Advances in Artificial Intelligence and Machine Learning; Short Communication 3 (4) 1768-1786 Received 18-10-2023; Accepted 21-12-2023; Published 28-12-2023

A Horizon Line Annotation Software for Streamlining Autonomous Sea Navigation Experiments

Yassir Zardoua yassirzardoua@gmail.com
FSTT, Smart Systems Emerging Technologies,
FSTT, Abdelmalek-Essaadi University, Tetouan,
Morocco

Brahim Beguiel Bergor
Industrial Systems Engineering and Energy Conversion,
FSTT, Abdelmalek-Essaadi University, Tetouan,
Morocco

Abdelhamid El Wahabi
Industrial Systems Engineering and Energy Conversion,
FSTT, Abdelmalek-Essaadi University, Tetouan,
Morocco

Anouar El Mourabit
Industrial Systems Engineering and Energy Conversion,
FSTT, Abdelmalek-Essaadi University, Tetouan,
Morocco

Moussab Chbeine
Computer Science and Smart Systems , FSTT,
Abdelmalek-Essaadi University, Tetouan,
Morocco

Abdelali Astito
Smart Systems Emerging Technologies, FSTT,
Abdelmalek-Essaadi University, Tetouan,
Morocco

Corresponding Author: Yassir Zardoua

Abstract

Horizon line (or sea line) detection (HLD) is a critical component in multiple marine autonomous navigation tasks, such as identifying the navigation area (i.e., the sea), obstacle detection and geo-localization, and digital video stabilization. A recent survey highlighted several weaknesses of such detectors, particularly on sea conditions lacking from the most extensive dataset currently used by HLD researchers. Experimental validation of more robust HLDs involves collecting an extensive set of these lacking sea conditions and annotating each collected image with the correct position and orientation of the horizon line. The annotation

Citation: Yassir Zardoua, et al. A Horizon Line Annotation Software for Streamlining Autonomous Sea Navigation Experiments. Advances in Artificial Intelligence and Machine Learning. 2023;3(4):102.

task is daunting without a proper tool. Therefore, we present the first public annotation software with tailored features to make the sea line annotation process fast and easy. We will provide the software and an 8-minute video tutorial at: `https://drive.google.com/drive/folders/1aGUlhw4PWsKWQ_FK_txoicmeykcqE4zm?usp=drive_link`.

Keywords: Horizon line, Sea-sky line, Annotation software, Horizon line dataset, Maritime video processing, Autonomous sea navigation.

1. INTRODUCTION

1.1 Definition of the Horizon Line

In maritime images captured from terrestrial platforms, such as buoys, ships, and USVs, the horizon feature (sometimes called the sea-sky line) is defined as the line separating the sea region and the region right above it [1] (see FIGURE 1(a) and 1(b)). The literature includes several ways to represent the location of the horizon line. The position and tilt representation, which we denote by the pair of parameters $\{Y, \phi\}$ (see FIGURE 1(c)), is by far the most common and useful representation [1–3], because it provide a direct and precise measure of the horizon's location. Additionally, measuring the detection error based on this representation provides a clear and intuitive idea of how well the horizon detection algorithm is performing.

1.2 Applications of the Horizon Line

The published literature shows that the sea horizon line exhibits a wide range of applications, especially in terms of autonomous navigation. In the case of autonomous sea navigation, the horizon feature is involved on almost every stage of the processing pipeline. We'll cite four major application categories. First, numerous papers and patents used the horizon line to auto-calibrate the camera [4–6], which include non-maritime scenes as well [7, 8]. Second, the position and tilt of the horizon allows the computation of rotation and translation matrices that would digitally stabilize captured video frames [9–17]. A stabilized video is not only easier to watch but facilitates subsequent analyses as well [2, 11, 17]. Third, many works used the horizon to segment the image (into sea and non-sea region). In the same context, the horizon feature provides computational benefits to the object detection task by reducing the search region down to that around the horizon or by eliminating the sky pixels that will obviously not depict any obstacle [5, 18–35]. Fourth, even after the detection of maritime objects on the image coordinates, some researchers used the horizon line to geo-localize surrounding objects and obstacles [36–39], i.e., find the distance in meters from the camera to the obstacle.

2. RELATED WORK

This section discusses a comprehensive set of the most relevant and existing annotation software that could be used for the purpose of annotating the horizon line. Based on such discussion, we

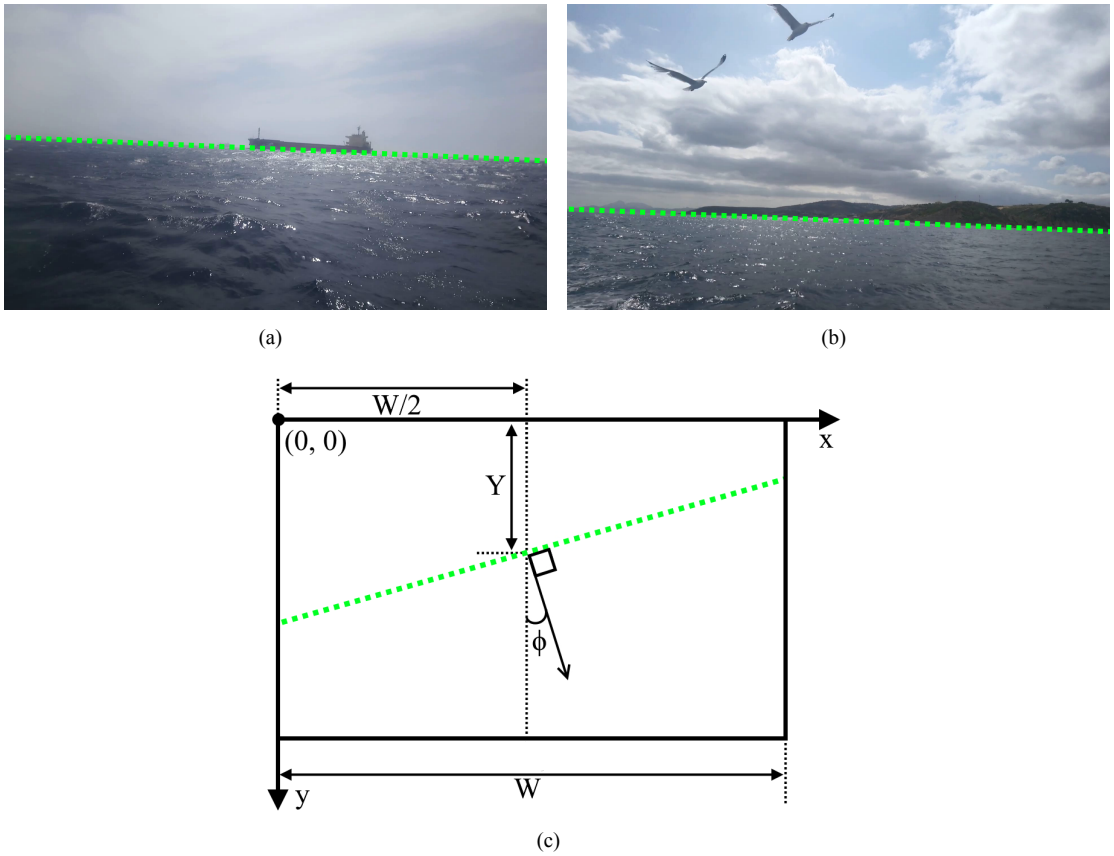


Figure 1: Definition and representation of the horizon: the horizon line separates the sea from the sky (a) and the sea from the coast (b); (c) $\{Y, \phi\}$ representation on Cartesian coordinates xy of images

highlight the need for our customized software and its unique features, setting it apart from other tools.

2.1 Existing Annotation Software

Given the rise of machine learning research in the recent decade, various annotation software have been developed. We will discuss the most relevant software, particularly the tools that include the feature of line annotation for image and video data. Firstly, we will discuss the tools that have been used by maritime video processing researchers for line annotation; the authors of the most extensive horizon line dataset, the SMD (Singapore Maritime Dataset) [2], state that a Matlab annotation tool was used to label the horizon line on each video frame. However, this tool or its nature is private and was not provided along with the annotated dataset shared in this footnoted link[1]. Hashmani and Umair [40], used *SuperAnnotate* software [41], to label the horizon line on

[1] `https://sites.google.com/site/dilipprasad/home/singapore-maritime-dataset?authuser=0`

their datasets. *SuperAnnotate* is a paid software and the authors of [40], reported that they had to perform a post processing on the generated annotation file to produce the final GT (Ground Truth) labels for the horizon line.

Authors of the recent horizon line detection algorithm [42], used *labelme* [43], which labels image classes by drawing polygons on the objects to label. This tool can be used for line annotation by stopping the polygon drawing process at the second point. However, the authors of [42], report that they annotated image files, not videos. After exploring the interface and features of *labelme*, it seems that the annotation of video files is not supported, as evidenced in FIGURE 2. The availability of annotated video files, instead of image files, is a necessity and not a matter of preference. We justify this claim by the fact that some horizon detection algorithms, such as [13, 44, 45], rely on the temporal information available only in video sequences to detect the horizon line, making their evaluation on image files impossible. The incorporation of the temporal information in horizon detection algorithms is natural and beneficial since the eventual applications (e.g., video stabilization, target tracking), which we highlighted in 1.2, require video input.

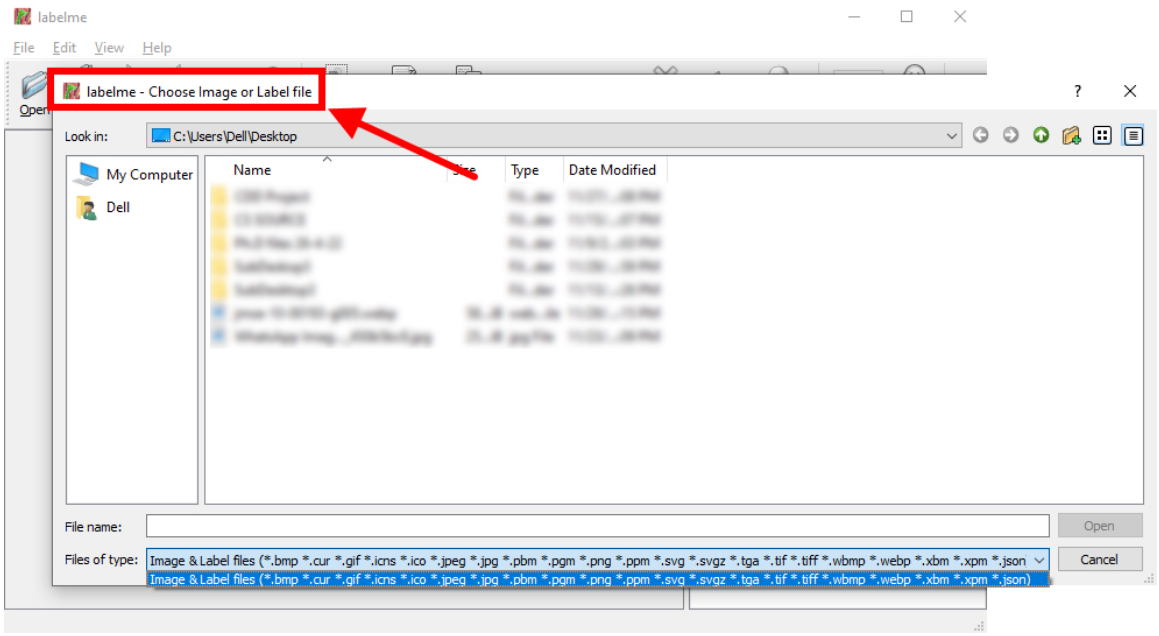


Figure 2: The main interface of *labelme* and its *open file* window; the red arrow indicates that *labelme* supports only image files

The reader may find other annotation software, with the most popular options being *ReactLabel* [46], *CVAT* (Computer Vision Annotation Tool) [47], and *Supervisely* [48]. *RectLabel* is available only on MacOS users and it is not tailored for swift line annotation. *CVAT* provides a free version with an online access to the annotation server, but it does not allow a frame by frame annotation, at least in its free version. Also, the line annotation process is not comfortable because the user must activate the polyline tool for each annotation and manually indicate the stopping point as the second one. *Supervisely* is also a paid software but offers a free version for video annotation, but the free trial has a limited period, and it does not allow to annotate data exceeding a certain threshold.

It is worth mentioning that even if the user buys one of the paid software we mentioned, it would be much more challenging to annotate video files for the horizon line when compared to the tailored software we developed, mainly because the user will have to distinguish only the line annotation features among all the features that are originally dedicated to the other computer vision labeling tasks, such as object detection, segmentation, and pose estimation. Moreover, as evidenced by the work of [40], the output annotation file contains a lot of unnecessary information and the user will have to perform a post processing to extract the line parameters and put them in a proper iterable format for the eventual algorithm test.

The software we propose is free, open-source, and designed specifically to annotate video files for the horizon line. The user can open the interface and start the line annotation task directly, without having him to figure out which tool to use and which tool to ignore. Moreover, the user can save only one annotation file for each video as simple and intuitive format representing only the information required for the experimental validation. While the tailored nature of our software represents its key strength, it integrates several features allowing an easy and comfortable annotation process. We discuss in Section 2.2 more details regarding the need for a custom annotation tool and the features setting it apart from the other software.

2.2 The Need of Custom Horizon Annotation Tool

Firstly, we discuss the need for a proper tool to annotate the sea horizon line, which we clarify in the following points:

- The most extensive dataset, the SMD, lacks various sea conditions that must be collected and properly annotated
- The sea line annotation of the SMD include significant mistakes and must be corrected before the experiment, which requires an annotation tool.
- Most horizon detection papers experiment with their own video dataset but do neither mention how did they annotate their data nor share the used software

Currently available tools [41, 43, 46–48], lack public free access or customization for quick and easy annotation, as we explained in Section 2.1. The main novelty of our software is that it freely provides tailored features focusing on quick and easy annotation of horizon lines; Section 3 outlines the primary components and features of the proposed software, which we summarize below:

- An intuitive graphical user interface (GUI) containing only the features needed for the line annotation task. Thus, the user will avoid wasting time and effort identifying which features he needs and which features he ignores.
- Unlike other software, the generated annotation file contains only the line parameters that are widely accepted among the maritime video processing researchers [1, 2, 49–51]. Therefore, the user will not have to write any additional script to post-process the generated annotation file for extracting and computing the relevant parameters and storing them in a proper format.

- Visualization tools, including the cursor's position on the frame's coordinates (x, y), the ability to adjust the thickness of the annotated sea line, extending the starting and ending line points with a full line, the indication of the current frame order, hiding and showing the current annotation, and more.
- Quick and easy line annotation: the user does not need to select an annotation tool for each new image and the entire annotation process requires only a mouse.
- Easy browsing over the video frames (using the mouse wheel), with the ability to browse with customized offsets.
- The option to duplicate current annotations onto previous non-annotated frames, which can save valuable time when annotating videos from stationary cameras.
- The possibility to annotate all video frames only with a mouse, reducing the user movement that becomes significant when they add up over thousands of annotations.
- Misuse warnings, such as alerting the users if they missed one or more frames.
- The user can save an incomplete annotation file and load it later for completion, which is helpful for video files with a large number of frames.
- The ability to load existing annotation files for visual verification of their accuracy.
- There is no need to wait for the upload of large video files since the software is an executable desktop application.
- The software does not store the loaded video into the RAM; it only loads the frame requested by the user using the video file directory.

Overall, most of these features are unique to our tailored software and enhance the functionality and user-friendliness of the user interface.

2.3 A Short Description of the Annotation Process

This section provides a short description of the annotation process, providing a practical understanding of the ease of using our software for the task of line annotation. For a visual explanation of this process, we recommend that the reader watch the 40-second-long video *Short tutorial.mkv* from the link we provided in the abstract. We would like to emphasize that the annotation process is entirely manual. Automating it with horizon detection algorithms could potentially introduce an unfair performance bias, favoring algorithms that employ similar approaches to the one used for automation. Consequently, our software adheres to the manual annotation process, consistent with relevant research like [2, 40].

After the installation, the user will execute the .exe file. The software's interface will pop up and the user can load a video file to annotate by clicking on *Load video file* button. The first video frame will appear on the interface, which can directly be annotated by drawing a line using the left mouse button. The software will automatically extrapolate and draw the full line from the two points of the drawn line, allowing the user to verify if that line matches the horizon. The brown color of

the extrapolated line indicates that it is not yet accepted a valid annotation. The user can validate the annotation by right-clicking the mouse anywhere on the current frame, turning the drawn line in red to indicate its validation. The user can navigate to the next frame by scrolling the mouse wheel upward, allowing him to repeat the same annotation process. After annotating all the frames, the user can click on *Save annotated file* button to generate a ready-to-use GT file, prefixed with the same name as the annotated video file for easy identification of its video correspondence. This entire process can be carried out with the computer's mouse alone.

3. DESCRIPTION OF THE APPLICATION: INTERFACE BLOCKS AND FEATURES

FIGURE 3 shows the first interface that pops-up when running the application. It is subdivided into four blocks: image display, load directories, browse, and annotation. This Section discusses the full details of each of mentioned blocks.

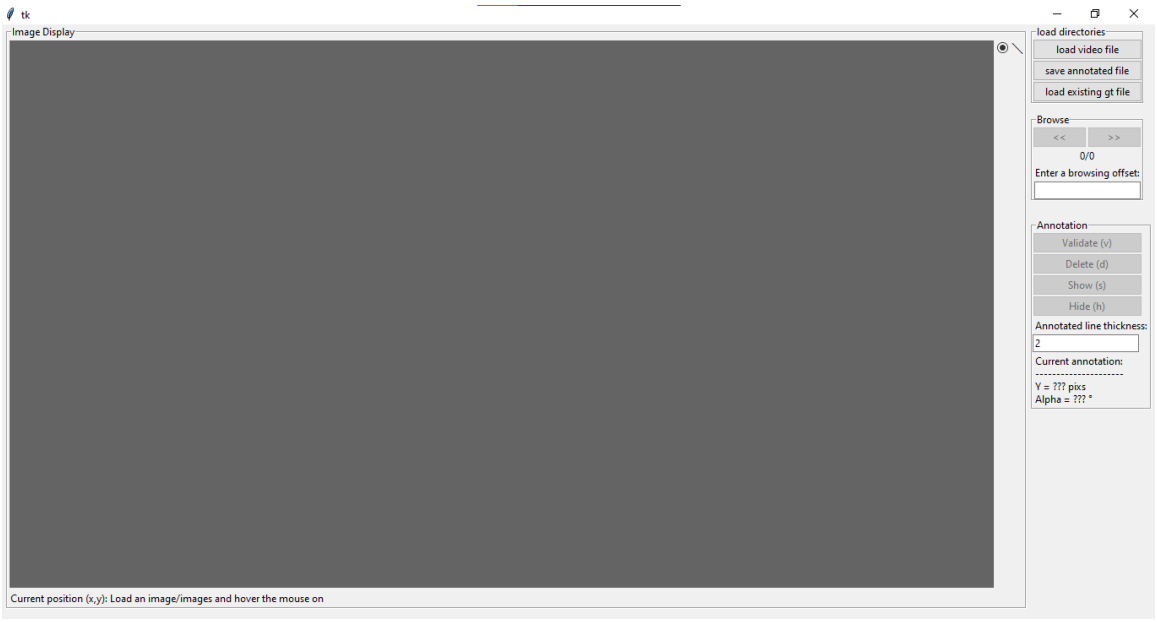


Figure 3: The GUI before video loading

3.1 Load Directories Block

Load directories is the first block to use. As shown in FIGURE 4, the block contains three self-explanatory buttons: *load video file*, *save annotated file*, and *load existing gt file*. The software supports two video file formats: *.avi* and *.mp4*. The loaded video will appear on the *Image display block*, which we detail in Section 3.2. The third button (*load existing gt file*) allows the user to start the annotation process from an existing annotated file (also known as the Ground Truth file). As the

user browses through the video frames, the annotated line of the viewed frame will be drawn in red, as shown in FIGURE 5. Such a feature allows the user to review the annotated frames and correct the eventual mistakes. Whether the user annotates the video from scratch or starts the annotation from an existing GT file, clicking on the *save annotated file* will always save in the directory specified by the user an annotation file by suffixing the current video file name with *LineGT.npy*.

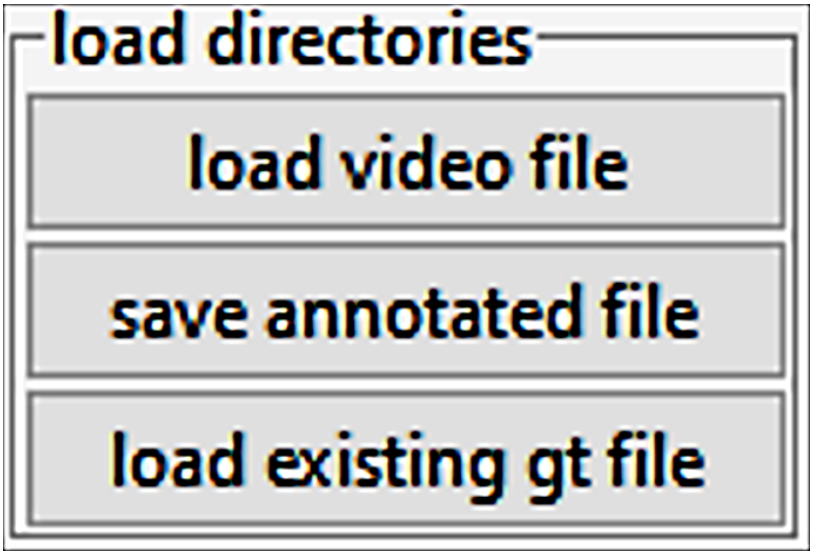


Figure 4: The *Load directories* block

3.2 Image Display Block

The image display block will show the first frame of the loaded video file. The frame size will not change as long as the entire GUI fits into the computer screen. Otherwise, the frame will be down-scaled to the size that would allow the entire GUI to fit onto the computer screen. If the current frame corresponds to an annotated line, that line will be automatically drawn and shown on the displayed frame. The user can draw the annotation line by maintaining the left mouse button. one that button is released, the application will automatically infer and draw the full line on the image display, as shown in FIGURE 6. If the user is not satisfied with the line he is drawing, i.e., before releasing the left button of the mouse, he can abort the drawing process by right-clicking the mouse before releasing the left mouse button. While the mouse is on the image display, its Cartesian Coordinates (x, y) are constantly updated and displayed on the bottom left of the Image display block, as shown in FIGURE 7. We note that the image frame displayed in FIGURE 6 is downsized so that the GUI fits in the screen.

Even in such a case, the Cartesian coordinates displayed are computed to correspond to the original size of the frame. This choice is not arbitrary for two reasons: (1) the downsizing factor changes according to the frame size and the computer screen; (2) the position of the annotated line must correspond to the original frame size.

(a)

(b)

Figure 5: The GUI after loading (a) a video file and an (b) existing annotation file.

3.3 Annotation Block

We show the annotation block in FIGURE 8. The features of this block involve the manipulation of annotated lines. The *Validate* button validates the non-aborted line drawn, such as the line we

(a)

(b)

Figure 6: The process of line drawing: (a) before releasing the left mouse button; (b) after releasing the left mouse button; (c) the mouse position coordinates.

mentioned in FIGURE 6. The *Delete* button deletes the annotated line of the current frame. The *Hide* button hides the annotation of the current frame. It can be shown again using the *Show* button. Instead of using these buttons, the user may prefer keyboard keys. The key corresponding to each

Figure 7: Cartesian coordinates (x, y) of the current mouse position

button is written between parenthesis (e.g., (v) for *Validate* button). Section 5 provides a keymap containing the full list of the GUI shortcuts.

The *Annotated line thickness* shown under the four buttons in FIGURE 8, is a text entry that allows the user to specify the thickness of the annotated line. We note that such a thickness is not an annotation information and serves only a visualization purpose. This entry has built-in features to handle non-conforming thickness values, such as texts or negative numbers. The last element of the *annotation block* is the *Current annotation*, which displays the position and tilt of the annotated line. When no annotation line exists for the current frame, the values displayed will be replaced by three interrogation marks ???.

3.4 Browse Block

After loading a video file, drawing a line, and validating it using the *Validate Button* or its corresponding shortcuts, the user will browse to the next frames. The *Browse block* and its features ensure easy browsing through the video frames. As we show in FIGURE 9, the two buttons << and >> browse to the previous and next frames, respectively. Easier browsing alternatives are listed in Section 5. Right under the two browsing buttons, the interface shows the index of the current frame and the total number of frames. The browsing offset defaults to 1. This means that the index of the next (or previous) frame is equal to the index of the current frame + 1 (or - 1). Browsing with an offset of 1 slows down the browsing process when the video file has a large number of frames. Thus, the user can quickly go to any video frame by changing the browsing offset in the *Enter a browsing offset* entry.

4. ANNOTATION FILES

The annotation files take the *.npy* format, which can be loaded on Python using the *Numpy* package [52]. Each *.npy* file will contain N rows, where N is the number of frames in the corresponding video file. We note that even if the user does not annotate all the frames, the number of rows in

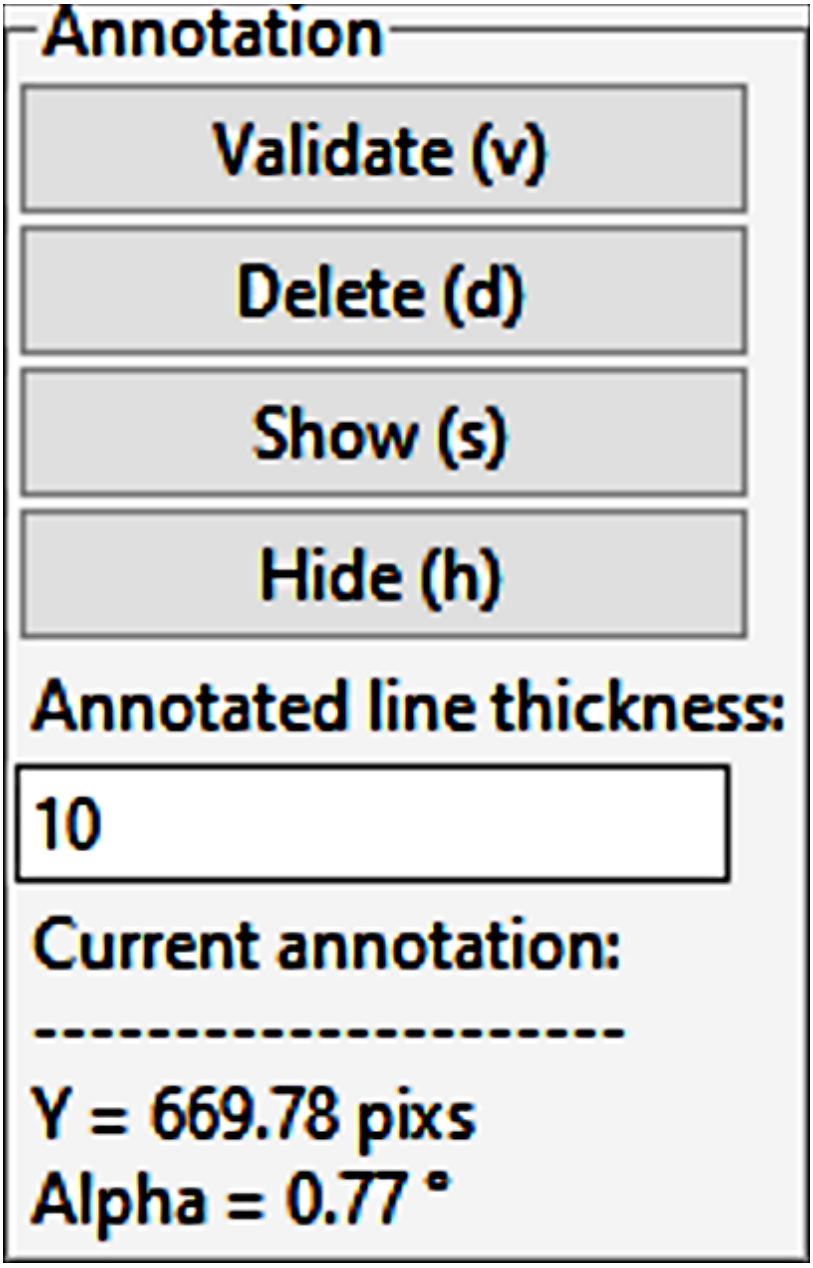


Figure 8: The *Annotation* block

the saved *.npy* will always be N. Each row contains five columns, and each column is a scalar number representing the following parameters: Y, ϕ, x_s, x_e, y_s, y_e, where Y is the line position in pixels, ϕ is the line tilt in degrees, (x_s, y_s) are Cartesian coordinates of the starting point, and (x_e, y_e) are Cartesian coordinates of the ending point. We visually illustrate these five parameters in FIGURE 10. We note that ϕ is zero when the line is horizontal and increases in the anti-clockwise rotation; for instance, the line in FIGURE 10, has a positive tilt value ($\phi > 0$). We note that non-

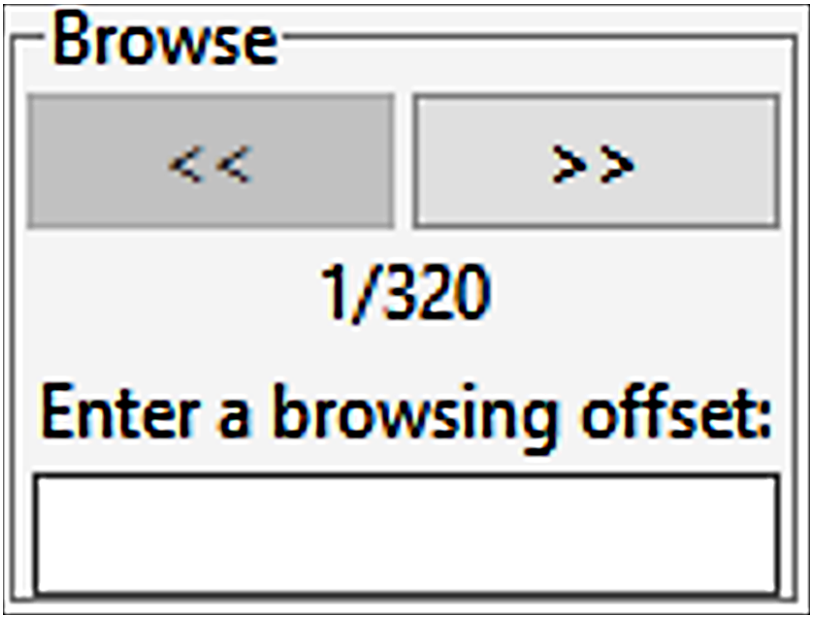


Figure 9: The *Browse* block

annotated frames will correspond to rows where scalars Y, ϕ, x_s, x_e, y_s, and y_e will be *np.nan*, which is the *Numpy* object for *Not A Number*.

5. KEYMAP FOR QUICK AND EASY ANNOTATION

This Section lists all the shortcut keys we chose to quickly and easily annotate a video file. These keys allow the user to annotate an entire video file with minimal hand movement. For instance, the keymap in TABLE 1, indicates that the user can annotate the entire video file using the mouse only: draw a line with the left mouse button, validate that line with the right mouse button, and browse to the next frame by rotating the mouse wheel instead of clicking on the next button (>>) shown in FIGURE 9. Another significant productivity feature in our software is the annotation replication, which can be done using the *w* key. For instance, let's assume that the camera does not move for a given period, which is very likely to happen for cameras on non-moving platforms such as shore pylons. The SMD [2], contains more than 17,222 video frames captured with such a camera set-up. If the period we just mentioned lasts, for instance, only five seconds, there would be 150 frames[2] where the horizon's position and orientation is the same (because the camera's position and orientation is the same). In other words, all 150 frames we mentioned will have the same annotation. In such a case, the user should browse to the 150-th non-annotated frame, annotate it, and then click on the *w* key to replicate that annotation on all the previous 149 non-annotated frames. This is much easier and faster than drawing and validating a line for each of the 150 non-annotated frames. TABLE 1 shows the complete list of keys facilitating the annotation process.

[2] 5× Frames per seconds = 5 × 30

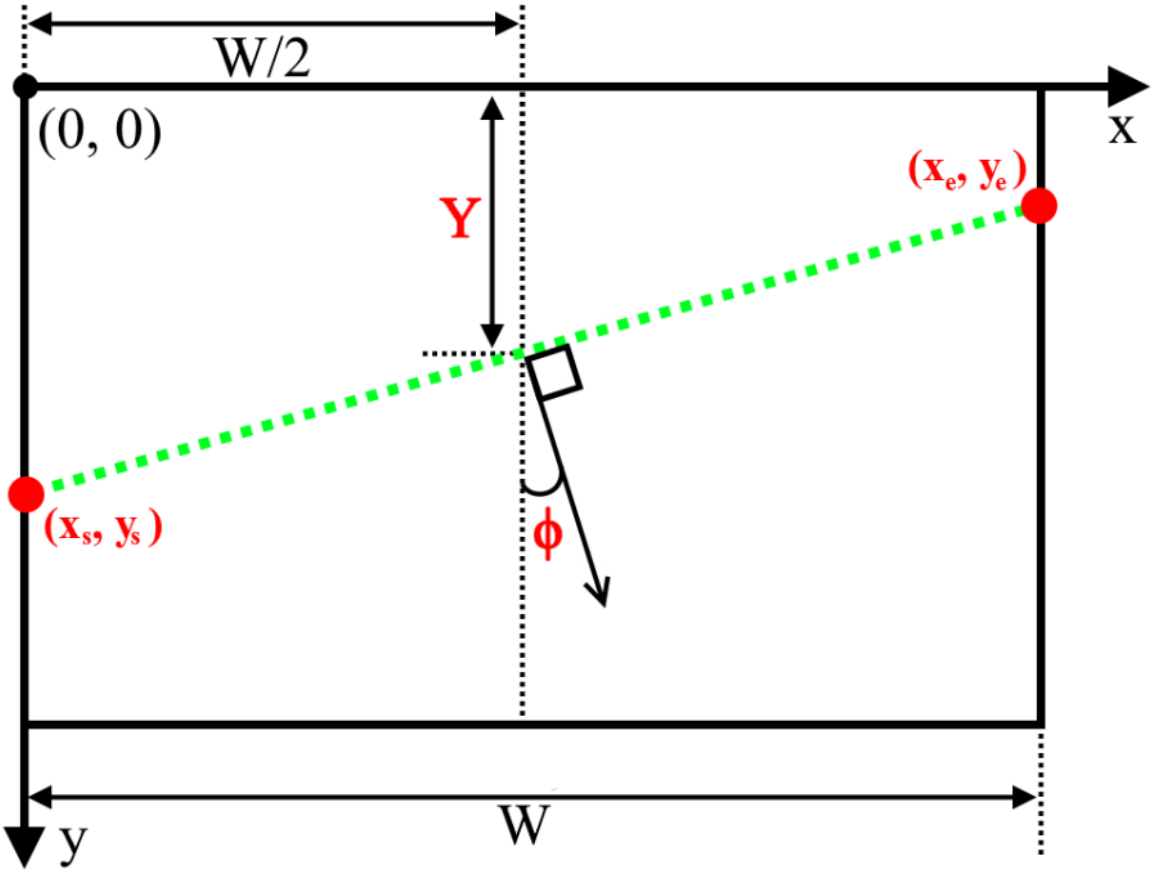


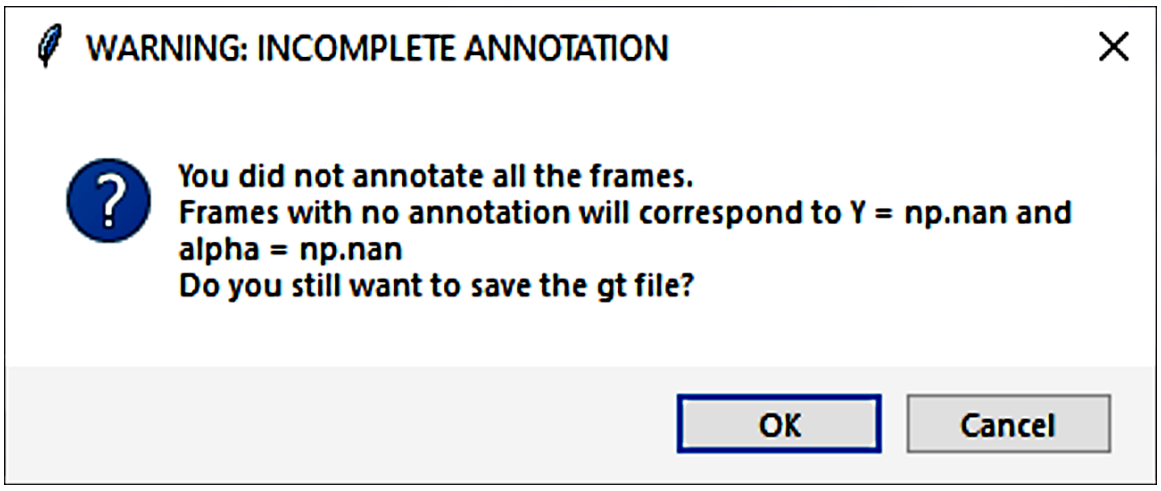
Figure 10: The line parameters saved in the annotation files

Table 1: Keymap of the GUI

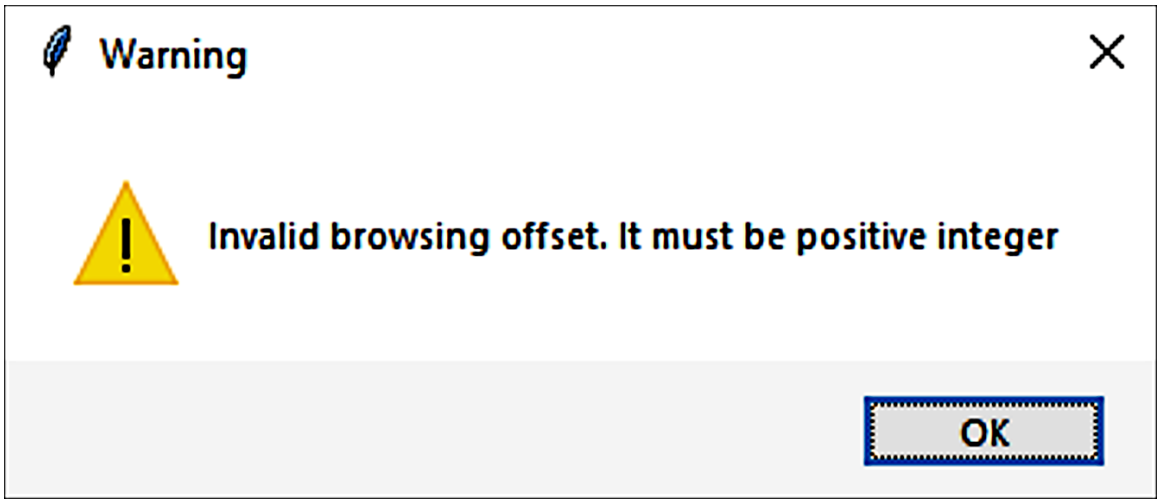
Keys	Actions
Mouse wheel (roll-up)	Browse to the next frame
Right arrow key (→)	Browse to the next frame
Mouse wheel (roll-down)	Browse to the previous frame
Left arrow key (←)	Browse to the previous frame
v	Validate the drawn line
Left mouse button	Validate the drawn line
w	Replicate the annotation of the current frame on all previous *non-annotated* frames
s	Show the annotated line (if it exists)
h	Hide the annotated line (if it exists)
d	Delete the annotated line
Enter key	Validates entered texts (e.g., *Annotated line thickness* in Figure 8)

6. MESSAGE BOXES

The GUI includes several message boxes to guide the user or warn him with relevant information. FIGURE 11(a) shows message box examples that would pop-up when the user tries to save an annotation file with missing annotations or enters an invalid browsing offset.



(a)



(b)

Figure 11: Two instances of message boxes

7. CONCLUSION

We proposed in this paper a horizon line annotation software and justified its importance in several aspects, which include the lack of current public datasets in terms of more maritime conditions as well as the mistakes in their annotation files, the preference of researchers to collect an annotate their

own maritime images, and the absence of a free public software dedicated to the horizon annotation task. We discussed and described all blocks of the graphical user interface. More importantly, we provided a complete keymap list that corresponds to several productivity features we designed to quickly and easily annotate the video frames. We provided the software as one file that would install all required files, including the executable application *Horizon Annotator.exe*.

References

[1] Zardoua Y, Astito A, Boulaala M. A Survey on Horizon Detection Algorithms for Maritime Video Surveillance: Advances and Future Techniques. Vis Comput. ISSN: 1432-2315. 2023;39:197-217.

[2] Prasad DK, Rajan D, Rachmawati L, Rajabally E, Quek C, et al. Video Processing From Electro-Optical Sensors for Object Detection and Tracking in a Maritime Environment: A Survey. IEEE Trans Intell Transp Syst. 2017;18:1993-2016.

[3] Ganbold U, Sato J, Akashi T. Coarse-To-Fine Evolutionary Method for Fast Horizon Detection in Maritime Images. IEICE Trans Inf Syst. 2021;2226-2236.

[4] Simarro G, Calvete D, Souto P. Ucalib: Cameras Autocalibration on Coastal Video Monitoring Systems. Remote Sens. 2021;13:2795.

[5] Shin B-S, Mou X, Mou W, Wang H. Vision-Based Navigation of an Unmanned Surface Vehicle With Object Detection and Tracking Abilities. Mach Vis Appl. 2018;29:95-112.

[6] Assadzadeh A, Arashpour M, Bab-Hadiashar A, Ngo T, Li H, et al. Automatic Far-Field Camera Calibration for Construction Scene Analysis. Computer-Aided Civ Infrastruct Engi. 2021;36:1073-1090.

[7] Zhang W, Naik SM. Camera Auto-Calibration by Horizon Estimation. Filed: Feb 6, 2009 and Date of Patent: Sep 4, 2012. US Patent 8,259,174.

[8] Ali A, Smrz P. Camera Auto-Calibration for Complex Scenes. In Thirteenth International Conference on Machine Vision. 2021;11605:544-554.

[9] Zardoua Y, Astito A, Boulaala M, Dokkali Y. A Short Overview of Horizon Detection Methods Applied to Maritime Video Stabilization. In advanced intelligent systems for Sustainable development (AI2SD'2020) Kacprzyk J, Balas VE, Ezziyyani M, editors. Cham: Springer International Publishing; 2022: 857-864.

[10] Cai C, Weng X, Zhu Q. Sea-Skyline-Based Image Stabilization of a Buoy-Mounted Catadioptric Omnidirectional Vision System. EURASIP J Image Video Process. 2018;2018:1.

[11] Fefilatyev S, Goldgof D, Shreve M, Lembke C. Detection and Tracking of Ships in Open Sea With Rapidly Moving Buoymounted Camera System. Ocean Eng. 2012;54:1-12.

[12] Ma Y, Wang H. Video Stabilization for Sea Scenes via Low-Rank Alignment and Effective Visual Cues. In 2021 China Automation Congress (CAC). 2021:5852-5857.

[13] Schwendeman M, Thomson J. A Horizon-Tracking Method for Shipboard Video Stabilization and Rectification. Journal of Atmospheric and Oceanic Technology. 2015;32:164-176.

[14] Liu YP, Wang RF, Xu HX, Sun DC. Shipborne Camera Video Stabilization Base on Horizon. Appl. Mech. Mater. 2014;543:2770-2774.

[15] https://apps.dtic.mil/sti/tr/pdf/ADA308861.pdf

[16] Duric Z, Rosenfeld A. Image Sequence Stabilization in Real Time. Real Time Imaging. 1996;2:271-284.

[17] Reichert GMW, Pieras M, Marroquim R, Vilanova A. Stabilization and Visual Analysis of Video-Recorded Sailing Sessions. Vis Comput Ind Biomed Art. 2021;4:26.

[18] Pires N, Guinet J, Dusch E. Asv: An Innovative Automatic System for Maritime Surveillance. Navigation. 2010;58:1-20.

[19] Qiao D, Liu G, Li W, Lyu T, Zhang J, et al. Automated Full Scene Parsing for Marine Asvs Using Monocular Vision. J Intell Robot Syst. 2022;104:37.

[20] Kong X, Liu L, Qian Y, Cui M. Automatic Detection of Seasky Horizon Line and Small Targets in Maritime Infrared Imagery. Infrared Phys Technol. 2016;76:185-199.

[21] Bai Y, Lei S, Liu L. The Ship Target Detection Based on Sea-Sky-Line. In 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE). 2021:456-460.

[22] Lee JM, Lee KH, Nam B, Wu Y. Study on Image-Based Ship Detection for AR Navigation. In 2016 6th International Conference on IT Convergence and Security (ICITCS). 2016:1-4.

[23] Shan X, Zhao D, Pan M, Wang D, Zhao L, et al. Sea–Sky Line and Its Nearby Ships Detection Based on the Motion Attitude of Visible Light Sensors. Sensors (Basel). 2019;19.4004.

[24] Bian W, Zhu Q. The Ship Target Detection Based on Panoramic Images. In 2015 IEEE International Conference on Mechatronics and Automation (ICMA). 2015:2397-2401.

[25] Kocak G, Yamamoto S, Hashimoto T. Analyzing Influence of Ship Movements on Stereo Camera System Set-up on Board Ship. Mar Eng. 2012;47:888-895.

[26] Zhang Y, Li Q-Z, Zang F-N. Ship Detection for Visual Maritime Surveillance From Nonstationary Platforms. Ocean Eng. 2017;141:53-63.

[27] Wang L, Fan S, Liu Y, Li Y, Fei C, et al. A Review of Methods for Ship Detection With Electro-Optical Images in Marine Environments. J. Mar. Sci. Eng. 2021;9:1408.

[28] Lin C, Chen W, Zhou H. Multi-Visual Feature Saliency Detection for Sea-Surface Targets Through Improved Sea-Sky-Line Detection. J Mar Sci Eng. 2020;8.799.

[29] Chen Z, Li B, Tian LF, Chao D. Automatic Detection and Tracking of Ship Based on Mean Shift in Corrected Video Sequences. In 2017 2nd International Conference on Image, Vision and Computing (ICIVC). 2017;449-453.

[30] Ma D, Dong L, Xu W. A Method for Infrared Sea-Sky Condition Judgment and Search System: Robust Target Detection via Pls and Cedog. IEEE Access. 2021;9:1439-1453.

[31] Ying-dong H, Jie L, Ning-jun F. An Algorithm for Locating Naval Ships on the Sea-Sky-Line. Transactions of the Beijing Institute of Technology. 2008;302-305.

[32] Kong W, Hu T, Deep A. Neural Network Method for Detection and Tracking Ship for Unmanned Surface Vehicle. In 2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS). 2019:1279-1283.

[33] Zhang Y, Liu H, Su A, Gui Y, Shang Y. Real-Time Estimation of Ship's Horizontal Attitude Based on Horizon Tracking. Optik. 2015;126:4475-4483.

[34] Shi J, Jin J, Zhang J. Object Detection Based on Saliency and Sea-Sky Line for Usv Vision. In 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC). 2018;35:1581-1586.

[35] Liu J, Li H, Liu J, Xie S, Luo J, et al. Real-Time Monocular Obstacle Detection Based on Horizon Line and Saliency Estimation for Unmanned Surface Vehicles. Mob Netw Appl. 2021;26:1372-1385.

[36] Wang H, Wei Z, Wang S, Ow CS, Ho KT, et al. A Vision-Based Obstacle Detection System for Unmanned Surface Vehicle. In 2011 IEEE 5th International Conference on Robotics. Automation and Mechatronics (RAM). 2011:364-369.

[37] Brejcha J, Čadík M. State-Of-The-Art in Visual Geo-Localization. Pattern Anal Appl. 2017;20:613-637.

[38] Praczyk T. Detection of Land in Marine Images. Int J Comp Intell Syst. 2018;12:273-281.

[39] Gladstone R., Moshe Y., Barel A., Shenhav E. Distance Estimation for Marine Vehicles Using a Monocular Video Camera. In 2016 24th European Signal Processing Conference (EUSIPCO). 2016:2405–2409.

[40] Hashmani MA, Umair M. Sea–Sky Line Detection Using Gray Variation Differences in the Time Domain for Unmanned Surface Vehicles. J Mar Sci Eng. 2022;10:193.

[41] https://superannotate.com/

[42] Fu J, Li F, Zhao J. Real-Time Infrared Horizon Detection in Maritime and Land Environments Based on Hyper-Laplace Filter and Convolutional Neural Network. IEEE Trans Instrum Meas. 2023;72:1-13.

[43] https://github.com/wkentaro/labelme

[44] Li F, Zhang J, Sun W, Jin J, Li L, et al. Sea–Sky Line Detection Using Gray Variation Differences in the Time Domain for Unmanned Surface Vehicles. Signal Image Video Process. 2021;15:139-146.

[45] Zardoua Y, Abdelali A, Boulaala M. A Horizon Detection Algorithm for Maritime Surveillance.2021. ArXiv preprint: https://arxiv.org/vc/arxiv/papers/2110/2110.13694v1.pdf

[46] https://rectlabel.com/

[47] https://www.cvat. ai/

[48] https://supervisely.com/

[49] Jeong C, Yang HS, Moon K. A Novel Approach for Detecting the Horizon Using a Convolutional Neural Network and Multiscale Edge Detection. Multidimensional Syst Signal Process. 2019;30:1187-1204.

[50] Prasad DK, Rajan D, Rachmawati L, Rajabally E, Quek C, et al. Muscowert: Multi-Scale Consistence of Weighted Edge Radon Transform for Horizon Detection in Maritime Images. J Opt Soc Am A Opt Image Sci Vis. 2016;33:2491-2500.

[51] Liang D, Liang Y. Horizon Detection From Electro-Optical Sensors Under Maritime Environment. Ieee Trans Instrum Meas. IEEE transactions on Instrumentation and measurement. 2019;69:45-53.

[52] Harris CR, Millman KJ, Van Der Walt SJ, Gommers R, Virtanen P, et al. Array Programming With Numpy. Nature. 2020;585:357-362.