Alaska
Fisheries Science
Center

National Marine
Fisheries Service

U.S DEPARTMENT OF COMMERCE

AFSC PROCESSED REPORT 2016-03

doi:10.7289/V5/AFSC-PR-2016-03

Sebastes Stereo Image Analysis Software

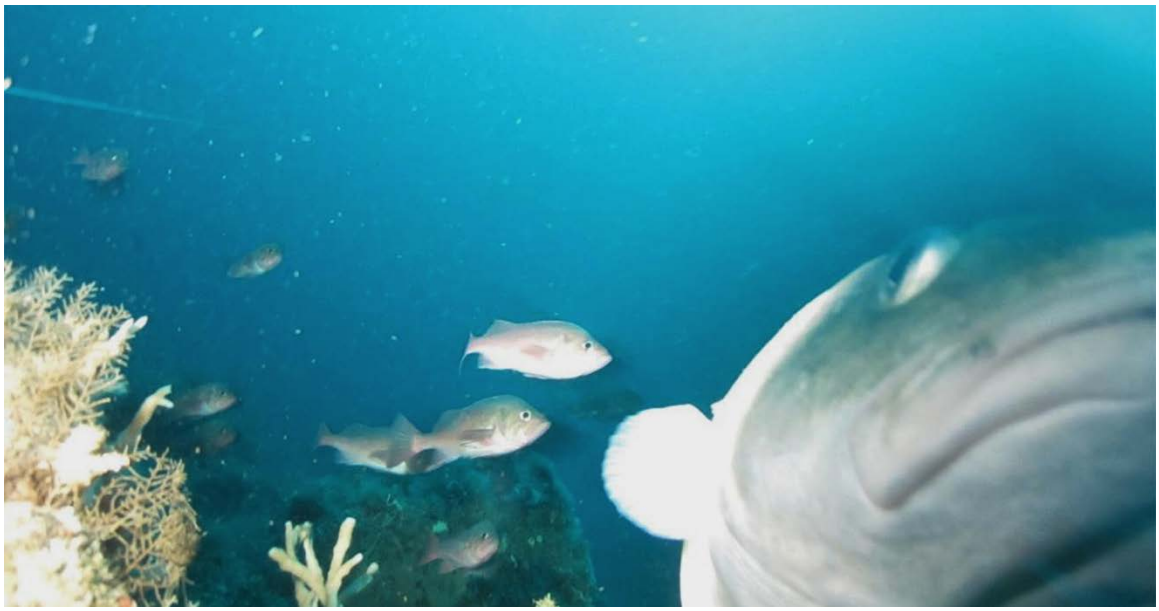
June 2016

This document should be cited as follows:

Williams, K., R. Towler, P. Goddard, R. Wilborn, and C. Rooper. 2016. Sebastes stereo image analysis software. AFSC Processed Rep. 2016-03, 42 p. Alaska Fish. Sci. Cent., NOAA, Natl. Mar. Fish. Serv., 7600 Sand Point Way NE, Seattle WA 98115. doi:10.7289/V5/AFSC-PR-2016-03.

Available at http://www.afsc.noaa.gov/Publications/ProcRpt/PR2016-03.pdf

Sebastes Stereo Image Analysis Software



K.Williams, R. Towler, P. Goddard, R. Wilborn, and C. Rooper

Alaska Fisheries Science Center
NOAA, National Marine Fisheries Service
7600 Sand Point Way NE
Seattle WA 98115

June 2016

Abstract

This report describes a set of software programs that were developed by the Alaska Fisheries Science Center for analyzing stereo images. The main application is called SEBASTES, and is used to count, range, and measure fish using stereo-image algorithms. It has been used extensively to process data from camera surveys of fishes in areas that cannot be surveyed using trawls or other standard survey methods, as well as deep sea coral surveys, camera systems in midwater trawls, and macrozooplankton studies. Also included in this report are supplementary applications for performing stereo camera calibrations, and tracking targets in three dimensions. The report is meant as a resource for those using the software, as well as a technical reference for the methods used to extract quantitative data from stereo images.

Contents

Introduction **1**
SEBASTES Software 1
Stereo Image-based Ranging and Measurement 1
Stereo Camera Calibration 2
Stereo Triangulation 2
SEBASTES Application **4**
Image Data Structure 4
Analysis Setup 5
Loading a Deployment 8
Browsing 9
Performing an Analysis 10
Habitat Classification 12
Fish Counting and Identification 13
Fish Measurement 16
Error Estimation 17
Scene Ranging and Measurements 20
Target Association 20
Data Viewing 21
Deleting Targets 21
Target Classification Review 21
Data Outputs 23
Camera Calibration Application **24**
General Notes on Performing an Underwater Camera Calibration 25
Conducting a Calibration Using StereoCalibrate 26
Stereo Tracker Application **30**
Citations **33**
Appendix **35**
Software Code Overview 35
GUI Design Tools 37
Pystereocomp Functions 73
Dialogs 41
Qimageviewer 42

Introduction

SEBASTES Software

SEBASTES software is a free, open source software package developed for the specific purpose of underwater fish length measurement. It can also be used to determine the range of objects to the camera in images, such as the sea floor, and to determine the range, position, and orientation of fish and other targets. SEBASTES is based on several open source projects, including the camera calibration toolbox (Bouguet 2008) and OpenCV (Bradski 2000). It is written in Python (Python Software Foundation, V 2.7, https://www.python.org/) language using the PyQt (V. 4.4, Qt GUI development environment for python, Riverbank, riverbankcomputing.com). Two additional software tools are included with the SEBASTES distribution: StereoCalibrate for performing a stereo calibration using the OpenCV package, and Stereo Tracker for manually tracking targets in three-dimensional (3D) space.

Stereo Image-based Ranging and Measurement

Paired stereo images can be used to estimate the distances between objects by using the shift in perspective, much like how stereoscopic vision works in humans. To quantify the exact range of objects from a camera, several factors about the camera system need to be known which are referred to as intrinsic and extrinsic factors. Intrinsic factors describe the properties of a camera's individual view field, which includes focal length, image sensor size, and the principal point. For an ideal "pinhole" camera, these three parameters describe the pyramidal field of view of the camera. Most camera lenses add some distortion to an image, making the simple "pinhole" model inaccurate. For this reason, distortion parameters are also included in intrinsic factors as they are specific to each camera and need to be known to adjust the image. Extrinsic factors

apply to a stereo- or other multi-camera system, and specify how the view-field of one camera is geometrically related to another. For the stereo camera case, this consists of a translation matrix (the offset in 3D of one camera relative to the other) and the rotation matrix, which relates one camera's heading, or pointing direction, to the other.

Luckily, well-established procedures exist to derive all intrinsic and extrinsic parameters required for ranging objects in the images. The process of deriving these parameters is called stereo camera calibration. Upon deriving these parameters, we can use an algorithm called stereo triangulation to determine the exact 3D positions of objects simultaneously seen in both cameras.

Stereo Camera Calibration

Stereo camera calibration is used to derive the parameters necessary for stereo triangulation. The SEBASTES system uses the computer vision standard checkerboard method for calibration, as described in the camera calibration toolbox webpage (http://www.vision.caltech.edu/bouguetj/calib_doc/; Bouguet 2008). SEBASTES requires calibration parameters derived using this process as implemented in the Matlab programming environment, or an alternative method based on the OpenCV computer vision software library (http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html). The latter is implemented in the Stereo Camera Calibration GUI described in detail later in this document.

Stereo Triangulation

Stereo triangulation refers to the process of estimating, or reconstructing, the 3D position of an object in real world coordinates based on locations of the object in a pair of images and the

known properties of the stereo camera system. For example, we would like to know the length of a fish seen in both left and right images from an underwater stereo camera as shown in Figure 1.

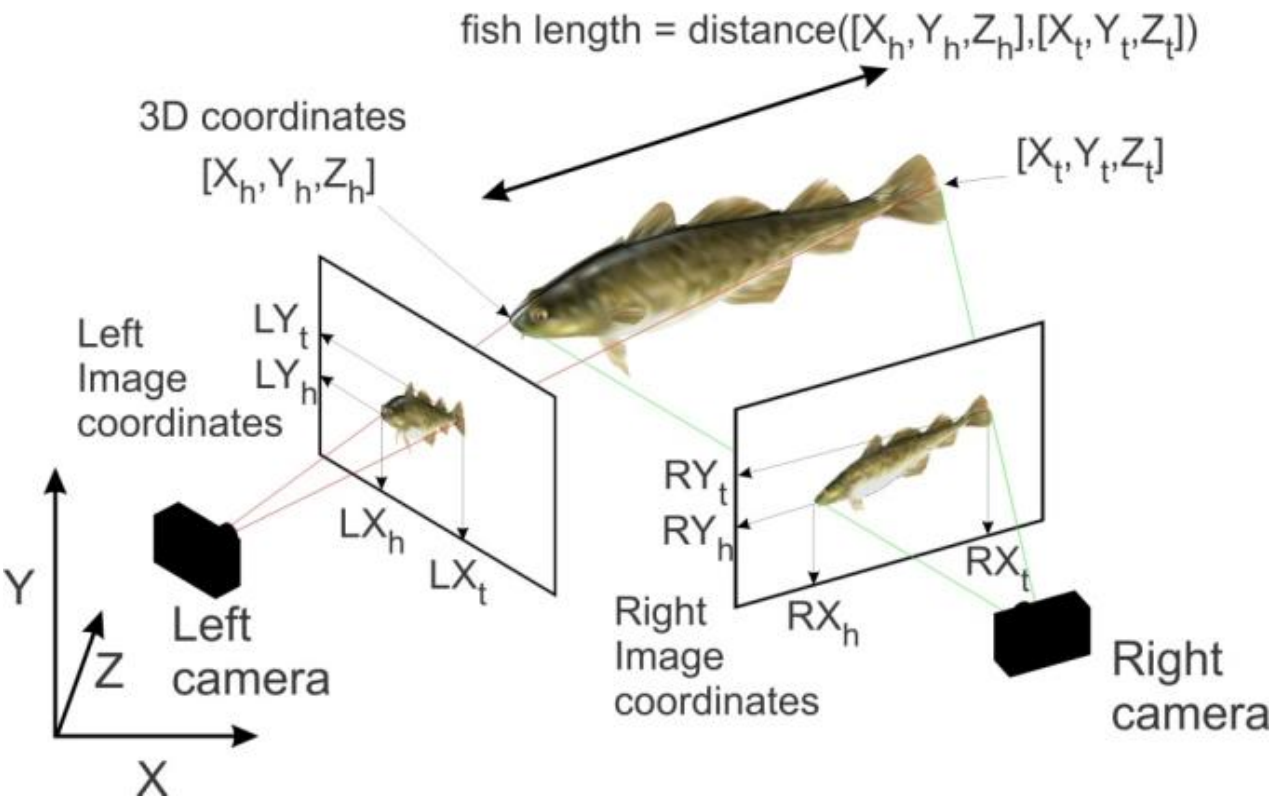


Figure 1. -- Length estimation using stereo triangulation methods.

By establishing the position of the snout in both images (LX_h,LY_h,RX_h,RY_h), we can use triangulation to derive X_h, Y_h, Z_h, which is the location of the tip of the snout in 3D space. Once the process is done for the fish tail, the linear distance between the two points (fish length) in 3D space is equivalent to the straight line length. The accuracy of this process depends on the accuracy of the intrinsic and extrinsic parameter estimates, and the accuracy of the equivalent point location (e.g., tip of the snout, fork in the tail) based on human input (clicking on image) in both images. When images are derived from independent left and right video streams as opposed

to simultaneously triggered still cameras, additional error may be derived from asynchronous frame grabs from the video files.

SEBASTES Application

Image Data Structure

SEBASTES requires still images from separate folders for the left and right (or up and down) cameras. The image names should be formatted as TIF or JPG and contain a frame number which matches the image pairs. For example, in the file name IMG_0040.JPG the ”0040” identifies the numeric frame number. The cameras used do not have to be identical: For example, one can be monochrome while the other can be in color, or one can have a wider field of view, etc.; the camera calibration accounts for heterogeneous system setups. Paired images from each camera form a set, defined as a discrete sampling event or deployment, and both are treated as an analysis unit in SEBASTES. Each image set should be in a separate directory, and the name of the directory containing the left and right image folders is used as the deployment ID. SEBASTES writes analysis results in a folder called “data” that is created within the deployment folder. Figure 2 illustrates the folder structure. In this example, the “example_images” folder is the deployment folder and that name becomes the deployment identification (ID).

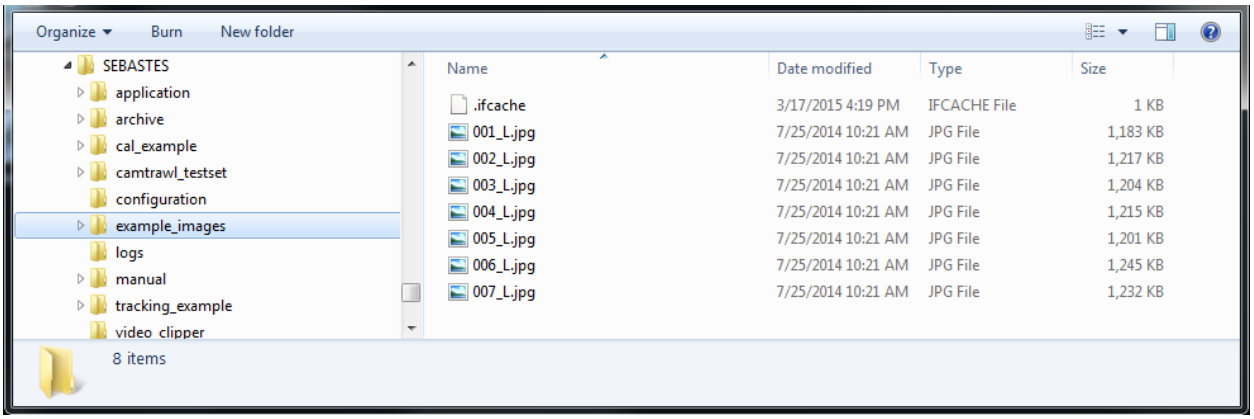
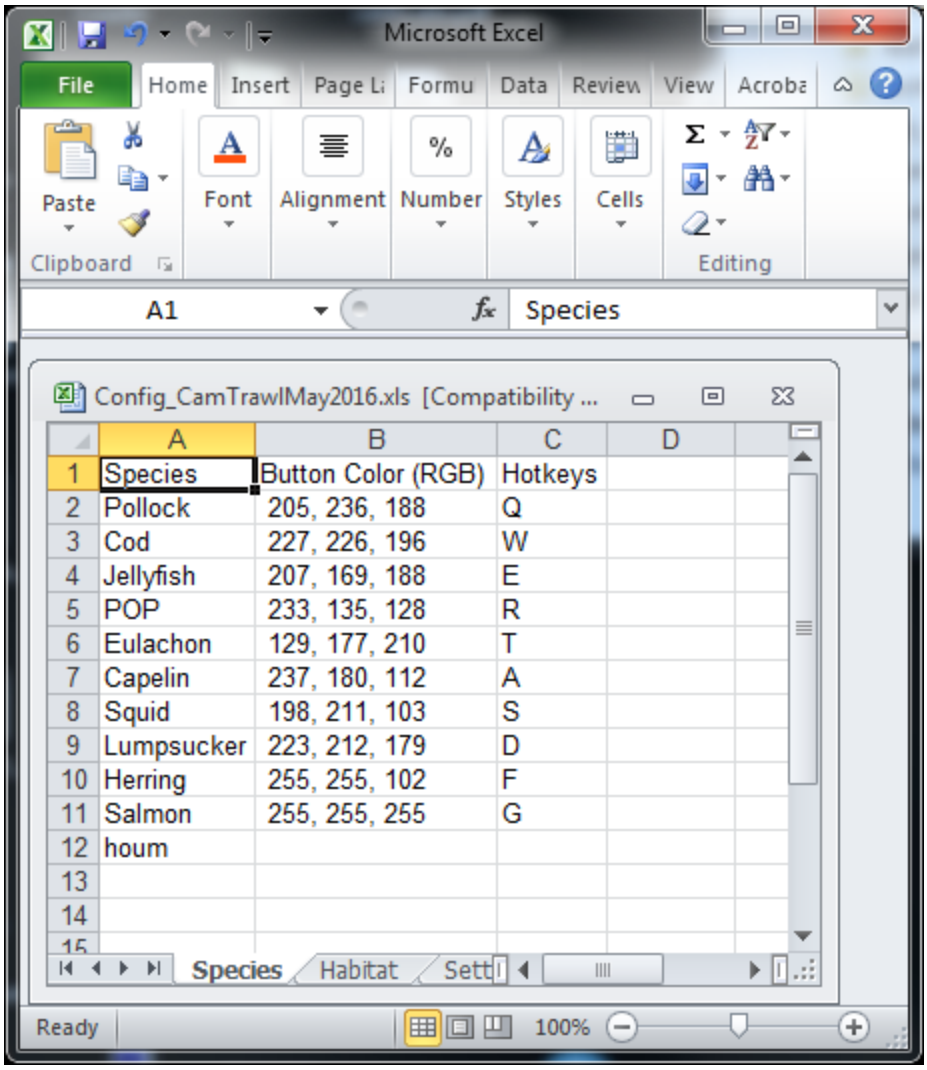


Figure 2. -- Example of image data structure used for still stereo images for SEBASTESanalysis.

Analysis Setup

SEBASTES uses a template file to assign settings to each deployment analysis and determines what options the user will see on screen when they open the SEBASTES application. The template file is a Microsoft Excel file (.XLS) with three tabs: “Species”, “Habitat”, and “Settings” (Fig. 3).



	A	B	C	D
1	Species	Button Color (RGB)	Hotkeys	
2	Pollock	205, 236, 188	Q	
3	Cod	227, 226, 196	W	
4	Jellyfish	207, 169, 188	E	
5	POP	233, 135, 128	R	
6	Eulachon	129, 177, 210	T	
7	Capelin	237, 180, 112	A	
8	Squid	198, 211, 103	S	
9	Lumpsucker	223, 212, 179	D	
10	Herring	255, 255, 102	F	
11	Salmon	255, 255, 255	G	
12	houm			
13				
14				

Figure 3. -- The Excel template file, showing the species tab.

The contents of each of tab are described below:

1. Species Tab

The first column holds a list of species that can be used in the analysis. There is no maximum number for the list, but only 20 species are visible on species “buttons” in the application at any given time (this is discussed later in the *Fish Counting and Identification* section).

The second column contains a red-green-blue (RGB) triplet to designate the color desired for that button slot. All of these settings are fully configurable and can be modified depending on preferences or the focus of the study.

2. Habitat Tab

This tab allows the user to specify what types of habitat (seafloor substrate) to select from during the analysis (e.g., Sand, Cobble, Rock, etc.). Up to 10 categories can be listed in the first column. As many as five benthic biota coverage categories (e.g., sponges, coral) can be specified in the second column, and percent coverages entered. Two additional categorical Boolean habitat descriptors can be designated in the third column. For example, if the checkboxes column lists a value such as “trawlable”, the application will allow the user to check a checkbox with the label “trawlable” for that image frame.

3. Settings Tab

This tab allows a number of analysis settings to be specified, primarily those that should remain constant for a group of deployments. The parameters are listed in Table 1 with their descriptions.

Table 1. -- SEBASTES template parameters and settings.

LeftCam	Relative folder path to the left images. Leading or terminal slashes are not required.
RightCam	Same as above for right camera images.
ImageNumInds	The c index into the image name string that contains the frame number designation. For example, for an image named L_0001.jpg, it would be 3-6. Number boundaries are inclusive.
ImageTimestampFormat	If image names contain the date/time the image is taken, the format here is used to extract the time from the image name. If the user desires the image time to be taken from the exif image metadata, type in “exif”. Else leave blank and no time data will be written in the data file
ImageTimestampStart	Index into name string for timestamp, same as for frame number above. Only used if timestamp is in the name.
Calibration	Calibration file name to be used. The file needs to be located in SEASTES/configuration/ folder. Either .mat (matlab calibration) or .npz (openCV calibration) files can be listed.
HabitatAction1	Specify this as “sticky” if you want the habitat specifications to remain checked as you move from one frame to the next, otherwise “non-sticky”.
DoHabitat	Specify “Yes” if you want to do habitat analysis, otherwise this panel will not appear.
MeasureLineWidth	How wide you want the measurement line to be
MeasureLineTailLength	How long you want the measurement “tails” to be
TargetLineColor	Measurement line color (RGB triplet)
SceneColor	Color of scene ranging (non-species objects)

Loading a Deployment

After starting SEBASTES, go to File->Load Deployment and browse to the deployment directory. Images should load up and display the starting frame (Fig. 4). The slider bar at the bottom of the application window shows the position of the frame displayed relative to the entire frame set. The deployment name is highlighted in bold near the top left of the window. The current frame number is shown in the box at the upper right of the window, with the total number of frames indicated to the right. Replacing the text in this box allows you to jump to a specific frame number if required.

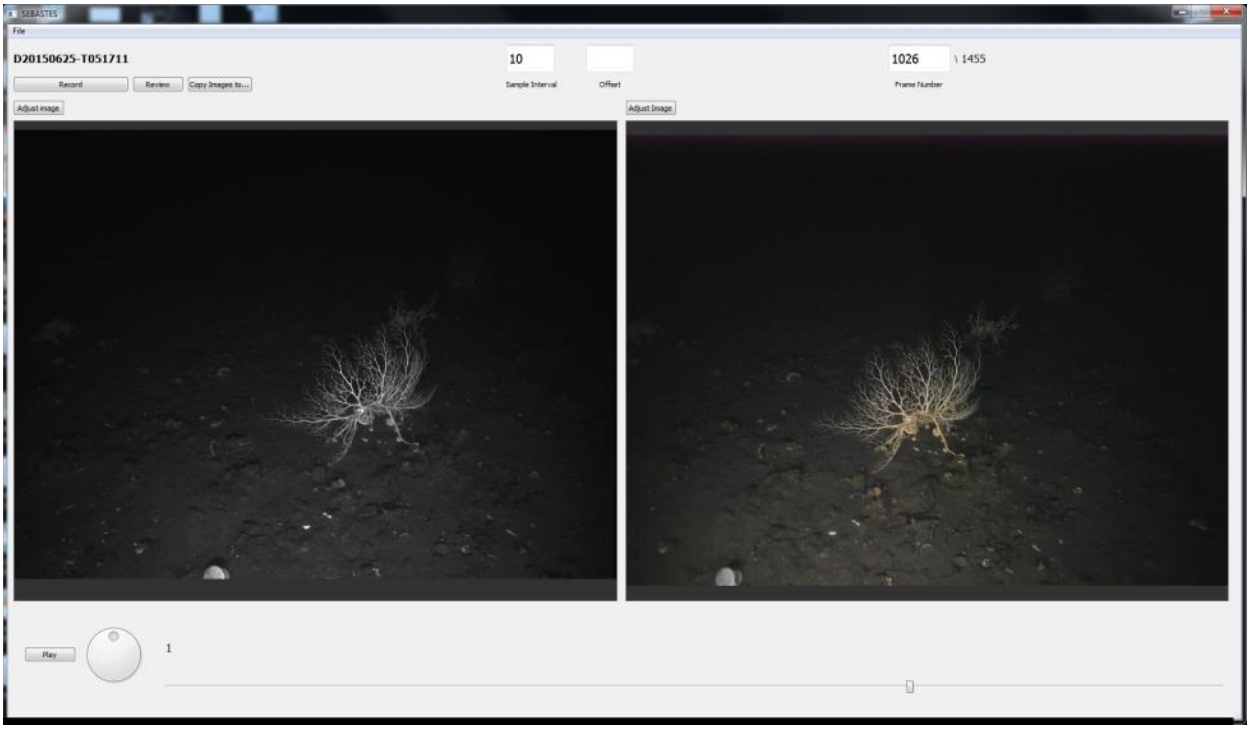


Figure 4. -- Initial view of SEBASTES application in browsing mode.

Browsing

The program defaults to “browse mode”, where the user can quickly look through the stereo data set of images by sliding the bar on the bottom of the screen or using the arrow keys on your keyboard. Data are not recorded during browsing and this mode is intended to give the user a general sense of the data and to see which parts of the deployment need to be more carefully analyzed. Pressing the *Play* button automatically increments the frames at a rate set by the dial to the right of the button.

In browse mode, image pairs can be copied to another directory using the “*Copy Images to*…” button. This enables storage of image pairs into a location where they can be used later. This is especially convenient for copying images for presentations or species identification.

Images can be independently adjusted for brightness, contrast, and color balance by pressing the *Adjust Image* button above each image. These settings persist as the user moves from image to image and are saved in the local data folder for the next time the deployment is opened.

Performing an Analysis

Data collection starts by pressing the record button. The species count panel is always displayed to the right of the image pair (Fig. 5). If habitat classification is set in the template file, a habitat panel will be shown on the left side. At this point, the slider is disabled and the only method for advancing to the next frame is using the left and right arrow keys or entering a frame number in the text box at the top of the right hand image. For arrow keys to advance a frame, the

cursor has to be inside the image window. Every time the arrow keys are used, any data that has been entered are recorded. Any data associated with the new frame are then displayed.

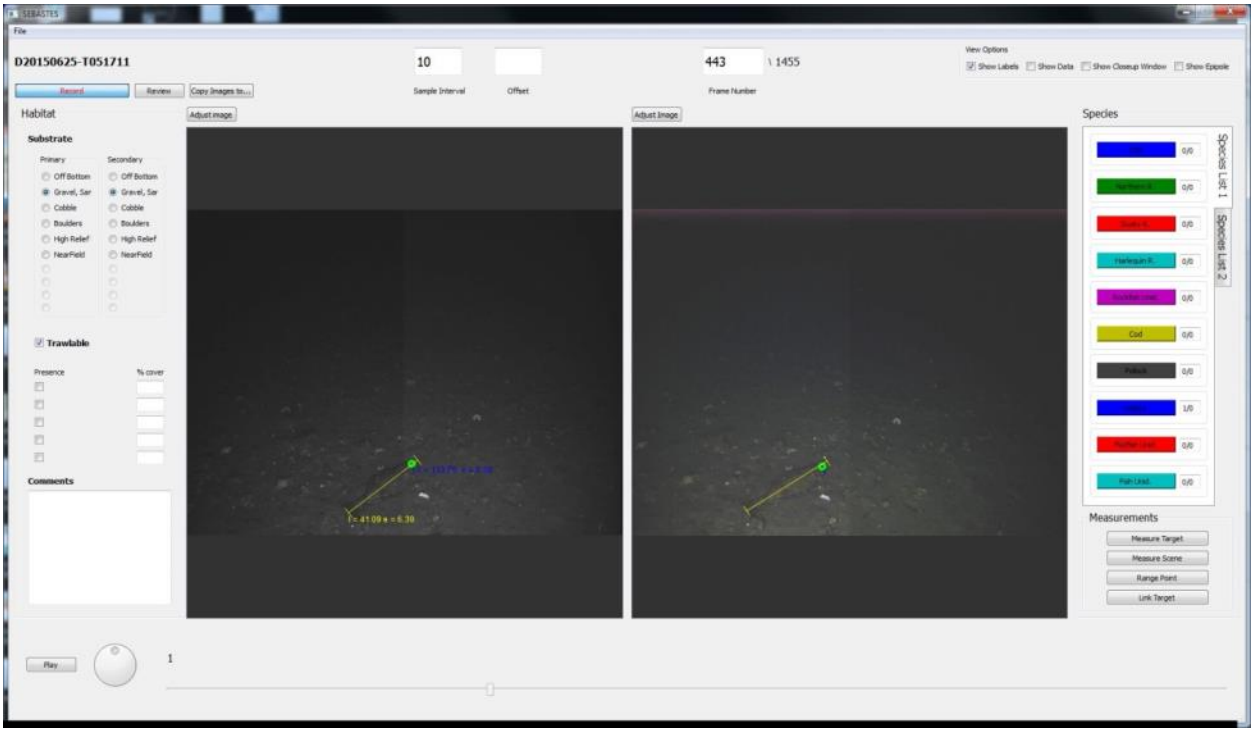


Figure 5. -- Record mode showing data collection panels on each side of the window.

The “sample interval” value determines how many frames the user wishes to advance with each arrow button press, and should be used if subsampling of the data is desired. The sample interval works on round numbers, so if the interval is set at 10, the frame numbers that are analyzed will all be multiples of 10. If the user wanted to change this, a value can be placed in the offset box. For example, a sample interval of 10 with an offset of 5 will display frames 15, 25, 35, etc. for analysis.

Habitat Classification

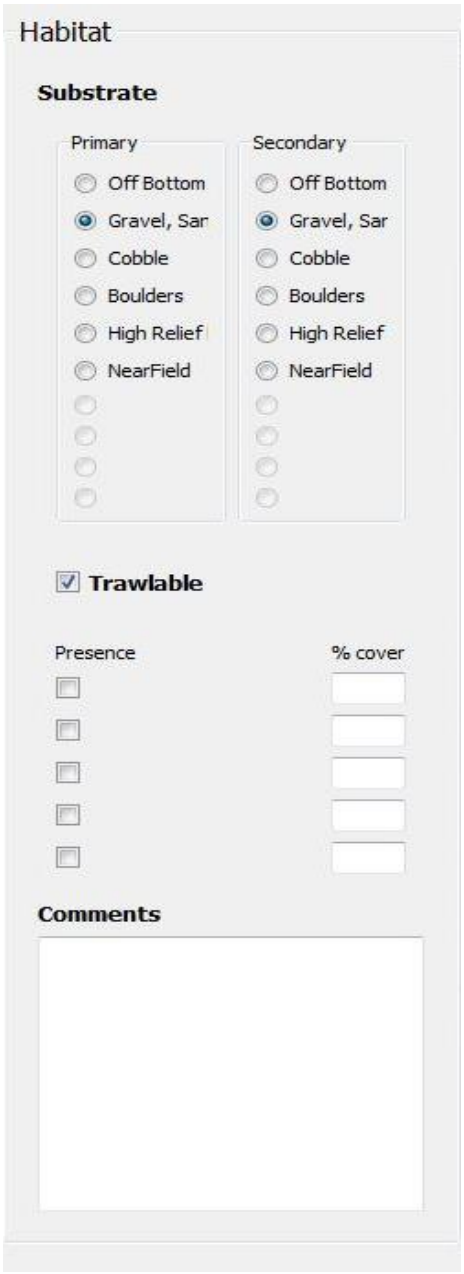


Figure 6. -- The SEBASTES habitat classification GUI.

SEBASTES contains features that allow the collection of habitat information for each frame. The Substrate buttons allow the user to indicate a primary and secondary substrate type.

The substrate classification system is based on Stein et al. (1992) and Yoklavich et al. (2000), where the primary substrate type is identified as having > 50% coverage on the image and a secondary substrate type can have 20-49% coverage of the seafloor. The categories are specified in the "Settings" template.xls file, up to a maximum of 10 categories. It is important to note that when HabitatAction1 is set to "sticky", the substrate type will be carried from frame to sequential frame until changed by the user.

In the example illustrated in Figure 6, a single Boolean habitat descriptor is set to allow the user to indicate that the habitat is "trawlable". The value of this button could also be changed to some other value of interest.

Finally, there are up to five spaces available for recording the percentage cover of items in each frame. Examples could include coral or algae cover, which can be used to indicate presence (check box) and percentage coverage. Text comments of up to 1,000 characters about each frame can be entered in the comment box.

Fish Counting and Identification

When in "*Record"* mode, buttons on the right side of the images are populated with up to 20 of the species provided in the template file, in the order they are listed. To keep the species buttons a manageable size, they were placed into two separate tabs (Species List 1 and Species List 2) with 10 species/classes each, and are easily accessed by toggling between the two tabs. Ideally, the 10 most common species would be on Species List 1, with more rarely encountered species on Species List 2. Toggling the buttons in the panel sets the "Active" species for the current counting operation (Fig. 7). Only one species can be active at a time. Button and mark colors can be set in the template file. To change the species order or to add a species that is not in

the 20 classes currently shown, press and hold a button down for 3 sec. This will bring up a dialog box allowing any species listed in the template file to be chosen to occupy that button (Fig. 8). If an additional species not on the existing list is needed, use the *Add New Class* button on the Select a Class dialog box and enter a species name and description. Upon exiting the program, the new class is written to the local configuration file and will be available for the next time that deployment is visited. For the new class to be available to other deployments, it should be added to the template file.

Once a class is selected and the user clicks on an image, a mark with the corresponding color is placed at the cursor location with the target number for that frame. As targets are often seen in both left and right images in a stereo-image pair, users can avoid double-counting by specifically indicating that a target is present in both frames. This is done by holding down the *Shift* key, then clicking on the target in both frames. If the *Shift* key is released before both images are

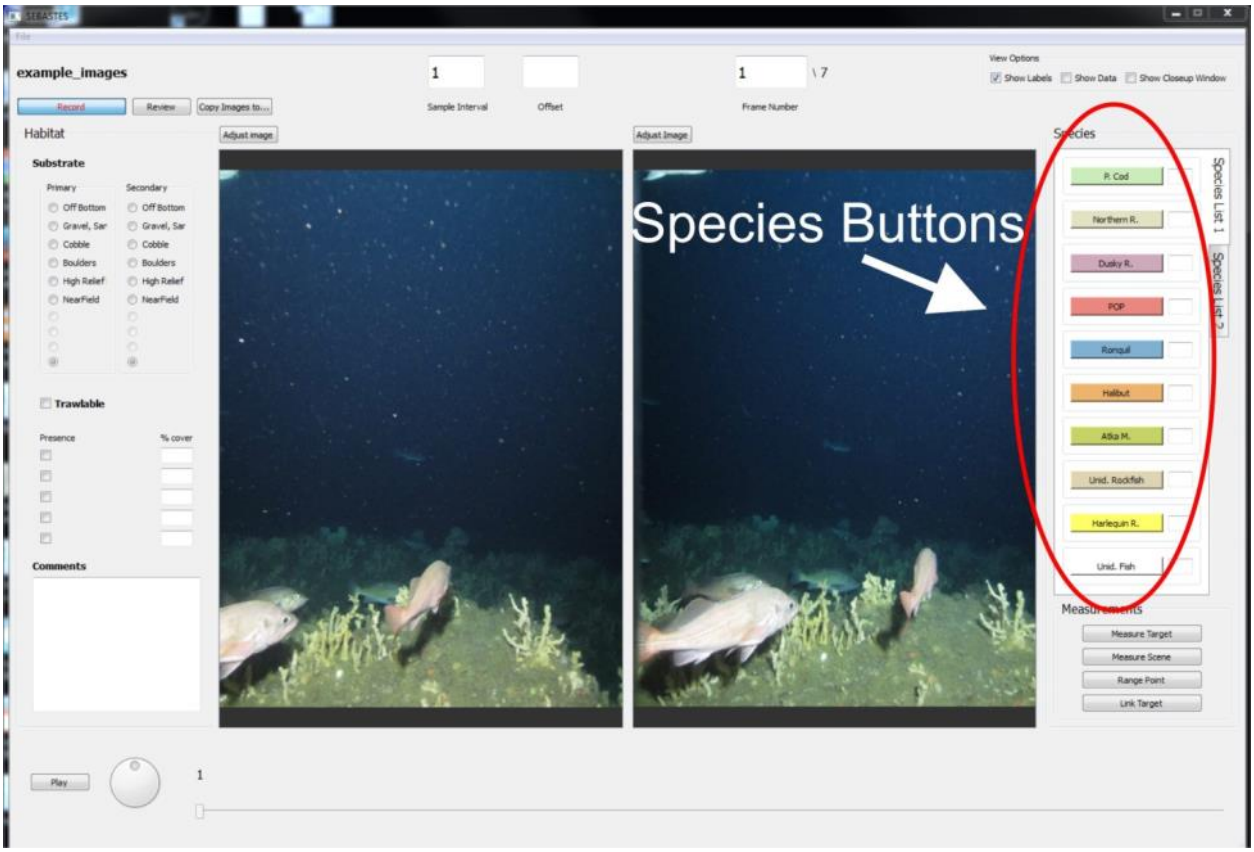


Figure 7. -- Species buttons in SEBASTES during record mode.

clicked on, no target is indicated. To use the stereo target click locations for range estimation, care should be taken to click on the same part of the target in each frame, for example, the fish eye or other conspicuous marking. A typical workflow would be to click on all the stereo-imaged targets, and then indicate ones seen only in the left frame, followed by the right frame (the last two steps without using the *Shift* key). The result would be a full visual count without double counting, as well as a set of stereo targets that include range and are ready to be measured if desired.

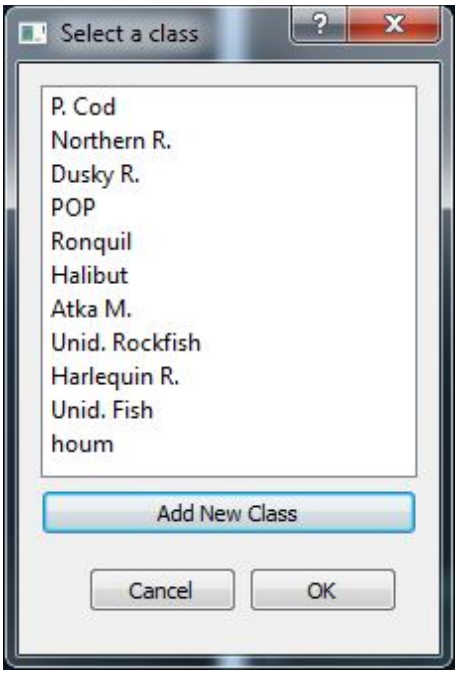


Figure 8. -- Dialog box to change values on species buttons.

In both browse and record modes, the user can zoom in on the image by pressing and holding the *Control* key while moving the mouse roller. To pan around the image, press and hold the *Alt* key while holding the left mouse button down (cursor will turn into a hand).

Fish Measurement

Length estimates can be made on stereo targets using the basic stereo triangulation concepts discussed earlier. The input points required for this operation are the pixel location of the snout and tail of the target in each frame. The entire measurement process is shown in detail in Figure 9. First, a stereo target must be set as active. If the user marks a stereo target using the *Shift* key, the most recent target will be the active one. If a previously marked stereo-target is to be measured, right click in the vicinity of the target to make it active. The active target will be indicated by a bright green outline. Once the active target is set, press the *Measure Target*

button. It will stay checked until enough data is collected to estimate the length. At this point, the user can click on the snout, and while holding the left mouse button down, move cursor to the tail and release (or other dimensional span of an organism) of the target in both images. After clicking on the snout, a yellow dimension line will extend from the initial click to the cursor location to help you visualize the linear distance you are measuring. Measurements can be initiated in either right or left image, but the same order of points on the target must be followed, e.g. snout-tail on the left image followed by snout-tail on the right image. It is convention to start with the snout, or head region, followed by the tail to indicate the correct body orientation (e.g., fish body tilt).

Once the user clicks on the snout and tail in both images a length measurement appears on the target and the measure button becomes unchecked. If another measurement is desired of the same animal, the measurement button can be pressed again, and the process repeated although this will replace the original measurement. Measurement accuracy is very sensitive to the placement of the end points on the image. To aid in this, the user can activate the close-up window by checking the close-up box in the upper right part of the main window. This will display a small magnified window of the image surrounding the current cursor location. It is especially useful with high-resolution images.

Error Estimation

Whenever a measurement or ranging action is made, SEBASTES computes an error value and places it on the screen next to the measurement/range value. This value is meant to provide relative guidance for how close the correspondence is on the two user-selected equivalent points in the left and right image. The error value is in pixel units that are

standardized by the image width to make them comparable across different image resolution sets. Usually, error values of < 20 are considered good. Consistently higher error values may indicate problems with the stereo calibration or frame synchronization.

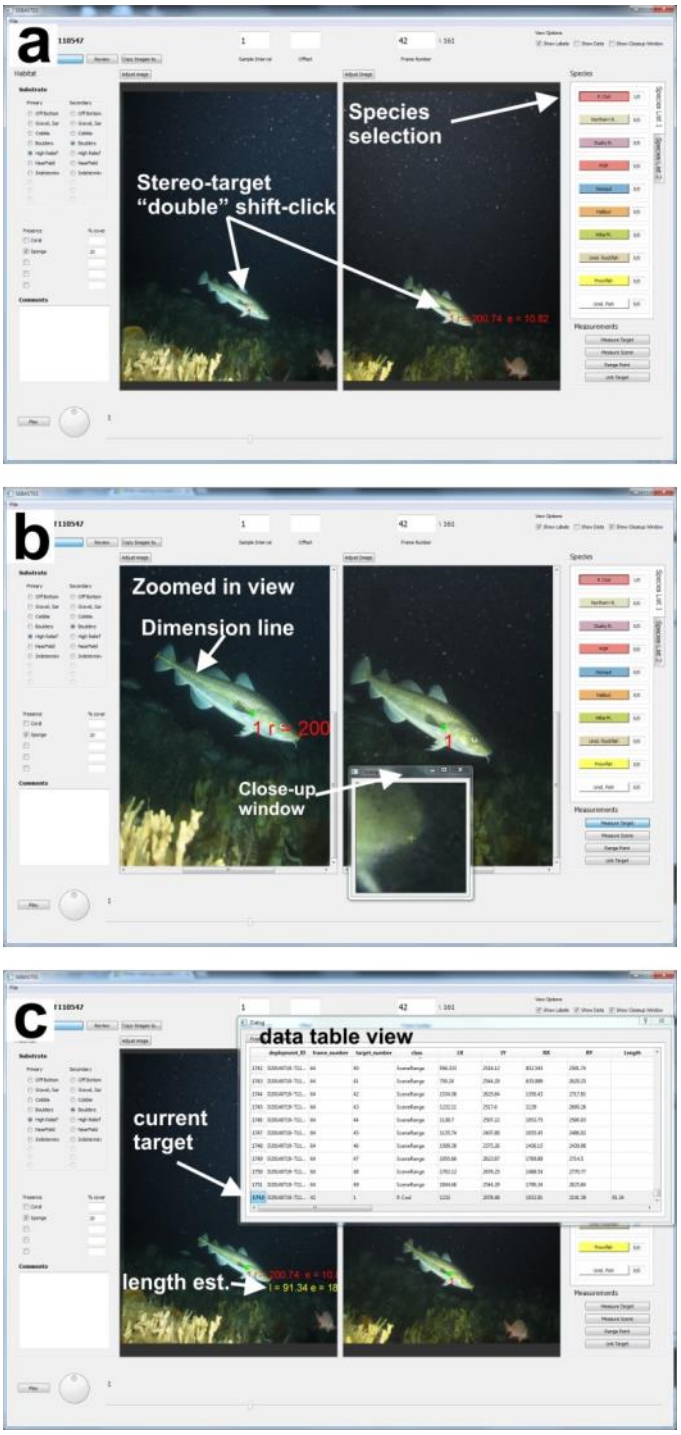


Figure 9. -- Example of fish measurement in SEBASTES. Panel a shows initial stereo target selection (a Pacific cod in this example). Once the target is selected, rolling the mouse zooms in the fish for measurement. In panel b, the *Measure Target* is depressed and measurement takes place. Panel c shows the resulting length estimate and the row added to the data view table.

Scene Ranging and Measurements

It may be desirable to make measurements of the image space not necessarily associated with any target. These are done using the *Measure Scene* and *Range Point* buttons. These actions do not require an active target, so they are shown in a red color. As a first step, press the *Range Point* button. It will stay depressed as the user clicks on two analogous points or features in the images: for example, an easily discernable feature of a coral head. Once the program receives two points, one from each image, the *Range Point* button becomes unchecked and the range value is shown on the image (and 3D points written into the database). Determining the range of locations in images is especially useful when trying to reconstruct a 3D scene or to estimate 3D seafloor features.

Target Association

Some individual fish may be seen from one frame to the next. SEBASTES allows the user to link targets across frames using the *Link Target* button. If the button is pressed, the target number of the currently active target in frame (t) will be written in the record for the next target identified in frame (t+1). For example, a fish is selected as active by right clicking on the circular mark, and then the link target button is pressed. It stays checked as the user presses the right arrow key to go to the next frame. In this frame the user indicates the same fish by clicking on it. The *Link Target* button becomes un-checked. The target is now associated between frames. This functionality only works moving forward. If the target is not present in the following frame, simply uncheck the *Link Target* button to cancel the linking.

Data Viewing

Data are being continuously written into a SQLLite file (*.sql3) as fish are being counted, measured and habitat is being classified. To view the data in tabular form, check the *Show Data* checkbox, which will then display the data on two tabs, one for frame level data and the other for target level data. This can also be useful for navigation, as the user can select a row for a particular target and the frame with that target will appear, with that target being selected as active.

Deleting Targets

Targets can be deleted by selecting them or making them active (right click or select from data view) and pressing the delete button. The user is then asked to confirm the deletion action, and if that is done, the target is permanently removed from the data.

Target Classification Review

The Review button in the upper left corner of the GUI provides a method for quickly reviewing all targets of a certain class as a secondary quality control measure. For example, a primary analysis may code some fish targets as “Fish Unidentified” due to limited experience by the analyst. Afterwards, the review feature can allow a more experienced reviewer to further classify the targets. In the following example, a reviewer wants to check identification of a northern rockfish (*Sebastes polyspinis*). They set the current class dropdown to the northern rockfish and set the range of frames they want to inspect. The software then advances through frames containing that class and highlights the targets of that class. If the classification is correct,

the reviewer just skips to the next target, or can change the class at that point to the one indicated in the “change to” dropdown. The review window is shown in Figure 10.

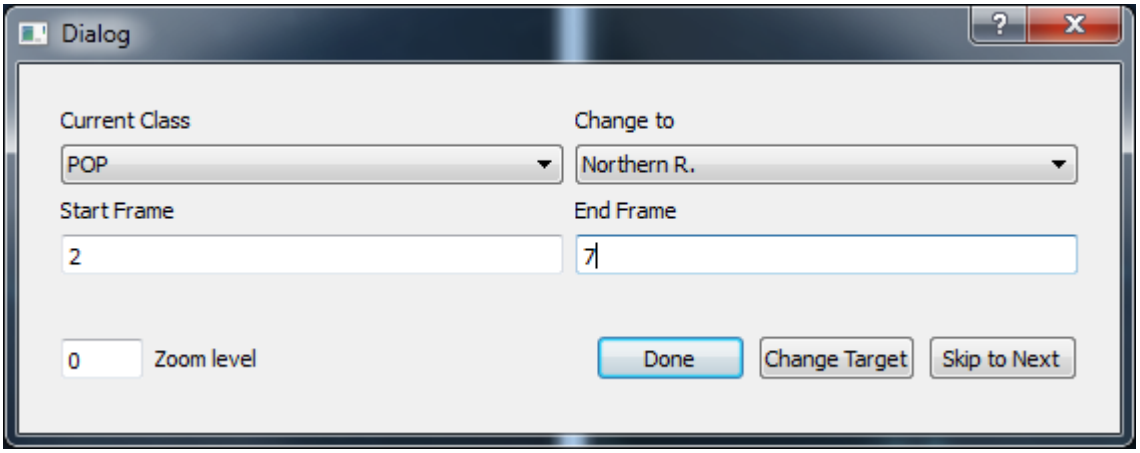


Figure 10. --The species classification review dialog in SEBASTES.

Data Outputs

When a user finishes an analysis session and closes the application, the data contained in the .sql3 file is mirrored into an EXCEL spreadsheet with the same name as the deployment, and contained inside the “data” folder created by the application. Tables 2 and 3 present a detailed description of all the fields found in these outputs.

Table 2. -- Fields in the *frames* table in the .sql3 and .xls data output files.

deployment_ID	The name of the primary data folder, meant to represent the identification of a deployment, or a sequence of stereo pair images
frame_number	Number of the stereo-pair frame number as extracted from the image file name
frame_time	The time stamp of the frame, taken from the image name or from the exif data

primary_habitat_type	Primary habitat type as indicated by an analyst choice from the possible types listed in the template file
secondary_habitat_type	Secondary habitat type as indicated by an analyst choice from the possible types listed in the template file
BX_present	Boolean Flag indicating whether coverage type X is present (max 5 types) - NULL if no coverage X specified
BX_coverage	Numeric value indicating percent coverage of coverage type X - NULL if no coverage X specified
AX_checked	Flag indicating whether additional condition tag X (max = 2) is checked - NULL if condition not specified
comment	A 1000 character text comment field

Table 3. -- Fields in the *targets* table in the .sql3 and .xls data output files.

deployment_ID	The name of the primary data folder, meant to represent the identification of a deployment, or a sequence of stereo pair images
frame_number	Number of the stereo-pair frame number as extracted from the image file name
target_number	Number of the target in a given frame
class	Target class - either one of the options under the species list in the template file or "SceneRange" for ranged points or "SceneMeasurement" for measurements of non-targets in the scene
LX	Pixel column of the target mark in the left frame (0 base, left to right order)
LY	Pixel row of the target mark in the left frame (0 base, up to down order)
RX	Same as LX for right image - null for targets only in left frame and vice versa
RY	Same as LY for right image
Length	Length estimate in cm for the given target (see appendix for computation method)
Range	Estimate of target distance to the left camera based on points LX,LY and RX, RY
Error	Estimate of the deviation between points labeled on image and back-projected points, normalized for image resolution. See appendix for more details.
LHX	For measured stereo targets this is same as LX for the point indication the snout
LHY	For measured stereo targets this is same as LY for the point indication the snout

LTX	For measured stereo targets this is same as LX for the point indication the tail
LTY	For measured stereo targets this is same as LX for the point indication the tail
RHX	Same as LHX for right image
RHY	Same as LHY for right image
RTX	Same as LTX for right image
RTY	Same as LTY for right image
hx	3D x coordinate of head in cm with origin in optical center of left camera. Also 3D x coordinate for ranged point or target no measured for length.
hy	3D y coordinate of head in cm with origin in optical center of left camera (Also same as above)
hz	3D z coordinate of head in cm with origin in optical center of left camera. Also same as above.
tx	3D x coordinate of tail in cm with origin in optical center of left camera. NULL for ranged point or target no measured for length
ty	3D y coordinate of tail in cm with origin in optical center of left camera. NULL for ranged point or target no measured for length
tz	3D z coordinate of tail in cm with origin in optical center of left camera. NULL for ranged point or target no measured for length
target_link	The target number in previous frame linked to this target
comment	A 1000 character text comment field

Camera Calibration Application

As part of the SEBASTES application, a basic calibration application was constructed to allow easy camera and stereo calibration using the OpenCV computer vision library (Bradski 2000). Like SEBASTES, the application is written in Python using the QT GUI toolkit and utilizing pyOpenCV (http://pythonhosted.org/pyopencv/2.1.0.wr1.2.0/index.html), which is a Python function library that links to the OpenCV (written in C++) functions.

General Notes on Performing an Underwater Camera Calibration

The goal of the stereo calibration procedure is to capture 20-30 image pairs of a flat checkerboard pattern with known dimensions (Fig. 11). The checkerboard should be visible in both images. To reduce ambiguity in vertical and horizontal orientations, it is best to use a checkerboard with a different number of columns and rows. The checkerboard should be imaged at ranges that approximate expected viewing distances to the fish targets. Images should encompass a variety of viewing angles and positions within the view field. For a detailed example of how to capture a set of calibration images, please see http://www.vision.caltech.edu/bouguetj/calib_doc/index.html#examples or http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html#calibration.

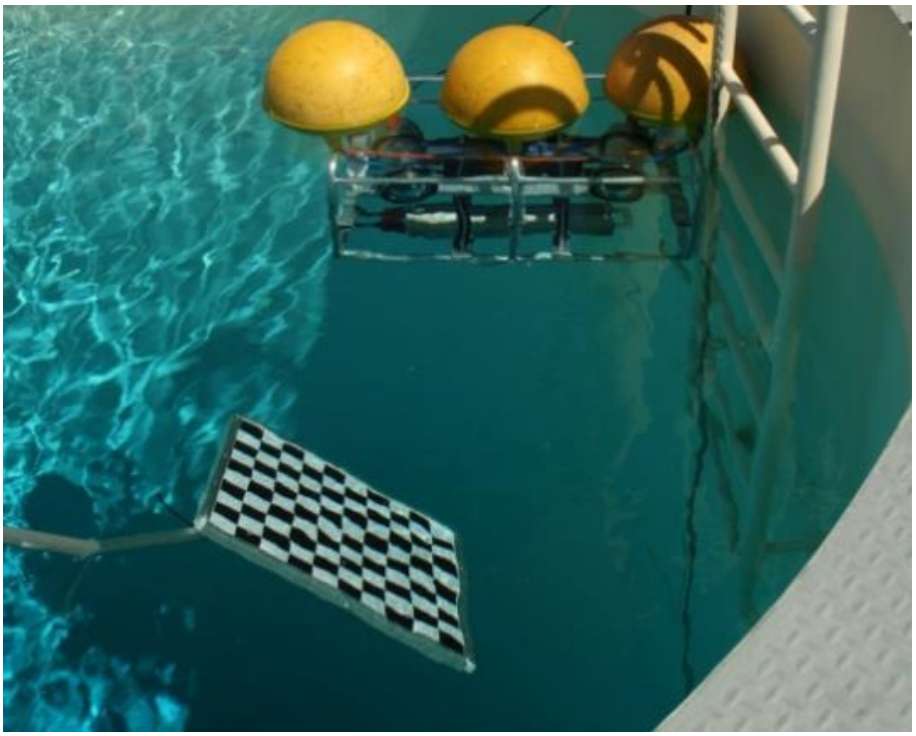


Figure 11. -- Underwater stereo camera undergoing calibration

Usually, a large set of images is collected and then subsampled to identify 20-30 good pairs. For the calibration procedure, the left and right images should be placed in a single folder with image names indicating the pair number and camera designation (e.g., “left_01.jpg”). Ideally the calibration images would be collected under similar environmental conditions as the data due to the influence of temperature and salinity on refraction; however, these effects usually only contribute a small amount (< 1%) to overall uncertainty in the stereo methodology.

Conducting a Calibration Using StereoCalibrate

Once the application is started, press the *New Cal* button. The calibration does not have to be completed in a single session; all work is saved in a .sql3 file and can be reloaded using the *Reload* button. When starting a new calibration, the following dialog window is shown (Fig. 12):

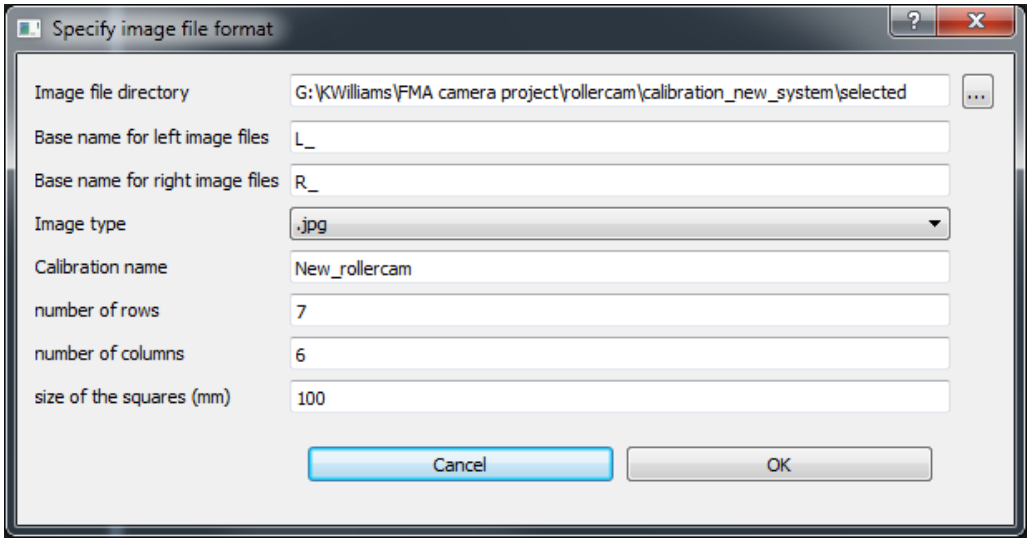


Figure 12. -- Loading a calibration image set into the StereoCalibrate application.

All of the fields are required. Currently only jpg, png and tif image formats are supported. The number of row and column intersections refers to the intersections of the black and white squares, excluding any outer boundary as shown below in Figure 13.

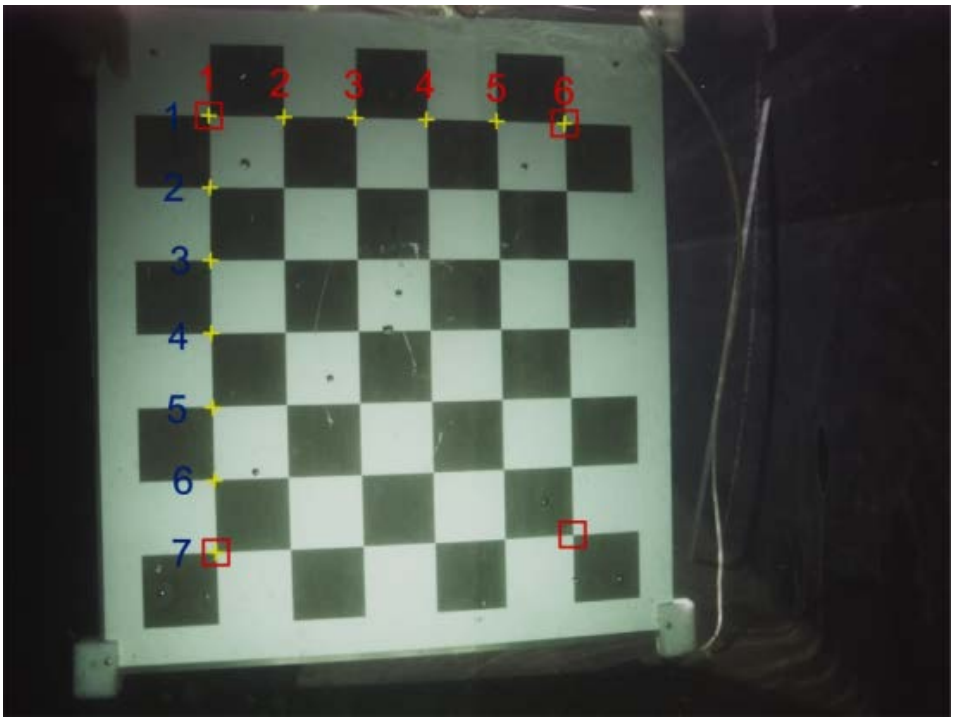


Figure 13. -- Identifying checkerboard calibration pattern dimensions. Valid intersections are shown in yellow and enumerated in blue for rows and red for columns. In this example, number of row intersections = 7 and column intersections = 6.

After pressing OK on this dialog, the images should load and the user then can extract the corners from each pair. The task is to click on the four corners that define the legitimate portion of the checkerboard (in example above, a 7 × 6 grid, with corners indicated by the four red squares). The calibration algorithm will try to locate the exact corner using a Harris corner detector algorithm (Harris and Stephens 1988), making it unnecessary to click on the exact intersection of the squares. The search for the corner finding algorithm is set by the user in the *search box dim* on the lower portion of the window (Fig. 14).

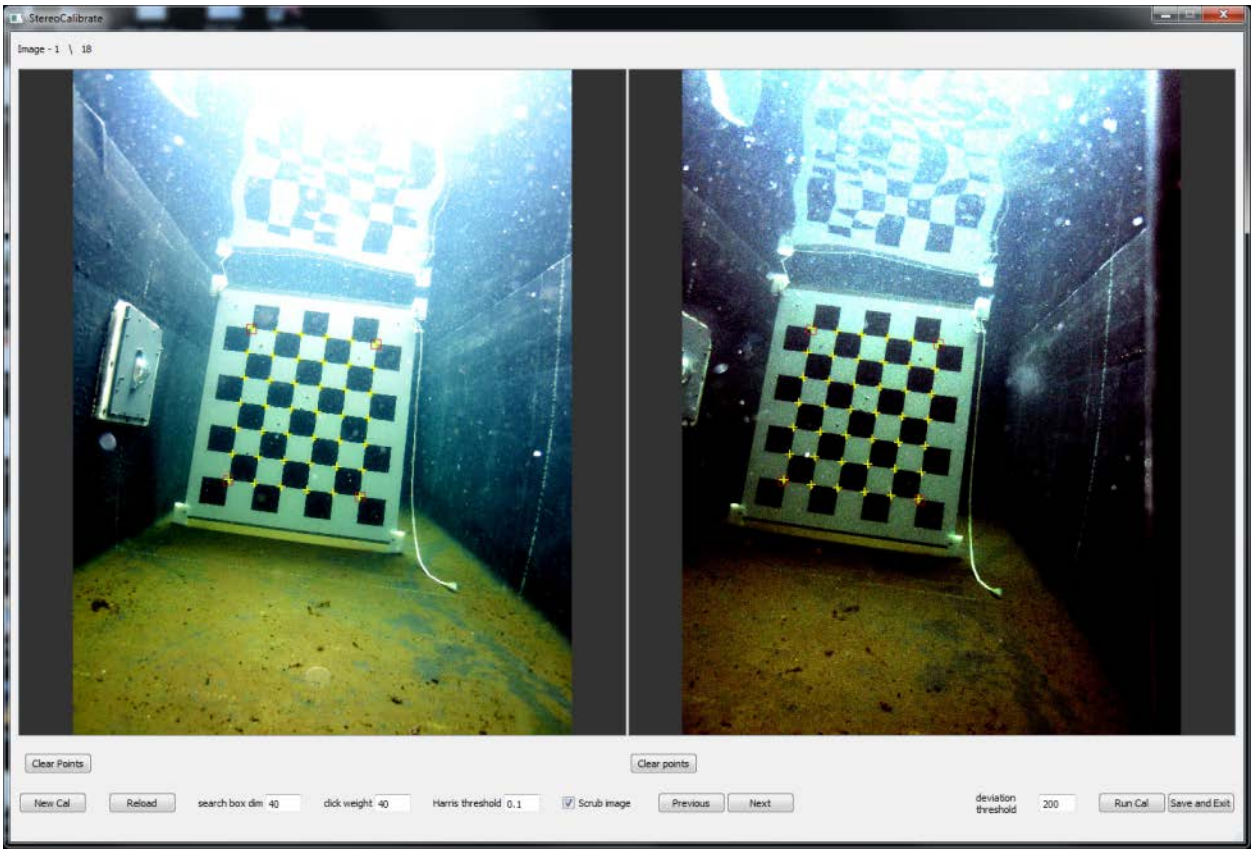


Figure 14. -- The stereo calibrate main GUI window.

If the corner detector is inaccurate (this can happen in low contrast images or where water is high in suspended particulates), you can modify the Harris threshold value (valid values are between 0 and 1). Another option is to uncheck the “scrub image” checkbox. This will then use the raw image for finding corners rather than a contrast enhanced image. An alternative is to increase the “click weight” value, which weights the Harris corner finder by how far it is from the click location. If this value id > 40, the corner will be estimated at exactly where the cursor is located when the user clicks.

To zoom in on the image, press and hold the *Control* key while moving the mouse roller. To pan around the image, press and hold the *Alt* key while holding mouse button down (cursor will turn into a hand).

After the four corners are extracted, all the intermediate points should fill in. If this fails or corners do not look good, press the *Clear Points* button and try again, after changing corner finding settings. If the point extraction looks adequate in both images, press the *Next* button. This will save the click positions for use in the calibration. If adequate points cannot be extracted after trying several settings, clear the points and proceed to the next pair. All point extractions can be reviewed by pressing the *Previous* button.

After all the images are processed, press the *Run Cal* button. This will perform the calibration and output the results into a pop-up dialog. The dialog will indicate if particular pairs exceed the deviation threshold, perhaps indicating non-synchronous pairs or bad points, you can run the calibration excluding these. The rejection rate can be set by changing the deviation threshold value. The objective is to have 15-20 good images for an acceptable calibration.

A quick basic check of the calibration results can be made by looking at the translation matrix, which represents how far away the right camera is relative to the left in units specified in the setup dialog (mm, Fig. 12). The inter-camera distance should be

$$d = \sqrt{T(1)^2 + T(2)^2 + T(3)^2}\,,$$

where T is the three element vector making up the translation matrix. This distance should closely match the known camera separation value.

If the results are adequate, press the *Save and Exit* button, which will create an .npz file (Python Numpy data structure). This file can be specified in the SEBASTES configuration template file as the calibration file for stereo data analysis.

Stereo Tracker Application

The ability to track individuals across frames exists in the main SEBASTES interface (see *Target Link* function); however, if tracking of animals is the primary objective of an analysis the Stereo Tracker application is optimized for this purpose. This application shows the previous image frame pair in the upper panel, and the current slide in the lower panel (Fig. 15). The user interacts with the lower panel by creating new tracks and clicking on the same fish in the left and right frames (marks are shown as red circles), advancing one frame and clicking on the same fish from the previous frame in both images, until the fish passes out of view. The previous frame indicates the target marks for the previous tracking step (blue circles) to assist in keeping the tracks consistent. This is especially helpful when tracking individuals in schools of similar appearing fish. The stereo-triangulation back-projected point is shown as a red cross, and this allows the user to detect when there are large departures from epipolar constrains; for example, when the targets in left and right image do not match, this will result in differing locations of the circles and crosses in the upper panel. To correct this, it is necessary to go back to the previous frame and re-click in each window. This will overwrite previous target position. Moving from frame to frame is accomplished by either pressing the *Previous* and *Next* buttons or using the arrow keys on the keyboard with the cursor in the image window.

The analysis requires that the deployment first be opened in SEBASTES to create the data folder and structures. In addition to the existing *Frames* and *Targets* tables, the tracker

creates a new table called *Tracks*, and lists the individual target positions for each track. The available classes attached to each track come from the LocalTemplate.xls file.

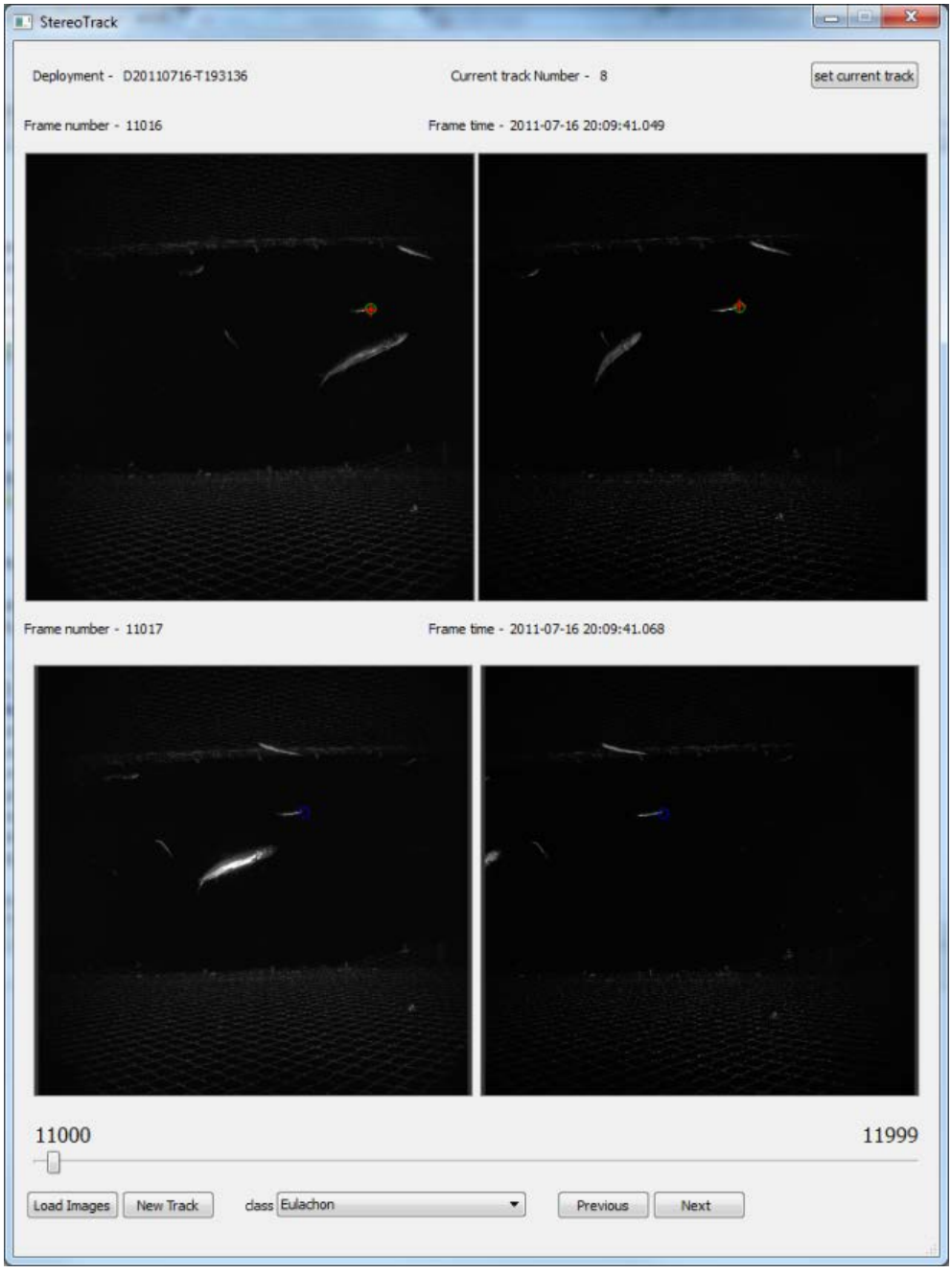


Figure 15. – Stereo tracker application for optimized manual fish tracking in 3D.

Citations

Bouguet, J.Y., 2008. Camera calibration toolbox for Matlab [online]. [Available from http://vision.caltech.edu/bouguetj/calib_doc/index.html (accessed September 2008)].

Bradski, G., 2000. OpenCV. Dr. Dobb's Journal of Software Tools. [http://www.drdobbs.com/]

Harris, C., and M. Stephens. 1988. A combined corner and edge detector, pp. 147–151. Proceedings of the Fourth Alvey Vision Conference. University of Manchester, 31 Aug. to 2 Sept. 1988.

Stein, D. L., Tissot, B. N., Hixon, M. A., and Barss W. 1992. Fish-habitat associations on a deep reef at the edge of the Oregon continental shelf. Fish. Bull., U.S. 90:540-551.

Yoklavich, M.M., H.G. Greene, G.M. Cailliet, D.E. Sullivan, R.N. Lea. and M.S. Love, 2000. Habitat associations of deep-water rockfishes in a submarine canyon: an Example of a natural refuge. Fish. Bull., U.S. 98:625-641.

Appendix

Software Code Overview

The descriptions and specifications below are applicable to the software state at the time of writing. Versions of underlying code libraries may be updated to maintain compatibility with operating system requirements of for the addition of new features to the software. All URLs referenced in the text may also become obsolete as web sites change their contents and structure.

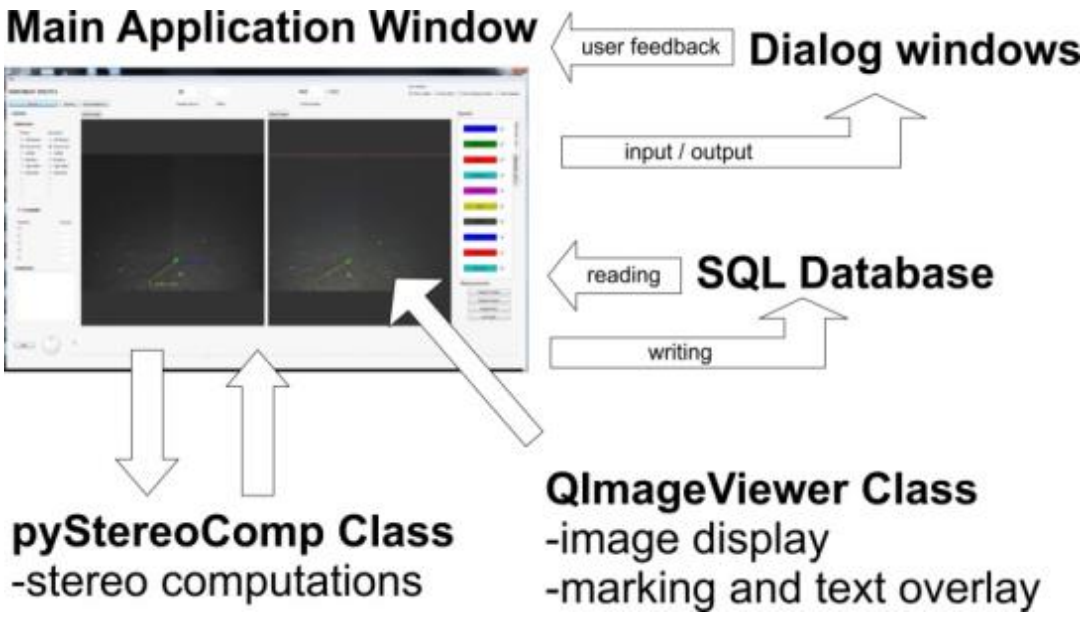
SEBASTES consists of several code components. The primary application involves a main Python script (MainWindow.pyw), which involves a GUI object application form using PyQt (ui_SEBASTES.py) located in the *ui* folder. Separate python scripts exist for all the dialog windows (*.py), including individual GUI forms (ui_*.py). The latter are located in the *dialogs* folder. StereoCalibrate.pyw is the script that runs the StereoCalibrate application. Python version 2.7 was used for all development.

Appendix Table A1. -- A list of additional Python packages (not included in the Python standard library) used in SEBASTES.

package	version	Description
Python 2	2.7.8	Main code environment
PyQt4	4.11	Python bindings for Qt4
Numpy	1.8.2	Numerical Python for matrix operations
Scipy	0.14.0	Scientific Python – library for mathematics, science, and engineering

pyOpenCV	2.1.0	Python bindings for the openCV open-source computer vision library
xlrd, xlwt, xlutils	N/A	Microsoft Excel support for python

For convenience, SEBASTES is also packaged with WinPython (https://sourceforge.net/p/winpython/wiki/Welcome%20to%20WinPython!/), a self-contained, portable Python distribution for windows that enables the user to run SEBASTES without having to install Python or any additional required packages.



Appendix Figure A1. -- Main code components of the SEBASTES application environment.

GUI Design Tools

SEBASTES utilizes the PyQt GUI development toolkit (https://riverbankcomputing.com/), which provides the basic classes for collecting user inputs, displaying data, and the SQL database functionality. PyQt is a Python bindings library for the Qt cross-platform application framework (http://www.qt.io/), utilizing open source GPL v3 licence. SEBASTES was constructed using PyQt version 4 (PyQt4), which uses Qt version 4.3.

pyStereoComp Functions

importCalData(*filename*)

- Imports calibration file
  - If calibration was performed using the Matlab camera calibration calibration toolbox (Bouguet 2008), the pyStereoComp operational mode is set to “matlab”. The Matlab calibration file (*.mat) is imported using the Scipy mat file reader
- If calibration is performed using the stereo-calibrate application, the resulting Numpy data file (*.npz) is imported and operational mode is set to “openCV”.
- Output consists of a python dictionary class object (“*calibration dictionary*”) with the following fields:

  T – translation matrix (1×3)

  R- rotation matrix (3×3)

  om[1]- rotation vector (1×3)

  F[2] – fundamental matrix (3×3)

  fc_left[1] – focal length in pixels (2×1, vertical & horizontal)

  cc_left[1] – principal point coordinates (2×1, vertical & horizontal)

alpha_c_left[1] – angular skew coefficient, x to y (scalar)

kc_left[1] – radial and tangential image distortion coefficients (5×1)

cameraMatrixL[2] – camera matrix containing distortion and other parameters (3×3)

The last five parameters are repeated for the right camera. If the calibration origin is OpenCV, they are derived from the respective camera matrices. Superscripts denote 1) only needed for "matlab" operation mode, and 2) only used in "openCV" mode.

XL,XR = **triangulatePoint**(*xL, xR*)

in "matlab" mode:

- A python transcription of *stereo_tringulate.m* function from the Matlab camera calibration toolbox.
- Inputs:

  xL, xR – pixel coordinates (x,y) of an analogous point in the left and right image

  calibration dictionary - (see above)
- Outputs:

  XL, XR – real space coordinates (x,y,z) in the left (XL)or right camera origin coordinate space

In "openCV" mode:

- Using the *triangulatePoints* function in the pyOpenCV library (python bindings for openCV functions; V 2.1.0; http://pythonhosted.org/pyopencv/2.1.0.wr1.2.0/index.html)
- Inputs and outputs same as Matlab version outlined above

x = **normalizePixel**(*x_kk, fc,cc,kc,alpha_c*)

- Removes distortion from pixel coordinates, resulting in pinhole camera model approximation. Direct Python (Numpy) translation of *normalize_pixel.m* in Matlab camera calibration toolbox. Required for "matlab" operational mode. Called by *triangulatePoint.*
- Inputs:

  fc,cc,kc,alpha_c - camera distortion, principal point, and focal length parameters for given camera

  x_kk – raw pixel coordinates containing distortion
- Outputs*:*

  x – undistorted coordinates

x = **compDistortion**(*xd, k*)

- Direct Python (Numpy) translation of *comp_distortion.m* in Matlab camera calibration toolbox. Required for "matlab" operational mode. Called by *normalizePixel*.

om = **rodrigues**(*R*)

- Direct Python (Numpy) translation of *rodriguez.m* in Matlab camera calibration toolbox. Required for "matlab" operational mode. Called by *importCalData.*
- Inputs:

  R – rotation matrix
- Outputs:

  om – rotation vector

xd = **applyDistortion**(x, k)

- Direct Python (Numpy) translation of *apply_distortion.m* in Matlab camera calibration toolbox. Opposite of comp_distortion. Required for “matlab” operational mode. Called by *projectPoint.*

xP = **projectPoint**(X, I)

- Projects 3d points onto the image plane using the calibration coefficients

  in “matlab” mode:

- Python (Numpy) translation based on translation of project_points2.m in Matlab camera calibration toolbox.

  in “openCV’ mode:

- Uses the pyOpenCV function *projectPoints* in the pyOpenCV library (python bindings for openCV functions; V 2.1.0;

  http://pythonhosted.org/pyopencv/2.1.0.wr1.2.0/index.html)

- Inputs:

  X – 3D point vector (x,y,z)

  I – camera flag (string: ‘L’ or ‘R’)

- Outputs:

  xP – projected 2D point pixel coordinates (x,y) for camera I image.

E = **computeError**(*xL,xpL,xR,xpR*)

- Computes the Euclidian distance between the points in left and right images clicked on by the user and assumed to represent corresponding spatial locations and the “back-projected” points based on first deriving the 3D coordinates for the manual inputs and then projecting them back onto the image plane.
- Inputs:

  xL,xR - pixel coordinates (x,y) of an analogous point in the left and right image

  xpL,xpR - projected 2D point pixel coordinates (x,y) for left and right camera image.
- Outputs:

  E – scalar indicating pixel distance between the sets of points

Dialogs

The main SEBASTES script calls several subscripts as dialog windows to perform specific sub-routines or provide additional data for the user. Below is a list of all these dialogs and their basic description.

Appendix Table A2. -- SEBASTES dialog windows and description.

classdlg.py	Used to change assignment of species to a count button in SEBASTES. Presents users with available species listed in template file.
addclassdlg.py	Allows users to add new species in SEBASTES. Activated in the classdlg.py
closeupdlg.py	Shows a zoomed in view of the cursor position for more precise lengthing
dataviewdlg.py	Shows a tabular view of the two results tables; the *frames* table and *targets* table
reviewdlg.py	Allows user to review a set of targets for species classification
imgfiledlg.py	Used to set up a calibration data set in the StereoCalibrate application
infodlg.py	Used to display the results of the calibration in the StereoCalibrate application

QImageViewer

This Python class was created by Rick Towler (rick.towler@noaa.gov) specifically for use in Qt applications where images need to be displayed and overlaid with annotation. It provides support for making dimension line markings, zooming and panning in images, standard image adjustments, HUD displays, and text marks. It is used in SEBASTES and in StereoCalibrate applications.