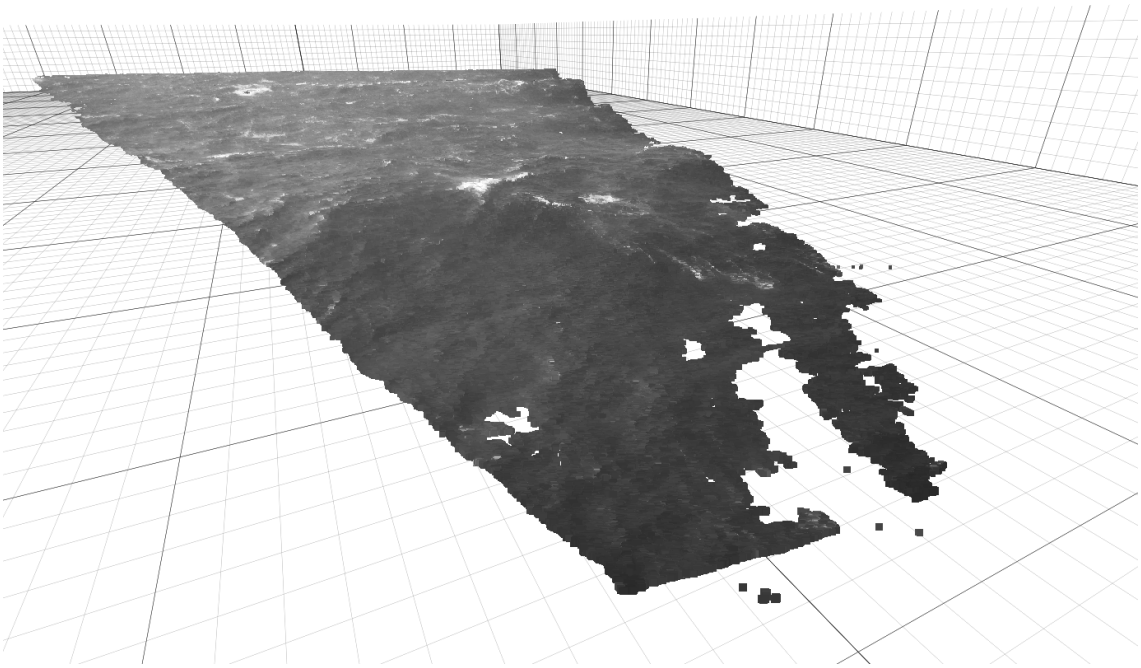
WASS and WASSfast Quick-Start Guide

Filippo Bergamasco

April 19, 2024



Contents

Version history

2024.04.18 Initial release

Links

WASS GitHub page:
https://github.com/fbergama/wass
wasscli:
https://pypi.org/project/wasscli
wassgridsurface:
https://pypi.org/project/wassgridsurface
wassncplot:
https://github.com/fbergama/wassncplot
wassfast:
https://gitlab.com/fibe/wassfast

1 Introduction

Introduced in 2017, WASS (Waves Acquisition Stereo System) was the first open-source pipeline for the 3D reconstruction of sea waves. Nowadays, improved with recent additions like WASSfast, wassgridsurface, and wassncplot it is used by several researchers and practitioners worldwide to analyze sea waves in their space and time dimensions. This document serves as a practical guide to get started with this technology.

1.1 Architecture

WASS comprises a set of program executables executed in sequence to perform different steps of the reconstruction process (that's why it is called a pipeline). Some programs are written in C++ and must be compiled before usage. Some others are in Python and require a pre-configured Python environment. Here is a brief list of all such components:

wass_prepare Takes a couple of stereo images as input and places them into a proper working directory. Images are also undistorted according to the provided intrinsic calibration. (Executable)

wass_match Matches corresponding features of a stereo image pair. After the matching, the Essential matrix is recovered, and the extrinsic parameters (rotation and translation) between the two cameras are defined. In doing so, matched features are filtered according to epipolar consistency with a game-theoretic approach. (Executable)

wass_autocalibrate Matches between multiple stereo frames are all collected. Sparse Bundle Adjustment (SBA) is used to optimize the extrinsic camera parameters simultaneously, and the 3D points triangulated from all the matches. Final extrinsic parameters are saved to all the working directories. (Executable)

wass_stereo Performs dense stereo 3D reconstruction on a given stereo pair. After the reconstruction, multiple filters are applied to the point cloud to remove outliers. Finally, a plane is robustly fitted to the point cloud. (Executable)

wasscli A command line interface to run the WASS executables in the correct order on an image sequence. It is the primary tool for executing the whole pipeline from image undistortion to point cloud generation. (Python script)

wassgridsurface Transforms each point cloud to a common (mean) sea plane and interpolates it on a user-defined regular grid. The output is a NetCDF file storing the sea surface elevation over time. (Python script)

wassncplot Primarily used to render the sea surfaces stored in the NetCDF file back to the original images for visual evaluation. Additionally, it can produce the mapping between each image pixel and the corresponding 3D point of the sea surface. (Python script)

wassfast An alternative approach for sea waves reconstruction focused on speed more than accuracy. It takes input stereo frames in input and directly produces a NetCDF file with no intermediate step. To be used, the mean sea plane must be estimated a priori. See §4 for usage instructions. (Python script)

Given a sequence of stereo frames, the typical procedure involves the following steps:

1. Run `wasscli` and follow the instructions to reconstruct a point cloud for each stereo pair. Each output is stored in a separate working directory;
2. Run `wassgridsurface` to interpolate each point cloud on a regular user-defined grid. The resulting NetCDF file contains the reconstructed surface over time;
3. Run `wassncplot` to see the reconstruction results and/or produce additional files useful for surface analysis (like whitecap detection, etc.);

Alternatively, WASSfast can be used if processing time is a concern. In that case, the wass executables `wass_prepare`, `wass_match`, `wass_autocalibrate`, and `wass_stereo` are not used since `wassfast` will manage the creation of the NetCDF file directly. Nevertheless, the mean sea plane must be provided, so the classic procedure involving the WASS executables must be performed at least once for every new camera installation. Refer to §4 for details on running WASSfast.

2 Installation

2.1 WASS executables

On Windows, the easiest way is to download the precompiled binary version from https://www.dais.unive.it/wass/download.html. Then, unzip the package on a directory of your choice and add the subdirectory `dist\bin` to the system path. For example, if the package was unzipped in `C:\WASS`, add the directory `C:\WASS\dist\bin` to the path. Check if the installation is successful by opening a command prompt and then typing the command `wass_stereo`. You should get an output like the following:

```
wass_stereo  v. 1.11_heads/master-0-g6b82aeb
-----------------------------------------------
 [Release] Windows-10.0.19044 - MSVC, OpenCV 4.6.0

Usage:
wass_stereo [--genconfig] <config_file> <workdir> [--measure] [--rectify-only]

Not enough arguments, aborting.
```

On Linux and OSX it is better to compile the executables from the source code. Before doing so, you need to install some dependencies. For Ubuntu-like variants, use the apt package manager:

```
sudo apt install git build-essential cmake curl liblapack-dev \
libblas-dev libboost-all-dev libopencv-dev
```

On OSX, install homebrew first (https://brew.sh/) and then use the command:

```
brew install cmake boost opencv ffmpeg
```

To build WASS, start by cloning the WASS git repository and prepare it for building:

```
git clone https://github.com/fbergama/wass
cd wass
git submodule update --init
mkdir build
```

then, proceed with the build:

```
cd build
cmake ../src/
make
make install
cd ..
```

To be used with `wasscli`, the wass binaries compiled in `wass/dist/bin` must be added to the shell path. Typically, you might have to change the `PATH` environment variable by modifying `.bashrc` or your shell startup script. For example, assuming that WASS has been cloned into `~/wass` (i.e. your home directory), and the shell you are using is Bash, add the following at the end of `.bashrc`:

```
export PATH=~/wass/dist/bin:$PATH
```

If the path is set correctly, running `wass_stereo` from the shell should output the following:

```
   wass_stereo  v. 1.11_heads/master-0-g78cff07
-----------------------------------------------
 [Release] Linux-6.5.11-linuxkit - GNU, OpenCV 4.5.4

Usage:
wass_stereo [--genconfig] <config_file> <workdir> [--measure] [--rectify-only]

Not enough arguments, aborting.
```

2.2 Python scripts

To use the WASS components implemented as Python scripts you need a Python interpreter. Even if Python is already installed in your system, it is better to create a dedicated Python virtual environment for WASS and WASSfast. This avoids clashing the required dependencies with other Python modules you might have already installed.

Start by installing the Miniconda https://docs.anaconda.com/free/miniconda/index.html package manager. Please be sure to add conda to the shell path when prompted. After a successful install, your shell prompt should display the text `(base)`, indicating that we are now working in the base environment[1].

When conda is installed, create a new environment named wass:

```
conda create --name wass python=3.11
```

and then activate it:

```
conda activate wass
```

Note how the command prompt now displays `(wass)` instead of `(base)`. At this point, install the WASS Python components by running the following command:

```
python -m pip install --upgrade wasscli wassgridsurface wassncplot
```

If the installation is successful, running the `wasscli` command should output the following:

```
 WASS-cli v. 0.1.4
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Copyright (C) Filippo Bergamasco 2022


Searching for WASS pipeline executables...OK
Current directory is:  /home/fibe
Current directory does not appear as a WASS working directory.
? Current directory is not a WASS working directory. Do you want to initialize it?  (Y/n)
```

type `n` and then select `Quit`.

2.3 WASSfast

At the present state, WASSfast requires a specific version of Python and the Tensorflow library to implement the internal Convolutional Neural Network (CNN) for surface interpolation. Therefore, it is suggested to create a different conda environment than the one used for WASS.

Start by creating a new environment named `wassfast` using Python 3.8:

```
conda create --name wassfast python=3.8
```

then, activate the environment and install WASSfast:

```
conda activate wassfast
python -m pip install --upgrade wassfast
```

At this point, you can test if everything was installed correctly by running the command `wassfast`. The output should be something like:

```
     WASSFAST
     _.~'~._.~'~._.~'~._.~'~._
      v. 1.5.4 - Copyright (C)
                 Filippo Bergamasco 2023

usage: wassfast [-h] [--generate_settings] [--continuous_mode] [--kafka_mode]
                [--debug_mode] [--debug_stats] [--batchsize BATCHSIZE]
                [--start_from_plane] [--demosaic] [--save_polarization]
                [--current_u CURRENT_U] [--current_v CURRENT_V] [--depth DEPTH]
```

[1] On Windows, start the **Anaconda Prompt** from the Start menu to get a command prompt with conda already added to the path

```
                [-dd DEBUGDIR] [-s] [--nographics]
                [-n NFRAMES] [--first_frame FIRST_FRAME] [-r FRAMERATE] [--nfft] [--fft]
                [--upload_url UPLOAD_URL] [--location LOCATION] [--savepts] [-o OUTPUT]
                imgdata configfile calibdir settingsfile wavedirection processingmode
wassfast: error: the following arguments are required: imgdata, configfile, calibdir,
settingsfile, wavedirection, processingmode
```

2.4 Notes about the Python environments

If you have followed the guide so far, you should have installed two different environments. One is named `wass` and the other `wassfast`. An environment acts like a sandbox where Python scripts and modules can be installed without affecting the content of other modules. On the other hand, to use a program installed in a specific environment, you have to activate it first with the command `conda activate <envname>`.

Therefore, before using `wasscli`, `wassgridsurface` or `wassncplot` remember to activate the `wass` environment with the command:

```
conda activate wass
```

similarly, to use `wassfast`:

```
conda activate wassfast
```

At any time, you can deactivate the environment with the command `conda deactivate <envname>`. If a new version is released, you can upgrade the tools by repeating the abovementioned installation steps. In that case, it is unnecessary to create the environments again, just execute the `python -m pip install ...` commands.

3 Using WASS

To reconstruct a new stereo-frame sequence, start by creating a new directory to store the input and output data necessary at each pipeline stage. For the sake of this quick-start guide, let's suppose to use a Linux system and choose `~/tmp/test_sequence` as the place to store the data:

```
mkdir -p ~/tmp/test_sequence
cd ~/tmp/test_sequence
```

Now, activate the `wass` environment and start `wasscli`. Select `Y` when asked to initialize the directory:

```
conda activate wass
wasscli
```

```
 WASS-cli v. 0.1.7
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Copyright (C) Filippo Bergamasco 2022


Searching for WASS pipeline executables...OK
Current directory is:  /home/fibe/tmp/test_sequence
Current directory does not appear as a WASS working directory.
? Current directory is not a WASS working directory. Do you want to initialize it? Yes

WASS working directory initialized. Please:
 1) copy your calibration data intrinsics_00.xml, intrinsics_01.xml,
    distortion_00.xml and distortion_01.xml in the config directory.
 2) copy your images into input/cam0 and input/cam1 directories
```

notice that `wasscli` exits but the `~/tmp/test_sequence` directory have been populated with some directories that will be used by `wass`:

```
tree
```

```
.
   config
      matcher_config.txt
      prepare_config.txt
      stereo_config.txt
   input
      cam0
      cam1
   output
5 directories, 3 files
```

Now, as indicated by `wasscli`, you have to:

1. copy stereo images in the `input/cam0` and `input/cam1` directories respectively
2. copy calibration data into the `config/` directory

For this tutorial, download the test dataset from https://www.dais.unive.it/wass/wass_wassfast_dataset_100frames.zip, extract the package, and locate the stereo images. Copy camera 0 and camera 1 images into the respective directories in `input/cam0` and `input/cam1`. Then, copy `intrinsics_00.xml`, `intrinsics_01.xml`, `distortion_00.xml` and `distortion_01.xml` in the `config/` directory.
After the copy, `~/tmp/test_sequence` should look as follows:

```
tree
```

```
.
   config
      distortion_00.xml
      distortion_01.xml
      ext_R.xml
```

```
      ext_T.xml
      intrinsics_00.xml
      intrinsics_01.xml
      matcher_config.txt
      prepare_config.txt
      stereo_config.txt
   input
      cam0
         000000_0000000000000_01.jpg
         000001_0000000000199_01.jpg
         000002_0000000000399_01.jpg
         000003_0000000000599_01.jpg
         000004_0000000000800_01.jpg

        ...

         000045_0000000009005_01.jpg
         000046_0000000009205_01.jpg
         000047_0000000009405_01.jpg
         000048_0000000009605_01.jpg
         000049_0000000009805_01.jpg
      cam1
          000000_0000000000000_02.jpg
      cam1
          000000_0000000000000_02.jpg
          000001_0000000000199_02.jpg
          000002_0000000000399_02.jpg
          000003_0000000000599_02.jpg
          000004_0000000000800_02.jpg

        ...

          000045_0000000009005_02.jpg
          000046_0000000009205_02.jpg
          000047_0000000009405_02.jpg
          000048_0000000009605_02.jpg
          000049_0000000009805_02.jpg
   output

5 directories, 109 files
```

Notes:

1. In this example, images are provided in JPEG format, offering good compression at the price of losing some detail and possibly introducing unwanted artifacts. For best results, use either TIFF or PNG formats.

2. Camera intrinsic calibration must be performed before running WASS. There are several alternatives to that. The most straightforward is to use the Matlab Camera Calibrator or the Bouget camera calibrator toolbox for Matlab. Once calibrated,manually copy the values of the Matlab variable `KK` into the data section of `intrinsics_00.xml` (or `intrinsics_01.xml` depending on which camera you are calibrating) and the values of the variable `kc` into the data section of `distortion_00.xml` (or `distortion_01.xml` for the second camera).

3. Image frame filenames follow the convention of using 6 digits for the frame number, 13 digits for the timestamp (usually the milliseconds from the acquisition start), and 2 digits for the camera number (either 01 or 02). Please rename your images if you are using a different convention.

```
<frame number (6)>_<timestamp (13)>_<camera number (2)>.tif
```

3.1 From stereo frames to point clouds

To compute the point clouds, the reconstruction process involves the execution of `wass_prepare`, `wass_match`, `wass_autocalibrate` and `wass_stereo` in this order. Everything is managed by

`wasscli` by selecting the appropriate menu entry. Start by running `wasscli` again, select `Prepare`, and press `ENTER`. Then, choose how many images to prepare (usually you select all of them but you may want to process a shorter sequence). Since we don't have polarimetric images, select N when prompted for demosaicing.

```
 WASS-cli v. 0.1.7
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Copyright (C) Filippo Bergamasco 2022


Searching for WASS pipeline executables...OK
Current directory is:  /home/fibe/tmp/test_sequence
? What do you want to do? Prepare
Checking calibration files...
Checking input/cam0 and input/cam1...
50 stereo pairs found!
? How many stereo frames do you want to prepare? (3 ... 50) 50
? Should I attempt to demosaic polarimetric images? No
Running wass_prepare... please be patient
```

When the prepare process is completed, you'll be prompted again with the selection menu:

```

Prepare completed!
? What do you want to do?
  Prepare
   Match
   Autocalibrate
   Stereo
   ---------------
   Set number of parallel workers
   Quit
```

The prepare process created a set of directories `output/000000_wd`, `output/000001_wd`, etc. In each one, the `undistorted/` subdirectory contains the two undistorted stereo images used for the next steps of the pipeline.

Let's continue by selecting `Match` in the `wasscli` menu prompt and press `ENTER`. You will be asked to select how many frames to Match. For a quick test, you can choose just 10. In general, it is suggested to use 50 frames.

```
? What do you want to do? Match
? How many frames do you want to match? (1 ... 50, suggested: 50) 50
Matcher will use the following frames:
[48  1 20 21 25 16 28 19  6 30  4  8 15  7 42  2 23 35 37 33 11 13 44  9
 12  3 10 31 29 43  0 26 18 38 14 49 32 34 39 27 46 36 41 47 45 22 40 24
 17  5]
Running wass_match... please be patient
```

During the matching, you will see some output. Check that every stereo frame produces a fair amount of matches passing the chirality check. You should expect at least 100 matches here, with an epipolar error of at most 1.0:

```
GTMatcher [info ] 262 matches have passed the chirality check
GTMatcher [info ] Epipolar error: 0.33841+-0.211403 px. Min: 1.70965e-13 Max: 0.703118
```

Once the match is completed, it's time for the final extrinsic calibration. Select `Autocalibrate` in `wasscli` and press `ENTER`. After a few seconds, the operation is completed and everything is ready for stereo reconstruction. Before continuing, check the output lines:

```
sba_driver [info ] Final SBA-optimized R/T pair:
sba_driver [info ] R:
[0.9984428249545942, -0.03160410782267195, 0.04596852907612378;
 0.03154104193952011, 0.9995002581760439, 0.002096802182178232;
```

```
 -0.04601182424180707, -0.000643641790656914, 0.9989406878064314]
sba_driver [info ] T:
[-0.9993563333960376;
 -0.03584910519252846;
 0.001326860249844967]
sba_driver [info ]  SBA-optimized epipolar error: 0.357962+-0.270613 px.
                    Min: 4.08768e-05 Max: 2.29965
```

containing the estimated $\mathbf{R}, T$ and the SBA-optimized epipolar error. Also in this case, the lower the better. Generally, an error below 0.5 px. is recommended for an accurate reconstruction.

Stereo reconstruction

Now that the camera position has been recovered, it is possible to continue with the stereo reconstruction. This step requires adjusting some parameters in `config/stereo_config.txt` to get the optimal results. This file is already populated with the default values, which offers a fair starting point. However, expect to use some try-and-error here because every setup is slightly different and requires some tweaking.

In `wasscli` select `Stereo`. Then, select `N` to reconstruct `000000_wd` only. The idea is to check the result, modify `config/stereo_config.txt` accordingly, and then try again. To check if the settings were correctly set, open the directory `~tmp/test_sequence/output/000000_wd`. The important files to consider are the following:

`mesh_cam.xyzC` The reconstructed point cloud in WASS custom format

`mesh.ply` The reconstructed point cloud in PLY format to be opened with the free tool Meshlab https://www.meshlab.net/

`stereo.jpg` Stereo-rectified frames useful to check if the extrinsic parameters have been correctly calibrated

`stereo_input.jpg` Stereo frames stacked vertically, useful to determine the disparity offset

`disparity_final_scaled.png` Disparity map produced by the dense stereo algorithm

`graph_components.jpg` Connected components of the reconstructed point cloud

`undistorted/R0.jpg` Camera 1 image rendered to the camera 0 reference frame. Useful to debug masked regions, erroneous pixels, etc.

`undistorted/R1.jpg` Camera 0 image rendered to the camera 1 reference frame. Useful to debug masked regions, erroneous pixels, etc.

`plane.txt` Mean sea plane estimated from this stereo pair

`P0cam.txt` Camera 0 projection matrix mapping points from 3D to cam0 image plane

`P1cam.txt` Camera 1 projection matrix mapping points from 3D to cam1 image plane

`00000000_s.png` Scaled version of the cam0 frame (see also scale.txt)

`00000001_s.png` Scaled version of the cam1 frame (see also scale.txt)

`scale.txt` Scale factor of the images `00000000_s.png` and `00000001_s.png`

If the reconstruction is not good on the first try, don't worry. It is probably because the disparity offset has not been correctly set. In this example, we can observe from `disparity_final_scaled.png` that the top part of the image was wrongly reconstructed. This is because images must be shifted (left/right) so that all the pixels to reconstruct on the left camera are at the right of the corresponding pixels on the right camera. To check that, open `stereo_input.jpg` and look at the points far from the camera. There are some points on the top image that are on the right with respect to the points on the bottom image. To correct this, open `~/tmp/test_sequence/config/stereo_config.txt`, and modify the line:

```
# Offset in pixel to be applied. Positive: move right image to the right.
# Negative: move right image to the left
#
#DISPARITY_OFFSET=0
```

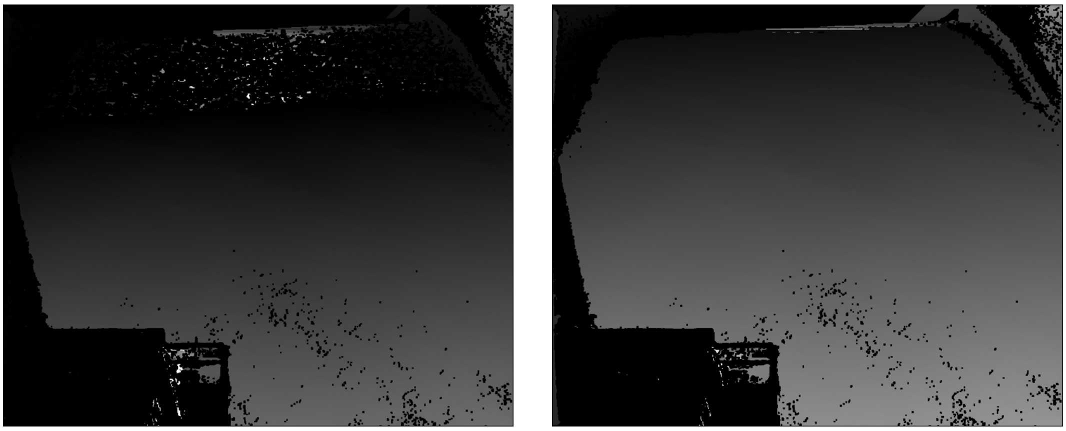


Figure 1: Disparity map before (left) and after (right) adjusting the DISPARITY_OFFSET parameter.

to

```
# Offset in pixel to be applied. Positive: move right image to the right.
# Negative: move right image to the left
#
DISPARITY_OFFSET=-100
```

Now, in `wasscli`, select `y` to try again and reopen `stereo_input.jpg`. You can notice that now the bottom image has moved 100 pixels to the right (Fig. 2), and the disparity map (`disparity_final_scaled.png`) now shows good values on the upper part of the image (Fig. 1). To visually observe the point cloud with Meshlab, set:

```
# Save final reconstructed point cloud also in PLY format
#
SAVE_AS_PLY=true
```

in `stereo_config.txt` and reconstruct again to produce the file `mesh.ply` (Fig. 3).

3.2 From point clouds to surfaces

After producing a point cloud for each stereo pair, it is convenient to interpolate them over a regular grid. `wassgridsurface` is the tool designed to produce a NetCDF containing the interpolated surface for each frame. The execution is divided into two phases. In the `setup` phase, the grid is defined and the mean sea plane is computed by averaging the individual sea planes for each stereo pair. In the `grid` phase, the surface is estimated by interpolating each point cloud.

Start by creating a directory named `gridding` in `~/tmp/test_sequence` and then run `wassgridsurface` to generate a default grid config file:

```
cd ~/tmp/test_sequence
mkdir gridding
wassgridsurface --action generategridconfig . gridding
```

```
 WASS surface gridder v. 0.6.0
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Copyright (C) Filippo Bergamasco 2022

Generating  gridding/gridconfig.txt
All done, exiting.
```

The file `gridding/gridconfig.txt` is generated with some default grid settings:

```
more gridding/gridconfig.txt
```

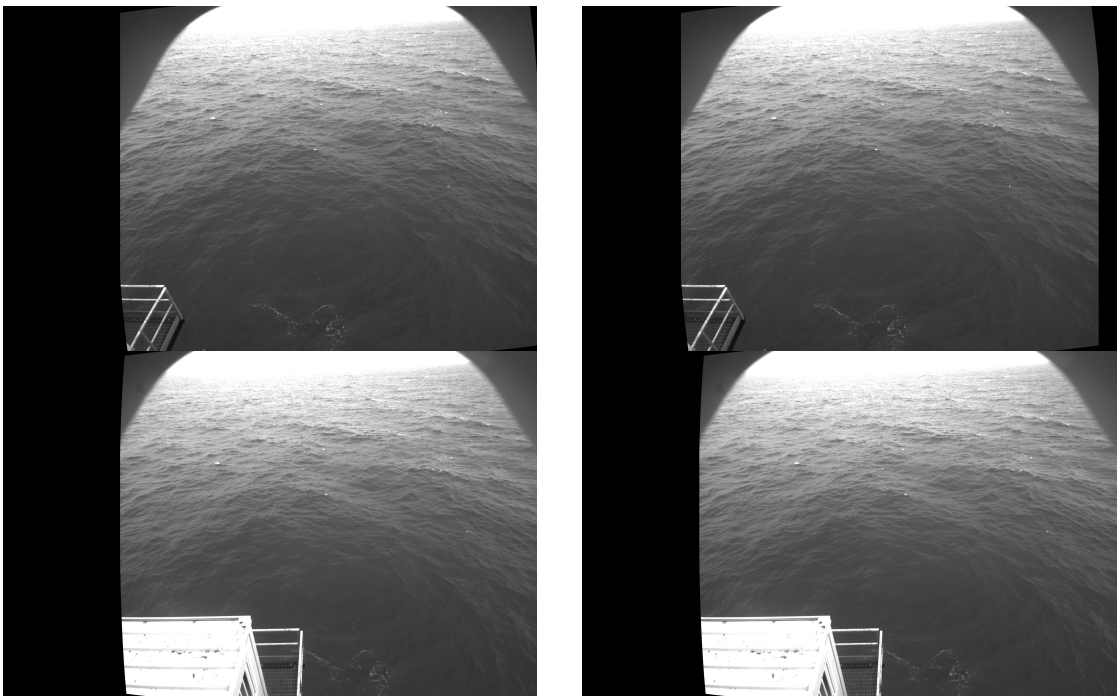


Figure 2: The file stereo_input.jpg shows the right image stacked on top of the left image to adjust the DISPARITY_OFFSET parameter visually. For a correct reconstruction, every point in the top image should be at the left of the corresponding point in the bottom image. Left: alignment with DISPARITY_OFFSET=0. Right: alignment with DISPARITY_OFFSET=100.

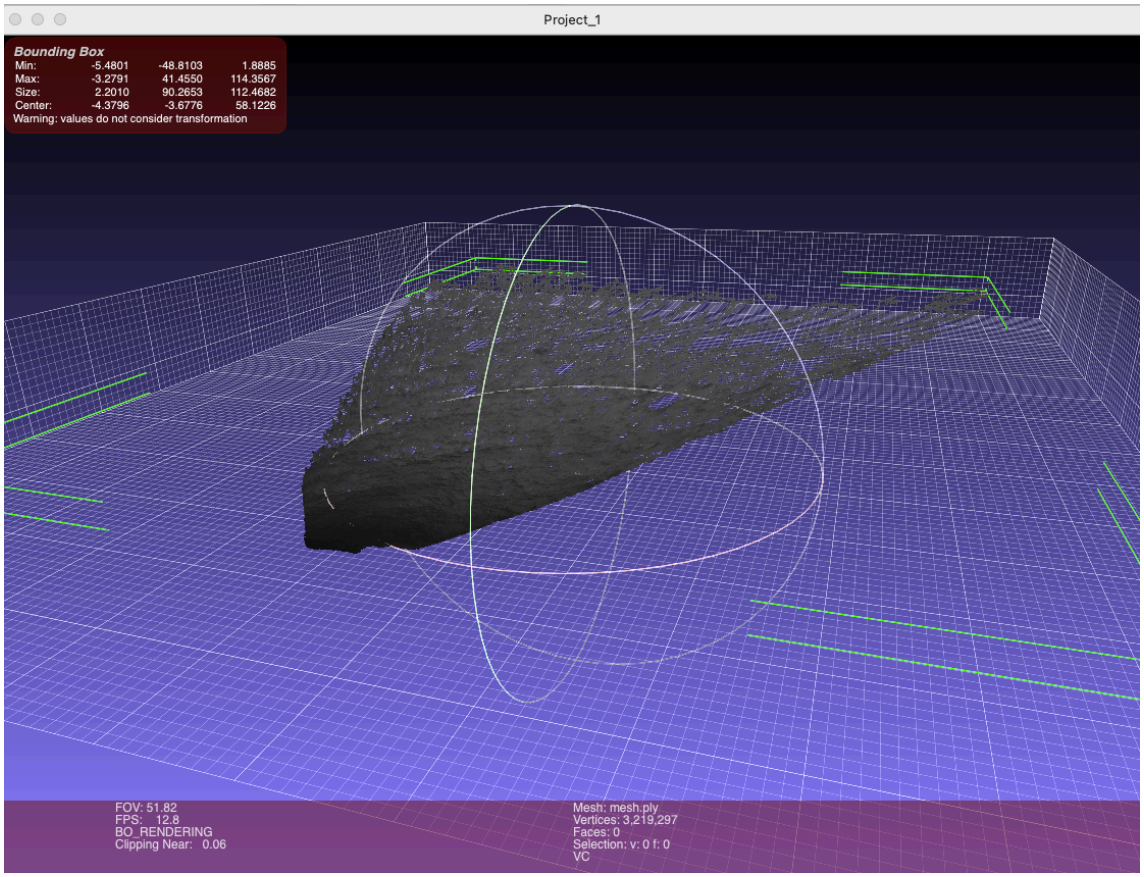


Figure 3: Example of a reconstructed point cloud visualized with Meshlab

```
[Area]
area_center_x=0.0
area_center_y=-35.0
area_size=50
N=1024
```

Now, let's proceed with the setup phase to check how the interpolation grid looks like. Run:

```
wassgridsurface --action setup ./output ./gridding \
--gridconfig ./gridding/gridconfig.txt --baseline 3.0
```

```
 WASS surface gridder v. 0.6.0
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Copyright (C) Filippo Bergamasco 2022

Looking for WASS reconstructed stereo frames in  ./output
50 frames found.
Looking planes definition file  ./output
Found! Loading...
Loading grid configuration file: ./gridding/gridconfig.txt
Baseline: 3.00
Loading  ./output/000000_wd/mesh_cam.xyzC
Camera image size: 2456x2058 (override with -Iw -Ih program arguments)
zmin .. zmax = -0.87 ... 0.87
0.048875855327469964
0.04887585532746641
Generating grid area plot...
Please open ./gridding/area_grid.png and check if everything is ok
Saving grid setup to ./gridding/config.mat
```

Note that we had to specify the baseline (i.e., the camera distance) because, from now on, the reconstruction will use the correct scale. Instead, WASS point clouds are up to scale and assume a unitary baseline by default. Open the file `./gridding/area_grid.png` as instructed to check if the reconstructed area satisfies your needs. Figure 4 (left) shows how the area looks like with the default grid parameters. Let's modify some of those by setting:

```
[Area]
area_center_x=0.0
area_center_y=-60.0
area_size=100
N=1024
```

in `gridding/gridconfig.txt` and then run the setup again:

```
wassgridsurface --action setup ./output ./gridding \
--gridconfig ./gridding/gridconfig.txt --baseline 3.0
```

Now, the `area_grid.png` should look like Fig.4 (right), reflecting the changes we've just made. Now that we are satisfied with the settings, we can start the actual grinding process:

```
wassgridsurface --action grid --gridsetup ./gridding/config.mat ./output ./gridding
```

after a while, the file `gridding/gridded.nc` is produced. You can now skip to §5 to learn how to plot the surfaces above the original images.

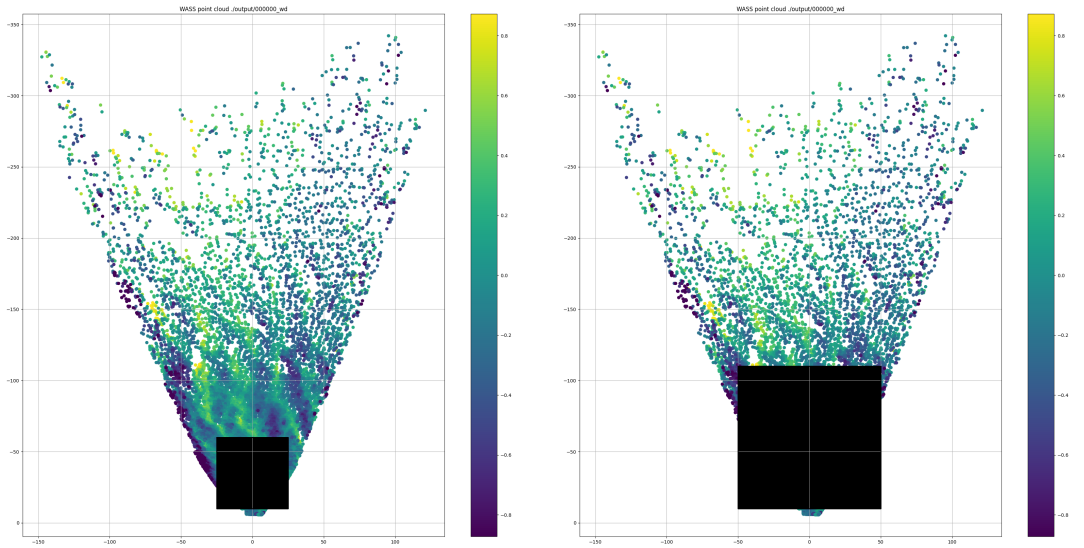


Figure 4: Left: default grid location, showing a square centered at (0,-35) meters from the camera and with a side of 50 m. Right: updated grid centered at (0,-60) and with a side of 100 m.

4 Using WASSfast

If it's the first time you run WASSfast on a sequence, start by entering the working directory you previously used with WASS. Then, activate the wassfast environment, and run `wassfast` to generate a default configuration file:

```
cd ~/tmp/test_sequence
conda activate wassfast
wassfast --generate_settings
```

```
 WASSFAST
_.~'~._.~'~._.~'~._.~'~._
 v. 1.5.4 - Copyright (C)
            Filippo Bergamasco 2023

default_settings.txt generated.
```

The file `default_settings.txt` is generated in the current directory, containing the default settings controlling different parameters of the WASSfast program. For a common sequence like the one in this example, there is no need to change them. Then, you must provide a grid configuration file created during the wassgridsurface setup phase. Currently, WASSfast can only work with grids composed of 256×256 points. If you followed this guide from the beginning, in §3.2 we set an area composed by 1024×1024 points (see line `N=1024` in `gridding/gridconfig.txt`). So, open `gridding/gridconfig.txt` again and change `N=1024` to `N=256`. The resulting file should be as follows:

```
[Area]
area_center_x=0.0
area_center_y=-60.0
area_size=100
N=256
```

Now, rerun the setup phase to produce the correct `gridding/config.mat` file:

```
conda activate wass

wassgridsurface --action setup ./output ./gridding \
--gridconfig ./gridding/gridconfig.txt --baseline 3.0

conda activate wassfast
```

We are now ready to launch `wassfast`. It will directly process the input images and produce a new NetCDF file with the surfaces. The command line is the following[2]:

```
wassfast ./input ./gridding/config.mat ./config \
./default_settings.txt AUTO CNN --batchsize 16 \
-r 15.0 -o gridding/gridded_wassfast.nc
```

After a (relatively) short processing time, the resulting NetCDF file can be found in `gridding/gridded_wassfast.nc`. Additionally, WASSfast produces a simple HTML file `gridding/gridded_wassfast.html` summarizing some basic sea state condition variables.

4.1 Do I need WASS to run WASSfast?

In general, WASS is not needed to run WASSfast. Indeed, WASSfast was born to replace WASS in all the situations in which you want to process your data quickly by sacrificing grid resolution and accuracy. However, WASSfast needs to know the grid configuration (`gridding/config.mat`) in the reference system of the mean sea plane. To obtain that, you'll need to run WASS at least once every time you install or move the cameras to a new location. This also implies that WASSfast cannot be used with cameras mounted on a ship since their relative position to the mean sea plane change at every frame.

To summarize:

- Every time you install or move the stereo cameras to a new location, acquire a short sequence (300 frames in sufficient) and run WASS as described in §.3.2. Remember to set `N=256` when you configure the grid.
- To run WASSfast on a new sequence, just replace the `input/` directory with the new images and run wassfast as described in §.4.

[2]Since WASSfast works by exploiting the linear dispersion relation, it needs to know the framerate of the stereo sequence. You can set it with the -r argument. In the example, the frame rate is 15 Hz.

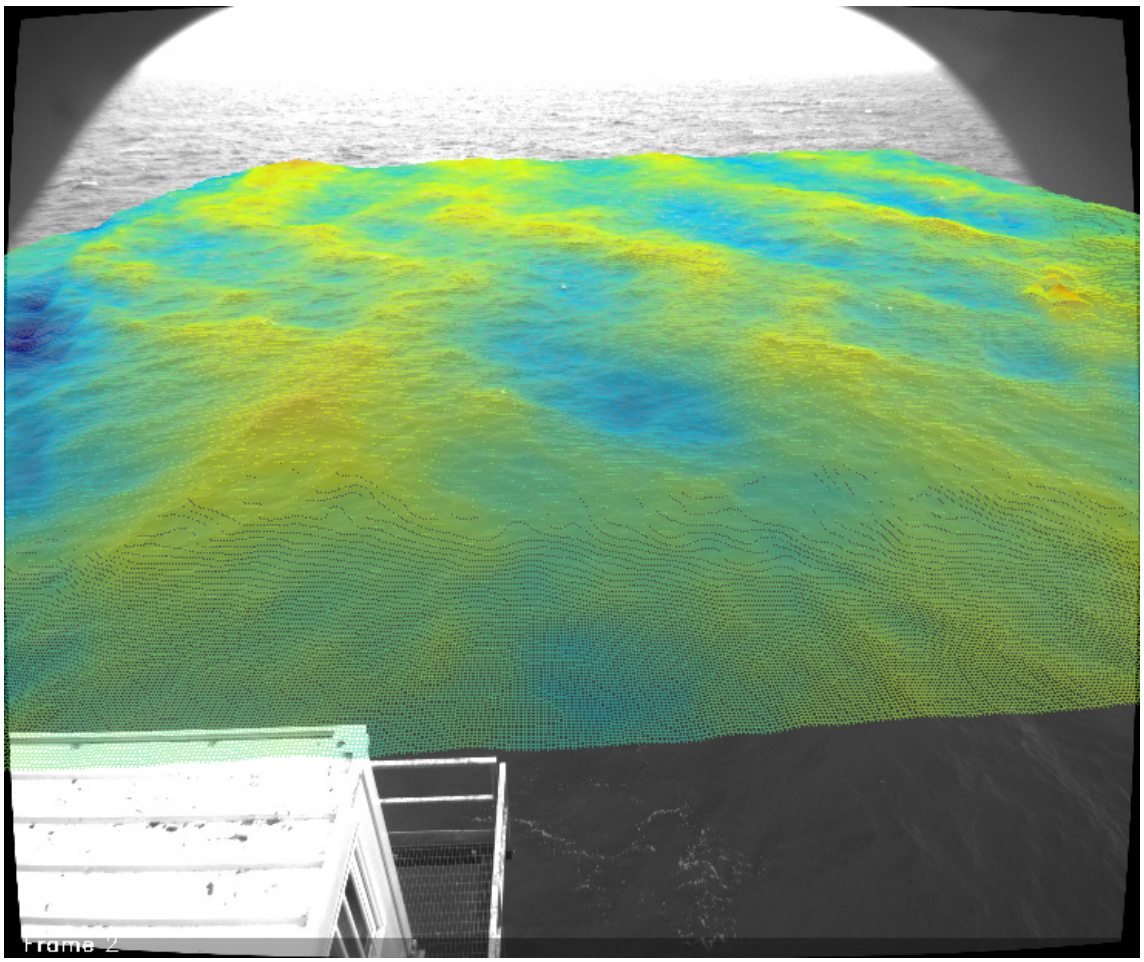


Figure 5: Output produced by wassncplot

5 Rendering the result

By using `wassncplot` you can plot the 3D surfaces contained in the NetCDF on top of the original images. This is handy to check if the reconstruction looks accurate before proceeding with further analysis.

Create a `frames/` directory to store the images and run the tool:

```
mkdir frames
wassncplot gridding/gridded.nc frames/
```

when the process is finished, the `frames/` directory contains the produced images (Fig. 5). Note how the grid also extrapolates areas where no 3D point samples are present. It is recommended to use the central area of the grid when doing measurements.

6 Final remarks

I hope you find this how-to guide helpful in getting started with WASS and WASSfast. For suggestions on how to improve this guide feel free to write me an email at `filippo.bergamasco@unive.it`.

6.1 Citing

If you use WASS to reconstruct point clouds from stereo images, please consider citing the following article:

```
@article{bergamasco2017wass,
  title={WASS: An open-source pipeline for 3D stereo reconstruction of ocean waves},
  author={Bergamasco, Filippo and Torsello, Andrea and Sclavo, Mauro and
```

```
  Barbariol, Francesco and Benetazzo, Alvise},
  journal={Computers \& Geosciences},
  volume={107},
  pages={28--36},
  year={2017},
  publisher={Elsevier}
}
```

If you are using WASSfast, please cite both:

```
@article{pistellato2021physics,
  title={A physics-driven CNN model for real-time sea waves 3D reconstruction},
  author={Pistellato, Mara and Bergamasco, Filippo and Torsello, Andrea and
  Barbariol, Francesco and Yoo, Jeseon and Jeong, Jin-Yong and Benetazzo, Alvise},
  journal={Remote Sensing},
  volume={13},
  number={18},
  pages={3780},
  year={2021},
  publisher={MDPI}
}

@article{bergamasco2021toward,
  title={Toward real-time optical estimation of ocean waves' space-time fields},
  author={Bergamasco, Filippo and Benetazzo, Alvise and Yoo, Jeseon
  and Torsello, Andrea and Barbariol, Francesco
  and Jeong, Jin-Yong and Shim, Jae-Seol
  and Cavaleri, Luigi},
  journal={Computers \& Geosciences},
  volume={147},
  pages={104666},
  year={2021},
  publisher={Elsevier}
}
```