I_1 - I_0|$")


ax = axes[-1]
im = ax.imshow(np.abs(warped_image_pair[1] - warped_image_pair[0]), vmax=2**14)
fig.colorbar(im, ax=ax)
ax.set(title=f"$|\hat{{I}}_1 - \hat{{I}}_0|$, sum = {np.abs(warped_image_pair[1] -␣
↪warped_image_pair[0]).sum():g}")

box = [200, 300, 150, 250]

for ax in axes:
    ax.set(xlim=(0, 450), ylim=(0, 800), xlabel="$j$", ylabel="$i$")
    ax.plot(
        [box[0], box[1], box[1], box[0], box[0]],
        [box[2], box[2], box[3], box[3], box[2]],
        "--",
        c="r",
    )
```



3.5.4 Disparity vector computation

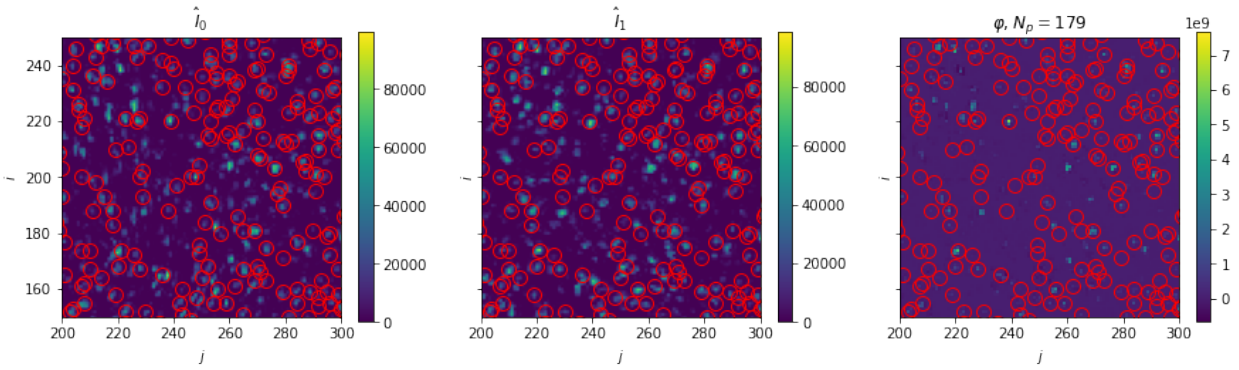
[7]:
```
%%time
D, c = pivuq.lib.disparity_vector_computation(warped_image_pair, radius=1, sliding_
↪window_size=16)

peak_coords = np.stack(np.where(c))
```

```
CPU times: user 727 ms, sys: 285 ms, total: 1.01 s
Wall time: 1.14 s
```

[8]:
```
warped_image_pair_filt = pivuq.lib.sliding_avg_subtract(warped_image_pair, window_
↪size=16)
```

3.5.5 Plot: Disparity map and peaks

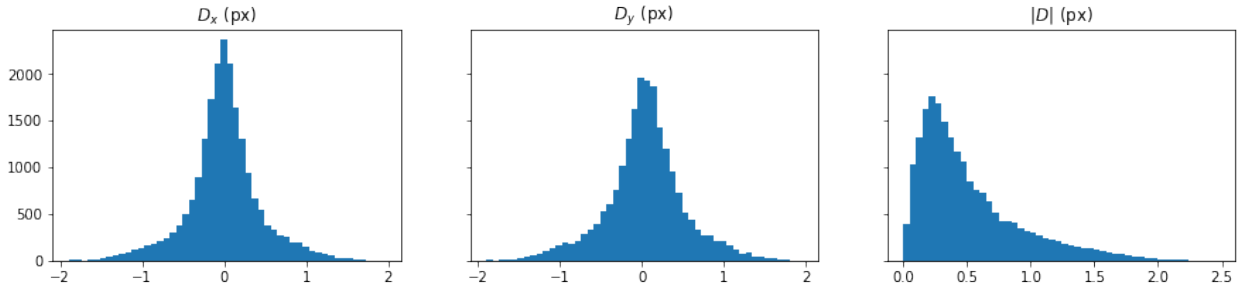
[9]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 3.75))

for i, ax in enumerate(axes[:2]):
    im = ax.imshow(warped_image_pair[i])
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$\hat{{I}}_{i}$")

ax = axes[2]
im = ax.imshow(warped_image_pair_filt[0] * warped_image_pair_filt[1], interpolation="none
↪")
fig.colorbar(im, ax=ax)
ax.set(title=rf"$\varphi, N_p={np.count_nonzero(c[box[2]:box[3],box[0]:box[1]])}$")

for ax in axes:
    ax.plot(
        peak_coords[1],
        peak_coords[0],
        "o",
        markerfacecolor="none",
        markersize=10,
        c="r",
    )
    ax.set(xlim=box[:2], ylim=box[2:], xlabel="$j$", ylabel="$i$")
```



3.5.6 Plot: Disparity histogram

[10]:
```
D_hist = [D[i][(np.abs(D[0]) > 0) & (np.abs(D[1]) > 0)] for i in range(2)]
```

[11]:
```
fig, axes = plt.subplots(ncols=3, sharey=True, figsize=(15, 3))

for ax, D_hist_i, label in zip(axes[:2], D_hist, ["$D_x$ (px)", "$D_y$ (px)"]):
    ax.hist(D_hist_i, bins=50)
    ax.set(title=label)

ax = axes[-1]
ax.hist(np.linalg.norm(D_hist, axis=0), bins=50)
ax.set(title="$|D|$ (px)");
```



3.6 Uncertainty Quantification using Image Matching

3.6.1 Description

Based on paper and source code: - Sciacchitano, A., Wieneke, B., & Scarano, F. (2013). PIV uncertainty quantification by image matching. *Measurement Science and Technology*, 24 (4). https://doi.org/10.1088/0957-0233/24/4/045302. - http://piv.de/uncertainty/?page_id=221

Step 1: Particle peak detection

As described by (Sciacchitano et al., 2013, Eq. 1), the image intensity product Π from image matching intensities is defined as:

$$\Pi = \hat{I}_1 \hat{I}_2$$

The peaks image φ is defined as:

$$\varphi(i,j) = \begin{cases} 1 & \text{if } \Pi(i,j) \text{ is a relative maximum} \\ 0 & \text{otherwise} \end{cases}$$

Step 2: Disparity vector computation

The sub-pixel peak position estimator adopted here is the standard 3-point Gaussian fit. The particle positions of times t_1 and t_2 are defined as $\boldsymbol{X}^1 = \{x_1^1, x_2^1, ..., x_N^1\}$ and $\boldsymbol{X}^2 = \{x_1^2, x_2^2, ..., x_N^2\}$. Discrete disparity vectors are defined as:

$$\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_N\} = \boldsymbol{X}^2 - \boldsymbol{X}^1$$

Step 3: Disparity ensemble statistics inside window

The mean of disparity set inside a window is defined as:

$$\mu = \frac{1}{N} \sum_{i \in N} c_i d_i$$

where $c_i = \sqrt{\Pi(x_i)}$ for $i = 1, 2, ..., N$.

The standard deviation of disparity set inside a window is defined as:

$$\sigma = \sqrt{\frac{\sum_{i \in N} c_i (d_i - \mu)^2}{\sum_{i \in N} c_i}}$$

Finally, the instantaneous error (estimate) vector is defined as:

$$\hat{\boldsymbol{\delta}} = \{\hat{\delta}_u, \hat{\delta}_v\} = \sqrt{\boldsymbol{\mu}^2 + \left(\frac{\boldsymbol{\sigma}}{\sqrt{N}}\right)^2}$$

3.6.2 Setup

Packages

[1]:
```
%reload_ext autoreload
%autoreload 2
```

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import os

import pivuq
```

Load images

[3]:
```
parent_path = "./data/particledisparity_code_testdata/"
image_pair = np.array(
    [plt.imread(os.path.join(parent_path + ipath)).astype("float") for ipath in ["B00010.
↪tif", "B00011.tif"]]
)
```

Load reference velocity

[4]:
```
data = np.loadtxt(os.path.join(parent_path + "B00010_UQ.dat"), skiprows=3).T

I, J = 128, 128
X_ref = np.reshape(data[0], (I, J)) - 1  # zero-index
Y_ref = np.reshape(data[1], (I, J)) - 1
U_ref = np.stack((np.reshape(data[2], (I, J)), np.reshape(data[3], (I, J))))
e_ref = np.stack((np.reshape(data[4], (I, J)), np.reshape(data[5], (I, J))))
N_ref = np.reshape(data[6], (I, J))
```

3.6.3 Uncertainity quantificiation using image matching

[5]:
```
%%time
X, Y, delta, N, mu, sigma = pivuq.disparity.sws(
    image_pair,
    U_ref,
    window_size=16,
    grid_size=4,
    window="gaussian",
    radius=1,
    sliding_window_subtraction=True,
    ROI=[10, 450, 220, 430],
    velocity_upsample_kind="linear",
    warp_direction="center",
    warp_order=-1,
    warp_nsteps=1,
)
```

```
CPU times: user 535 ms, sys: 27.3 ms, total: 562 ms
Wall time: 274 ms
```

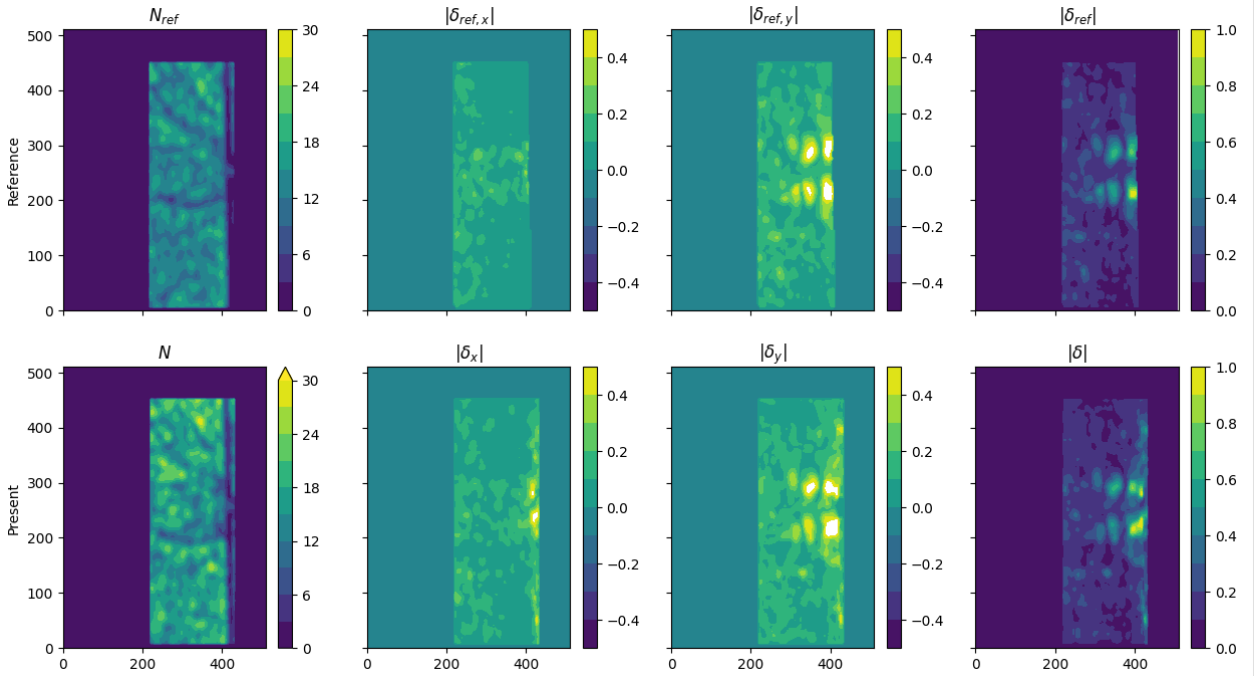
3.6.4 Plot: Instantaneous error map

[6]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, sharex=True, sharey=True, figsize=(15, 8))

# References
ax = axes[0, 0]
im = ax.contourf(X_ref, Y_ref, N_ref, np.linspace(0, 30, 11))
fig.colorbar(im, ax=ax)
ax.set(title="$N_{ref}$", ylabel="Reference")

for i, (ax, var) in enumerate(zip(axes[0, 1:3], ["x", "y"])):
    im = ax.contourf(X_ref, Y_ref, e_ref[i], np.linspace(-0.5, 0.5, 11))
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$|\delta_{{ref,{var}}}|$")

ax = axes[0, 3]
im = ax.contourf(X_ref, Y_ref, np.linalg.norm(e_ref, axis=0), np.linspace(0, 1, 11))
fig.colorbar(im, ax=ax)
ax.set(title="$|\delta_{ref}|$")

# Present
ax = axes[1, 0]
im = ax.contourf(X, Y, N, np.linspace(0, 30, 11), extend="max")
fig.colorbar(im, ax=ax)
ax.set(title="$N$", ylabel="Present")

for i, (ax, var) in enumerate(zip(axes[1, 1:3], ["x", "y"])):
    im = ax.contourf(X, Y, delta[i], np.linspace(-0.5, 0.5, 11))
    fig.colorbar(im, ax=ax)
    ax.set(title=f"$|\delta_{var}|$")
```

(continues on next page)

(continued from previous page)

```
ax = axes[1, 3]
im = ax.contourf(X, Y, np.linalg.norm(delta, axis=0), np.linspace(0, 1, 11))
fig.colorbar(im, ax=ax)
ax.set(title="$|\delta|$");
```



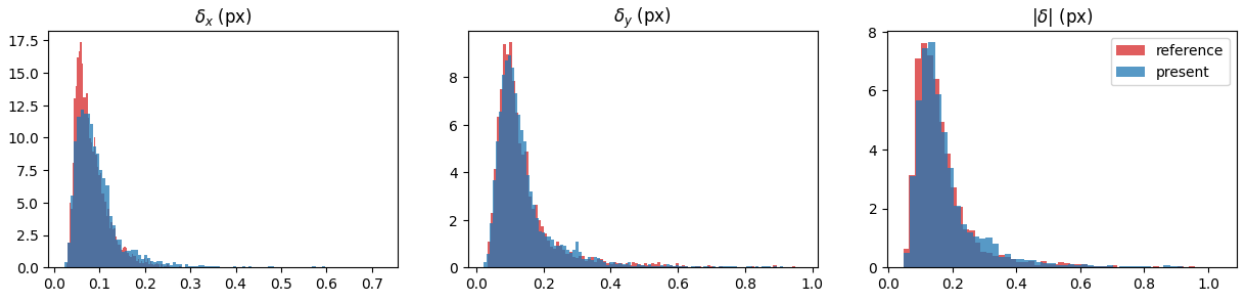
3.6.5 Plot: Instantaneous error histogram

[7]:
```
fig, axes = plt.subplots(ncols=3, figsize=(15, 3))

for i, (ax, label) in enumerate(zip(axes[:2], [r"$\delta_x$ (px)", r"$\delta_y$ (px)"])):
    values = e_ref[i].ravel()
    ax.hist(
        values[np.abs(values) > 0],
        bins=100,
        density=True,
        color="tab:red",
        label="reference",
        alpha=0.75,
    )
    ax.set(title=label)

for i, (ax, label) in enumerate(zip(axes[:2], [r"$\delta_x$ (px)", r"$\delta_y$ (px)"])):
    values = delta[i].ravel()
    ax.hist(
        values[np.abs(values) > 0],
        bins=100,
        density=True,
        color="tab:blue",
```

(continues on next page)

(continued from previous page)

```
        label="present",
        alpha=0.75,
    )
    ax.set(title=label)

ax = axes[-1]
values = np.linalg.norm(e_ref, axis=0).ravel()
ax.hist(
    values[np.abs(values) > 0],
    bins=50,
    density=True,
    color="tab:red",
    label="reference",
    alpha=0.75,
)

values = np.linalg.norm(delta, axis=0).ravel()
ax.hist(
    values[np.abs(values) > 0],
    bins=50,
    density=True,
    color="tab:blue",
    label="present",
    alpha=0.75,
)
ax.set(title="$|\delta|$ (px)")
ax.legend();
```



3.7 Uncertainty Quantification using Image Matching - Cylinder

3.7.1 Description

Based on paper and source code: - Sciacchitano, A., Wieneke, B., & Scarano, F. (2013). PIV uncertainty quantification by image matching. *Measurement Science and Technology*, 24 (4). https://doi.org/10.1088/0957-0233/24/4/045302. - http://piv.de/uncertainty/?page_id=221

Step 1: Particle peak detection

As described by (Sciacchitano et al., 2013, Eq. 1), the image intensity product Π from image matching intensities is

defined as:

$$\Pi = \hat{I}_1 \hat{I}_2$$

The peaks image φ is defined as:

$$\varphi(i,j) = \begin{cases} 1 & \text{if } \Pi(i,j) \text{ is a relative maximum} \\ 0 & \text{otherwise} \end{cases}$$

Step 2: Disparity vector computation

The sub-pixel peak position estimator adopted here is the standard 3-point Gaussian fit. The particle positions of times t_1 and t_2 are defined as $\boldsymbol{X}^1 = \{x_1^1, x_2^1, ..., x_N^1\}$ and $\boldsymbol{X}^2 = \{x_1^2, x_2^2, ..., x_N^2\}$. Discrete disparity vectors are defined as:

$$\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_N\} = \boldsymbol{X}^2 - \boldsymbol{X}^1$$

Step 3: Disparity ensemble statistics inside window

The mean of disparity set inside a window is defined as:

$$\mu = \frac{1}{N} \sum_{i \in N} c_i d_i$$

where $c_i = \sqrt{\Pi(x_i)}$ for $i = 1, 2, ..., N$.

The standard deviation of disparity set inside a window is defined as:

$$\sigma = \sqrt{\frac{\sum_{i \in N} c_i (d_i - \mu)^2}{\sum_{i \in N} c_i}}$$

Finally, the instantaneous error (estimate) vector is defined as:

$$\hat{\boldsymbol{\delta}} = \{\hat{\delta}_u, \hat{\delta}_v\} = \sqrt{\boldsymbol{\mu}^2 + \left(\frac{\boldsymbol{\sigma}}{\sqrt{N}}\right)^2}$$

3.7.2 Setup

Packages

[1]:
```
%reload_ext autoreload
%autoreload 2
```

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import os

import pivuq
```

Load images

[3]:
```
parent_path = "./data/cylinder_wake/"
image_pair = np.array(
    [plt.imread(os.path.join(parent_path + ipath)).astype("float") for ipath in ["frameA.
↪tif", "frameB.tif"]]
)
```

Load reference velocity

[4]:
```
data = np.load(os.path.join(parent_path + "vectors_OF.npz"))

X_ref = data["X"]
Y_ref = data["Y"]
U_ref = data["U"]
```

3.7.3 Uncertainity quantificiation using image matching

Method: ILK

[5]:
```
%%time

X_ilk, Y_ilk, D = pivuq.disparity.ilk(
    image_pair,
    U_ref,
    window_size=16,
    window="gaussian",
    velocity_upsample_kind="linear",
    warp_direction="center",
    warp_order=-1,
    warp_nsteps=1,
)
```

```
CPU times: user 8.57 s, sys: 126 ms, total: 8.7 s
Wall time: 7.71 s
```

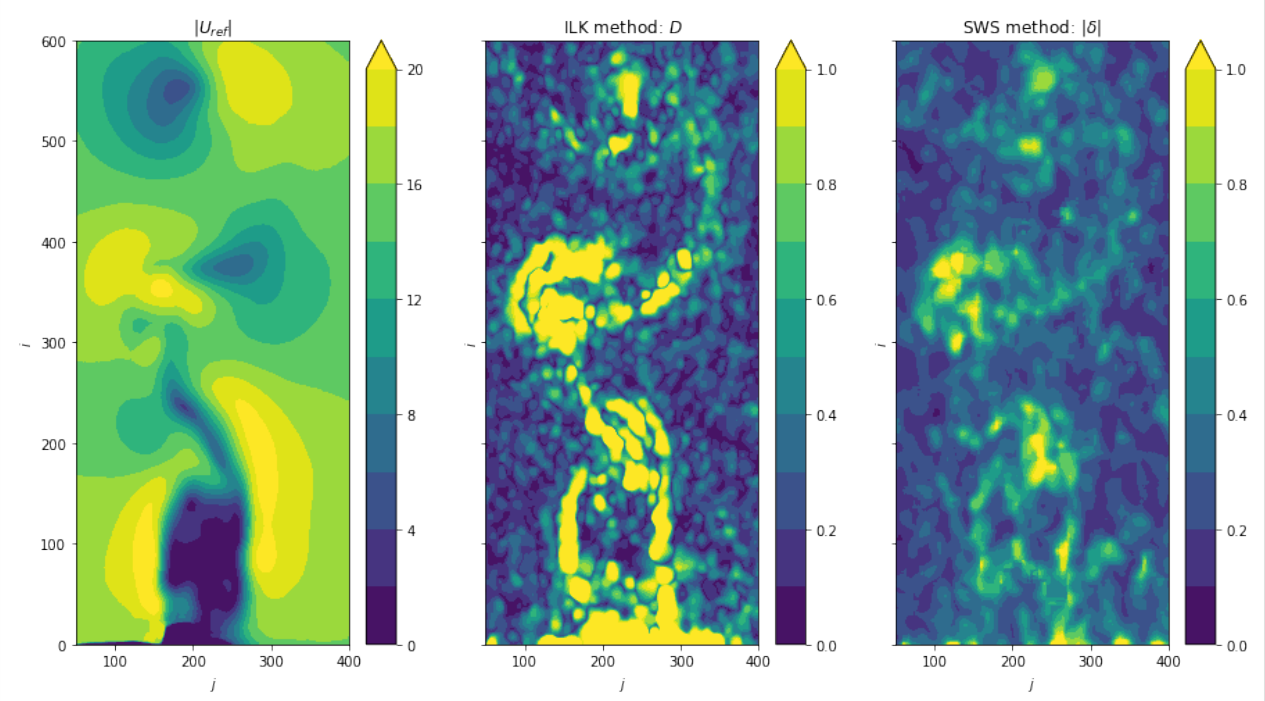
Method: SWS

[6]:
```
%%time
X_sws, Y_sws, delta, N, mu, sigma = pivuq.disparity.sws(
    image_pair,
    U_ref,
    window_size=16,
    window="gaussian",
    radius=1,
    sliding_window_subtraction=True,
    ROI=None,
    velocity_upsample_kind="linear",
    warp_direction="center",
    warp_order=-1,
    warp_nsteps=1,
)
```

```
CPU times: user 15.1 s, sys: 18.6 ms, total: 15.1 s
Wall time: 4.38 s
```

3.7.4 Plot: Instantaneous error map

[7]:
```
fig, axes = plt.subplots(ncols=3, sharex=True, sharey=True, figsize=(15, 8))

# References velocity
ax = axes[0]
im = ax.contourf(X_ref, Y_ref, np.linalg.norm(U_ref, axis=0), np.linspace(0, 20, 11),␣
↪extend="max")
fig.colorbar(im, ax=ax)
ax.set(title="$|U_{ref}|$")

# Uncertainty: ILK
ax = axes[1]
im = ax.contourf(X_ilk, Y_ilk, np.linalg.norm(D, axis=0), np.linspace(0, 1, 11), extend=
↪"max")
fig.colorbar(im, ax=ax)
ax.set(title="ILK method: $D$")

# Uncertainty: SWS
ax = axes[2]
im = ax.contourf(X_sws, Y_sws, np.linalg.norm(delta, axis=0), np.linspace(0, 1, 11),␣
↪extend="max")
fig.colorbar(im, ax=ax)
ax.set(title="SWS method: $|\delta|$")

for ax in axes.ravel():
    ax.set(xlim=(50, 400), ylim=(0, 600), xlabel="$j$", ylabel="$i$")
```



CHAPTER

FOUR

API REFERENCE

4.1 pivuq.disparity

pivuq.disparity.**ilk**(*image_pair*, *U*, *window_size=16*, *prefilter=True*, *window='gaussian'*, *velocity_upsample_kind='linear'*, *warp_direction='center'*, *warp_order=1*, *warp_nsteps=1*)

Disparity map calculation using iterative Lucas Kanade ("ilk").

Parameters

- **image_pair** (np.ndarray) – Image pairs $\mathbf{I} = (I_0, I_1)^\top$ of size (2 x rows x cols).
- **U** (np.ndarray) – Sparse or dense 2D velocity field $\mathbf{U} = (u, v)^\top$ of (2 x U_rows x U_cols).
- **window_size** (int, *default*: 16) – Window size around the pixel to consider the disparity for optical flow estimator.
- **window** ({"gaussian", "tophat"}, *default*: "gaussian") – Windowing kernel type for integration around the pixel.
- **prefilter** (bool, *default*: True) – Whether to prefilter the estimated optical flow before each image warp. When True, a median filter with window size 3 along each axis is applied. This helps to remove potential outliers.
- **velocity_upsample_kind** ({"linear", "cubic", "quintic"}, *default*: "linear") – Velocity upsampling kind for spline interpolation *scipy.interpolation.interp2d*.
- **warp_direction** ({"forward", "center", "backward"}, *default*: "center") – Warping direction.
- **warp_order** (1-5, *default*: 1) – The order of interpolation for *skimage.transform.warp*.
- **warp_nsteps** (int, *default*: 5) – Number of sub-steps to use for warping to improve accuracy.

Returns

- **X, Y** (np.ndarray) – x and y coordinates of disparity map.
- **D** (np.ndarray) – pixel-wise 2D disparity map $\mathbf{D} = (d_x, d_y)^\top$ of size (2 x rows x cols).

See also:

skimage.registration.optical_flow_ilk
Coarse to fine optical flow estimator.

skimage.transform.warp
Warp an image according to a given coordinate transformation.

pivuq.disparity.**sws**(*image_pair*, *U*, *window_size=16*, *grid_size=4*, *window='gaussian'*, *radius=1*, *sliding_window_subtraction=True*, *ROI=None*, *velocity_upsample_kind='linear'*, *warp_direction='center'*, *warp_order=1*, *warp_nsteps=1*)

Python implementation of *Sciacchitano-Wieneke-Scarano* algorithm of PIV Uncertainty Quantification by image matching[1].

Parameters

- **image_pair** (np.ndarray) – Image pairs $\mathbf{I} = (I_0, I_1)^\top$ of size (2 x rows x cols).
- **U** (np.ndarray) – Sparse or dense 2D velocity field $\mathbf{U} = (u, v)^\top$ of (2 x U_rows x U_cols).
- **window_size** (int, *default*: 16) – Window size around the pixel to consider the disparity ensemble.
- **grid_size** (int, *default*: 4) – Disparity ensemble grid resolution in pixels.
- **window** ({"gaussian", "tophat"}, *default*: "gaussian") – Window type for the disparity statistics.
- **radius** (int, *default*: 1) – Search radius for particle peak position.
- **sliding_window_subtraction** (bool, *default*: False) – Whether to use the sliding window subtraction before disparity vector calculation.
- **ROI** (tuple, *default*: None) – Region of interest to use for calculating the disparity ensemble (*i_min*, *i_max*, *j_min*, *j_max*).
- **velocity_upsample_kind** ({"linear", "cubic", "quintic"}, *default*: "linear") – Velocity upsampling kind for spline interpolation *scipy.interpolation.interp2d*.
- **warp_direction** ({"forward", "center", "backward"}, *default*: "center") – Warping direction.
- **warp_order** (1-5, *default*: 1) – The order of interpolation for *skimage.transform.warp*.
- **warp_nsteps** (int, *default*: 5) – Number of sub-steps to use for warping to improve accuracy.

Returns

- **X, Y** (np.ndarray) – *x* and *y* coordinates of disparity map.
- **delta** (np.ndarray) – Instantaneous error estimation map of size $2 \times N \times M$ defined by Eq. (3)[1].
- **N** (np.ndarray) – Number of peaks inside the window.
- **mu** (np.ndarray) – Mean disparity map of size $2 \times N \times M$ defined by Eq. (3)[1].
- **sigma** (np.ndarray) – Standard deviation disparity map of size $2 \times N \times M$ defined by Eq. (3)[1].

[1] Sciacchitano, A., Wieneke, B., & Scarano, F. (2013). PIV uncertainty quantification by image matching. Measurement Science and Technology, 24 (4). https://doi.org/10.1088/0957-0233/24/4/045302

References

4.2 pivuq.lib

pivuq.lib.**construct_subpixel_position_map**(*im*)

Construct sub-pixel position map based on 3-point Gaussian fit stencil.

Parameters
im (np.ndarray) – Image of size $N \times M$.

Returns
X_sub, Y_sub – Subpixel position map of size $N \times M$.

Return type
np.ndarray

pivuq.lib.**disparity_ensemble_statistics**(*D*, *c*, *weights*, *wr*, *grid_size*, *coeff*, *ROI*)

Numba accelerated loop for computing the disparity statistics inside a window of radius *wr*.

Parameters

- **D** (np.ndarray) – Disparity map D of size $2 \times N \times M$ defined by Eq. (2)[1].
- **c** (np.ndarray) – Disparity weight map c of size $N \times M$ defined by Eq. (3)[Page 37, 1].
- **weights** (np.ndarray) – Windowing weights of size $N \times M$ defined by Gaussian or tophat filter.
- **wr** (int) – Window radius.
- **ws** (int) – Disparity resolution size.
- **coeff** (float) – Confidence interval coefficient.
- **ROI** (tuple) – Row and column indices of the ROI: (*i_min*, *i_max*, *j_min*, *j_max*).

Returns

- **delta** (np.ndarray) – Instantaneous error estimation map of size $2 \times N \times M$ defined by Eq. (3)[1].
- **N** (np.ndarray) – Number of peaks inside the window.
- **mu** (np.ndarray) – Mean disparity map of size $2 \times N \times M$ defined by Eq. (3)[1].
- **sigma** (np.ndarray) – Standard deviation disparity map of size $2 \times N \times M$ defined by Eq. (3)[1].

References

pivuq.lib.**disparity_vector_computation**(*warped_image_pair*, *radius=2.0*, *sliding_window_size=16*)

Python implementation of *Sciacchitano-Wieneke-Scarano* disparity vector computation algorithm for PIV Uncertainty Quantification by image matching[1].

Parameters

- **warped_image_pair** (np.ndarray) – Warped image pair $\hat{\mathbf{I}} = (\hat{I}_0, \hat{I}_1)^\top$ of size $2 \times N \times M$.

[1] Sciacchitano, A., Wieneke, B., & Scarano, F. (2013). PIV uncertainty quantification by image matching. Measurement Science and Technology, 24 (4). https://doi.org/10.1088/0957-0233/24/4/045302.

- **radius** (int, *default*: 2) – Discrete particle position search radius from the centroid defined by φ.
- **sliding_window_size** (int, *default*: 16) – Sliding window average subtraction window size.

Returns

- **D** (np.ndarray) – Disparity map D of size $2 \times N \times M$ defined by Eq. (2).
- **c** (np.ndarray) – Disparity weight map c of size $N \times M$ defined by Eq. (3).

pivuq.lib.**find_particle**(*im*, *ic*, *jc*, *radius=1*)

Particle peak position finder around the radius of centroid.

Parameters

- **im** (np.ndarray) – Image array of size $N \times M$.
- **ic** (int) – Row and column index of centroid.
- **jc** (int) – Row and column index of centroid.
- **radius** (int) – Search radius of centroid.

Returns
Row index and column index of peak.

Return type
int, int

pivuq.lib.**find_peaks**(*imgPI*) → ndarray

Particle peak position detection according to Eq. (1)[1].

Parameters
imgPI (np.ndarray) – Image intensity product Π of size $N \times M$.

Returns
Peak map φ of size $N \times M$.

Return type
np.ndarray

pivuq.lib.**sliding_avg_subtract**(*im*, *window_size*) → ndarray

Perform sliding window average subtraction.

Parameters

- **im** (np.ndarray) – Image of size $N \times M$.
- **window_size** (int) – Window size.

Returns
Average subtracted image of size $N \times M$.

Return type
np.ndarray

4.3 pivuq.warping

pivuq.warping.**interpolate_to_pixel**(*U*, *imshape*, *kind='linear'*) → ndarray

Interpolate velocity field to pixel level.

Parameters

- **U** (np.ndarray) – Sparse 2D velocity field.
- **imshape** (tuple) – Image frame dimension (rows, cols).
- **kind** ({"linear", "cubic", "quintic"}, *default*: "linear") – The kind of spline interpolation for *scipy.interpolation.interp2d*.

Returns

Pixel-wise 2D velocity field.

Return type

np.ndarray

pivuq.warping.**warp**(*image_pair*, *U*, *velocity_upsample_kind='linear'*, *direction='center'*, *nsteps=1*, *order=-1*, *radius=2*) → ndarray

Warp image pair pixel-wise to each other using *skimage.transform.warp*.

Parameters

- **image_pair** (np.ndarray) – Image pairs $\mathbf{I} = (I_0, I_1)^\top$ of size $2 \times N \times M$.
- **U** (np.ndarray) – Sparse or dense 2D velocity field $\mathbf{U} = (u, v)^\top$ of size $2 \times U_N \times U_M$.
- **warp_direction** ({"forward", "center", "backward"}, *default*: "center") – Warping warp_direction.
- **velocity_upsample_kind** ({"linear", "cubic", "quintic"}, *default*: "linear") – Velocity upsampling kind for spline interpolation *scipy.interpolation.interp2d*.
- **nsteps** (int, *default*: 1) – Number of sub-steps to use for warping to improve accuracy. Although, the original flow estimator (e.g. PIV) most likely uses $n_{\mathrm{steps}} = 1$.
- **order** (-1, 1, 2 (bug), 3, *default*: -1) – The order of interpolation for *skimage.transform.warp*. If order is negative, using *Whittaker-Shannon interpolation*.
- **radius** (int, *default*: 3) – Radius of the Whittaker-Shannon interpolation stencil.

Returns

Warped image pair $\hat{\mathbf{I}} = (\hat{I}_0, \hat{I}_1)^\top$ of size $2 \times N \times M$.

Return type

np.ndarray

pivuq.warping.**warp_skimage**(*frame*, *U*, *coords*, *order=1*, *mode='edge'*) → ndarray

Warp image frame pixel-wise using *skimage.transform.warp*.

Parameters

- **frame** (np.ndarray) – image frame.
- **U** (np.ndarray) – pixel-wise 2D velocity field.
- **coords** (np.ndarray) – 2D image coordinates (row, cols).
- **order** (1-5, *default*: 1) – The order of interpolation for *skimage.transform.warp*.

- **mode** ({"constant", "edge", "symmetric", "reflect", "wrap"}, *default*: "edge") – Points outside the boundaries of the input are filled according to the given mode.

Returns
Warped image frame

Return type
np.ndarray

See also:

skimage.transform.warp
Warp an image according to a given coordinate transformation.

pivuq.warping.**warp_whittaker**(*frame*, *U*, *coords*, *radius=3*) → ndarray

Warp image using Whittaker-Shannon interpolation

Parameters

- **frame** (np.ndarray) – image frame.
- **U** (np.ndarray) – pixel-wise 2D velocity field.
- **coords** (np.ndarray) – 2D image coordinates (row, cols).
- **radius** (int, *default*: 3) – Radius of the interpolation stencil.

Returns
Warped image frame

Return type
np.ndarray

See also:

whittaker_interpolation
Whittaker-Shannon interpolation.

pivuq.warping.**whittaker_interpolation**(*im*, *xi*, *yi*, *r=3*)

Whittaker-Shannon interpolation[1].

Parameters

- **im** (np.ndarray) – Image frame of shape (rows, cols).
- **xi** (np.ndarray) – The *x* and *y*-coordinates at which to evaluate the interpolated values of shape (rows, cols).
- **yi** (np.ndarray) – The *x* and *y*-coordinates at which to evaluate the interpolated values of shape (rows, cols).
- **r** (int, *default*: 3) – Radius of the interpolation stencil.

Returns
Interpolated image frame.

Return type
np.ndarray

[1] Wikipedia contributors. (2022, March 18). Whittaker-Shannon interpolation formula. In Wikipedia, The Free Encyclopedia. Retrieved 12:34, April 20, 2022, from https://en.wikipedia.org/w/index.php?title=Whittaker%E2%80%93Shannon_interpolation_formula&oldid=1077909297

References

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX