A Vision-based Obstacle Detection System for Unmanned Surface Vehicle

Han Wang[1], Zhuo Wei[1], Sisong Wang[1], Chek Seng Ow[2], Kah Tong Ho[2], Benjamin Feng[2]
[1] School of Electrical & Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798
[2] Singapore Technologies Electronics Limited, 24 Ang Mo Kio Street 65, Singapore 569061
[1] hw@ntu.edu.sg
[2] owcs@stee.stengg.com

***Abstract*—This paper describes a vision-based obstacle detection system for Unmanned Surface Vehicle (USV) towards the aim of real-time and high performance obstacle detection on the sea surface. By using both the monocular and stereo vision methods, the system offers the capacity of detecting and locating multiple obstacles in the range from 30 to 100 meters for high speed USV which runs at speeds up to 12 knots. Field tests in the real scenes have been taken and the obstacle detection system for USV is proven to provide stable and satisfactory performance.**

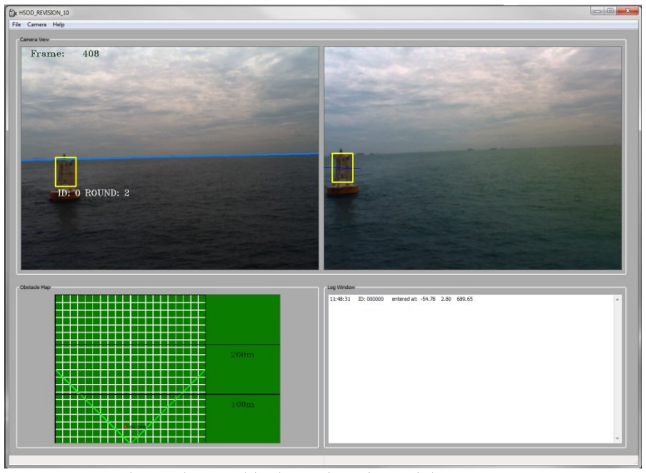
***Keywords*— computer vision, obstacle detection, Unmanned Surface Vehicle (USV)**

1. INTRODUCTION

Unmanned Surface Vehicles (USV) are expected to act important role in future civilian and military operations in place of human. Functioning on the surface of water, the USV may be required to complete various tasks autonomously so it is a must for a practical USV to include situational awareness by sensing the immediate environment to locate obstacles. In order to fulfil the requirements, a vision-based obstacle detection system for Unmanned Surface Vehicle is developed, which offers good performance and is low in cost. Both monocular and stereo vision methods are implemented. It is capable of detecting near-field obstacles on the sea surface, such as buoys, ships and so on.

The hardware of the system is composed of two Point Grey grasshopper CCD cameras with high speed FireWire connection, which are both mounted parallel about 1.5 meters apart on a metal bar. The software, of which the performance is boosted by applying multi-threading programming to achieve real-time obstacle detection, runs under the Windows XP. For ease of use, we have created a graphical user interface for the system. Fig. 1 shows the interface in which the left and right camera views, the obstacle map and the obstacle detection log window are displayed.

We use the image from the camera on the left to perform monocular obstacle detection initially. The stereo approaches are then applied to process the images from the cameras on both sides to compute the 3D detection results.



```
11:48:31    ID: 000000    entered at:  -54.78   2.80   689.65
```

Fig. 1 The graphical user interface of the system

2. OBJECT DETECTION BY MONOCULAR CAMERA

A. sea-sky line detection by pixel profile analysis and RANSAC

An important first step is to detect the sea-sky line so that we could distinguish the sea surface for further obstacle detection and calculate the sea surface normal for 3D to 2D projection discussed in the latter sections.

In the sea-sky background images, the brightness of sky area is generally much different from that of the sea area. The boundary between the two reveals a sharp transition from bright to dark. Based on this feature, we have proposed a method by using pixel profile analysis and RANSAC to detect the line.

The gray-level image is firstly blurred by a two dimensional Gaussian filter with the three by three pixel kernel window to reduce the level of noise. Following the smoothing operation, the Sobel operator is applied and calculates the gradient approximations for vertical at each pixel. By examining the pixel profile from a random vertical line on the resulting gray-level image, the maximum or minimum value, which indicates the abrupt image changes, most likely occurs at the boundary pixel on the sea-sky line. Thus we measure the pixel profiles of 20 equidistant parallel lines on the image and keep a record of the points with maximum or minimum value. A fair portion of the points are on the sea-sky line while others are error. Then it is rational to use the RANSAC method performing the line fitting and extract the sea-sky line, as shown in Fig. 2. The sea-sky line detection method is robust and works fast. Fig. 3 shows some results in different scenarios.

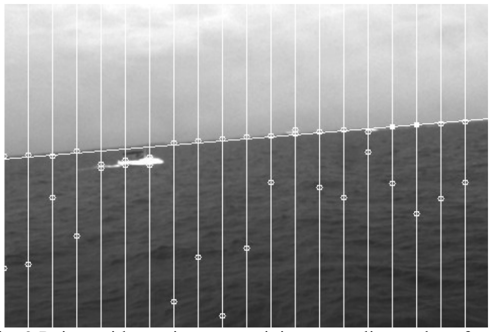


Fig. 2 Points with maximum or minimum gradient values from equidistant vertical lines

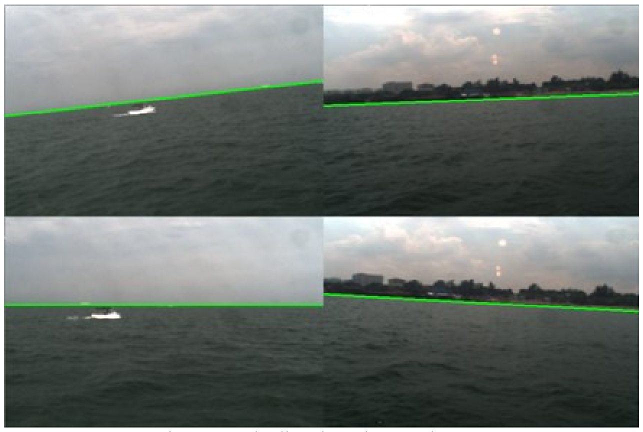


Fig. 3 Sea-sky line detection results

B. Saliency detection

Most of the time, the colours and texture patterns of the sea surface represent in a relatively stable state, so the obstacles on the sea are visual pop-outs which can be located by the saliency detection. The approach efficiently suggests the salient regions in the image which may contain potential obstacles, although it forms the rough estimation stage of the proposed obstacle detection.

We use the saliency method proposed by R. Achanta [1]. It finds the Euclidean distance between the L*a*b* color pixel vector in the Gaussian blurred image and the average L*a*b* vector for the original color image. The saliency map S for the image I of width W and height H is computed as:

$$\mathrm{S(x,y)}=\|I_m(x,y)-I_G(x,y)\|$$

where $I_m(x,y)$ is the mean image vector calculated by $I_m=\frac{\sum I}{H\times W}$ and $I_G(x,y)$ is the Gaussian blurred image (the kernel size is 3 by 3). The norm is the L_2 norm giving the Euclidean distance. For each pixel location, the vector is in the form of $[L,a,b]^T$. After the saliency computation conducted, the achieved saliency map is then normalized to limit the saliency values in the range $[0.0, 255.0]$.

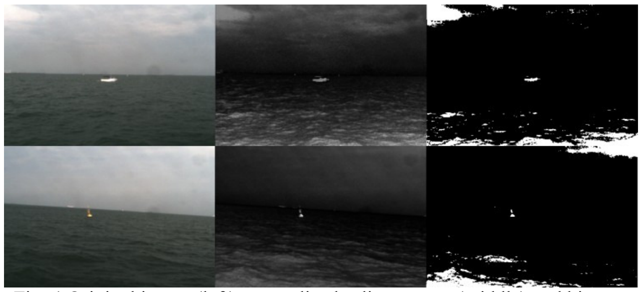


Fig. 4 Original image (left), normalized saliency map (middle) and binary mask (right)

In order to obtain a binary mask for the salient regions in the image, an adaptive threshold T_α is applied to all pixels of the saliency map. The threshold T_α value is determined as α times the mean saliency map:

$$T_\alpha=\alpha\times\frac{\sum \mathrm{S(x,y)}}{\mathrm{W\times H}}$$

where the value of α depends on the pixel location with respect to the sea-sky line. For pixels below the sea-sky line, α equals to 1.1, or otherwise 2.5. Fig. 4 shows some resultant examples. The detected saliency results will be given in the form of bounding boxes around positive regions in the binary mask.

C. Harris corner extraction and tracking

Saliency detection gives a rough estimation for the obstacle detection by finding the visually salient regions on the sea surface. However, it is not sufficient since it may generate false alarms by labelling sea waves and sun reflections as positive detections. After carefully measuring the movement of both the sea surface and obstacles in many different scenarios, we have noticed that obstacles are travelling continuously in a regular manner in the image sequence while other non-obstacles are not. Based on this characteristic of the surface obstacles, motion evaluation is possible to distinguish surface obstacles from the potentials suggested by the saliency detection. In our system, Harris corner detector [2] extracts the corner features and feeds them to the pyramidal Lucas-Kanade feature tracker [3] for motion estimation.

1) Harris corner detector: The algorithm of Harris corner detector is briefly presented here. In the first place, smooth the image by Gaussian filter and compute the partial derivatives I_x and I_y of the blurred image I:

$$\begin{cases} I_x(x,y)\approx I(x+1,y)-I(x,y) \\ I_y(x,y)\approx I(x,y+1)-I(x,y) \end{cases}$$

Then compute the sums of the products of derivatives at each pixel:

$$\begin{cases} A(x,y)=\sum_W I_x(x,y)^2 \\ B(x,y)=\sum_W I_x(x,y)\,I_y(x,y) \\ C(x,y)=\sum_W I_y(x,y)^2 \end{cases}$$

where W specifies a image window that pixel $(x, y) \in W$, image $A(x, y)$ is the convolution of image $I_x(x, y)^2$ and so on.

After that, define at each pixel the 2×2 matrix M:

$$M(x, y) = \begin{bmatrix} A(x, y) & B(x, y) \\ B(x, y) & C(x, y) \end{bmatrix}$$

and find corner points as local maxima of $\mathrm{M}(\mathrm{x}, \mathrm{y})$. The local maximum is defined as a pixel greater than its neighbours.

*2) Pyramidal implementation of the Lucas-Kanade feature tracke*r: The detected corner features are fed to the pyramidal implementation of the Lucas-Kanade optical flow feature tracker. It provides the feature correspondences between two consecutive image frames.

Let I and J be two grayscale images. The two quantities $I(\boldsymbol{x}) = I(x, y)$ and $\mathrm{J}\,(\boldsymbol{x}) = J(x, y)$ are then the grayscale value of the two images at the location $\boldsymbol{x} = [x\ y]^T$, where x and y are the two pixel coordinates of a generic image point $\boldsymbol{x}$. The image I will be referenced as the first image, and the image J as the second image.

Consider an image point $\boldsymbol{u} = [u_x\ u_y]^T$ on the first image I. The goal of feature tracking is to find the location $\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{d} = [u_x + d_x\ \ u_y + d_y]^T$ on the second image J such as $I(\boldsymbol{u})$ and $J(\boldsymbol{v})$ are "similar". The vector $\boldsymbol{d} = [d_x\ d_y]^T$ is the image velocity at $\boldsymbol{x}$, also known as the optical flow at $\boldsymbol{x}$. It is essential to define the notion of similarity in a 2D neighborhood sense. Let ω_x and ω_y two integers determine the half size of the so called integration window above $\boldsymbol{x}$. We define the image velocity $\boldsymbol{d}$ as being the vector that minimizes the residual function ϵ defined as follows:

$$\epsilon(\boldsymbol{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} \big(I(x,y) - J(x + d_x, y + d_y)\big)^2$$

The pyramidal Lucas-Kanade optical flow feature tracker is the way to solve the problem. The iterative implementation offers an accurate solution. The summary of the entire tracking algorithm in the form of pseudo-code is shown as below.

```
Goal: Let $\boldsymbol{u}$ be a point on image $I$. Find its corresponding location $\boldsymbol{v}$ on image $J$.
Build pyramid representations of $I$ and $J$: $\{I^L\}_{L=0,\ldots,L_m}$ and $\{J^L\}_{L=0,\ldots,L_m}$
Initialization of pyramidal guess: $g^{L_m} = [g_x^{L_m}\ g_x^{L_m}]^T = [0\ 0]^T$
for $L = L_m$ down to 0 with step of -1
    Location of point $\boldsymbol{u}$ on image $I^L$: $\boldsymbol{u}^L = [p_x\ p_y]^T = \boldsymbol{u}/2^L$
    Derivative of $I^L$ with respect to $x$:
        $I_x(x,y) = \dfrac{I^L(x+1,y) - I^L(x-1,y)}{2}$
    Derivative of $I^L$ with respect to $y$:
        $I_y(x,y) = \dfrac{I^L(x,y+1) - I^L(x,y-1)}{2}$
    Spatial gradient matrix:
        $G = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2 \end{bmatrix}$
    Initialization of iterative L-K: $\bar{\boldsymbol{v}}^0 = [0\ 0]^T$
    for $k = 1$ to K with step of 1 (or until $\|\bar{\boldsymbol{\eta}}^k\|$ accuracy threshold)
        Image difference:
            $\delta I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + v_x^{k-1}, y + g_y^L + v_y^{k-1}$
        Image mismatch vector:
            $\bar{\boldsymbol{b}}_k = \sum_{x=p_x-\omega_x}^{p_x+\omega_x} \sum_{y=p_y-\omega_y}^{p_y+\omega_y} \begin{bmatrix} \delta I_x(x,y) & I_x(x,y) \\ \delta I_x(x,y) & I_y(x,y) \end{bmatrix}$
        Optical flow (Lucas-Kanade): $\bar{\boldsymbol{\eta}}^k = G^{-1}\bar{\boldsymbol{b}}_k$
        Guess for next iteration: $\bar{\boldsymbol{v}}^k = \bar{\boldsymbol{v}}^{k-1} + \bar{\boldsymbol{\eta}}^k$
    end of for-loop on $k$
    Final optical flow at level $L$: $d^L = \bar{\boldsymbol{v}}^k$
    Guess for next level $L - 1$:
        $g^{L-1} = [g_x^{L-1}\ \ g_y^{L-1}]^T = 2(g^L + d^L)$
end of for-loop on $L$
Final optical flow vector: $\boldsymbol{d} = \boldsymbol{g}^0 + \boldsymbol{d}^0$
Location of point on $J$: $\boldsymbol{v} = \boldsymbol{u} + \boldsymbol{d}$
Solution: The corresponding point is at location $\boldsymbol{v}$ on image $J$
```

At each computation iteration the Harris corner features are extracted from the cached previous image and then tracked by the optical flow approach in the current image to find the feature correspondences. Fig. 5 shows the example results in which green points are referenced as the successfully tracked corner features in the current image, and the blue ones as their counterparts in the previous image. The white yacht at the close range and the oil tankers in the distance have been detected and tracked. However, there are tracked corner features which belong to the sea waves.

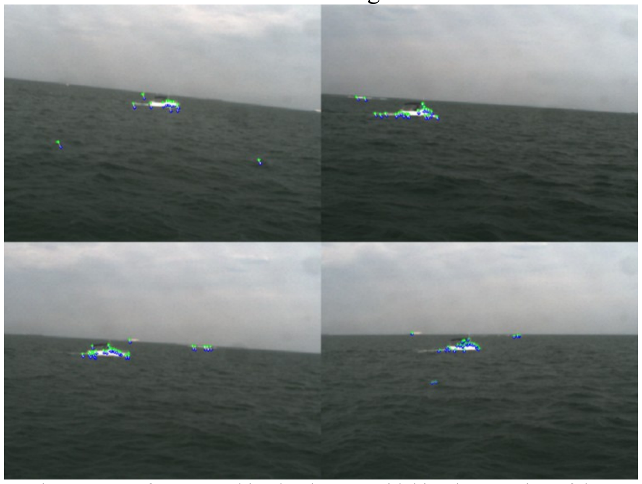


Fig. 5 Corner feature tracking by the pyramidal implementation of the Lucas-Kanade optical flow feature tracker.

D. Combined monocular obstacle detection

In the monocular obstacle detection module, the sea-sky line detection, saliency detection and motion estimation outputs are combined to generate the final obstacle detection results. The rationale behind the proposed approach is based on the fact that meaningful obstacles have continuous trajectory while disturbances, such as sun waves and sun reflections, merely appear in the field of view for a short period of time. Consequently, it is practical to measure the dynamics of potential object to verify its validity as an obstacle. Sea-sky line detection helps to determine off-shore and on-surface objects. Saliency detection suggests the locations of potential obstacles which could be false positives or true positives. The motion estimation by optical flow tracker is calculated to refine the outputs from saliency detection.

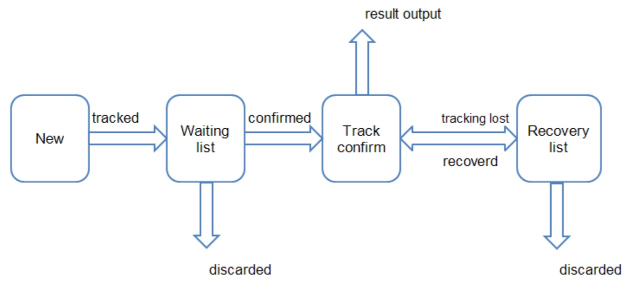


Fig. 6 Status transitions between different statuses in the obstacle detection module

In image $I(t)$ a newly appeared object in the field of view is firstly discovered by the saliency detection and labelled as “NEW”. The tracked corner features between image $I(t)$ and $I(t+1)$ are then checked. If there are a certain number of corner features inside the object bounding box of image $I(t)$ whose correspondences fall inside a saliency bounding box in image $I(t+1)$, these two respective bounding boxes in image $I(t)$ and $I(t+1)$ will be associated and referenced as tracked. The bounding box in image $I(t+1)$ is updated in the waiting list and has its status transited from “New” to “Waiting list”. In the following frames, the bounding box in the waiting list will be detected and tracked by the same means to be confirmed as a meaningful obstacle and labelled as “Track confirm” until 15 consecutive times of association are achieved. Otherwise, it will be discarded if fails to complete the confirmation procedure. The objects with the status of “Track confirm” are outputted as detected obstacles for further stereo detection. The tracking and association of the confirmed obstacle is still performed afterwards. For the confirmed obstacle a failure in the tracking and association will not collapse the whole procedure. Instead the obstacle is put into the recovery list for observation. If the tracking and association works successfully on the object again, it will return to the status “Track confirm”, or it will be discarded. See Fig. 6 for the status transitions.

Fig. 7 shows some results of monocular obstacle detection in two different scenarios.

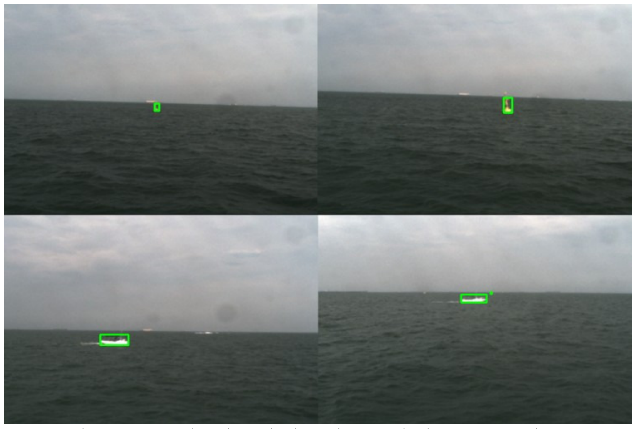


Fig. 7 Monocular obstacle detection results in two scenarios

3. STEREO OBSTACLE DETECTION

A. Camera calibration and image rectification

We use the Camera Calibration Toolbox for Matlab by Jean-Yves Bouguet [4] to calibrate the stereo system intrinsically and extrinsically and use the calibration results for stereo image rectification and 3D reconstruction. The calibration computation is based on a number of images of a planar checkerboard. Since the baseline of our stereo system is relatively wide (up to 1.5m) to ensure the long range stereo detection accuracy, we have built a large 9×6 checkerboard of which the size is about 1750×1180 mm for calibration. The size of the black and white squares is uniformly 189.40×187.70mm. Sequences of multi-viewed images of the checkerboard patterns are taken and fed to the toolbox. Thus, for the individual cameras and the stereo system the intrinsics, focal length, principal point, radial distortion coefficient and distortion, can be calculated together with the extrinsics (rotations and translations). Fig. 8 shows the images of multiple views of the checkerboard taken by the left camera for calibration. By using the calibration results, the image pairs from the stereo system can be rectified [6].

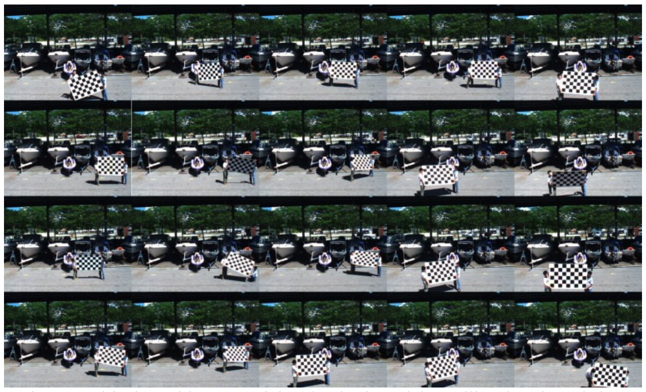


Fig. 8 Multi-viewed images of the checkerboard for calibration from left camera

B. Epipolar constraint and stereo correspondence

The epipolar line of the obstacle detected in the left image can be found in the right image as well. In fact, an epipolar line is the intersection of an epipolar plane with the image plane. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines [5]. Thus, a point in the left image generates a line in the right on which its corresponding point must lie. It is only necessary to search along the epipolar line for the obstacle correspondence. The epipolar constraint thus reduces the possible 2D search for correspondences to a 1D search along the epipolar line. Moreover, the system has a limited detection depth range so the search band is restricted along the epipolar line in the range of 20 meters to infinity. These constraints result in considerable computational savings and good correspondence accuracy.

We implement the normalized cross correlation template matching method [6] to find the stereo correspondence. It is a fast and relatively accurate matching method of which the similarity criterion is a measure of the correlation between windows in the two images. The bounding box of obstacles by monocular obstacle detection in the left image is considered as the template window while the search is conducted along its epipolar line in the right image. Since the left and right cameras are nearly configured frontal parallel and the obstacles are mostly in the distance, it is not necessary to check the rotations and scaling in the template match algorithm. If the length of constrained epipolar line is l pixels and the size of template window is $w \times h$ pixels, the size of I will be $(w+l) \times h$ pixels. The result map R of template matching is produced from the normalized cross correlation of the template window T and the image I.

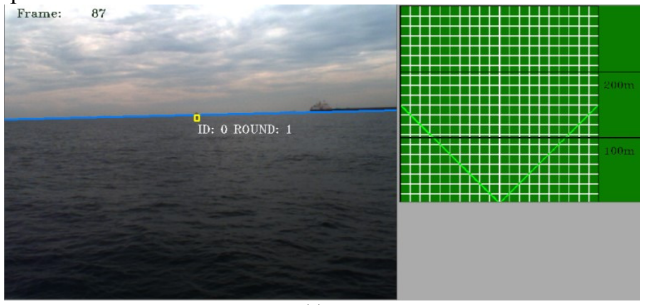
$$R(x,y) = \frac{\sum_{x',y'}[T(x',y') \cdot I(x+x',y+y')]}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

where $x' \in [0,(w+l)-1]$ and $y' \in [0,h-1]$.

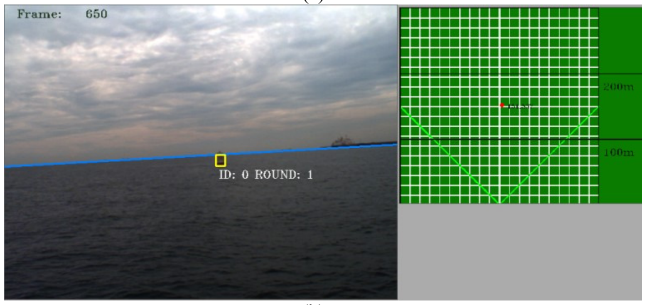
The obstacle location with sub-pixel accuracy is obtained by finding the maximum value in the result map of template matching with interpolation. Consequently, the correspondence can be done and then the 3D reconstruction.

4. OBSTACLE TRAJECTORY ESTIMATION BY KALMAN FILTER

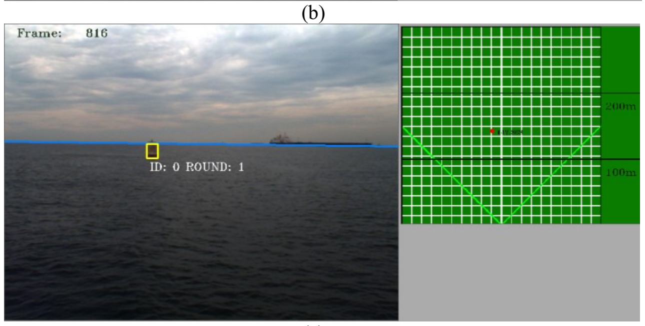
Before proceeding to the obstacle trajectory tracking, the 3D location of the obstacle is projected to 2D plane by rotating the left camera coordinate system to align with the sea surface. The sea surface normal can be derived from the sea-sky line since the line is regarded as the vanishing line of the sea surface in the image. An obstacle map which covers the filed in front of the USV has been created to demonstrate the obstacle location with respect to the USV. The range of the grid map is from 30m to 300m. In order to eliminate the detection errors and further improve the accuracy, a Kalman filter is applied in the system for the trajectory estimation. It enables us to check the current detection results by comparing with the estimation of the obstacle's position according to the past states. Fig. 9 shows that a buoy is detected in the field of view while the USV is headed for it. Fig. 10 shows a scenario of multiple obstacles at different speeds that all the obstacles are detected and located.



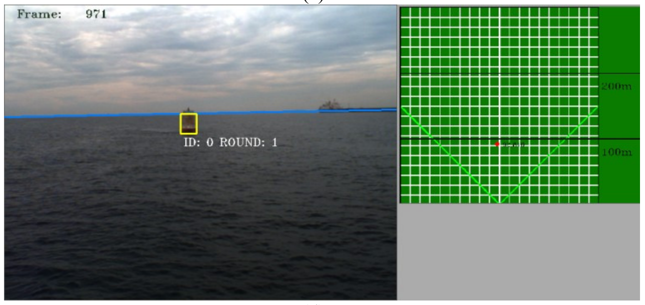
(a)



(b)

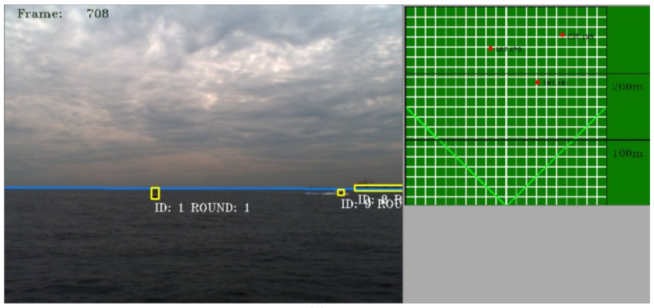


(c)

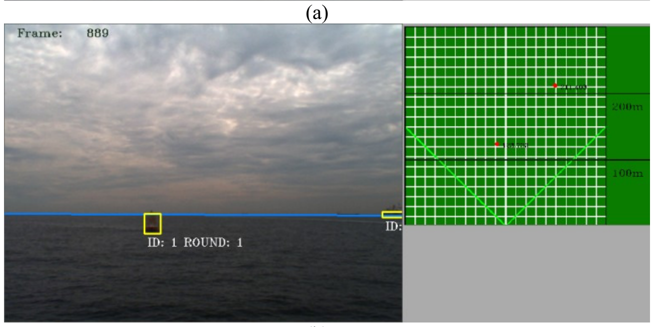


(d)

Fig. 9 A buoy detected while the USV approaching it. (a) At frame 87 the buoy is detected but not displayed in the grid map since it is outside of the upper limit of the obstacle detection depth. (b) At frame 650 the buoy is detected 150.5 meters away from the USV. (c) At frame 816 the buoy is 142.8 meters away. (d) At frame 971the buoy is 93.8 meters away.



(a)



(b)

Fig. 10 Multiple obstacles are detected. (a) At frame 708 there are 3 obstacles on the sea surface detected and located. (b) At frame 889 there are 2 detected obstacles.

5. CONCLUSION AND FUTURE WORK

This paper describes a vision-based obstacle detection system for unmanned surface vehicle. The system operates in real-time with images of 640×480 at about 12Hz. Field tests against the real scenes have been taken and shown quite stable and satisfactory results invariant to the target speed. The system can detect obstacles up to 300 meters away, although it is more accurate in the range from 30 to 100 meters. Furthermore, the obstacle detection is capable of handling the situation that the USV is moving at a high speed, up to 12 knots.

We have plans to integrate the vision-based obstacle detection system with a GPS receiver and digital compass to obtain the global position of the obstacle. The future work also includes improving the stereo correspondence method to extend the accurate detection range and the robustness of monocular obstacle detection to adapt to more complex scenarios.

REFERENCES

[1] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, *Frequency-tuned salient region detection*, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1597–1604, June 2009.

[2] C. Harris and M. Stephens, *A combined corner and edge detector*, In Alvey Vision Conference, pages 147-151, 1988.

[3] J. Y. Bouguet, *Pyramidal Implementation of the Lucas-Kanade Feature Tracker*, Tech. Rep., Intel Corporation, Microprocessor Research Labs, 1999.

[4] J. Y. Bouguet, *Camera Calibration Toolbox for Matlab*, http://www.vision.caltech.edu/bouguetj/calib_doc/.

[5] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision Second Edition*, Cambridge University Press, ISBN 9780521540513, March 2004.

[6] G. R. Bradski and A. Kaehler, *Learning OpenCV - Computer Vision with the OpenCV Library*, 1st ed. O'Reilly, October 2008.