OpenPIV mod ReadMe

Leanne Friedrich

November 2019

1 Overview of computational approach

This code was developed to analyze fluid flows during direct ink writing with acoustophoresis and yield stress fluid support.[1, 2, 3] A square capillary was used as a 3D printing nozzle. Ink consisting of a shear thinning polymer resin and metallic microparticles flows through the square capillary. Particles were acoustically focused to the center of the nozzle as described in Refs.[1, 2, 3]. Inks were printed onto a moving glass slide which contained a transparent Carbopol-based support material. Videos were collected from underneath the nozzle through the glass substrate (Fig. 1). Single-layer, three-pass polygons containing a narrow distribution of particles in the center of each filament were printed. Several polygons were printed in each video.

For videos of the region around the print nozzle, this computational approach has the following objectives:

- Tabulate metadata about experimental conditions for each frame, including the polygon shape and size, printing speed, ink composition, edge number, pass number, etc.
- Use PIV to collect transverse flow velocities from specified regions around the print nozzle for each pair of frames.
- Use digital image analysis to collect particle distribution widths for each printed line ahead of and behind the nozzle relative to the print path for each frame.

This code builds on OpenPIV[4] to achieve the aforementioned objectives.

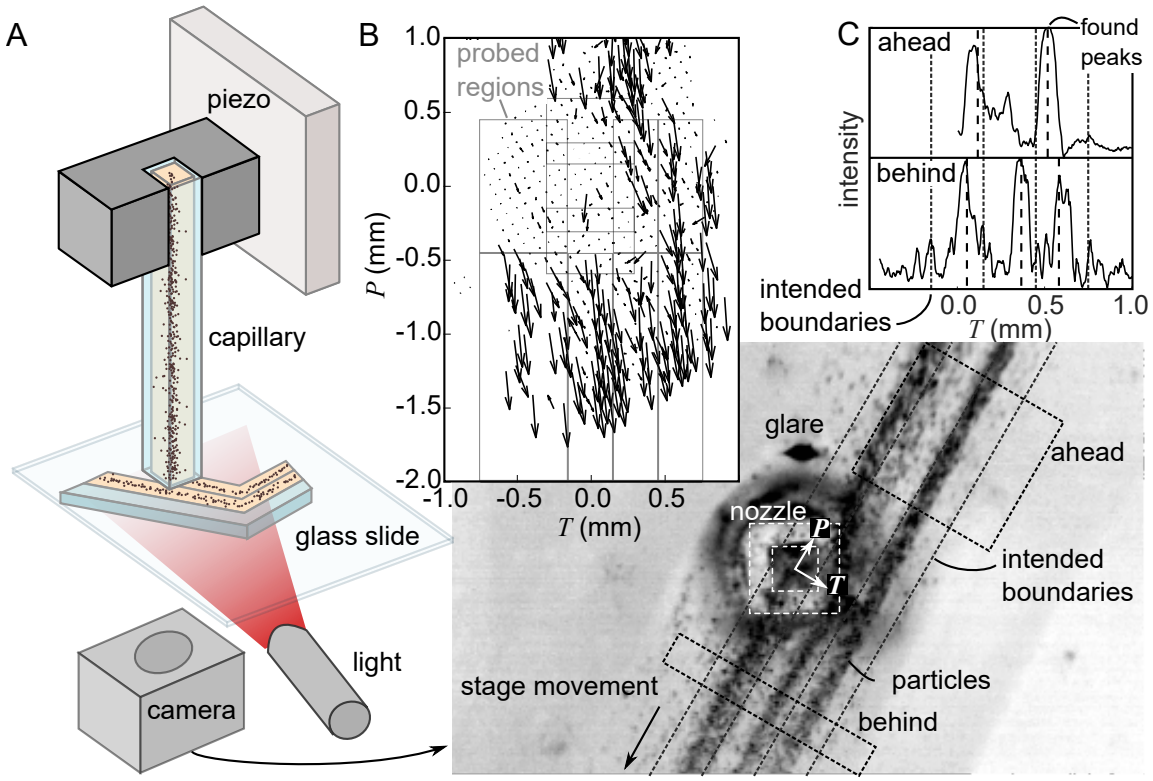

Figure 1: A) Schematic of video collection setup and example of analyzed frame. Image is inverted. B) Fluid displacement field is measured using PIV and rotated into the *P*arallel-*T*ransverse coordinate system. Probed regions are outlined in white. C) Intensities are summed along the parallel direction ahead of and behind the nozzle. Distribution positions and widths are found as a function of transverse position.

2 Functions

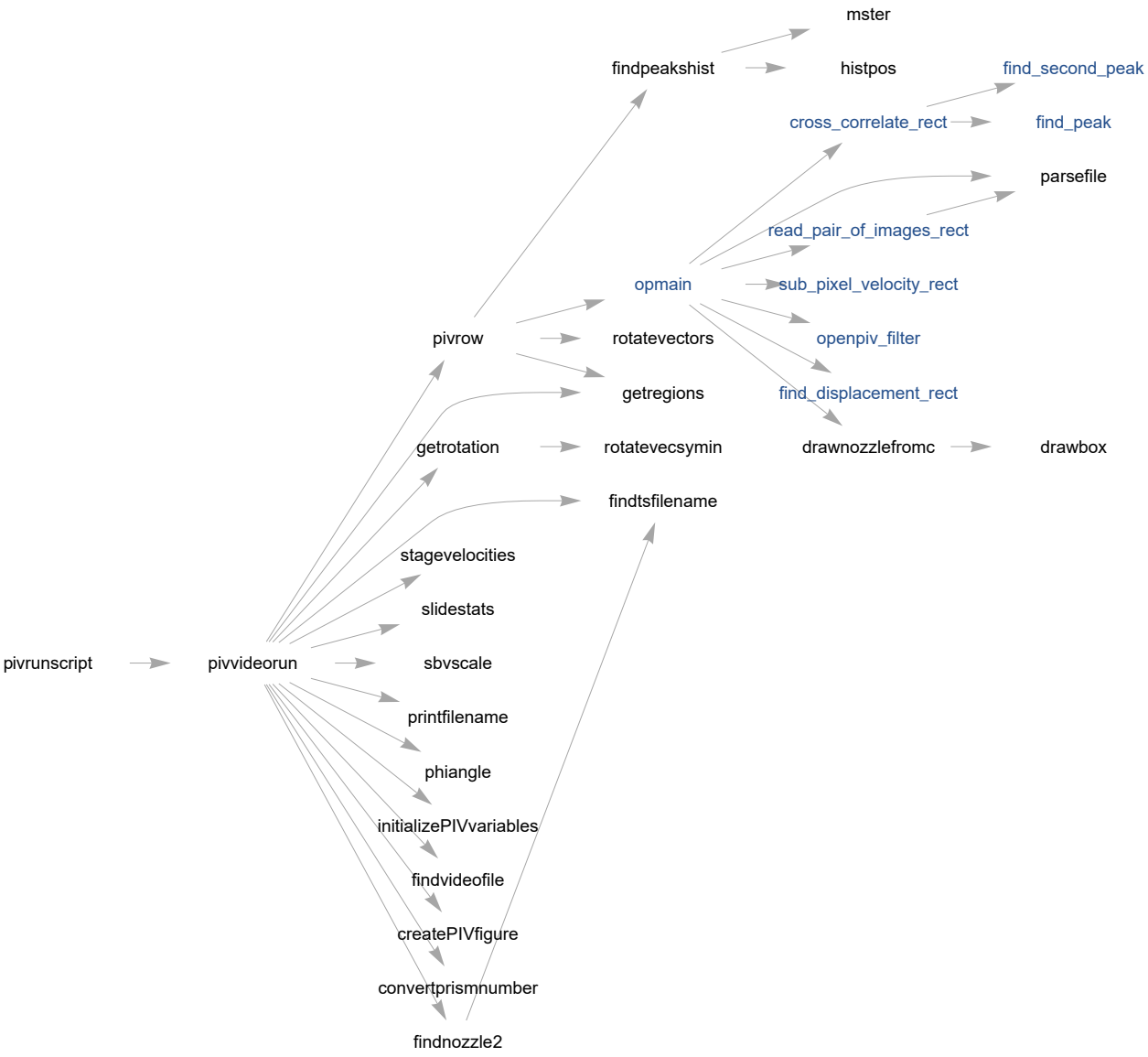

Figure 2: Graph of function calls. Workflow starts with the script `pivrunscript`. Black functions are new to this code set, while blue functions originated in OpenPIV.

Blue functions are already in OpenPIV. For information of functions already in OpenPIV, see www.openpiv.net.[4]

2.1 `convertprismnumber`

Several polygons are printed on each slide. This function outputs the size, shape, and corner angle of the requested prism number on the requested slide number.

Called by: `pivvideorun`

2.2 `createPIVfigure`

Creates a MATLAB figure to track the PIV and particle distribution detection process.

Called by: `pivvideorun`

2.3 `drawbox`

Draws a box on the open figure.
Called by: `drawnozzlefromc`

2.4 `drawnozzlefromc`

Draws a nozzle on the open figure.
Called by: `opmain`
Calls: `drawbox`

2.5 `findnozzle2`

Finds the square nozzle by rotating the image and determining the coordinates that line up the nozzle with the rotated nozzle (Fig. 3).
Called by: `pivvideorun`
Calls: `findtsfilename`

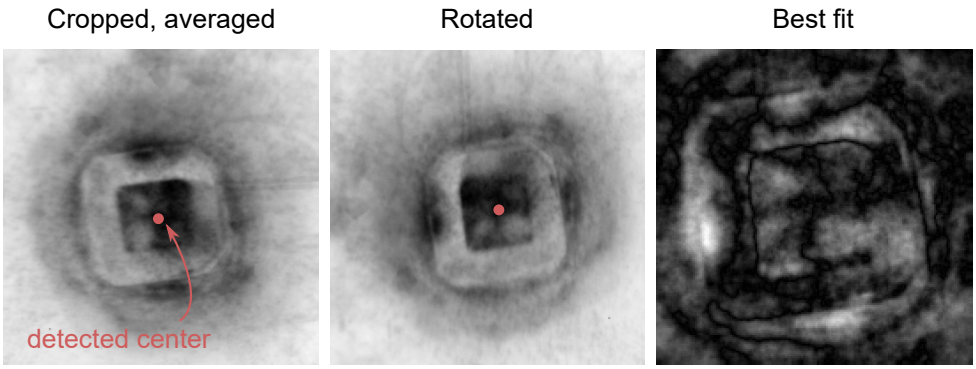

Figure 3: The nozzle is detected by cropping several frames and averaging them together, then rotating that averaged frame, and finding the coordinates which minimize the difference between the two frames.

2.6 `findpeakshist`

Finds locations and widths of peaks in the intensity histogram. Filters peaks to remove glare.
Called by: `pivrow`
Calls: `histpos`, `mster`

2.7 `findtsfilename`

Finds the timestamp file name for a given video file.
Called by: `pivvideorun`, `findnozzle2`

2.8 `findvideofile`

Finds the video file name for a given set of printing parameters.
Called by: `pivvideorun`
Calls: `printfilename`

2.9 getregions

Gets a list of region coordinates in microns.
Called by: pivvideorun, pivrow

2.10 getrotation

Rotates a frame and determines the new coordinates of the nozzle.
Called by: pivvideorun
Calls: rotatevecsymin

2.11 histpos

Crops the image to the region of interest and gets the intensity histogram across the print path.
Called by: findpeakshist

2.12 initializePIVvariables

Creates a struct to hold the input variables for PIV.
Called by: pivvideorun

2.13 mster

Finds the mean and standard error of a distribution.
Called by: findpeakshist

2.14 openpiv_filter

This filters vectors to remove outliers. **Modifications:** Vectors are filtered, but holes are not filled.
Called by: opmain

2.15 opmain

This is the main PIV processing loop used by OpenPIV. For more information, see Taylor, et. al.[4] **Modifications:** In openpiv_filter
Called by: pivrow
Calls: cross_correlate_rect, drawnozzlefromc, find_displacement_rect, openpiv_filter, parsefile, quiverm, read_pair_of_images_rect, sub_pixel_velocity_rect

2.16 parsefile

Imports a frame, converts to gray, and adjusts contrast.
Called by: opmain, read_pair_of_images_rect

2.17 phiangle

Gets the angle ϕ, which is the angle between the printing direction and the printer x-axis.
Called by: pivvideorun

2.18 pivrow

Gets flow field with PIV and particle distribution peaks with digital image analysis.
Called by: pivvideorun
Calls: opmain, getregions, rotatevectors, findpeakshist

2.19 pivrunscript (Topline parent function)

Runs through the files in the data folder `datafolder`, given PIV ROI variables `isize`, `overlap`, and `picchunk`. Looks for files constructed from the inputs in the lists `type`, `ink`, `v`, and `slide`.

Calls: `pivvideorun`

2.20 pivvideorun

Runs through a set of frames in each video, where each video consists of several polygons which have manually determined time stamps. Frames are evenly collected from each edge of each pass of each polygon.

Called by: `pivrunscript`

Calls: `convertprismnumber`, `createPIVfigure`, `findnozzle2`, `findtsfilename`, `findvideofile`, `getregions`, `getrotation`, `initializePIVvariables`, `phiangle`, `pivrow`, `printfilename`, `sbvscale`, `slidestats`, `stagevelocities`

2.21 printfilename

Prints the given file name.

Called by: `pivvideorun`

2.22 read_pair_of_images_rect

Imports frames and crops them. **Modifications:** Import procedure is modified to use `parsefile`.

Called by: `opmain`

Calls: `parsefile`

2.23 rotatevectors

Rotates flow field so that the print path is vertical, extending from the nozzle to the bottom of the image.

Called by: `pivrow`

Calls: `drawnozzle`, `drawnozzlefromc`, `quiverm`

2.24 rotatevecsymin

After a rotation, an empty triangle fills the bottom of the image. If we try to collect particle distributions from inside that triangle, particle distributions will be skewed to one side since only one side will contain part of the image. As such, we need to crop the bottom of the image to avoid this skewing. This function finds the new bottom of the interrogation region.

Called by: `getrotation`

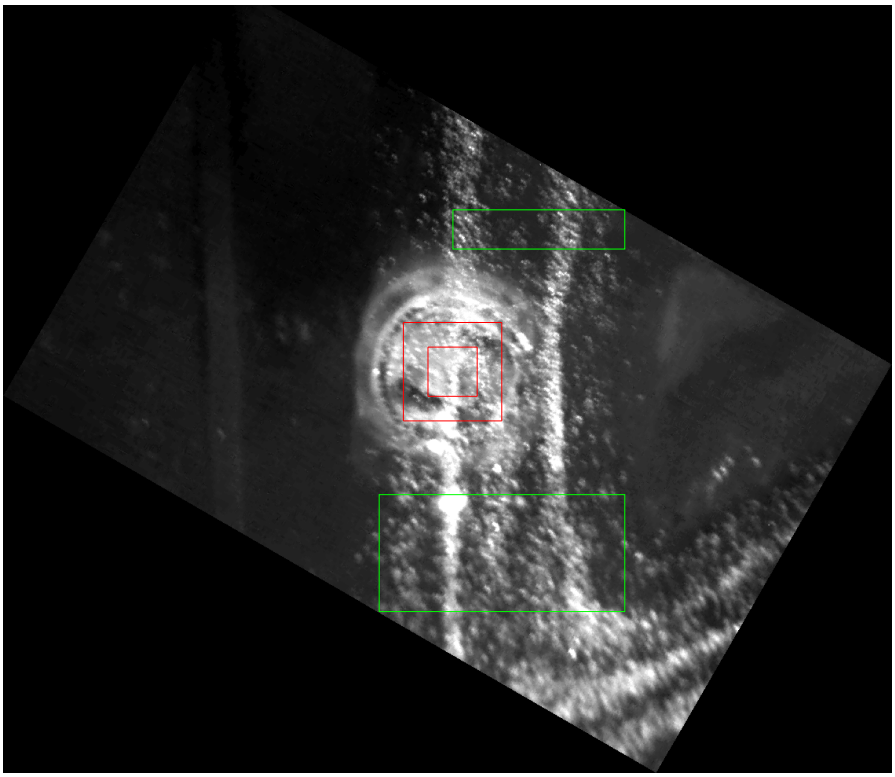

Figure 4: An image is rotated so the print path is below the nozzle. Green boxes indicate the interrogation regions for histogram collection. Red boxes indicate the nozzle position.

2.25 `sbvscale`

This is the scale of the image, in meters per pixel.

Called by: `pivvideorun`

2.26 `slidestats`

Dimensions and coordinates of the polygons on each slide.

Called by: `pivvideorun`

2.27 `stagevelocities`

Finds the velocities in x-y coordinates of each edge of the polygon.

Called by: `pivvideorun`

References

[1] L. Friedrich and M. Begley, "Printing direction dependent microstructures in direct ink writing," *Submitted for publication*, 2019.

[2] L. Friedrich and M. Begley, "Changes in filament microstructures during direct ink writing with yield stress fluid support," *Submitted for publication*, 2019.

[3] L. Friedrich and M. Begley, "Corner accuracy in direct ink writing with support material," *Submitted for publication*, 2019.

[4] Z. J. Taylor, R. Gurka, G. A. Kopp, and A. Liberzon, "Long-duration time-resolved PIV to study unsteady aerodynamics," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 12, pp. 3262–3269, 2010.