Sandru, Andrei; Kujala, Pentti; Visala, Arto

Horizon detection and tracking in sea-ice conditions using machine vision

Published in:
IFAC-PapersOnLine

DOI:
10.1016/j.ifacol.2023.10.377

Published: 01/07/2023

Document Version
Publisher's PDF, also known as Version of record

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 6724–6730

Horizon detection and tracking in sea-ice conditions using machine vision

Andrei Sandru * **Pentti Kujala** ** **Arto Visala** *

* *Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland (e-mail: firstname.lastname@aalto.fi).*
** *Department of Mechanical Engineering, Aalto University, Espoo, Finland (e-mail: firstname.lastname@aalto.fi)*

Abstract: An automated process is proposed for horizon detection and tracking using machine vision cameras and in polar, sea-ice conditions. These conditions present unique challenges for machine vision applications, such as a large amount of clutter (e.g. icebergs) and secondary edge lines from broken ice pieces. The process is divided in two parts: a more computationally expensive, yet robust detection algorithm in the first stage, based on Convolutional Neural Networks, and used to detect the horizon line in an arbitrary sea-ice image; followed by a tracking algorithm, responsible of efficiently detecting the horizon line in the subsequent images of a sequence. We propose two tracking algorithms, one based on the traditional Canny and Hough line detection methods; and a second novel approach using entropy as a measure of randomness, to segment between sea-ice and sky. Our automated process was compared to manually obtained ground-truth data and the results indicate good agreement, especially for the texture-based tracking algorithm.

Keywords: machine vision, sea ice, horizon, detection, tracking, CNN, Hough, entropy

1. INTRODUCTION

In recent years, marine operations have experienced a boost (see e.g. Nations (2019)), and since the climate change has affected the severity of the ice conditions (e.g. concentration, thickness), new routes and operation areas in polar regions are being explored (Bekkers et al. (2018)). However, the navigation or operation in ice-infested waters requires a high degree of experience and expertise, and reinforced platforms and instrumentation, which are nevertheless susceptible to failure. From the perspective of a ship navigating in icy waters, any additional information that may increase or maintain its situational awareness is highly valuable, even more so when such data can be obtained and processed locally onboard, without the need of an expensive, and often unavailable network connection. One such addition, not limited to vessels only, can be a machine vision camera oriented such that, among other salient features, it captures the horizon line.

In maritime context, the horizon is generally regarded as the line (at infinity) separating the sky and the body of water underneath, commonly a sea or ocean. The estimation of the horizon line within an image can provide highly valuable information, that can be used for example in image stabilization for data captured on dynamic platforms (e.g. vessels, buoys) (Schwendeman and Thomson (2015), Morris et al. (2007), Fefilatyev et al. (2012)), camera auto-calibration (Zhang and Naik (2012)), camera (and platform) attitude estimation for roll and pitch (Cornall et al. (2006), Grelsson et al. (2016)), camera to Inertial Measurement Unit (IMU) extrinsics calibration (Bovcon et al. (2018)), or even the detection of dangerous or abnormal situations (e.g. extreme roll and pitch angles for a vessel or buoy). It is worth noting that images collected from maritime applications and in icy waters present a non-uniformity and arbitrariness in their content, which hinders the task of extracting useful information (e.g. horizon detection). For instance, a pack-ice field can present a large amount of straight edges around the ice floes. Another challenge comes from large icebergs overlapping the horizon and creating a sharp, straight line above it. Another important topic is the consideration of real-time processing. Given the usually slow dynamics involved in marine applications, real-time constraints can be more relaxed than, for example, automotive applications. Nevertheless, this needs to be taken into account and a strive towards efficiently meeting such real-time requirements should be the objective.

In the current literature, there has been multiple attempts at estimating the horizon line given an arbitrary image (see Zardoua et al. (2021)), yet only one that we are aware of studies the particular case of horizon tracking (Schwendeman and Thomson (2015)). In our work, at first we study the applicability of a simple Convolutional Neural Network (CNN) to detect the horizon line from images captured in remote polar waters, and then we propose two methods, one of which completely novel, for tracking a previously detected horizon line in icy waters. To the best of our knowledge, this type of scenarios have not been studied before in horizon detection and tracking, and they present a particular set of challenges. For instance, the broken ice field creates a large number of secondary lines, especially close to the horizon and which can easily confuse an algorithm. These floating ice pieces (i.e. ice floes or simply floes) can at times present sharp

10.1016/j.ifacol.2023.10.377

and completely straight edges bound to confuse an edge detecting algorithm. Another challenge may come from the presence of large icebergs with a flat top surface, creating a perfectly straight division line with the sky.

In their work, Jeong et al. (2019) propose the use and prove the effectiveness of a CNN-based approach for horizon detection. Continuing on the same topic, we propose to use a different CNN architecture, much simpler and which is run on the whole image using a GPU, rather than only on the detected edges. This would increase the detection rate of the horizon, given sufficient examples, in such situations where the horizon line appears diffuse. Additionally, we propose to use as a line fitting algorithm the robust estimator described in Torr and Zisserman (2000) and termed MLESAC. Then, we continue with two methods to efficiently track the horizon line given an initial estimate. Similar to Schwendeman and Thomson (2015), the first method for horizon tracking is an edge-based approach using Canny (Canny (1986)) and the Hough transform (Duda and Hart (1972)); where we introduce the novelty of a two-stage Region Of Interest (ROI) approach, that both increases the computational efficiency and decreases the estimation error. Then, in the second method we introduce a texture based approach for horizon tracking that has not been used before and presents promising results.

The present work is structured as follows. In the subsequent Section 2, we describe the methodology required to analyse the imagery data coming from a machine vision camera, and obtain the results presented in Section 3. Later, those results are discussed in Section 4, followed by a conclusion in Section 5.

2. METHODS

The main principle of the proposed process for performing horizon detection and tracking is to use a computationally expensive, yet robust algorithm to initially estimate the horizon location; then, using a computationally efficient algorithm, the horizon line is tracked in subsequent images. Occasionally, the more expensive method can be run in parallel, to check the accuracy of the tracked horizon line and re-initialize the tracking algorithm when necessary. In the following subsections, each part of the process and their related methods are described. For all the methods described below, the horizon line is characterized following the Standard Hough Transform (SHT) representation of a line:

$$\rho = x \cdot cos(\theta) + y \cdot sin(\theta) \tag{1}$$

where ρ is the perpendicular distance (in pixels) between the line and the origin (located at the upper-left corner of the image), and θ is the angle (in radians) between the aforementioned ρ line and the x-axis (which is parallel to the top, horizontal line of the image).

2.1 Image pre-processing

Images collected using machine vision cameras typically present a series of artifacts and distortions, which are (usually) not present in consumer cameras. Such distortions include, among others, a vignetting effect (pixels in an image tend to appear darker the further they are from an image's optical centre), and lens imperfections which geometrically distort the image. Before proceeding with the analysis to extract the horizon line, images collected with a machine vision camera need to undergo a series of operations, in which image artifacts are removed and then are geometrically rectified, such that straight lines (e.g. the horizon) appear straight. To remove the vignetting effect, the captured image is divided by a mask that models the camera's vignetting effect, followed by a normalization operation. Lens distortions are inherent to each individual camera setup, and affect the way a camera maps the 3D world into a 2D image. The model used for such mapping can be estimated and compensated for errors up to a degree. This estimation is referred to as camera calibration and which, among other parameters such as radial and tangential distortions, provides the camera's intrinsic matrix K, as defined by Corke (2011). Aforementioned estimation was done by means of the Matlab® Calibrator App as described in Matlab (2022b).

2.2 Horizon detection

This section describes two methods, one manual and one automatic, to detect and describe the horizon line in an image which also contains icy waters.

Manual horizon identification. The ground truth data, both for horizon tracking initialization and results validation, is obtained manually by selecting two points belonging to the horizon line, and which are as far apart as possible to minimize location error. Then, through equations in (2 - 4) a program computes the horizon line parameters in terms of the Hough line representation described earlier. First, using the pixel coordinates of the two selected points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, the slope of the line m is calculated in (2), which is then used in (3) to calculate θ and in (4) to calculate ρ.

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{2}$$

$$\theta = \frac{\pi}{2} + atan(m) \tag{3}$$

$$\rho = \frac{|y_1 - m \cdot x_1|}{\sqrt{m^2 + 1}} \tag{4}$$

Automatic horizon detection using CNN. By using a Convolutional Neural Network (CNN), it is possible to segment all the pixels in an image as either part of the horizon or not. This straightforward approach would infer an approximate location of the horizon line, albeit other pixels can also be incorrectly classified as *horizon*. To overcome this a line fitting algorithm, robust against outliers, is required. The proposed process for automatically detecting the horizon line from an arbitrary image and using a CNN, is described next.

Since a pixel-wise segmentation of an image is desired, the two classes are divided as follows: a training patch is labelled as *horizon* iff the horizon line strictly passes through its centre; while any other patch would be considered as *non-horizon*, including those where the horizon is visible,

Layer	Type	Properties
1	Image input	64×64×3 images with 'zerocenter' normalization
2	Convolution	64 9×9 convolutions with stride [1 1] and padding [2 2 2 2]
3	Convolution	64 9×9 convolutions with stride [1 1] and padding [2 2 2 2]
4	Convolution	64 9×9 convolutions with stride [1 1] and padding [2 2 2 2]
5	ReLU	
6	Max Pooling	5×5 max pooling with stride [2 2] and padding [0 0 0 0]
7	Convolution	64 9×9 convolutions with stride [1 1] and padding [2 2 2 2]
8	ReLU	
9	Max Pooling	5×5 max pooling with stride [2 2] and padding [0 0 0 0]
10	Convolution	128 9×9 convolutions with stride [1 1] and padding [2 2 2 2]
11	ReLU	
12	Max Pooling	3×3 max pooling with stride [2 2] and padding [0 0 0 0]
13	Fully Connected	128 fully connected layer
14	ReLU	
15	Fully Connected	2 fully connected layer
16	Softmax	
17	Classification Output	Cross-entropy loss function

Fig. 1. Proposed network architecture for horizon segmentation.

but does not cross the patch through its centre. Then, since the CNN needs to discern between only two classes, it does not need to be too complicated nor computationally expensive. We propose a network architecture based on Girshick (2015) and presented in Table 1, where the input layer consists of a three-dimensional matrix containing three patches of 64x64 pixels, one for each color channel of red, green and blue.

To train the CNN, a suitable data-set needs to be assembled. It would consist of only two classes: horizon and non-horizon, as introduced before. The ground truth data from the previous section can be used to this end, where a number of patches of a pre-defined size are extracted along the horizon line (passing through their centre). Then, the same amount of patches are extracted at random for the non-horizon case. The number of examples for each of the classes needs to be similar, as to not over-train the network with one of the classes, thus creating a bias towards the non-horizon class in the present case.

Lastly, as mentioned before the segmentation process may not be completely accurate, and to estimate correctly the horizon line, a line fitting algorithm with strong robustness against outliers is required. For this, we propose the use of the robust estimator described in Torr and Zisserman (2000) and termed MLESAC. This estimator is based on the well known Random Sample Consensus approach (RANSAC) for producing hypothetical solutions, however it chooses as best solution the one which maximizes a log likelihood estimate (including both inliers and outliers), instead of the one which maximizes the number of inliers (as in the traditional RANSAC algorithm).

2.3 Horizon tracking

Once the horizon line has been identified in the first of a sequence of images, the analysis of subsequent images may be focused towards tracking such line using other more efficient algorithms. The two algorithms described below use the Standard Hough Transform (SHT) to search for the horizon line within a binary image containing multiple edge line candidates. In each method, a different approach is taken to obtain the binary image.

Two-stage ROI, Hough transform analysis. Using as *a priori* information the location of the horizon line in the previous image, the dynamics dominating the platform on which the camera is attached to, as well as the period between captured frames; it is possible to substantially reduce the search-space for the horizon line in the current image.

In the first stage, a Region of Interest (ROI) is defined in the current image I_k using the detected horizon line in the previous image I_{k-1} with parameters θ_{k-1} and ρ_{k-1}. The ROI area is delimited between two new lines, with parameters θ_{k-1} and $\rho_{k-1} - \Delta\rho$ (top line) and θ_{k-1} and $\rho_{k-1} + \Delta\rho$ (bottom line); and the image's vertical sides. The value of $\Delta\rho$ is unique for each platform type and conditions that will influence the expected rate of change in the platform's attitude; and a sensible value can be obtained empirically from a short sequence of images, where setting too large values may lead to poor tracking performance, and too small values can prevent the algorithm from correctly tracking the horizon in highly dynamic situations. Then, an edge detection algorithm is run on the ROI patch to highlight all potential lines that might belong to the horizon line. The algorithm selected for edge detection is the well-known Canny edge detection, since it will look both for strong and weak edges. The result from the Canny edge detection process is a binary image containing information of a large number of edges detected in the image, and from which the true horizon line needs to be extracted.

In the second stage, the previously described standard Hough transform is used to extract the true horizon line. To reduce the computation time and direct the search, the computation of the Hough transform space is limited by $\Delta\theta_2$ and $\Delta\rho_2$. That is, the parameter space is computed between $\theta_{k-1} \pm \Delta\theta_2$ and $\rho_{k-1} \pm \Delta\rho_2$. Last, the strongest peak within the parameter space is selected as the horizon line and its parameters are defined as θ_{peak} and ρ_{peak}.

Texture based analysis. Generally, textured images that include a large amount of edges (i.e. lines) will difficult the task of finding the edges and lines of interest, in the present case, the horizon. However, such cases can also be used as an advantage, specifically when the secondary lines are located within a fast and randomly changing background (e.g. the area of the image right under the horizon line); and the principal lines divide such background with another that presents a smoother texture. In such context, the horizon will be the separation line between two, distinct regions of the image, namely the sky right above the horizon line, and the icy waters right below. The aforementioned two regions can be segmented by running a statistical analysis of the local variation in texture. This analysis can be done using the local Shannon entropy (Shannon (2001)), which is a statistical measure of randomness, present in this case in a patch or neighbourhood around a pixel.

Similarly to the previous method, at first a rectangular Region Of Interest is selected in the current image using the extreme, vertical pixel coordinates from the horizon line detected in the previous image, thus disregarding its inclination angle. Then, the entropy filter is used to process this part of the image, obtaining a grayscale matrix with higher values corresponding to higher randomness present around the current pixel. To obtain a binary image

(sky/sea), a clustering algorithm is used, namely the k-means algorithm (Arthur and Vassilvitskii (2006)) with only two classes. Next, the binary image is cleaned of sporadic *holes* within the icy water or high intensity variations detected in the sky using morphological operations (Peterlin (1996)), specifically an operation of closing the image (that is, a dilation operation followed by an erosion), and then an opening (an erosion operation followed by a dilation).

Then, the Canny edge detection process is run on the binary image, providing a substantially reduced number of line candidates. Lastly, the Hough transform as described previously is used to extract the best possible candidate line, corresponding to the horizon.

3. EXPERIMENTS AND RESULTS

The experimental setup used for capturing data comprises a standard laptop PC and a Basler acA4112-8gc machine vision camera (Basler (2022)). Among the most relevant features of the camera, it is worth mentioning its large resolution at 12.3 megapixels and a global shutter. The camera was set to record at 0.5Hz and the images were saved in raw format, and then converted to the standard RGB format. The image capturing system was mounted in the crow's nest of the S.A. Agulhas II, and data capture was performed during the vessel's resupply voyage to Antarctica in 2018-2019. The implementation of the methods and processing of data was performed in Matlab® environment and using functions from the *Image processing toolbox* Matlab (2022a). Data collection was performed first, and the processing was done offline at a later stage, using an Intel® Core i7-9750H CPU and an NVIDIA® Geforce RTX 2070 8Gb GPU (for training and applying the neural network to images).

The process of running the CNN-based method for horizon detection is presented in Fig. 2. The feature map from the softmax layer is obtained and overlayed on the input image, from which a region is shown in Fig. 2a. The feature map is an intensity image with values [0-1] displaying the certainty with which the network considers a pixel as part of the horizon. Then, Fig. 2b presents a binary view of the horizon as the result of the last, classification layer. The coordinates of all *horizon* considered pixels are then used to fit a line (Fig. 2c, as described in the methods section. Lastly, the resulting line and thus the detected horizon is presented in Fig. 2d.

Next, in Fig. 3a an example image is presented where a previously detected horizon line is highlighted as a dashed line. In addition, the bounding box around the previous horizon line is shown with two continuous lines, and where $\Delta\theta$ was set to 100 pixels. The Canny edge detection result of the ROI area (a small patch is presented in Fig. 3b), is the input for the Hough transform.

Then, Fig. 4 depicts the process of using the texture-based horizon tracking on a single image, given the horizon parameters from the previous image in the sequence. First, the input image is converted to grayscale (Fig. 4a), and then processed using the entropy filter with a neighbourhood of 9x9 pixels (Fig. 4b). The result of running the k-means segmentation with only two classes is shown in Fig.

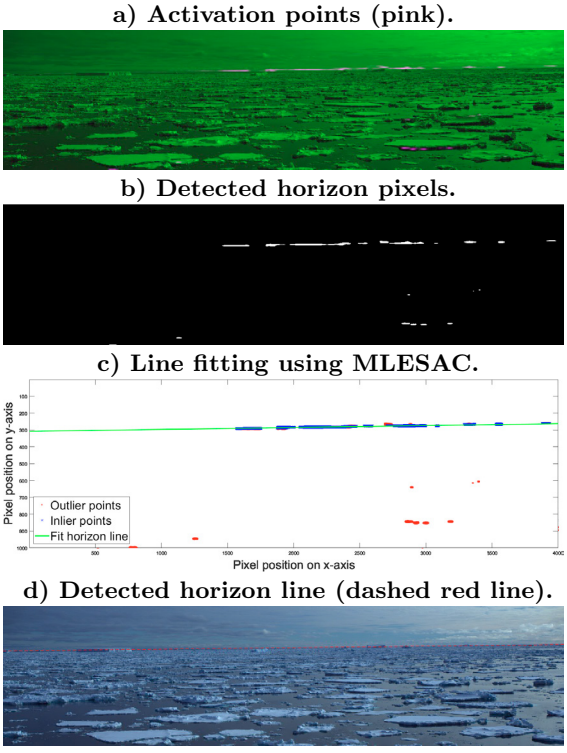


Fig. 2. Horizon detection using CNN.

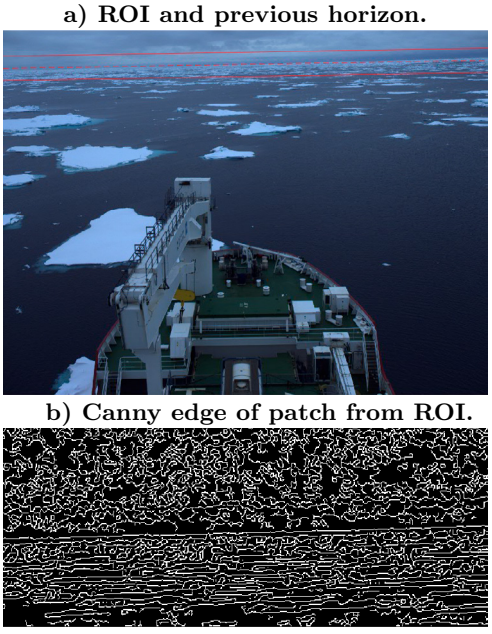


Fig. 3. The horizon line detected in a previous image is plot over the current image, as well as the Region of Interest (ROI) in the top figure (a). Then, the bottom figure (b) presents a small patch of the result of running Canny edge detector on the above ROI, highlighting the challenges of such environments.

4c, which is then *cleaned* using morphological operations (Fig. 4d). It is important to note that in Fig. 4e the discontinuous line is, in fact, uninterrupted but due to the detected edges being only one pixel wide using the Canny method, it is not displayed correctly. The resulting horizon

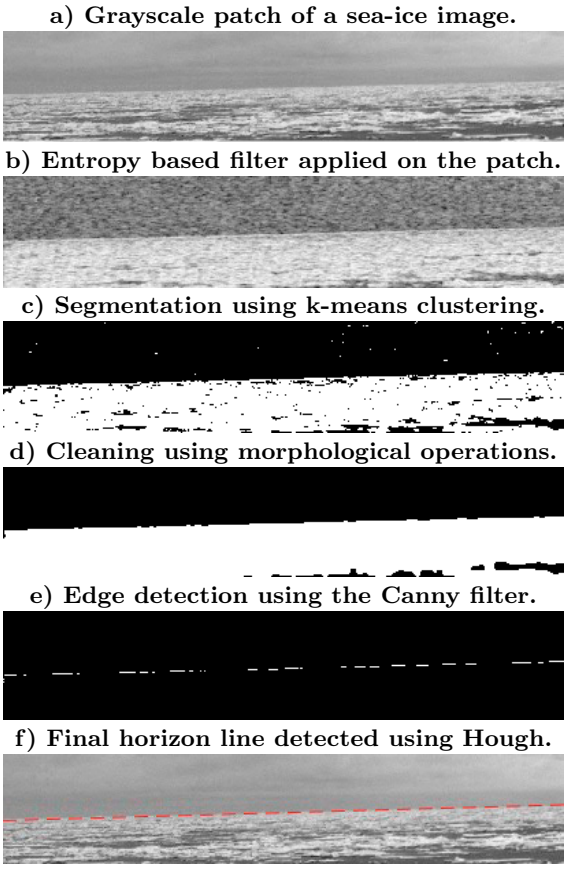


Fig. 4. Horizon detection using texture analysis.

line is presented in Fig. 4f as a dashed, red line overlaid on the ROI of the input image.

The processes described in the methods section were applied on a total of nine case studies, which span over several days and each comprising sequences of 100 images where the ground truth line for the horizon was manually identified. The first image from each sequence is presented in Fig. 5. In the particular case of the CNN-based method for horizon detection, the first 90 images in each of the selected sequences were used to train the network, totalling almost 50000 example patches for the horizon class, and as many for the non-horizon class. The training of the network took around three hours. Then, the same trained network was run on the remaining 10 images from each case.

Next, the detection (CNN) and tracking (Hough, texture) accuracy results are presented and compared in Tables 1 and 2. The error metric used is the well known Root Mean Square Error (RMSE), computed as the error between the parameters of the ground truth line and the estimation of the algorithm used. First, the RMSE (in radians) of the θ angle is introduced in Table 1, followed by the RMSE (in pixels) of the ρ distance in Table 2. For both tables, results with a superscript are further examined in Fig. 6 for the CNN-based results, in Fig. 7 for the Hough-based results, and lastly in Fig. 8 for the texture-based ones. Other results are considered within an acceptable error margin, and hence not displayed. Lastly, the average processing time per image for each of the methods and

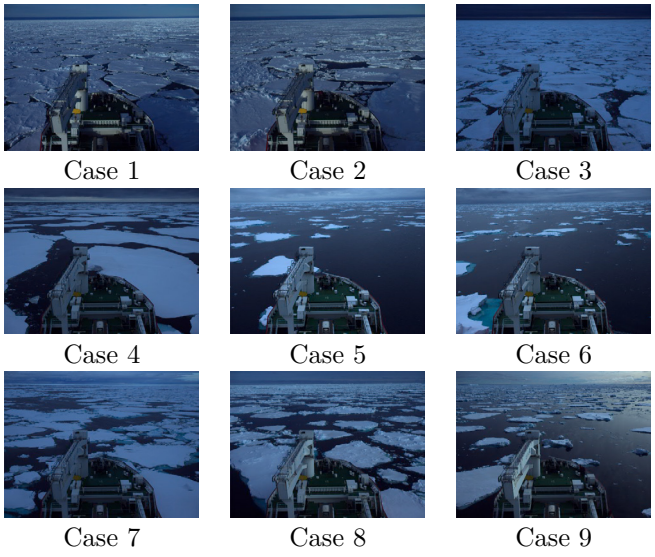


Fig. 5. Example images from each study case.

cases selected, are presented in Table 3. In parenthesis, the minimum and maximum times are given as well.

Table 1. Theta RMSE (radians) per case and method used for detection or tracking.

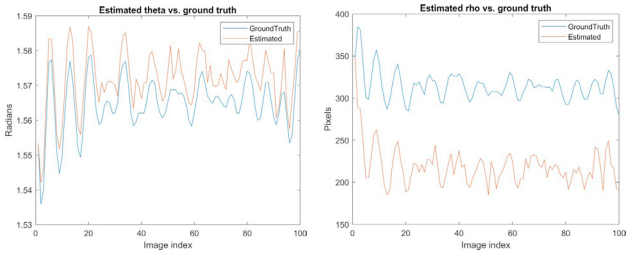
	CNN	Hough	Texture
Case 1	0.0036rad	0.0025rad	0.0024rad
Case 2	0.0075rad[1]	0.0010rad	0.0026rad
Case 3	0.0205rad[1]	0.0007rad	0.0044rad[3]
Case 4	0.0018rad	0.0010rad	0.0011rad
Case 5	0.0046rad	0.0012rad	0.0043rad
Case 6	0.0333rad[1]	0.0034rad	0.0017rad
Case 7	0.0112rad[1]	0.0024rad[2]	0.0009rad
Case 8	0.0022rad	0.0036rad	0.0008rad
Case 9	0.0046rad[1]	0.0086rad[2]	0.0015rad

Table 2. Rho RMSE (pixels) per case and method used for detection or tracking.

	CNN	Hough	Texture
Case 1	12.02px	5.55px	4.86px
Case 2	96.60px[1]	2.26px	4.86px
Case 3	168.57px[1]	3.52px	21.75px[3]
Case 4	4.77px	2.01px	2.37px
Case 5	15.17px	2.24px	10.56px
Case 6	147.77px[1]	6.79px	3.80px
Case 7	124.14px[1]	18.65px[2]	2.01px
Case 8	3.83px	26.07px	2.09px
Case 9	99.04px[1]	25.33px[2]	2.60px

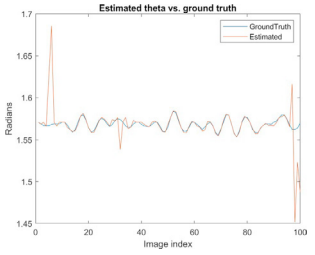
Table 3. Average processing times per image and method in seconds.

	CNN	Hough	Texture
Case 1	1.47s (1.39/1.65)	0.19s (0.17/0.21)	0.38s (0.28/0.51)
Case 2	1.44s (1.38/1.70)	0.20s (0.19/0.22)	0.35s (0.29/0.47)
Case 3	1.53s (1.39/2.16)	0.21s (0.20/0.23)	0.39s (0.32/0.56)
Case 4	1.52s (1.38/2.25)	0.24s (0.21/0.32)	0.64s (0.47/0.86)
Case 5	1.60s (1.42/3.57)	0.28s (0.24/0.33)	0.57s (0.41/0.75)
Case 6	1.52s (1.40/1.84)	0.16s (0.15/0.23)	0.49s (0.40/0.62)
Case 7	1.47s (1.41/1.66)	0.13s (0.11/0.15)	0.52s (0.42/0.67)
Case 8	1.48s (1.39/1.70)	0.17s (0.13/0.28)	0.42s (0.35/0.53)
Case 9	1.50s (1.41/1.92)	0.15s (0.13/0.21)	0.54s (0.42/0.67)

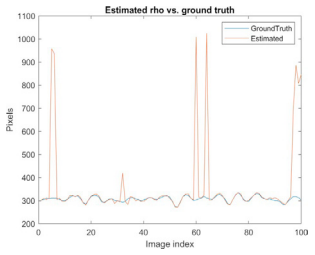


(a) Case 2 CNN θ

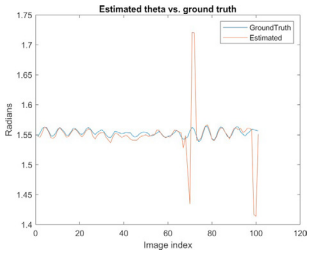
(b) Case 2 CNN ρ



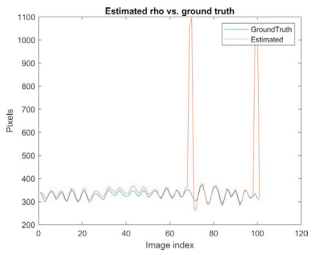
(c) Case 3 CNN θ



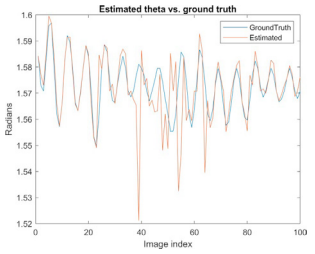
(d) Case 3 CNN ρ



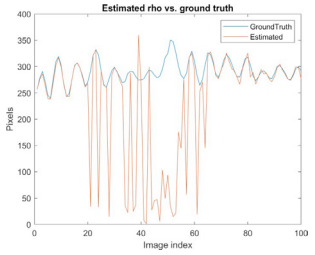
(e) Case 6 CNN θ



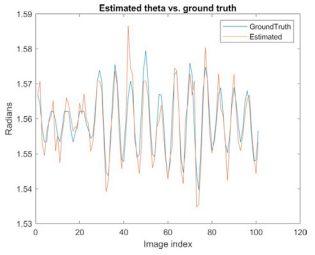
(f) Case 6 CNN ρ



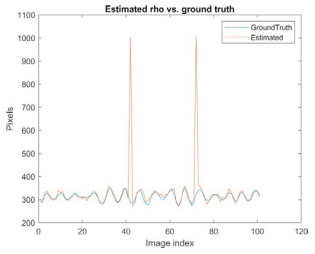
(g) Case 7 CNN θ



(h) Case 7 CNN ρ



(i) Case 9 CNN θ

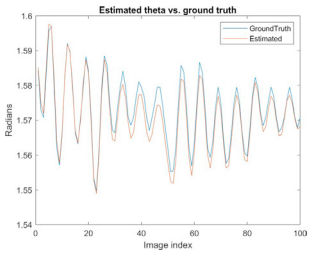


(j) Case 9 CNN ρ

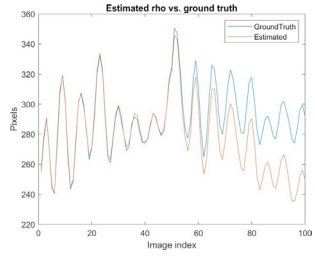
Fig. 6. Visual presentation of only the problematic results from the CNN-based horizon estimator against the manually obtained ground truth line parameters.

4. DISCUSSION

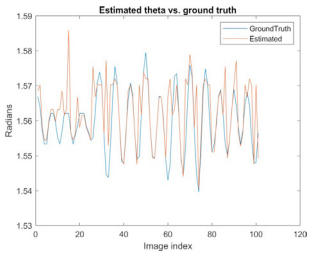
The proposed Convolutional Neural Network performed well in most cases, but it has become clear that a larger training data-set is required to obtain more accurate results. This can be seen from Fig. 2a-b, where only a small amount of pixels were correctly classified as horizon. Nevertheless, as expected the RANSAC based line fitting algorithm performed well and managed to find and disregard outliers. The cases which presented a large error are further analysed in Fig. 6, where sub-figures a-b and g-h present an error that we relate to an insufficiently trained network. However, in the cases of Fig. 6 c-f and i-j, the error was sporadic and not reproducible, and a closer inspection indicated good agreement between the estimates and ground truth. We have not managed to identify the culprit causing these errors, and further investigation is needed.



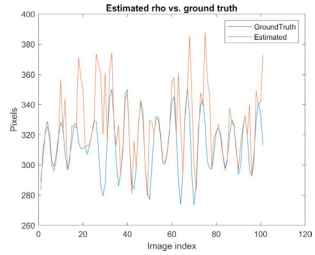
(a) Case 7 Hough θ



(b) Case 7 Hough ρ

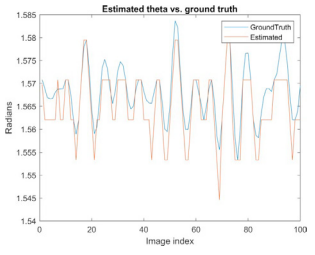


(c) Case 9 Hough θ

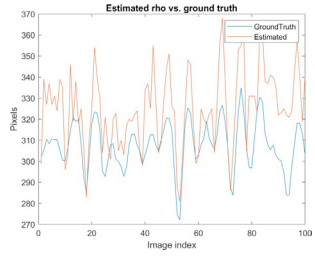


(d) Case 9 Hough ρ

Fig. 7. Visual presentation of only the problematic results from the Hough-based horizon estimator against the manually obtained ground truth line parameters.



(a) Case 3 Texture θ



(b) Case 3 Texture ρ

Fig. 8. Visual presentation of only the problematic results from the texture-based horizon estimator against the manually obtained ground truth line parameters.

Then, Fig. 3b confirms our initial affirmation regarding the challenges of a sea-ice environment: the broken ice field creates a large amount of secondary lines, many of which can be extended into a single and incorrect horizon line. Next, Fig. 7b presents the risk of accumulated error from an initial incorrect solution, a drift that purely tracking algorithms may encounter if not paired with *control points*. On the other hand, in Fig. 7c-d it can be observed how the algorithm can recover itself from incorrectly estimated horizon lines, due to a high clutter of edges

All in all, the proposed texture-based algorithm has managed to achieve the lowest overall error score as can be seen from Tables 1 and 2, although at the expense of a slightly higher computational cost when compared to the Hough-based algorithm. The slightly higher error in the study case number 5 was due to a small number of images where the horizon line was incorrectly detected, although the algorithm quickly recovered in all cases. Then, the error depicted in Fig. 8 is due to an insufficient amount of texture contrast close to the horizon line, mainly due to poor visibility conditions and darkness.

Lastly, Table 3 introduces the average times each algorithm spent per image, as well as the minimum and maximum times present in that run sequence. The processing times for the CNN-based method was one order of magnitude higher when compared to the tracking algorithms. Overall, all those processing times offer only an estimate, as the processing platform itself (i.e. computer) and its operating system can greatly influence the actual values.

5. CONCLUSION

We proposed an automated process for horizon detection and tracking in polar sea regions, which present unique challenges for machine vision applications. With the proposed methods, we are able to estimate the location of the horizon line in an arbitrary image containing sea-ice and other clutter elements, such as icebergs. Data was collected onboard the ice-breaker S.A. Agulhas II during its resupply voyage to Antarctica in 2018-2019. Our automated process was compared to manually obtained ground-truth data and the results indicate good agreement, especially in the texture-based tracking algorithm.

In the future, more development should be done to increase the accuracy and precision of the Convolutional Neural Network based method for horizon detection, which may include a larger training data-set and a refinement of the network's architecture. Then, new cases with a larger variety of weather conditions should be introduced, for a more comprehensive case study. Such additional cases may include for example fog, rain and snow; all of which may heavily impact a camera's operation.

The initial results show that our automated horizon detection and tracking methods have great potential in providing valuable information for navigating a ship in ice infested waters and for any other platform located in such remote areas.

REFERENCES

Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Technical report, Stanford.

Basler (2022). aca4112-8gc - basler ace gige. URL `https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca4112-8gc/`. [Accessed: 2022-10-30].

Bekkers, E., Francois, J.F., and Rojas-Romagosa, H. (2018). Melting ice caps and the economic impact of opening the northern sea route. *The Economic Journal*, 128(610), 1095–1127.

Bovcon, B., Perš, J., Kristan, M., et al. (2018). Stereo obstacle detection for unmanned surface vehicles by imu-assisted semantic segmentation. *Robotics and Autonomous Systems*, 104, 1–13.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6), 679–698.

Corke, P. (2011). *Robotics, Vision and Control: Fundamental Algorithms In MATLAB®*, volume 73. Springer.

Cornall, T.D., Egan, G.K., and Price, A. (2006). Aircraft attitude estimation from horizon video. *Electronics Letters*, 42(13), 744–745.

Duda, R.O. and Hart, P.E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1), 11–15.

Fefilatyev, S., Goldgof, D., Shreve, M., and Lembke, C. (2012). Detection and tracking of ships in open sea with rapidly moving buoy-mounted camera system. *Ocean Engineering*, 54, 1–12.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 1440–1448.

Grelsson, B., Felsberg, M., and Isaksson, F. (2016). Highly accurate attitude estimation via horizon detection. *Journal of Field Robotics*, 33(7), 967–993.

Jeong, C., Yang, H.S., and Moon, K. (2019). A novel approach for detecting the horizon using a convolutional neural network and multi-scale edge detection. *Multidimensional Systems and Signal Processing*, 30(3), 1187–1204.

Matlab (2022a). *Image Processing Toolbox*. Matlab. URL `https://se.mathworks.com/products/image.html`. [Accessed: 2022-10-30].

Matlab (2022b). *Single camera calibrator app*. Matlab. URL `https://se.mathworks.com/help/vision/ug/single-camera-calibrator-app.html`. [Accessed: 2022-09-30].

Morris, D.D., Colonna, B.R., and Snyder, F.D. (2007). Image-based motion stabilization for maritime surveillance. In *Image Processing: Algorithms and Systems V*, volume 6497, 128–136. SPIE.

Nations, U. (2019). Review of maritime transport 2019. URL `https://unctad.org/system/files/official-document/rmt2019\_en.pdf`. [Accessed: 2020-10-13].

Peterlin, P. (1996). Morphological operations: an overview. URL `http://www.inf.u-szeged.hu/ssip/1996/morpho/morphology.html`. [Accessed: 2022-11-05].

Schwendeman, M. and Thomson, J. (2015). A horizon-tracking method for shipboard video stabilization and rectification. *Journal of Atmospheric and Oceanic Technology*, 32(1), 164–176.

Shannon, C.E. (2001). A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1), 3–55.

Torr, P.H. and Zisserman, A. (2000). Mlesac: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding*, 78(1), 138–156.

Zardoua, Y., Astito, A., and Boulaala, M. (2021). A survey on horizon detection algorithms for maritime video surveillance: advances and future techniques. *The Visual Computer*, 1–21.

Zhang, W. and Naik, S.M. (2012). Camera auto-calibration by horizon estimation. US Patent 8,259,174.