Title
Meso-Scale Particle Image Velocimetry Studies of Neurovascular Flows In Vitro

Permalink
https://escholarship.org/uc/item/7dq09764

Journal
Journal of Visualized Experiments

ISSN
1940-087X

Authors
Peck, Ryan A
Bahena, Edver
Jahan, Reza
et al.

Publication Date
2018-12-03

DOI
10.3791/58902

Peer reviewed
Meso-Scale Particle Image Velocimetry Studies of Neurovascular Flows In Vitro

Ryan A. Peck¹, Edver Bahena¹, Reza Jahan², Guillermo Aguilar¹,³,⁴, Hideaki Tsutsui¹,⁴, Marko Princevac¹, Monica M. Wilhelmus¹, Masaru P. Rao¹,³,⁴

¹Department of Mechanical Engineering, University of California, Riverside
²Division of Interventional Neuroradiology, University of California, Los Angeles
³Materials Science and Engineering Program, University of California, Riverside
⁴Department of Bioengineering, University of California, Riverside

These authors contributed equally

Correspondence to: Masaru P. Rao at mprao@engr.ucr.edu

URL: https://www.jove.com/video/58902
DOI: 10.3791/58902

Keywords: Bioengineering, Issue 142, Particle image velocimetry, PDMS tissue phantom, 3-D printing, fluid mechanics, signal processing, neurovascular

Date Published: 12/3/2018


Abstract

Particle image velocimetry (PIV) is used in a wide variety of fields, due to the opportunity it provides for precisely visualizing and quantifying flows across a large spatiotemporal range. However, its implementation typically requires the use of expensive and specialized instrumentation, which limits its broader utility. Moreover, within the field of bioengineering, in vitro flow visualization studies are also often further limited by the high cost of commercially sourced tissue phantoms that recapitulate desired anatomical structures, particularly for those that span the mesoscale regime (i.e., submillimeter to millimeter length scales). Herein, we present a simplified experimental protocol developed to address these limitations, the key elements of which include 1) a relatively low-cost method for fabricating mesoscale tissue phantoms using 3-D printing and silicone casting, and 2) an open-source image analysis and processing framework that reduces the demand upon the instrumentation for measuring mesoscale flows (i.e., velocities up to tens of millimeters/second). Collectively, this lowers the barrier to entry for nonexperts, by leveraging resources already at the disposal of many bioengineering researchers. We demonstrate the applicability of this protocol within the context of neurovascular flow characterization; however, it is expected to be relevant to a broader range of mesoscale applications in bioengineering and beyond.

Video Link

The video component of this article can be found at https://www.jove.com/video/58902/

Introduction

PIV is widely used in experimental fluid mechanics for flow visualization and quantitative investigations of fluid motion that vary in length scale from atmospheric to microcirculatory flows¹,²,³. While the specifics of its implementation can vary as widely as its applications, one aspect common to nearly all PIV studies is the use of video imaging of tracer particles seeded within the working fluid, followed by a pair-wise analysis of consecutive image frames to extract desired flow characteristics. Typically, this is accomplished by first subdividing each image frame into smaller regions termed interrogation windows. As a consequence of the random positions of the dispersed particles, each interrogation window contains a unique distribution of pixel intensities. If the window size and data acquisition rate are chosen appropriately, cross-correlation of the intensity signal in each window can be used to estimate the average displacement within that region. Finally, given that the magnification and frame rate are known experimental parameters, an instantaneous velocity vector field can be readily computed.

A major advantage of PIV over single-point measurement techniques is its ability to map vector fields across a two- or three-dimensional domain. Hemodynamic applications, in particular, have benefited from this capability, since it allows a thorough investigation of local flows, which are known to play a significant role in vascular disease or remodeling (e.g., atherosclerosis, angiogenesis)⁴,⁵,⁶. This has also been true for the evaluation of neurovascular flows, and the interactions thereof with endovascular devices (e.g., flow diveters, stents, intrasaccular coils), since the relevant length-scales in such applications can often span one or more orders of magnitude (e.g., from micrometer to millimeter), and device geometry and placement can significantly impact the local fluid mechanics⁷.

Most groups conducting PIV-based hemodynamic studies have relied on experimental set-ups that closely mimic some of the earliest investigations of stent influence on vascular flow⁸,⁹. Typically, these include a) pulsed lasers and high-speed cameras, to capture high-velocity flows; b) synchronizers, to prevent aliasing between the pulse frequency of the laser and the camera acquisition frame rate; c) cylindrical optics, to form a light sheet and, thus, minimize the background fluorescence from tracer particles above and below the interrogation plane; d) in the case of commercial turn-key systems, proprietary software packages, to perform the cross-correlation analyses. However, while some applications require the performance and/or versatility collectively afforded by these components, many others do not. Moreover, the high cost of...
commercially sourced tissue phantoms that recapitulate desired vascular structures can also prove limiting for many in vitro studies, particularly for phantoms with features that bridge the mesoscale regime (> 500 USD/phantom). Herein, we report the development of a simplified protocol for implementing PIV for the in vitro visualization of neurovascular flows, which typically lie both spatially and temporally within the mesoscale regime (i.e., length scales ranging from submillimeter to millimeter, and velocities up to tens of millimeters/second). The protocol seeks to leverage resources already at the disposal of many bioengineering researchers, thus lowering the barrier to entry for nonexperts.

The first element of this protocol involves the use of an investment casting technique to enable the in-house fabrication of transparent, polydimethylsiloxane (PDMS)-based tissue phantoms from 3-D-printed sacrificial molds. By leveraging the increasing availability of 3-D printers in recent years, particularly those in shared/multi-user facilities (e.g., institutional facilities or public makerspaces), this methodology cuts costs significantly (e.g., < 100 USD/phantom in the case presented here), while enabling a rapid turnaround for the fabrication of a wide variety of designs and geometries. In the current protocol, a fused deposition modeling system is used with acrylonitrile butadiene styrene (ABS) as the building material, and the printed part serves as a sacrificial mold for the subsequent phantom casting. Our experience has shown that ABS is well-suited for such use since it is soluble in common solvents (e.g., acetone), and it has sufficient strength and rigidity to maintain mold integrity after the removal of the support material (e.g., to prevent the deformation or fracture of diminutive mold features). In the current protocol, mold integrity is further ensured using solid printed models, although this comes at the expense of increased dissolution time. The use of hollow models may also be possible in some cases, to enhance solvent access, and thus, reduce dissolution time. However, careful consideration should be given to the effect this may have on mold integrity. Finally, while the phantoms fabricated herein are based upon idealized representations of neurovascular structures generated using a common computer-aided design (CAD) software package, the protocol is expected to be amenable to the fabrication of more complex, patient-specific geometries as well (e.g., via the use of model files generated by the conversion of clinical imaging data to the .STL file format used by most 3-D printers). Further details regarding the phantom fabrication process are provided in section 2 of the protocol.

The second element of the protocol involves the use of an open-source plug-in for ImageJ to conduct the cross-correlation analyses. This is coupled with the implementation of a simple statistical thresholding scheme (i.e., intensity capping) to improve the image signal prior to cross-correlation, as well as a postcorrelation vector validation scheme, the normalized median test (NMT), to eliminate spurious vectors through a comparison of each to its nearest neighbors. Collectively, this allows imaging to be accomplished using equipment commonly found in many bioengineering laboratories, thus eliminating the need for the acquisition of many of the costly components of typical PIV systems (e.g., pulsed laser, synchronizer, cylindrical optics, and proprietary software). Further details regarding the video collection, image processing, and data analysis are provided in sections 5 and 6 of the protocol.

Figure 1 illustrates the PIV set-up used in this protocol, which relies upon a fluorescence microscope equipped with a high-speed camera for imaging, as well as an external, continuous white-light source (i.e., metal halide lamp) for through-objective volumetric illumination. A variable-speed gear pump is used to impose the recirculating flow of a transparent mock blood solution through the neurovascular tissue phantoms. The solution is composed of a 60:40 mixture of deionized (DI) water and glycerol, which is a common substitute for blood in hemodynamic studies, due to a) its similar density and viscosity (i.e., 1,080 kg/m$^3$ vs. 1,050 kg/m$^3$ and 3.5 cP vs. 3 - 5 cP for blood), b) its transparency in the visible range; c) its similar refractive index as PDMS (1.38 vs. 1.42 for PDMS), which minimizes optical distortion; d) the ease with which non-Newtonian behavior can be introduced, if needed, via the addition of xanthane. Finally, fluorescent polystyrene beads are used as tracer particles (10.3 µm in diameter; 480 nm/501 nm excitation/emission). While neutrally buoyant beads are desired, sourcing tracer particles with optimal fluid mechanical properties (e.g., density, size, composition) and emission wavelength can prove challenging. For example, the beads used herein are slightly less dense than the glycerol solution (1,050 kg/m$^3$ vs. 1,080 kg/m$^3$). However, the hydrodynamic effects, thereof, are negligible, given that the duration of a typical experiment is far shorter than the time scale associated with buoyancy effects (i.e., 5 min and 20 min, respectively). Further details regarding the mock blood solution formulation and in vitro circulatory system set-up are provided in sections 3 and 4 of the protocol.

**Protocol**

1. **ABS-based Sacrificial Mold Fabrication**
   1. Design an inverse model of the desired tissue phantom using CAD software.
   2. Print the model using a 3-D printer with ABS as the building material.

2. **PDMS-based Vascular Phantom Fabrication**
   1. **Mixing**
      1. Mix the PDMS prepolymer base and curing agent in a 10:1 ratio (by weight); a 66 g mixture provides sufficient material for the fabrication of phantoms with volumes up to 50 cm$^3$.
      2. Place the mixture in a vacuum desiccator for 60 min to degas and minimize the bubble entrapment. Use cyclic pressurization/depressurization to facilitate bubble rupture.
   2. **Casting**
      1. Mount the printed ABS mold on a glass slide using molding putty to seal the interface.
      2. Carefully pour the PDMS mixture into the mold while trying to minimize bubble entrapment. Lingering bubbles can be manually ruptured using a needle.
      3. Cure the cast phantom at room temperature (25 °C) for at least 24 h.
         **NOTE:** At higher temperatures, this process can be accelerated.
   3. **Demolding**
      1. Dissolve the ABS by submerging the phantom in acetone and sonicating for at least 15 min, using powers up to 70 W.
CAUTION: Acetone has a high vapor pressure at room temperature and a low flash point. Consequently, always work under a fume hood and away from potential ignition sources. Wear proper personal protective equipment (e.g., goggles or face shield, lab coat, acetone-resistant gloves).

2. Thoroughly rinse the phantom with isopropyl alcohol and, then, DI water to remove solvent residues.
   NOTE: PDMS swells upon exposure to acetone; however, the swelling subsides once the phantom is rinsed and dried sufficiently.

4. Confirmation of phantom fidelity using optical microscopy
   1. Using an optical microscope with an attached camera and image capture software, capture an image of a critical feature within the phantom under a magnification that maximizes the feature within the field of view.
   2. Capture an image of an appropriate calibration reticle at the same magnification.
   3. Load both images into ImageJ by dragging them onto the Toolbar.
   4. Click on the calibration reticle image to make it active and, then, select the Line tool. Using the mouse, draw a line along a feature of a known distance and select Analyze > Set Scale from the ImageJ menu.
   NOTE: In the Set Scale window, the field labeled Distance in pixels should be prepopulated with the length of the drawn line in units of pixels.
   5. Enter the length of the feature in the field labeled Known Distance, and its unit in the field labeled Unit of Length. Check the box labeled Global to apply this calibration factor to all open images.
   6. Make the image of the phantom critical feature active and use the Line tool to draw a line along a feature of interest. From the ImageJ menu, select Analyze > Measure (or press Ctrl + M) to measure the length of the line.
   7. Compare the expected value against the value in the column marked Length in the Results window to confirm phantom fidelity.

3. Mock Blood Solution Formulation
   1. Mix DI water and glycerol in a 60:40 ratio (by volume).
      NOTE: A 100 mL volume is sufficient for the in vitro circulatory system described herein.
   2. Add 1 mL of 2.5% w/v fluorescent polystyrene bead solution (i.e., tracer particles) to the mock blood solution.
   3. Homogenize the mixture on a magnetic stir plate at 400 rpm for 10 min.

4. In Vitro Circulatory System Set-up
   1. Pump set-up
      1. Use a wire stripper tool to cut off the DC-end plug from the AC-to-DC adapter power source.
      2. Strip the coating off the power and ground wires and connect them to the input terminal of the pulse width modulation (PWM) voltage regulator.
      3. Connect the power and ground wires from the pump's DC motor to the output terminal of the PWM voltage regulator.
      NOTE: The PWM's seven-segment display outputs the duty cycle (0% - 100%) used to achieve a variable voltage to the DC motor.
   2. Pump calibration
      1. Prepare 200 mL of mock blood solution (see section 3).
      2. Place tubing from the pump inlet to the beaker holding the mock blood solution.
      3. Place tubing from the pump outlet to an empty beaker.
      4. Select a desired duty cycle set point (0% - 100%). Press the On button and start a timer.
      5. Stop the timer once the pump has transferred the entire volume of mock blood solution. Use this time to calculate the volumetric flow rate.
      6. Repeat steps 4.2.1 - 4.2.5 for at least five different duty cycle set points to establish a least-squares regression curve.
      NOTE: A minimum of three replicate points per duty cycle set point is recommended. This relationship can be used to correlate the desired flow rate to the required PWM duty cycle.

5. Video Collection
   1. Image calibration
      1. Determine the calibration ratio for the video imaging (see section 2).
   2. Apparatus set-up
      1. Place the PDMS phantom on the stage of the fluorescence microscope.
      2. Connect the phantom to the gear pump and introduce the mock blood solution.
      NOTE: Optionally, prefill the model with ethanol to facilitate full wetting; then, flush and fill it with the mock blood solution. This may be particularly beneficial for models with smaller vessels and/or blind features.
      3. Set the pump motor controller for the desired flow rate based on the pump calibration curve.
      4. Run the pump for 1 - 5 min prior to the experiment to ensure steady-state conditions.
      5. Turn on the external lamp to illuminate the field of view. Select an appropriate filter based on the excitation wavelength of the fluorescent beads.
      6. Adjust the imaging focal plane to the vessel midplane.
      NOTE: This can be achieved by using a focal length that maximizes the imaged vessel cross-section (e.g., when using phantoms with circular vessel cross sections); and/or indexing off of a phantom feature designed to facilitate the identification of the vessel mid-plane.

3. Video recording
1. Select the video recording parameters to optimize the signal-to-noise ratio (SNR). Key parameters include exposure time, frame rate, and gain. 
   NOTE: In this protocol, we use a frame rate of 2,000 fps and a gain of 1.0. However, these parameters may vary based on the application (see the discussion section for further details).
2. Collect the video and save it in AVI format.

4. Phantom clean-up
1. If bead-sticking is observed after an experiment, sonicate the phantom in an aqueous detergent solution using powers up to 70 W.

6. Image Processing and Data Analysis

1. Image preprocessing
   1. Drag the saved AVI file onto the ImageJ window to import it. Select the box marked Convert to Grayscale.
   2. From the ImageJ menu, select Analyze > Generate Histogram (or press Ctrl + H) to generate a histogram of image pixel intensities. Take note of the mean and standard deviation for the unprocessed image.
   NOTE: At high frame rates, it is not unusual for the distribution to be skewed heavily toward zero (i.e., no signal).
   3. From the ImageJ menu, select Image > Adjust > Brightness and Contrast (or press Shift + Ctrl + H) to apply a brightness/contrast filter.
   4. On the Brightness and Contrast menu, press the Set button to define the image limits. Set the minimum value to be the mean value plus one standard deviation, and the maximum value to be the maximum intensity of the image (both based on statistics obtained in step 6.1.2).
   NOTE: This typically eliminates all but the top 10% of the pixel intensities. The number of standard deviations may be varied depending on the desired distribution of the pixel intensities. A custom macro script for performing the intensity capping operation is provided in the Supplemental Materials.
   5. From the ImageJ menu, select Process > Noise > Despeckle to reduce the number of saturated pixels.
   NOTE: This operation is necessitated by the increased potential for pixel saturation that arises during the optimization of the brightness and contrast, which can produce spurious vectors during subsequent cross-correlation.
   6. From the ImageJ menu, select Process > Filters > Gaussian Blur with a radius of 1.5 to reduce artifacts arising from the occasional removal of illuminated pixels in a 3 x 3 neighborhood by the prior despeckling operation.
   7. Click on the Polygon tool and, then, click on the image to outline the region of interest (ROI).
   8. From the ImageJ menu, select Edit > Clear Outside to remove sensor noise in locations where no signal is expected (e.g., areas beyond the vessel wall boundary), which can decrease the overall SNR.

2. PIV calculation
   NOTE: This portion of the protocol employs a third-party PIV plug-in for ImageJ, which relies upon Gaussian peak-fitting to enable an estimation of displacement with subpixel accuracy.
   1. From the ImageJ menu, select Plugins > Macros > Run… and navigate to the saved macro Supplemental Code 2.ijjm to cross-correlate successive image pairs.
   NOTE: The macro proceeds as follows. 1) A cross-correlation of the intensity field within consecutive images is first performed to determine the local displacement of advected tracer particles (i.e., the first image pair consists of the first and second images, the second image pair consists of the second and third images, etc.). 2) A two-step multipass evaluation is then performed with initial and final interrogation window sizes of 256 x 256 pixels and 128 x 128 pixels, respectively. Finally, 3) the macro performs a temporal average to further reduce the appearance of spurious vectors.

3. Normalized median test (NMT)
   1. From the ImageJ menu, select Plugins > Macros > Run… and navigate to the saved macro Supplemental Code 3.ijjm to validate the velocity fields via the normalized median test.
   NOTE: The macro proceeds as follows. 1) Each vector in an instantaneous vector field is first compared to its eight nearest neighbors to compute the median value. 2) The array of residual errors is then calculated as the difference between each neighboring vector and the calculated median. 3) The difference between the vector under investigation and the median neighboring vector value is then normalized by the median of the residuals. 4) This is then compared to a threshold value (typically, 0.2 pixels), which can be varied based on a priori knowledge of noise during the image acquisition. Finally, 5) a temporal average of all validated instantaneous vector fields is performed to produce a composite field, as this has been shown to increase the vector field quality.

Representative Results

Figure 2 illustrates the PDMS tissue phantom fabrication process. The phantoms designed herein are intended for the study of flow in idealized wide-necked, saccular, intracranial aneurysms, as well as proximal branching perforator arteries. Important additional design features include 1) a common reservoir that all vessels drain into, to ensure unencumbered fluid egress from the phantom - otherwise, droplet formation may occur at the smaller vessel outlets; 2) a bubble trap, to facilitate bubble removal; 3) an outer cavity wall, to ensure parallelism of the vessel with the horizontal plane, as well as a precise definition of the final phantom slab height, length, and width; 4) the use of a 21 G hypodermic needle shank (620 µm in nominal outer diameter) for the molding of the perforator artery, due to our printer's inability to define such features with sufficient fidelity. Faithful reproduction of all design features is observed throughout.
Representative results of a PIV-based flow characterization performed using the current protocol are presented in Figure 3 and Figure 4. These studies were performed using phantom inlet flow rates of 100 mL/min, data acquisition rates of 2,000 fps, and a temporal averaging over spans of 0.05 s. Figure 3 shows representative image frames within the perforator artery, before and after intensity capping, as well as corresponding surface plots of the 8-bit pixel intensity values. Both demonstrate that intensity capping significantly increases the peak definition above the noise floor (i.e., increases the SNR), which is critical to ensuring accuracy when performing subsequent cross-correlation. Figure 4 shows the effects of intensity capping and NMT operations on the velocity vector field. Marked improvement in field uniformity is observed, thus further underscoring the importance of maximizing the SNR to minimize data dropout.

Figure 1: Particle image velocimetry set-up. Reliance upon an open-source image analysis and a pre-/postprocessing framework reduces the demand upon the instrumentation for measuring mesoscale flows, thus eliminating the need for many of the costly components of typical PIV systems (e.g., pulsed laser, synchronizer, cylindrical optics, and/or proprietary software). Please click here to view a larger version of this figure.

Figure 2: PDMS-based tissue phantom fabrication process. The images illustrate (a) a CAD model of the neurovascular phantom mold, (b) the printed ABS mold after the removal of the support material, (c) the casting and curing of PDMS within the ABS mold, (d) partial dissolution of ABS mold material, and (e) the completed PDMS phantom, with the inset showing the final dimensions of critical features, as well as the region of interest (ROI) in the perforator artery where the PIV measurements were made. Please click here to view a larger version of this figure.
Figure 3: Effect of the intensity capping operation on the image SNR. These panels show representative image frames and the corresponding pixel intensity surface plots within the perforator artery, (a and b) before and (c and d) after applying the intensity capping operation. Please click here to view a larger version of this figure.

Figure 4: Effects of intensity capping and NMT operations on velocity vector fields. These panels illustrate the representative instantaneous velocity vector field within the perforator artery derived from (a) unprocessed image data, (b) intensity-capped data, and (c) intensity-capped data + NMT postprocessing. Please click here to view a larger version of this figure.

Figure 5: Effect of interrogation window sizing on correlation quality. Optimal window sizing occurs when the value of the zero-normalized correlation coefficient is maximized, and the standard deviation is minimized. Please click here to view a larger version of this figure.

Discussion

The protocol described herein outlines a simplified method for performing PIV studies to visualize neurovascular flows at physiologically relevant dimensions and flow conditions in vitro. In doing so, it serves to complement protocols reported by others that have also focused on simplifying the quantification of vector fields, but within very different contexts that require the consideration of far larger length scales or lower flow rates (e.g., atmospheric or microcirculatory flows), and thus, with a reliance upon schemes that are incompatible with the current application.

The most important considerations for the successful implementation of PIV lie in the minimization of flow field artifacts and the maximization of image quality. Several steps in the tissue phantom fabrication process are critical to both of these criteria. For example, thorough degassing is crucial since air entrained within the PDMS during mixing can lead to bubble formation within the final phantom, which can adversely affect both feature fidelity and optical clarity. Additionally, minimization of surface roughness of the ABS mold is desired, since the PDMS casting process faithfully reproduces even the most minute imperfections (e.g., build lines, surface pores, scratches), thus resulting in surface roughness in the final phantom that can decrease optical clarity and increase the potential for bead accumulation. While the protocol described herein has proven
sufficient for the current application, there are numerous reports in the literature of means of reducing such roughness, should there be any need (e.g., acetone vapor smoothing or the optimization of layer thickness and part orientation with respect to the building direction).

The parameter selection for video capture is also critical to ensure a high-fidelity vector field. An optimal SNR is typically achieved at the highest achievable frame rate that still allows sufficient bead exposure (the maximum frame rate being limited by the minimum exposure time). Gain can be used to amplify the signal, but this also increases sensor noise. If the maximum velocity can be estimated from other flow parameters (e.g., inlet volumetric flow rate), then a lower bound on the required frame rate can be estimated using the following relation:

\[
f_{\text{sampling}} > \frac{v_{\text{max}} \times \text{calibration}}{h_{\text{interrogation\ window}}}
\]

Here, \(f_{\text{sampling}}\) is the camera acquisition rate (Hz), \(v_{\text{max}}\) is the maximum expected velocity (mm/s), \(c_{\text{calibration}}\) is the calibration constant (pixels/mm), and \(h_{\text{interrogation\ window}}\) is the size of the interrogation window (pixels). However, more optimal values can be determined using so-called correlation quality estimation techniques, such as the zero-normalized correlation coefficient \(r\). In this technique, the averages of complementary signals from each frame pair are first subtracted and, then, normalized by the standard deviation of their intensities. If a displacement of the original signal exists, such that all peaks and valleys match, the time-shifted value of this signal will be equal to one. Conversely, if there is no displacement that can align these signals, the value will be zero. This information is included in the ImageJ PIV output for each vector, and it can be plotted as its own field to verify whether there are spatial effects contributing to poor correlation (e.g., uneven lighting). The correlation coefficient can also be averaged over a field as an overall estimate of its quality. Finally, this quantity may also be plotted against varying frame rates or interrogation window sizes to determine an optimum. Figure 5 illustrates the results from such an analysis using a Monte Carlo-synthesized particle field with displacements consistent with our experimentally measured flows (a typical technique for characterizing correlation quality). The results show that the interrogation window size and frame rate should be chosen such that a particle field is displaced by ≤ 20% of the interrogation window size per frame pair to maximize the correlation coefficient while minimizing its variability.

Although the protocol described herein has proven sufficient for meeting the needs of the current application, it is important to acknowledge its limitations. For example, while contrast enhancement via intensity capping offers ease of implementation, transformations of the entire distribution of pixel intensities may improve the SNR further. Similarly, although correlation-based tracking is well established and provides sufficient resolution for reliably estimating first-order flow characteristics relevant to hemodynamics (e.g., intra-aneurysmal velocity), other techniques may offer a higher spatial resolution (e.g., hybrid PIV/PTV, least-squares matching) and, thus, greater accuracy when considering characteristics that are more sensitive to the velocity field resolution (e.g., wall shear stress, in-plane vorticity). Likewise, while the NMT provides a means for improving the velocity vector field after cross-correlation, it is important to emphasize that this is just one of many vector validation techniques that could be used, each with their own unique advantages and disadvantages that may make their use more suitable for applications beyond those described here. Lastly, while the experimental set-up described here seeks to mimic physiologically relevant flow rates and length scales for the neurovasculature, it does not currently allow the analysis of pulsatile flows. This has not been a limitation for the current application, since the range of Womersley numbers tends to be ≤ 1 (i.e., there is a minimal additive effect of multiple cardiac cycles), which suggests that steady-state conditions are sufficient to recapitulate discrete time points along the cardiac waveform in which the flow rate is comparable. However, for applications where the Womersley number is larger (e.g., vasculature closer to the heart), we envision a potential for introducing pulsatility through the use of an Arduino, which could be used to send the pump a time-varying PWM voltage waveform that enables the mimicking of a cardiac flow profile.

### Disclosures

The authors have nothing to declare.

### Acknowledgements

The authors acknowledge partial support for this project provided by a Collaborative Seed Grant from the Office of Research and Economic Development at UC Riverside.

### References

Materials List for:

Meso-Scale Particle Image Velocimetry Studies of Neurovascular Flows In Vitro

Ryan A. Peck*1, Edver Bahena*1, Reza Jahan2, Guillermo Aguilar1,3,4, Hideaki Tsutsui1,4, Marko Princevac1, Monica M. Wilhelmus1, Masaru P. Rao1,3,4

1Department of Mechanical Engineering, University of California, Riverside
2Division of Interventional Neuroradiology, University of California, Los Angeles
3Materials Science and Engineering Program, University of California, Riverside
4Department of Bioengineering, University of California, Riverside

*These authors contributed equally

Correspondence to: Masaru P. Rao at mprao@engr.ucr.edu

URL: https://www.jove.com/video/58902
DOI: doi:10.3791/58902

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>Catalog Number</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solidworks 2015</td>
<td>Dassault Systems</td>
<td>N/A</td>
<td>CAD Software</td>
</tr>
<tr>
<td>Dow Corning Sylgard 184 Kit</td>
<td>Ellsworth Adhesive</td>
<td>184 SIL ELAST KIT 3.9KG</td>
<td>PDMS Kit</td>
</tr>
<tr>
<td>Stratasys Dimension Elite</td>
<td>Stratasys</td>
<td>9180-00105</td>
<td>3D printer</td>
</tr>
<tr>
<td>P430 Model Material Cartridge</td>
<td>Stratasys</td>
<td>340-21202</td>
<td>ABS build material</td>
</tr>
<tr>
<td>P400 SR Soluble Support Material Cartridge</td>
<td>Stratasys</td>
<td>340-30200</td>
<td>Support material</td>
</tr>
<tr>
<td>CleanStation DT3</td>
<td>PM3 Technologies</td>
<td>00-00300R</td>
<td>Base bath</td>
</tr>
<tr>
<td>Lindberg Blue M LGO Box Furnace</td>
<td>Thermo Scientific</td>
<td>LB305745M</td>
<td>Oven</td>
</tr>
<tr>
<td>21G BD PrecisionGlide Needle</td>
<td>Betcon Dickenson</td>
<td>BD 305167</td>
<td>Branching perforator mold segment</td>
</tr>
<tr>
<td>Desiccator (Vacuum)</td>
<td>Polylab</td>
<td>55205</td>
<td>Desiccator</td>
</tr>
<tr>
<td>Branson 1800 Ultrasonic Cleaning</td>
<td>Branson</td>
<td>CPX-952-116R</td>
<td>Sonicator</td>
</tr>
<tr>
<td>Acetone</td>
<td>Fisher Chemical</td>
<td>A9494</td>
<td>Acetone</td>
</tr>
<tr>
<td>Isopropol Alcohol</td>
<td>Fisher Chemical</td>
<td>A4514</td>
<td>Isopropol Alcohol</td>
</tr>
<tr>
<td>Glycerol</td>
<td>Fisher Chemical</td>
<td>GW33500</td>
<td>Glycerol</td>
</tr>
<tr>
<td>10um Polystyrene Yellow-Green Fluorescent Particles</td>
<td>Magsphere</td>
<td>PSF-010UM</td>
<td>Fluorescent beads</td>
</tr>
<tr>
<td>Phantom Miro</td>
<td>Vision Research</td>
<td>Miro M310</td>
<td>High speed camera</td>
</tr>
<tr>
<td>Micropump</td>
<td>Cole-Parmer</td>
<td>81101</td>
<td>Recirculating pump</td>
</tr>
<tr>
<td>Leica 10X Objective</td>
<td>Leica Microsystems</td>
<td>506259</td>
<td>Objective for perforator</td>
</tr>
<tr>
<td>Leica 2.5X Objective</td>
<td>Leica Microsystems</td>
<td>11506083</td>
<td>Objective aneurysm sac</td>
</tr>
<tr>
<td>Leica Blue Filter Cube L5</td>
<td>Leica Microsystems</td>
<td>513840</td>
<td>Blue filter cube</td>
</tr>
<tr>
<td>Leica EL6000</td>
<td>Leica Microsystems</td>
<td>11504115</td>
<td>Light source</td>
</tr>
<tr>
<td>Alconox</td>
<td>Alconox Inc</td>
<td>1104-1</td>
<td>Detergent</td>
</tr>
</tbody>
</table>
| ImageJ                                   | NIH              | N/A            | Open source image analysis software
|                                          |                  |                | https://imagej.nih.gov/ij/ |
| Particle Image Velocimetry PIV Plugin    | Qingson Tseng    | N/A            | https://sites.google.com/site/qingzongtseng/piv |
/*This code performs image thresholding based on pixel intensity distribution*/

for (i=1; i<=nSlices; i++) {
    setSlice(i);
    getStatistics(area, mean, min, max, std);
    //run("Log", "slice");
    setMinAndMax(mean + std, max);
    run("Apply LUT", "slice");
    run("Despeckle", "slice");
    run("Gaussian Blur...", "sigma=1.5 slice");
}
/* This code runs PIV on an image sequence and performs a temporal average of each velocity vector*/

macro "Run PIV on Stack [f1]"{
    id0 = getImageID();
slices = nSlices;
outputStr = "";
xcomp = 0;
ycomp = 0;
mag = 0;

    // creates dialog boxes to enter important parameters, these may vary by experimental setup
    Dialog.create("Multi-slice PIV");
    Dialog.addNumber("Interrogation window 1", 256);
    Dialog.addNumber("search window 1", 512);
    Dialog.addNumber("vector spacing 1", 128);
    Dialog.addNumber("Interrogation window 2", 128);
    Dialog.addNumber("search window 2", 256);
    Dialog.addNumber("vector spacing 2", 64);
    Dialog.addNumber("Interrogation window 3", 0);
    Dialog.addNumber("search window 3", 0);
    Dialog.addNumber("vector spacing 3", 0);
    Dialog.addNumber("correlation threshold", 0.8);
    Dialog.addNumber("Video Frames Per Second", 2000);
    Dialog.addNumber("Pixels per mm", 475);
    Dialog.show;

    piv1 = Dialog.getNumber();
    sw1 = Dialog.getNumber();
    vs1 = Dialog.getNumber();
    piv2 = Dialog.getNumber();
    sw2 = Dialog.getNumber();
    vs2 = Dialog.getNumber();
    piv3 = Dialog.getNumber();
    sw3 = Dialog.getNumber();
    vs3 = Dialog.getNumber();
    corr = Dialog.getNumber();
    fps = Dialog.getNumber();
    pixelspermm = Dialog.getNumber();

    path = getDirectory("select a folder to save PIV results");

    for(s=1; s<slices; s++){
        selectImage(id0);
        call("java.lang.System.gc");
        run("Duplicate...", "title=[seq_"+s+"o] duplicate range="+s+-"+s+1""");
        run("iterative PIV(Advanced)", " piv1="+piv1+" sw1="+sw1+" vs1="+vs1+"
            piv2="+piv2+" sw2="+sw2+" vs2="+vs2+" piv3="+piv3+" sw3="+sw3+" vs3="+vs3+
            correlation="+corr+" batch path=["+path+"]");
        for(p=1; p<2; p++){
            selectWindow("seq_"+ s + "_PIV" + p);
            close();
        }
        selectWindow("seq_"+ s);
        close();
    }

    for(i=1; i<slices; i++){
        dataPath = File.openAsString(path + "seq_"+i+"_PIV2_disp.txt");
        rows = split(dataPath, "\n");
        
    }
}
//On first pass initialize displacement arrays to be same size as PIV data arrays
if(i == 1){
    print("outputting average..." ideology);
    xcoords = newArray(rows.length);
    Array.fill(xcoords, 0);
    ycoords = newArray(rows.length);
    Array.fill(ycoords, 0);
    mags = newArray(rows.length);
    Array.fill(mags, 0);
    xdisp = newArray(rows.length);
    Array.fill(xdisp, 0);
    ydisp = newArray(rows.length);
    Array.fill(ydisp, 0);
    xZeros = newArray(rows.length);
    Array.fill(xZeros, 0);
    yZeros = newArray(rows.length);
    Array.fill(yZeros, 0);
}
for(j=0; j<rows.length; j++){
    cols = split(rows[j], " ");
    xcomp = 0 + cols[2];
    ycomp = 0 + cols[3];
    mag = sqrt(xcomp*xcomp + ycomp*ycomp);
    //if no velocity vector at point, exclude it from the noise averaging
    if(mag == 0){
        xZeros[j]++;
        yZeros[j]++;
    } else{
        //otherwise, sum in time to be averaged later
        xdisp[j] = xdisp[j] + cols[2];
        ydisp[j] = ydisp[j] + cols[3];
    }
}
if(i == slices-1){
    //on last pass divide summed velocities by number of frames to average
    if(i-xZeros[j] !=0){
        xdisp[j] = xdisp[j]/abs(i-xZeros[j]) * fps/pixelspermm;
    } else{
        xdisp[j] = 0;
    }
    if(i-yZeros[j] != 0){
        ydisp[j] = ydisp[j]/abs(i-yZeros[j]) * fps/pixelspermm;
    } else{
        ydisp[j] = 0;
    }
    xcoords[j] = cols[0];
    ycoords[j] = cols[1];
    mags[j] = sqrt(xdisp[j]*xdisp[j] + ydisp[j]*ydisp[j]);
    outputStr += xcoords[j] + " " + ycoords[j] + " " + xdisp[j] + " " + ydisp[j] + " " + mags[j] + " " + "\n";
}
File.saveString(outputStr, path + "\averaged.txt");
print("Done!!!!");

function closeAllWindows()
{
    for(i=1; i<= nImages; i++)
    {
        selectImage(i);
        if(getImageID() != id0)
        {
            close();
        }
    }
}
/This code performs the normalized median test on ImageJ PIV output*/

function median(a){
    Array.sort(a);
    med = 0.0;
    if(a.length%2 == 0){
        med = (0 + a[(a.length/2 - 1)] + a[(a.length/2)])/2;
    }else{
        med = 0 + a[a.length/2 -1];
    }
    return med;
}

function setElement(myArray, height, row, col, value){
    //hashes 2-D array values in a 1-D array in row major order
    myArray[col * height + row] = (0 + value);
}

function getElement(myArray, height, row, col){
    //returns values from the hashed 1-D array
    return (0 + myArray[col * height + row]);
}

Dialog.create("Normalized Median Test");
Dialog.addNumber("eta (Noise Parameter)", 0.2);
Dialog.addNumber("threshold", 2);
Dialog.addNumber("magnitude threshold (lower bound)", 0);
Dialog.addNumber("magnitude threshold (upper bound)", 99999);
Dialog.addNumber("Video Frames Per Second", 2000);
Dialog.addNumber("Pixels per mm", 475);
Dialog.addNumber("frames", 2000);
Dialog.show;

eta = Dialog.getNumber();
rtthresh = Dialog.getNumber();
magThreshLower = Dialog.getNumber();
magThreshUpper = Dialog.getNumber();
fps = Dialog.getNumber();
pixelspermm = Dialog.getNumber();
slices = Dialog.getNumber();
replacedVectors = 0;
totalVectors = 0;

path = getDirectory("select a folder with Multiple PIV Results");

for(z=1; z <=slices; z++){
    call("java.lang.System.gc");
    dataPath = File.openAsString(path + "seq_" + z + "_PIV2_disp.txt");
    outputStr = ""
    rows = split(dataPath, \\
    cols = split(rows[0], \\
    temp = split(rows[rows.length-1], \\

    xlength = 0 + temp[0];
```javascript
length = 0 + temp[1];

//initialize 2-D arrays
u = newArray((xlength+1) * (ylength+1));
Array.fill(u, 0.0);
v = newArray((xlength+1) * (ylength+1));
Array.fill(v, 0.0);
height = ylength;
width = xlength;

xcoords = newArray(rows.length);
Array.fill(xcoords, 0);
ycoords = newArray(rows.length);
Array.fill(ycoords, 0);

for(i=0; i< rows.length; i++) {
    temp = split(rows[i], " ");
    xcoords[i] = temp[0];
ycoords[i] = temp[1];
    setElement(u, height, temp[0], temp[1], temp[2]);
    setElement(v, height, temp[0], temp[1], temp[3]);
}

step = 0 + xcoords[1] - xcoords[0];

uNeighbs = newArray(8);
vNeighbs = newArray(8);
uResiduals = newArray(8);
vResiduals = newArray(8);

//
for(x = xcoords[0] ; x<=xlength; x+= step){
    for(y = ycoords[0]; y<=ylength; y+=step){
        utmp = getElement(u, height, x, y);
        //if vectors are on edges, set the appropriate neighbors to zero
        if(x-step <= 0){
            uNeighbs[0] = 0;
vNeighbs[0] = 0;
uNeighbs[3] = 0;
vNeighbs[3] = 0;
uNeighbs[5] = 0;
vNeighbs[5] = 0;
        }

        if(y - step <= 0){
            uNeighbs[0] = 0;
vNeighbs[0] = 0;
uNeighbs[1] = 0;
vNeighbs[1] = 0;
uNeighbs[2] = 0;
vNeighbs[2] = 0;
        }

        if(x+step >= xlength){
```
uNeighbs[2] = 0;
vNeighbs[2] = 0;
uNeighbs[4] = 0;
vNeighbs[4] = 0;
uNeighbs[7] = 0;
vNeighbs[7] = 0;
}

if(y + step >= ylength){
  uNeighbs[5] = 0;
  vNeighbs[5] = 0;
  uNeighbs[6] = 0;
  vNeighbs[6] = 0;
  uNeighbs[7] = 0;
  vNeighbs[7] = 0;
}

//if vector is in the middle, populate an array of all its neighbors
if(x-step > 0 && y-step > 0 && x+step < xlength && y+step < ylength){
  uNeighbs[0] = 0 + getElement(u,height, x-step, y-step);
  uNeighbs[1] = 0 + getElement(u,height,x, y-step);
  uNeighbs[2] = 0 + getElement(u,height,x+step, y-step);
  uNeighbs[3] = 0 + getElement(u,height,x - step, y);
  //print("getElement (" + (x + step) + ", " + (y+step) + 
" + getElement(u,height, x + step, y));
  uNeighbs[4] = 0 + getElement(u,height,x+step, y);
  uNeighbs[5] = 0 + getElement(u,height,x-step, y+step);
  uNeighbs[6] = 0 + getElement(u,height,x, y+step);
  uNeighbs[7] = 0 + getElement(u,height,x+step, y+step);
  uMedian = 0 + median(uNeighbs);

  vtmp = getElement(v,height,x,y);
  vNeighbs[0] = 0 + getElement(v,height, x-step, y-step);
  vNeighbs[1] = 0 + getElement(v,height,x, y-step);
  vNeighbs[2] = 0 + getElement(v,height,x+step, y-step);
  vNeighbs[3] = 0 + getElement(v,height,x - step, y);
  vNeighbs[4] = 0 + getElement(v,height,x+step, y);
  vNeighbs[5] = 0 + getElement(v,height,x-step, y+step);
  vNeighbs[6] = 0 + getElement(v,height,x, y+step);
  vNeighbs[7] = 0 + getElement(v,height,x+step, y+step);
  vMedian = 0 + median(vNeighbs);

  for(p=0; p<8; p++){ //calculate array of residuals
    uResiduals[p] = abs(uNeighbs[p] - uMedian);
    vResiduals[p] = abs(vNeighbs[p] - vMedian);
  }

  if( abs(utmp - uMedian)/(median(uResiduals) + eta) > rthresh ||
      abs(vtmp - vMedian)/(median(vResiduals) + eta) > rthresh){ //perform normalized median test
    setElement(u,height, x,y, 0);
    setElement(v,height, x,y, 0);

    //Uncomment the following four lines to replace the vector with a bilinear interpolation
    uiinterp = 0.25* (uNeighbs[0] + uNeighbs[2] + uNeighbs[5] + uNeighbs[7]); uncomment this line to

for(j=0; j< rows.length; j++){  
    //selectWindow("U");
    uout = getElement(u,height, xcoords[j], ycoords[j]);
    //selectWindow("V");
    vout = getElement(v,height, xcoords[j], ycoords[j]);
    if(uout != 0 || vout != 0){
        totalVectors = totalVectors + 1;
    }

    outputStr += xcoords[j] + " " + ycoords[j] + " " + 0+uout*fps/pixelspermm + " " + 0+vout*fps/pixelspermm + " " + "\n";
}

File.saveString(outputStr, path + '\\NMT' + z + '.txt');
print("Iteration " + z + " complete");
call("java.lang.System.gc"); //garbage collection
}

outputStr2 = '';  
//open each NMT'ed frame and perform a temporal average
for(q=1; q<slices; q++){  
    call("java.lang.System.gc"); //garbage collection
    NMTdata = File.openAsString(path + "\\NMT" + q + '.txt');
    NMTrows = split(NMTdata, "\n");

    //On first pass initialize displacement arrays to be same size as PIV data arrays
    if(q ==1){
        print ("outputting average...");
        NMTxcoords = newArray(NMTrows.length);
        Array.fill(NMTxcoords, 0);
        NMTycoords = newArray(NMTrows.length);
        Array.fill(NMTycoords, 0);
        mags = newArray(NMTrows.length);
        Array.fill(mags,0);
        xdisp = newArray(NMTrows.length);
        Array.fill(xdisp, 0);
        ydisp = newArray(NMTrows.length);
        Array.fill(ydisp, 0);
        xZeros = newArray(NMTrows.length);
        Array.fill(xZeros, 0);
        yZeros = newArray(NMTrows.length);
        Array.fill(yZeros,0);
    }

    for(h=0; h<NMTrows.length; h++){  
        cols = split(NMTrows[h], " ");

xcomp = 0 + cols[2];
ycomp = 0 + cols[3];
mag = sqrt(xcomp*xcomp + ycomp*ycomp);

//if no velocity vector at point, exclude it from the noise averaging

if(mag ==0){
    xZeros[h]++;
    yZeros[h]++;
}

if(mag != 0){
    xdisp[h] = xdisp[h] + cols[2];
    ydisp[h] = ydisp[h] + cols[3];
}

if(q == slices-1){
    if(i-xZeros[h] !=0 && q != xZeros[h]){ //if there are zeros for some, but not all vectors at this location throughout the stack then only average non-zero vectors
        xdisp[h] = xdisp[h]/abs(q-xZeros[h]);
    } else{
        xdisp[h] = 0;
    }

    if(i-yZeros[h] !=0 && q != yZeros[h]){ //if there are zeros for some, but not all vectors at this location throughout the stack then only average non-zero vectors
        ydisp[h] = ydisp[h]/abs(q-yZeros[h]);
    } else{
        ydisp[h] = 0;
    }

    NMTxcoords[h] = cols[0];
    NMTycoords[h] = cols[1];

    mags[h] = sqrt( xdisp[h]*xdisp[h] + ydisp[h]*ydisp[h] );

    outputStr2 += NMTxcoords[h] + " " + NMTycoords[h] + " " + xdisp[h] + " " + ydisp[h] + " " + mags[h] + " " + "\n";
}

File.saveString(outputStr2, path + "\NMTaveraged(No Interpolation).txt");

//uncomment to report vector validation statistics
//File.saveString("Number of Replaced Vectors:" + replacedVectors + "\n Number of Total Vectors: " + totalVectors + " Percent Replaced: " + replacedVectors/totalVectors, path + "\\replacedVectors.txt");

print("Done!!!!");