Debris Detection Python

Quick User Guide

Note:

The current version of the code presented is a beta\research version, which requires improvement and further development. It should only be used for research purpose and to provide rough estimates of debris in water flows. It should be noted that the current setup is based on a standard camera, however a fisheye camera was used in the data gathering. The Debris Detection work can be found in the sub-folder 'DebrisDetection'

Installation

The automated python script requires the installation of a python 3 environment on the user's machine. (It was tested using: conda create -n debrisdetection python==3.10 -c conda-forge, conda activate debrisdetection)

Once installed run:

```
pip install -r requirements.txt
```

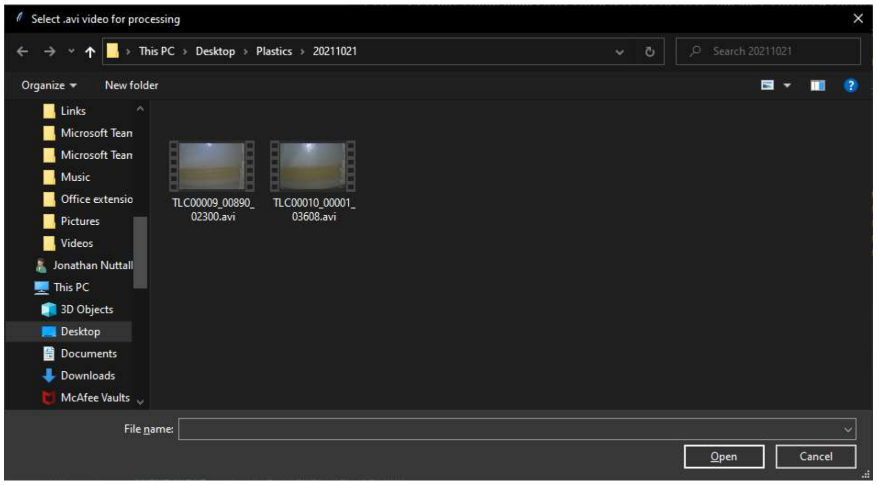
from the files provided, within the command line for the python environment you are using.

The provide calibration folder should also be place alongside the DebrisDetection folder, as the calibration routines are utilized in the debris analysis.

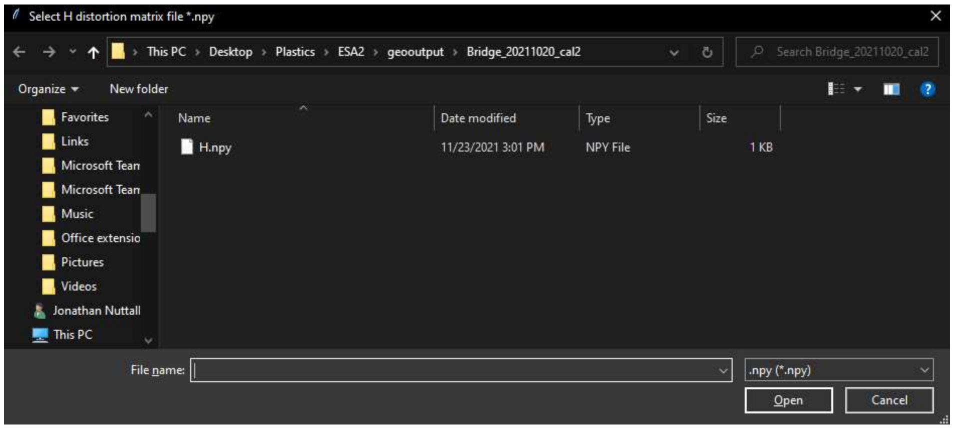
Running the code

Within your python environment type python debrisDetection.py

1. You will be asked to select an avi video file. Please select a video file and click ok.
   Note: Currently a frame rate = 1 frame per second. Although this can be changed in future.



2. You will then be asked to open the H.npy matrix that was saved during the camera / gps calibration, of the type and position of the camera from which the video opened in step 1 was captured.



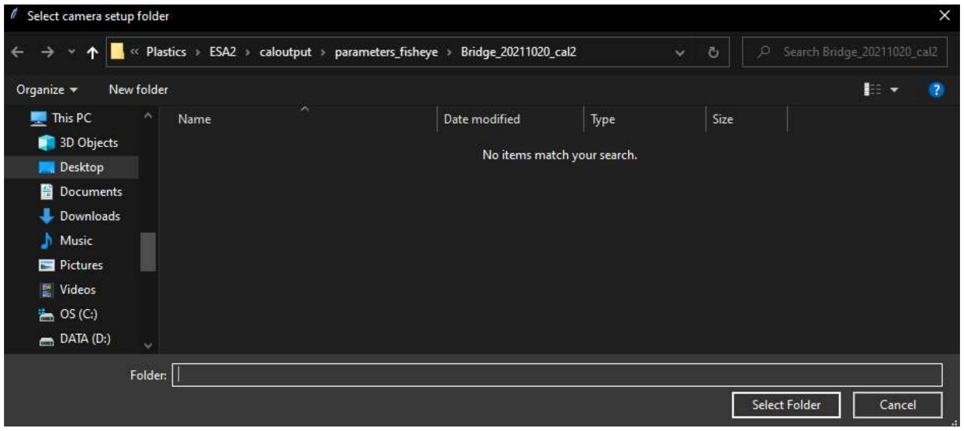
Select H distortion matrix file *.npy

This PC › Desktop › Plastics › ESA2 › geooutput › Bridge_20211020_cal2

Name	Date modified	Type	Size
H.npy	11/23/2021 3:01 PM	NPY File	1 KB

File name: .npy (*.npy)

3. You will then be asked to select the folder containing the calibrated parameters K and Distortion (dist.npy and matrix.npy) for the camera (i.e. fisheye lens) that was generated in the calibration prior to using this tool.



Select camera setup folder

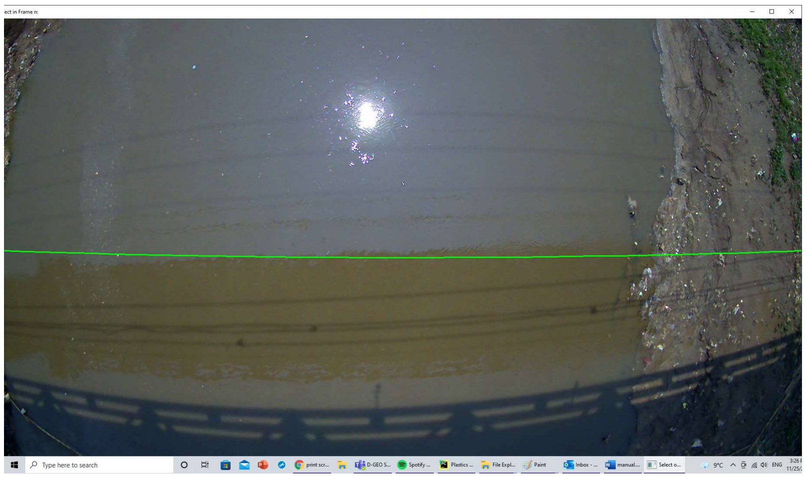
« Plastics › ESA2 › caloutput › parameters_fisheye › Bridge_20211020_cal2

Name	Date modified	Type	Size

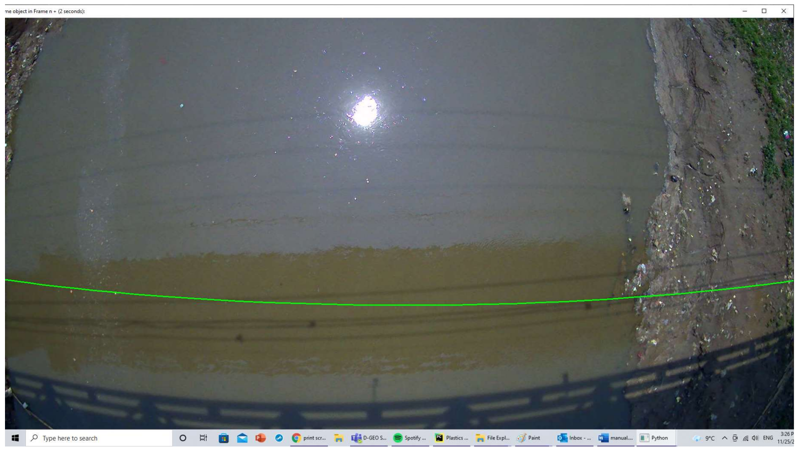
No items match your search.

Folder:

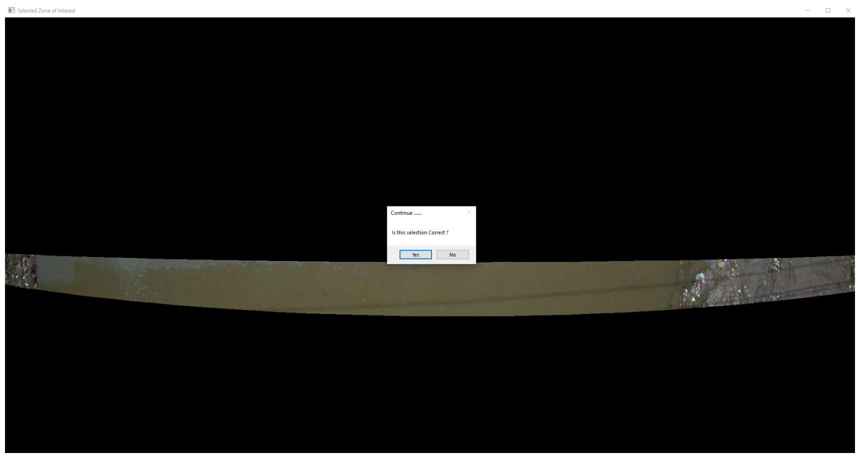
4. You will then presented with the a random frame from the chosen video with a green line. Using the left mouse button select an object in the water. Note the curvature of the line based on lens distortion (e.g. fisheye lenses) **Hint:** try to place this in the upper portion of a shaded area as this will improve the debris detection and reduce sun glare. It is also assumed that the video is stable and has no obstructions from start to finish.



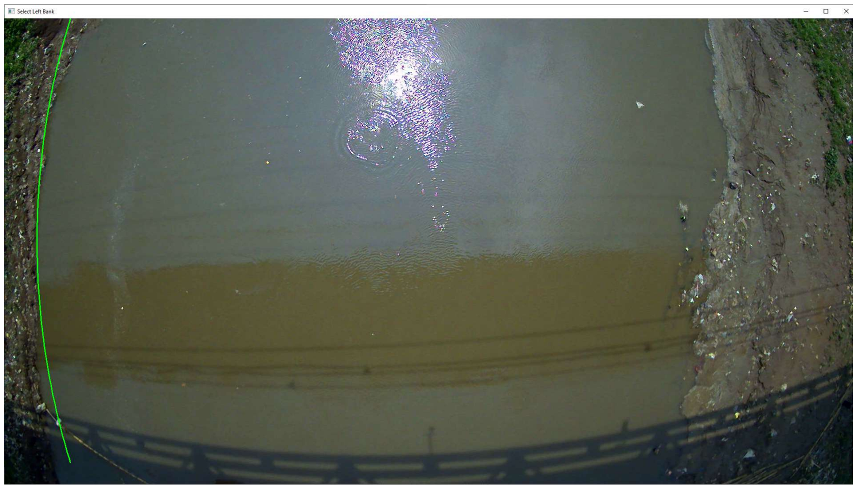
5. You will be asked to repeat this for the 2 frame afterwards (i.e. 2 seconds) Select the same object from the image. This will give us a rough estimate of the flow rate in the water and provide a 2 second zone across the river on which to base our statistics.



6. You will be asked whether the lines presented are correct, (could you not find the object in the second frame, or is the lighting area poor) Click yes (to continue) or no (to repeat). The image in the background indicates the zone to be analyzed to capture debris.

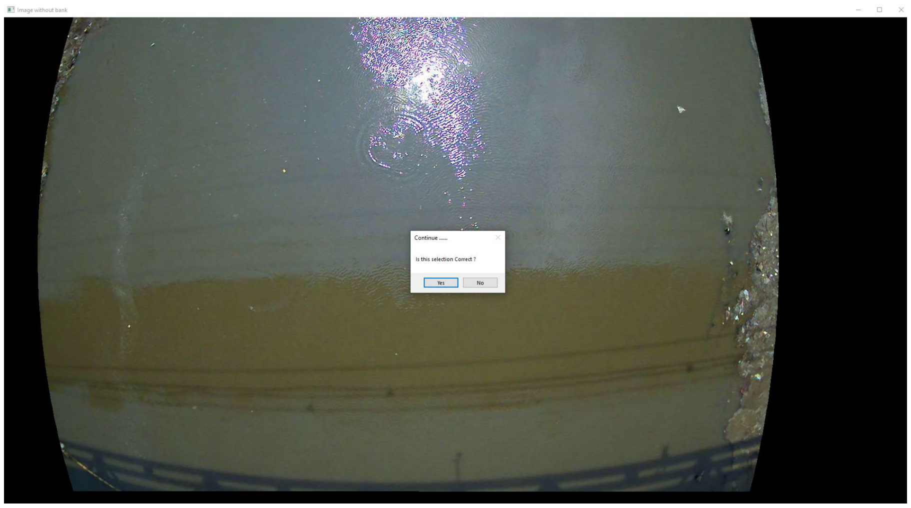


7. You will now be presented with another random image of the river and asked to draw a line for the left bank of the river again with curved lines due to lens distortion:



8. Then asked to repeat the exercise for the right bank (left click top right and bottom right):

9. You will be asked to confirm this is as you wish. Click yes (to continue) or no (to repeat).

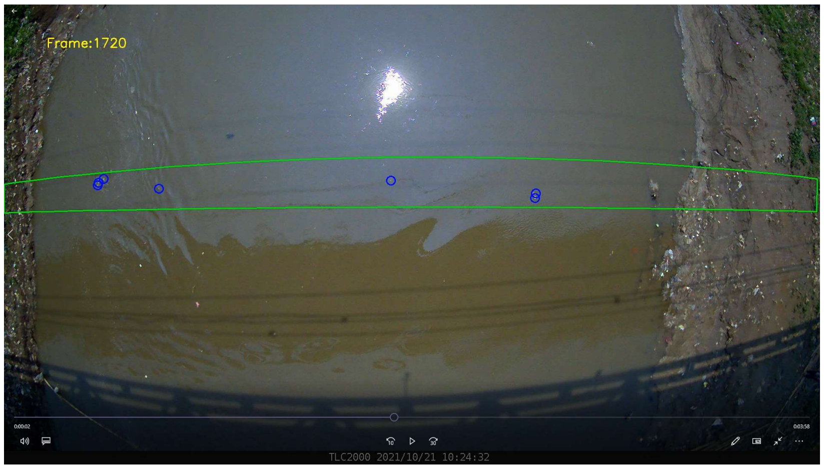


Is this selection Correct ?

10. The process is now automatic and will process your video for debris. You will get graphs which you can save and close, and in the command line (or python ide terminal\console) you will receive printed text feedback and the resulting statistics.

    Output:

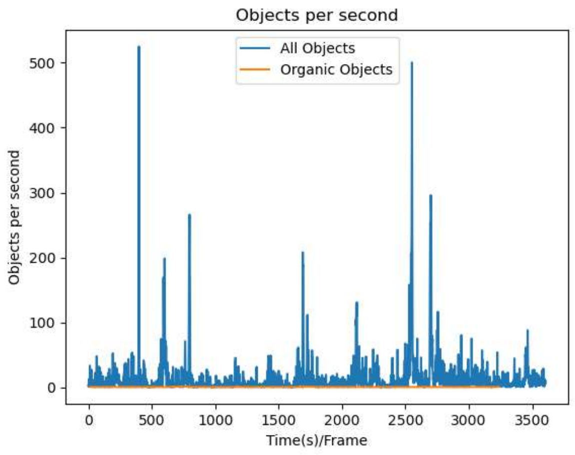
    - A video with highlighted debris passing through the chosen debris zone, blue circles indicating non-organic detection and green circles organic objects.

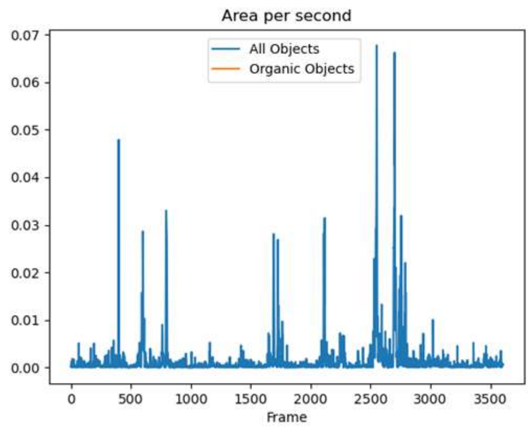


- Total Statistics for the debris (pixel based at the moment)

```
Statistics
===========================
Total Number of Frames: 3604
Frame rate used in calculations: 1

Total Number of Debris Items Detected: 171516.0
Total Number of Organic Debris Items Detected: 50.0

Total Area of Debris Detected: 18.663585756574115 m²
Total Area of Organic Debris Detected: 0.001474047784256527

Max Number of Debris per second: 1074.0
Max Number of Organic Debris per second: 1.0

Max Area of Organic Debris per second: 6.23110895503487e-05 m²
Max Area of Debris per second: 0.11576094574100182 m²
```

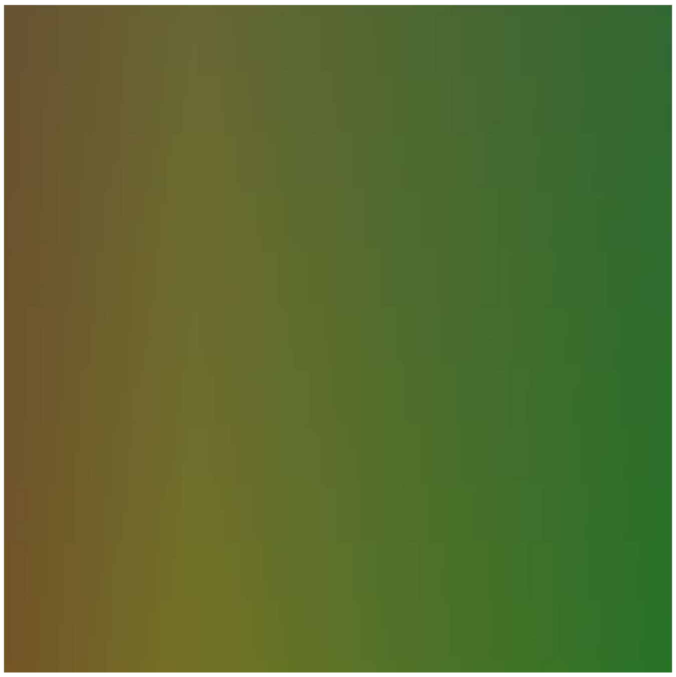
- Two time series graphs (Can be saved in the graph GUI. Closing one graphs opens the next):


Objects per second
All Objects
Organic Objects
Objects per second
Time(s)/Frame
0
100
200
300
400
500
0
500
1000
1500
2000
2500
3000
3500


Area per second
All Objects
Organic Objects
Frame
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0
500
1000
1500
2000
2500
3000
3500

APPENDIX

Organics Detection
As a note the organics detection is based upon colour. Having changed the RGB colour to HSL colour representation it is possible that organic colours fit into a single band of the spectrum. As shown in the following diagram (it should be noted that lighter\brighter colours are also included in a 3-dimensional representation of the spectrum).



As this is a very primitive inspection the measure of the value obtained should only be used as indicative measurements. Colour calibration in varying light conditions is difficult, especially when the images provided do not show the objects detected close or clear enough to identify.

Calibrations

Note:
The calibration step is currently disconnected with the Debris Detection step. (You work with the provided calibration files). As a follow-up the intention is to harmonise these post-processing activities. The calibrations work can be found in the sub-folder 'calibrations'.

Image preparation
Before the application of image vision on frames and estimation of floating object properties, one needs to cater for camera lens distortion and image geo-referencing. The former is achieved through lens calibration, the latter via image rectification.
Lens calibration should be performed per camera and per camera settings, and it makes sure that original (distorted) images can be undistorted from lens distortion. Image rectification should be carried out every time the field of view changes [i.e. if camera changes position and/or pointing direction].
The final result of lens calibration is the so-called K matrix and the distortion vector, whereas the main result of image rectification is the estimation of a homographic matrix H, which is used to transform from undistorted UV image coordinates to XYZ real-world coordinates.

Lens calibration
Lens distortion from manufacturer is taken into account first, so that objects appearing in the field of view would appear as though no distortion existed. The package provides a script that makes a lens calibration model, based on checkerboard camera snapshots.
- *lenses.py*: this reads checkerboard images and estimates so-called camera intrinsic parameters, namely a matrix K and a distortion array. Given a path of checkerboard images, this file can return undistorted images after the application of calibration model. Output parameters, stored as numpy files, can be selected.

Image rectification
This step requires the use of in-field GCPs [Ground Control Points] and camera snapshots related to those GCP measurements, to create a model that would relate UV to XYZ coordinates.
- *makegcp.py*: This script makes use of a csv of GCP XYZ coordinates, it helps to identify GCP UV coordinates, and returns a list of those as csv in geooutput folder.
- *georef.py*: This set of functions make use of lens calibration parameters and GCP UV and XYZ coordinates, returning the H matrix, ensuring image coordinates to real-world coordinates transformation.