J Mar Sci Technol (2018) 23:164–177
https://doi.org/10.1007/s00773-017-0464-8

ORIGINAL ARTICLE

A quick algorithm for horizon line detection in marine images

Tomasz Praczyk[1]

Received: 4 April 2016 / Accepted: 11 June 2017 / Published online: 5 July 2017

Abstract The information about ship spatial orientation is required by different ship systems. For example, dynamic positioning, artillery, or video tracking systems, to stabilize a ship, a gun or a camera, have to work in advance, and therefore, they need the information about future ship orientation. To obtain this information, a current ship orientation and sometimes also historical orientations are necessary. Another example is ship echo-sounders applied to hydro-graphic surveys. Since each list of the ship affects echo-sounder depth measurement, to determine true depth of the sea based on the echo-sounders, the spatial orientation of the ship at the moment of each measurement has to be known. One method for determining the ship spatial orientation is an optical system including video cameras, a computer and specialized software. The system works through extracting the horizon line from each video shot and calculating an angle between the line and the horizontal border of the shot. The main disadvantage of this approach is its slowness. There are a number of methods which can be used to extract the horizon line from an image; however, they are so computationally complex that their application to high-resolution images is limited rather to only post-processing. In the paper, a quick method for horizon line detection is proposed. The experiments on real marine images showed that it is at least as accurate as other methods, what is more, it is also a number of times faster and it seems to be more reliable than its rivals.

✉ Tomasz Praczyk
t.praczyk@amw.gdynia.pl

[1] Institute of Naval Weapon, Polish Naval Academy, Gdynia, Poland

Keywords Horizon line detection · Marine images · Ship spatial orientation

1 Introduction

Ship motion in waves has to be taken into consideration by most ship systems. For example, during underwater works when a ship supports divers and is rigidly connected with some underwater infrastructure, it is necessary to precisely maintain ship position and heading. To this end, the dynamic positioning systems are used which control ship propellers and thrusters based on the information about the current and historical states of the ship.

To effectively track an object, ship vision and artillery systems have to be stabilized, angle settings of the ship camera/gun have to take into account orientation of the ship with reference to the object and the Earth, the orientation which changes all the time due to ship motion in waves.

The same applies to ship echo-sounders used to hydrographic surveys. Each nonzero value of ship roll or pitch means that echo-sounder measurement is not performed horizontally towards the bottom but is at an angle to it. Knowing the ship spatial orientation for the moment of the measurement, it is possible to correct the measured value and to obtain a true depth of the sea. Whereas in the case of the previously mentioned systems, the ship orientation has to be known in advance that is before applying the system, correction of hydro-graphic measurements can be performed in post-processing.

Currently, to determine the spatial orientation of different objects, inertial systems (IS) are the most often used. They, usually, include three mutually perpendicular accelerometers, gyroscopes, and a three axis

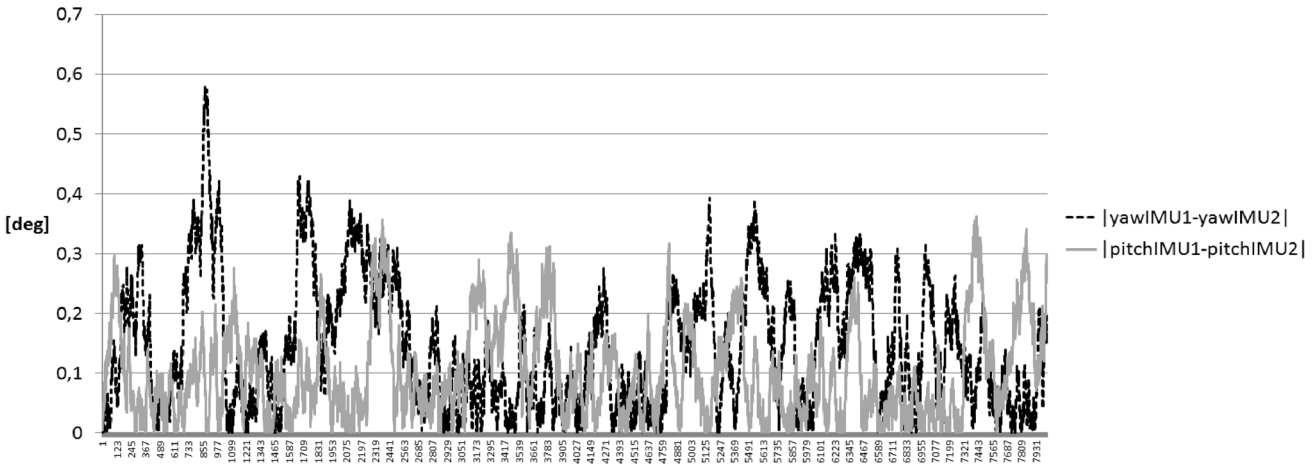


Fig. 1 Differences in measurements of ship spatial orientation performed by two identical ISs (MTx XSENS)

magnetometer. A great advantage of ISs is their high working frequency which amounts even to 100–200 Hz. Thanks to it, ISs can be used, for example, on rackets, planes, and other fast objects whose effective control requires frequent information about their spatial orientation. Accuracy of ISs depends on different factors: first, quality of their components and technology in which each of them is built affect precision of measurements; second, working conditions decide also about indications of ISs, that is, first of all, temperature and magnetic field near the inertial unit. Other important factor affecting accuracy of ISs is an algorithm or algorithms used to fuse data acquired from all the IS components. For example, MTx XSENS (MEMS IS), according to producer website,[1] guarantees 0.5–1 deg of static accuracy and 2 deg of dynamic accuracy. In some cases, e.g., for the dynamic positioning systems, ISs accuracy mentioned above seems to be sufficient; however, in other cases, e.g. for the artillery systems, 2 degree accuracy is definitely too small (Fig. 1).

The optical systems (OS) that determine the spatial orientation through extracting horizon line from an image and calculating angle between the line and the top/bottom border of the image appear to be a valuable alternative for ISs, particularly when a high accuracy is required, speed is not a critical factor, however, the price is. To obtain the high accuracy when using ISs, it is necessary to apply very expensive products, such a situation takes place, for example, on the board of submarines where the OSs and other solutions rather cannot be used. In other cases, indicated above, application of the OSs may yield a satisfied combination of the accuracy and price. For instance, they are often used as a point of reference for ISs in tests whose goal is to indicate the true accuracy of the latter systems. The drawback of the OSs is, however, their low speed compared to ISs, which makes them rather applicable in problems in which spatial orientation can be calculated in post-processing, for example, to determine the sea depth based on hydro-graphic data.

Low speed of OSs is caused by two factors, that is, high-resolution of images acquired from contemporary cameras and thereby large amount of information necessary to process, and, complexity of algorithms dedicated to detect horizon line. An overview of the algorithms is given in [1–3].

Regional covariance-based algorithm (H-COV-LUM) divides the image into two regions, i.e., the sky and ground (sea) region, and for each region, it calculates the variance of luminance. According to H-COV-LUM, the horizon line is the line which minimizes sum of variances for both regions. To detect the horizon line, the method requires intense calculations of region variances for all possible horizon line localizations, with the effect that the method is very complex and thus rather improper for on-the-fly processing.

Edge detection and Hough transform-based algorithm (H-HC) [1, 3–6] works in four stages; first, the original image is filtered to remove slight distortions of high frequency, and erosion or low pass filter is used for that purpose; then, Canny [7] filter is applied to extract strong edges in the image; next, straight lines are detected by means of Hough transform [4], and finally, the horizon line is identified as the longest line from those previously detected. Like the previous method, also H-HC is computationally complex; the cause is the Hough transform in this case.

The main idea of edge detection and least-squares calibration-based algorithm (H-LSC) [1] is to determine maximal vertical local edges in each column of the image and then to find the optimal horizon line by the least-squares method. To remove small distortions of the image

[1] https://www.xsens.com/products/mtx/.

and then outliers, that is, maximal column edges which "deform" the optimal horizon line, morphological erosion and median filter are used in separate stages of the algorithm. In general, such an approach seems to be faster than the previous ones; however, applying the least-squares method, searching each column, pixel after pixel, for vertical edges and double use of morphological filters makes also H-LSC rather improper for on-the-fly processing in maritime conditions, especially, for high-resolution images and standard computer hardware used to process them.

The next algorithm mentioned in [1] is median filtering and linear regression based algorithm (H-MED) which in the idea is very similar to H-LSC. In principle, the only significant difference between the algorithms is the method for vertical edges detection. In this case, each pixel in a column is characterized by absolute difference between two median values determined for the five pixels above and including the considered pixel and the five pixels below. Local edge is located in the pixel characterized by the highest difference. Since calculations performed in H-MED for each pixel are more complex than in H-LSC, the method like the previous one is unsuitable for on-the-fly image processing.

Regional edge magnitudes and least-squares calibration-based algorithm (H-REM) [1] is, in principle, very similar to H-LSC and H-MED. As in the previous case, the difference is in calculating edge pixel in each image column. To this end, H-REM divides the image into vertical stripes each consisting of a number of columns. For each pixel in the stripe, the edge magnitude is calculated, and for that purpose, the method is used which is a combination of the corresponding methods from to H-LSC and H-MED. In the next step, H-REM determines the cumulative edge magnitude for each row in the stripe. The row with the highest value of the cumulative magnitude indicates position of the edge for the central column of the stripe. After locating edges in each column, least-squares method phase follows which finally determines parameters of horizon line. Due to the fact that the general scheme of operation is in H-REM very similar to those applied in H-LSC and H-MED, its usefulness for the on-the-fly processing is more or less at the same level as the usefulness of the two previous methods.

In [8], an approach is presented which applies k-means clustering method to divide the image into two clusters, i.e., the ground (sea) and the sky cluster. An iterative nature of k-means makes, however, that approach improper for real-time processing.

A similar method that is based on intensity-based or k-means clustering is given in [9]. In this case, there are more than two clusters, and horizon line is considered as a cluster that satisfies two criteria, i.e., it stretches through the entire image from left to right and it has the fewest pixels out of all the clusters which satisfy the first criterion—very narrow and long cluster. Since pixels belonging to the horizon line cluster are rarely collinear, to eventually determine the horizon line, the least-squares method can be used. Moreover, k-means and intensity-based clustering can divide the image into clusters with non-connected pixels. To solve this problem and to find cohesive clusters, the union-find algorithm is applied which, however, further increases the computational complexity of the method. Test results mentioned in [9], i.e., 1.5 s of computational time for intensity-based clustering and 10 s for k-means clustering, show unfortunately that also this approach is rather improper for real-time processing in maritime conditions.

In addition to the methods that search for straight horizon lines in the image, there are also the ones which do not assume linearity of the horizon line [10–13] and their purpose is robot localization or visual geo-localization, e.g., during planetary missions. Due to their specificity and low dynamics of objects for which these methods are typically designed (ground robots), there is no high demands on their processing speed; moreover, diverse content of images with which these methods have to cope, usually, affects their greater complexity compared to the methods operating on simpler images with the sea in the background.

In the paper, a new algorithm for horizon line detection and tracking in maritime images called quick horizon line detection (QHLD) is presented. The main characteristic of the QHLD is simplicity and high speed compared to the methods outlined above. The high speed of the algorithm is a necessary condition of its application to real-time calculations of spatial orientation of maritime objects and then, for example, for stabilization of on-board devices. The QHLD was verified with the use of real images recorded during 2 months voyage on Atlantic ocean, and compared with four accelerated variants of H-HC method (H-AHC). The paper reports all the verification and comparison experiments, and it is organized as follows: Sect. 2 is the presentation of the QHLD; Sect. 3 is outline of the H-AHC; Sect. 4 is the report on the experiments; and Sect. 5 is the summary.

2 Quick horizon line detection algorithm

The general idea of the QHLD is to detect the horizon line in a number of iterations in which the original image resized to different sizes is processed. Moreover, the images analyzed in each iteration differ in the applied region of interest (ROI), that is, the area where the line is likely to be located and where it is sought by the algorithm. In the early iterations, size of the images is the smallest, whereas the

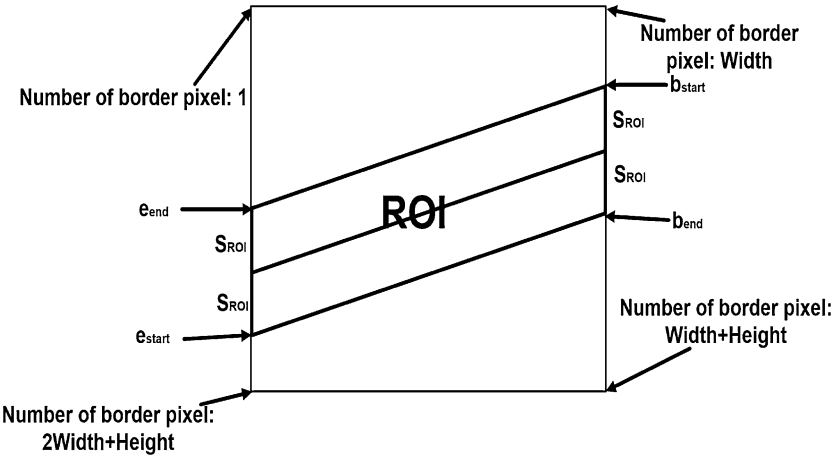


Fig. 2 Example image with ROI and parameters of ROI

ROI is the largest; then, the size increases up to the size of the original image, whereas ROI narrows to only a few possible locations of lines.

In the QHLD, lines in the image are identified by image border pixels which indicate the beginning and end of each line. The pixels are numbered from 1 to 2 (Height $\times$ Width) (see Fig. 2), so to indicate an individual line, two numbers are necessary.

The shape of the ROI in the QHLD is not rectangular as in other approaches, but it is defined by border pixels which indicate possible starting and ending points of the horizon line. Determination of the ROI is different for the first iteration of the algorithm and the remaining iterations. In the first iteration, the ROI includes either all lines that can be drawn in the image, except border lines, or only lines close to the line detected in the first iteration of the previous run of the algorithm. The former case takes place when the first video frame is processed and there is no information where the horizon line can be located in that frame. In the latter case, the ROI is significantly narrowed which is the result of the assumption that the spatial orientation of a maritime object cannot change drastically, and in consequence, the horizon line cannot considerably change its location in successive images.

In the remaining iterations, information from the previous ones is used, that is, the central line of the ROI in the iteration n is a horizon line detected in the image processed in the iteration $n-1$. The size of the ROI is again narrowed to only lines lying in the close proximity of the detected line. Moreover, as mentioned above, the size of ROI is decreased in successive iterations which is due to the assumption that information about the horizon line is more and more precise from iteration to iteration.

To detect the horizon line inside the ROI, the QHLD compares brightness of potential horizon lines with the brightness of parallel lines lying one pixel above[2] (see Fig. 3), and the line characterized by the highest absolute difference in brightness is considered to be the horizon line. In the first iteration, the difference is calculated with a high accuracy, i.e., for all pixels belonging to the compared lines, whereas in later iterations, the difference is approximate and is calculated only for a portion of pixels. The general idea of the algorithm is presented in Fig. 4, whereas its formal definition is given in Figs. 5 and 6.

3 Algorithms compared to QHLD

In the experiments reported further, the QHLD was compared with four accelerated variants of H-HC [1, 3], say, H-AHC1, H-AHC2, H-AHC3, and H-AHC4, that differ in ROI determination and ROI preprocessing. In all the variants, the ROI which, in H-AHC, is traditionally rectangular is determined based on the information about approximated location of the horizon line in the image. To this end, two different solutions are used: the first one applies the QHLD supplied with an input image of highly reduced size and the second one directly situates the horizon line in a region in which it was found earlier (Figs. 2, 3).

With regard to the preprocessing of ROI image, two further solutions are used. The first one blurs the image reducing this way the number of hardly distinctive straight lines which may appear in further stage of processing, and then extracts contours in the blurred image. The second solution, first, extracts contours, then dilates the image a number of times, and finally, extracts contours once again. The objective of dilation is to prepare the image in which the only area characterized by high gradient is the area between the sea and the sky. Contours derived from waves are this way mostly removed from the image, and in effect, they do not cause unnecessary lines to be generated.

In the H-AHC, blurring of an image is performed by means of simple convolution with Gaussian mask. To

[2] Gray-scale images are used.

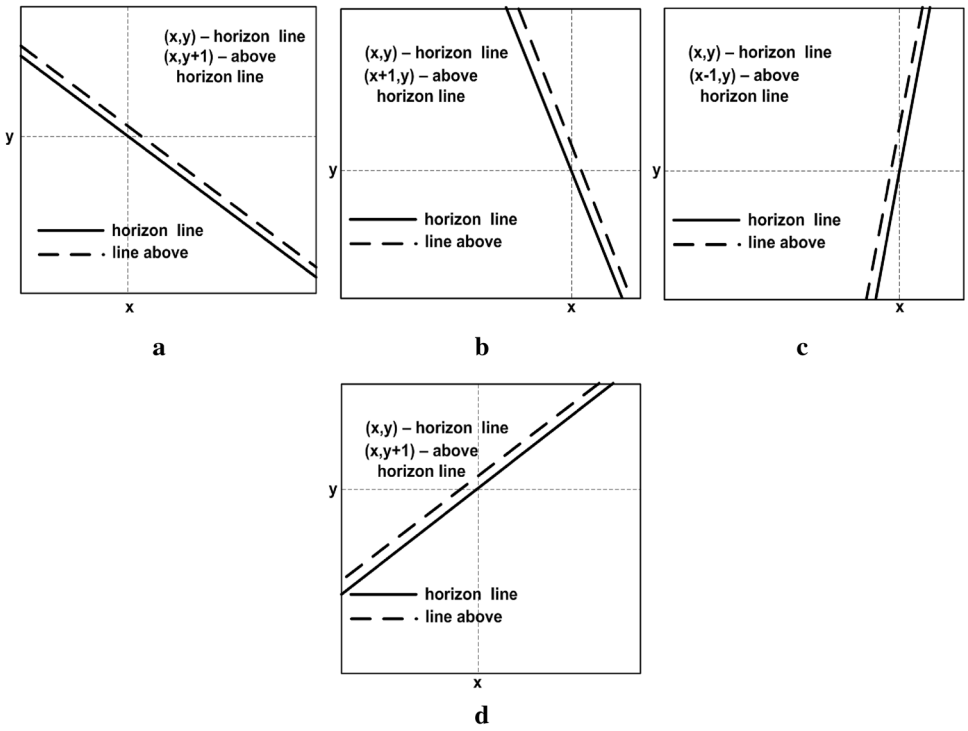


Fig. 3 Way of determining pixels that are above other pixels used in QHLD

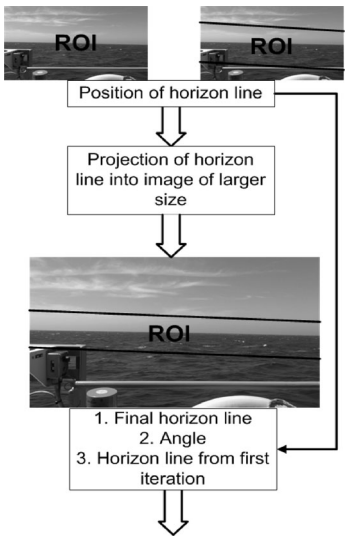


Fig. 4 Operation of QHLD in two iterations

extract contours, Canny filter is used [7], whereas dilate operation is a simple morphological filter which all pixels in the immediate proximity of an active pixel set also active (dilate is used after Canny filter which produces black-and-white image, active pixels are pixels set to 1). Example use of all image processing methods mentioned above is depicted in Fig. 7.

After the ROI determination and the preprocessing stage, the probabilistic Hough transform [4] is applied in all H-AHC variants with the objective to extract straight lines from the ROI image. Since the probabilistic Hough transform extracts a set of lines, to obtain a single horizon line, it is necessary either to select/identify one of the extracted lines as the horizon line, or to build the horizon line with a set of selected lines, or to state that the horizon line is not included in the whole set of extracted lines. In the H-AHC, the solution mentioned in [1] is applied, that is, the longest line out off all lines produced by the Hough transform is considered to be the horizon line. To accelerate calculations and to reduce probability of identification error, a threshold can also be used, in this case, which eliminates lines impossible to represent roll or pitch of the ship, e.g., pitch equal to 45 deg is rather an infrequent phenomenon at sea. Of course, the threshold should be adjusted to conditions at sea; other threshold value should be used for zero and other for three in Beaufort scale.

Four H-AHC variants compared to the QHLD can be eventually and informally defined as follows:

1. H-AHC1:
   - QHLD (ROI determination)
   - Gaussian Blur, Canny (ROI preprocessing)
   - Hough transform (set of lines)
   - longest line (horizon line.

Input:
$\mathbf{O}_{W\times H}$ - image of size WxH
N - number of iterations
$\mathbf{R}_{1\times N}$ - scale factors along the horizontal and vertical axes used to resize input image
$\mathbf{S}^{ROI}_{1\times N}$ - sizes of ROI in pixels
$\mathbf{A}_{1\times N}$ - parameters that reflect accuracy of horizon line search in ROI
$\mathbf{P}_{1\times N}$ - number of pixels used to evaluate a line
b_p^{in}, e_p^{in} - horizon line detected in first iteration of previous algorithm run

$$\texttt{range}(a,b) = \begin{cases} a \text{ if } 1 \le a < b \\ a-b+1 \text{ if } a \ge b \\ b+a-1, \text{ otherwise} \end{cases}$$

Output:
$angle$ - angle determined by horizon line
b_h, e_h - horizon line detected in image $\mathbf{O}$
b_p^{out}, e_p^{out} - horizon line detected in first iteration

```
FOR i := 1 , i ≤ N, i := i + 1
    IF i = N ⇒ O_i := O
    ELSE ⇒ O_i :=resize(O, R_i)
    s_i := 2O_i.Height+2O_i.Width
    bestEval := 0
    IF first run of algorithm AND i = 1 ⇒ b_i := 1; e_i := O_i.Width
        FOR j := 1, j < s_i, j := j + 1
            FOR k := j + 1 , k < s_i, k := k + 1
                evaluateLine(j, k, O_i, P_i, b_i, e_i, bestEval)
    ELSE ⇒
        IF i = 1 ⇒ b_i^t := b_p^in; e_i^t := e_p^in
        ELSE ⇒ b_i^t := b_{i-1} s_i / s_{i-1}; e_i^t := e_{i-1} s_i / s_{i-1}
        b_i := b_i^t; e_i := e_i^t
        b_i^start :=range(b_i^t − S_i^ROI, s_i); b_i^end :=range(b_i^t + S_i^ROI, s_i)
        e_i^start :=range(e_i^t − S_i^ROI, s_i); e_i^end :=range(e_i^t + S_i^ROI, s_i)
        FOR j := b_i^start , j ≤ b_i^end, j := j + A_i
            FOR k := e_i^start , k ≤ e_i^end, k := k + A_i
                evaluateLine(j, k, O_i, P_i, b_i, e_i, bestEval)
    END IF
    IF i = N ⇒ b_h := b_i; e_h = e_i; angle :=getAngle(b_h, e_h)
    IF i = 1 ⇒ b_p^out := b_i; e_p^out = e_i
ENF FOR
```

Fig. 5 Pseudocode of QHLD (b number of border pixel which begins line, e number of border pixel which ends line, b_i, e_i beginning and end of best line in ith iteration, $b_i^{\text{start}}, e_i^{\text{start}}, b_i^{\text{end}}, e_i^{\text{end}}$ vertexes of ROI in ith iteration, evaluateLine() is defined in Fig. 6)

2. H-AHC2:
   - QHLD (ROI determination)
   - Canny, Dilate, Canny (ROI preprocessing)
   - Hough transform (set of lines)
   - longest line (horizon line).
3. H-AHC3:
   - Earlier horizon line (ROI determination)
   - Gaussian Blur, Canny (ROI preprocessing)
   - Hough transform (set of lines)
   - Longest line (horizon line).
4. H-AHC4:
   - Earlier horizon line (ROI determination)
   - Canny, Dilate, Canny (ROI preprocessing)
   - Hough transform (set of lines)
   - Longest line (horizon line).

Formal definition of all the variants is given in Fig. 8.

4 Experiments

All the experiments were divided into two parts. In the first part, the QHLD was compared to H-AHC algorithms presented in Sect. 3, whereas in the second part to the ones described in [1]. The comparisons in both parts were made with the use of images derived from three gray-scale video movies (resolution = 1920 × 1080 px, sampling frequency = 25 Hz, duration = movie no. 1–6.32 min, movie no. 2–10.09 min, and movie no. 3–9.36 min) recorded on Atlantic ocean in different weather conditions (see Fig. 9).

The QHLD and H-AHCs were implemented in C++ programming language with the use of the following OpenCV [14] image processing functions: resize, Canny, dilate, GaussianBlur, and HoughLinesP. In addition to the parameters of the above-mentioned functions that were optimized during the tunning process, there were also parameters whose values were constant and fixed manually: Canny (first threshold for the hysteresis procedure= 0,

Fig. 6 Pseudocode of evaluateLine function

```
Input:
j, k - numbers of border pixels that determine start and end of line
O - image
P - number of pixels used to evaluate line
Input/Output:
b, e - beginning and end of line best so far
bestEval - best evaluation so far

IF j and k do not belong to the same border of image
    //L - set of all pixels on line indicated by j and k
    L :=pixels(j, k, O)
    //L^P - set of P pixels uniformly distributed along line indicated by j and k
    L^P :=select(L, P)
    //A - set of P pixels above pixels belonging to L^P
    A :=above(L^P)
    eval := 0
    FOR i := 1 , i ≤ |L^P|, i := i + 1
        eval := eval + L_i^P.value−A_i.value
    eval := |eval|
    IF eval > bestEval ⇒ bestEval := eval; b := j; e := k
END IF
```

aperture size for the Sobel() operator =3), HoughLinesP (distance resolution of the accumulator in pixels = 1, and angle resolution of the accumulator in radians = 0.05/180).

All the tests were performed on the following computer platform: 64-bit Windows 8.1, Intel(R) Core(TM) i5-2430M CPU 2.4 GHz.

4.1 First part of experiments

4.1.1 Conditions of experiments

To test accuracy, speed, and reliability of the QHLD and to compare it with the methods presented in Sect. 3, two different tests were performed. First, the accuracy and speed were measured, and for that purpose, 45 marine images were applied, each of which contained the pattern horizon line marked manually, and in consequence, a pattern spatial orientation angle is assigned. The images were randomly derived from three gray-scale video movies mentioned above, 15 images for each movie.

To measure the accuracy and speed, each algorithm was run twice for the same image and the measurements were only performed for the second run. The objective was to examine algorithms during normal work, that is, when they can use the information from their previous runs through reduction of ROI size. Since the situation when ROI corresponds to the whole full-size image occurs only ones, i.e., at the very start of the algorithm work, for the first shot from the camera, when comparing QHLD with H-AHCs, this situation was not analyzed.[3]

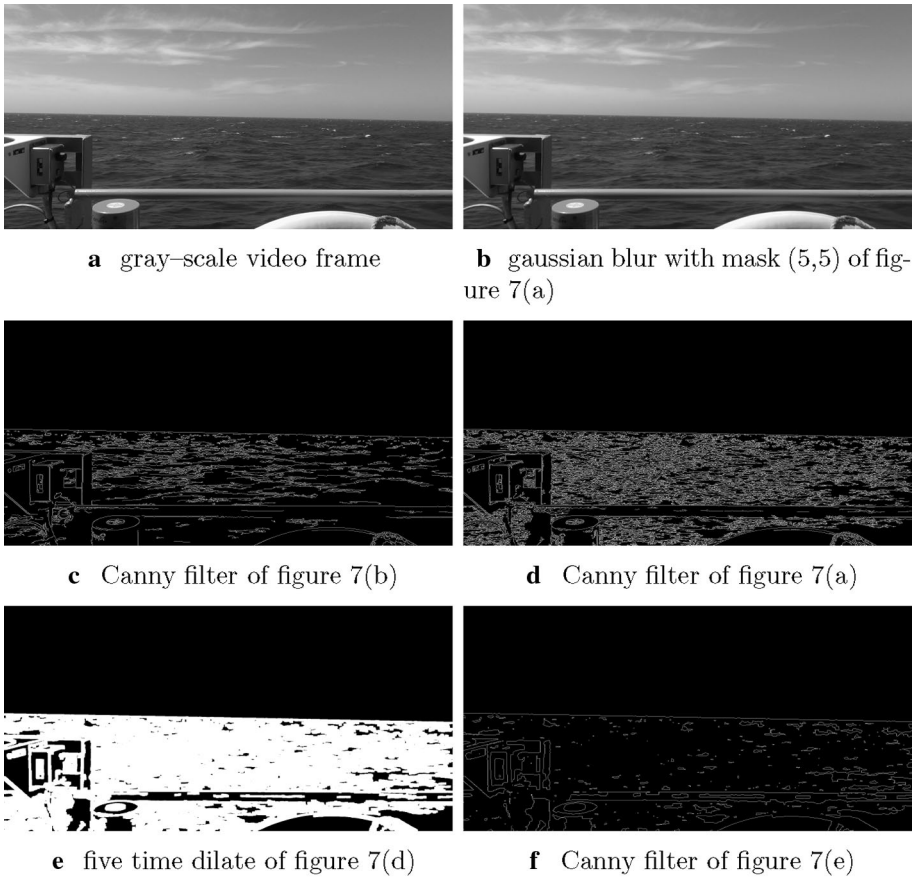
As mentioned above, to test the algorithms on narrowed ROIs, all them were run twice for the same input image. The goal of the first run was mainly to localize ROI in the original full-size input image, as if the first shot from the camera was processed, whereas the task of the second run was to detect horizon line in the previously localized ROI. The reason of such solution is high similarity between two adjacent camera shots for sampling frequency 25 Hz, and in consequence a high probability of the same position of ROI in both images.

The accuracy of each algorithm was determined as an absolute difference between the pattern angle and the angle calculated by the algorithm, expressed in degrees, whereas the speed was measured in milliseconds and it corresponded to the run time of the algorithm.

In addition to tests focused on the accuracy and speed, a long-term reliability of the compared algorithms was also estimated, understood here as the immunity to gross errors. To this end, the algorithms were intensely tested on all the three movies mentioned above. The comparison was, in this case, subjective and was performed through rough visual evaluation of algorithm work, that is, the horizon line detected by the algorithm was drawn in each movie frame and the task of the evaluator was to assess whether the line is in more or less proper location or the gross error occurred.

All the tests performed in the first part of the experiments were preceded by tunning of the algorithms, that is, values of parameters specified in Figs. 5 and 8 were found which minimized maximum error of spatial orientation angle committed for all 45 testing images.

[3] Detection of the horizon line in full-size images, i.e., when ROI is equivalent to the whole image, is considered when comparing QHLD with the methods presented in [1].



a gray–scale video frame

b gaussian blur with mask (5,5) of figure 7(a)

c Canny filter of figure 7(b)

d Canny filter of figure 7(a)

e five time dilate of figure 7(d)

f Canny filter of figure 7(e)

Fig. 7 Image processing methods used in H-AHC

4.1.2 Experimental results

The results of the first part of the experiments are summarized in Tables 1 and 2. With regard to accuracy of each algorithm in determining an angle, it appeared that all them are characterized by more or less the same accuracy. Analysis of horizon lines which were detected by the algorithms showed that, in all cases, that is, for all the testing images and for all the algorithms, the lines were generated properly, which means that they, generally, overlapped with the pattern lines determined manually. Slightly higher accuracy of the QHLD (example operation of QHLD is presented in Fig. 10) compared to all rival algorithms is due to the fact that the length of lines produced by H-AHC was, usually, shorter, in some cases, even significantly shorter than width of the original input image. In consequence, the parameters of the lines were calculated based on only a fragment of the entire horizon line, a part of information available in the image was neglected during calculations which resulted in the situation in which detected horizon lines were imprecisely situated in the image.

With regard to speed of the algorithms, there are significant differences between the QHLD and the remaining algorithms. In the case of the QHLD with three iterations ($N = 3$, example operation of the QHLD with $N = 3$ is presented in Fig. 10), the average run time is almost 20 times shorter than for the fastest H-AHC algorithm, that is, H-AHC2. The remaining H-AHC variants are much slower than both the QHLDs, and acceleration, in this case, amounts even to 60 times for average result and almost 100 times for the maximum run times.

The main cause of that situation is application of the Hough transform which, on the one hand, searches ROI images for all possible straight lines, and on the other hand, it typically produces numerous set of lines which have to be further processed—the length has to be calculated for all them and the longest line has to be indicated. Of course, the size of the set of lines generated by the Hough transform has to be reduced using appropriate values of parameters, for example, through expanding minimum

Fig. 8 Pseudocode of H-AHC

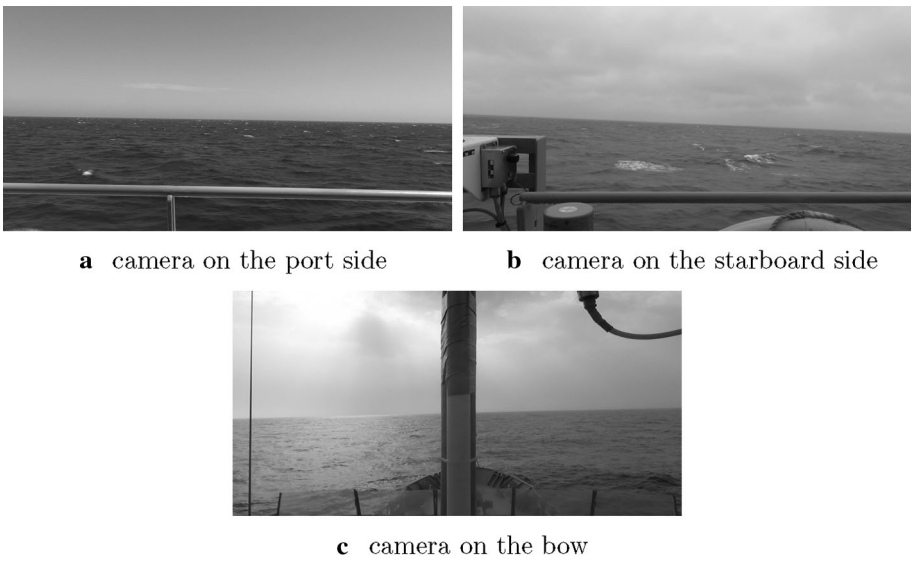
Input:
$\mathbf{O}$ - image
b_p, e_p - horizon line detected in previous run of algorithm
N - number of QHLD iterations
R - scale factor
$\mathbf{P}^{QHLD}$ - other parameters of QHLD
S_{enl}^{ROI} - parameter that determines size of enlarged ROI, in pixels
ROI^{enlarged}.Height=ROI.Height+$2S_{enl}^{ROI}$
P^{Blur} - Gaussian kernel size
P^{Dilate} - number of times dilation is applied
$\mathbf{P}_{1\times2}^{Canny}$ - thresholds for the hysteresis procedure (see OpenCV documentation)
$\mathbf{P}_{1\times3}^{Hough}$ - OpenCV HoughLinesP parameters:
$\mathbf{P}_1^{Hough}$ =Accumulator threshold parameter
$\mathbf{P}_2^{Hough}$ =Minimum line length
$\mathbf{P}_3^{Hough}$ =Maximum allowed gap between points on the same line to link them
Output:
$angle$ - angle determined by horizon line

```
IF QHLD //H-AHC1, H-AHC2
    (b_h, e_h) :=QHLD(N,resize(O, R), b_p, e_p, P^QHLD)
    O_ROI :=getRectangularEnlargedROI(b_h, e_h, S_enl^ROI, O)
ELSE //H-AHC3, H-AHC4
    IF b_p = −1 OR e_p = −1
        O_ROI := O
    ELSE
        O_ROI :=getRectangularEnlargedROI(b_p, e_p, S_enl^ROI, O)
    END
END
IF Dilate //H-AHC2, H-AHC4
    O_ROI :=Canny(O_ROI, P_1^Canny)
    O_ROI :=Dilate(O_ROI, P^Dilate)
    O_ROI :=Canny(O_ROI, P_2^Canny)
ELSE //H-AHC1, H-AHC3
    O_ROI :=Blur(O_ROI, P^Blur)
    O_ROI :=Canny(O_ROI, P_1^Canny)
END
L :=Hough(O_ROI, P^Hough)
(b_h, e_h) :=getLongestLine(L)
angle :=getAngle(b_h, e_h)
```

acceptable length of lines. This solution, however, brought satisfying results only in H-AHC2 (example operation of H-AHC2 is presented in Fig. 11), in the remaining cases, the effect was that the horizon line was not detected in some images—no line met requirements imposed on their length.

The other possible acceleration solution is to decrease the number of lines that can be produced by the Hough transform, and this solution was also applied in H-AHC2 with a good effect. To this end, the dilation was applied several times which resulted in reduction of lines that appeared below the horizon line (lines produced by the sea). In H-AHC1 (example operation of H-AHC1 is presented in Fig. 12), appropriate parameter setting of Canny filter was used to achieve the same effect; however, in this case, lowering sensitivity of Canny filter, once again, resulted in "losing" the horizon line in some cases—there was simply too few edge pixels that could form a straight line.

Worse results of H-AHC3 and H-AHC4 compared to H-AHC algorithms with the QHLD application are generally due to the procedure applied in the tunning process that modified parameters of the former algorithms so as to obtain proper horizon lines for all testing images in the starting iteration when the entire image was processed as ROI. Wrong location of ROI in that iteration resulted in inability of algorithms to find the true horizon line in the next iteration in which they were evaluated. The procedure started with the parameters optimized for H-AHC1 and H-AHC2, and then, they were manually modified to achieve the effect mentioned above. The modification, unfortunately, always led to loosing requirements imposed on lines produced by the Hough transform which increased



a camera on the port side **b** camera on the starboard side

c camera on the bow

Fig. 9 Example video frames used in the experiments. Elements of ship construction visible in the frames were neglected by the algorithms during detection of *horizon line*: they were simply very close to camera and, in effect, they produced very clear edges

Table 1 Angle error in degrees in first part of experiments

Algorithm	Max. error	Min. error	Mean error
QHLD $N = 2$	0.268	0	0.06
QHLD $N = 3$	0.268	0	0.06
H-AHC1 $N = 2$	0.596	0.001	0.103
H-AHC2 $N = 2$	0.445	0	0.075
H-AHC3	0.84	0	0.107
H-AHC4	0.468	0	0.054

Table 2 Run time in milliseconds in first part of experiments

Algorithm	Max. time	Min. time	Mean time
QHLD $N = 2$	78	62	67.4
QHLD $N = 3$	36	34	34.888
H-AHC1 $N = 2$	3841	719	1446.64
H-AHC2 $N = 2$	859	375	588.911
H-AHC3	3482	984	2204.4
H-AHC4	4183	812	2232.02

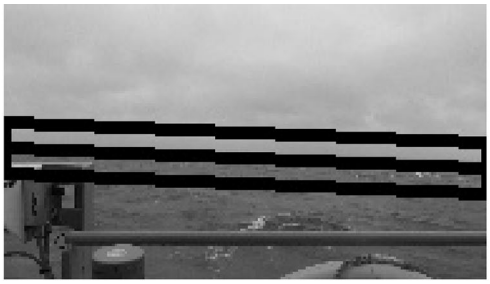
their number and, in consequence, duration of the whole processing.

In the experiments, two variants of the QHLD were tested, i.e., the one with $N = 2$ and the other one with $N = 3$, with the purpose to not excessively prolong calculations. It was assumed that two or three iterations are enough to find a proper horizon line (parameter setting for the QHLD and for the remaining algorithms is given in Appendix). As it appeared, both variants achieved the same accuracy which is due to the fact that they both always detected the same horizon line. The difference between the variants is noticeable when comparing the run time. In this case, surprisingly, the algorithm with $N = 3$ appeared a more effective solution, in spite of the fact that it contains one iteration more than the algorithm with $N = 2$. The cause is the size of ROI in the second iteration of the QHLD with $N = 2$ which was larger than the total size of ROIs in the second and third iterations for $N = 3$. Such size of ROI was necessary because of imprecise information about the horizon line after the first iteration of the algorithm.

After comparing the accuracy and speed of the algorithms, their reliability during long-term operation was estimated. To this end, the three already mentioned movies were applied and the task of the algorithms was to continuously determine horizon lines for all frames of the movies. In this case, only the first frame in each movie was processed as one compact ROI, reduced ROIs of the remaining frames were determined based on the information from previous frames. Such an approach means that gross errors made in one frame may cause errors in the following frames and in consequence "detunning" of the entire horizon line detection process.

The experiments in this stage showed, generally, that the QHLD is the only algorithm that is able to reliably keep track of the horizon line recognized subjectively by a man as a true horizon line. As it turned out, the concept applied in the QHLD made the algorithm able to effectively, and what is particularly worth emphasizing also quickly, detect horizon lines in all the three movies, without any noticeable error.

Algorithms without the QHLD committed frequent errors, mainly because of wrong selection of the horizon line out of all lines produced by Hough transform. There were simply situations in which the true horizon line was



a 1st iteration, scale factor=0.1

b 2nd iteration, scale factor=0.5

c 3rd iteration, original image

Fig. 10 Example operation of QHLD: ROIs and *horizon lines* detected by the algorithm in subsequent iterations (the *horizon line* is a *line* between *two lines* that indicate ROI, sizes of images in the paper do not reflect their true sizes)

not the longest line which appeared in the image. The errors made while processing the movies did not always lead to subsequent errors; in some cases, the errors were small enough, so that the algorithms quickly came back to a proper ROI and horizon line. However, in most cases, when a gross error occurred, detected ROI and horizon line gradually moved away from the true ROI and horizon line.

In H-AHC1 and H-AHC2, application of the QHLD to locate ROI improved the situation. In this instance, errors resulting from the Hough transform occurred just as often as in the previous case; however, they did not cause a permanent displacement of ROI in the wrong place, and in consequence, inability of the algorithms to detect the horizon line.

4.2 Second part of experiments

4.2.1 Conditions of experiments

In addition to H-AHCs, the QHLD was also roughly compared to the algorithms presented in [1]. To this end, extra tests were carried out whose results were compared to those given in [1].

Since all the algorithms described in [1] were applied to detect horizon lines in full-size images, not in small size ROIs, extra measurements of QHLD speed and accuracy, also for original full-size images, were necessary. In this case, a variant of the algorithm was applied which assumes that the horizon line lies between left- and right-side border of each image. Such solution adjusted QHLD to H-LSC, H-MED, and H-REM which through analysis of all columns in the image make the same assumption with respect to the horizon line.

Four factors had influence, in this instance, on reliable comparison, i.e., computer and implementation platforms applied during the tests as well as resolution and color depth of testing images. Unfortunately, the work [1] does not provide any information about the testing computer platform; however, we can assume that its performance is comparable to our 64-bit Intel Core platform.

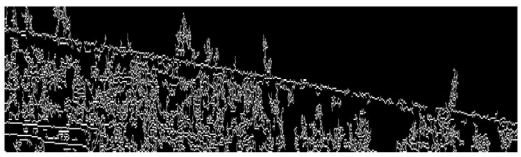
As for the algorithm implementation platform, authors of [1] chose MATLAB environment which seems to be slightly slower solution than ours based on executable C++/OpenCV program, with the effect that run times given in [1] may be somewhat longer in relation to ours regardless of the algorithm.

The color depth of the testing images was also different in both cases, in the experiments reported in [1], 24-bit color images were used, whereas in ours, 8-bit gray-scale ones were applied. The effect of that difference is the same as above, that is, longer run time of algorithms working on 24-bit images independent of the algorithms themselves.

Resolution of the testing images seems, however, to have the greatest influence on reliable comparison of the algorithms: 1920 $\times$ 1080 px images tested in our experiments contain almost four times as many pixels as 900 $\times$ 675 px images and 50 times as many pixels as 249 $\times$ 169 px images used in the experiments reported in [1]. To make comparisons as much reliable as possible, they were performed with the use of the same 45 marine images applied in the previous part of the experiments reduced, however, to the following sizes: 269 $\times$ 151 and 960 $\times$ 540 px. The tests were performed for two new resolutions, because [1] does not specify which results reported in the paper were achieved for which resolution.

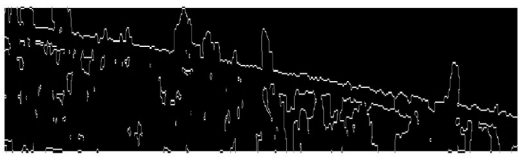
In this part of the experiments, the most effective variant of the QHLD from the previous part was applied. There was not extra tunning process, in this case, which means

a 1st iteration of QHLD, scale factor=0.1 **b** 2nd iteration of QHLD, scale factor=0.5



c ROI after 1st Canny

d ROI after Dilate



e ROI after 2nd Canny

Fig. 11 Example operation of H-AHC2: ROIs in subsequent iterations of the algorithm after QHLD preprocessing (the *horizon line* in images **a** and **b** is a *line* between *two lines* that indicate rectangular ROI, images no. **c**–**e** are intentionally resized in *vertical axis*)

that the algorithm used the parameters adjusted to images of larger size.

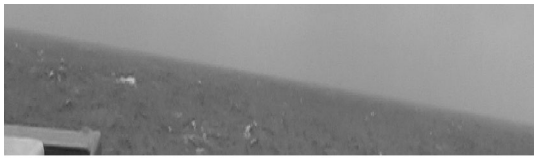
4.2.2 Experimental results

The second part of the experiments showed that the QHLD is less accurate than the algorithms presented in [1]. Regardless of the applied image resolution, the average error of the QHLD (0.985 and 0.924 deg) is almost twice as high as the error of the least accurate rival method (0.47 deg)—see Table 3.

There are two likely explanations of such result. The first is lower resolution of images, whereas the second is calculation of errors based on improper pattern angle values. The pattern values used in this part of experiments were the same as those applied earlier. To fix them, pattern horizon lines marked manually in original 1920 × 1080 px images were used. Since location of pattern horizon lines can be slightly different in original images and resized ones, accuracy of the QHLD obtained in this stage of the experiments may be slightly distorted in relation to the previous stage. The evidence of this can be Fig. 13 with horizon lines fixed by the QHLD. Even though they seem to be located in a proper place, they are characterized by the maximum angle error (2.69, 2.41deg—for both resolutions, maximum error occurred for the same image).

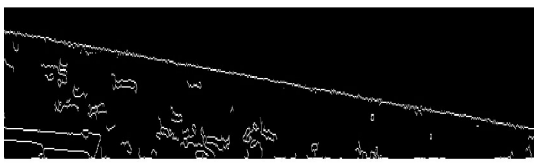
The algorithms were also compared in terms of processing speed—see Table 4. Again, it appeared that the algorithm of horizon line detection applied in the QHLD is the fastest out of all compared solutions. The average QHLD run times amount, in this case, to 11.4 and 47.28 ms, whereas the best result of other algorithms is 140 ms, which means at least triple acceleration of the QHLD.

In the algorithms presented in [1], the horizon line is detected as a result of intense processing of original size image. Meanwhile, in the QHLD, a number of images are processed; however, all of them are significantly smaller in size than the original image. For example, to detect the horizon line in an image of size 269 × 151 px, the following three images were processed: 27 × 15 px image, ROI in 133 × 125 px image, and ROI in the original image. The consequence of this is a significant acceleration of the QHLD with respect to other methods.



a ROI

b ROI after Gaussian blur



c ROI after Canny

Fig. 12 Example operation of H-AHC1: ROIs in subsequent iterations of the algorithm (all images are intentionally resized in *vertical axis*)

Table 3 Angle error in degrees in second part of experiments (results of the last five algorithms taken from [1])

Algorithm	Max. error	Min. error	Mean error
QHLD 269 × 151 $N = 3$	2.69	0.059	0.985
QHLD 960 × 540 $N = 3$	2.41	0.062	0.924
H-LSC			0.23
H-COV-LUM			0.47
H-HC			0.13
H-MED			0.44
H-REM			0.19

5 Summary

The paper proposes the QHLD, i.e., a quick algorithm for horizon lines detection in marine images. The experiments reported in the paper compared the QHLD with a number of algorithms based on the Hough transform. Moreover, rough comparison was also made with algorithms presented in [1]. All the comparisons proved that the QHLD is an effective tool for detection of the horizon line in maritime conditions. Accuracy of determining the line by the algorithm is more or less at the same level as the accuracy of other top algorithms. Noteworthy is, however, the speed of the QHLD which is considerably higher than the speed of all rival algorithms. This feature of the QHLD when combined with a powerful computer platform makes it an appropriate tool for real-time calculations, e.g., for stabilization of different ship devices.

a 269x151 image – 2.69 deg error of fixing pattern horizon line

b 960x540 image – 2.41 deg error of fixing pattern horizon line

Fig. 13 *Horizon lines* with maximum angle errors

Table 4 Run time in milliseconds in second part of experiments (results of the last five algorithms taken from [1])

Algorithm	Max. time	Min. time	Mean time
QHLD 269 × 151 $N = 3$	14	11	11.4
QHLD 960 × 540 $N = 3$	63	46	47.28
H-LSC			330
H-COV-LUM			2240
H-HC			450
H-MED			1460
H-REM			140

Appendix: Parameter setting after tunning process

Parameters of the QHLD:

- QHLD $N = 2$: $\mathbf{R} = \{0.1\}$, $\mathbf{S}^{ROI} = \{10, 20\}$, $\mathbf{A} = \{1, 1\}$, $\mathbf{P} = \{image.\text{Width}, 150\}$
- QHLD $N = 3$: $\mathbf{R} = \{0.1, 0.5\}$, $\mathbf{S}^{ROI} = \{10, 10, 6\}$, $\mathbf{A} = \{1, 1, 1\}$, $\mathbf{P} = \{image.\text{Width}, 100, 100\}$

Parameters of the H-AHC:

- H-AHC1: $N = 2, R = 0.1, \mathbf{S}^{ROI} = \{10, 20\}, \mathbf{A} = \{1, 2\}, \mathbf{P} = \{image.\text{Width}, 100\}, S_{enl}^{ROI} = 20, P^{Blur} = 9, \mathbf{P}^{Canny} = \{30\}, \mathbf{P}^{Hough} = \{160, 200, 10\}$
- H-AHC2: $N = 2, R = 0.1, \mathbf{S}^{ROI} = \{10, 10\}, \mathbf{A} = \{1, 2\}, \mathbf{P} = \{image.\text{Width}, 100\}, S_{enl}^{ROI} = 20, P^{Blur} = 9, \mathbf{P}^{Canny} = \{60, 50\}, P^{Dilate} = 4, \mathbf{P}^{Hough} = \{10, 400, 60\}$
- H-AHC3: $S_{enl}^{ROI} = 20, P^{Blur} = 9, \mathbf{P}^{Canny} = \{30\}, \mathbf{P}^{Hough} = \{160, 200, 30\}$
- H-AHC4: $S_{enl}^{ROI} = 20, P^{Blur} = 9, \mathbf{P}^{Canny} = \{60, 10\}, P^{Dilate} = 3, \mathbf{P}^{Hough} = \{110, 200, 60\}$.

References

1. Gershikov E, Libe T, Kosolapov S (2013) Horizon line detection in marine images: which method to choose? Int J Adv Intell Syst 6(1 and 2):79–88
2. Libe T, Gershikov E, Kosolapov S (2012) Comparison of methods for horizon line detection in sea images. In: Proc. CONTENT 2012, Nice, pp 79–85
3. Lipschutz I, Gershikov E, Milgrom B (2013) New methods for horizon line detection in infrared and visible sea images. Int J Comput Eng Res (ijceronline.com) 3(3):226–233
4. Duda RO, Hart PE (1972) Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 15:11–15
5. Wang Y, Liao Z, Guo H, Liu T, Yang Y (2009) Anapproach for horizon extraction in ocean observation. In: Proc. IEEE congress on image and signal processing, Tianjin, pp 1–5
6. Wipping D, Klauer B, Seidel O, Zeidler H (2006) Removing the horizon in the edge representation of infrared images. In: Proc. of the ASEE mid-atlantic section spring 2006 Conference, 28–29 Apr 2006, New York
7. Canny J (1986) A computational approach to edge detection. IEEE Trans PAMI 8:679–697
8. Lee JM, Lee KH, Kim DS, Nam BW, Li R (2014) Image-based ship pose estimation for AR sea navigation. In: Advanced science and technology letters, vol 58 (Software 2014), pp 14–20
9. Boroujeni NS, Etemad SA, Whitehead A (2012) Robust horizon detection using segmentation for UAV applications. In: 2012 ninth conference on computer and robot vision, 28–30 May 2012, Toronto, pp 346–352
10. Ahmad T, Bebis G, Regentova E, Nefian A (2013) A machine learning approach to horizon line detection using local features. In: 9th international symposium, ISVC (2013) Rethymnon, Crete, July 29–31, 2013. Proceedings, Part I, pp 181–193
11. Ahmad T, Bebis G, Nicolescu M, Nefiany A, Fong T (2015) An edge-less approach to horizon line detection. In: 14th IEEE international conference on machine learning and applications (ICMLA'15), Miami, December 9-11, 2015. https://www.researchgate.net/publication/291697063
12. Fefilatyev S, Smarodzinava V, Hall LO, Goldgof DB (2006) Horizon detection using machine learning techniques. In: Proc. international conference on machine learning and applications, pp 17–21
13. Yazdanpanah AP, Regentova EE, Muthukumar V, Bebis G (2015) Real-time horizon line detection based on fusion of classification and clustering. Int J Comput Appl (0975 8887) 121(10):5–11
14. http://docs.opencv.org/2.4.0/