**Supporting Information**

US-Pro: An application enabling efficient, high-throughput, ultrasound video processing

AD Haimovich, Z Lehmann, RA Taylor

[Supplementary methods 2](#_Toc507343938)

[Installation Instructions 2](#_Toc507343939)

[Executing US-Pro 2](#_Toc507343940)

[Preparing a directory 2](#_Toc507343941)

[Mounting a directory in Docker 2](#_Toc507343942)

[Complete run command 2](#_Toc507343943)

[Example macOS 3](#_Toc507343944)

[Example Windows 10 Home 3](#_Toc507343945)

[Using the GUI to process data 3](#_Toc507343946)

[Navigating to the US-Pro GUI 3](#_Toc507343947)

[Checking directory 4](#_Toc507343948)

[Output video files 4](#_Toc507343949)

[Output information 4](#_Toc507343950)

[Output data structures 4](#_Toc507343951)

[HDF5 4](#_Toc507343952)

[Pickle 5](#_Toc507343953)

# Supplementary methods

## Installation Instructions

Docker is a well-supported platform and can be installed according to directions found on their website. The work described here was tested on Docker Community Edition for macOS High Sierra, Windows 10 Home Edition, and Windows 8. At the time of testing, Windows 10 Home Edition and Windows 8 had distinct installation instructions relating to the use of Docker Toolbox. In general, we recommend running the Docker provided “hello-world” application prior to moving forward with US-Pro to ensure your Docker environment is working properly.

The latest US-Pro image can be downloaded directory from the Docker repository at: adhaimo/ultrasound:

docker pull adhaimo/ultrasound

This step is not required for installation as the download will occur automatically when docker run is first executed in subsequent sections.

The image will then appear under the docker image list:

docker image ls

To remove the image, use the *docker rmi* command.

# Executing US-Pro

## Preparing a directory

US-Pro uses directories as surrogates for classes in data processing. As a result, the input data formatting is a critical step. The directories must appear with the following formatting - an example will follow. Note that the text provided below is a placeholder intended only to show hierarchy, while curly brackets ( {} ) indicate that multiple files or folders may exist. Italicized text is intended as a placeholder and will require user entry.

/*main\_data\_directory*/{class folders}/{files}.dcm

## Mounting a directory in Docker

When executing the US-Pro Docker container, it is necessary to mount the data-containing directory to the container using the “-v” command. The mount destination must be a folder called “/data”. Without this step, the Docker container will not be able to access user data.

-v /*main\_data\_directory*:/data

## Complete run command

docker run –it –d –p 8000:8000 -v /*main\_data\_directory*:/data --name uspro

adhaimo/ultrasound

Explanation of tags:

d – Run in detached mode – the command line will return.

p – Enables port forwarding so the GUI is readable on-screen.

rm – deletes the container when the process is completed.

name - an optional tag that makes it easier to close the Docker container.

## Example macOS

For this example, the user has data contained in the following folders.

/Users/Researcher\_1/Ultrasound\_data/{folders containing data}/

US-Pro will not process any DICOM files found in Ultrasound\_data as the use of sub-directories is required. Within the Ultrasound\_data folder, the user has a number of folders that contain ultrasounds. Each folder represents a different phenotype under study (*e.g.*, “Pleural\_effusion” and “No\_pleural\_effusion”). In this example, the following directories contain DICOM files:

/Users/Researcher\_1/Ultrasound\_data/Pleural\_effusion

/Users/Researcher\_1/Ultrasound\_data/No\_pleural\_effusion

Successful execution of this example would require the following mount command:

-v /Users/Researcher\_1/Ultrasound\_data:/data

The user can learn the absolute path of a given folder in macOS by dragging the folder into the Terminal application available under the Utilities folder in Applications.

To complete this example, the full run command for macOSX would be:

docker run –d --rm –p 8000:8000 -v /Users/Researcher\_1/Ultrasound\_data:/data --name uspro adhaimo/ultrasound

Note the above line does not include carriage returns.

## Example Windows 10 Home

The installation of Docker on machine running Windows 10 Home is notably different. At the time of writing, both the Oracle VM Virtual Box and Kitematic applications were required to run Docker. This process is explained in detail in the Docker documentation. The critical step in running this application in Windows 10 Home is identifying the port.

## Terminating a docker container

Docker containers can be terminated with the *docker kill* command followed by the name of the container.

docker kill uspro

Note, if no name was assigned using the --name option, the user will need to identify the container using docker ps prior to termination.

# Using the GUI to process data

## Navigating to the US-Pro GUI

The US-Pro GUI is hosted as a local webpage. To access, the user opens a preferred web browser and navigates to the following:

<http://localhost:8000/welcome>

User changes to the port command (-p) described in the previous section should be followed here as well. For example, for systems running a virtual machine (e.g., Windows 8, Windows 10 Home), a virtual-machine specific web address may be required. For example, if the virtual machine is hosted at 192.168.99.100, the site will appear at: <http://192.168.99.100:8000/welcome>.

## Checking directory

We have included a directory checking function that enables the user to confirm mounting. This function is accessed through the US-Pro home page. We recommend using this function prior to any subsequent analysis.

# Output video files

## Output information

Video files will be saved in a new directory created by the program called “Videos”. The directory location will be within the main data folder. For organization within the Videos directory, there will be subdirectories that match the class names and the filenames will share the base with the original DICOM file (e.g., “Patient\_A.dcm” will be saved as “Patient\_A.avi”).

We recommend using this feature on a select number of video files prior to producing large HDF5 or pickle files. This is the easiest way to visualize the effects of preprocessing on a dataset.

Videos may not open correctly with default operating system. An open source media player such as VLC (<https://www.videolan.org/index.html>) may provide better results.

# Output data structures

## HDF5

Once opened using the *h5py* python library, HDF5 files are automatically decompressed. The video data are stored by class, *e.g.*, a two-class dataset will have data in “0” and “1”. Video data are stored with the following dimensions:

Total frames for a given class x rows x columns

Since it may be important to the user to keep frames from a single video together, frame counts are stored in a separate vector named with the following convention “{class number}\_frames” where {class number} is an integer starting with 0 and counting up. The integer values of the frame vector describes contiguous frames such that if the first value of the vector is 10, the first 10 rows of the matrix “0” are from a single video. The dimensions of this vector is:

number of videos in the class directoryx 1

Example:

Data have been stored in a file out.hdf contained in the directory “/Users/Researcher\_1/Ultrasound\_data”.

import h5py

f = h5py.File(‘Users/Researcher\_1/Ultrasound\_data/out.hdf’,’r’)

To list all datasets available in that savefile:

print('Keys {}'.format(list(f.keys())))

To access the data from class 0:

f[‘0’][:]

To access the corresponding frame counts:

f[‘0\_frames’][:]

Further examples are included in the github directory.

## Pickle

Pickle files are not compressed. A single run will yield a single pickle file which contains a python dictionary. In this case, the dictionary key-value pairs are the video class, an ascending integer starting at zero, and a list of matrices. Each list entry is a matrix representing the total data output of a single video. The number of list entries is equivalent to the number of DICOM videos contained within the class directory. Each pickle file contains a key “num\_classes” whose value indicates the total number of processed classes. No frames variable is required since the DICOM videos are already separated as elements of the list. Each matrix has the following dimensions:

Frames for a given video x rows x columns

Example:

Data have been stored in a file out.p contained in the directory “/Users/Researcher\_1/Ultrasound\_data”.

import pickle

dataset = pickle.load(open(‘Users/Researcher\_1/Ultrasound\_data/out.p’, ‘rb’ ) )

To list all datasets available in that savefile (includes num\_classes variable):

for key, value in dataset.items(): print (key)

To access the data from class 0:

class\_0 = dataset[“0”]

To access the data contained in the first video file from class 0:

first\_video\_file = dataset[“0”][0]