IVER3 AUV IMAGE PROCESSING SOFTWARE

Mark Lundine

**Problem:**

The Iver3 autonomous underwater vehicle (AUV) can collect red, green, blue (RGB) imagery at set time-intervals during its underwater mapping missions. This imagery is highly useful for ground-truthing sidescan sonar (*e.g.* seabed sediment texture), for identifying various types of aquatic life, and for identifying shipwrecks. It would be desirable to have a method for quickly constructing coarse and georeferenced mosaics of all of the underwater imagery collected during an Iver3 mission. This would allow scientists using an Iver3 to look at all of the images at once as a complete picture, as well as see how that mosaic correlates with the other datasets typically collected by this AUV (*e.g.,* sidescan sonar, bathymetry, magnetometer data).

**Methods:**

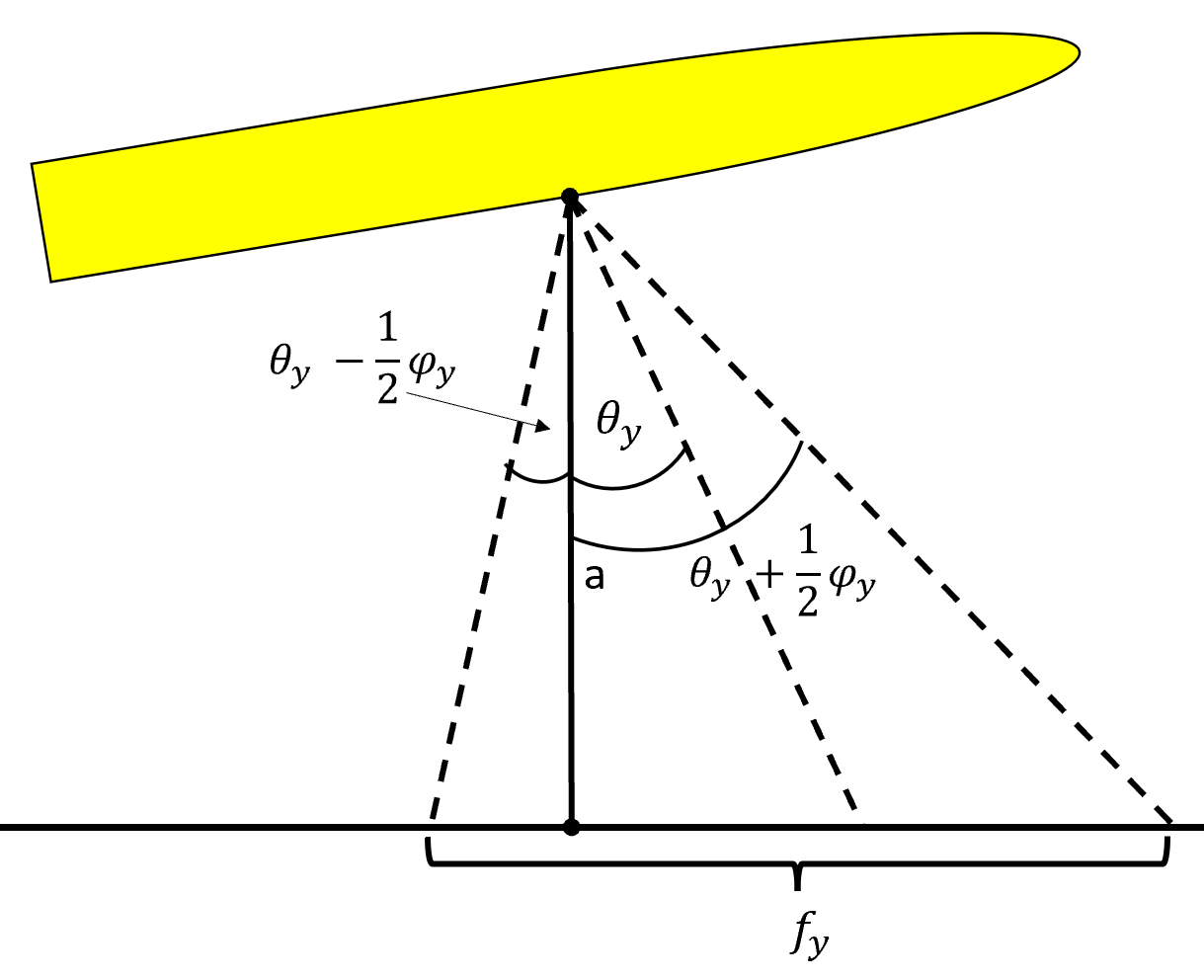
If the camera is enabled during an Iver3 mission, all images get saved to the mission’s ‘Camera’ directory. Inside the ‘Camera’ directory, images are found in subdirectories, with each subdirectory named as the starting waypoint for each line in the mission. Inside each waypoint subdirectory, there is an additional subdirectory (‘VC1’). Inside the ‘VC1’ subdirectory are the images for that line. The folder structure for a mission with one line would look like this:

Camera

* WP1
  + VC1
    - Images

For each image, important pieces of metadata are found as EXIF tags. This would include the vehicle’s latitude, longitude, depth, altitude, heading, pitch, roll, and speed. The first set of tools allow users to parse the ‘Camera’ folder for all of the image paths collected during a specific mission, and then output a csv file containing entries with image paths and various metadata properties.

The mosaic we aim to construct is one that places each image in the correct location and at the correct altitude of the seabed, rotating it to the correct perspective with respect to the seabed that it was captured at. We also want the size of the image (in meters) to match its actual footprint on the seabed. Using basic trigonometry, we can construct this coarse mosaic (see Figure D.1).



**Figure D.1:** Geometric schematic of computing the image footprint from the Iver3 AUV.

From the image metadata, we have our altitude (*a)*, the pitch (), the roll (), the heading (*h),* the latitude (*y),* the longitude (*x*), and the depth (*d*). We are assuming the latitude and longitude are taken from the camera sensor location. Additional variables gathered from Iver3 product information include the focal length of the camera lens (*l,* 12 mm), the width of the camera sensor (, 10.67 mm), and the height of the camera lens (, 8.00 mm). We aim to compute the horizontal and vertical field of view of the image ( and ), the horizontal and vertical footprint ( and ) of the image in meters, as well as the image’s corner coordinates (in latitude and longitude).

First, we need to compute the horizontal and vertical field of view of the image, using the sensor width, the sensor height, and the focal length:

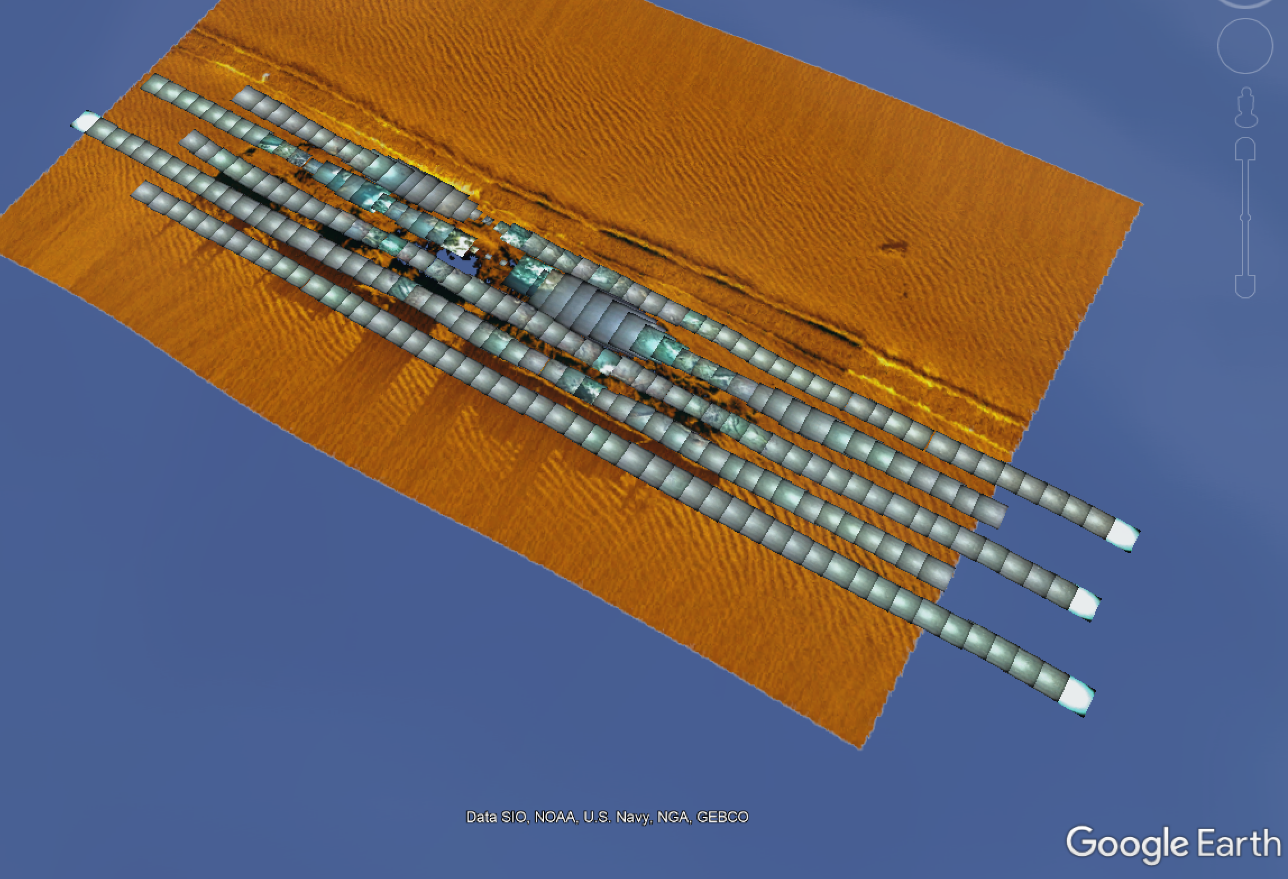
Next, we compute the horizontal and vertical footprint:

With the horizontal and vertical footprints, we can compute the footprint of the diagonal of the image ():

We then divide this value by two to get the distance from the center of the image to the corner. Next, we use our heading to compute the compass bearing to each corner (, , , ). The ‘ul’ subscript corresponds to the upper left corner, ‘ur’ to the upper right corner, ‘lr’ to the lower right corner, and ‘ll’ to the lower left corner.

Next, we compute the corner coordinates () using our image’s center coordinates (), the bearing from the center to each corner (), and our diagonal length divided by two (), with *GeoPy*. See the source code for more details on how this is implemented (GitHub link specified later in this Appendix).

The last step is to take all of the computed corner coordinates, the image footprints, and the images themselves and put them into a KMZ for visualization in Google Earth. This is accomplished using the package *simplekml*, where we take all of our computed information as well as the photos themselves to make a type of KMZ class called ground overlays. See Figure D.2 for an example of several filmstrips overlain on a sidescan sonar image. At the center of this figure, the AUV came quite close to the shipwreck it was mapping. Consequently, the footprint of the image became quite small. It then sensed it was too close to the wreck and increased its altitude, leading to a drastic increase in the image footprint.



**Figure D.2:** Example of KMZ strips from Iver3 camera data overlain on sidescan sonar.

**Software Solution:**

A set of Python-based tools were developed to allow others to construct KMZ strips of Iver3 camera data.

**Software Location:**

This project is housed at <https://github.com/mlundine/IverCameraTools>, where it can be downloaded, cloned, or forked. The README documents how to set up and use this set of software.

**Software Use:**

The required libraries include Python 3.7, simplekml, pandas, numpy, geopy, opencv, PIL, glob, and os. These should be downloaded into an Anaconda environment where the Iver Camera Tools are going to be utilized from. The file Iver\_camera\_tools.py is the only Python file needed to run these tools. The only function the user needs to run is ‘main’. This function takes three parameters:

mission\_id (str): a name for the mission

camera\_folder (str): the path to the camera folder for the Iver mission

photo\_or\_ground (str): ‘ground’ for ground overlays, or ‘photo’ for photo overlays

An example of running this function would be:

main('uss\_nina\_footprint\_seafloor\_enhance',

r'C:\MarkLundineSurface\iver\Camera',

'ground')

Running this would make all of the KMZ strips for that mission as well as a csv containing all of the image metadata and computed footprints and corner coordinates. This would all be saved in the Camera folder given as a parameter for this function.