ROSPIV: A Robot Operating System (ROS) network for performing Particle Image Velocimetry (PIV) in rivers

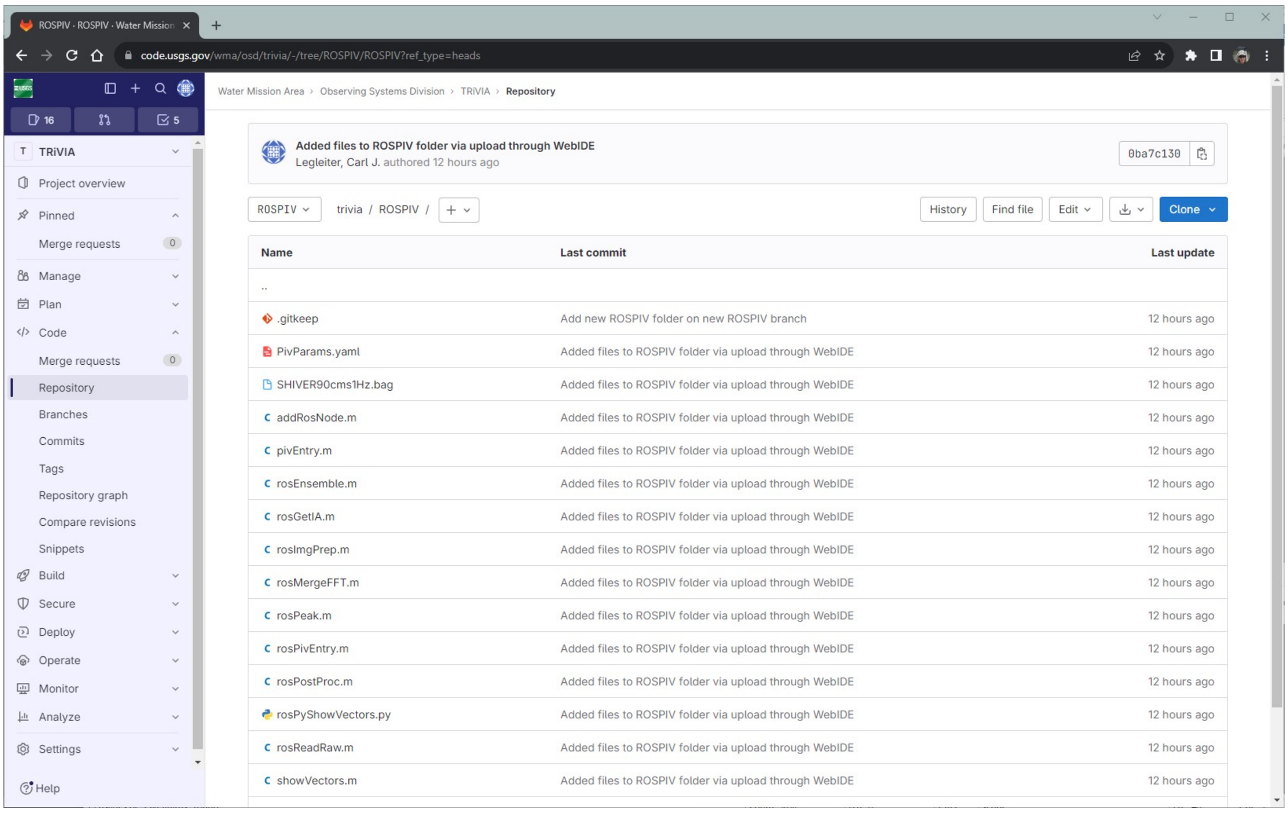
Carl J. Legleiter (cjl@usgs.gov)

September 12, 2023

0. Preliminaries

1. The ROSPIV codebase is available from the USGS GitLab repository for the Toolbox for River Velocimetry using Images from Aircraft (TRIVIA) under a distinct branch named `ROSPIV`. To clone this branch of the repository, use the following command:

   ```
   git clone --single-branch --branch ROSPIV https://code.usgs.gov/wma/osd/TRiVIA.git
   ```



Added files to ROSPIV folder via upload through WebIDE
Legleiter, Carl J. authored 12 hours ago
0ba7c130

Name	Last commit	Last update
..		
.gitkeep	Add new ROSPIV folder on new ROSPIV branch	12 hours ago
PivParams.yaml	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
SHIVER90cms1Hz.bag	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
addRosNode.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
pivEntry.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosEnsemble.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosGetIA.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosImgPrep.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosMergeFFT.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosPeak.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosPivEntry.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosPostProc.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosPyShowVectors.py	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
rosReadRaw.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago
showVectors.m	Added files to ROSPIV folder via upload through WebIDE	12 hours ago

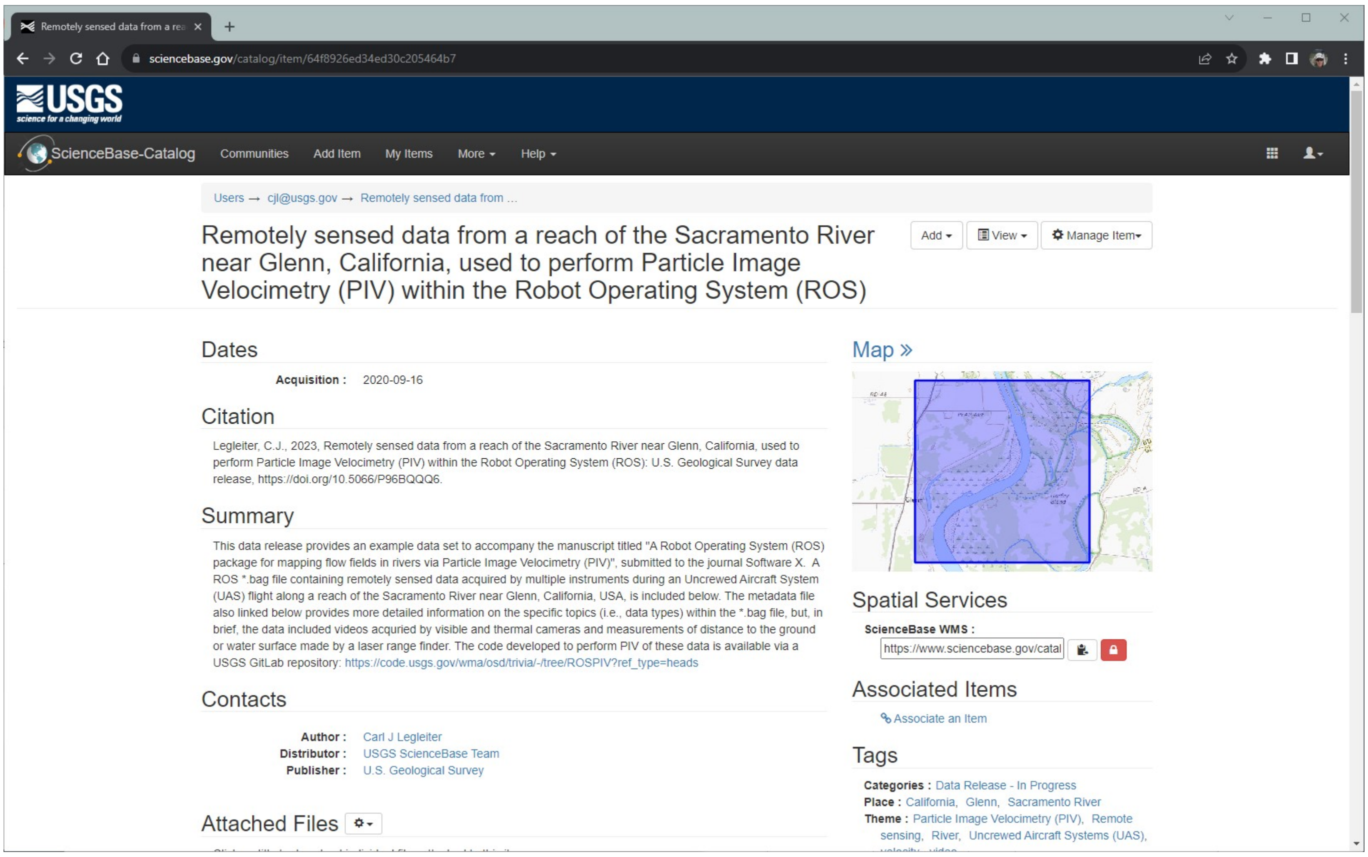
2. For guidance on setting up a ROS development environment on a Windows desktop computer, see the separate document `ROS4windows.pdf` provided as an additional supplement to this article.

1. Obtaining ROS `*.bag` files for testing

1. To see how to make a ROS `*.bag` file from a sequence of images stored in a MATLAB structure array, please refer to the first cell of code in `demoScript.m`, which makes use of the function `stack2bag.m`. This example is based on a simulated image sequence generated using the Simulating Hydraulics and Images for Velocimetry and Refinement (SHIVER) framework described in Legleiter and Kinzel (in review). The file resulting from this process is named `SHIVER90cms1Hz.bag` and is available in the `ROSPIV` folder on the `ROSPIV` branch of the TRIVIA repository.

2. For a sample `*.bag` file with real data from a reach of the Sacramento River in northern California, USA, please refer to the data release hosted on the USGS ScienceBase catalog:

   Legleiter, C.J., 2023, Remotely sensed data from a reach of the Sacramento River near Glenn, California, used to perform Particle Image Velocimetry (PIV) within the Robot Operating System (ROS): U.S. Geological Survey data release, https://doi.org/10.5066/P96BQQQ6.



Remotely sensed data from a rea
sciencebase.gov/catalog/item/64f8926ed34ed30c205464b7
USGS
science for a changing world
ScienceBase-Catalog
Communities
Add Item
My Items
More
Help
Users → cjl@usgs.gov → Remotely sensed data from ...
Add
View
Manage Item

Remotely sensed data from a reach of the Sacramento River near Glenn, California, used to perform Particle Image Velocimetry (PIV) within the Robot Operating System (ROS)

Dates

Acquisition : 2020-09-16

Citation

Legleiter, C.J., 2023, Remotely sensed data from a reach of the Sacramento River near Glenn, California, used to perform Particle Image Velocimetry (PIV) within the Robot Operating System (ROS): U.S. Geological Survey data release, https://doi.org/10.5066/P96BQQQ6.

Summary

This data release provides an example data set to accompany the manuscript titled "A Robot Operating System (ROS) package for mapping flow fields in rivers via Particle Image Velocimetry (PIV)", submitted to the journal Software X. A ROS *.bag file containing remotely sensed data acquired by multiple instruments during an Uncrewed Aircraft System (UAS) flight along a reach of the Sacramento River near Glenn, California, USA, is included below. The metadata file also linked below provides more detailed information on the specific topics (i.e., data types) within the *.bag file, but, in brief, the data included videos acquried by visible and thermal cameras and measurements of distance to the ground or water surface made by a laser range finder. The code developed to perform PIV of these data is available via a USGS GitLab repository: https://code.usgs.gov/wma/osd/trivia/-/tree/ROSPIV?ref_type=heads

Contacts

Author : Carl J Legleiter
Distributor : USGS ScienceBase Team
Publisher : U.S. Geological Survey

Attached Files

Map »

Spatial Services

ScienceBase WMS : https://www.sciencebase.gov/catal

Associated Items

Associate an Item

Tags

Categories : Data Release - In Progress
Place : California, Glenn, Sacramento River
Theme : Particle Image Velocimetry (PIV), Remote sensing, River, Uncrewed Aircraft Systems (UAS),

2. Create a parameter file for the simulated image sequence

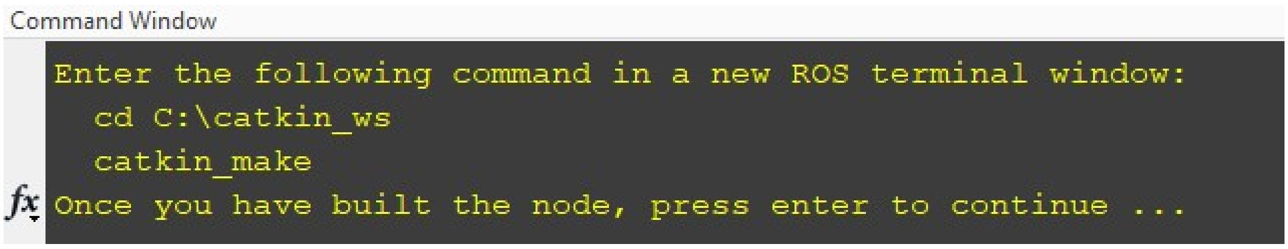
1. Follow the instructions in the second cell of code in `demoScript.m` to create a parameter file for the simulated image sequence named `PivParams.yaml` using the function `pivEntry.m`.
2. The documentation for `pivEntry.m` provides more detail on the parameters. To view this information, type `doc pivEntry` at the MATLAB command prompt.

3. Building the network of ROS nodes for PIV from MATLAB source code

1. The function `addRosNode.m` was designed to facilitate the process of building a ROS node from MATLAB source code and placing the new node within a ROS package.

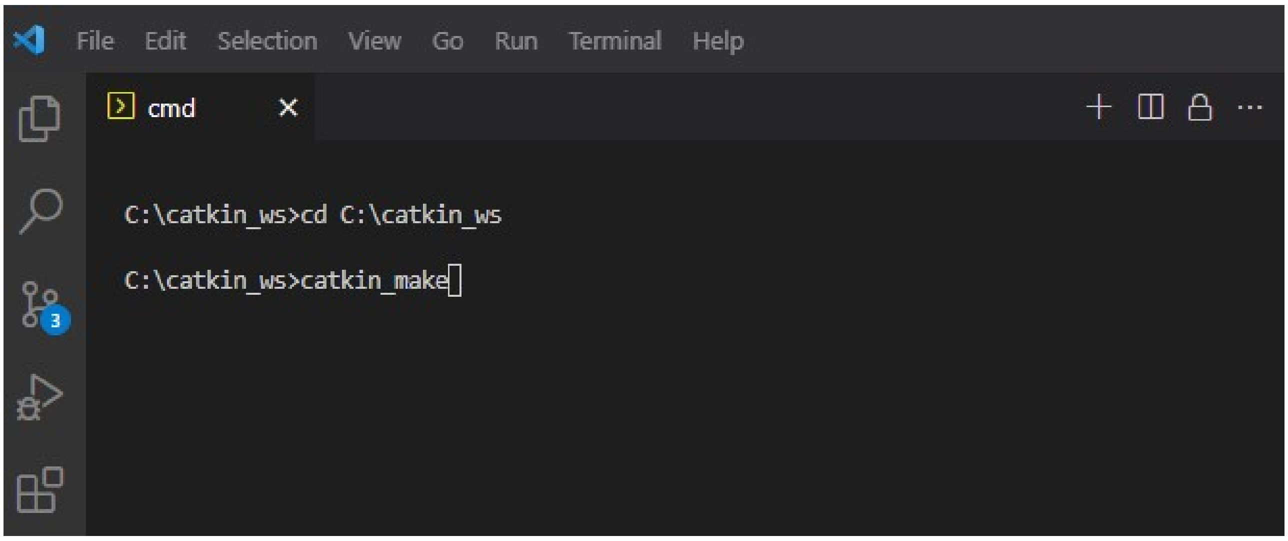
2. The third cell of `demoScript.m` provides an example for the first of the eight nodes that will need to be built. Note that although this function can also be used to test the ROS network by providing interactive guidance on how to play a `*.bag` file and run all the nodes in turn, at this stage we will only use it to build the nodes one at a time.

3. Leave the `testFlag` input argument set to `true` but rather than running through a complete test, just follow the instructions that appear in the MATLAB command window to build the first node, which is named `rosPivEntry`.



```
Enter the following command in a new ROS terminal window:
  cd C:\catkin_ws
  catkin_make
Once you have built the node, press enter to continue ...
```

4. To do so, you will need to open a ROS terminal window using either the Windows Terminal program as described in the `ROS4windows.pdf` document or within an integrated development environment (IDE) such as Visual Studio Code (VS Code), as shown here:



```
C:\catkin_ws>cd C:\catkin_ws

C:\catkin_ws>catkin_make
```

5. After completing this step in a ROS terminal, switch back to MATLAB and press `Ctrl+C` to abort the process and return to the MATLAB command prompt. You will then need to return to the directory containing the MATLAB source code: `cd(matFuncDir)`.

6. To assemble the entire ROS network for PIV, you will need to run `addRosNode.m` for a series of eight nodes:

    1. `rosPivEntry`
    2. `rosRedRaw`
    3. `rosImgPrep`
    4. `rosGetIA`
    5. `rosMergeFFT`
    6. `rosEnsemble`
    7. `rosPeak`
    8. `rosPostProc`

4. Use the ROS network to perform PIV for simulated image sequence

1. After building all eight ROS nodes, you can do a full PIV run for the simulated image sequence now stored in the file `SHIVER90cms1Hz.bag`. To do so, you will need to open a total of eight separate ROS terminals: one for the ROS core, one to play the bag file, and one for each of seven of the eight ROS nodes. VS Code is conducive to this type of workflow because you can arrange multiple tiles within the IDE.

2. First, copy the file `PivParams.yaml` produced by `pivEntry.m` from the directory with the MATLAB source code to the appropriate location in the catkin workspace: `C:\catkin_ws\src\streamflowpiv`.

3. You can then load these parameters to the ROS server and confirm their values by issuing these commands in a ROS terminal:

   ```
   cd C:\catkin_ws
   devel\setup.bat
   roslaunch streamflowpiv streamflowpiv.launch
   rosparam get /
   ```

   Note that the `roslaunch` command also starts a ROS master (i.e., equivalent to running `roscore`), so you will need to leave this terminal open, at least in the background.

4. In a second ROS terminal, start playing the bag file using the following command, but then immediately press the space bar to pause playback until all the other nodes are ready to go.

   ```
   rosbag play C:\directory\with\bag\SHIVER90cms1Hz.bag
   ```

5. For each of the six remaining ROS terminals that will be used to actually run the nodes, first enter two setup commands:

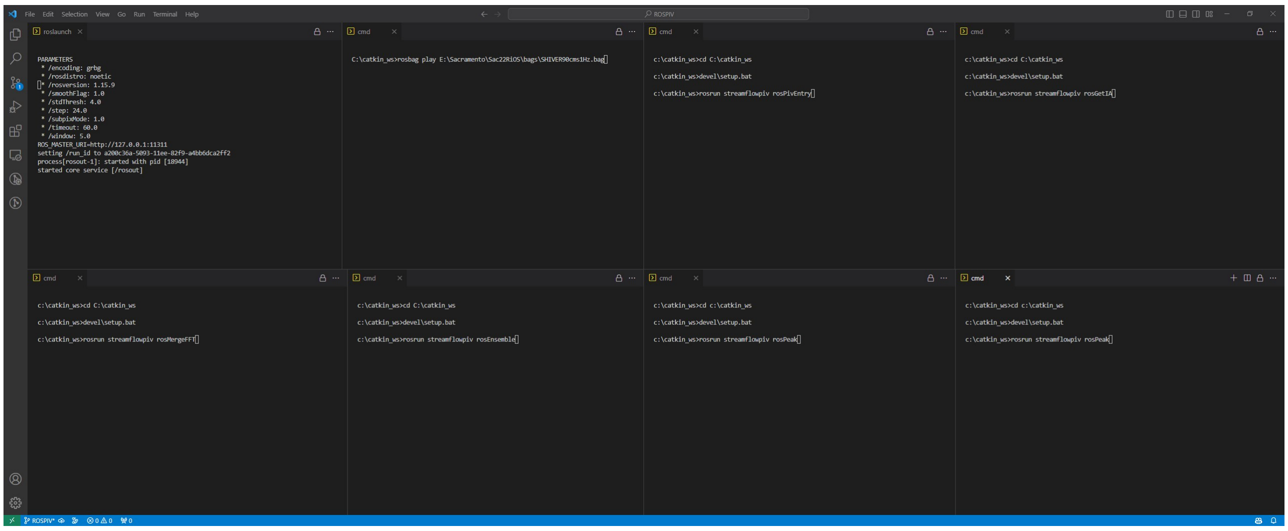
   ```
   cd C:\catkin_ws
   devel\setup.bat
   ```

6. When you are ready to proceed, go back to the terminal that will be used to play the ROS bag and press the space bar again to start the playback.

7. Then, go to each of the other terminals in turn and issue a command like:

   ```
   rosrun [packageName] [nodeName]
   ```

   In this case, the variable `[packageName]` is `streamflowpiv` and the variable `[nodeName]` will take on each of the six node names in turn: `rosPivEntry`, `rosGetIA`, `rosMergeFFT`, `rosEnsemble`, `rosPeak`, and `rosPostProc`. Note that once you start the bag playing you will have to quickly launch the other nodes in turn, otherwise the bag could finish playing before the nodes are ready to receive the messages. If this happens, you will need to restart the bag playing and then quickly launch the nodes. To get a head start, set up the ROS terminals in advance as shown below so that all you have to do is click in each terminal to make it the active window and the press enter to run the node. As each node runs, status information will be displayed in the terminal.



```
PARAMETERS
 * /encoding: grbg
 * /rosdistro: noetic
 * /rosversion: 1.15.9
 * /smoothFlag: 1.0
 * /stdThresh: 4.0
 * /step: 24.0
 * /subpixMode: 1.0
 * /timeout: 60.0
 * /window: 5.0
ROS_MASTER_URI=http://127.0.0.1:11311
setting /run_id to a200c36a-5093-11ee-82f9-a4bb6dca2ff2
process[rosout-1]: started with pid [18944]
started core service [/rosout]
```

```
C:\catkin_ws>rosbag play E:\Sacramento\Sac22RiOS\bags\SHIVER90cms1Hz.bag
```

```
c:\catkin_ws>cd C:\catkin_ws

c:\catkin_ws>devel\setup.bat

c:\catkin_ws>rosrun streamflowpiv rosPivEntry
```

```
c:\catkin_ws>cd C:\catkin_ws

c:\catkin_ws>devel\setup.bat

c:\catkin_ws>rosrun streamflowpiv rosGetIA
```

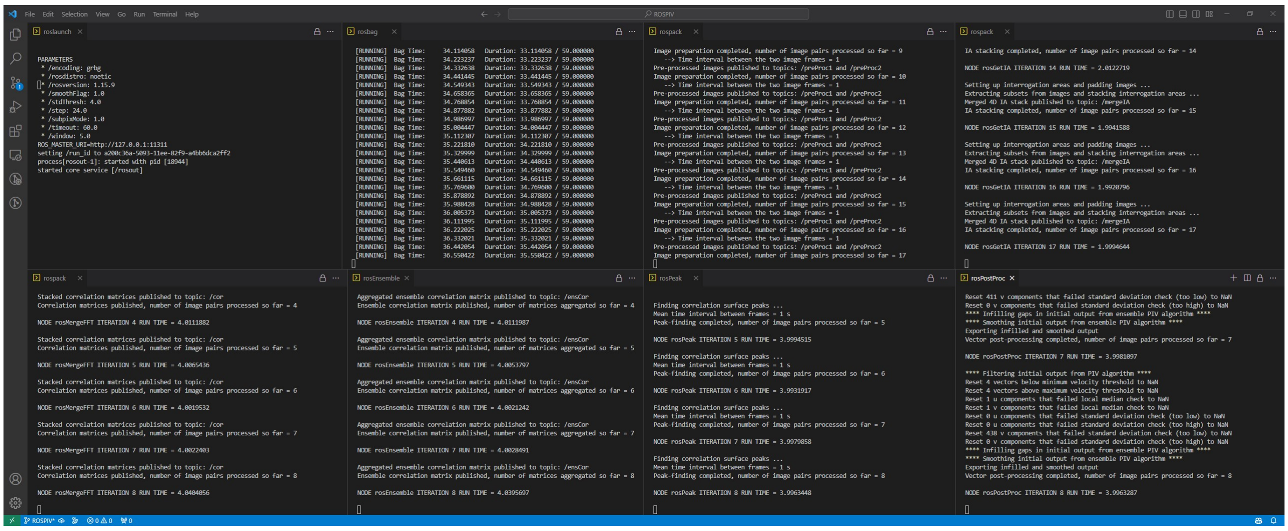
```
c:\catkin_ws>cd C:\catkin_ws

c:\catkin_ws>devel\setup.bat

c:\catkin_ws>rosrun streamflowpiv rosMergeFFT
```

```
c:\catkin_ws>cd C:\catkin_ws

c:\catkin_ws>devel\setup.bat

c:\catkin_ws>rosrun streamflowpiv rosEnsemble
```

```
c:\catkin_ws>cd c:\catkin_ws

c:\catkin_ws>devel\setup.bat

c:\catkin_ws>rosrun streamflowpiv rosPeak
```

```
c:\catkin_ws>cd c:\catkin_ws

c:\catkin_ws>devel\setup.bat

c:\catkin_ws>rosrun streamflowpiv rosPeak
```

As each node runs, status information will be displayed in the terminal.



```
PARAMETERS
 * /encoding: grbg
 * /rosdistro: noetic
 * /rosversion: 1.15.9
 * /smoothFlag: 1.0
 * /stdThresh: 4.0
 * /step: 24.0
 * /subpixMode: 1.0
 * /timeout: 60.0
 * /window: 5.0
ROS_MASTER_URI=http://127.0.0.1:11311
setting /run_id to a200c36a-5093-11ee-82f9-a4bb6dca2ff2
process[rosout-1]: started with pid [18944]
started core service [/rosout]
```

```
[RUNNING]  Bag Time:    34.114058   Duration: 33.114058 / 59.000000
[RUNNING]  Bag Time:    34.223237   Duration: 33.223237 / 59.000000
[RUNNING]  Bag Time:    34.332638   Duration: 33.332638 / 59.000000
[RUNNING]  Bag Time:    34.441445   Duration: 33.441445 / 59.000000
[RUNNING]  Bag Time:    34.549343   Duration: 33.549343 / 59.000000
[RUNNING]  Bag Time:    34.658365   Duration: 33.658365 / 59.000000
[RUNNING]  Bag Time:    34.768854   Duration: 33.768854 / 59.000000
[RUNNING]  Bag Time:    34.877882   Duration: 33.877882 / 59.000000
[RUNNING]  Bag Time:    34.986997   Duration: 33.986997 / 59.000000
[RUNNING]  Bag Time:    35.004447   Duration: 34.004447 / 59.000000
[RUNNING]  Bag Time:    35.112307   Duration: 34.112307 / 59.000000
[RUNNING]  Bag Time:    35.221810   Duration: 34.221810 / 59.000000
[RUNNING]  Bag Time:    35.329999   Duration: 34.329999 / 59.000000
[RUNNING]  Bag Time:    35.440613   Duration: 34.440613 / 59.000000
[RUNNING]  Bag Time:    35.549460   Duration: 34.549460 / 59.000000
[RUNNING]  Bag Time:    35.661115   Duration: 34.661115 / 59.000000
[RUNNING]  Bag Time:    35.769600   Duration: 34.769600 / 59.000000
[RUNNING]  Bag Time:    35.878892   Duration: 34.878892 / 59.000000
[RUNNING]  Bag Time:    35.988428   Duration: 34.988428 / 59.000000
[RUNNING]  Bag Time:    36.005373   Duration: 35.005373 / 59.000000
[RUNNING]  Bag Time:    36.111995   Duration: 35.111995 / 59.000000
[RUNNING]  Bag Time:    36.222025   Duration: 35.222025 / 59.000000
[RUNNING]  Bag Time:    36.332021   Duration: 35.332021 / 59.000000
[RUNNING]  Bag Time:    36.442054   Duration: 35.442054 / 59.000000
[RUNNING]  Bag Time:    36.550422   Duration: 35.550422 / 59.000000
```

```
Image preparation completed, number of image pairs processed so far = 9
   --> Time interval between the two image frames = 1
Pre-processed images published to topics: /preProc1 and /preProc2
Image preparation completed, number of image pairs processed so far = 10
   --> Time interval between the two image frames = 1
Pre-processed images published to topics: /preProc1 and /preProc2
Image preparation completed, number of image pairs processed so far = 11
   --> Time interval between the two image frames = 1
Pre-processed images published to topics: /preProc1 and /preProc2
Image preparation completed, number of image pairs processed so far = 12
   --> Time interval between the two image frames = 1
Pre-processed images published to topics: /preProc1 and /preProc2
Image preparation completed, number of image pairs processed so far = 13
   --> Time interval between the two image frames = 1
Pre-processed images published to topics: /preProc1 and /preProc2
Image preparation completed, number of image pairs processed so far = 14
   --> Time interval between the two image frames = 1
Pre-processed images published to topics: /preProc1 and /preProc2
Image preparation completed, number of image pairs processed so far = 15
   --> Time interval between the two image frames = 1
Pre-processed images published to topics: /preProc1 and /preProc2
Image preparation completed, number of image pairs processed so far = 16
   --> Time interval between the two image frames = 1
Pre-processed images published to topics: /preProc1 and /preProc2
Image preparation completed, number of image pairs processed so far = 17
```

```
IA stacking completed, number of image pairs processed so far = 14

NODE rosGetIA ITERATION 14 RUN TIME = 2.0122719

Setting up interrogation areas and padding images ...
Extracting subsets from images and stacking interrogation areas ...
Merged 4D IA stack published to topic: /mergeIA
IA stacking completed, number of image pairs processed so far = 15

NODE rosGetIA ITERATION 15 RUN TIME = 1.9941588

Setting up interrogation areas and padding images ...
Extracting subsets from images and stacking interrogation areas ...
Merged 4D IA stack published to topic: /mergeIA
IA stacking completed, number of image pairs processed so far = 16

NODE rosGetIA ITERATION 16 RUN TIME = 1.9920796

Setting up interrogation areas and padding images ...
Extracting subsets from images and stacking interrogation areas ...
Merged 4D IA stack published to topic: /mergeIA
IA stacking completed, number of image pairs processed so far = 17

NODE rosGetIA ITERATION 17 RUN TIME = 1.9994644
```

```
Stacked correlation matrices published to topic: /cor
Correlation matrices published, number of image pairs processed so far = 4

NODE rosMergeFFT ITERATION 4 RUN TIME = 4.0111882

Stacked correlation matrices published to topic: /cor
Correlation matrices published, number of image pairs processed so far = 5

NODE rosMergeFFT ITERATION 5 RUN TIME = 4.0065436

Stacked correlation matrices published to topic: /cor
Correlation matrices published, number of image pairs processed so far = 6

NODE rosMergeFFT ITERATION 6 RUN TIME = 4.0019532

Stacked correlation matrices published to topic: /cor
Correlation matrices published, number of image pairs processed so far = 7

NODE rosMergeFFT ITERATION 7 RUN TIME = 4.0022403

Stacked correlation matrices published to topic: /cor
Correlation matrices published, number of image pairs processed so far = 8

NODE rosMergeFFT ITERATION 8 RUN TIME = 4.0404056
```

```
Aggregated ensemble correlation matrix published to topic: /ensCor
Ensemble correlation matrix published, number of matrices aggregated so far = 4

NODE rosEnsemble ITERATION 4 RUN TIME = 4.0111987

Aggregated ensemble correlation matrix published to topic: /ensCor
Ensemble correlation matrix published, number of matrices aggregated so far = 5

NODE rosEnsemble ITERATION 5 RUN TIME = 4.0053797

Aggregated ensemble correlation matrix published to topic: /ensCor
Ensemble correlation matrix published, number of matrices aggregated so far = 6

NODE rosEnsemble ITERATION 6 RUN TIME = 4.0021242

Aggregated ensemble correlation matrix published to topic: /ensCor
Ensemble correlation matrix published, number of matrices aggregated so far = 7

NODE rosEnsemble ITERATION 7 RUN TIME = 4.0028491

Aggregated ensemble correlation matrix published to topic: /ensCor
Ensemble correlation matrix published, number of matrices aggregated so far = 8

NODE rosEnsemble ITERATION 8 RUN TIME = 4.0395697
```

```
Finding correlation surface peaks ...
Mean time interval between frames = 1 s
Peak-finding completed, number of image pairs processed so far = 5

NODE rosPeak ITERATION 5 RUN TIME = 3.9994515

Finding correlation surface peaks ...
Mean time interval between frames = 1 s
Peak-finding completed, number of image pairs processed so far = 6

NODE rosPeak ITERATION 6 RUN TIME = 3.9931917

Finding correlation surface peaks ...
Mean time interval between frames = 1 s
Peak-finding completed, number of image pairs processed so far = 7

NODE rosPeak ITERATION 7 RUN TIME = 3.9979858

Finding correlation surface peaks ...
Mean time interval between frames = 1 s
Peak-finding completed, number of image pairs processed so far = 8

NODE rosPeak ITERATION 8 RUN TIME = 3.9963448
```

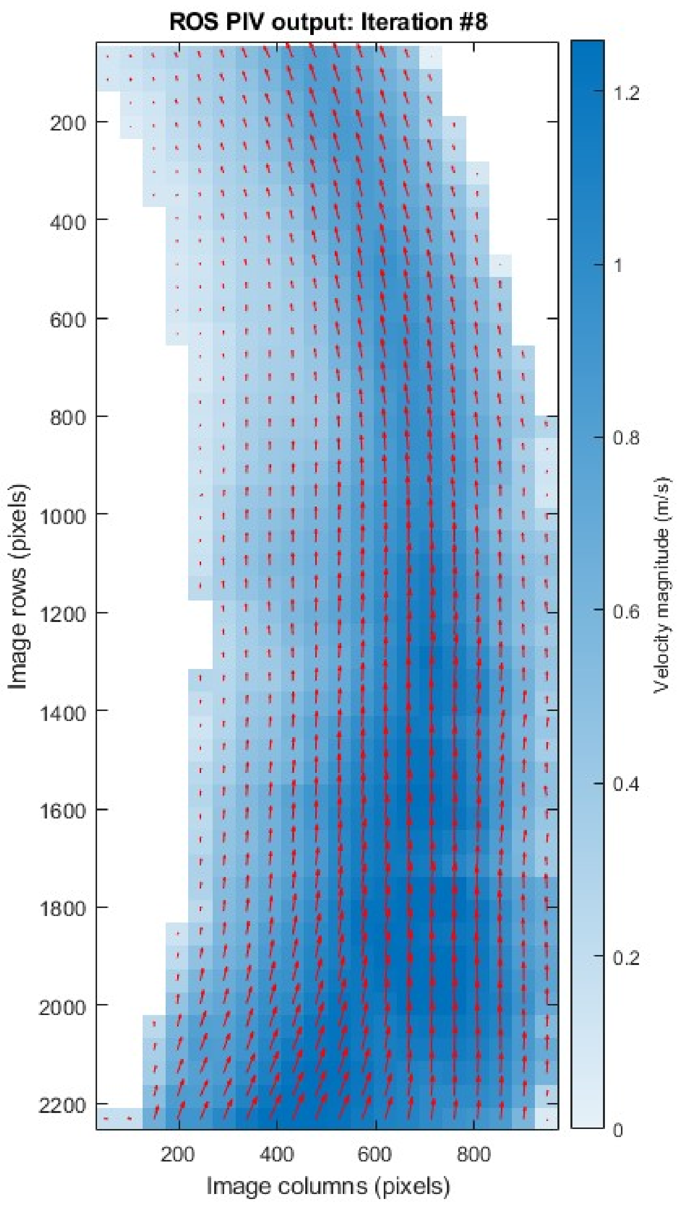
```
Reset 411 v components that failed standard deviation check (too low) to NaN
Reset 0 v components that failed standard deviation check (too high) to NaN
**** Infilling gaps in initial output from ensemble PIV algorithm ****
**** Smoothing initial output from ensemble PIV algorithm ****
Exporting infilled and smoothed output
Vector post-processing completed, number of image pairs processed so far = 7

NODE rosPostProc ITERATION 7 RUN TIME = 3.9981097

**** Filtering initial output from PIV algorithm ****
Reset 4 vectors below minimum velocity threshold to NaN
Reset 4 vectors above maximum velocity threshold to NaN
Reset 1 u components that failed local median check to NaN
Reset 1 v components that failed local median check to NaN
Reset 0 u components that failed standard deviation check (too low) to NaN
Reset 0 u components that failed standard deviation check (too high) to NaN
Reset 438 v components that failed standard deviation check (too low) to NaN
Reset 0 v components that failed standard deviation check (too high) to NaN
**** Infilling gaps in initial output from ensemble PIV algorithm ****
**** Smoothing initial output from ensemble PIV algorithm ****
Exporting infilled and smoothed output
Vector post-processing completed, number of image pairs processed so far = 8

NODE rosPostProc ITERATION 8 RUN TIME = 3.9963287
```

8. To view the output in MATLAB, first enter the following at the MATLAB command prompt, with the last one just to confirm communication with the ROS master:

```
setenv("ROS_HOSTNAME","localhost");
setenv("ROS_MASTER_URI",'http://localhost:11311');
setenv('ROS_IP','127.0.0.1');
rosinit
rostopic list
```

9. Then, specify the name of the ROS topic to which the PIV output is being published and run the helper function `showVectors.m` to read in the PIV-derived velocity field, plot the output as shown below, and save the results to a set of four variables in your MATLAB workspace:

```
pivTopic  =   '/pivFilt'; % for filtered output after running rosPostProc
[xGrid,yGrid,uGrid,vGrid] = showVectors(pivTopic);
```



5. ROS-based PIV for a recorded bag file from the Sacramento River

1. To perform a second PIV run for a different input data set based on real images from the Sacramento River, we will need to update the parameter file `PivParams.yaml` to reflect the new input data. To do so, follow the instructions in cell #5 of `demoScript.m` to create a new parameter file for the Sacramento River image sequence using the function `pivEntry.m`.

2. Before proceeding, kill the current ROS master by pressing `Ctrl+C` in the ROS terminal window where the ROS master is currently running.

3. You will then need to copy the resulting `PivParams.yaml` file to `C:\catkin_ws\src\streamflowpiv` and then load these parameters into the ROS server by issuing these commands in the ROS terminal that will be used to run the ROS master:

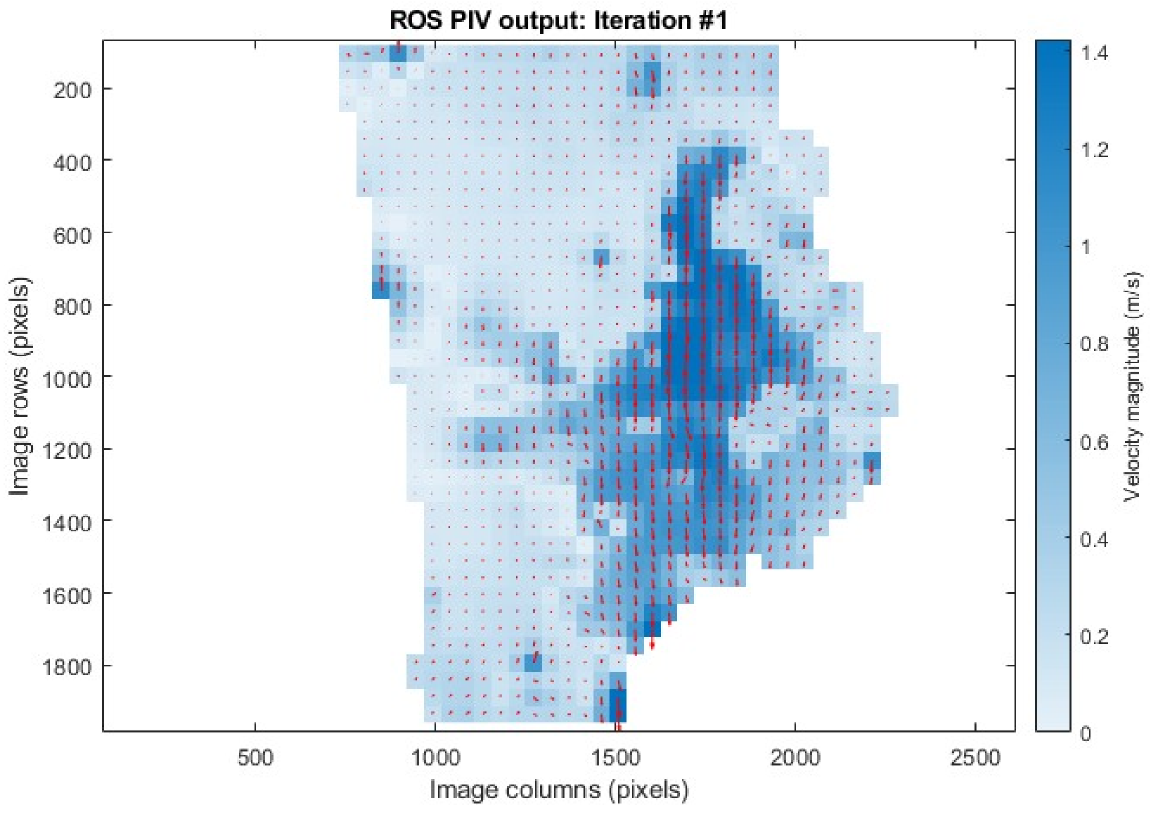
```
cd C:\catkin_ws
devel\setup.bat
roslaunch streamflowpiv streamflowpiv.launch
rosparam get /
```

You should see the new values of each parameter displayed in the terminal window.

4. In this case, performing a PIV run will require two additional steps that were not necessary when working with simulated images that were "PIV ready." Rather than the `rosPivEntry` node we used for the image sequence derived from SHIVER, you will need to use the `rosReadRaw` node to read in the raw images from the bag file and then the `rosImgPrep` node to mask, stabilize, and enhance the images to prepare them for PIV. To accommodate the extra node, you will now need a total of nine separate ROS terminals: one for the ROS core, one to play the bag file, and one for each of eight ROS nodes.

5. Please refer to the instructions in Section #4 above for how to set up the ROS terminals and then run each node in turn. Now, the sequence of nodes to be run becomes: `rosReadRaw`, `rosImgPrep`, `rosGetIA`, `rosMergeFFT`, `rosEnsemble`, `rosPeak`, and `rosPostProc`.

6. Steps 6-9 from Section #4 also pertain here and provide guidance on how to perform the PIV run and view the output in MATLAB.



6. Using the ROS PIV network with a Linux-type operating system

1. Although the preceding sections described how to set up and use the ROS PIV network on a Windows desktop computer, the same network can also be used on a Linux-type operating system. The following instructions are based on a Linux virtual machine (VM) running Ubuntu 20.04.3 LTS on a Windows 10 host computer, but the same steps should also work on a dedicated Linux computer.
2. However, because the Linux (virtual) machine might not have MATLAB, an additional, platform-independent plotting utility is provided in the form of a Python script named `rosPyShowVectors.py`. This script can be run from a Linux terminal and will read in the PIV output from a ROS topic and then plot the results. To make use of this plotting routine ...

   1. Copy the `rosPyShowVectors.py` file to a new folder `C:\catkin_ws\src\streamflowpiv\scripts\` on the Windows machine.

   2. Open a git bash (i.e., Linux-like) terminal and use these commands to make the script executable:

   ```
   cd /c/catkin_ws/src/streamflowpiv/scripts
   chmod +x rosPyShowVectors.py
   ```

   3. Open the file `C:\catkin_ws\src\streamflowpiv\CMakeLists.txt` and add the following at the end:

   ```
   # TACK THIS ON TO MAKE USE OF A PYTHON SCRIPT:
   catkin_install_python(PROGRAMS scripts/rosPyShowVectors.py
   DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
   ```

3. The `addRosNode.m` function described above in Section #3, along with the `catkin_make` command issued in a ROS terminal, serve to generate C++ code from the MATLAB source code and then build the nodes in the catkin workspace for the ROS package, named `streamflowpiv` in this case. Once all the nodes have been built in this manner, the entire catkin workspace can be copied to a Linux (virtual) machine with the same ROS installation and then built using `catkin_make`. More specifically, ...

   1. Copy the contents of the `C:\catkin_ws\src` directory on the Windows machine to the `~/catkin_ws/src` directory on the Linux (virtual) machine.

   2. In a Linux terminal, issue the following commands to build all the nodes at once:

   ```
   cd ~/catkin_ws
   catkin_make
   source devel/setup.bash
   ```

4. To then perform a ROS-based PIV run on the Linux (virtual) machine ...

   1. First start a ROS master and load the appropriate version of the `PivParams.yaml` file using these commands:

```
cd ~/catkin_ws/src/streamflowpiv/
rosparam load PivParams.yaml
rosparam get /
```

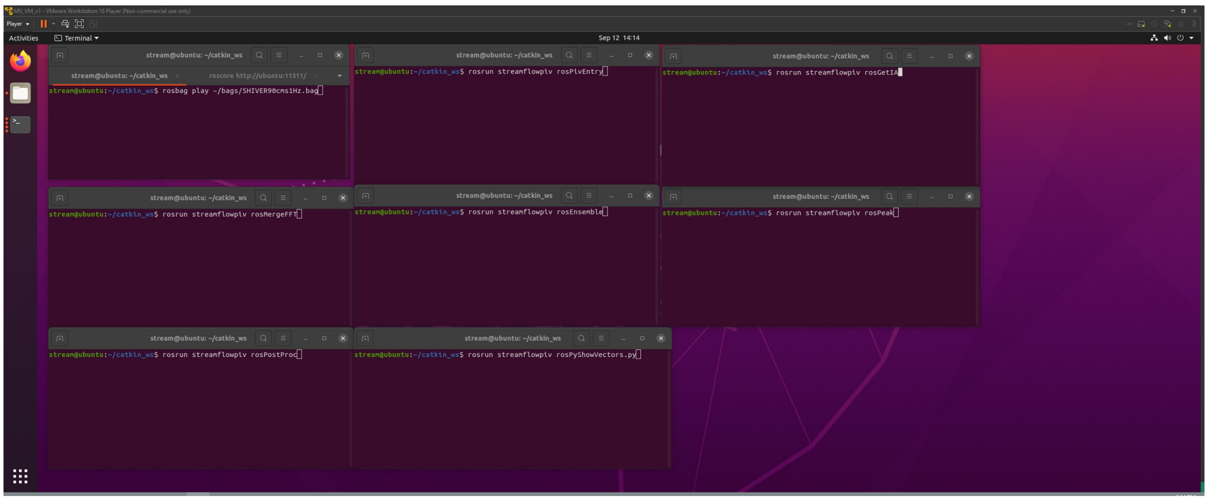
2. Next, play one of the bag files, either the one with simulated images produced via SHIVER or real data from the Sacramento River, using one of these commands:

```
rosbag play ~/bags/bag6xs600.bag
rosbag play ~/bags/SHIVER90cms1Hz.bag
```

3. Open up a series of additional terminals (seven for the simulated image sequence or eight for the real bag file from the Sacramento River) and in each one issue these commands

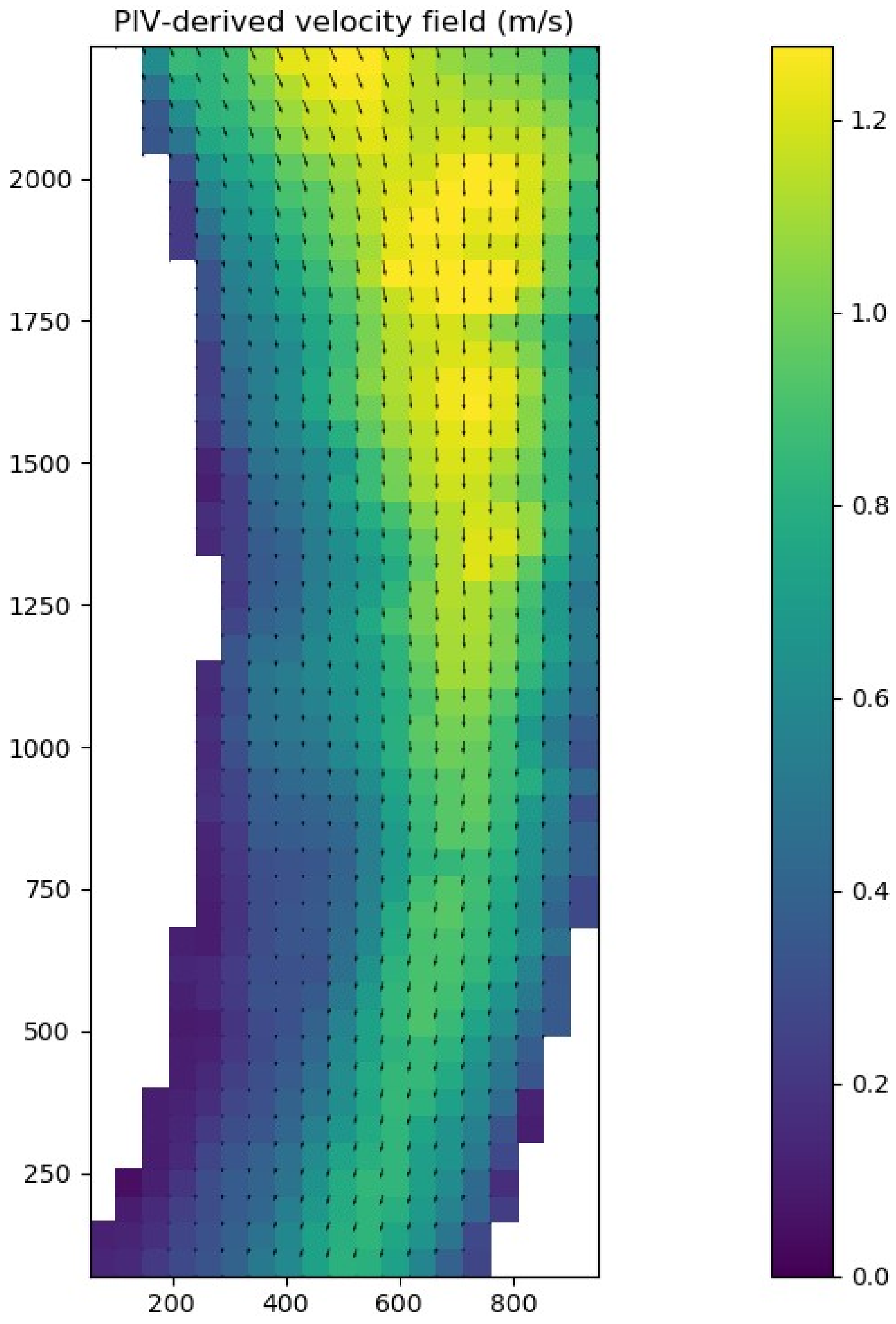
```
cd ~/catkin_ws
source devel/setup.bash
rosrun streamflowpiv [nodeName]
```

Here, the variable where the `[nodeName]` variable takes on the following values: `rosPivEntry` for SHIVER output or `rosReadRaw` followed by `rosImgPrep` for the recorded bag file, then `rosGetIA`, `rosMergeFFT`, `rosEnsemble`, `rosPeak`, `rosPostProc`, and `rosPyShowVectors.py`.


MV_VM_v1 - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Sep 12 14:14
stream@ubuntu: ~/catkin_ws
roscore http://ubuntu:11311/

```
stream@ubuntu:~/catkin_ws$ rosbag play ~/bags/SHIVER90cms1Hz.bag
stream@ubuntu:~/catkin_ws$ rosrun streamflowpiv rosPivEntry
stream@ubuntu:~/catkin_ws$ rosrun streamflowpiv rosGetIA
stream@ubuntu:~/catkin_ws$ rosrun streamflowpiv rosMergeFFT
stream@ubuntu:~/catkin_ws$ rosrun streamflowpiv rosEnsemble
stream@ubuntu:~/catkin_ws$ rosrun streamflowpiv rosPeak
stream@ubuntu:~/catkin_ws$ rosrun streamflowpiv rosPostProc
stream@ubuntu:~/catkin_ws$ rosrun streamflowpiv rosPyShowVectors.py
```

An example of the output from this process, based on the simulated image sequence, is shown below. Press `CTrl+C` in the terminal used for the Python plotting routine to kill the process and return to the Linux command prompt once the run has completed and no more output is being produced by the ROS PIV network.


PIV-derived velocity field (m/s)