A study on an integrated observation and collision avoiding support system for merchant ships

Doctoral Dissertation

A STUDY ON AN INTEGRATED OBSERVATION AND COLLISION AVOIDING SUPPORT SYSTEM FOR MERCHANT SHIPS

March 2012

Graduate School of Marine Science and Technology

Tokyo University of Marine Science and Technology

Doctoral Course of Applied Marine Environmental Studies

MINH DUC NGUYEN

Table of Contents

Abstract

Thank to the development of technology and the abundance of on board equipment, the modern ship officers now have the accessibility to a huge amount of information relating to movements of nearby targets as well as other environmental conditions. However, even with those advanced equipments, maneuvering has never been an easy task, especially when ships are navigating in congested waters. The overloaded information, if not appropriately presented and analyzed, may cause the ship officer to diffuse rather than help him mastering the situation. Thence, marine accidents are still occurring. Different researches have been carried out in ship collision avoiding support as well as tracking control. Those works however have been shown to be insufficient for practical application due to the following remaining problems:

- Most of them focus only on One-Ship to One-Ship encountering cases, using conventional DPCA/TCPA criterion for risk adjustment while in practice, we normally witness situations in which the actions must be taken to avoid collision with 2 or more target ships simultaneously.
- The Ship maneuvering model is not properly applied and therefore the route is actually not realizable.
- The rules of the road (traffic law) are not sufficiently taken into consideration.

Basing on the facilities available at TUMSAT (Radar, AIS traffic observation system) and the All-Time All-Weather floating object observation basing on Cameras (NMRI), this work is an effort to enable a safer, more favorable and efficient operation of the merchant ships by conducting a structured study of an integrated observation and collision avoiding support for the ship officers, especially in congested waters. The following 3 tasks are therefore to be solved in the study:

- Developing an automatic floating objects observing system using cameras.
- Proposing algorithms to generate collision avoiding route for the Own Ship in normal marine traffic situations at sea.
- Proposing an algorithm for collision avoiding strategy producing for the Ship in critical cases (i.e. extremely dangerous encountering situations in which the target ship takes wrong action while the distance between the 2 ships is small).

The target observation system basing on cameras is a supplement to the observation aids by Radar / AIS and has revealed its usefulness in detecting small floating objects around the ship position that do not appear on either Radar screen or AIS receiver display. With the aim of detecting and tracking floating objects continuously (including the determination of object position, speed and course), in this study, we first suggest an algorithm for the sea horizon detection by multi-frequency gradient variation analysis. Then, the algorithm for detecting floating objects on the camera images is proposed. The object position on the image is later converted to its equivalent position at sea. Then, Least Mean Square Method is applied to

determine object positions, speed and courses. Experiments have shown that the camera system is an efficient observing method that allows the detection and tracking of targets at short distance from the ship (around 2000m).

Utilizing the available target information which has been acquired by the above mentioned observing tools as well as other environmental constraints (manually input or extracted from ECDIS e.g.), various algorithms are proposed to generate a safe and economic collision avoiding route for the own ship in encounters normally faced at sea, including the algorithm basing on Dynamic Programming (DP), the on using Ant Colony Optimization (ACO) and that based on Bacteria Foraging Optimization Algorithm (BFOA). While producing route, different risk assessing criteria are applied for various navigation conditions. DP algorithm allows the route to be produced quickly, i.e. it requires minimum calculation effort among the 3 algorithms, even for situations in which the ship officer can hardly determine an appropriate collision avoiding strategy himself. However, DP is just an approximation method as it has treated the problem as time invariant while it is in fact varying with time. Another deterrence of the algorithm is that the application of traffic laws is complicated. To overcome these limitations, population based approximation searching algorithms have been applied and modified to fit our specified task of collision avoiding route producing and a suitable route cost evaluating function so as to take into account the traffic laws. The ACO algorithm was proposed with a local search (post-processing) and pheromone manipulating mechanism to provide better convergence property. The algorithm is able to generate a collision avoiding route close to the optimal one in a short period of time. With the choice of route cost function the generated route is more appropriate from the experienced seamanship view point. The algorithm enables the route producing for extremely difficult situations in which the DP algorithm fails. It will also be shown later that a limitation of the ACO algorithm is that its performance is strongly influenced by the choice of designing parameters. Then a BFOA is produced with a suitable swim length adapting algorithm. It will be shown from simulation studies that the proposed BFOA possesses all the positive points of the ACO algorithm. Additionally, it is better than ACO in searching speed and convergence property. Another advantage of BFOA is that it allows a flexible choice of designing parameters. Then BFOA is should be the method of choice for route producing.

Current researches on automatic ship controlling reveal also their shortages in providing the ship officer a recommended collision avoiding strategy in critical cases. Then, in this study, the game theory is used to analyze the collision avoiding problem in critical cases, taking into consideration the fact that the nature of encountering in these cases is more or less similar to a pursuit- evasion game in which the Own Ship is an evader, trying to avoid capture. T-K model is used for the Own Ship to make the produced collision avoiding strategy realizable. A BFO algorithm is then used to solve the arising optimization problem for the own ship collision avoiding strategy. The algorithm is later verified with computer simulations and the motions of targets navigating in Tokyo Bay.

Chapter 1 Introduction

1.1 Current State of Navigational Aids and Collision Avoiding Support Studies

Ensuring the safety and efficiency of navigation has always been a vitally important duty of the ship operators and traffic controlling officers as the marine accidents, if occurred, may result in not only loss of human lives and properties but catastrophic damages to the environment as well. Along with the rapid development of the shipping industry and the growing concerns about environment protection, the navigation safety has been gaining a lot more attention recently.

Apart from the human training, law enforcement and other factors, the tendencies of researches on the navigation safety can be classified into 2 categories:

- Studies on the observation supports for the ship officers and the communication links between ships as well as between ship and shore.
- Studies on the support in decision making for collision avoidance for the ship officer, especially in congested waters.

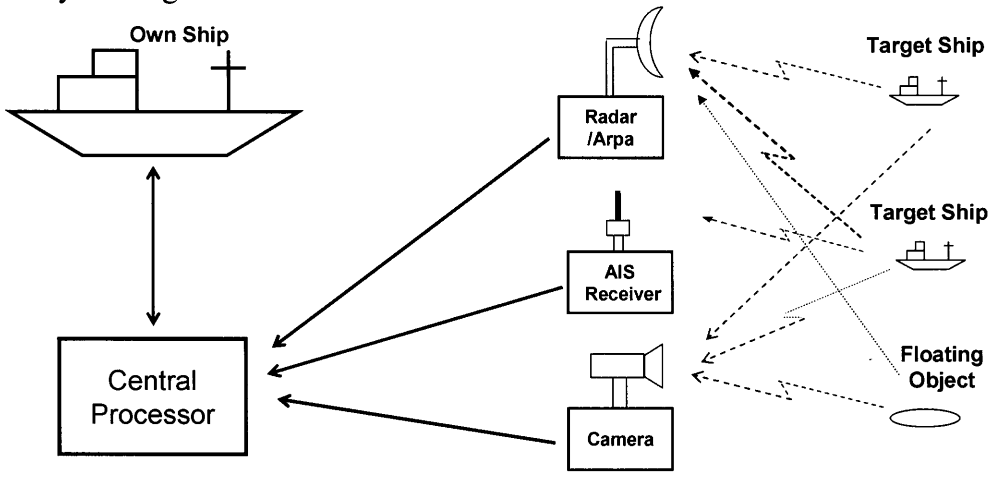


Fig. 1.1 Outline of observing system

On the Observation Support

Thank to the wide-spread application of modern technologies, Radar/ARPA systems and AIS receiver have become available onboard almost every merchant ship and have proven to be effective means of observation i.e. getting traffic information of the water around the ship. Additionally, sea surface observation by camera has been increasingly popular in the last decades. Different researches on sea objects detection by camera image analyzing have been published such as the work of M.U Selvi [6], M. Tello[5], F. Meyer[1] etc. These studies use images of cameras equipped on satellite or helicopter to detect ships and other discontinuities, e.g. oil spills on the sea surface. The works of S. Fefilatyev[9], etc. are based on images of camera installed on the coastal or sea buoy for ships detection.

Letting alone the detection capacity of the algorithms applied, a major shortage of all the above mentioned studies (from navigation ensuring aspect) is that they aim at neither enabling the observation at the ship position nor providing sea object information continuously and therefore have very little contribution to the insurance of the navigation safety, from the ship officers' view point, at least. Several other paper have also been published on the ship detection at sea from camera images like those of J. Liu, H. Wei[2] but their contributions are more or less theoretical and the practical application is obscured.

On the Collision Avoiding Support

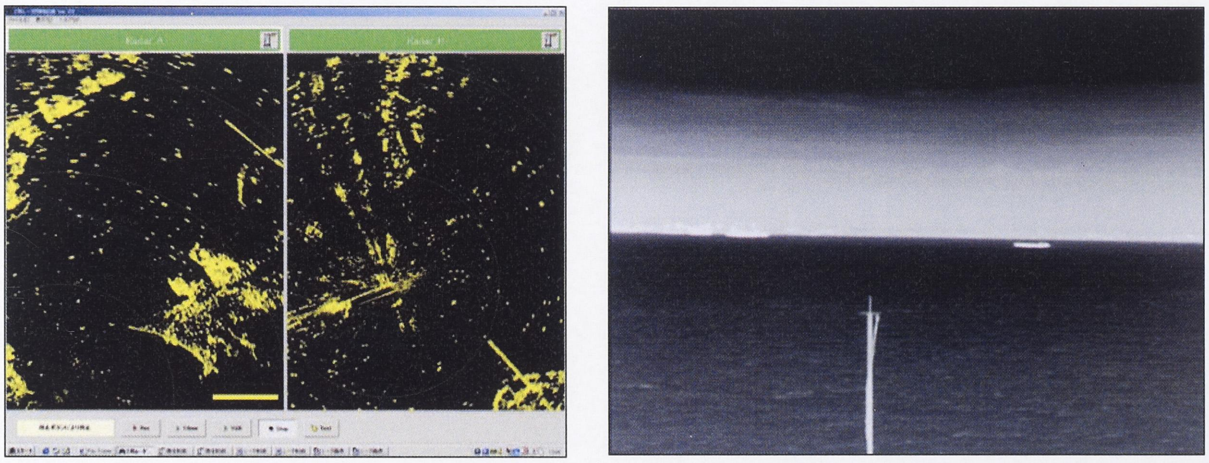
For collision avoiding support purpose, the more popular works that should be mentioned includes the work of W. Lang [11], N. Ward, S. Leighton [7], etc.

A shortage of the above studies is that they mostly deals with the cases in which the own ship has to take collision avoiding action against a single target while in practice, officer of the own ship often faces situations with several target ships involved. Additionally, the researches rely solely on traditional DCPA/TCPA risk assessment criterion that has been shown to be ineffective in many cases, especially in congested waters.

In their study, R. Smierzchalski et al. [8], V.H Tran et al. [10] did mention the collision avoiding strategy for the own ship in multi target ship cases. However, the ship dynamics is not included in the algorithms and the collision avoiding route is therefore hard to realize. Another deterrence of these works is that the marine traffic rules have not been properly taken into consideration while producing collision avoiding route.

The overview of the modern navigation support system is illustrated in Fig.1.1 where the observing means are used to acquire information about the motions of nearby ships and floating objects. Then the central processor is to analyze the obtained information and seek a strategy for the ship to avoid collision. The strategy is later used to control the ship so that it will pass all the dangers on an appropriate route, given the actual traffic conditions.

1.2 Study Purposes



```
2006-03-28 23:14:39 !AIVDM,1,1,,B,19NS0Fh000:1F?hDG4KIOmC>0D1u,0*5E
2006-03-28 23:14:40 !AIVDM,1,1,,A,15AAA20002b05pND@QQL`oC>05hp,0*2D
2006-03-28 23:14:40 !AIVDM,1,1,,A,A04757QAv0agH2Jd;Vp`1Or3sRT6wdCdKQsvs>pN,0*0B
2006-03-28 23:14:40 !AIVDM,1,1,,A,369ffh50019wvldDC0NWP`3@0000,0*2B
2006-03-28 23:14:40 !AIVDM,1,1,,A,33:gEp1001b05J0D@vmB0J?>0000,0*75
```

Fig. 1.2 Traffic Observing Tools

In Tokyo University of Marine Science and Technology, a marine traffic observation system has been established to supervise the marine traffic inside Tokyo bay with several radar stations and AIS transponders. Furthermore, under the sponsorship of NTT Communication Corporation and Japan Oil, Gas and Metals National Corporation, a research project has been conducted at National Maritime Research Institute (NMRI) on an All Time All Weather Floating Object Detection System Using Cameras.

Basing on these available facilities, the subject of this study is chosen in an effort to enable a safer, more favorable and efficient operation of the merchant ships.

Noticing the burden that the ship officer has to bear to ensure the safety of the own ship and the shortage of studies on the collision avoiding support means so far available, the focus of this work is on a structured study of an integrated observation and collision avoiding support for the ship officers, especially in congested waters.

Then, the study aims at solving the following individual component parts of the supporting system as followings:

- Developing a target ships/floating objects observing system using camera. The system must be able to detect sea objects, determine object positions and track the objects (calculating the object moving speed and course). It is a supplement to the observation aids by Radar / AIS (which was the subject of my Master Thesis) and must be independent from these observing means. These tasks should be solved without human intervention to make the system helpful to the ship officer.
- Utilizing the available target information (received by the above mentioned observing tools) as well as other environmental constraints (manually input or extracted from ECDIS e.g.) to generate a safe and economic collision-avoiding route for the own ship in all types of encounters normally faced at sea. For this purpose, various algorithms will be proposed and analyzed in the following chapters. The route produced should meet marine traffic law as far as possible to eliminate the possibility of conflicting actions among ships in collision avoiding. Additionally, the dynamic model of the own ship should be used to make the route realizable.
- Providing the officer with a collision-avoiding strategy in critical cases (i.e. extreme dangerous cases in which the target ship is close to the own ship, its intention is not understandable and it is moving in a collision course).

These problems, if properly solved, would pave the way for much more favorable condition of merchant ship operation in which the computing capacity of the computer is exploited to reduce the work load of the navigator, to eliminate the possibility of human error in judgments and decisions making. The utmost achievement, as mentioned earlier, would be a tiny contribution to a safer, greener and more economic shipping industry.

1.3 Dissertation Outline

Solving the above mentioned tasks step be step, the dissertation will be arranged in the following order:

Chapter 2 Floating Objects Observation and Tracking by a Camera System: The chapter deals with the development of a system for floating object detection and tracking basing on a camera system including an Infra-Red camera, a Night Vision camera and a Laser Camera (Lidar). Firstly, an algorithm for the sea horizon detecting will be introduced. Then, an algorithm is proposed for detecting floating-objects from camera images. The object motions can be deduced from a sequence of images. Later, the tracking accuracy is tested by actual sea experiments.

Chapter 3 Automatic Collision Avoiding Support System and Optimal Route Generation by Dynamic Programming: The chapter gives the overview of an automatic system for generating the collision-avoiding route and analyzes the inputs necessary for route-producing. Different risk-assessing criteria will be introduced for various navigation conditions. In this chapter, the Dynamic Programming Algorithm will be used to determine collision-avoiding optimal route. The advantages and disadvantages of the algorithm and the type of route it produces will be thoroughly studied.

Chapter 4 Collision-Avoiding Route Generation by Ant Colony Optimization: Also targeting at producing an optimal collision avoiding route for the ship, given the encounter case and accompanying environmental constraints, the chapter will propose an Ant Colony Optimization algorithm to find the route. It will be shown in the chapter that by applying a suitable route cost function (as composed then) the rules of the road can be properly satisfied while figuring out the collision-avoiding strategy. Pros and Cons of the ACO algorithm will be analyzed in details.

Chapter 5 Collision-Avoiding Route Generation by Adaptive Bacterial Foraging Optimization Algorithm: Noticing the disadvantages of the DP algorithm and ACO algorithm for route-producing, this chapter is to propose a route producing algorithm imitating foraging behavior of a population of E.Coli bacteria. The bacteria foraging phenomenon will be introduced first. Then, an Adaptive-BFOA specified for the purpose will be suggested. It will be shown later that the algorithm is more efficient than both the algorithms proposed earlier.

Chapter 6 Collision Avoiding Strategy in Critical Cases by Games Theory: Current researches on automatic ship controlling reveal their shortages in providing the ship officer a recommended collision-avoiding strategy in critical cases (which will be defined later). Then, this chapter treats the collision-avoiding problem in critical cases as a game, using Game Theory (Pursuit-Evasion game specifically). An Adaptive-BFO algorithm will be proposed to solve the arising optimization problem. The algorithm will later be verified with computer simulations.

Chapter 7 Conclusion and Future Study: Summarizing results of the study and mentioning subjects for later study.

References

1. F. Meyer, S. Hinz, "Automatic Ship Detection in Space-Borne SAR imagery", online, available at http://www.isprs.org/proceedings/XXXVIII/1_4_7-W5/paper/Meyer-187.pdf
2. J. Liu, H. Wei et al., "An FLIR Video Surveillance System to Avoid Bridge-Ship Collision", Proceedings of the World Congress on Engineering 2008, Vol I, 2008
3. J. Wu, "Development of ship-bridge collision analysis," Journal of Guangdong Communication Polytechinic, Vol. 4, pp.60-64, 2004
4. L. Hao, Z. Minhui, "A Novel Ship Wake Detection Method of SAR Images Based on Frequency Domain", Journal Of Electronics, Vol. 20 No. 4, pp. 313-321, 2003
5. M. Tello, L.M. Carlos, J.M. Jordi, "A novel algorithm for ship detection in SAR imagery based on the wavelet transform", IEEE Geoscience and Remote Sensing Letters, Vol. 2, No. 2, 2005
6. M. U. Selvi, S. S. Kumar, "A Novel Approach for Ship Recognition using Shape and Texture", International Journal of Advanced Information Technology, Vol. 1, pp. 23 - 29, 2011

7. N. Ward, S. Leighton, "Collision Avoidance in the e-Navigation Environment", available at "http://www.gla-rrnav.org/pdfs/ca_in_enav_paper_iala_2010.pdf"
8. R. Smierzchalski, "Evolutionary algorithm in problem of avoidance collision at sea", Proceedings of the 9th International Conference, Advanced Computer Systems,2002
9. S. Fefilatyev, D. Goldgof and C. Lembke, "Tracking Ships from Fast Moving Camera through Image Registration", Pattern Recognition (ICPR), pp.3500-3503, 2010
10. V. H. Tran, H. Hagiwara, H. Tamaru, K. Ohtsu, R. Shoji, "Strategic Collision Avoidance Based on Planned Route and Navigational Information Transmitted by AIS", The Journal of Japan Institute of Navigation, 2005
11. W. Lang, "Ship collision avoidance route planning tool", available at "http://www.bairdmaritime.com/index.php?option=com_content&view=article&id=5288:ship-collision-avoidance-route-planning-tool-&catid=78&Itemid=75"

Chapter 2 Floating Objects Observation and Tracking by Camera System

2.1 Introduction

Floating object detecting and tracking has always been an important task not only for ensuring marine traffic safety but for search and rescue missions as well. Together with the technology advancement, different techniques (e.g. Radar, AIS) are available onboard modern merchant vessels for this purpose. Each observing method has its own advantages and disadvantages and is therefore applied in its appropriate fields. AIS data, for example, is rather accurate and convenient for data analysis but the installation of AIS is not compulsory for small vessels such as vessels less than 500 GT, fishing and pleasure boats. Radar is a much more efficient onboard observation method. However, different object surfaces present diversified reflection properties for Radar signal. Then, for various reasons, many objects, especially those small and not protruding high above the sea level can not be detected by the ship Radar.

The observation by camera is achieving a lot of attention recently. Needless to say, the camera images are perfectly favorable for human eyes. Several works have been published on the automatic detection of target from camera static images or videos.

In an effort to support real-time observation at scene, especially for small objects that are otherwise not detectable by the ship Radar, a hybrid observation system basing on cameras has been developed at the National Maritime Research Institute (NMRI) [4][6]. The system consists of a Laser Camera (Lidar), a Night-vision Camera and an Infra-red (IR) Camera. These 3 different cameras are situated in a camera-box located on a stabilizer.

This study is a part of the observation-system developing project that deals mainly with the object-tracking and watch-keeping tasks. Using the collected images (mostly the IR camera images), the study aims at developing a program for estimating the floating-object track to support observation and provide warnings. For this purpose, the object-tracking program must be able to solve the following tasks simultaneously:

- Collecting IR camera images and detecting floating-objects from the images.
- Transferring the object positions from the image-coordinate system to the ship-coordinate system and then to equivalent positions on the sea surface.
- Predicting the object moving track from its consecutive positions which have been extracted from camera images.
- Providing warnings if the tracked floating-object is entering a Guard Area.

The tracking program is installed in a computer connected to the cameras as well as other components of the observing system.

In this chapter, the observing-system outline and the object-tracking program outline, together with algorithms for coordinates transferring will be mentioned in section 2.2. The algorithms for sea-horizon line detection and floating-object detection will be described in sections 2.3 and 2.4 respectively. Then section 2.5 is to discuss the object-tracking and object motion-fitting problem. In section 2.6, the target-tracking errors will be illustrated with some onboard-experiment data. To increase the system flexibility, a manual tracking method using NV Camera or Lidar Camera images will be proposed in section 2.7. Lastly, the chapter conclusions will be summarized in section 2.8.

2.2 System Overview and Coordinates-Transforming Algorithms

2.2.1 System Overview

At the core of the system (called All-Time All-Weather Floating Object Observing System) are three cameras to function in various sea and weather conditions.

NV camera provides continuous color images of the sea area around Own Ship (OS) position in both day and night time. The camera zoom (focus) can be adjusted to provide close range pictures of the sea objects. A disadvantage of NV camera is that the image quality is heavily affected by noise and objects at larger distance are not clear, especially in night time.

IR camera detects objects from the temperature discrepancy between the objects and sea water or air temperature. It can therefore be effective in all conditions as far as the object is hotter (or cooler) than its surrounding environment.

The use of Lidar Camera is more complicated. Though, the proper choice of parameters of the generated pulse can produce object reflection even for quite small objects on the images. It is a supplement to the above 2 cameras for the cases where a small and cold object needs detecting.

Those 3 cameras are situated inside a box on a stabilizer. This stabilizer has the function of maintaining camera box in a horizontal plane while the ship, on which the system is installed, is fluctuating with 6 degrees of freedom. The stabilizer is automatically controlled by a computer and can be rotated around the ship heading. To do this, the controlling-computer has to use pan data from an external gyro. The camera attitude can also be manipulated manually to follow targets. This enables the system to provide real-time images of the sea surface around the ship (or camera) position.

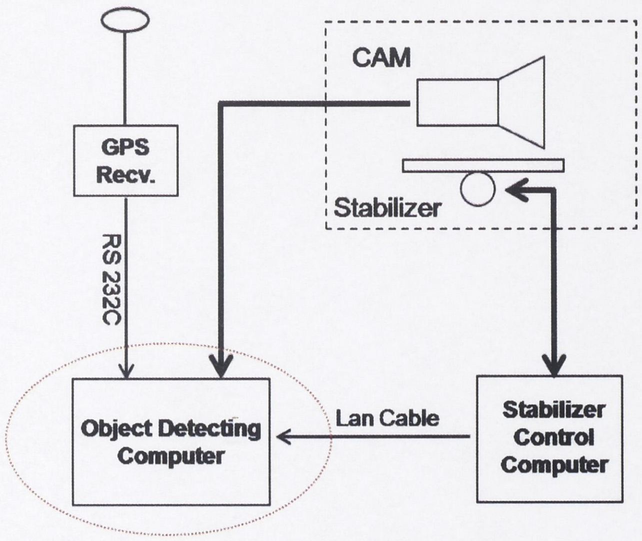


Fig. 2.1 Camera Observing System Overview

To get the camera position, communication link is established between the system and a GPS receiver. Using Novatel OPAC 3 GPS receiver, camera position is highly accurate.

Data sharing between Stabilizer-Control-Computer and Object-Detecting-Computer is conducted through a local network cable. This allows detecting program to access to camera attitude as well as ship attitude data. Cameras are connected to the latter computer by coaxial cable for high speed data transferring. The rest of the chapter will focus on the object detecting program installed on the Object-Detecting Computer (Fig. 2.1).

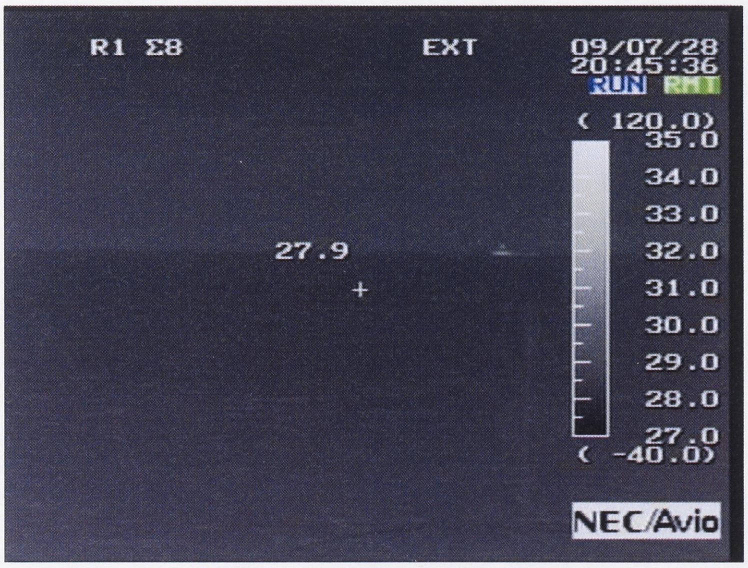


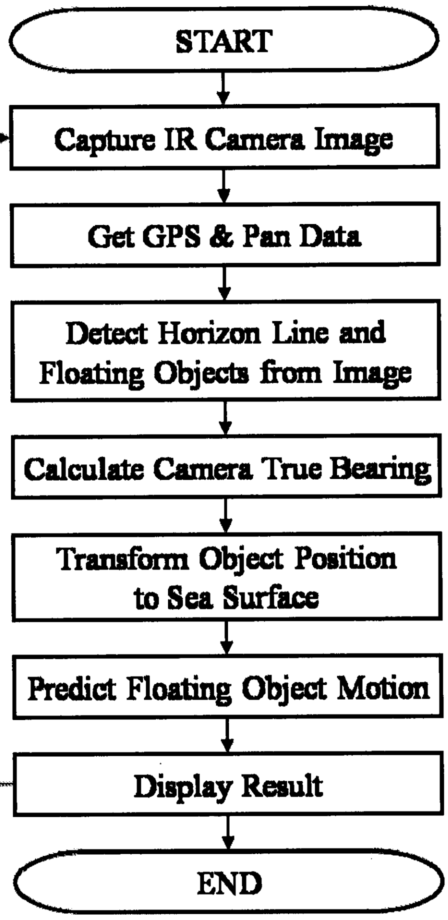
Fig. 2.2 IR Sample Image and Specification

2.2.2 Object Tracking Program Outline

In this study, the automatic object detecting algorithm is designed solely for IR camera images. An example IR image is shown in Fig. 2.2, with a temperature mapping scale to its equivalent brightness of pixels on the image. Target Ship (TS), with its engine or generators in operation is a strong heat radiation source and therefore clearly visible on the image.

The program outline is described by the flow chart in Fig. 2.3. Going down the flow chart, IR camera images are acquired periodically by using an image capturing-board (matrox). Capturing interval can be decided by the user. In the study, the interval is set to be 2 to 5 [sec], taking into account the existing-interval of waves and movement of floating-objects. Field of view of the IR Camera is 21.7° horizontally by 16.4° vertically. It uses 8-13μm wavelength, with minimum detectable temperature-difference of 0.08°C, and produces 640 by 480 pixels images (see Ref. [4] for more details).

Then, the OS position and course are extracted from the GPS receiver logs, using the established RS-232C serial communication link. The pan data, which is necessary for determining camera direction, is acquired from the Stabilizer-Control computer through a local network cable. The dataset contains ship roll, pitch (ship attitude) and stabilizer roll, pitch and yaw angles which must be used later to determine camera attitude.


START
Capture IR Camera Image
Get GPS & Pan Data
Detect Horizon Line and Floating Objects from Image
Calculate Camera True Bearing
Transform Object Position to Sea Surface
Predict Floating Object Motion
Display Result
END
Fig. 2.3 Detection Program Outline

Next, the sea-horizon line is searched and floating-objects are detected from the image. These are the major tasks of the program and will be discussed in later sections.

As mentioned above, to convert the object positions from the image to the sea surface (i.e. earth-fixed) coordinate system, camera-bearing must be known. In this step, OS direction and camera pan data collected in the previous steps are used for the calculation. The transforming algorithm will be discussed in more details in the next section.

In the following steps, object tracks are predicted from its consecutive positions and the result is to be displayed to the user.

The process jumps up to the 1st step to collect sequential images. The program thereby follows floating-objects continuously as required.

2.2.3 Coordinates Transforming Algorithms

This section deals with the conversion of the position of an object at sea, as seen on the camera image to its relative position to the camera position. For this coordinates conversion, the ship yaw, pitch, and roll angles and stabilizer yaw, pitch and roll angles must be used. These data, as mentioned earlier, are mobilized from stabilizer-control computer through a local network cable.

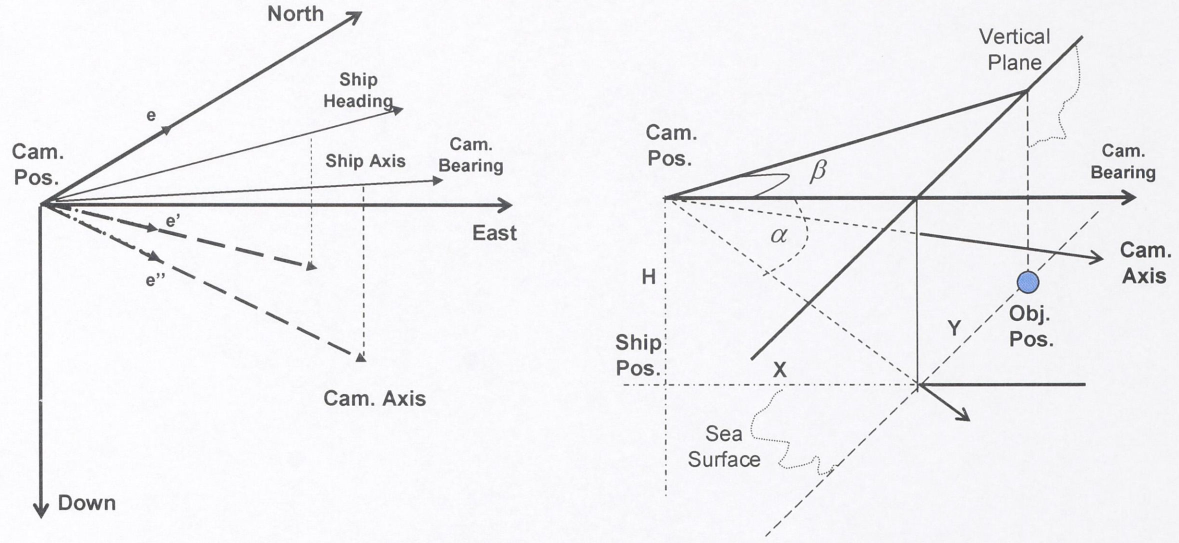


Fig. 2.4 NED and Camera Fixed Coordinate Systems

Assume that we have a unit vector e pointing north in a North-East-Down (NED) coordinate system originated at the current position of the ship. An equivalent unit vector e' on the ship longitudinal axis is the result of rotating e through the ship yaw, pitch and roll angles sequentially. Then,

$$\begin{aligned} &e = [1\ \ 0\ \ 0] \\ &e' = [e'_n\ \ e'_e\ \ e'_d] = R_{Ship} \times e \\ &where\ R_{Ship} = R_z^{Ship-yaw} \times R_y^{Ship-pitch} \times R_x^{Ship-roll} \end{aligned} \tag{2.1}$$

In the same manner, an equivalent unit vector e'' on the camera axis can be derived by rotating e' around axe of ship body fixed coordinate system by the angles equivalent to the stabilizer yaw, pitch and roll respectively.

$$\begin{aligned} &e'' = [e''_n\ \ e''_e\ \ e''_d] = R_{Stabilizer} \times e' = R_{Ship} \times R_{Stabilizer} \times e \\ &where\ R_{Stabilizer} = R_z^{Stabilizer-yaw} \times R_y^{Stabilizer-pitch} \times R_x^{Stabilizer-roll} \end{aligned} \tag{2.2}$$

R$_{\text{Ship}}$, R$_{\text{Stablizer}}$ are called rotation matrices and can be calculated from the conventional rotation matrices around the z-axis (R$_z$), y-axis (R$_y$), and x-axis (R$_x$) in order. In these calculations, the x-axis of the ship-body coordinate system is defined as its longitudinal axis and the x-axis of camera-fixed coordinate system is the camera lens axis. Those matrices are determined with their respective rotation angles as the followings:

$$R_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \quad R_{y,\theta} = \begin{bmatrix} coa\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad R_{z,\psi} = \begin{bmatrix} coa\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Camera axis bearing is then calculated by

$$Cam\_Bearing = \arctan(e''_e / e''_n) \quad (2.4)$$

Then, from Fig. 2.4, the object position relative to that of the camera can be deduced by:

Object relative Position: $[X \ \ Y]$

$$X = H/\tan(\alpha) \quad (2.5)$$
$$Y = X \times \tan(\beta)$$

where

X, Y: longitudinal and traverse distance, relative to camera position and bearing.

H: Camera height from the sea surface.

α: vertical angle between true horizon direction and the object line of sight

β: horizontal angle between camera axis and the object line of sight.

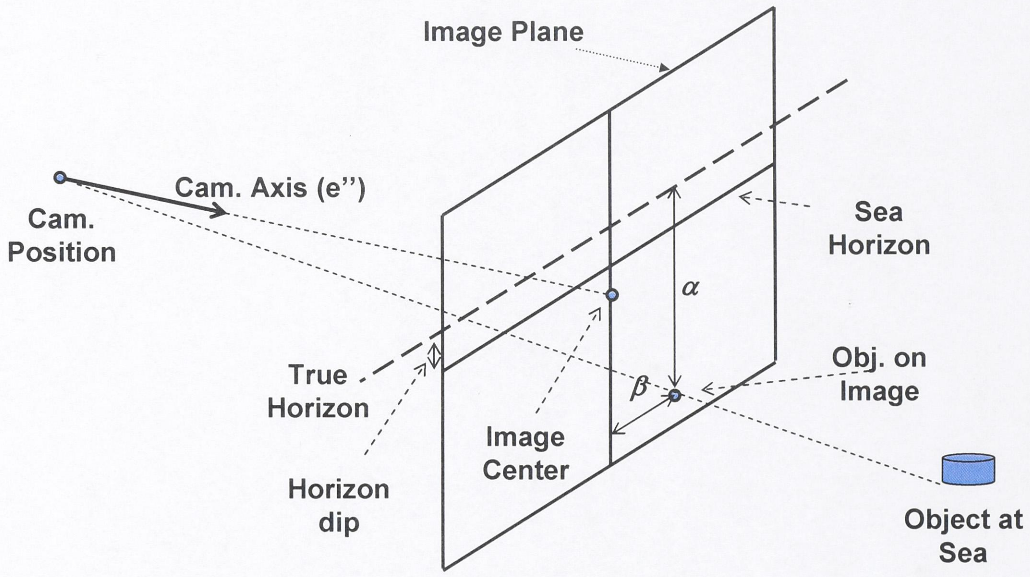


Fig. 2.5 Object Position at Sea and on Image Relation

The relation between an object position at sea level and its position on the image plane is denoted in Fig. 2.5. To make it more understandable, the 2 angles (α and β) have been used to replace 2 parameters t and v (Fig. 2.6) on the images which are needed to determine them, given the opening angles of camera lens.

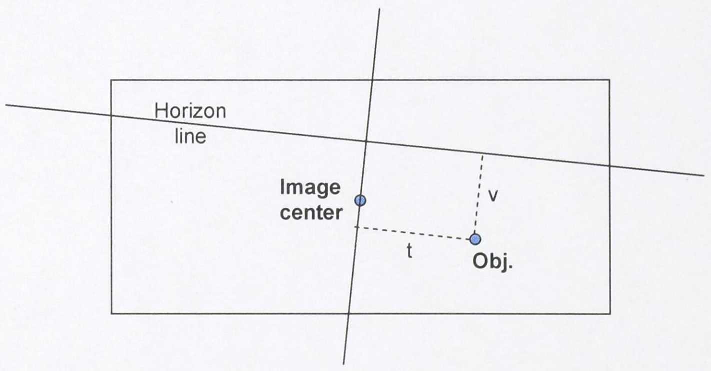


Fig. 2.6 Object Position on Image

The true-horizon line is, however an imaginary line and does not appear on the image. Thus, it is necessary to determine this line indirectly from the sea-horizon line where the term refers to a line separating the sea-water and the sky above it. These two horizons are separated by an angle called the horizon-dip in celestial navigation, which in turn can be calculated approximately from the camera height above the sea water.

Knowing the position of the object on the image and the true horizon line, α and β can be determined as followings (Fig.2.6)

$$\begin{aligned}
&\alpha[\deg] = v[pixel] \times Rat \\
&\beta[\deg] = t[pixel] \times Rat \\
&\quad where \qquad\qquad\qquad\qquad (2.6) \\
&\quad Rat = \frac{21.7[\deg] + 16.4[\deg]}{640[pixel] + 480[pixel]} \\
&\quad \textit{Rat is the coefficient for converting image pixel unit to degree unit}
\end{aligned}$$

Then, the object position in an earth-fixed coordinate system can be calculated from the camera position in the same system, by

$$\begin{aligned}
&N_{Obj} = N_{Ship} + X\cos(CB) + Y\cos(CB+90) \\
&E_{Obj} = E_{Ship} + X\sin(CB) + Y\sin(CB+90) \\
&where \qquad\qquad\qquad\qquad (2.7) \\
&\quad N_{Ship}, E_{Ship}: \textit{Own Ship (Camera) position in an earth fixed coordinate system.} \\
&\quad N_{Obj}, E_{Obj}: \textit{Object position in the same system} \\
&\quad CB: \textit{Camera true bearing}
\end{aligned}$$

It can be seen from (2.5) that the object relative-position (X, Y) is most sensitive to error in α. Therefore, it is important to determine this parameter (or the horizon-line position, equivalently) accurately. Theoretically, the line can be directly specified from the attitude of camera axis by, e.g.

$$Cam\_Pitch = \arctan(e''_d / \sqrt{e''^2_e + e''^2_n})$$

However, due to the lateral error of the gyro measurements, the above calculation is not reliable. Therefore the true-horizon line is to be specified from the sea-horizon line which appears on the images. The detecting algorithm will be discussed in the following section.

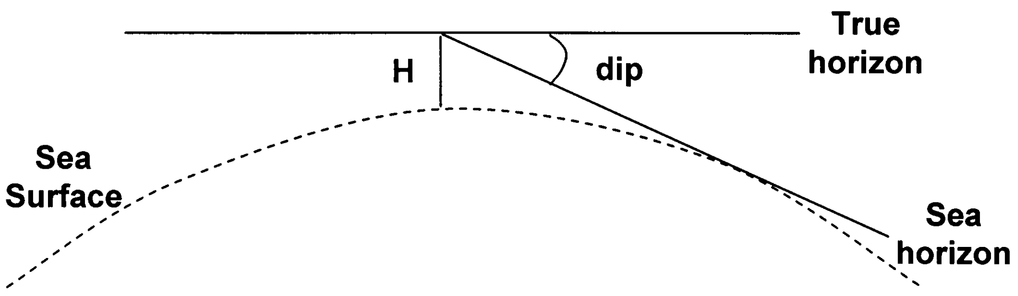


Fig. 2.7 True-Horizon and Sea-Horizon

2.3 Sea Horizon Line Detection

As stated above, detecting the sea-horizon is an important task to ensure the accuracy of object-position calculation. The sea-horizon is, in fact, an edge separating the sea water and the sky above. Then, naturally an edge-detection technique like those proposed in [1][5][8] etc. should be applied for this purpose.

Edge-detection aims at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The techniques may be grouped into two categories, search-based and zero-crossing-based techniques. Search-based methods detect edges by computing a first-order derivative expression (gradient magnitude) and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge. The zero-crossing based methods search for zero-crossings in a second-order derivative expression (Laplacian e.g.) computed for the image to find edges.

Zero-crossing based methods are more sensitive to disturbances than search-based ones. A technique similar to the latter therefore is used in this study.

2.3.1 Gradient Expression

The core of any edge-detection algorithms is the calculation of an expression of image brightness gradient. Thank to the stabilizing function of the stabilizer, the sea-horizon does not deviate largely from the image horizontal direction or the direction of image row. As a result, the horizontal component GX contributes much larger part to the gradient value. Conversely, the vertical component GY of the gradient is small on the sea-horizon line. Therefore, it is reasonable to take just horizontal gradient-expression into consideration so that the vertical edges produced by waves or floating objects are ignored in gradient calculation.

Among search-based edge-detection methods, the most popular one is probably the Canny method [8] using a Sobel operator. Sobel proposed 3x3 kernel matrices for vertical and horizontal gradient components respectively. The kernel matrix for horizontal gradient is given in (2.8) and used to convolve over the input image to produce gradient.

$$G_X = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad IMageGradX = IMage * G_X \qquad (2.8)$$

** denotes the convolution operation*

The convolving operator is simply the shift of the kernel matrix through the image and multiplying its components with the brightness of the corresponding image pixel underneath.

Due to the textural characteristic of IR sea-surface image, the Sobel and other common operators have poor performance. Noticing that the frequency of brightness gradient is incoherent for waves and other disturbances but coherent for the sea-horizon, the author expresses the image horizontal gradient by convolving the image with the following operators

$$
\begin{aligned}
G_x^1 &= [0\ 0\ 0\ 0\ 0\ 0\ 1\ -1\ 0\ 0\ 0\ 0\ 0\ 0]^T \\
G_x^2 &= \frac{1}{3}[0\ 0\ 0\ 0\ 1\ 1\ 1\ -1\ -1\ -1\ 0\ 0\ 0\ 0]^T \\
G_x^3 &= \frac{1}{5}[0\ 0\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ -1\ 0\ 0]^T \\
G_x^4 &= \frac{1}{7}[1\ 1\ 1\ 1\ 1\ 1\ 1\ -1\ -1\ -1\ -1\ -1\ -1\ -1]^T \quad (2.9) \\
Grad_i &= Image * G_x^i \qquad i = 1 to 4 \\
Grad &= \prod_{i=1}^{4}(Grad_i) \\
&* : convolution\ operation
\end{aligned}
$$

The convolution with these 4 kernel matrices (G_X^{1-4}) gives 4 gradient components ($Grad_{1-4}$) respectively. These 4 components denote brightness variation at different frequencies, from high ($Grad_1$) to low ($Grad_4$). Then, the total gradient-expression is defined as the product of these 4 gradient components, expecting that the total gradient value will show a significant discrepancy between the sea-horizon and other disturbing edges.

To represent the advantage of the technique proposed, comparisons are shown below between the gradient-expression calculated by this technique and the one determined by Sobel operator with convolving matrix GX in (2.8) for 2 images captured by IR camera in different weather conditions at sea.

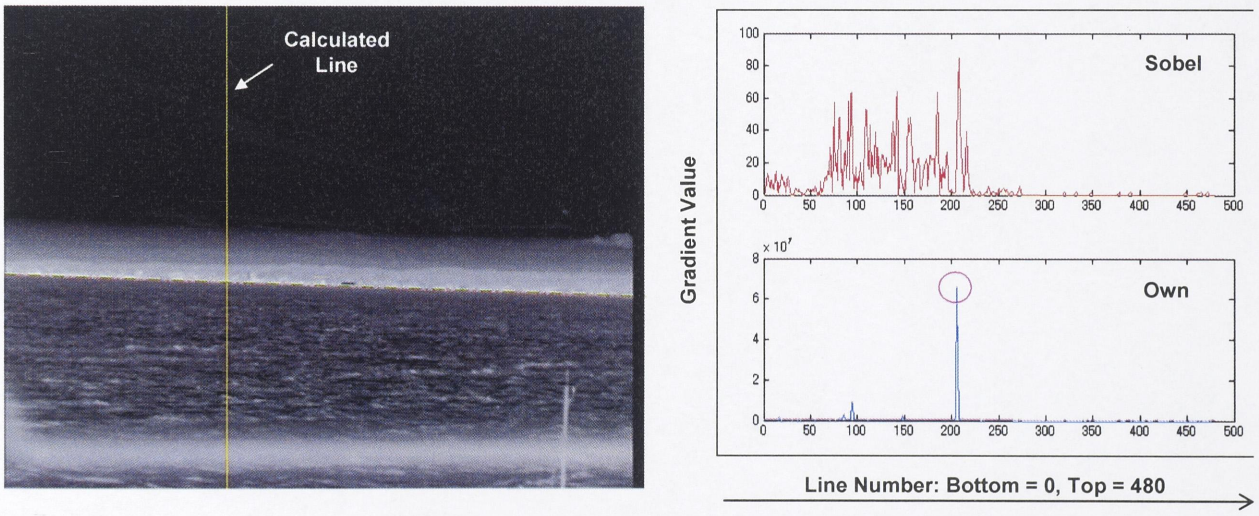


Fig. 2.8 Sea-Horizon Line Detection – Gradient Expression

In Fig. 2.8, water and air temperature difference is quite significant, resulting in a clear edge between the sea-water and the sky above. The gradient-expressions of a vertical line which is marked on the image calculated by the two methods are shown in the graphs on the right. It can be seen that the lower figure provides a clear and significant peak for the point on the sea-horizon. This peak is well over other disturbance edges and is easily detectable. With Sobel operator for this condition in specification, the gradient peak value for the sea-horizon is, though detectable, not very different from peaks of wave-edges. Therefore, it is difficult to set a threshold value for separating the wanted edge and noises.

The difference between the 2 approaches is even more obvious in the second example (Fig. 2.9). In this case, image was captured when the temperature difference between the sea-water and air is small, resulting in a vague sea-horizon line, if it is detectable.

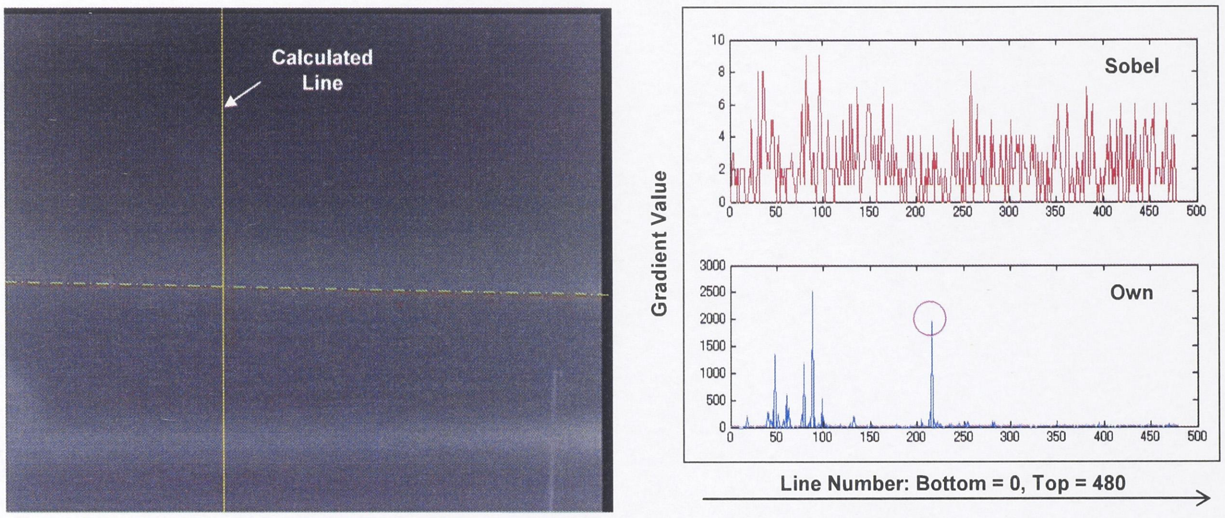


Fig. 2.9 Sea-Horizon Line Detection – Gradient Expression

In this example, it is almost impossible to recognize the horizon-line peak if Sobel operator is applied (Fig. 2.9, right upper graph). It is due the image nature that produces unwanted horizontal lines for which the edges are even more significant than the sea horizon edge when temperature difference is too low. On the other hand, using our proposed operator, gradient-expression graph still shows a significant peak for the sea-horizon line, as the result of coherent brightness variation at different frequencies.

These 2 images were captured in Jan. 2010. The average sea water temperature for this month was 15°C. The weather was clear for Fig.2.8 (27th, around 16:00), with air temperature (at sea level) of 11.3°C. On 28th, at around 10:00 am (Fig.2.9), it was rainy (0.5mm) and temperature was from 15.1 to 15.2°C. (See [3])

In addition to the temperature difference, quality of IR camera images depends on a variety of other conditions such as cloud conditions, humidity etc. which are hard to clearly determine. Therefore, weather condition is not further mentioned in this study (refer to [4] for more information in the performance of IR camera on different conditions).

2.3.2 Sea Horizon Line Detecting Procedure

Applying the proposed gradient-expression, the procedure for detecting sea-horizon is performed through steps shown in the flow chart in Fig. 2.10.

After gradient calculation, an edge thinning process is to be applied. The aim of this process is to remove the unwanted gradient values at pixels around the edge pixel. In our method of expressing gradient, it is easily seen that not only the edge pixel but also several pixels under or above that pixel, on the same vertical line do have significant gradient values. After edge thinning step (which is denoted as or non-maximal edge suppressing step in Fig. 2.10), only pixels on the true edge still possesses a significant value.

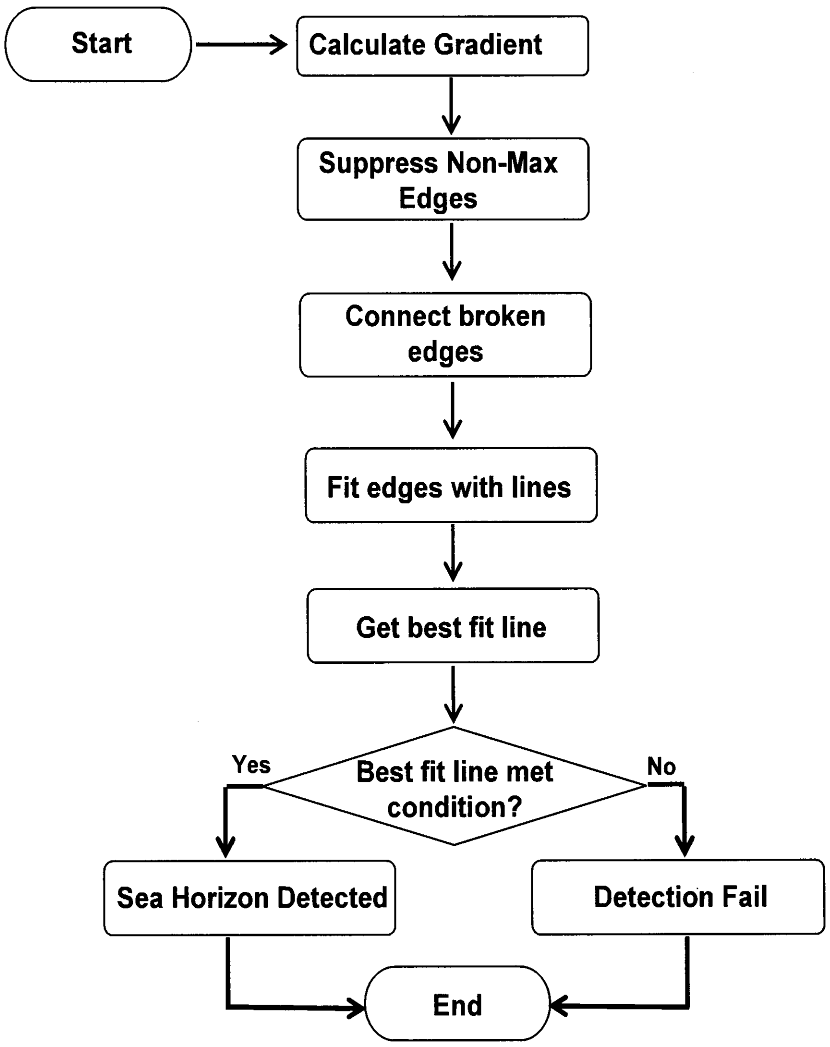


Fig. 2.10 Sea-Horizon Detecting Algorithm

Then, a threshold is selected to remove all non-significant edges. The aim of this step is to remove disturbance edges which usually have small gradient-values and therefore to eliminate the possibility of false detection of the sea-horizon. It is difficult to decide a single gradient Threshold-Value for removing non-significant edges in all weather conditions. Therefore, in this study, an adaptive scheme is proposed for selecting this value in which the Threshold-Value of gradient-expression on a vertical line is decided as followings, basing on actual gradient-values of all pixels on that line.

$$T_{upp} = \min\left(\frac{Max\ Gradient}{n}, T_0\right) \qquad (2.10)$$

$$where: \ Max\_Gradient \ is \ the \ \max imum \ gradient \ value \ on \ the \ line$$

$$n = 4$$

$$T_0 = 6^4$$

Due to noise and wave effects, edges may be corrupted (i.e. broken). This causes discontinuities of a long edge. To solve the problem, broken edge-parts nearby and of similar tendency should be connected to reconstruct the original edges. These edges are the sea-horizon line candidates. In this study, the relaxing threshold method is used for the edge connecting procedure with the notice that the sea-horizon edge should be a straight line. Its principle is illustrated in Fig. 2.11.

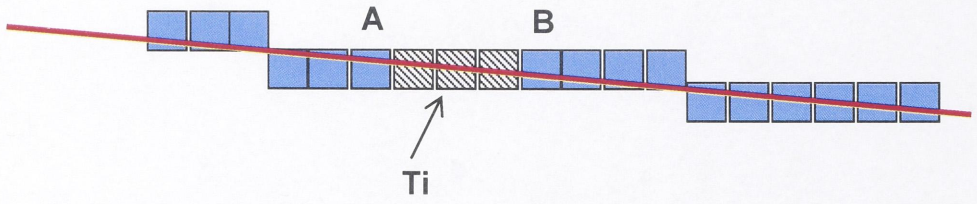


Fig. 2.11 Edges Connecting Principle

```
if $(T_{upp} > Ti > T_{low})$ And $(AB <= 3[pixel])$ then
    Connect $(AB)$
end if
```

After connecting, the edges are fitted by lines, knowing the fact that with the camera height of 10 – 15 [m], the sea-horizon line is very close to a straight line.

From those fitting lines, the best fit line is selected. This is the line with maximum edge points on it and does not deviate largely from the estimated position of the sea-horizon. The estimated position can be inferred from the ship attitude and the camera stabilizer pan data. As these data have the accuracy of around 0.5 [deg], the estimated sea-horizon line should not be used directly for the calculation of the object position, but it gives a good approximation of where to search for the sea-horizon.

To be accepted as the sea-horizon line, the best fit line must satisfy the following 2 decisive conditions:

- Number of edge points on this line must be larger than a threshold value.
- Deviation from estimated sea-horizon (roll, pitch differences), which is evaluated by (2.11) must be less than a threshold.

$$\begin{array}{l} \delta = \sqrt{\delta_{roll}^2 + c \times \delta_{pitch}^2} \\ if\left(\delta < \delta^* \;\; And\ (N < N_0)\right) then\ \mathrm{Pr}oduceHorizon() \\ where\ N\ denotes\ the\ number\ of\ connected\ pixels\ on\ the\ approximated\ line \\ \qquad \text{c (>1) is an adjusting factor used to put more weight on pitch deviation} \end{array} \tag{2.11}$$

If the fitting conditions are met, the line is taken as the sea horizon and is used for later calculation of object positions.

However, in unfavorable weather conditions, the sea-horizon is not detectable on the images, even with the human eyes. The algorithm therefore fails to detect the sea-horizon. Examples of these cases are shown in Fig.2.12 where the horizon line is obstructed from view due to the thick vapor layer.

For such cases, the estimated true horizon can be used instead. However, object-position accuracy is severely degraded accordingly and therefore should be treated with care. The horizon- detecting algorithm returns a fail.

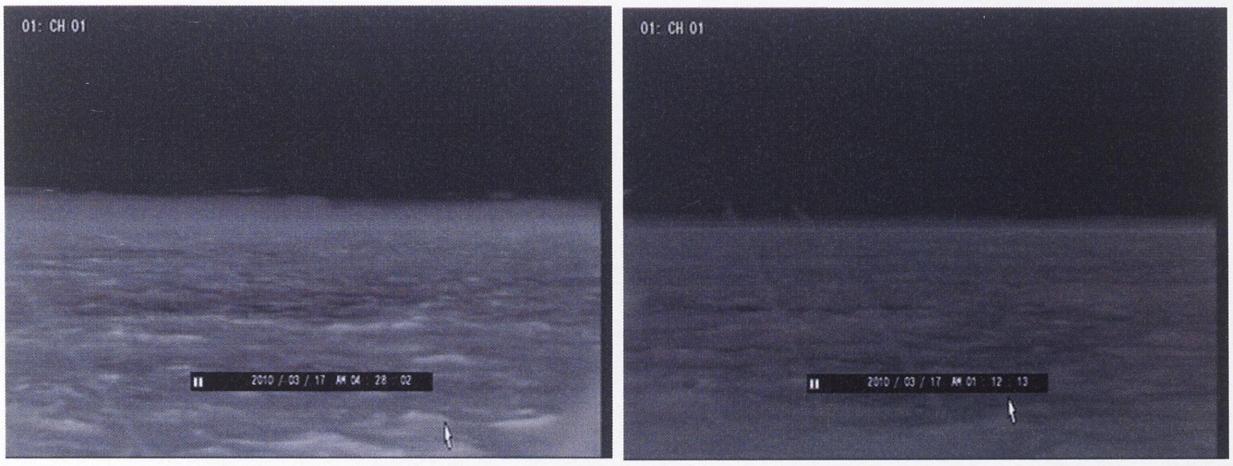

01: CH 01
2010 / 03 / 17 AM 04 : 28 : 02
01: CH 01
2010 / 03 / 17 AM 01 : 12 : 13

Fig. 2.12 Sea-Horizon Detection Fail

The horizon-line detecting algorithm, if successful, can ensure the accuracy of horizontal direction (direction of the true horizon, after dip correction, see Fig.2.7) to be better than 0.06 [deg] (2 pixels). Total effect of this error and other error sources such as the camera height variation is illustrated by experiment data as shown in Fig.2.21 and Fig.2.22 (section 2.5).

2.4 Floating Object Detection

2.4.1 General Principle

IR camera is temperature sensitive i.e. an object is detectable if there is a significant temperature difference between the object and its surrounding environment. The image is in gray format, with brightness value of pixel ranging from 0 to 255. As the image nature is similar to Radar images, the Constant False Alarm Rate (CFAR) method [2][9] can be used to extract objects.

Using this method, the existence of an object is detected by the intensity (or brightness) difference between the object pixels and the surrounding background pixels, including noise, clutter other disturbances. If brightness difference between a pixel and its surrounding pixels is above a threshold, the pixel is considered to be an object pixel; otherwise, it is simply background noise.

If the detection-threshold is set too low, unclear objects can still be detected at the expense of increased number of false alarms, i.e. background discontinuities are falsely seen as objects. Conversely, if the threshold is too high, just clear objects can be detected; the obscured objects as well as noise will not appear on the result image.

The method is used for cases in which it is difficult to decide the existence of an object just from its brightness peak. For example, for the sea surface images, pixel brightness varies as a function of distance to the IR camera, swell and wave pattern and thus it is impossible to apply a single value of the brightness-threshold to separate the objects with the wave crest etc.

For a floating-object on the sea-surface, its edge is usually clearer than those of waves. This should be taken into consideration in the object-detection program. In Fig. 2.13, an IR camera image in a wavy condition at sea is used as illustration. Variations of Pixel-brightness value in different directions (horizontal, vertical, $\pm 45^{\circ}$ upward) are shown in the respective graphs attaching to this figure.

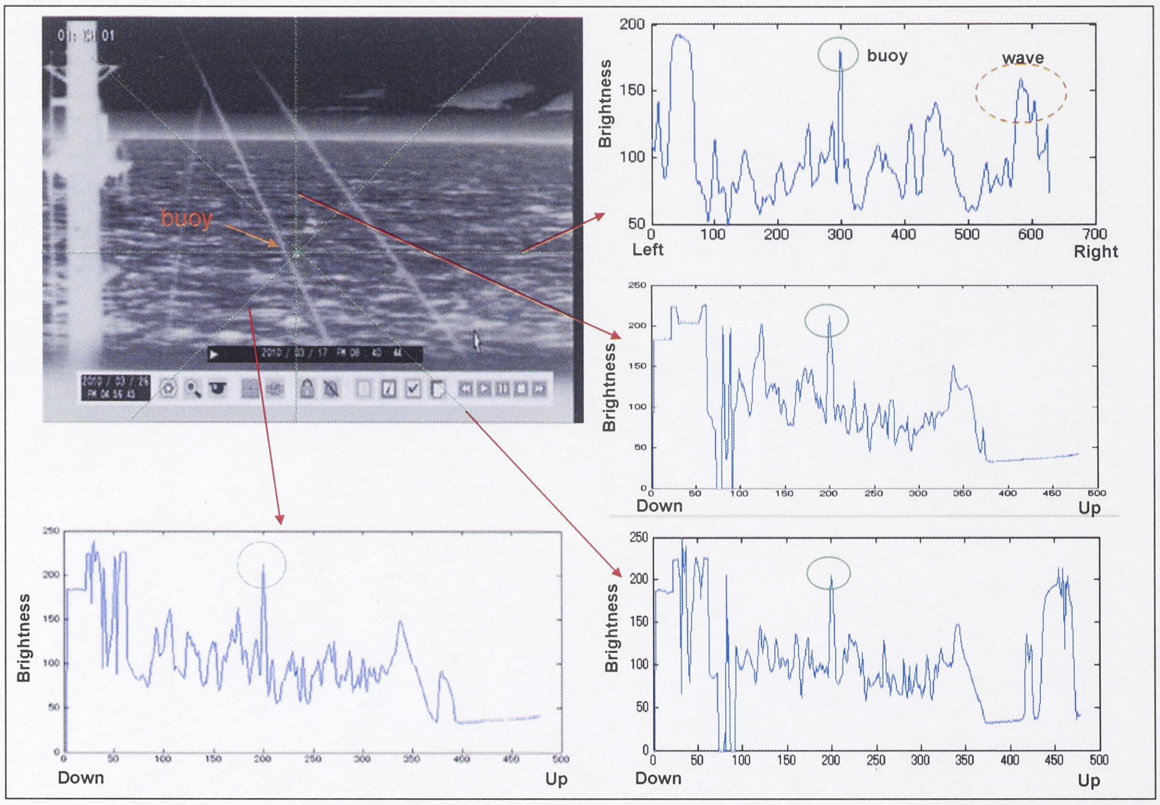


Fig. 2.13 Pixel Brightness Variation

In these graphs, a peak can be clearly seen for a buoy at a distance of approximately 150 [m] from camera position. Wave peaks, however, are also quite significant, in comparison with the object (the buoy). The edge of the buoy is sharp in all graphs while disturbance edges are more significant in the vertical direction than in the horizontal one. Another character of waves on the images is that it is unsteady.

2.4.2 Floating Object Detecting Algorithm

From the above perceptions, the object-detecting algorithm is performed through a procedure as shown in the flow chart in Fig. 2.14.

2.4.2.1 Image Median Filtering

The original camera image is pre-processed by passing through a Median Filter. The aim of this filter is to smooth the image. After smoothing, high-frequency variation like waves and other disturbances can be flattened, producing the filtered image in which the objects appear more clearly over the background.

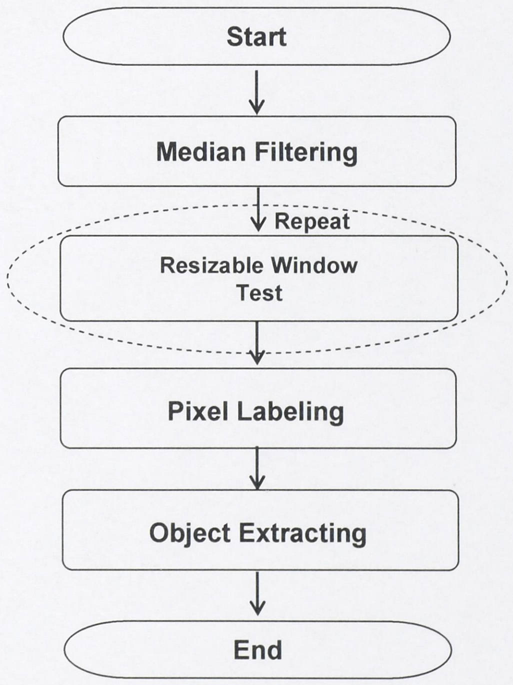


Fig. 2.14 Object-Detecting Algorithm

In this study, a 5x5 neighbor matrix is used for the filter. However, to increase the processing speed, it is applied by a 2 steps one-dimensional filtering:

- Step 1: Median filtering the image vertically
- Step 2: Median filtering the image horizontally

A pseudo-code for this operation is as following: for an input array in(), the value of equivalent output array out() at the position k is determined by

```
$for\ \ i = 1\ to\ \ NeighborSize\ \ (= 5)$
    $temp(i) = in(k - NeighborSize / 2 + i)$
$next\ \ i$
                                                        (2.12)
$sort(temp)$

$out(k) = temp(NeighborSize / 2)$
```

2.4.2.2 Resizable Sliding-Window Test

The existence of an object is tested by comparing the brightness of candidate-pixels with their surrounding pixels which are assumed background-pixels. These background-pixels are within an area called a window.

In this study, the authors employ the test recursively using a resizable window. The idea is illustrated as following, for a sample image row (or an image line):

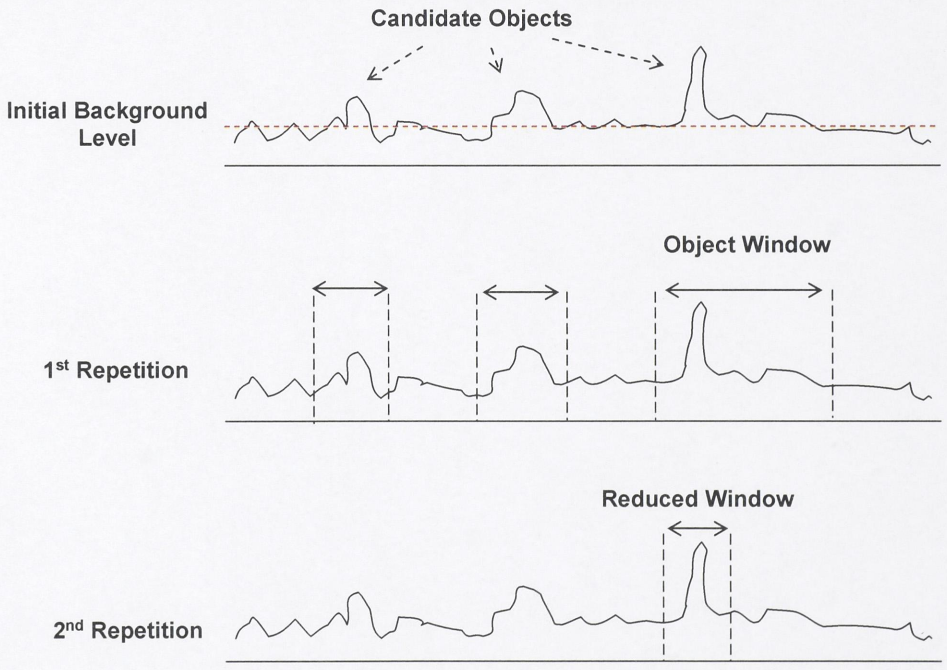


Fig. 2.15 Resizable Window Test

First, an initial brightness-level of background is set for the whole line. The brightness-level is chosen to be the average brightness of all pixels of which the brightness is smaller than the median brightness-value of that line.

$$\begin{aligned} &Median\_Brightness = \frac{Min\_Brightness + Max\_Brightness}{2} \\ &BG\_Level = Mean\left(pixel(i) \mid Brightness\,(pixel(i)) < Median\_Brightness\right) \end{aligned} \tag{2.13}$$

Candidate object-pixels are pixels which are brighter than the background-level a certain value called the Object_Threshold.

Then, the test is performed recursively for those candidate-objects with a suitable window size. The window is defined as shown in Fig. 2.16.

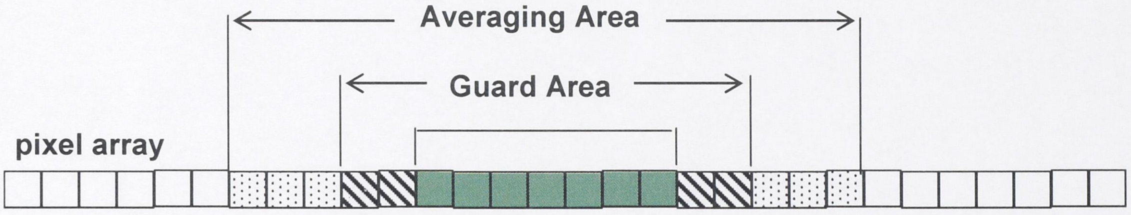


Fig. 2.16 Test Window

A guard-area (or guard-distance) is selected around the candidate-object so that the object edges can be skipped in the calculation. In this study, the guard-area width is set to be 2 pixels.

The averaging-area is an area outside the guard-area, with the size set wider to erase high frequency disturbances. Here, the width of the averaging-area has been selected to be 6 pixels on each side.

Then, the testing process is conducted using the following pseudo-code:

```
$Window\_Average = Mean\left(pixel(i) \mid pixel(i)\ inside\ averaging\_area\right)$
$for\ each\ object\_pixel\ obj$
 $if\ \left(brightness\,(obj) - Window\_Average < Object\_Threshold\right)\ then$
     $obj = Object\ Pixel$                                              (2.14)
 $else$
     $obj = Not\ Object\ Pixel$
 $end\ if$
```

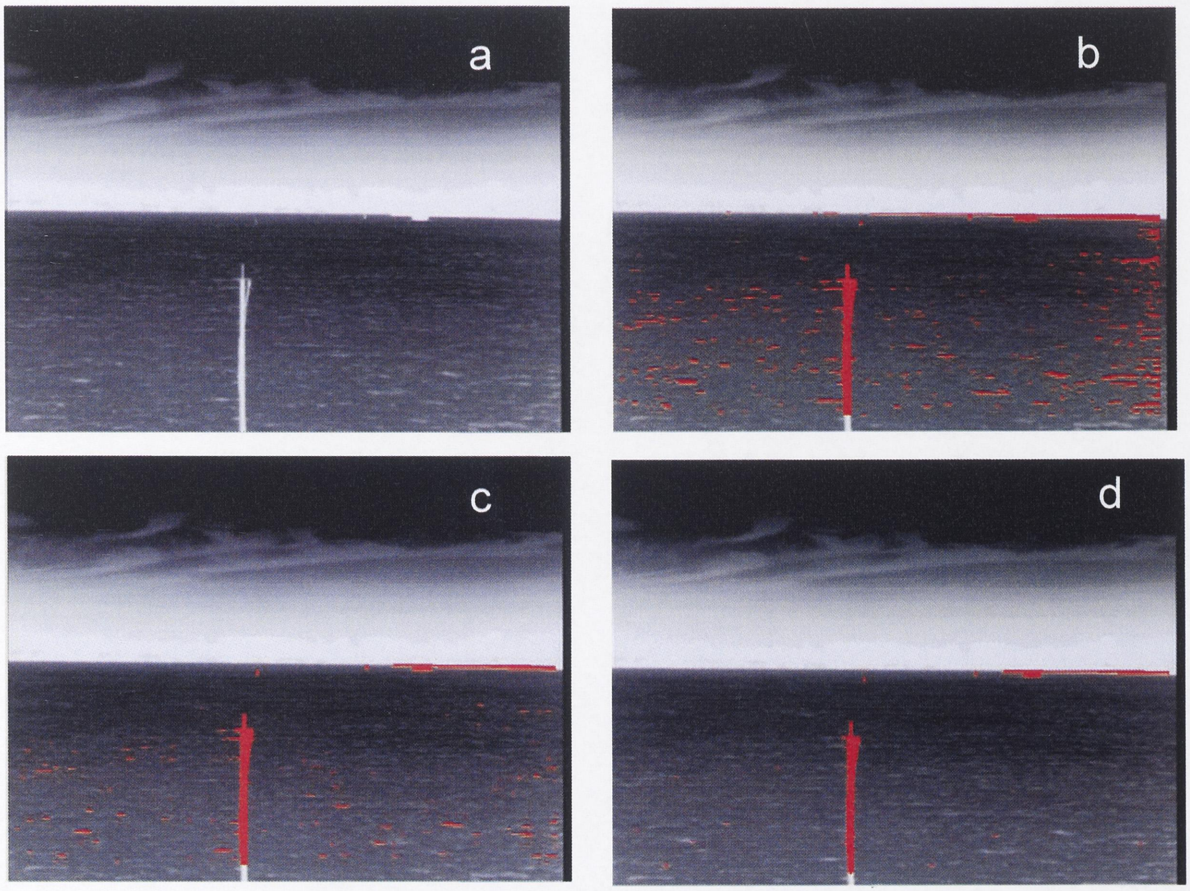


Fig. 2.17(a,b,c,d) Object-Pixels Detecting Process

Once status of a pixel changes from object-pixel to non-object-pixel, it is treated as a normal background-pixel for later processing. The process is repeated several times to gradually erase unwanted disturbances. The window-size for each candidate-object is reduced, according to the number object-pixels which have changed their status. This is illustrated in Fig. 2.15.

Starting with an initial image (Fig. 2.17a), the candidate-objects are marked in red in Fig. 2.17b. Then, after a number of repetitions, the final image with marked object-pixels can be achieved (Fig. 2.17d). It is clearly seen here that the wave-crests have largely been removed from the figure. Although false objects still exist in the image, they can be washed away later by checking their existence in consecutive images (see section 2.5).

2.4.2.3 Pixel Labeling and Object Extraction

Pixel labeling is the process of giving each object pixel a label. Pixels belong to a common group are members of a single object and therefore should be given the same label. This can be achieved by labeling the connecting pixels repeatedly.

After labeling, objects can be extracted from the image from pixel labels. Then a rectangle is defined to isolate the object using its topmost, leftmost, bottommost and rightmost pixels. The rectangle is called an object frame.

To reduce the possibility of mistakenly having several frames for a single object, deletion operator may also be applied. However this also may cause different objects to be grouped into one.

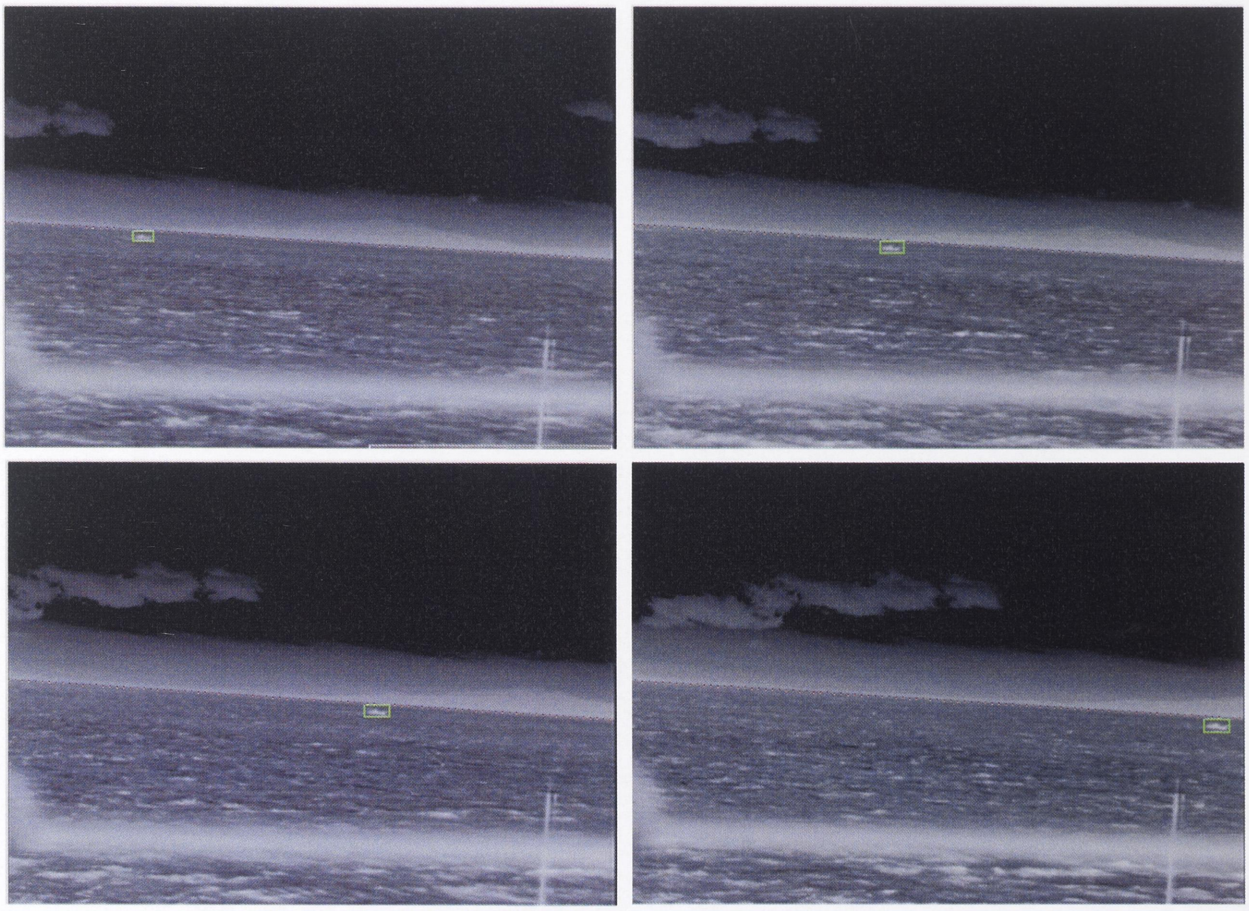


Fig. 2.18 Floating-Object Detection

Shown in Fig.2.18 is an object on 4 consecutive images received by IR camera. The object is a small fishing boat at about 1500m from the camera position. The sea-horizon is obviously seen and the program works as expected.

2.5 Floating-Object Tracking and Motion Fitting

The object is continuously tracked from its consecutive positions. The aim of the track prediction in NMRI project is to check whether floating objects are drifting into the Guard Area, which is an area behind our Own Ship. Then, it is necessary to gather the object-frames of the same target on sequential camera images. Target-following is also vitally important for other shipping application such as collision-avoiding support. This can also reveal objects that have been mistakenly detected from the previous step i.e. a correctly detected object should appear frequently on consecutive images.

In this study, the relation between object-frames on consecutive images (see Fig. 2.19) is evaluated by a relating-value. The value takes into consideration similarities in the object frame sizes (S_rel), distances to camera (D_rel) and bearings.

The relation is evaluated by (2.15) of which the components are defined as shown in the following equations. Two object-frames (a frame at time t and another at time t + 1) are considered to be sequential frames of a single floating object if the following 2 conditions are satisfied simultaneously:

- Their relating-value is the smaller than the relating-value between one object-frame (among the two frames) with any other object-frames on the other image.

- The relating-value is smaller than a threshold value.

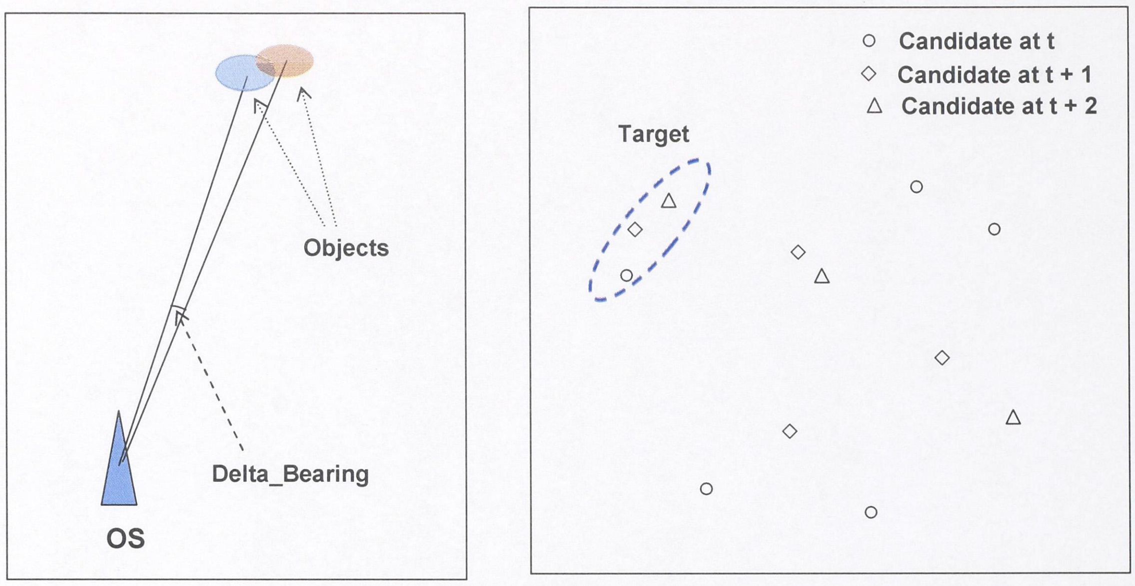


Fig. 2.19 Object Track Prediction

$$f(obj1, obj2) = S\_rel \times D\_rel \times B\_rel$$

where

$$\begin{aligned} S\_rel &= LargeObjectSize \;/\; SmallObjectSize \\ D\_rel &= DistBetweenObjects \;/\; Limit\_Dist \\ B\_rel &= (Delta\_Bearing + 0.5) \;/\; Limit\_Bearing \end{aligned} \qquad (2.15)$$

$$f(obj1, obj2) < Threshold\_relation$$
$$\Rightarrow \quad image\ object\ of\ the\ Same\ Target$$

The test with this relating function has proven that object-frames detected in Fig. 2.15 belong to a single floating-object (a fishing boat, actually). They are plotted on Fig. 2.20 (right hand side figure).

Due to the errors in position-determining algorithm (the sea-horizon detection error, antenna height fluctuation etc.), the consecutive positions of an object are fluctuating about its track. Then, in this study, object track is predicted using least square method (LMS). LMS is used with the assumption that target movement is constant. This assumption is appropriate as target can not change its speed and course much in a short period of time (less than 1 [min]).

Using this LMS algorithm (2.16), the latest position of the target (X0, Y0) and its 2 velocity components (Vx, Vy) can be determined so as to minimize the total square error which is denoted by J. For illustration, predicted track has been calculated and drawn in Fig. 2.20.

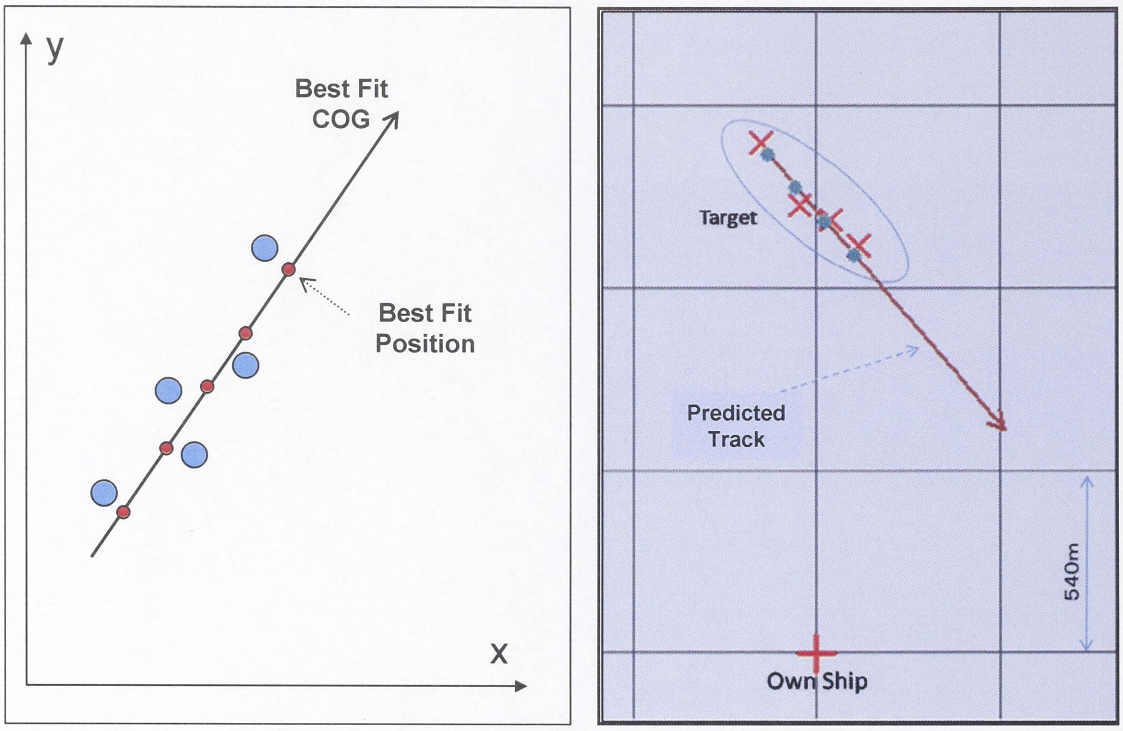


Fig. 2.20 Object Positions and Track Fitting

$$
\begin{aligned}
&X_i = X_0 - i \times \varDelta X \\
&Y_i = Y_0 - i \times \varDelta Y \\
&J = \sum (X_i^* - X_i)^2 + \sum (Y_i^* - Y_i)^2 \quad or \\
&J = \sum (X_i^* - X_0 - i \times \varDelta X)^2 + \sum (Y_i^* - Y_0 - i \times \varDelta Y)^2 \\
&i = 0 \ to \ NumberOfPoints \\
&X^*, Y^*: Observed\ target\ position \\
&X, Y: best\ fit\ target\ position \\
&\varDelta X, \varDelta Y: best\ fit\ x, y\ speed
\end{aligned}
\tag{2.16}
$$

Solution to this LMS algorithm is simple and will not be further mentioned here. A detail description of the method can be found in the Annex V.

2.6 Object Tracking Accuracy

Due to the effects of different error sources, including the sea-horizon detecting error, camera height etc., position of an object that is calculated by the program pertains to some uncertainty that closely depends on the relative distance from the object to the camera position. This dependence is illustrated in Fig. 2.21 and 2.22. Objects material may also contribute some error to the accuracy due to the error in object water-line detection and should be further studied.

Fig. 2.21 expresses the variation of distance from camera which is assumed to be fixed at sea to a non-moving object (an anchoring ship) at different distances. An increase in variation of the measurement with increasing distance can be easily seen.

- For a target at about 200[m], the deviation is around 2 [m]
- For a target at 600[m], the deviation is approximate 8 [m].

- For a target at 1100[m], the deviation is around to 20 [m] .

The experiment was conducted on Oct. 28th 2009 (17:00 to 18:00). The weather was fine with air temperature of approximately 21.0°C and the average sea-temperature to be 22°C. It should be noted here that the weather was quite favorable in this experiment thus the camera height (from sea level) does not vary much from one sampling to another.

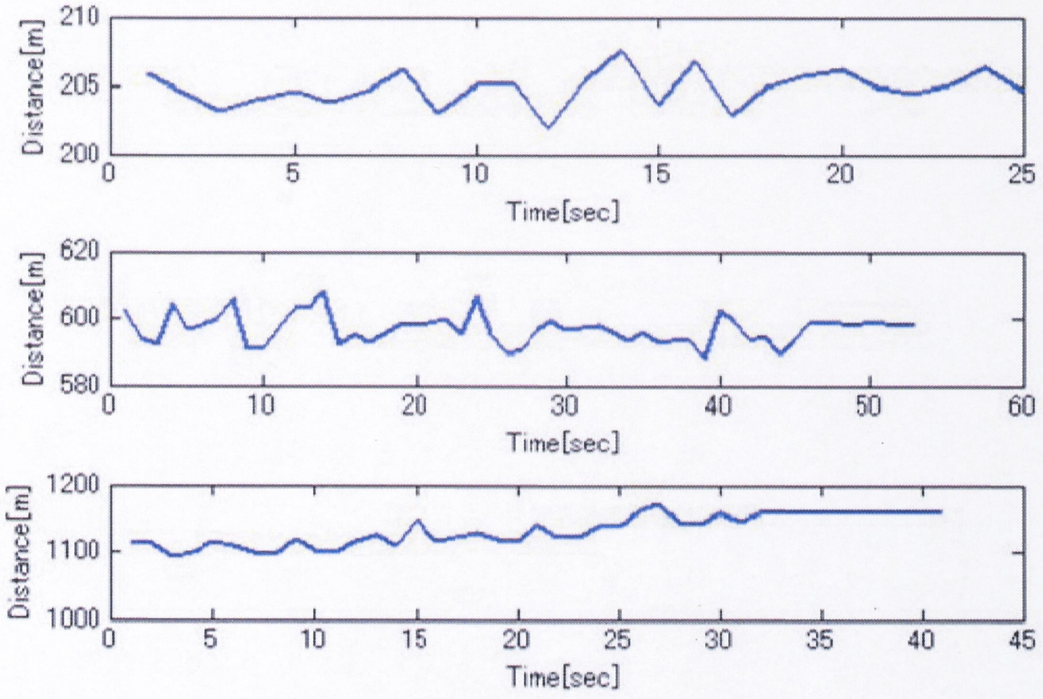


Fig. 2.21 Tracked-Distance Variation

The position determining algorithm accuracy is further verified by cross checking with values measured by Lidar. The distance to the object deduced from IR image is compared with its equivalent Lidar measurement and the result is shown in Fig.2.22.

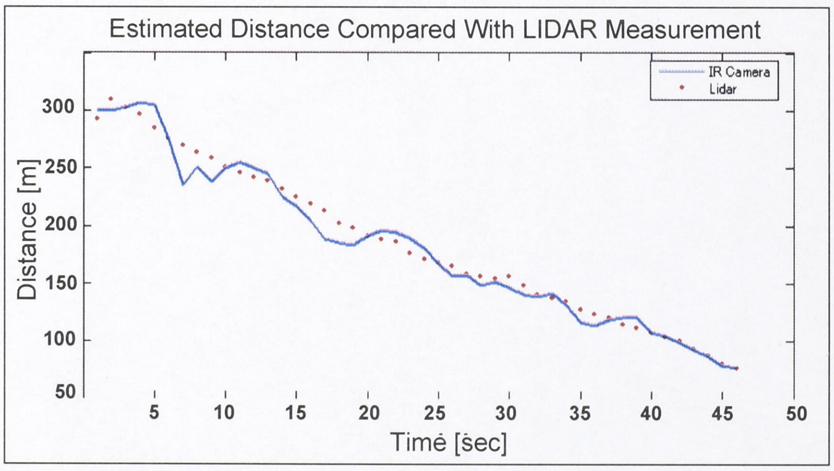


Fig. 2.22 Distance in Comparison with Lidar Measurement

In this figure, a fluctuation of the former about the latter is obviously noticeable. It is due to the fact that the camera height correction due to ship motion (rolling, pitching and heaving) is not applied. This factor can be taken into consideration by calculating the camera height at every sampling interval, using the ship attitude and camera position relative to the ship center. The experiment (20:00, Mar. 17th 2010) was in heavy weather (sea state), with air and average sea temperature to be 0.3°C and 4°C respectively.

The track prediction, accordingly, pertains to some error. It depends on, among others, the distance to the camera and number of observation used. In our experiment with an unmoving object (buoy or small boat), speed error for distance of app. 1200 [m] is 1.5 [m/s] (if 20 seconds of observation is used). This can be reduced by increasing the number of observations (50 seconds, e.g.) at the cost of more calculation needed. Longer data-sequence should be taken to minimize the fitting uncertainty, especially the course prediction.

2.7 Manual Object Tracking by Laser and Night Vision Cameras

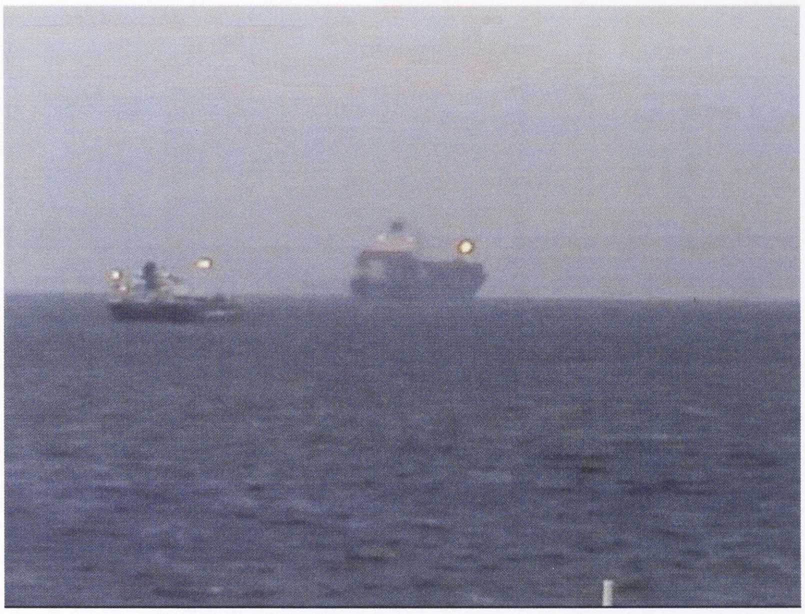


Fig. 2.23 Night Vision Camera Image

Apart from the IR camera, a NV camera and a Lidar camera are also used in the observing system. A sample image of the NV camera is shown in Fig. 2.23 and a Lidar image is in Fig. 2.25. These data enable the observation in many different conditions in which the observation by IR camera alone is impossible.

NV camera is suitable for the observation at longer distance where the object water-line does not deviate largely from the sea-horizon and therefore can hardly be detectable on fixed lens IR camera. For example, further target in Fig. 2.23 appears clearly on Fig. 2.24, thank to the adjustment of the camera focus.

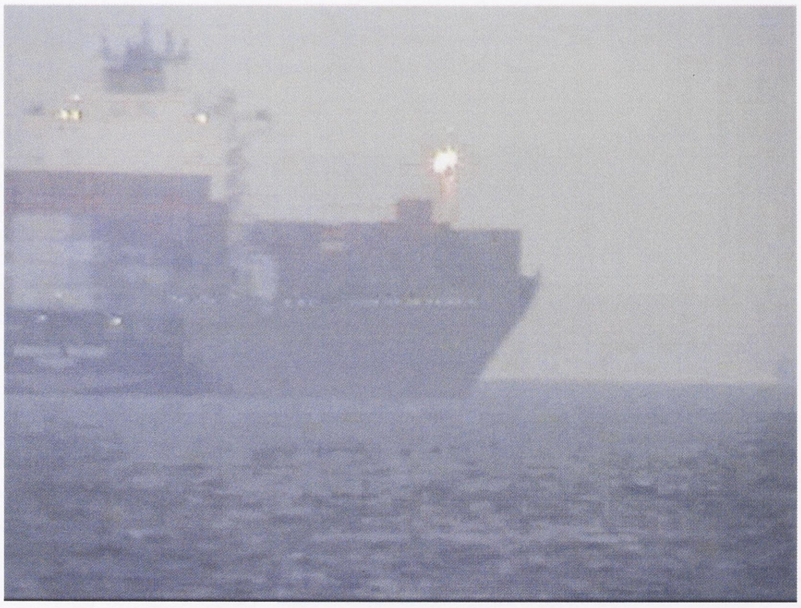


Fig. 2.24 Close Range Night Vision Camera Image

Lidar Camera, on the other hand, allows the detection of very small object (buoys or small floating objects) at a distance of less than 2000[m] from the camera, even in unfavorable weather condition.

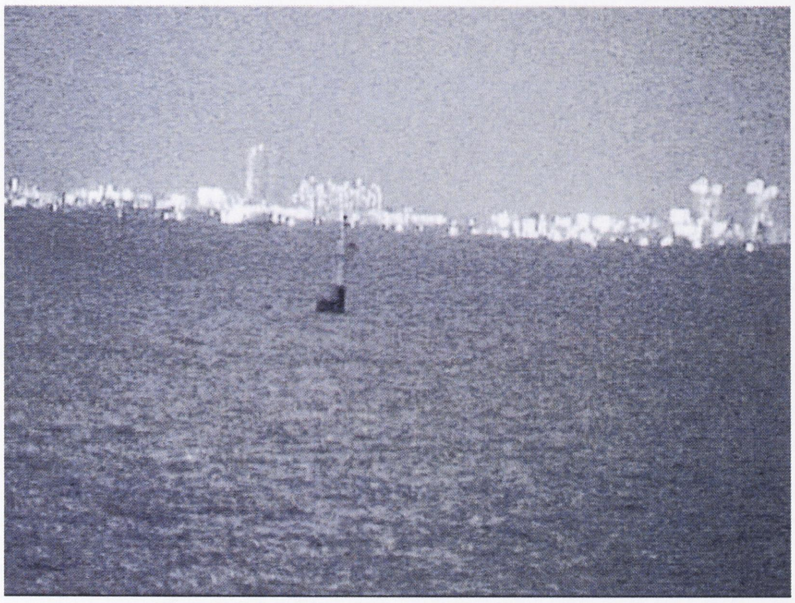


Fig. 2.25 Lidar Camera Sample Image

However, their applications have not been studied thoroughly in this study due to the expiration of the project. To provide a quick use of the data acquired by these 2 cameras, a manual tracking function was added to the program. Using this function, the user double clicks on the camera image at the object positions to calculate the object position or to track it manually.

With the horizon-line predicted using IR camera image or NV image directly, the 2 angles α and β can be recalculated by the following equation (2.17) (refer also to Fig. 2.6).

$$\begin{aligned}
&\alpha[\mathrm{deg}] = v[pixel] \times Rat \\
&\beta[\mathrm{deg}] = t[pixel] \times Rat \\
&\quad where \\
&\quad Rat = \frac{Camera\_Horizontal\_Angle[\mathrm{deg}] + Camera\_vertical\_angle[\mathrm{deg}]}{640[pixel] + 480[pixel]}
\end{aligned} \tag{2.17}$$

From these values, (2.5) and (2.7) can be applied to find the object position at sea.

2.8 Conclusions

In this chapter, an all-time all-weather observing system basing on cameras was described. Then a floating-object detecting and tracking program has been studied thoroughly. From the experiments, it has been proven that

- IR camera, if properly used, is a very effective tool for floating-object tracking purpose.
- The proposed algorithm has better performance than other available algorithm for the sea-horizon detection.
- The performance of the object-detecting algorithm is acceptable for weather conditions frequently met at sea.
- The system is able to detect objects, predict their track and give warnings if the objects are floating into the Guard Area.
- For track prediction purpose, system is reliable for targets at less than 2000 [m] distance. The further the target is, the less accurately its position can be estimated.
- Effectiveness of the system is, however, seriously reduced in bad weather condition.

In comparison with radar tracking, camera observing system performs rather poor in terms of tracking accuracy and effective range. However, it can be a supplement for other available observing methods (radar, AIS). The tracking accuracy of target at less than 1000[m] is acceptable for application like collision-avoiding support.

Acknowledgement

This study was a part of the research project on the "Development of Track Estimation System for Floating Object Surveillance" conducted at the Japan National Maritime Research Institute. The project itself belongs to the JOGMEC (Japan Oil, Gas and Metals National Corporation) Public Offering Type R&D project on the "Study of Cooperative Navigation Support System aimed for Safe and Effective Operation of the Seismic Vessel using High-speed and Large-capacity Network from Ship-to-Land, Study of All-Day All-Weather Marine Surveillance Technology". It was a great favor for me to be able to work as a Research-Assistant under the supervisions of prestige researchers: Dr Sasano, Dr Kiriya, Master Imasato and others (NMRI), Mr Futaki (NTT Communication Corporation), Mr Asanuma (JOGMEC), to whom, I would like to express my gratitude and all respects. My contribution to the project was actually too molest in comparison with what I could learn and was generously granted.

References

1. E. Nadernejad, "Edge Detection Techniques: Evaluations and Comparisons", Applied Mathematical Sciences, Vol. 2, No. 31, pp.1507-1520, 2008

2. H. You and G. Jian, "A New CFAR Detector with Greatest Option", Journal of Electronics, Vol. 14, pp. 125-132, 1997
3. Japan Meteorological Agency: Weather, Climate & Earthquake Information, http://www.jma.go.jp /jma/menu/report.html
4. M. Sasano, J. Kayano, Y. Futaki and K. Maeda, "Development of All-Day, All-Weather Hybrid Marine Surveillance System", Journal of Japan Institute of Navigation, Sep. 2010
5. M.B. Ahmad and T.S. Choi , "Local Threshold and Boolean Function Based Edge Detection", IEEE Transactions on Consumer Electronics, Vol. 45, No. 3, 1999
6. M.D. Nguyen, M. Imasato, Y. Futaki and T. Asanuma, "Development of Track Estimation System for Floating Object Surveillance", Journal of Japan Institute of Navigation, pp69-76, 2010
7. S. Peleg, J. Naor, R. Hartley, and D. Avnir, "Multiple resolution texture analysis and classification", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 6, pp.518-523, 1984
8. S. Price, "Edges: The Canny Edge Detector", July 4, 1996. available at "http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/edges/canny.htm"
9. T. Y. Liu, K.T. Lo et al., "A New Cut Detection Algorithm with Constant False-Alarm Ratio for Video Segmentation", Journal of Visual Communication and Image Representation, pp. 132-144, 2004

Chapter 3 Automatic Collision-Avoiding Support System and Optimal Route Generation by Dynamic-Programming

3.1 Introduction

Thank to the development of technology and the abundance of on board equipment, the modern ship officers now have the accessibility to a huge amount of information relating to movements of nearby targets as well as other environmental conditions. However, even with those advanced equipments, maneuvering has never been an easy task, especially when navigating in congested waters. Marine accidents are still happening mainly due to mistakes of the officer of watch in information judging and counter-action deciding. Furthermore, large amount of information may distract him from the most dangerous encounters. Thus, many different researches have been carried out on collision-avoiding support systems.

Those works mainly aim at providing the officer a recommended course (heading) to avoid collision with the most dangerous target ship nearby. The problem of this approach is that if there are 2 or more target ships (TS) navigating in the region, collision-avoiding course to Just One TS may navigate the own ship (OS) to an extremely difficult position for further maneuvering. This means that OS might be in a position that is too difficult to avoid the collision with the second, third... TS, after the first one has been safely passed. It is because those TS movements, except the most dangerous TS, were not sufficiently taken into consideration for route generating.

If, on the other hand, the collision-avoiding course is calculated for all TS at the same time, the resulting collision-avoiding course may cause the OS to deviate largely from its original route. It is often the problem accompanying with the traditional collision-avoiding support system basing on TCPA, DCPA criterion.

Another problem of the current approaches is that the quality of collision-avoiding route has not yet been properly evaluated. There are certainly thousands of collision-avoiding strategies for the OS, so why should not we choose the optimal strategy among them? The quality of a strategy must be evaluated by some suitable criteria attaching to the marine traffic rules and economic considerations.

Thence, the aim of this study is to generate a collision-avoiding route for the ship to pass all TS as well as other constraints, not a single target at a time, safely and economically. It should be noted here that the produced collision-avoiding strategy is a route or a trajectory for the OS from a starting-point to an end-point on the original route, NOT JUST a heading. For the system to be as helpful as possible (operating with little or without human efforts except for supervising), it must be able to solve simultaneously the following tasks

- Be on watch to detect any arising risk of collision, including the coming of a new TS, the deviation of the existing TS from their paths i.e. TS changes its course or speed, and the deviation of OS from its safe track.
- Be able to conduct real-time calculation of a minimum time route to be clear from any dangers, while maintaining OS in proximity of the pre-voyage planned route and the tendency to reach the destination.
- Maneuver OS to follow the previously calculated safe route.

Among these 3 tasks, our study concerns mostly the first two ones. The tracking-control problem has been extensively studied recently and is therefore not the research subject of this study. However, to fulfill the idea behind the collision-avoiding support system, tracking-control block is also included in system figures and other flow charts.

In this chapter, general concepts of the automatic collision-avoiding support system will be described in section 3.2. Then, the principle of route generation with necessary inputs, including

TS information sources, OS maneuvering characteristics and criteria for collision-risk assessment will be presented in section 3.3. The route generation by Dijkstra's algorithm (Dynamic-Programming - DP), its pros and cons will be the subject of section 3.4. Section 3.5 analyzes the route-producing algorithm through computer simulations. Conclusions on the route-generating method will be summarized in section 3.6.

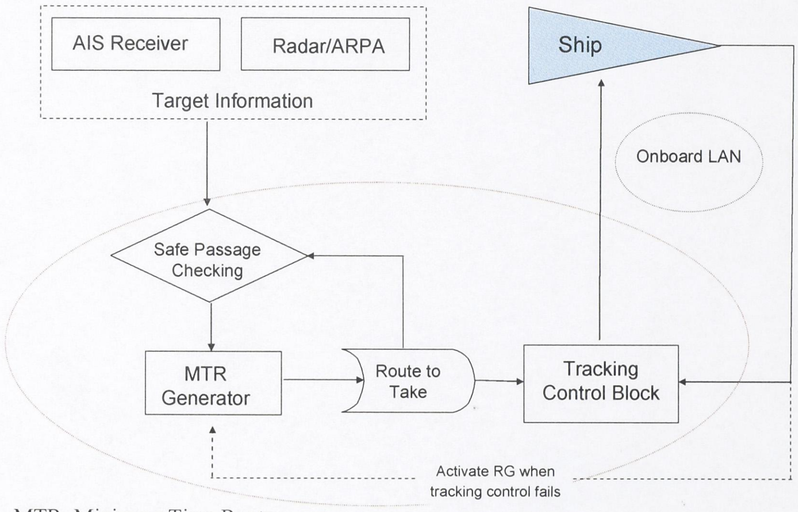
The algorithm is proposed basing on the following 2 assumptions:

- TS do not change their speeds and courses.
- Collision-avoidance is the duty of our OS alone, even for the cases where it is a stand-on vessel.

3.2 System Overview

Apart from the target motion observing unit and the communication link with OS data-collecting and controlling center system, the collision-avoiding support system includes a computer based program which consists of 3 modules solving the above-mentioned 3 tasks respectively.

A Watch-Keeping module continuously receives TS motion data through Radar and AIS. Camera system (Chapter 2) is also a possible source of TS motion information theoretically. However, as the effective range of cameras is heavily circumscribed and their horizontal opening-view is small, it is of limited use for the route-producing application. To assess the risk of collision, the OS data and its planned route must also be used. OS motion data is received through an onboard local network. The local network allows 2 ways data transferring, through which OS position, speed, etc. are available for calculation at the program side and OS-commands can be sent to the other side. The watch-keeping module is permanently on watch to detect any risk of marine traffic accident. The collision-risk may be the result of one or several of following unexpected evolutions:



MTR: Minimum Time Route

Fig. 3.1 System Configuration

- OS deviated dangerously from its planned route.
- An existing TS, i.e. TS already in TS database, changed its course or speed so that the encounter case between OS and the TS changed.
- A newly-coming TS is interfering in the OS planned route.

If the planned route is no longer safe, the Watch-Keeping block is to activate a route-generating module so as to produce another safe route for the OS.

The Route-Generating module takes target motion from target database and the environmental constraints (route limitation, fishing area, anchorage area etc.) from other static sources such as ECDIS as the input. The OS maneuverability must also be taken into consideration to make the produced route viable. In this study, the Dynamic-Programming (DP) Method basing on Dijkstra's algorithm is applied. The route thereby produced is the Minimum Time Route to safely avoid all the possible dangers. This is the subject of the following sections. The produced route can then be considered the collision-avoiding route, following which OS will be navigated.

A Tracking-Control block is used for ship tracking control. The responsibility of this block is to handle the ship following the route which has been calculated in the earlier step. In this study, rudder is used as the single actuator i.e. the under-actuated tracking control problem. It is assumed that the ship would not change its speed (engine revolution speed) to avoid collision. The assumption is reasonable, keeping in mind the conventional seamanship. This also simplifies the algorithm of the OS dynamics as the coupling between thrust force and rudder command changing is too much complicated. The rudder-control signal is sent to the ship main-board through the network. The overall system configuration is illustrated in Fig. 3.1.

3.3 Route Generating Principle

3.3.1 General Principle

As mentioned above, DP is used to generate collision-avoiding route with the inputs to be TS information, environmental constraints and the OS maneuvering model (Fig. 3.2).

The environmental constraints might be the water around a buoy, a military zone, a fishing area that OS should avoid, etc. Also, OS should not deviate largely from it original planned route so as not to loose much way. From this information, a graph which is hereafter referred to as a grid system is built for the navigable area around OS original route (Fig. 3.3).

The grid system between starting-point A and ending-point B on original route consists of grid lines and points on lines. Distance between the grid lines, distance between points on a line as well as number of points are designing parameters of the grid. These largely affect the performance of the route-producing algorithm. If the distance between lines is small, the number of calculations increases accordingly. However, the quality of the produced route can also be improved. On the other hand, if this distance is large, the number of calculation is limited but route quality is down-graded as a result. The grid is expanded some distance around the original route. If this distance is big, OS is more flexible for collision-avoiding maneuver at the price of the growth of calculation volume. Restricted areas are covered by suitable polygons such as the pentagons in Fig. 3.3. Those polygons can be automatically or manually input before the voyage for the whole planned route and are

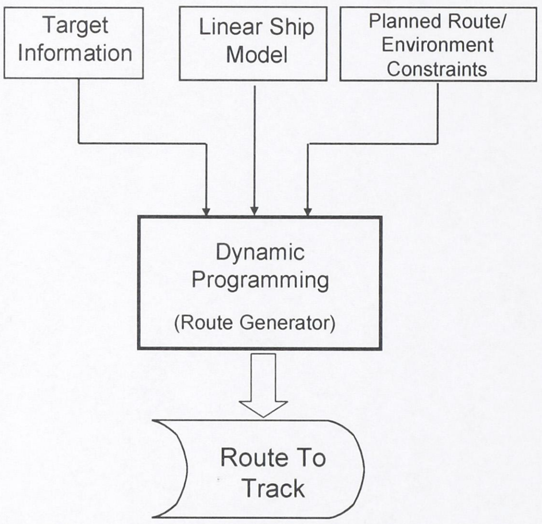


Fig. 3.2 Route Generating Module

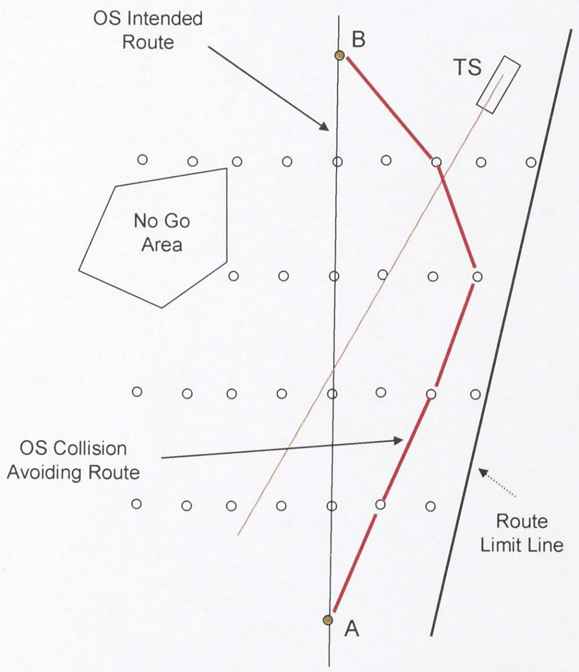


Fig. 3.3 Route-Producing Principle

kept in the voyage-database. These restricting polygons are recalled later when the OS approaches a certain sea area.

In the same manner, limiting lines can be manually input to figure out an area around the planned route through which officer wants the ship not to penetrate out.

A safe route for the OS is the shortest route from the starting-point, via exactly 1 grid point on every line to reach the end-point that does not cause the ship to enter a restricted area, to go out of limiting lines or to be in risk of collision with any TS.

This route is calculated with the assumption that course and speed of TS are constant. If these values change, the route is to be calculated again as Watch-Keeping module does its job to detect the risk and activate the route generator.

Throughout this study, distance between grid lines is set to be around 1700 [m] and distance between points is 50 [m]. These 2 values have been selected through a number of simulations, using try-and-error method. They appear to be suitable for the OS (around 100 [m] long) and the computer processing speed.

The DP algorithm is applied in this situation to provide just an approximation of the optimal solution due to the fact that environmental condition is time-variant, i.e. the cost of going from one grid point to another is varying as TS positions are changing. An optimal solution is theoretically possible but impracticable due to the increase in power-order of number of calculations and variables that must be kept in computer memory.

3.3.2 Evaluation of Collision-Risk

The task of the route-generator is to produce a safe route for the OS, given all TS motions. The safe passage is therefore must be judged using appropriate collision-risk assessing criteria. The criteria mentioned in this section deal only with collision-risk in Ship-to-Ship encounters. Since the dawn of navigation, many different criteria have been proposed including the Environment Stress Model, the Difficulty Value Model and the object domains etc. In this study, the following 4 criteria are applied. Each criterion has its own advantages and disadvantages, and is therefore applied in suitable condition of maritime traffic.

3.3.2.1 Evaluation of Collision-Risk by SJ Value

Subjective Judgment (SJ) value has long been used as a criterion of collision-risk assessment, representing the pressure of surrounding vessels on the officer of watch. It is a model for collision-avoidance with fuzzy reasoning [7]. SJ value is calculated for 3 following cases of Ship-Ship encounters

Crossing encounter:

Own ship is give-way: $SJ=6.00\Omega + 0.09\ Rp - 2.32$ (3.1)

Own ship is stand-on: $SJ=7.01\Omega + 0.08\ Rp - 1.53$ (3.2)

Head-on encounter: $SJ=6.00\ \Omega + 0.09\ Rp - 2.32$ (3.3)

Overtaking encounter: $SJ= 54.43\ \Omega + 0.24\ Rp + 2.77\ dRp/dt - 0.784$ (3.4)

where:

$\Omega = |d\theta/dt|\ Lo/Vo$: non-dimensional change rate of TS bearing

$Rp = R/\{(Lo + Lt)/2\}$: non-dimensional distance between OS and TS

$dRp/dt = Vr / Vo$: non-dimensional relative speed between OS and TS

$d\theta/dt$: change rate of TS bearing (rad/s)

Lo, Lt: length of own ship, target ship (m)

Vo: speed of own ship (m/s)

Vr: relative speed between OS and TS (m/s)

R: distance between OS and TS (m)

It is obvious that the parameters used in formulae (3.1) to (3.4) are those that can be acquired by the officer of watch visually or by available navigation aids such as RADAR/ARPA, AIS. These parameters are also taken into account by experienced navigators, intentionally or unintentionally, when considering the risk of collision with a TS.

Values of the factors and constants in the formulae are reasoned from simulation experiments. The values defined above have been generally accepted and are commonly in use.

Simulation has proven that there is a direct relation between SJ value and the risk of collision. The collision-risk of the encounter case can be assessed from SJ value perceived by OS officer as followings:

a. $SJ > 0$: Encounter is "Safe".
b. $0 \geq SJ > -1$: Encounter is "Cautious" and needs following.
c. $-1 \geq SJ > -2$: Encounter is "Dangerous".
d. $-2 \geq SJ$: Encounter is "Very Dangerous".

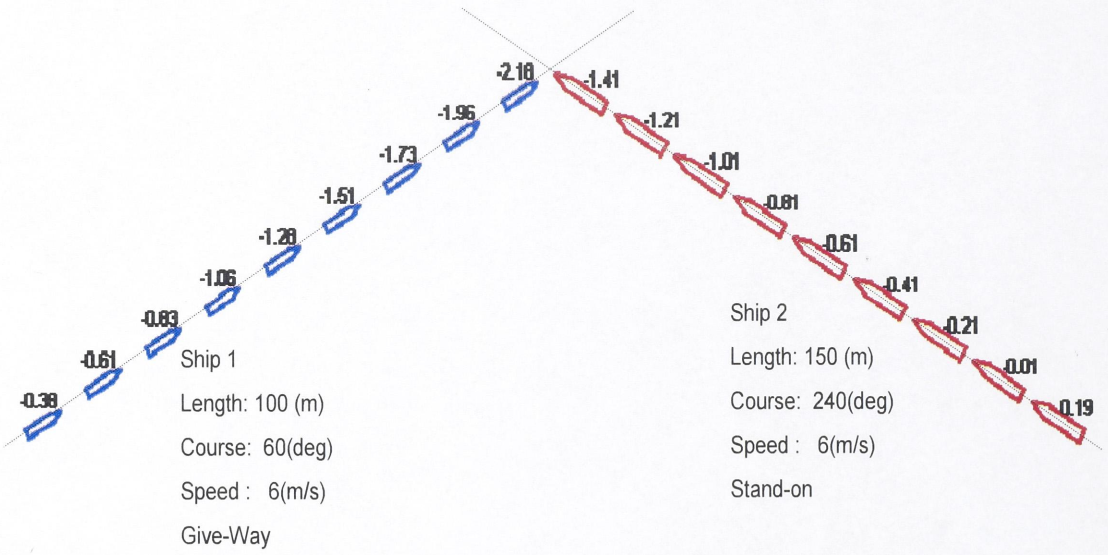


Fig. 3.4 SJ Value Evolution of Two Ships in Crossing

As an illustration, evolution of SJ value of 2 ships in a crossing encounter is shown in Fig. 3.4. Because ship 1 is "give-way", the SJ value it perceived is smaller than that for ship 2. When it is around 9L from the colliding position, SJ value falls below the safe limit (-1) and collision-avoiding action should be taken immediately.

3.3.2.2 Evaluation of Collision-Risk by Bumper Model

Using marine traffic data inside Tokyo Bay as observed by Radar and AIS systems, a simple model has been suggested for the assessment of collision-risk in congested waters. The model is named Bumper Model and has been applied extensively for route-planning purpose due to its simplicity and explicitness. Using the model, the watching-region for safe navigation of a ship is assumed to be the "Bumper" as defined in Fig. 3.5. The bumper consists of 2 parts separated by the traverse axis of the ship. The bow-part is a half of an ellipse along its major axis. Size of the ellipse is 6.4L for the semi-major axis and 1.6L for semi-minor axis. Stern-part of the bumper is a half of a circle of 1.6L in radius. Here, L is the length of the ship in concern [8].

As the name itself has implied, bumpers of 2 ships should not overlap each other. When it is the case (see Fig. 3.5) the 2 ships are considered to be in a situation with high risk of collision, and collision-avoiding action should be taken as quickly as possible.

To reduce the calculation volume, the model form is approximated by a rectangle which is externally tangential to it. The size of this rectangular bumper model is therefore 8.0x3.2 ship length (see Fig. 3.5).

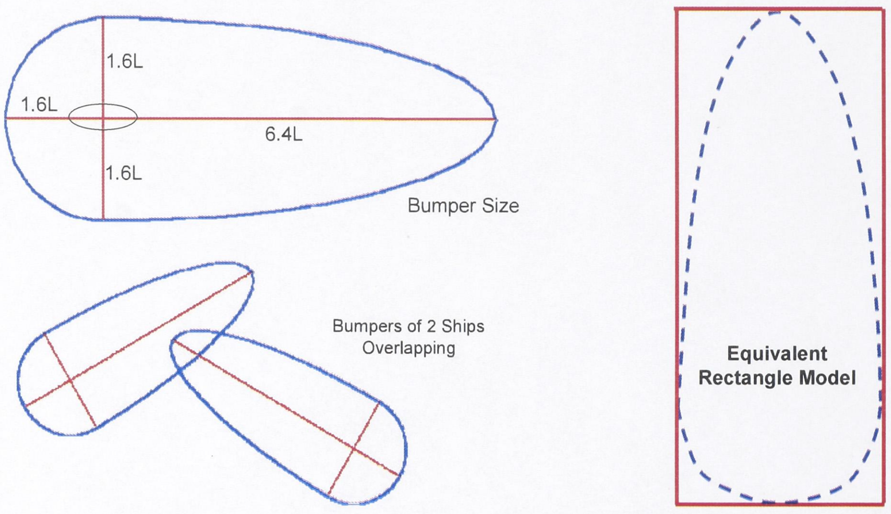


Fig. 3.5 Bumper and Simplified Models

It should be noted that the bumper model considers equally the dangers causing by targets approaching from starboard side and port side. This may sound unpopular at first, from the seamanship view point. However, keeping in mind that the marine traffic in congested waters is in concern, the use of the model form is appropriate.

In comparison with Bumper Model, SJ value has the advantage that the tendency of changing of SJ value is also available and can be used as the first clue of a coming dangerous encounter. This means that if SJ value is decreasing steadily, TS should be closely watched. However, simulation study reveals that SJ value is not really reliable in the overtaking encounters. The Bumper Model, on the other hand, has its limitation as the speeds of the ships are not taken into account and it is difficult to differentiate risk of collisions with TS approaching at different speeds. Therefore, Bumper Model and SJ Value should be used together for adequate risk assessment.

3.3.2.3 Evaluation of Collision-Risk by Object Domain

SJ value and Bumper model are suitable for risk-assessment in congested waters. However, at the open sea, the ship officers tend to take actions to avoid collision at much larger distances. Thus, a more proper criterion should be used for the open sea encounters.

A moving target represents a collision threat which is configured as an area of danger, moving with TS speed and direction. Goodwin [10] presented the method for estimating the area of danger on the basis of statistical data analysis. Following the maritime law, the area of the object-occurrence was divided into 3 sectors defined by the actual

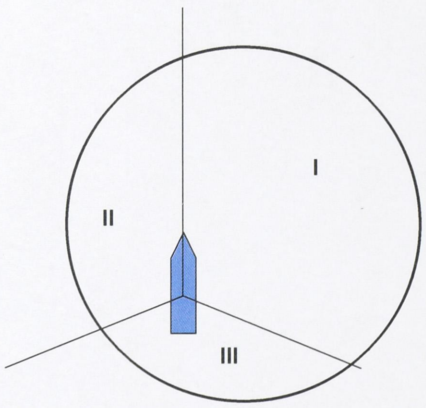


Fig. 3.6 Davis's Ship Domain

relative bearing from this object. Sector I is on the starboard side within the bearing limits of (0^o-112.5^o), sector II is on the port side with in the limits (247^o-360^o) and the stern sector i.e. sector III is within (122.5^o-247.5^o). The dimensions of the 3 sectors were estimated by statistical data. Davis et al. proposed a simplified version of this model that results in the domain form shown in Fig. 3.6. Davis's domain form is however, redundant in some aspects.

Recognizing the redundancy of Davis's domain, R. Smierzchalski et al. in their work suggested the hexagon domain as shown in Fig. 3.7. The appearance of a navigational constraint in the vicinity of the domain contour or at a distance ahead on the planned passing-trajectory that depends on the navigator's experience means the appearance of a navigational risk. The risk increases as a result of the decreasing distance to the detected constraints. Sizes of a domain on its course are computed from its length and speed, together with a chosen minimum time and distance of approaching (TCPA, DCPA) as the followings (see Fig. 3.7):

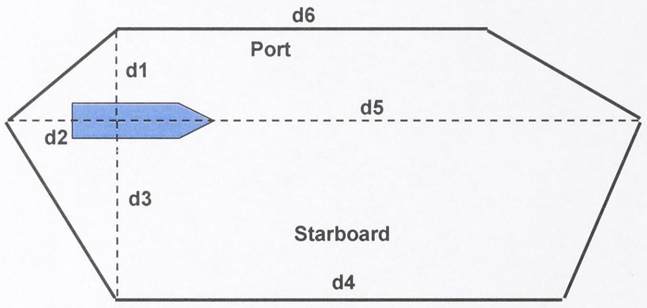


Fig. 3.7 Pentagon Object Domain

$$
\begin{aligned}
&d1 = DCPA / 2\\
&d2 = DCPA / 2\\
&d3 = B \times V^{0.44}\\
&d4 = TCPA \times V \qquad\qquad (3.5)\\
&d5 = L \times V^{1.26} + 30 \times V\\
&d6 = TCPA \times V\\
&where: L, B\ are\ ship\ lenth\ and\ breath\ [NM], V\ is\ the\ ship\ speed\ [kts]
\end{aligned}
$$

The concept of the object domain is similar to the bumper (section 3.3.2.2), except for the form and dimensions. It is easy to apply and appropriate for encounters at the open sea.

3.3.2.4 Evaluation of Collision-Risk by Obstacle Zone by Target

Another risk-assessing criterion which is gaining more and more attention recently is the Obstacle Zone by Target (OZT). The concept was initially proposed by H. Imazu and J. Fukuto in their paper in 2003 [3][4][5].

According to Imazu and Fukuto, the risk of collision between 2 ships can be represented by the possibility that these 2 ships appear at the same position at the same time. Due to the uncertainty in TS velocity as perceived by OS navigational aids, its arrival at a point is also uncertain.

Given the OS and TS as shown in Fig. 3.8, the possibility that they would collide at a

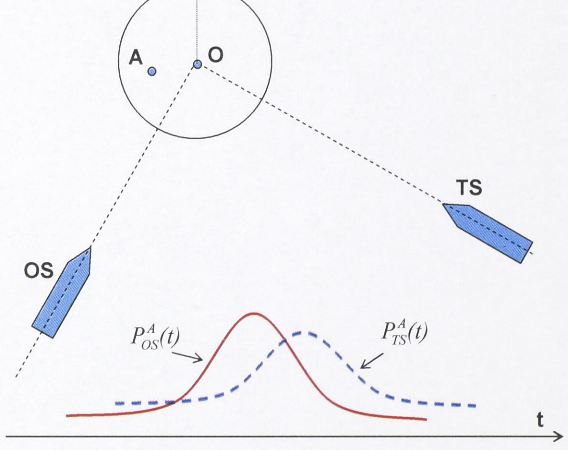


Fig. 3.8 Collision-Risk at a Point

random point A can be evaluated using the following formula (3.6)

$$P^{A} = \int_{-\infty}^{\infty} P_{OS}^{A}(t) x P_{TS}^{A}(t) dt \qquad (3.6)$$

A: The point of calculation

$P_{OS}^{A}(t)$: Probability for OS to reach the point A at time t

$P_{TS}^{A}(t)$: Probability for TS to reach the point A at time t

The probability distribution of arrival-time is usually presented by random (or Gaussian) distribution, with the bell peak lying at the time (t0) which is the time for the ship to reach the position concerned if it is actually navigating with its nominal speed. The bell spread, i.e. standard deviation of the distribution, is chosen to express the speed error.

Given a minimum distance of approaching MinDCPA, it is not expected that TS appears at any positions inside the circle centering at OS position and having radius to be MinDCPA at any time. Then, the collision-risk at a point O on the intended track of the OS is defined by (3.7)

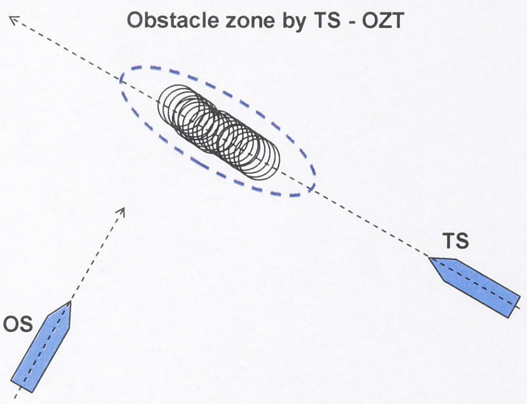


Fig. 3.9 Obstacle Zone by Target

$$P_{OZT}^{O} = Max(P^{A}) \qquad (3.7)$$

for any point A lying inside a circle centered at O with radius MinDCPA

MinDCPA: the selected minimum distance of approaching

A limiting value is selected for the collision possibility. If possibility of 2 ships arriving at a position simultaneously within a selected period (min TCPA) exceeds this limiting value, the intended track of OS is considered UNSAFE. In Fig. 3.9, positions on TS course to which the OS route is unsafe are marked by circles. The combination of those circles forms a region to which OS should not head to. It is called the Obstacle Zone by the Target and from where comes the OZT name.

The OZT criterion is simple to use and closely relates to the traditional DCPA, TCPA criterion which is generally accepted by mariners.

3.3.3 Target Motion Information

As mentioned before, target information can be extracted from Radar and AIS receiver through serial communication port in the form of NMEA sentences. The AIS data is readily in use simply by decoding those NMEA sentences. Target information can also be extracted directly, using ARPA function. In the master course, I have already worked on the target-tracking algorithms on Radar images and the combination of Radar and AIS data for better accuracy of target-tracking [8][9]. As mentioned also in Chapter 2, camera image is also a potential target motion information source even though its application is yet limited so far.

AIS data is accurate and easily accessible. However, AIS receiver is not required onboard merchant ships of less than 500 GT, pleasure boats as well as fishing ships, etc. Class B AIS is recommended for those small vessels but from maritime traffic observation in Tokyo Bay, it has been revealed that a large number of small ships have not yet been equipped with Class-B AIS. Additionally, AIS data is not always reliable, i.e. the error of AIS data, if any, can hardly be detectable. Another problem with AIS is that the update rate of ship position data does not always meet requirements.

Ship Radar, if properly used, is a very efficient and reliable source of target-motion data. Radar provides continuous images of the whole area around the OS position. The problem with Radar is that due to the poor signal reflecting characteristics, some objects do not appear on Radar screen. The Radar data (TS-motion data) are not as accurate as those received by AIS. However, it is still the best data source while navigating in coastal waters.

Camera observing system, due to camera-resolution and effective-range limitations as well as the internal error-sources of the tracking method, should be used only as a reference to the information provided by Radar, AIS.

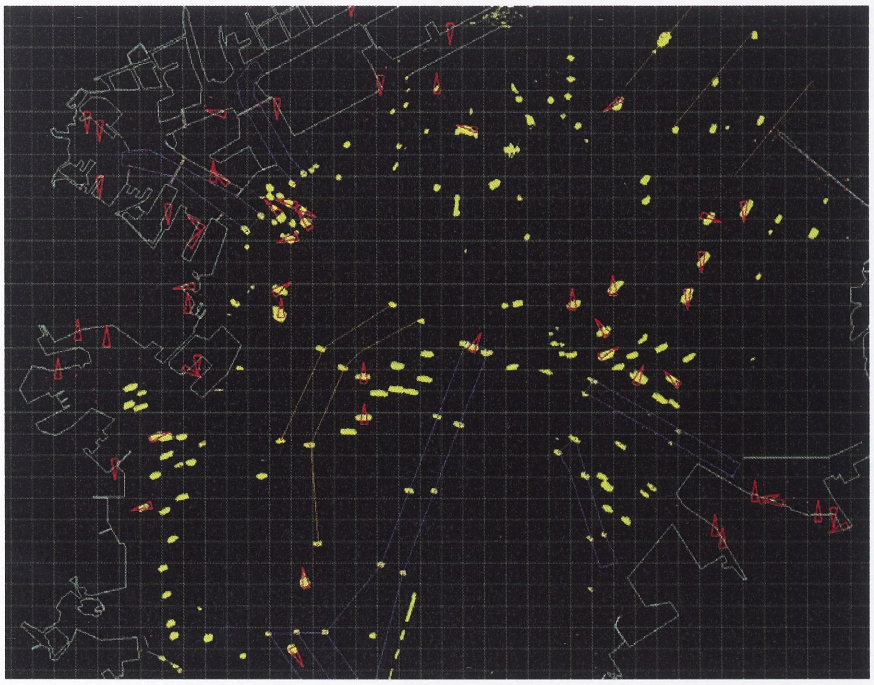


Fig. 3.10 Radar and AIS Data of Targets in Tokyo Bay

From the above navigational aids, target position, speed and course over ground can be deduced.

No matter what method used, there is always some error in TS motion detection. Then, suitable filtering algorithms should be used to remove these errors before applying TS data to produce route. In [8][9], we proposed a Kalman Filter for radar targets. Taking into consideration the slow speed change of TS, a simple low-pass filter or a moving-average filter can also be used to provide smoothed value of TS speed and course over ground.

After filtering, there are still some uncertainties accompanying with TS motion. Then, for an absolutely safe passage, future TS position is must be presented with an uncertainty-ellipse like that shown in Fig. 3.11. Further simplifying this, the error-ellipse can be replaced with an equivalent circle of error.

For instance, with the value of course uncertainty chosen to be 2 degree and speed uncertainty to be 2 % of speed, the error-circle radius at a time t in the future, R(t), is then calculated as following (3.8):

$$\delta_{Cog} = 2^o$$
$$\delta_{Sog} = 0.02 \qquad (3.8)$$
$$R(t) = Sog \times t \times \sqrt{\delta_{Cog}^2 + \delta_{Sog}^2}$$

where t is the elapse time from the moment data are acquired.

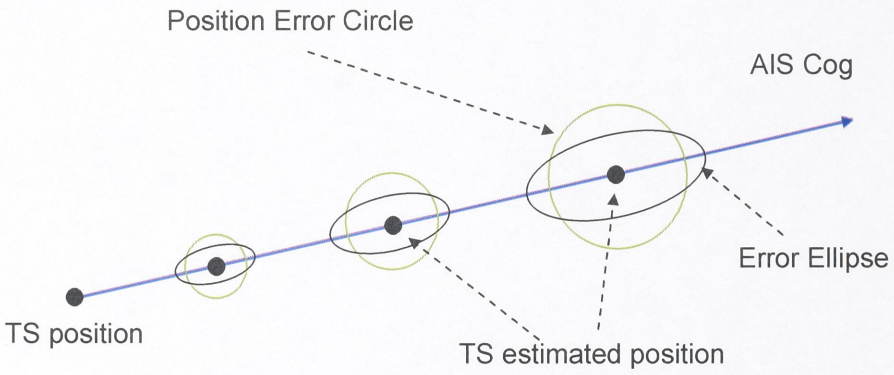
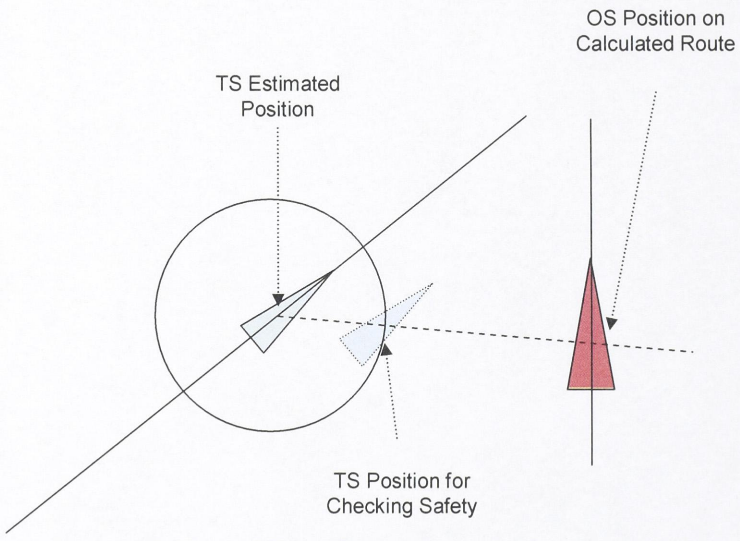


Fig. 3.11 Target Motion Uncertainty

Then, for safety checking while generating route, TS position is deemed to be its estimated position shifted to the boundary of the error-circle (for example in the direction from TS to OS as it is normally the most dangerous position of TS inside the error circle if bumper model or object domain is used) (see Fig. 3.12). The error circle is moving with TS and gradually increasing in size. If TS keeps on staying inside this error circle, its track is still safe for OS passing. Once it penetrates out of the circle, the safety of OS collision-avoiding strategy is in doubt. In that case, the strategy should be rechecked and if necessary, a new collision-avoiding strategy must be produced.



OS Position on Calculated Route
TS Estimated Position
TS Position for Checking Safety

Fig. 3.12 Use of Data Uncertainty in Risk Assessing

3.3.4 Own Ship Maneuvering Model

In maneuvering to avoid collision, apart from course, OS speed changes as a result of additional drag-force due to the rudder action and changes in the hull force also. These factors must be adequately taken into consideration. Then, a maneuvering model must be used for the OS. Throughout this study, an MMG model will be used for the OS. Basic characteristics of the ship model are as followings

Type:	Container ship
Lpp:	94[m]
Width:	15.0[m]
Draft:	6.516[m]
Block Coefficient:	0.71
Rudder Area:	7.32 [m^2]
Rudder Height:	4.89 [m]
Rudder Width:	1.5 [m]
Propeller Diameter:	3.57 [m]
Propeller Pitch:	2.36 [m]
Design Speed:	7.0 [m/s]

A first simplification is the assumption that OS forward speed does not change during maneuver. For a surface ship moving at constant speed and small rudder angle, it has been proven that its maneuvering model can be decoupled from forward speed and has the following form (Fossen 1994, 2002, etc. [12]):

$$M\dot{\nu} + N(u_0)\nu = b\delta \qquad (3.9)$$

More specifically, the following matrix equation can be written:

$$\begin{bmatrix} -Y_\delta \\ -N_\delta \end{bmatrix}\delta = \begin{bmatrix} m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ mx_g - Y_{\dot{r}} & I_z - N_{\dot{r}} \end{bmatrix}\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} Y_v & (m - X_{\dot{u}})u_0 - Y_r \\ (X_{\dot{u}} - Y_{\dot{v}})u_0 - N_v & (mx_g - Y_{\dot{r}})u_0 - N_r \end{bmatrix}\begin{bmatrix} v \\ r \end{bmatrix} \qquad (3.10)$$

where:

v, r: ship sway-velocity and yaw-rate

δ: rudder angle

Other parameters are coefficients of the inertial matrix, Coriolis-Centrifugal matrix, Damping matrix (linear) etc. (refer to [12] for more details).

Equation (3.9) can be re-written in the form:

$$\dot{\nu} = -M^{-1}N(u_0)\nu + M^{-1}b\delta \qquad (3.11)$$

Then, from (3.10), the following 6 parameters model is deduced.

$$\begin{bmatrix} \dot{v} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} v \\ r \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}\delta \qquad (3.12)$$

Model (3.12) is used to approximate the ship turning trajectory in maneuver. The model parameters can be estimated from experiments. For the MMG model in this study, the Least Square Method for System Identification could be applied. The MMG model is maneuvered with different rudder angle and values of v, r, δ are saved, along with their derivatives. The fitness between the MMG model and this simplified linear model is acceptable for applications where the rudder action and course changes are small. The maximum position mismatch is be less than 40 [m] and heading difference is less than 5 [deg] for course changes up to 60 [deg].

However, the determination of coefficients in practice is extremely challenging, especially with the presence of waves as well as other disturbances and the errors of measuring tools themselves. Even if it is possible, the assumption of constant forward speed is weak because the actual speed reduction may be up to 10%. It also takes time for the OS to regain the original speed after the course changing. The accumulated position error is therefore not negligible for longer period of calculation.

Then, the OS maneuverability should be modeled in other forms. In this study, instead of determining the linear model coefficients in (3.12), another approach, which has been proven to be much more appropriate, is to save the OS trajectories and speed-evolutions during the course change process as a whole.

For simplicity, the route is produced with just 3 rudder-command angles: small rudder command (5 [deg]), medium rudder command (10 [deg]) and heavy rudder command (15 [deg]). Practically, for collision-avoidance at sea, the ship officer usually chooses rudder command less than 15 [deg], except for critical cases.

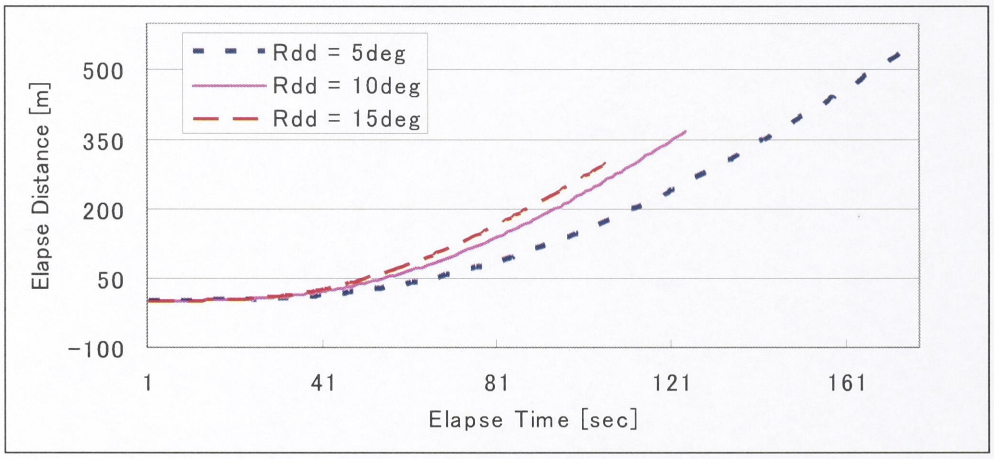


Fig. 3.13 Longitudinal Distance Variations

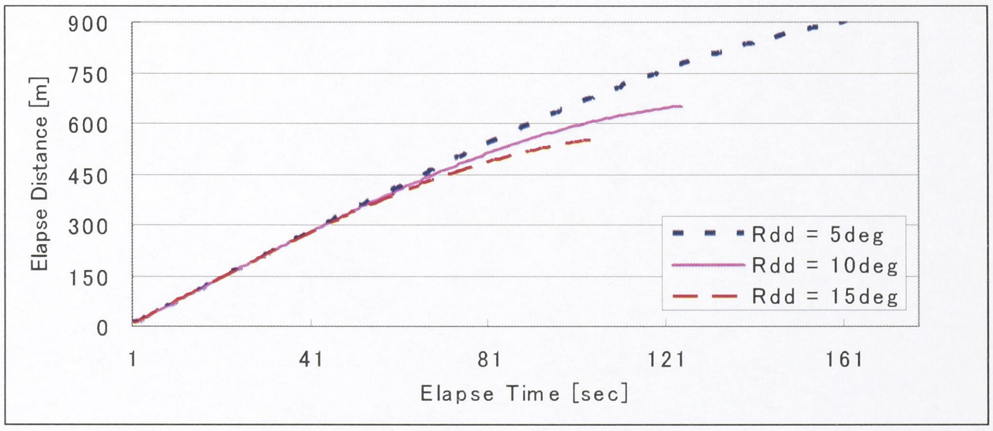


Fig. 3.14 Traverse Distance Variations

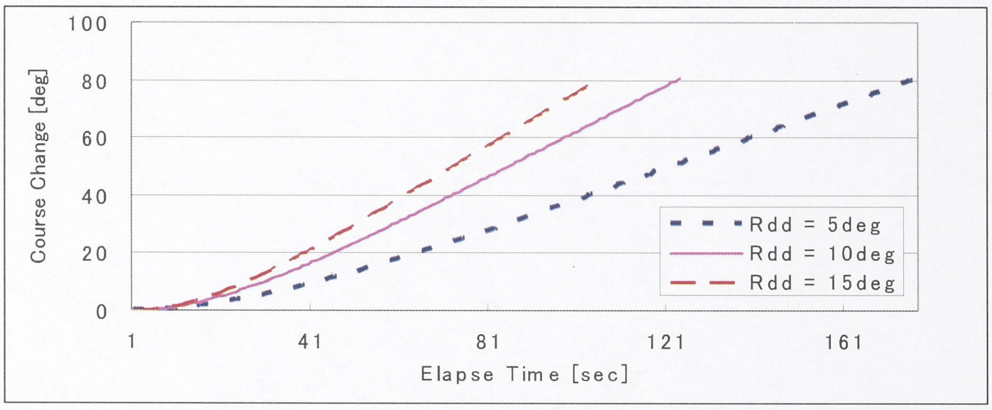


Fig. 3.15 Course-Changing Characteristics

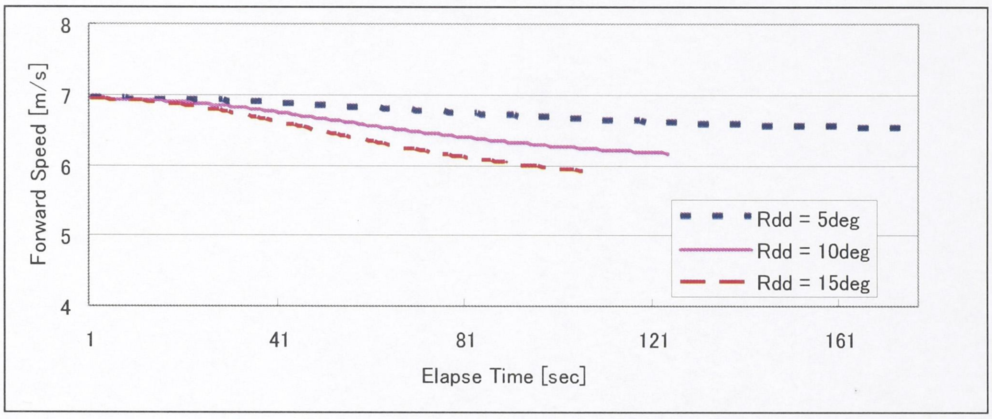


Fig. 3.16 Speed Reduction during Turning

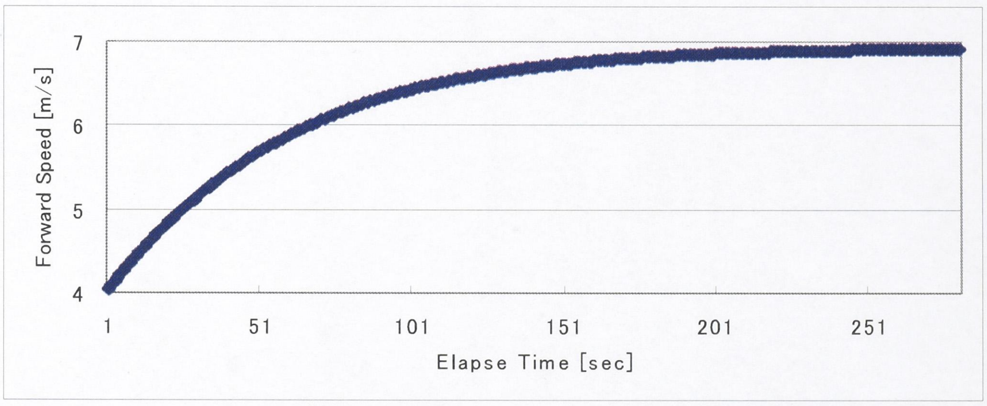


Fig. 3.17 Speed-Increasing Characteristic

Shown in figures 3.13 to 3.16 are the trajectories, speed and course changes of the MMG model for the 3 rudder angle commands. Instead of saving directly the trajectory, its 2 components are collected independently, namely the longitudinal elapse distance (Fig. 3.13) and the lateral elapse distance (Fig. 3.14) against elapse time. In the same manner, course changes (Fig. 3.15) and speed changes (Fig. 3.16) against time while taking maneuver are kept for route-producing.

After finishing the course change process, OS speed gradually increases to its equilibrium value. This speed increasing characteristics is shown in Fig. 3.17.

All these data will be tabulated for quick accessing to optimize the calculation time.

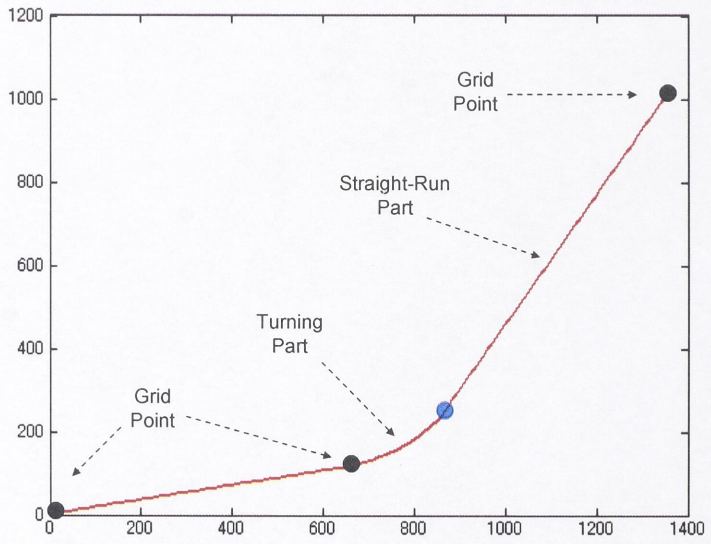


Fig. 3.18 Own Ship Path in Maneuvering

The OS path from one grid point to another is then composed of 2 parts, a turning part (course changing to destination point direction) and a straight run part (approaching the destination point).

During the course-change process, OS position at any time can be deduced from Fig. 3.13 and Fig. 3.14. The speed reduction at the end of this process is extracted from Fig. 3.16. On the other hand, OS speed and distance-run accordingly are deduced from Fig. 3.17 till it reaches the end-point. The whole process is illustrated in Fig. 3.18.

In comparison with the linearized model (3.12), the tabulation method is much preferable in the following terms:

- OS position estimation is more accurate as speed change has been included.
- The model is less affected by noise factors because derivatives-measurements are not required.

Therefore, it will be used throughout this study.

3.4 Route-Generating Algorithm by Dynamic-Programming

3.4.1 Djikstra's Algorithm for Shortest Route on Graph

Proposed by Dutch computer scientist Edsger Dijkstra in 1956, Dijkstra algorithm [1][2] is a popular graph-search algorithm that solves the single source shortest path problem for graphs with non-negative edge-path costs so as to produce the shortest path tree.

Given a graph with constant part cost between vertices (nodes), the algorithm is to find the minimum cost route i.e. a route from the initial node (starting-node), through a certain set of nodes to reach the end-node that requires minimum total cost.

Supposing that we have a graph denoted by G, the route from the starting-node to the end-node, passing through a set of N nodes is defined by G(i,j) where

$i = \overline{1, N}$

$j : index\ of\ node\ i^{th}\ of\ the\ set$

Call T(G(i,k), G(i+1,h)) the part-cost between nodes G(i,k) and G(i+1,h), and T(G(i,k)) the cost to reach node G(i,k) from the starting-node, Dijkstra's algorithm is then contrived from the equation

$$T_{min}(G(i+1,j) = \underset{k}{Min}\{T(G(i,k),G(i+1,j)) + T_{min}(G(i,k))\}$$

$$\textit{where } T_{min}(G(i,k)) \textit{ is the minimum cost to reach node } G(i,k)$$

Applying Dijkstra's algorithm, the minimum cost route to every node is determined, starting from the starting-node and expanding outward to the end-node. When the end-node is reached, the route is quoted by going back the graph from end-node to starting one.

Dijkstra's algorithm is commonly used in route-searching problems in static environment. An example is to find the shortest route between cities on a national highway network. In the following sections, the algorithm is used to determine the minimum time route for the OS to avoid collision, from a starting point to the destination.

3.4.2 Algorithm for Generating Collision-Avoiding Route

The overall algorithm of route-producing is described as in the flow chart below (Fig. 3.19). The starting-point and the end-point can be thought of as a special case of grid line with only one point on it.

Applying the equation

$$T_{min}(G(i+1,k) = \underset{j}{Min}\{T(G(i,j),G(i+1,k)) + T_{min}(G(i,j))\} \qquad (3.13)$$

the best route to a given point (point kth) on line (i+1)th is the extension of the best route to points on the previous line, i.e. line ith. For a point jth on line ith, the OS trajectory is produced by connecting this point with point kth, using a suitable OS maneuvering model which has been kept in the database and an appropriate rudder angle (see section 3.3.3).

Then, the safety of the produced track must be checked, using a collision- risk assessing criterion, with TS data and environmental constraints.

If the passage is safe, time to reach kth from point jth is compared with the time required for reaching kth from other points on line ith. The minimum time to reach this point from all possible path will be saved (T(i+1,k)). If there is no safe route to this point, T(i+1,k) is marked as infinite.

When the destination has been reached, the minimum time route can be deduced by simply walking back the graph to the starting-point. This route is deemed the collision-avoiding strategy for the OS.

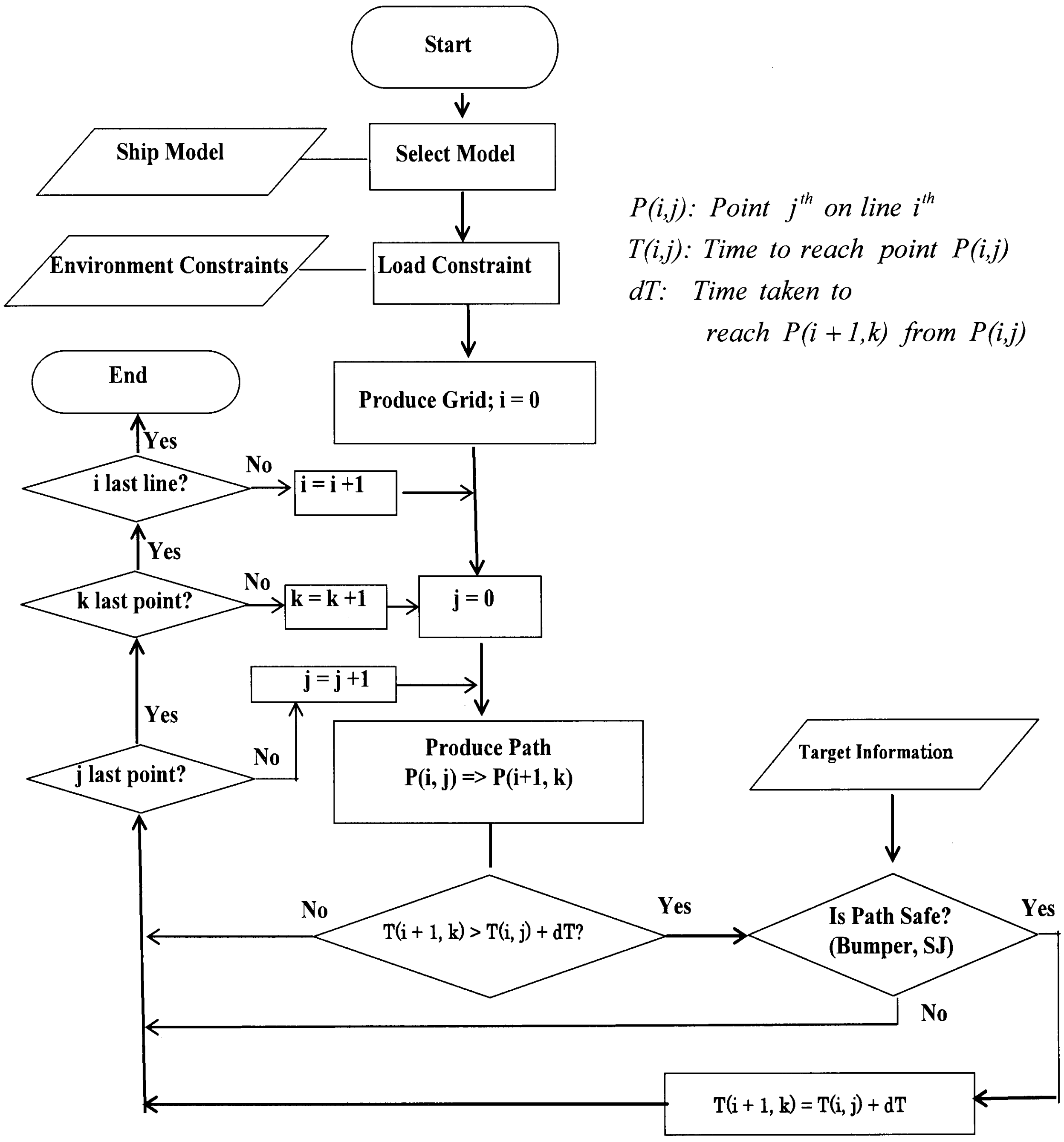


Fig. 3.19 Dynamic-Programming Algorithm

Another problem in route generation is that the OS route should not cause any misunderstanding on the TS side due to unclear maneuver. This is the case when the OS suddenly alters its course in a way that change a Port-to-Port passing to Starboard-to-Starboard passing situation, or vice verse, e.g. To reduce this possibility, several other constraints should be taken into account in producing the path, including:

- If just one target is nearby and this target is in head-on encounter, OS should not change course to port.

- OS should not alter course in a way that changes its course from passing the bow to passing the stern of the most dangerous target when the TCPA is less than 3 minutes.

- OS should not alter course in a way that changes a port-to-port passing to starboard-to-starboard passing when the TCPA is less than 3 minutes.

It can be easily proven that the route produced with Dijkstra algorithm is the optimum, i.e. minimum time, route ONLY IF the traffic environment is static (or Time-invariant). In our application, the environment is actually varying with time because the obstacles caused by TS change with TS-motions. The application of DP for route-producing can therefore just give an approximation of the optimal solution, maybe a local optimal. However, due to its simplicity, it is still widely used.

3.4.3 Examples of Route Generation

A. Example 1

This example is to illustrate the idea under the overall system. In the example, a ship is supposed to follow a planned route which is shown in light blue in Fig. 3.20. At point A on this route, the system detected the approaching of 2 ships with high risk of collision. Then the minimum time route (MTR) for collision-avoiding is calculated for a part on planned route, i.e. part AB. The result is shown in Fig. 3.21, including the MTR to every grid points and the MTR to the end-point B.

While navigating on route AB, the system detected a new collision-risk due to the approaching of 2 other TS. Thus, the route generator is activated again and the route part CD (Fig. 3.22) is produced.

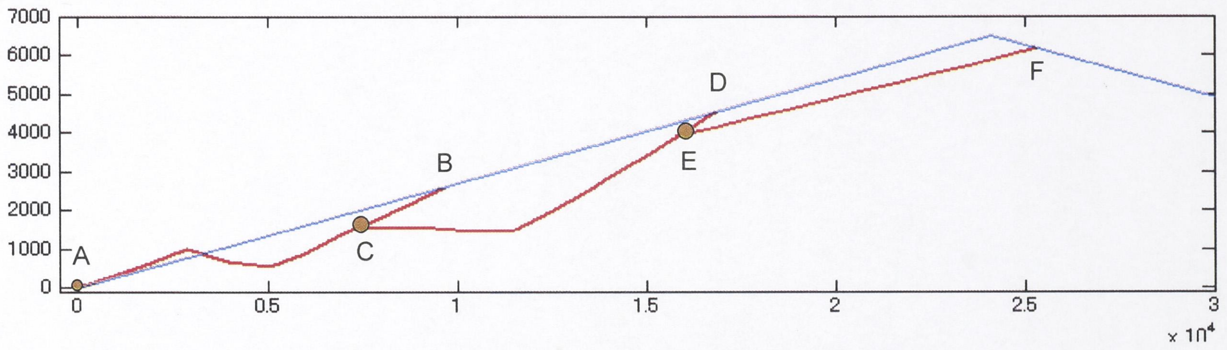


Fig. 3.20 Planned-Route and Collision-Avoiding Route

Note that the calculation end-point (point B, D e.g.) is chosen on the planned route, therefore the OS will not deviate largely from its originally planned route.

Even though there is no arsing dangers, route generator can be regularly activated to produce a better track. It is the case in which, for instance, TS-motion uncertainty is not as large as expected, or the other ship has change course in a way that risk of collision no longer exists. In Fig. 3.20, because there are no dangers, route generator simply produces a straight line.

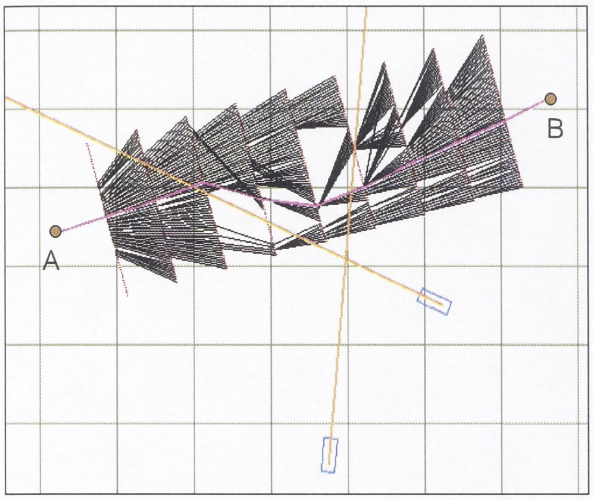


Fig. 3.21 Route Part 1

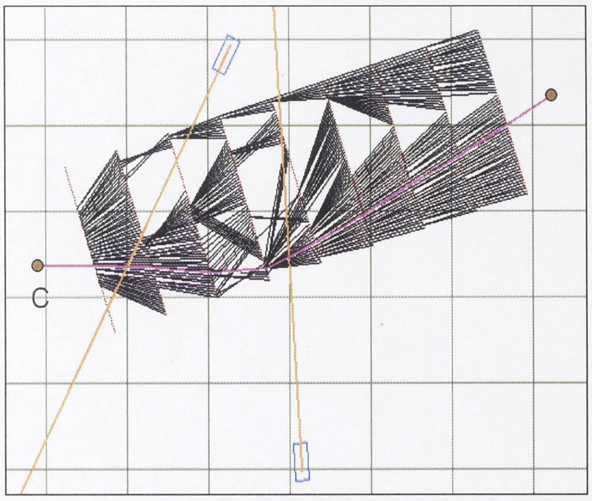


Fig. 3.22 Route Part 2

B. Example 2

The example is to express the effect of grid width on the produced collision-avoiding strategy. In this case, generator fails to determine the collision-avoiding route for the path EF due to the special distribution of dangerous targets and the grid is too narrow (Fig. 3.23). There is no safe route to reach any points on the 5th line.

A simple solution to the problem is to widen the grid, at the price of the increase in calculation volume. The route generated for the same situation, by widening the grid is shown in Fig. 3.24. The route causes OS to deviate further from the original route, but it is safe, at least.

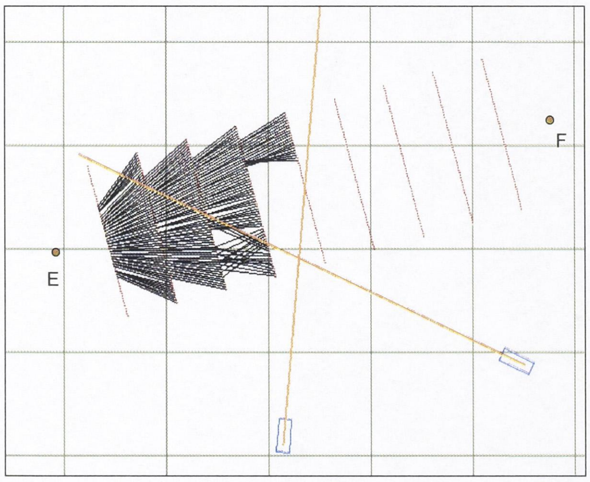


Fig. 3.23 Fail to Generate Route

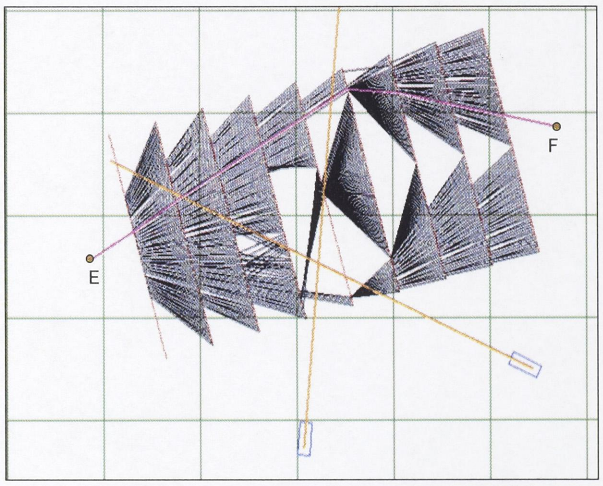
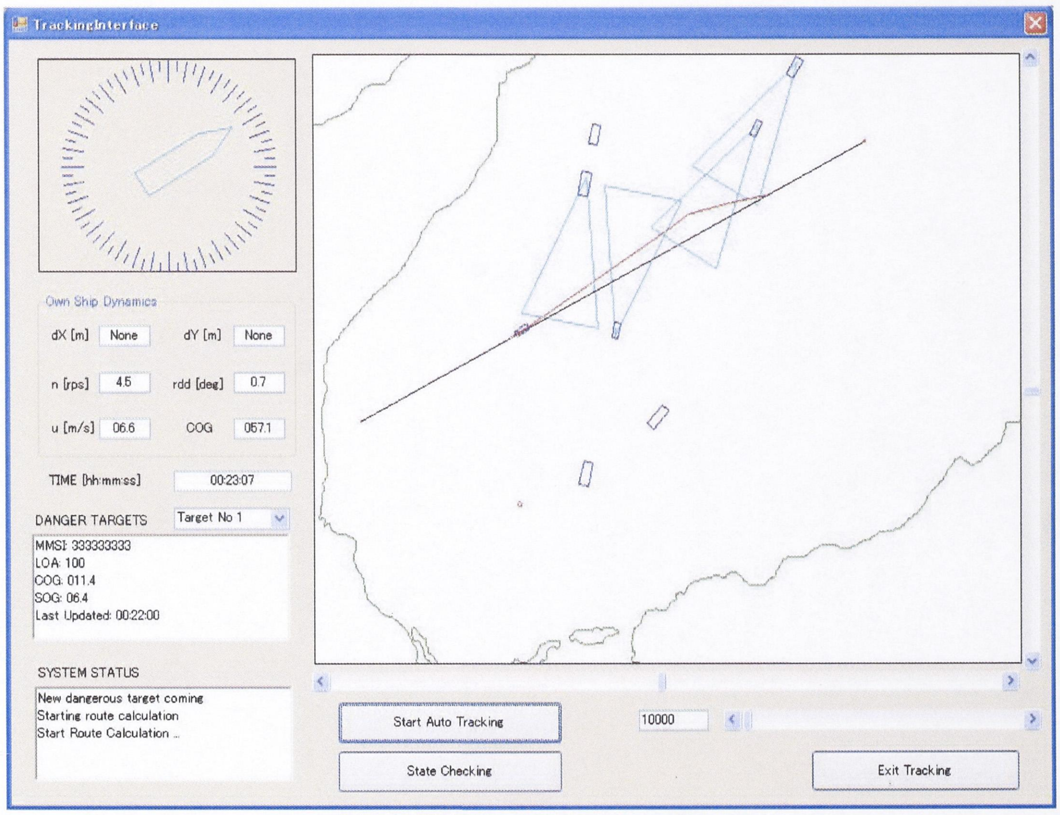


Fig. 3.24 Route Generated after Grid Widening

3.5 Simulation Studies

In this chapter, the route-producing algorithm is applied to generate collision-avoiding routes for several typical cases of marine traffic encounters. The same scenarios will later be used in simulation studies with route-producing algorithms basing on Ant Colony Optimization (ACO – Chapter 4) and Bacteria Foraging Optimization Algorithm (BFOA – Chapter 5) to provide a cross check of the algorithm validity and to reveal advantages, disadvantages of each algorithm. The OS model used is described in Section 3.3.3.

The simulation interface is similar to that shown in Fig. 3.25 in which the targets that cause imminent dangers are marked with triangular symbols on their courses.



dX [m] None dY [m] None
n [rps] 4.5 rdd [deg] 0.7
u [m/s] 06.6 COG 057.1

TIME [hh:mm:ss] 00:23:07

DANGER TARGETS Target No 1

MMSI: 333333333
LOA: 100
COG: 011.4
SOG: 06.4
Last Updated: 00:22:00

SYSTEM STATUS

New dangerous target coming
Starting route calculation
Start Route Calculation ...

10000

Fig. 3.25 Simulation Interface

3.5.1 Scenario 1 (Fig. 3.26)

In this scenario, 4 target ships are crossing OS from starboard side and port side. OS turns to its port side to avoid collision by passing starboard targets at their bows.

- Route is produced quickly.
- Collision-avoiding strategy is clear i.e. the possibility of misunderstanding from the TS side is low.
- Collision-avoiding route is acceptable from ship officer point of view.

It is advisable to refer also to section 4.4.1 (Chapter 4) and to section 5.4.1 (Chapter 5) for the result comparison with other route-producing algorithms.

3.5.2 Scenario 2 (Fig. 3.27)

In this scenario, 3 target ships are crossing from starboard, 2 targets are crossing from port and 1 target is in a head on encounter.

- Route is produced quickly.
- Collision-avoiding strategy is clear, i.e. low possibility of misunderstanding.
- OS passes the head-on encountering target and crossing targets all on its starboard side, the route is therefore UNUSUAL, from ship officer's view-point.

It is advisable to refer also to section 4.4.2 (Chapter 4) and to section 5.4.2 (Chapter 5) for the result comparison with other route-producing algorithms.

3.5.3 Scenario 3 (Fig. 3.28)

A target is overtaking OS on OS starboard side. OS is also under the risk of collision with 2 targets crossing from starboard side and 1 target crossing form port side. OS alters course first to its port side to pass 2 starboard crossing targets and then slightly changes course to starboard to avoid collision with the port side target.

- Route is produced quickly.
- Collision-avoiding strategy is clear, and appropriate.

It is advisable to refer also to section 4.4.3 (Chapter 4) and to section 5.4.3 (Chapter 5) for the result comparison with other route-producing algorithms.

3.5.4 Scenario 4 (Fig. 3.29)

The scenario is quite difficult and worrisome for the ship officer due to the involvement of many targets crossing from different directions. A good collision-avoiding strategy can not be easily decided by the OS officer by eye-judgment or radar screen observation. However, the support system can help much in the case.

- Route is produced quickly.
- Collision-avoiding strategy is clear, and appropriate.
- The produced route may be the best for the scenario.

It is advisable to refer also to section 4.4.4 (Chapter 4) and to section 5.4.4 (Chapter 5) for the result comparison with other route-producing algorithms.

3.5.5 Scenario 5 (Fig. 3.30)

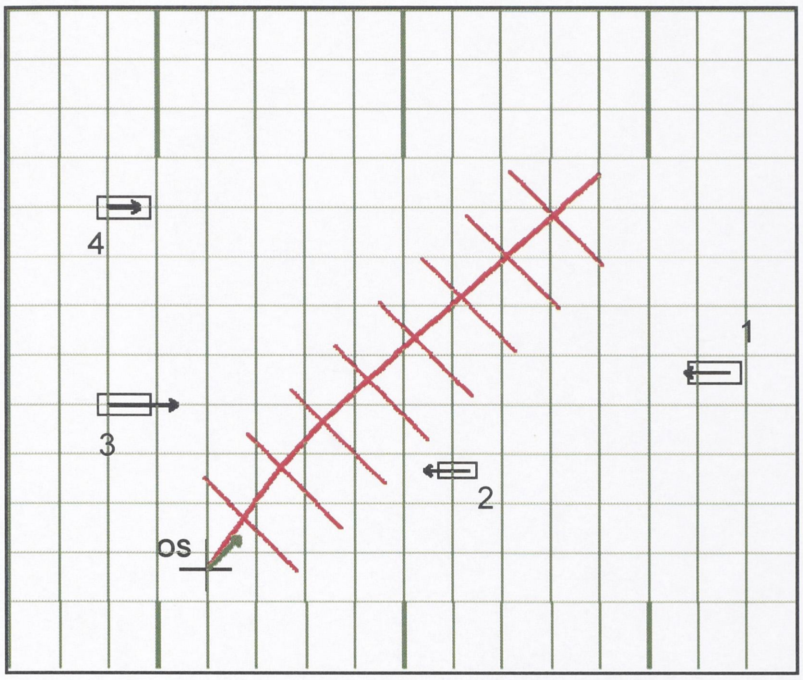
OS is overtaken by a target on OS port side. It in turn is overtaking a target on its starboard. The scenario is even more risky due to the involvement of 4 other targets coming from different directions. In this case also, an appropriate collision-avoiding strategy can not be easily decided by the OS officer from observations. The support system on the other hand can still produce a collision-avoiding route promptly. The route is theoretically safe. However, it can hardly be acceptable because OS is altering course dangerously toward the overtaking TS (TS5). Further more, the collision-avoiding strategy is not highly recommendable as OS is passing port side targets on their sterns and starboard side targets on their bows.

It is advisable to refer also to section 4.4.5 (Chapter 4) and to section 5.4.5 (Chapter 5) for the result comparison with other route-producing algorithms.

3.5.6 Scenario 6 (Fig. 3.31)

The scenario is also difficult, mostly due to the size of the targets involved and the given width of the grid. The OS need to avoid the collision-risk with an overtaking vessel on its starboard simultaneously. For the scenario, the algorithm fails to produce a collision-avoiding route for the OS. It is because of the fact that the algorithm keeps on seeking the shortest route to every point on every grid line, starting from its initial position, and this strategy may navigate the algorithm to inescapable positions (see 3.4.3.2).

It is advisable to refer also to section 4.4.6 (Chapter 4) and to section 5.4.6 (Chapter 5) for the result comparison with other route-producing algorithms.

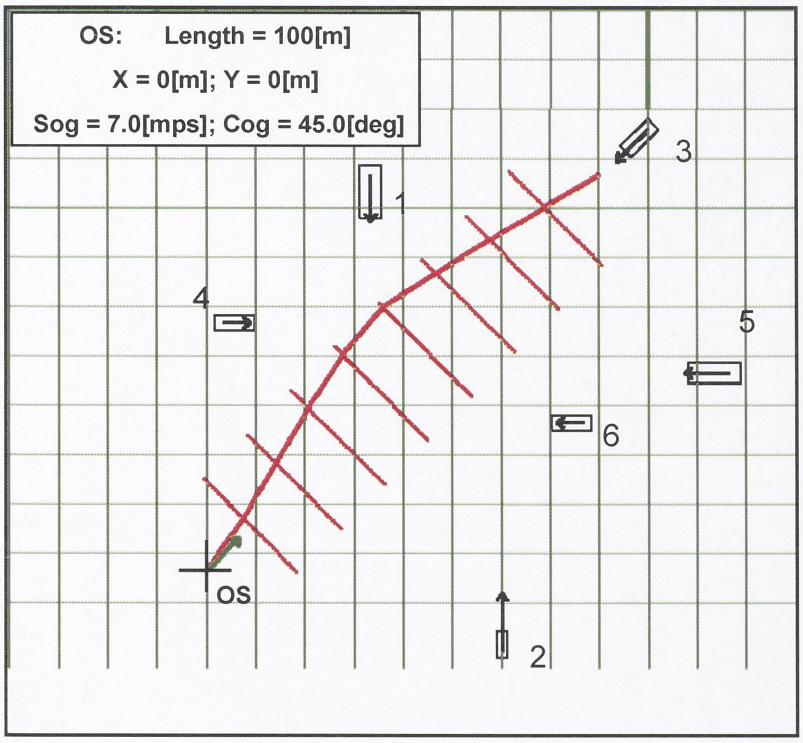


OS: Length = 100[m]
X = 0[m]; Y = 0[m]
Sog = 7.0[mps]; Cog = 45.0[deg]

TS	X [m]	Y [m]
1	16000	6000
2	8000	3000
3	-3000	5000
4	-3000	11000

TS	L[m]	Sog [mps]	Cog [deg]
1	200	7.0	270.0
2	150	7.0	270.0
3	200	10.5	90.0
4	200	4.9	90.0

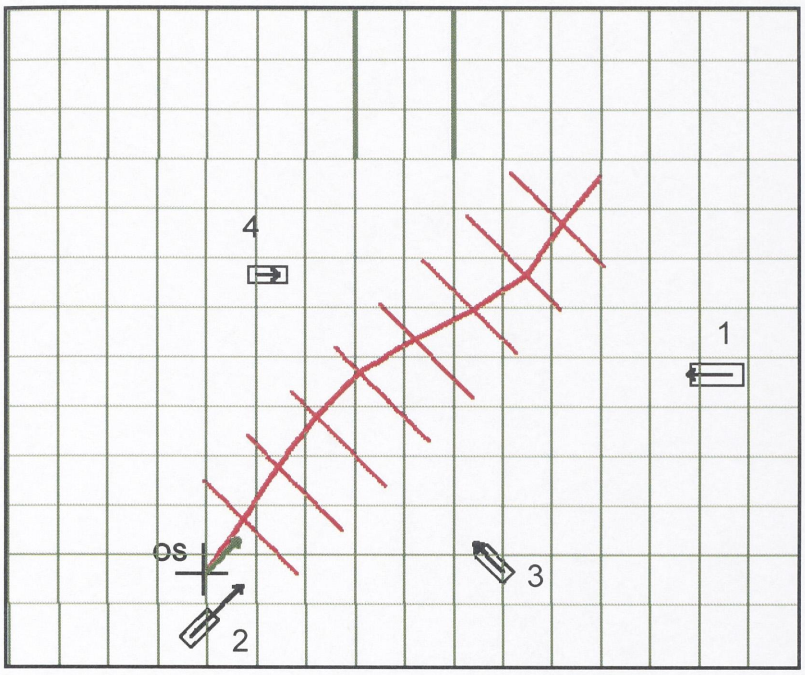
Fig. 3.26 Scenario 1



TS	X [m]	Y [m]
1	5000	12000
2	9000	-2500
3	13500	13400
4	500	7500
5	16000	6000
6	11500	4500

TS	L[m]	Sog [mps]	Cog [deg]
1	200	7.0	180.0
2	100	9.1	0.0
3	150	7.0	225.0
4	150	4.2	90.0
5	200	7.0	270.0
6	150	4.2	270.0

Fig. 3.27 Scenario 2

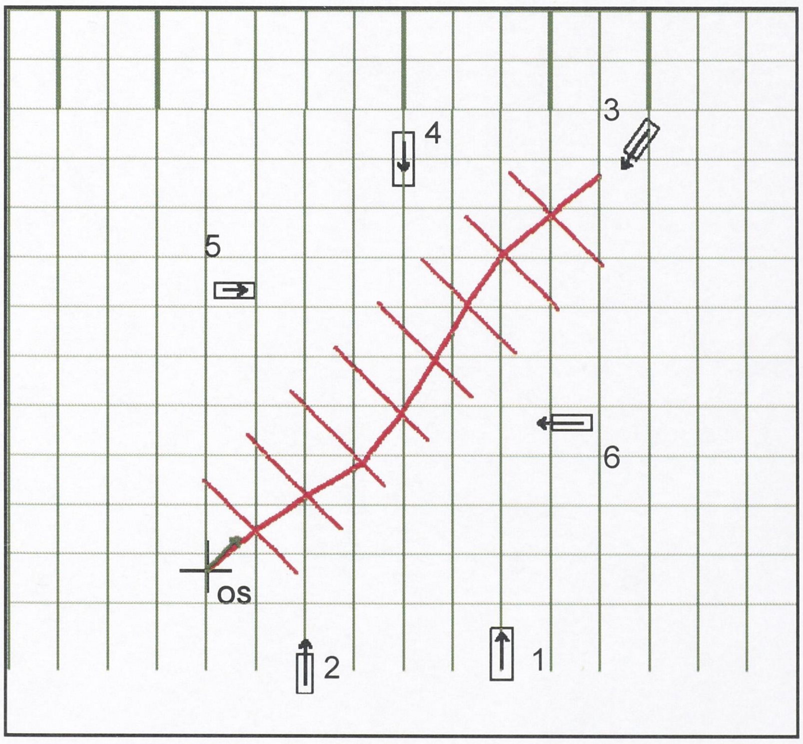


OS:	Length = 100[m]
	X = 0[m]; Y = 0[m]
	Sog = 7.0[mps]; Cog = 45.0[deg]

TS	X [m]	Y [m]
1	16000	6000
2	-500	-2000
3	9000	0
4	1500	9000

TS	L[m]	Sog [mps]	Cog [deg]
1	200	7.0	270.0
2	150	11.2	45.0
3	150	6.3	315.0
4	150	3.5	90.0

Fig. 3.28 Scenario 3

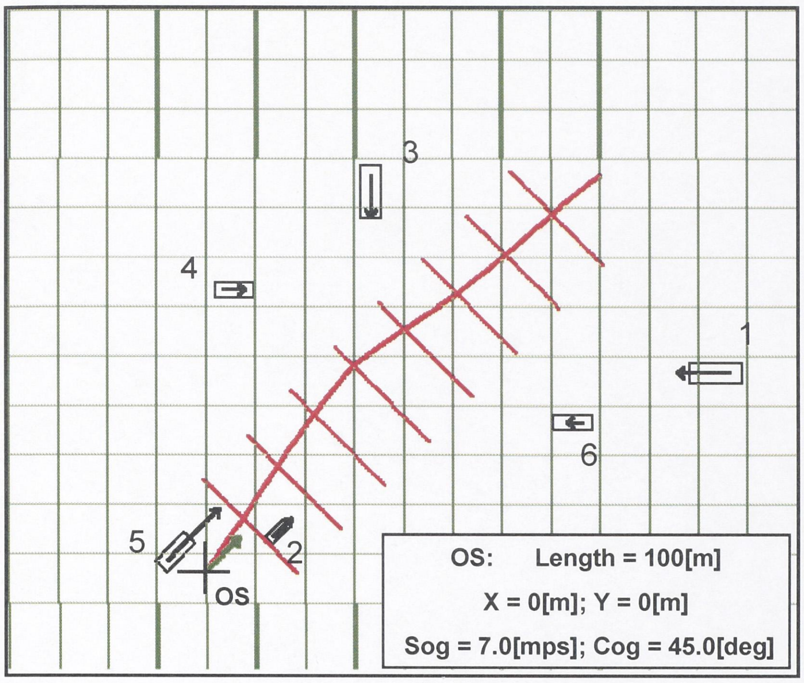


OS:	Length = 100[m]
	X = 0[m]; Y = 0[m]
	Sog = 7.0[mps]; Cog = 45.0[deg]

TS	X [m]	Y [m]
1	9000	-3000
2	3000	-3500
3	13500	13400
4	6000	13000
5	500	8500
6	11500	4500

TS	L[m]	Sog [mps]	Cog [deg]
1	200	5.6	0.0
2	150	7.0	0.0
3	150	7.0	215.0
4	200	4.2	180.0
5	150	3.5	90.0
6	150	7.0	270.0

Fig. 3.29 Scenario 4



OS: Length = 100[m]

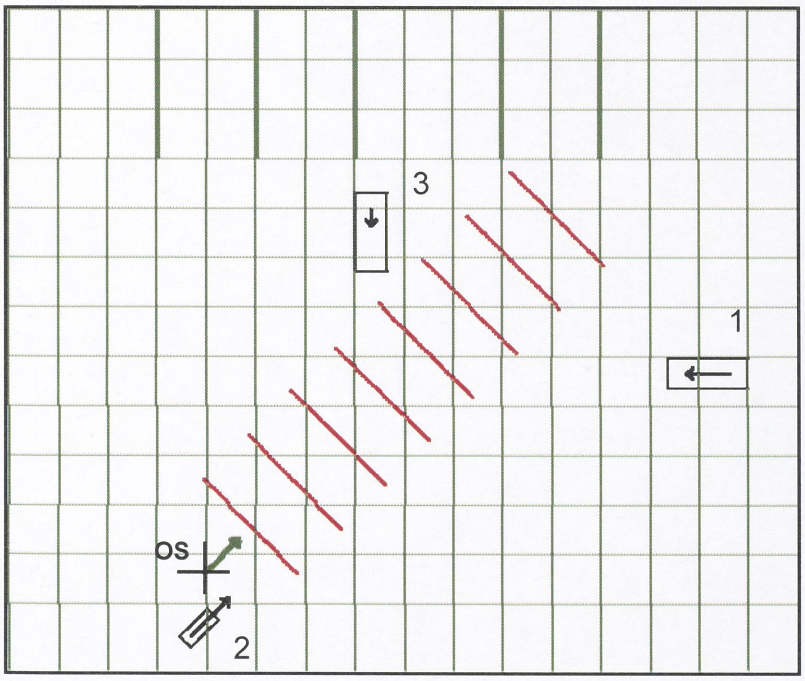
X = 0[m]; Y = 0[m]

Sog = 7.0[mps]; Cog = 45.0[deg]

TS	X [m]	Y [m]
1	16000	6000
2	2000	1000
3	5000	12000
4	500	8500
5	-1200	300
6	11500	4500

TS	L[m]	Sog [mps]	Cog [deg]
1	200	8.4	270.0
2	100	4.2	45.0
3	200	6.3	180.0
4	150	3.5	90.0
5	150	1.6	45.0
6	150	0.4	270.0

Fig. 3.30 Scenario 5



OS: Length = 100[m]

X = 0[m]; Y = 0[m]

Sog = 7.0[mps]; Cog = 45.0[deg]

TS	X [m]	Y [m]
1	16000	6000
2	-500	-2000
3	5000	11000

TS	L[m]	Sog [mps]	Cog [deg]
1	300	7.0	270.0
2	150	8.4	45.0
3	300	2.8	180.0

Fig. 3.31 Scenario 6

3.6 Conclusion

In this chapter, the configuration of an automatic navigating system is introduced and different application aspects have been analyzed in details. Experiments and simulation studies have revealed that:

- Dynamic-Programming algorithm is a simple but efficient algorithm for generating the minimum time route for OS. The route ensures a safe passage of OS, given the TS motions and environmental constraints.

- The calculation volume of DP algorithm is less than those required in revolutionary algorithm, thus calculation time is shorter than that of ACO algorithm and BFOA algorithm.

- It is complicated to apply the rules of the road (Colreg 72) because passage time is the single quality index for route assessing at each grid points.

- The solution produced, using DP is NOT always the global optima because of the fact that the traffic environment is not static. Due to the TS-motions, cost of a trajectory from 1 point to another changes with time.

- Different criteria can be used for risk assessment. SJ value and Bumper Model are more suitable for ships navigating in congested waters. Ship Domain criterion is more suitable for open sea passages while OZT can be used for either cases with proper choice of DCPA.

- OS maneuvering characteristics should be tabulated for easy access and to ensure the OS estimated position accuracy.

- A proper choice of Grid designing parameters allows the collision-avoiding route to be produced quickly and accurately.

The system, if further developed, is practicable and would provide the possibility of totally automatic navigation. It can thus reduce the work load of navigator and increase the safety and efficiency of navigation.

References

1. A. E. Bryson, Jr. C.H. Yu, Applied Optimal Control , pp.129-176, 1991
2. D. P. Bertsekas, "Dynamic Programming and Optimal Control: 2nd Edition", Athena Scientific, 2000
3. H. Imazu and J.Fukuto, "The Obstacle Zone by Target and Evasive Area", 11th IAIN World Congress, 2003
4. H. Imazu, J.Fukuto and M.Numano, "Obstacle Zone by Target and its Expression", The Journal of Japan Institute of Navigation, Vol.107, pp.191-197, 2002
5. H. Imazu, T.Fujisaka, J.Fukuto and Y.Otake, "Study of the Integration and Presentation of Navigational Information", The Journal of Japan Institute of Navigation, Vol.109, pp.133-140, 2003
6. J.S. Zelek, "Dynamic path planning", Proceedings of IEEE Conference on Systems, Man and Cybernetics, pp. 1285-1290, 1995
7. K. Hara, "Proposal of Maneuvering Standard to Avoid Collision in Congested Sea Area", The Journal of Japan Institute of Navigation, Vol.85, pp.33-40, 1991
8. M. D. Nguyen, "A Study on the efficiency enhancement of automatic radar tracking and analyses of marine traffic in Tokyo Bay", Tokyo University of Marine Science and Technology, 2009
9. M. D. Nguyen, H. Tamaru, "Application of Kalman Filter to Rebuild Targets

Movement from Radar Images of Marine Traffic in Tokyo Bay", Asia Navigation Conference, pp.115-123, 2008

10. R. Smierzchalski, Z. Michalewicz, "Modeling of Ship Trajectory in Collision Situations by an Evolutionary Algorithm", IEEE Transactions on Evolutionary Computation, Vol. XX, 1998, available at "http://cs.adelaide.edu.au/~zbyszek/Papers/p41.pdf"
11. S. M. LaValle, "Planning Algorithms", 2004, available at http://msl.cs.uiuc.edu/planning/
12. T. I. Fossen, "Marine Control System", Marine Cybernetics, pp. 172-200, 2002

Chapter 4 Collision-Avoiding Route Generation by Ant Colony Optimization

4.1 Introduction

The route-producing algorithm basing on Dynamic-Programming method is quite effective and easy to apply. However, it is neither the errorless nor the unique solution. The major limitations pertaining to this method can be listed as the followings:

- Firstly, the application of the rules of the road has not been properly covered. This is, unfortunately, the very first matter of concern for experienced officer in deciding if a collision-avoiding strategy is acceptable.
- Secondly, the algorithm is based on the assumption that the collision-avoiding problem is time-invariant. It is actually not the case as the obstacles presented by moving ships are time-varying. As a result, the solution produced may not be the optimal solution or even an approximation of the optimal.
- Thirdly, due to the nature of DP method that attempts to reach every node by shortest route, the algorithm is incapable of producing collision-avoiding route for certain cases (see 3.5.6).

A better route-producing algorithm should therefore be constructed so as to overcome these difficulties.

Getting along with the fast improvement of computer processing speed, population-based optimization methods inspired by nature phenomena have been gaining a lot of popularity in the last several years. These nature inspired algorithms are of great interest and deemed to be potential solutions to many real world optimization problems which have become too large, complex and dynamic to be covered by available analysis or numerical methods. Then, they require the development of solving methods of which the efficiency is measured by their ability to find acceptable result within a reasonable amount of time, rather than the ability to ensure an optimal solution. Some optimization-algorithms mimicking behaviors of natural spices have been proposed and found their application in different computational fields, including Genetic Algorithm (GA) ([10], Appendix II), Particle Swarm Optimization (PSO) ([9], Appendix III), and Artificial Bee Colony Optimization (ABC) etc.

Belonging to the family of nature-inspired optimization algorithms, Ant Colony Optimization (ACO) algorithm has proven its surpassing performance, compared to GA, PSO etc. in a vast realm of problems. ACO is the result of research works on computational intelligent approaches to combinatorial optimization originally conducted by Marco Dorigo, in collaboration with Alberto Colorni and Vittorio Maniezzo [3][4][5][6]. Since then, different ACO algorithms have been successfully developed in many hard combinatorial problems such as the traditional Travelling Salesman problem, NP-Hard problems, the problem of Data Network Routing and a number of other optimization problems in engineering applications [1][2][7][8].

Then, in this study, an algorithm for generating the optimal or an approximation of the optimal collision-avoiding route for the Own Ship (OS) will be proposed, given the Target Ship (TS) motions, OS maneuvering characteristics and environmental constraints. The algorithm proves to be very efficient in quick route-producing and allows the realization of the maritime traffic laws as set-forth in International Convention for Preventing Collision at Sea. It also overcomes the limitation of Dynamic-Programming Algorithm in treating the problem as static. With suitable choice of designing parameters, the algorithm provides admirable exploration and exploitation capacities.

In the rest of the chapter, foraging-behavior of ants and classical ACO algorithms will be described in Section 4.2. Section 4.3 deals uniquely with the ACO algorithm for producing the collision-avoiding route and its application-aspects in details. Simulation studies are discussed in Section 4.4 to verify the performance of ACO algorithm. Then, the chapter conclusions are summarized in Section 4.5.

As mentioned earlier, the algorithm will be proposed with the following 2 assumptions:

- Target Ships do not change their speeds and courses during the collision-avoiding process.

- Collision-avoidance is the duty of our Own Ship alone, even for the cases where it is a stand-on vessel.

4.2 Behavior of Ants and Ant Colony Optimization Algorithms

4.2.1 Foraging Behavior of Ants and Optimization Problem

Ant colonies are distributed systems that, in spite of the simplicity of their individuals, show a highly structured social organization. The strict organization enables the colonies to accomplish complex tasks which far exceed a single ant's capacities such as the division of labor, brood sorting, cooperative and foraging tasks.

The collective behavior of an ant colony can be illustrated as shown in Fig. 4.1. Assuming that an ant colony is nesting at N (Nest) and there is food source for ants located at F (Food), an interesting phenomenon has been experienced as followings:

- The first ants find a way to reach the food source (e.g. path a) and return to their nest, using path b. On the latter path, they leave trail-pheromone.

- In the beginning stage, ants use one among the four possible paths equally and randomly. However, the discrimination between them gradually increases as more ants finish their food-source searching and home returning cycles. More ants tend to choose the shorter path to reach the food source and return back. The trail-pheromone levels of these paths make them more attractive than other paths.

- After a period of exploiting, almost all ants choose the shortest path to the food source. Long portions of other paths loose their pheromone trail.

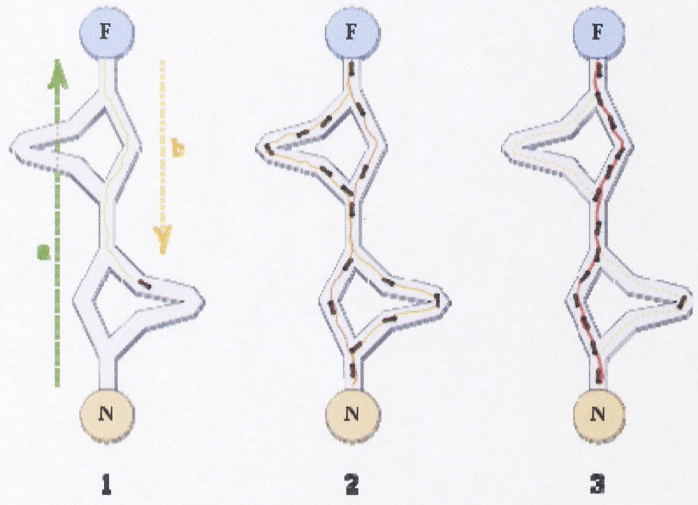


Fig. 4.1 Ant Colony Behavior (wikipedia)

The above mentioned phenomenon is a generalization of the double-bridge experiment in which ants converge to the bridge that provides a shorter path (among 2 possible paths). The reality is, however, more complicated

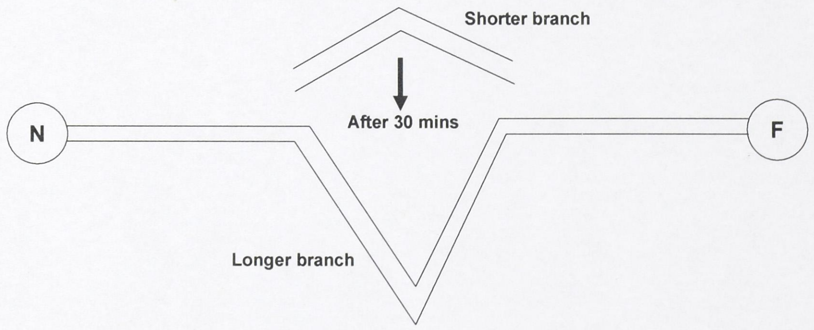


Fig. 4.2 Double Bridge Experiment

than that. In another experiment, only a long branch was initially offered to the colony. After 30 minutes, when a stable pheromone trail has formed on the only available branch, a new, shorter branch is added. The ants keep on accessing the food on the longer branch. This can be explained by the high pheromone concentration on the longer branch and the slow evaporation of the pheromone. As a majority of ants continue using and leaving pheromone on this path, its attractiveness is still strongly intensified.

The so far described behavior of the ant colony can be explained if the approximated ant behavior model as the followings is utilized:

- An ant runs more or less randomly around the nest.
- If it discovers a food source, it returns directly to the nest, leaving trail-pheromone on the path used.
- Nearby ants are attracted to follow this track more or less directly, due to the trail-pheromone.
- These ants in turn strengthen the path i.e. the pheromone level.
- If there are 2 paths to reach the same food source, the shorter one is used more frequently in a given time period. The latter path is therefore increasingly enhanced and becomes more attractive.
- As pheromone gradually evaporates, the long path eventually disappears.
- Eventually, all the ants choose the shortest path which is probably the only path left.

4.2.2 Ant Colony Optimization Meta-Heuristic Algorithm

As stated earlier, ACO algorithm is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. The problems are commonly characterized by the following features:

- A finite set of nodes is given.
- A finite set of possible connections are established between the nodes.
- There is a finite set of connection-costs, respective to the set of connections, i.e. a cost for each connection.
- A set of constraints is given for the problem.
- A solution of the problem is a route from a certain node, i.e. starting-node, passing through the connections in the connection set, to a given destination which is commonly called the end-node.
- The cost of a solution is the total connection-costs for travelling a route from starting-node to the end-node.
- The optimal solution is the solution which has minimal (or maximum) cost.

For this class of problems, the meta-heuristic ACO is proposed by Dorigo as followings [4]:

```
Procedure ACO_meta_heuristic()
    While (termination_criterion_not_sastisfied)
        Schedule_activities
            Ants_generation_and_activity();
            Pheromone_evaporation();
            Daemon_actions();
        End schedule_activities
    End while
End procedure
```

```
Procedure Ants_generation_and_activity()
    While (available_resources)
        Schedule_the_creation_of_a_new_ant();
        New_active_ant();
    End while
End procedure

Procedure New_active_ant()
    Initialize_ant();
    M = update_ant_memory();

    While (current_node <> end_node)
        A = get_local_connections_pheromone_data();
        P = compute_transition_probabilities(A, M);
        next_node = apply_ant_decision_policy(P);
        move_to_next_node(next_node);

        if (online_step_by_step_pheromone_update)
            deposit_pheromone_on_the_visited_connection();
            update_connections_pheromone();
        end if
        M = update_ant_memory();
    End while

    if (online_delayed_pheromone_update)
        for each visited_connection
            deposit_pheromone_on_the_visited_connect();
            update_connections_pheromone();
        next
    end if

    kill_ant();
End procedure
```

In this meta-heuristic algorithm, the Daemon action is optional. It can be considered as a mechanism to modify the pheromone state of the system compulsorily apart from the gentle, nature-like modification by deposit and evaporation processes so as to avoid the early convergence that may arise, e.g. the failure of intensifying the shorter branch in the double-bridge experiment mentioned above (Fig. 4.2).

It should also be noticed that there are 2 strategies of depositing pheromone for ants. Pheromone can be deposited right after a connection is used or it is just performed after the ant has reached the end-node.

4.2.3 Common ACO Algorithms

Sharing the common principle as described in ACO Meta-Heuristic, different ACO algorithms have been proposed and applied in computational and engineering problems. They are discriminated in the way a following-node is chosen and/or the trail-pheromone manipulation schemes [11][12][13].

The first algorithm is probably the Ant System (AS) algorithm. In AS algorithm, a node, or connection to it from the current node equivalently, is selected purely probabilistically, basing on the pheromone level of the connection and its desirability. The total effect of these is referred to as the attractiveness of the connection.

An improvement of AS algorithm is the Ant Colony System (ACS) algorithm. In the latter algorithm, the node-selecting mechanism depends on a chosen parameter q_0 ($0 < q_0 < 1$) and a randomly produced variable q:

```
q = random(0,1)
if (q < q0)
        Choose_most_attracive_connection(); //most attractive node
else
        Choose_node_probabilistically();    // like AS
end if
```

The tactic underlying this strategy is to spend more efforts on the most attractive connection so far. This allows the system to further exploit the search-space in the most promising regions that has been explored.

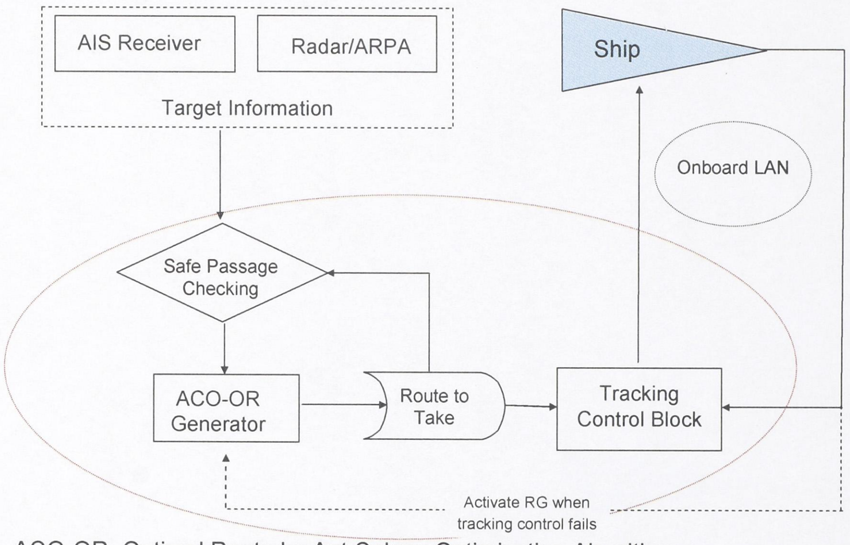
Another improvement to the AS algorithm is the MAX-MIN Ant System (MMAS). In comparison with original AS, MMSA differs in the following features:
(i) Only the best ant adds pheromone trails.
(ii) The minimum and maximum values of the pheromone are explicitly limited.
MMAS retains a balance between exploration and exploitation. (i) helps to reduce the noise effect or the chaos caused by ants which have poor performances. With (ii), no connection either receives too much attention or is totally eliminated from choice. Then, all search-space can be properly explored.

4.3 Collision-Avoiding Route Generation System Based on ACO

4.3.1 System Overview

The collision-avoiding support system basing on ACO shares almost all the designing feature with the one described in Chapter 3 (Fig.3.1). With the target information acquired from navigation aids, as well as environment constraints, the system is to check the safety of the OS passage. If current route of the OS is unsafe, a collision-avoiding route will be generated and OS is to continue on this new safe route.

The only difference with what has been presented in Chapter 3 is that an ACO Algorithm will be applied for route-generating purpose. Therefore, the application factors such as target motion extracting, risk-judging criteria will not be re-analyzed in this chapter to avoid repetition. These factors can be referred back directly in the appropriate sections of Chapter 3.



ACO-OR: Optimal Route by Ant Colony Optimization Algorithm
RG: Route Generator

Fig. 4.3 Collision-avoiding Support System Overview

The overall route-generating procedure is illustrated by the flow chart in Fig. 4.4 and summarized as followings:

- Environmental constraints for the navigation water are extracted from the environmental constraint database or manually input by the ship officer.
- A grid system is built for the navigable water between the current position of the OS, i.e. starting-node, and a point, namely the end-node on the intended route of OS. The end-node is to be located at a certain distance from the starting-node. Grid designing parameters, including distance between lines, number of points on a line, distance between points are chosen as suggested in Chapter 3.
- Connection-desirability is calculated for all connections of the grid. The connection is a path connecting a point on a line with another point on the following line. This will be discussed later in the study.
- The grid and target information is applied in ACO algorithm for searching a collision-avoiding route.
- The best solution, i.e. the optimal collision-avoiding route, is extracted from the algorithm.

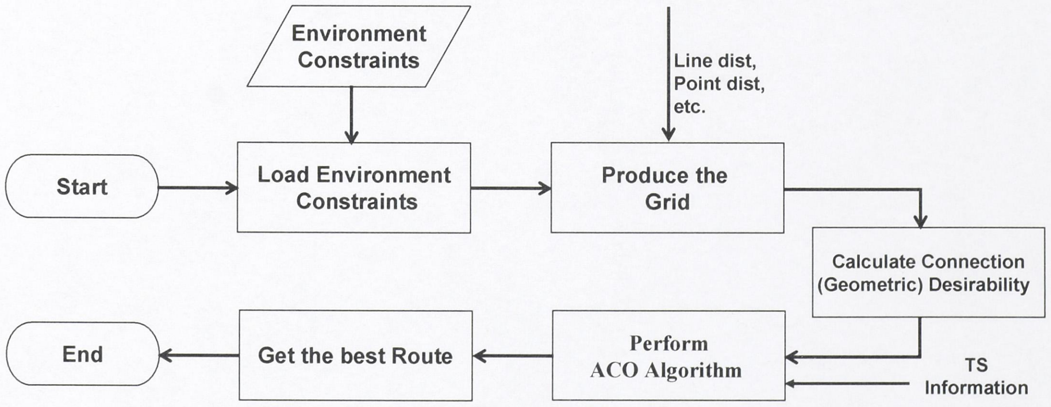

Environment Constraints
Line dist, Point dist, etc.
Start
Load Environment Constraints
Produce the Grid
Calculate Connection (Geometric) Desirability
End
Get the best Route
Perform ACO Algorithm
TS Information

Fig. 4.4 Route-generating Procedure

4.3.2 Route-Cost Function and Traffic Laws Keeping

For the marine-traffic routing application, the aim is to find an optimal collision-avoiding route for the ship. The route must as far as possible satisfy the following requirements:

- Route must be safe. This means that OS must be absolutely free from risk of collision with TS and must not infringe the environmental constraints.
- On the collision-avoiding route, OS must satisfy the marine-traffic rules as far as possible.
- The route travelling time should be minimized.

A solution to the problem is a route, from the starting-point, going through a point on each line, to reach the destination. It is therefore the combination of several connections, or paths connecting points, from the starting-point to the destination.

In terms of preventing collision at sea, the rules for ship to ship collision preventing maneuver are mainly defined in regulations 13 to 17, and 19 of Colreg 72 (The International Regulations for Preventing Collision at Sea). Details of these regulations are easily accessed in the appropriate text books and therefore will not be mentioned any further within the scope of this study.

However, to be readily applicable, those requirements are further generalized and simplified to a compact set of requirements as the followings:

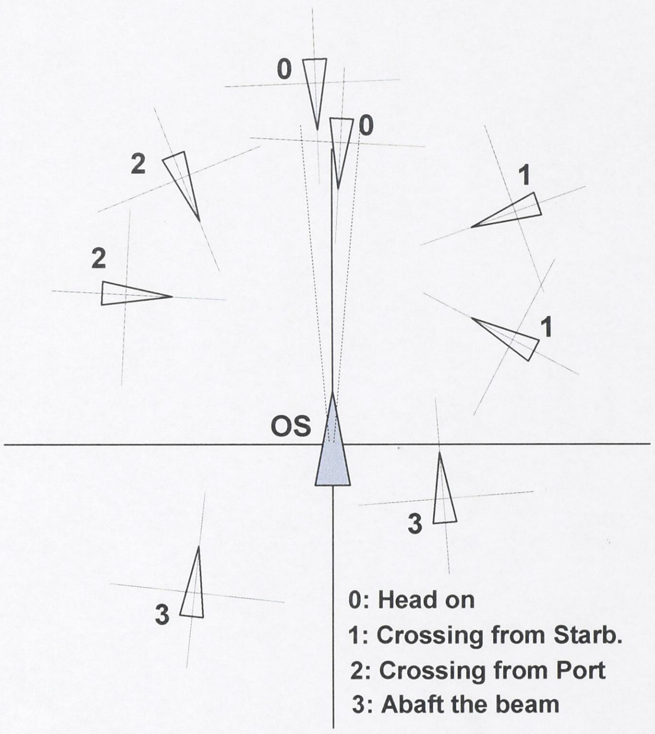


Fig. 4.5 Own Ship and Target Ship Relation

- Collision-avoiding action performed by OS should not cause any confusion on the TS side.
- OS must alter its course to starboard in a head on situation.
- OS should avoid turning to port in crossing situation, especially when it is a give-way vessel.
- OS should avoid the change of course toward a vessel abeam or abaft the beam if the later is an overtaking vessel.

All the above factors can be taken into consideration by the use of a suitable function for optimization problem which is later solved by ACO algorithm. The function hereafter will be referred to as Route-Cost (or Solution-Cost equivalently) and is defined as following (4.1)

Route-Cost evaluation:

$$Q = \left(\sum_i T_i \right) \times \left(1 + \sum_j K_j \right) \tag{4.1}$$

i = 1 to Number of Connections on the Route

j = 1 to Number of Target Ship involved in the situation

T_i *: Time required to travel connection i^th on the Route*

K_j *: Expressing additional cost due to rule infringement when avoiding TS j^th*

The choice of coefficient K in (4.1) reflects the level of pressure on the ship officer if a rule-violating strategy for collision-avoiding is accepted. Values of K for different encountering situations between the OS and a certain TS have been tested for a large number of scenarios. Then, the following set of K values has been chosen in this study (see Fig. 4.5):

(1) Passing a Head-On TS on Starboard Side: K = 0.2 (TS 0).

(2) Passing a TS Crossing from starboard side on OS Starboard side (OS give-way): K = 0.05 (TS 1).

(3) Turning to Port while the TS is crossing from Port Side (OS Stand-on): K = 0.1 (TS 2).

(4) Turning to Starboard while the TS is behind the OS traverse axis on starboard side and overtaking: K = 0.2.

(5) Turning to Port while the TS is behind the OS traverse axis on port side and overtaking: K = 0.2.

(6) Otherwise, K = 0.0.

It is easily seen here that K should be larger for the cases in which the TS may easily misunderstand the OS action, and its counter-action, if strictly following rules, may result in much dangerous situations. Examples of this are case (1) and (3) above, if TS is to alters its course to starboard. Conversely, in case (2), if the officer of the TS feels uneasy and alters course to starboard, the situation will not be worsen, at least.

For a TS on or abaft OS beam (case (4), (5)), from the ship officer view-point, it is very dangerous because of the following reasons:

- It is difficult to observe TS
- TS motion is doubtful
- The situation is persistent

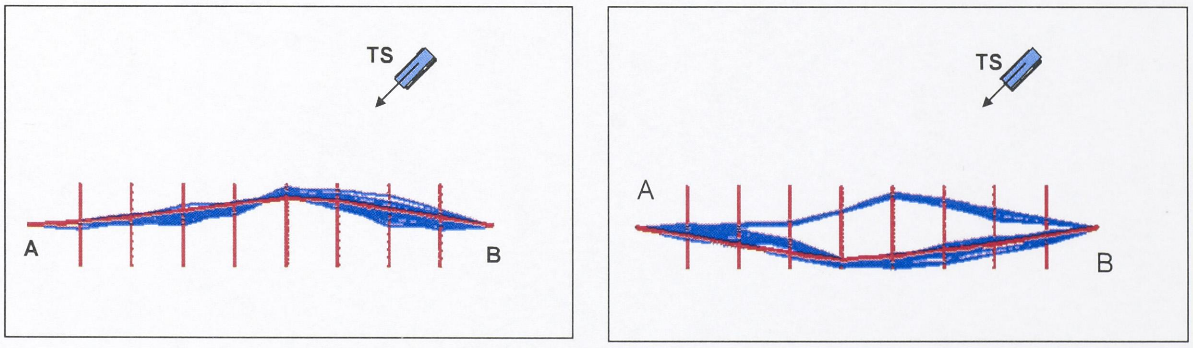


Fig. 4.6 Rule-Violating and Rule-Keeping Routes

An example illustrating the influence of K is shown in Fig. 4.6. Without taking the rules into consideration, when navigating from A to B, the OS alters its course to port, which is actually the shortest collision-avoiding route (left figure). However, an experienced officer is likely to choose to turn to starboard to ensure a safe passage (right figure). In fact, the OS is a stand-on vessel in this encounter and it is the duty of the TS to take action. But, if OS is to take action, its course change to port is more dangerous (there is a high possibility that TS also alters its course to port) than the course change to starboard. The cost of the collision-avoiding route in the right figure is therefore should be less than that of the one in the left figure.

4.3.3 Connection Desirability

Even without the environment constraints and risk of collision with TS, setting aside the trail-pheromone level, connections still possess different level of attractiveness. This component of the attractiveness is called the Connection-Desirability. The desirability parameter is initialized simultaneously when the grid is built. It plays the role of heuristic information that control the tendency and efficiency of the search.

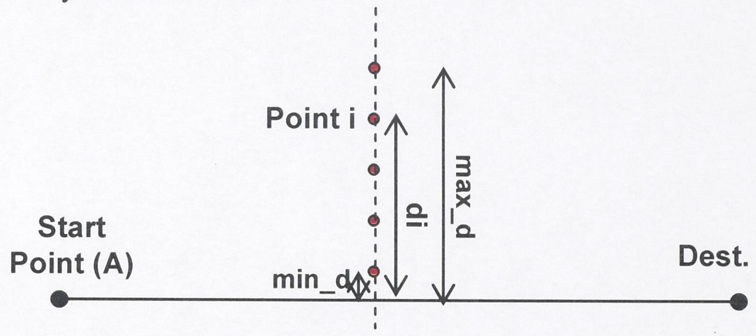


Fig. 4.7 Connect Desirability Initialization

In routing application, starting from a random node (point A), to reach a given destination (Dest.), the ship officer tends to choose nodes on the following line in such a way that the total route forms a straight line from A to Dest. or a path as close to this straight line as possible.

Noting this fact of the seamanship, the author proposes a formula for determining desirability of every connection on the grid as followings (4.2)

$$\eta_{A,i} = \exp\left(-\frac{d_i}{d_{\max} \times Scale\_factor}\right) \qquad (4.2)$$

where

A : *the current point*

i : *the point number i on the following line*

d_i : *deviation of point i from the straight line connecting A with the destination*

$d_{\max}$: *Maximum deviation from the straight line*

It can be seen from (4.2) that value of the desirability η for a connection (A-i) closer to the straight line to destination (smaller d_i) is larger than that of a connection further from this straight line (larger d_i value). This gets along well with the general expectation that a straight route to the destination is always the route of choice that is mentioned above if it ensures the safety of navigation.

To further reduce the total amount of calculation, connection-desirability is also used to express the static constraints of the area, i.e. the environmental constraints. If the connection leads the OS to enter a prohibited area, for example, its desirability is set to 0.

If it is impossible to reach point i from point A, then

$$\eta_{A,i} = 0 \qquad (4.3)$$

The connection-desirability is illustrated in Fig. 4.8 for different Scale_factor in (4.2), with the distance between points to be 50[m] and d_{max} to be 2000[m].

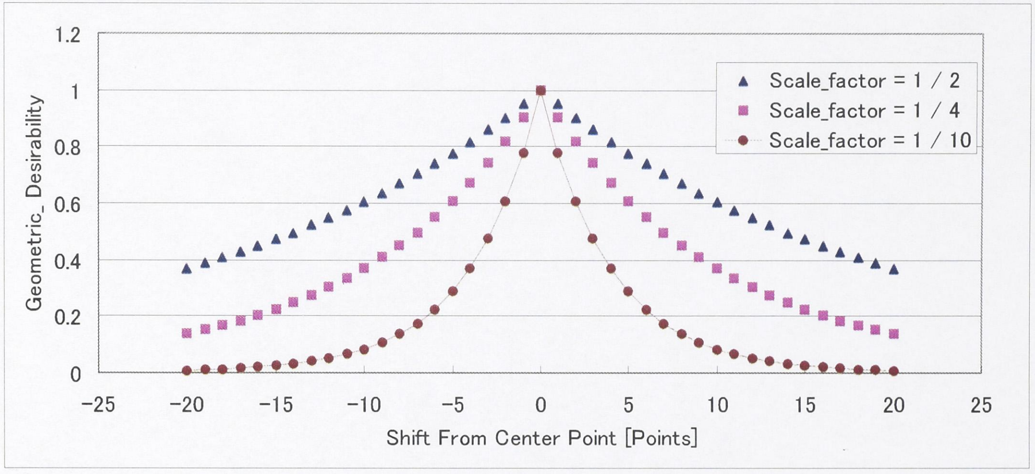


Fig. 4.8 Connection Desirability

The graphs reveal also that the connection-desirability falls too low for points far away from the center point. This means that the ants will not likely be attracted to these points and the route-searching algorithm can not explore regions far from the straight line properly. Then, connection-desirability should be modified as the following:

If ($\eta_{A,i} < \eta^*$), then:

$$\eta_{A,i} = \eta^*; \quad \textit{where}\ 1 > \eta^* > 0\ \textit{is a limit desirability level} \qquad (4.4)$$

Then, the desirability of connections varies as shown in the following figure (Fig. 4.9).

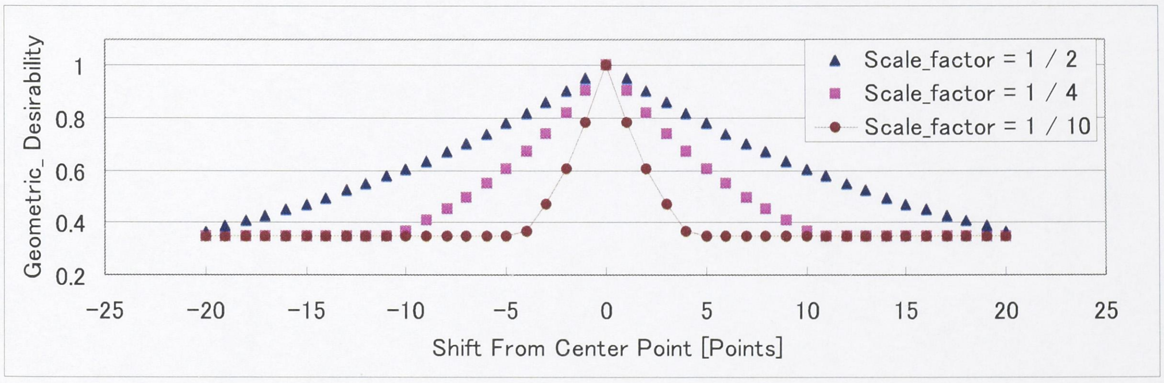


Fig. 4.9 Connection-Desirability Modification by (4.4)

From the figure above and through simulation studies, it has been seen that the choice of Scale_factor to be 0.5 satisfies both the convergence property and the exploration capacity of the route-searching algorithm.

The connection-desirability can also be modified by:

$$\eta_{A,i} = \eta_{A,i} + \eta^0 \textit{ where } \eta^0 \textit{ is a constant} \qquad (4.5)$$

Using (4.5), the desirability lies in the range from η^0 (for the furthest node) to $1+\eta^0$ for the center node and there is always a difference of desirability between neighboring nodes. The resulting Connection-Desirability is described in Fig. 4.10. If this form of modification is used, a Scale-Factor value in the range from 0.15 to 0.2 appears to be suitable for the route-searching algorithm.

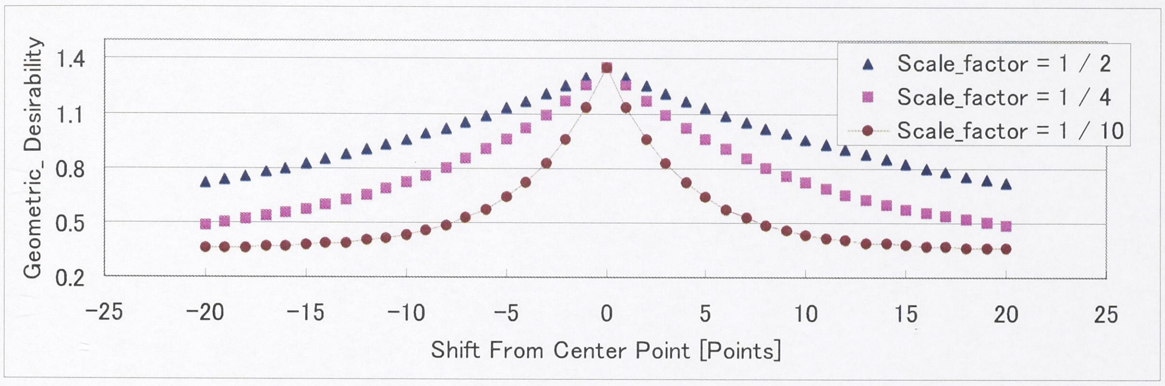


Fig. 4.10 Connection-Desirability Modification by (4.5)

The choice of connection-desirability explicitly affects the convergence of the route-searching algorithm. It should be chosen so as to keep a balance between the exploitation of the region around the straight line to the destination and exploration of other region of the search-space, where more reasonable solutions may be detected.

4.3.4 Probabilistic Node Selection

Assuming that an ant is currently at a point (c) on line (i-1)th of the grid, it is to choose a node i.e. a point on line ith probabilistically. This point is the ant's next-point.

The choice of the next-point depends on the connection attractiveness, where a connection is a path from the current point (c) to a point on line ith. The connection attractiveness is the combined effect of trail-pheromone level of the connection and its desirability.

The probability for the ant to select node k can therefore be determined by the following equation (4.6)

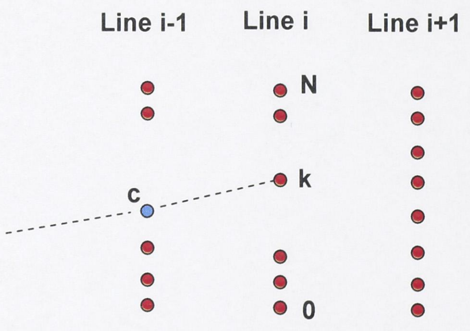


Fig. 4.11 Node Selection

$$p_{c,k} = \frac{\left(\tau_{c,k}^{\alpha}\right)\left(\eta_{c,k}^{\beta}\right)}{\sum_{h=0}^{N}\left(\tau_{c,h}^{\alpha}\right)\left(\eta_{c,h}^{\beta}\right)} \qquad (4.6)$$

where

$p_{c,k}$: *probabilit y of selecting node k*

$\tau_{c,k}$: *pheromone level on connection* c,k (*see* 4.3.6)

α : *a parameter to control the influence of* $\tau_{c,k}$

$\eta_{c,k}$: *desirabili ty of connection* j,k

β : *a parameter to control the influence of* $\eta_{c,k}$

The connection-desirability is defined in section 4.3.3 while pheromone level is the amount of trail-pheromone ants laid on the connection if they had successfully reached the destination through this connection. For simplicity, α, β are simply set to unity.

$$\alpha = \beta = 1 \quad then \qquad p_{c,k} = \frac{\left(\tau_{c,k}\right)\left(\eta_{c,k}\right)}{\sum_{h=0}^{N}\left(\tau_{c,h}\right)\left(\eta_{c,h}\right)} \qquad (4.7)$$

The node-selecting mechanism is illustrated in Fig. 4.12 where the subscript c is omitted for simplicity. A random value X is generated in the range (0, 1). Then, the next point is chosen to be point m (in line ith) accordingly.

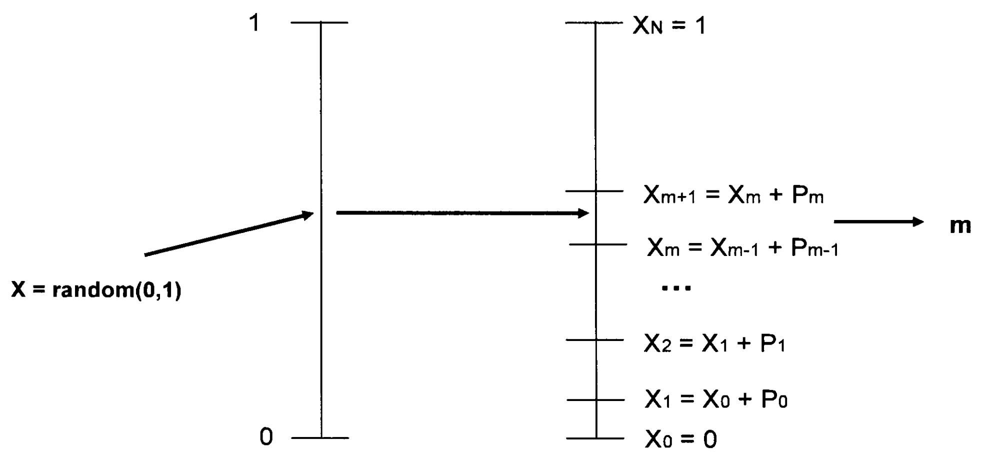


Fig. 4.12 Probabilistic Node Selecting Procedure

4.3.5 Solution (Route) Producing Procedure

The procedure modeling the food-searching process of an ant in the colony is presented in the flow chart in Fig. 4.13. To improve the success rate of the process, in this study, the following 2 looping parameters are employed:

- Rmax: Maximum number of tries on a line i.e. number of tries from a single point
- Kmax: Maximum number of tries for an ant in its life

Line number is denoted by l and initially set to 0. Variables r and k are used to count the tries. k is initially set to 0 and r is reset to 0 whenever the ant starts selecting a following point, from its current position.

The point is selected probabilistically as described in section 4.3.4. After selecting a point, ant tries to approach that point, using the ship dynamics.

- If the path is safe, the ant reaches the point and therefore the route-searching process continues from this new point, l is increased accordingly.
- If the path is unsafe, the ant tries another point. r is therefore increased.

If the ant fails to reach the following line from a certain point after Rmax tries, it is wise to shift the ant back several steps and continues the route-searching process from there.

If the ant can find a route till the destination, the procedure is successful. A solution (i.e. a route) is produced. The quality of the route is evaluated by (4.1). Conversely, if the ant can not reach the destination after Kmax tries, the route quality is then set to Infinite to express this failure.

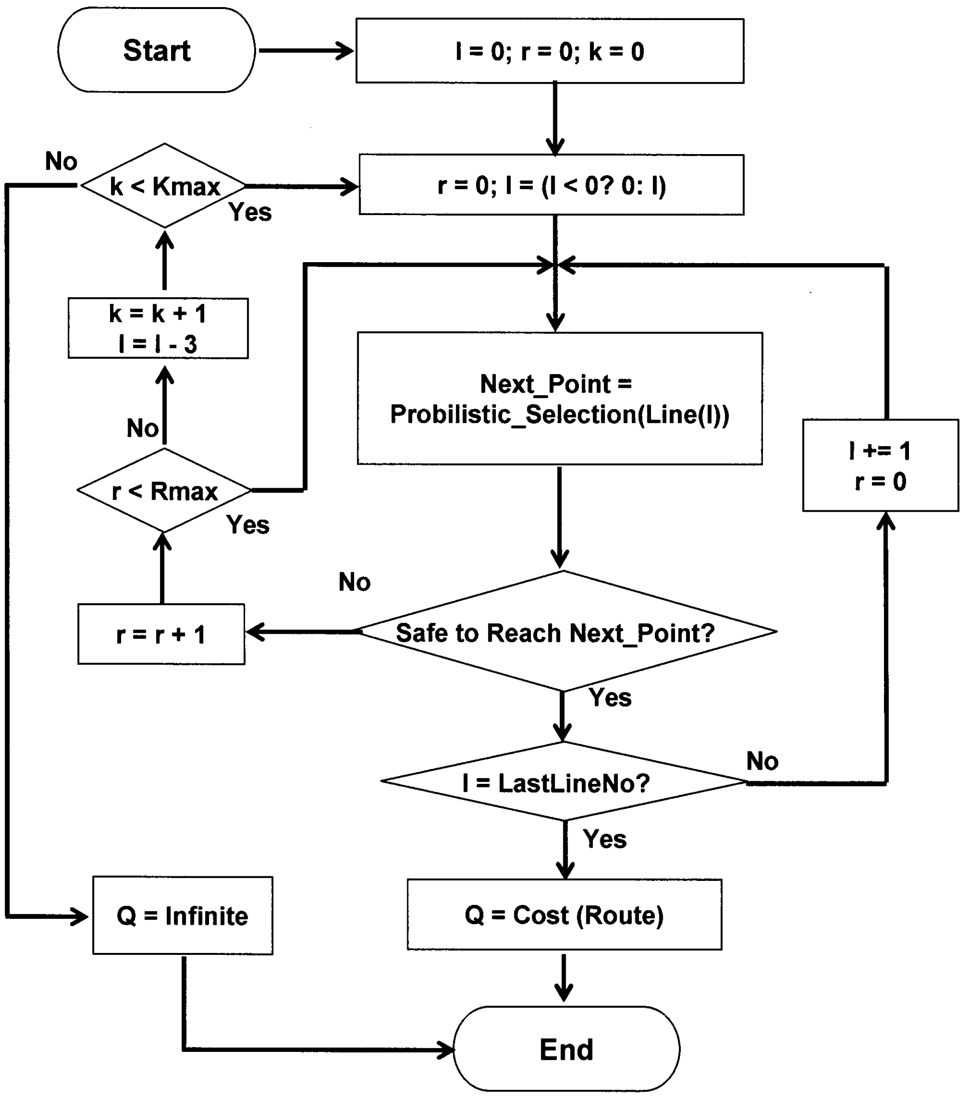


Fig. 4.13 Route Producing Procedure

4.3.6 Pheromone Manipulations

Manipulating the trail-pheromone level of the connections is perhaps one of the most important factors (together with the connection-desirability) which decide the searching performance of the algorithm. Different pheromone-varying techniques have been proposed, including the evaporation, pheromone-laying action of ants and deamon actions.

The performance deeply depends on the choice of the connection-desirability as well as other pheromone manipulating parameters including the pheromone-delivering amount, the pheromone-evaporating coefficient and etc. A good combination of these parameters is the one that properly reasons between the exploration and exploitation capacity of the ant colony:

- A rapid discrimination of pheromone level (e.g. larger coefficient of evaporation) enables the colony to spend most of its efforts on exploiting a promising region. The colony therefore comes quickly to a convergence but there also arise the risk that some region of the search-space might be ignored.
- Conversely, a slow discrimination allows the algorithm to explore thoroughly over the search-space at the price that the algorithm needs more runs to converge.

4.3.6.1 Pheromone Evaporation

Evaporating process simulates the actual nature phenomenon in which the trail- pheromone level of ants' track decreases gradually because pheromone is volatile. This allows the poor route which is not visited frequently to be eventually forgotten.

The process can be realized by the following equation (4.8), for every connection of the grid.

$$\tau_{A,B}^{new} = (1-\rho)\times\tau_{A,B}^{old} \qquad (4.8)$$

where

$\tau_{A,B}$: *pheromone level on the connection between point A and point B*

ρ:*rate of pheromone evaporation*, $\quad 0 < \rho < 1$

ρ is a designing parameter and is usually set in the range (0.05-0.1). An example of the effect of evaporation is shown in Fig. 4.14 for two different values of the pheromone-evaporation coefficient ρ .

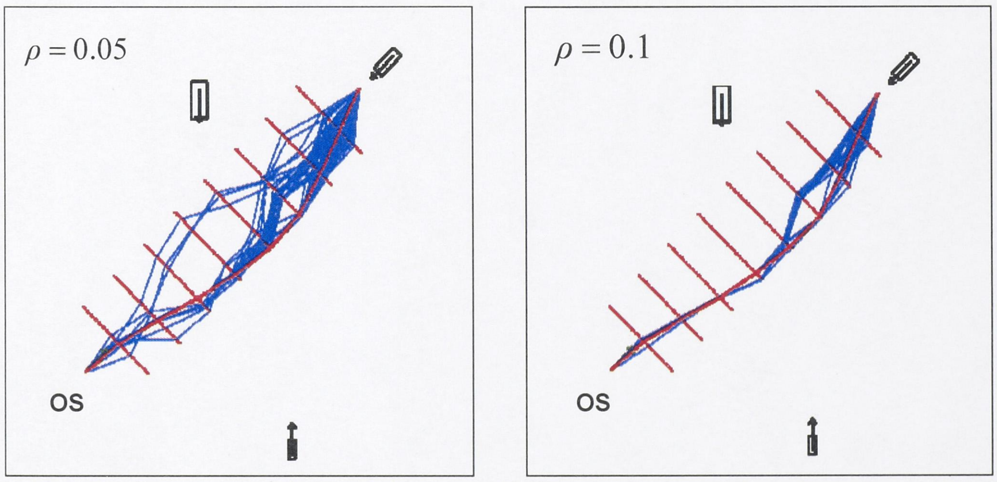


Fig. 4.14 Performance of Ant Colony for Different Evaporating-Coefficient Values

4.3.6.2 Pheromone Delivering

While traveling connections on the grid, ants continuously lay pheromone on their trails. This increases the pheromone level on these connections.

In this study, a pheromone-delivering mechanism is proposed that takes into consideration the cost of a route in comparison with that of other routes and of the best route which has been ever found. The route found by an ant consists of several connections and the pheromone amount the ant leaves on a connection (i) is calculated by (4.9)

$$\Delta\tau_i = \Delta\tau^* \times \{C - (Q - Q_{best})/(Q_{worst} - Q_{best})\} \qquad (4.9)$$

where

$\Delta\tau^*$: *A pre – fixed pheromone amount*

C : *A designing parameter controllin g the amount of pheromone an ant delivers* , $C \geq 1$

Q : *Cost of the route*

Q_{worst} : *Cost of the worst route in a Run*

Q_{best} : *Cost of the best route so far*

Formula (4.9) helps to limit the pheromone level laid by an ant on the connection to the range $(0, \Delta\tau^* \times C)$. The better the route is, the larger the amount of pheromone laid on its connections is.

To reinforce the "exploitation" of the searching procedure for the optimal route, a large increase of pheromone-level is applied for the connections on the best route that has been ever found.

$$\tau^{new}_{A,B} = \tau^{old}_{A,B} + \Delta\tau^{**} \qquad (4.10)$$

where AB is a connection on the best route

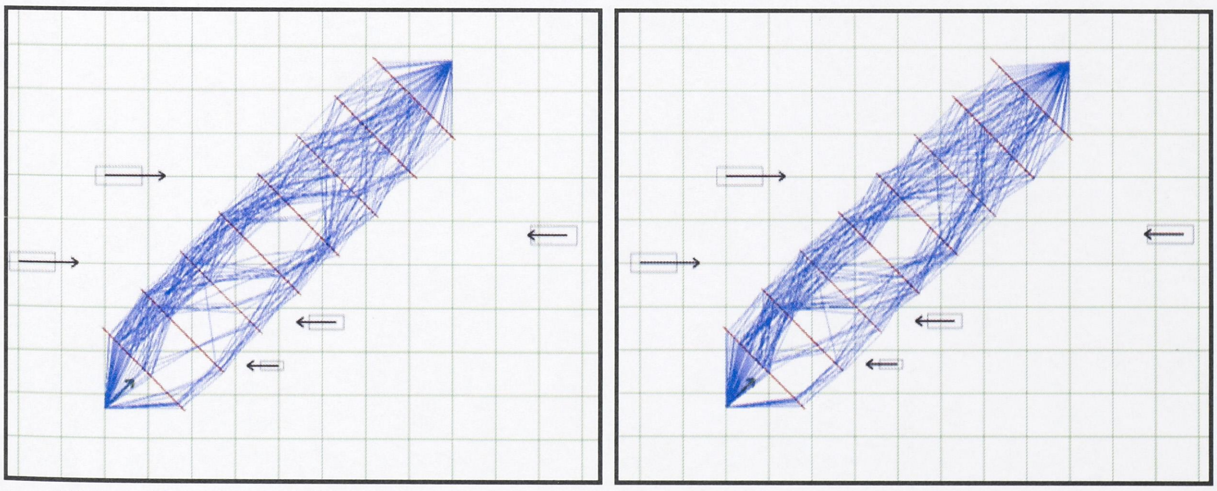


Fig. 4.15a Performance of Ant Colony with Small Trail-Pheromone Delivery Amount
(Right = 50 Runs, Left = 100 Runs)

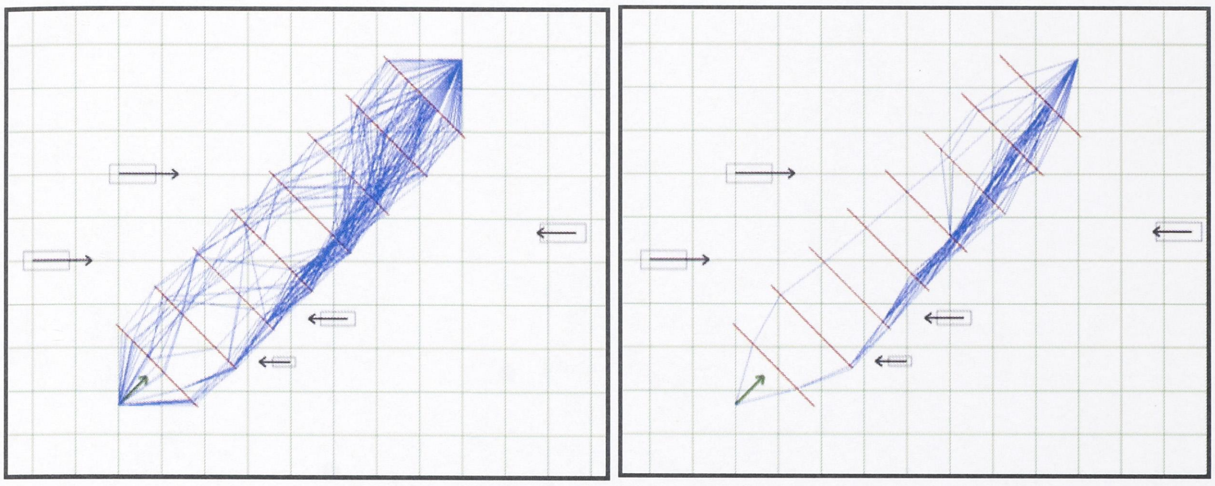


Fig. 4.15b Performance of Ant Colony with Medium Trail-Pheromone Delivery Amount (Right = 50 Runs, Left = 100 Runs)

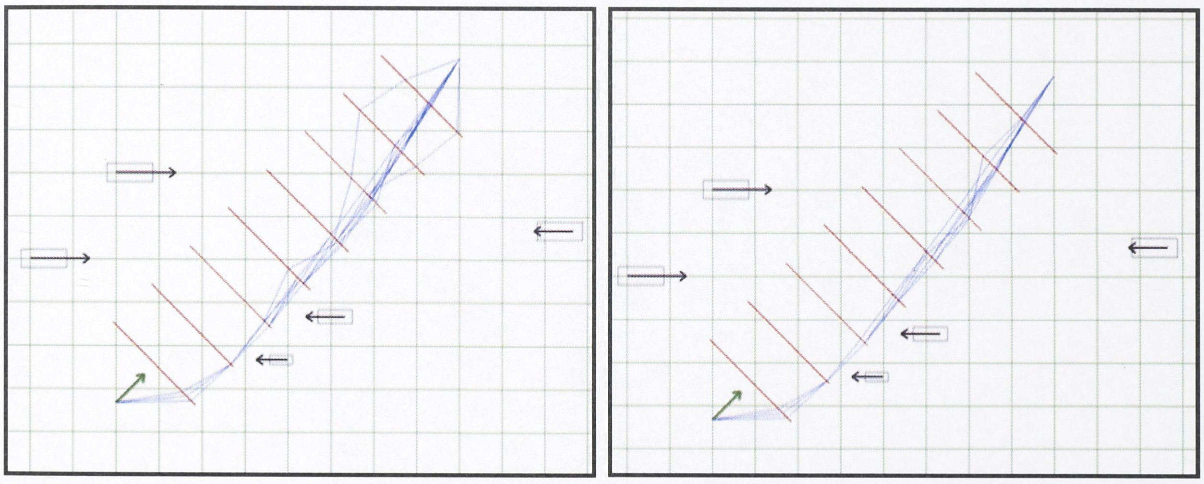


Fig. 4.15c Performance of Ant Colony with Medium Trail-Pheromone Delivery Amount (Right = 50 Runs, Left = 100 Runs)

To illustrate the effects of the amount of trail-pheromone delivered on the performance of the ACO algorithm, the route-searching simulation is conducted for a scenario where the OS has to take action to avoid collision with several TS. Random routes chosen by 70 ants after 50 runs and 100 runs are shown in the figure on the right and the left respectively. In Fig. 4.15a, the designing parameters ($\Delta\tau^{*}, \Delta\tau^{**}$) are (0.001, 0.01). The colony searches throughout the search-space but the search is has not converged to the optimal region. In Fig. 4.15b, these parameters are (0.03, 0.3). The ACO algorithm initially explored throughout the search-space (till 50 runs) and then gradually converged to the optimal (after 100 runs). If large amount of pheromone is delivered by ants ($\Delta\tau^{*} = 0.1, \Delta\tau^{**} = 1.0$ as in Fig. 4.15c), the ant colony concentrates quickly to the region on starboard side of the OS. The algorithm may therefore have ignored the possible solutions on OS port side. Furthermore, as the ants are strongly attracted to the nodes near the starboard-end of the first grid-line, the searching-algorithm actually missed a better solution in which the ants approach the second grid-line by a straight path (See Fig. 4.15b).

4.3.6.3 Deamon Actions

To increase the efficiency of the search, additional deamon-actions are applied in the algorithm as followings, with Max_Value, Min_Value and Non_Optimal_Max to be designing parameters:

- If pheromone-level of a connection is larger than Max_Value, it is set to Max_Value.
- If pheromone-level falls below Min_Value due to evaporation, it is set to Min_Value.
- If pheromone-level of a connection which is not on the best route exceeds Non_ Optimal_Max, it is set to Non_Optimal_Max.

4.3.7 Convergence Enhancement by Solution Post-Processing

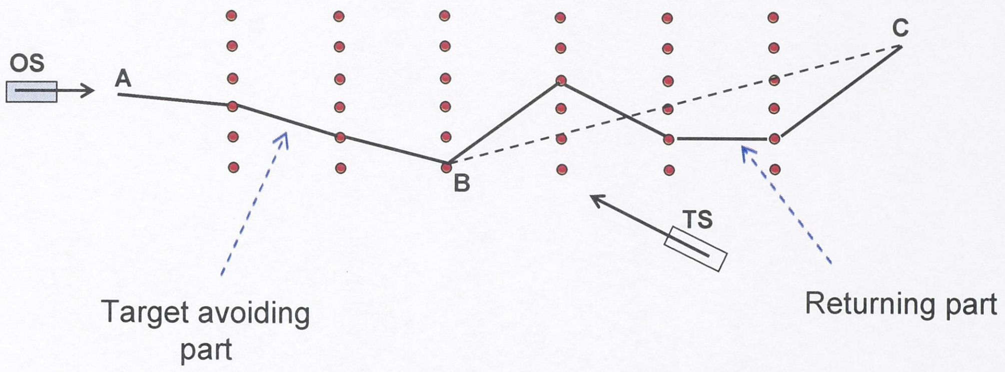


Fig. 4.16 Collision-avoiding Route Components

The collision-avoiding route for the OS commonly consists of 2 parts, a part to avoid collision (part AB in Fig. 4.16) and another for returning to the original course. The later should be a straight line, or more generally the shortest possible route to the destination (C). However, due to the relatively random nature of the search, particularly in “exploring” new solutions, the part for returning to the original course may be in a zigzag form (part BC in Fig. 13). This reduces the quality of the generated route and in turns limits the pheromone updating amount on that route. As a result, the “exploiting” is refrained in this new region of the solution-space. The delivery of pheromone on this unwanted zigzag route also produces chaos in the search process as other ants would be attracted to these connections.

To overcome this deterrence, the route generated by section 4.3.5 should be further processed before it is saved and used to perform pheromone-delivering. In other words, this part should to be straightened before evaluating the route-cost, like the dash line in Fig. 4.16. Then, quality of the route will be improved so as to improve the performance of the algorithm. This can be considered as an additional local-search process in the ACO algorithm.

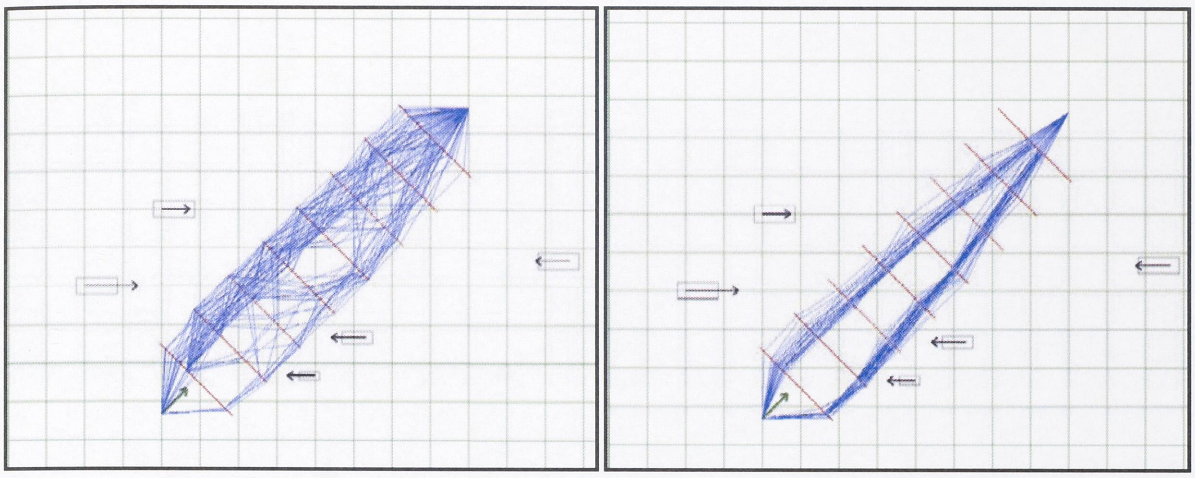


Fig. 4.17 Effect of Local-Search Procedure on Searching Performance

The outcome of the additional local-search process is illustrated in Fig. 4.17 for a colony of ant performing the search after 50 runs. Due to the randomness of the search, the left figure (without local-search) shows the chaos of the colony, especially in several last grid-lines. The right figure (local-search process included) expresses clearly the converging-tendency of the coolly as the chaotic effects have been largely reduced. In the latter algorithm, the route of every ant has been properly straightened so that the trail-pheromone is not delivered on the unwanted connections of the grid.

4.3.8 Overall ACO-Based Route-Generating Algorithm

The overall algorithm can be presented by the pseudo code as followings

```
A. Initialization
   Initialize_Grid(N_line, N_point, D_point);
   Initialize_ConnectionDesirability();
   Qbset = infinite;

B. Evolution
   For J = 1 to Nr
      For I = 1 to Na
            Produce_Solution_For_Ant(Ants(I));
            Post_Processing_the_Solution(Anst(I));

            If ( Route_Cost(Ants(I)) < Qbest ) then
                  Ant_best = Ants(I);
                  Qbest = Route_Cost(Ants(I));
            End If
      Next I

      Pheromone_Evaporation();
      Qmedium = Get_Medium_RouteCost(Ants());

      For I = 1 to Np
            If ( Qmedium > Route_Cost(Ants(I)) ) then
```

```
                Update_Pheromone(Ants(I)) by (4.9);
            End If
        Next I

        Reinforce_Pheromone(Abest) by (4.10);
        Perform_Deamon_Actions();
    Next J

C. Termination
    Return Ant_best;
```

Where the designing parameters and variables are defined as the followings:

- *N_line: Number of lines on the grid*
- *N_point: Number of points on a grid line*
- *D_point: Distance between points on a line*
- *Na: number of ants in population*
- *Nr: number of run*
- *Qbest: Best route-cost*
- *Qmedium: Medium route-cost (Determining the number of ants that produce pheromone)*
- *Ant_best: Best solution so far*
- *Ants(Na): set of ants*

It should be noted here that ONLY a number of best routes in a Run (routes having cost less than Qmedium in each a loop of J) are used for delivering trail-pheromone. It enables the algorithm not to spend resources on regions in which the route quality is poor.

4.4 Simulation Studies

The route-producing algorithm basing on ACO is applied to generate collision-avoiding route for a set of typical marine traffic encounters. The same scenarios are also used in simulation studies with route-producing algorithm by Dynamic-Programming (Chapter 3) and Bacteria Foraging Optimization Algorithm (BFOA – Chapter 5) to provide a cross check of the algorithm validity and to reveal advantages, disadvantages of each algorithm. The same OS model with that described Section 3.3.4 (Chapter 3) is used in this chapter. The details of own ship and target positions as well as their speeds and courses over ground are therefore not listed here to avoid repetition. This information can be found in corresponding section in Chapter 3.

The ACO algorithm is applied with $\alpha = \beta = 1$, $\rho = 0.95$. A colony of 150 ants is used (Na) to generate routes.

In all the scenarios, the randomly selected routes of 100 ants are shown after different number of runs of the colony (Nr = 20, 100, and 200). Then, the best route that the ant colony has found after each run (starting from 100 runs) are depicted, with the best route found after 200 runs drawn in red.

4.4.1 Scenario 1

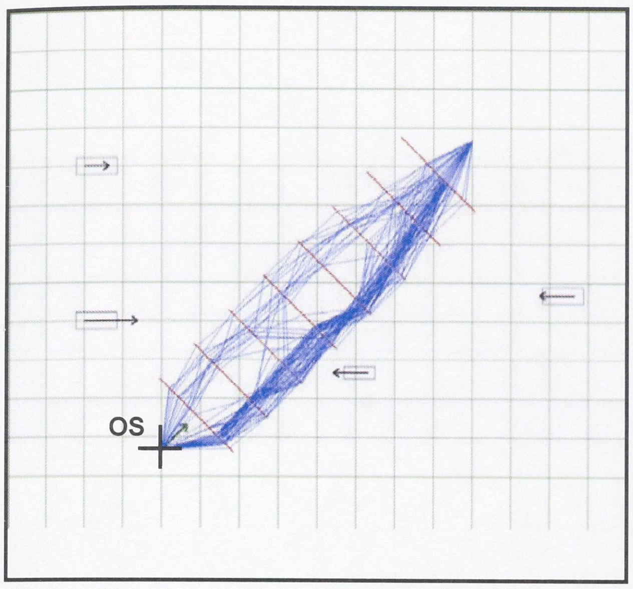


Fig. 4.18a Random Ant Routes after 20 Runs

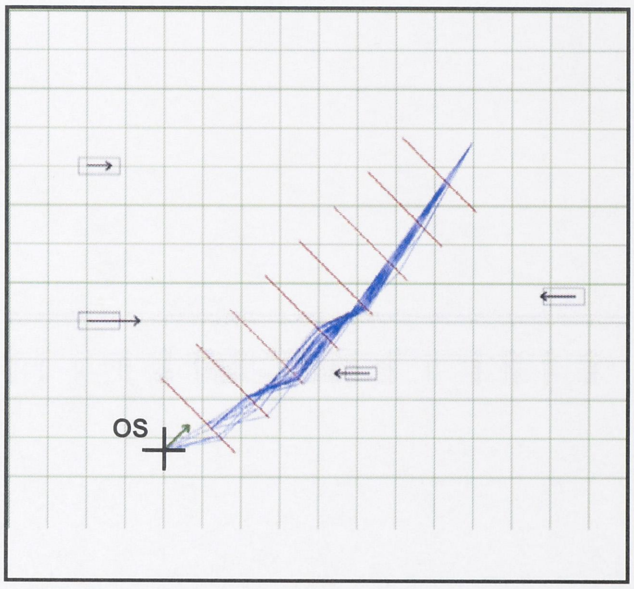


Fig. 4.18b Random Ant Routes after 100 Runs

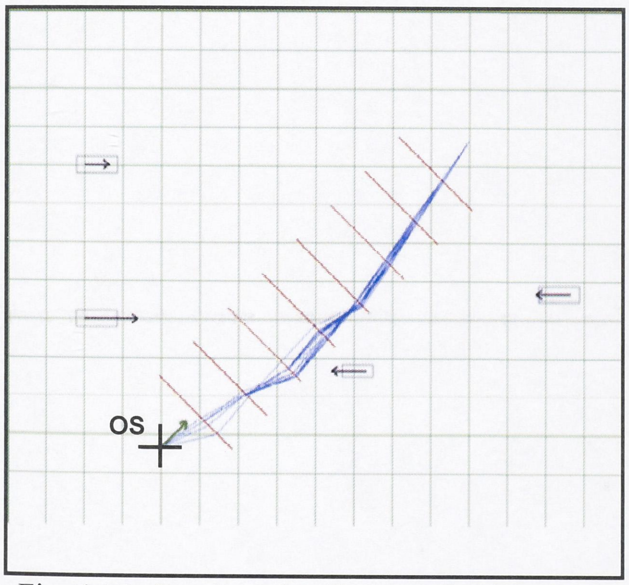


Fig. 4.18c Random Ant Routes after 200 Runs

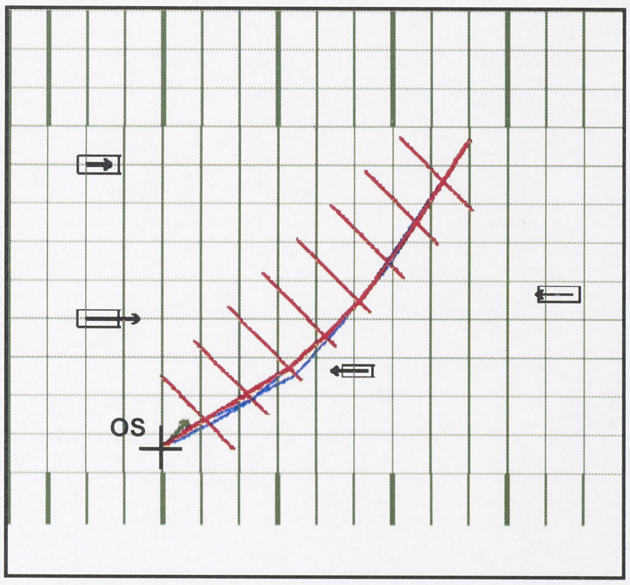


Fig. 4.18d Best Solutions after 100 Runs

Comments on the scenario:

- The search is initially scattered around the grid (Fig. 4.18a) so that the whole solution-space is explored thoroughly.
- Ant colony converges promptly to a region of good collision-avoiding strategy (Fig. 4.18b and Fig. 4.18c).
- Regulations of the road are properly satisfied (turning to starboard, passing starboard crossing targets on OS port side).
- Best solutions found are very close to the optimal solution after 100 runs (Fig. 4.18d). Strategy is more appropriate than that produced in 3.5.1.

(See 3.5.1 and 5.4.1 for result comparisons)

4.4.2 Scenario 2

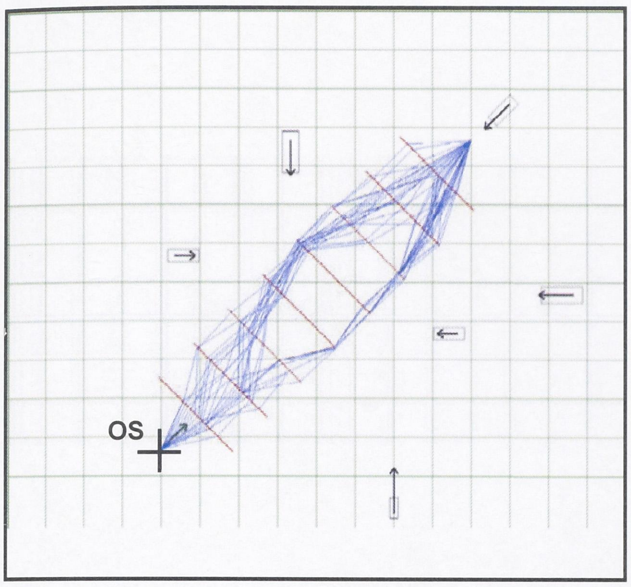


Fig. 4.19a Random Ant Routes after 20 Runs

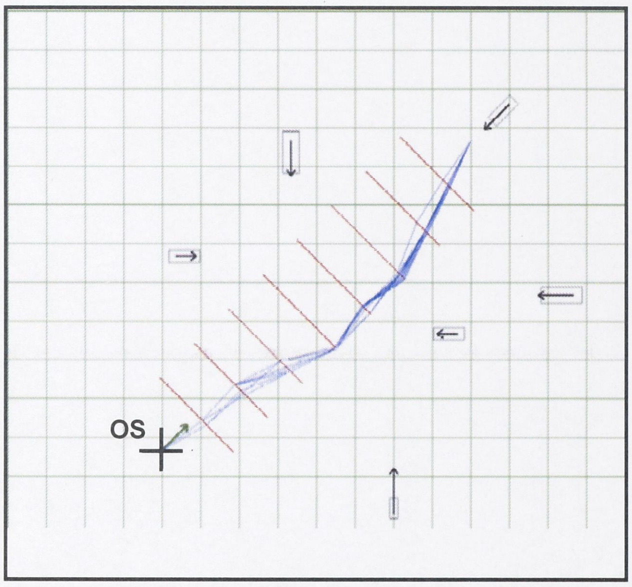


Fig. 4.19b Random Ant Routes after 100 Runs

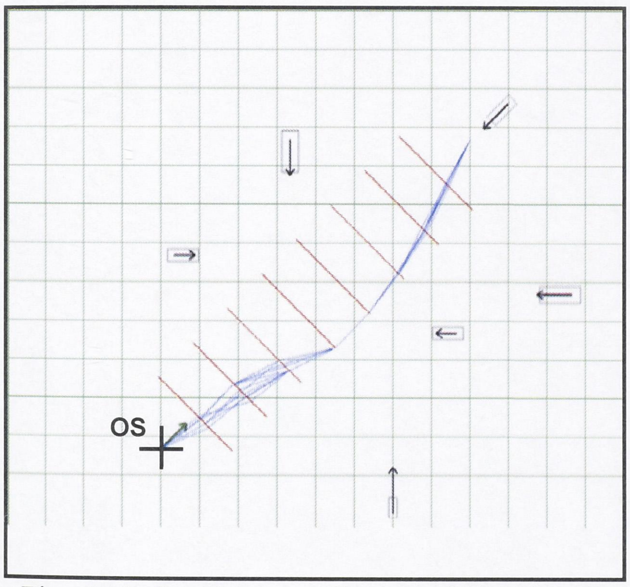


Fig. 4.19c Random Ant Routes after 200 Runs

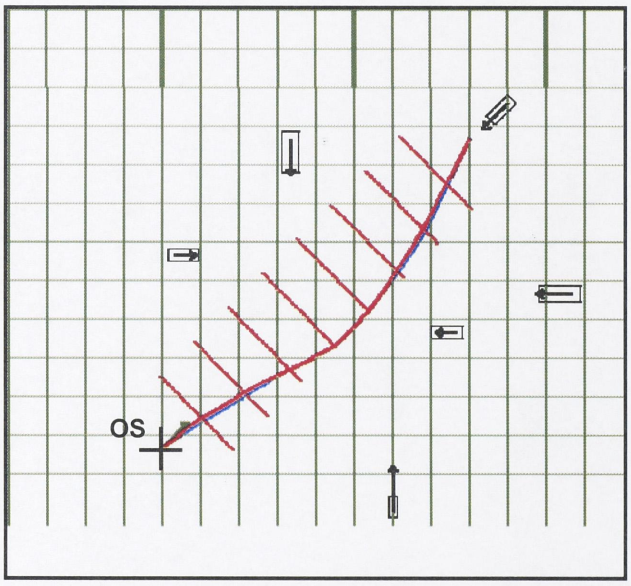


Fig. 4.19d Best Solutions after 100 Runs

Comments on the scenario:

- The search is initially scattered around the grid (Fig. 4.19a) so that the whole solution-space is properly explored.
- Ant colony converges promptly to a region of good collision-avoiding strategy (Fig. 4.19b and Fig. 4.19c).
- Regulations of the road are properly satisfied (turning to starboard, passing starboard crossing targets on OS port, passing head-on target on port).
- Best solutions found are very close to the optimal solution after 100 runs (Fig. 4.19d). Strategy is more appropriate than that produced in 3.5.2.

(See 3.5.2 and 5.4.2 for result comparisons)

4.4.3 Scenario 3

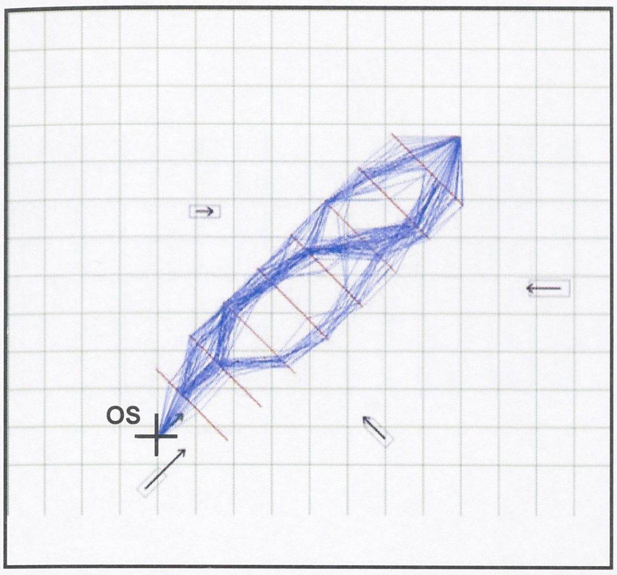


Fig. 4.20a Random Ant Routes after 20 Runs

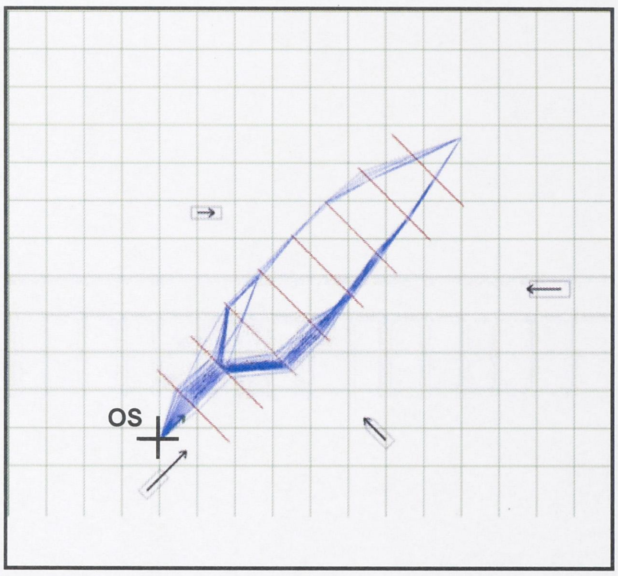


Fig. 4.20b Random Ant Routes after 100 Runs

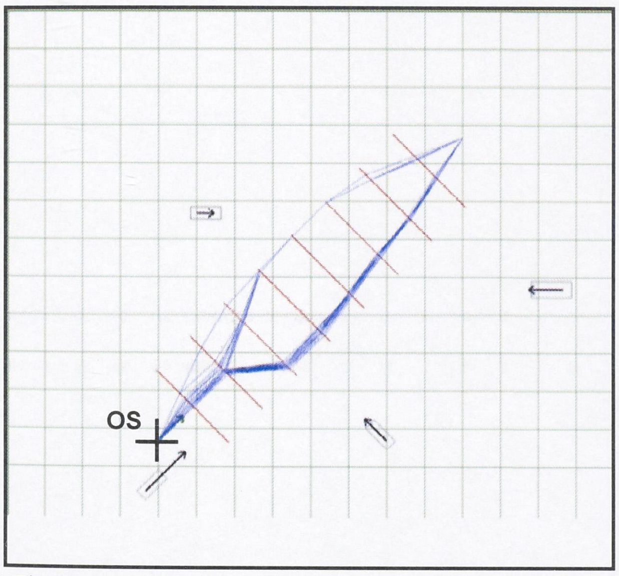


Fig. 4.20c Random Ant Routes after 200 Runs

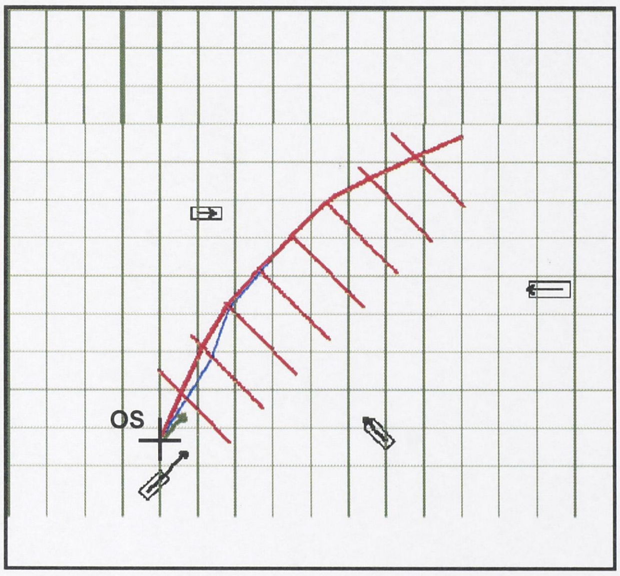


Fig. 4.20d Best Solutions after 100 Runs

Comments on the scenario:

- The search is initially scattered around the grid (Fig. 4.20a), thus the whole solution-space is explored thoroughly.
- Ant colony converges to 2 different regions as the route qualities relating to these 2 regions are more or less similar (Fig. 4.20b and Fig. 4.20c).
- Regulations of the road are satisfied to an acceptable extent. Although OS alters course to starboard and therefore passing starboard crossing targets on starboard, it does not cause any misunderstanding from TS side. Even if the targets actually judge the case wrongly, it is still safe as far as they follow rules of the road.
- Best solutions found are very close to the optimal solution after 100 runs (Fig. 4.20d).

(See 3.5.3 and 5.4.3 for result comparisons)

4.4.4 Scenario 4

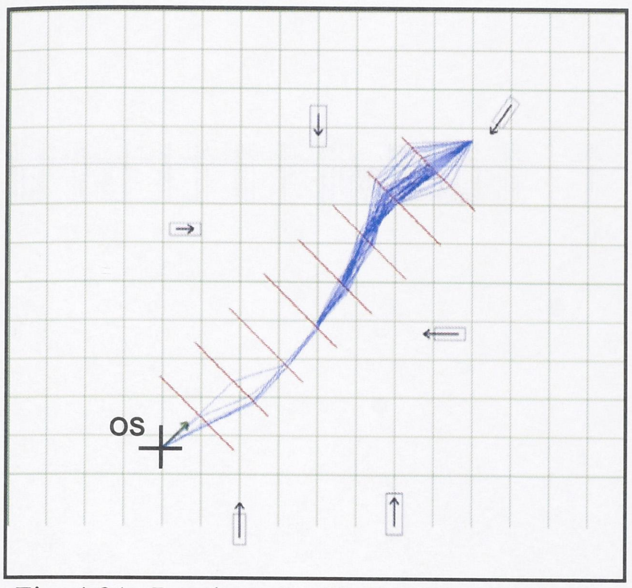


Fig. 4.21a Random Ant Routes after 20 Runs

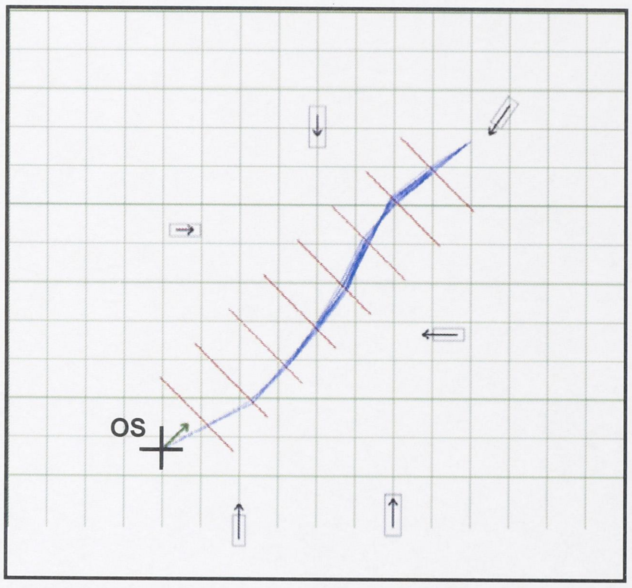


Fig. 4.21b Random Ant Routes after 100 Runs

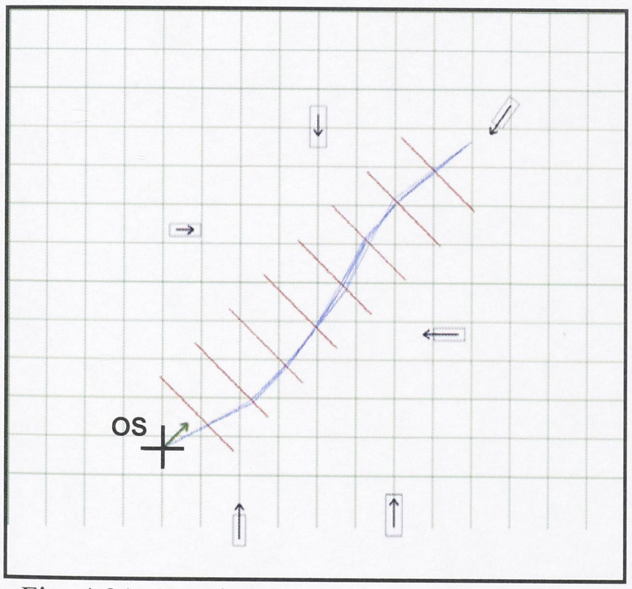


Fig. 4.21c Random Ant Routes after 200 Runs

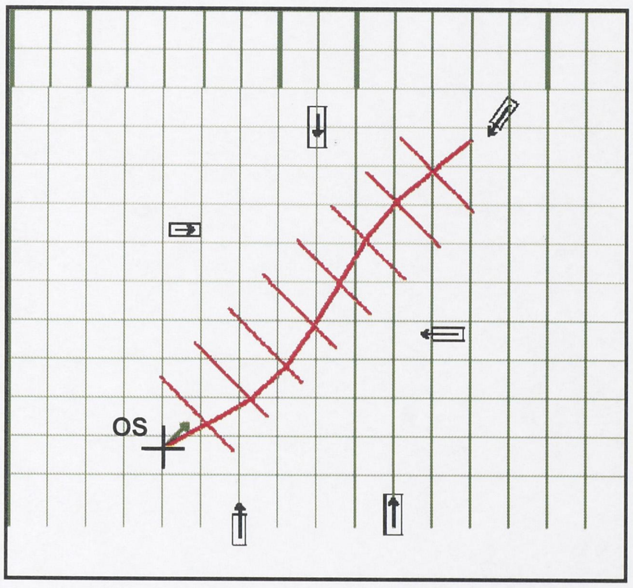


Fig. 4.21d Best Solutions after 100 Runs

Comments on the scenario:

- Ant colony converges promptly to a region of good collision-avoiding strategy because the route quality corresponding to this region is far better than that of other regions.
- Regulations of the road are properly satisfied (turning to starboard, passing starboard crossing targets on OS port as far as the situation allows).
- Best solutions found are very close to the optimal solution after 100 runs (Fig. 4.21d).

(See 3.5.4 and 5.4.4 for result comparisons)

4.4.5 Scenario 5

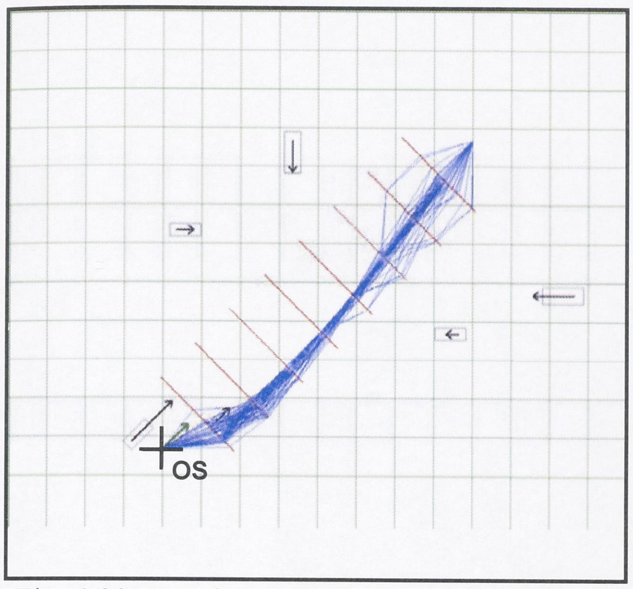


Fig. 4.22a Random Ant Routes after 20 Runs

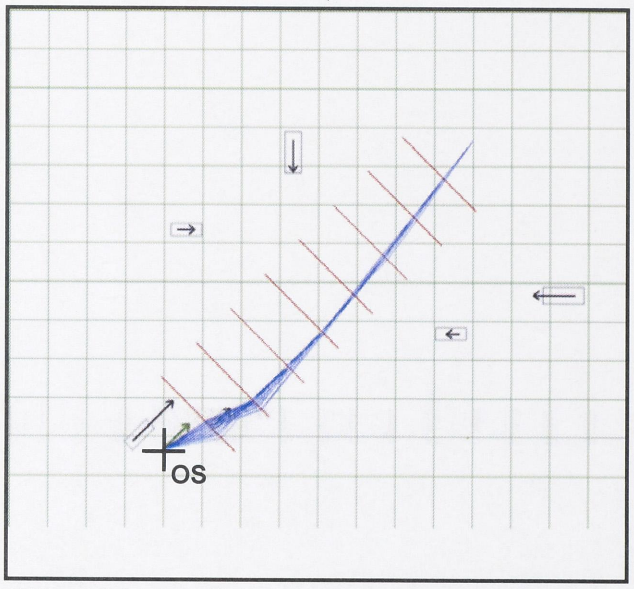


Fig. 4.22b Random Ant Routes after 100 Runs

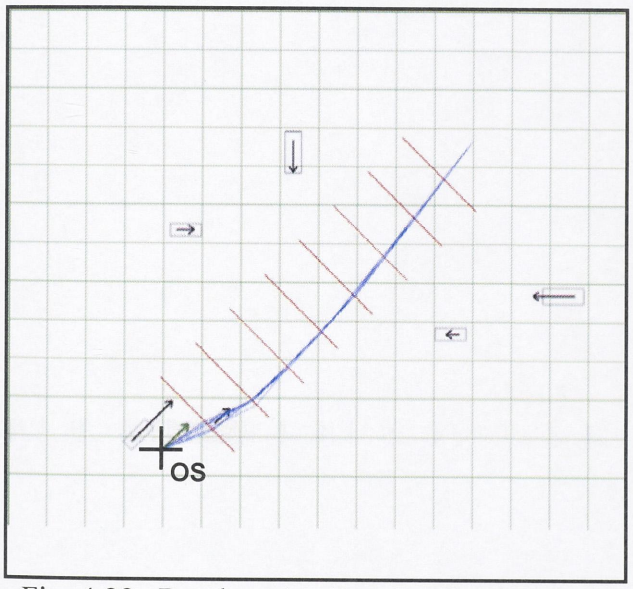


Fig. 4.22c Random Ant Routes after 200 Runs

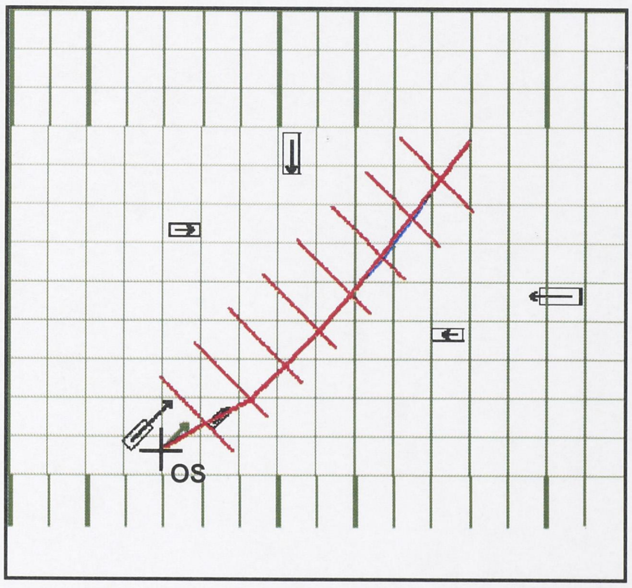


Fig. 4.22d Best Solutions after 100 Runs

Comments on the scenario:

- The search is initially scattered around the grid (Fig. 4.22a), then the whole solution-space is explored thoroughly.
- Ant colony converges promptly to the region of the globally optimum solution (Fig. 4.22b and Fig. 4.22c).
- Regulations of the road are perfectly satisfied.
- Best solutions found almost coincide with the optimal solution after 100 runs (Fig. 4.22d). Strategy is more appropriate than that produced in 3.5.5.

(See 3.5.5 and 5.4.5 for result comparisons)

4.4.6 Scenario 6

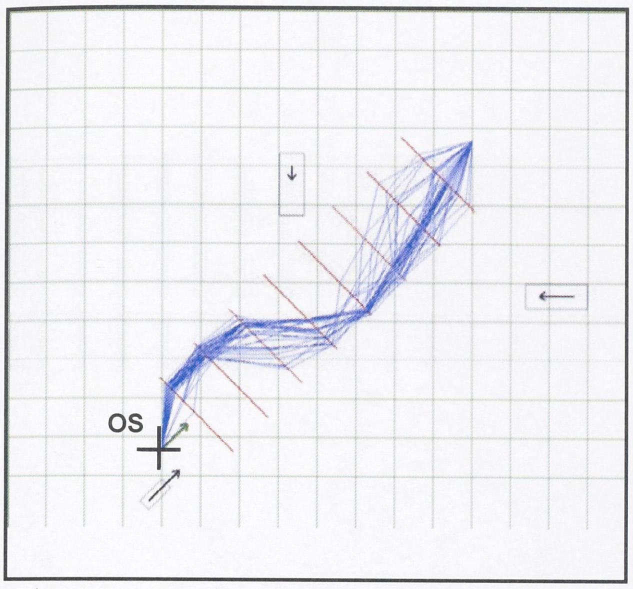


Fig. 4.23a Random Ant Routes after 20 Runs

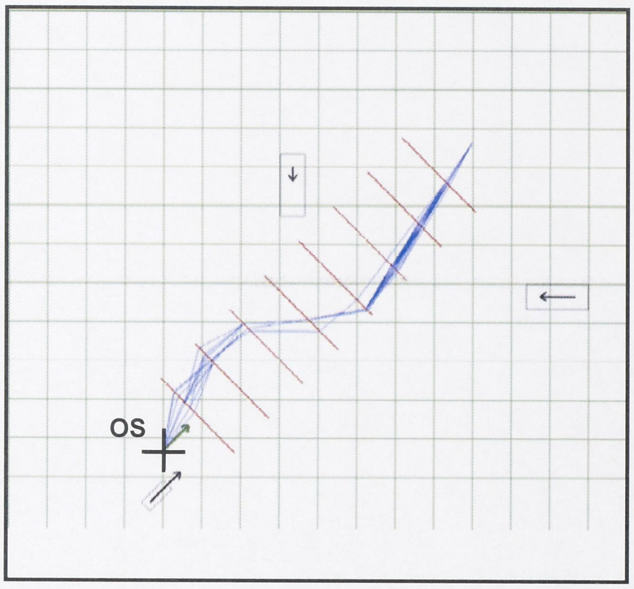


Fig. 4.23b Random Ant Routes after 100 Runs

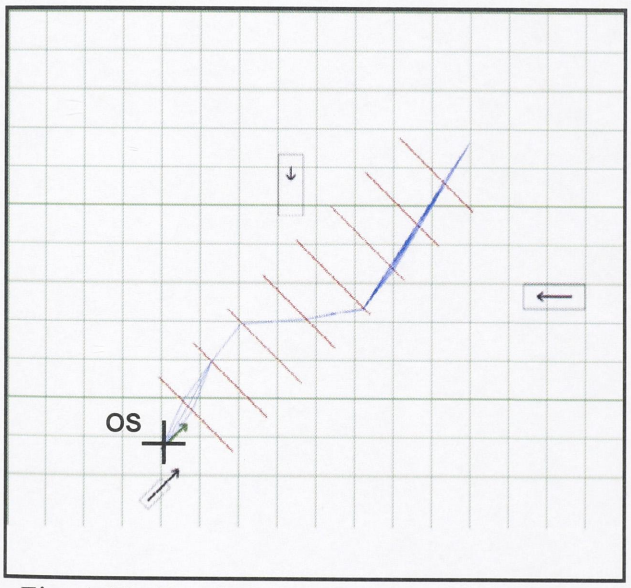


Fig. 4.23c Random Ant Routes after 200 Runs

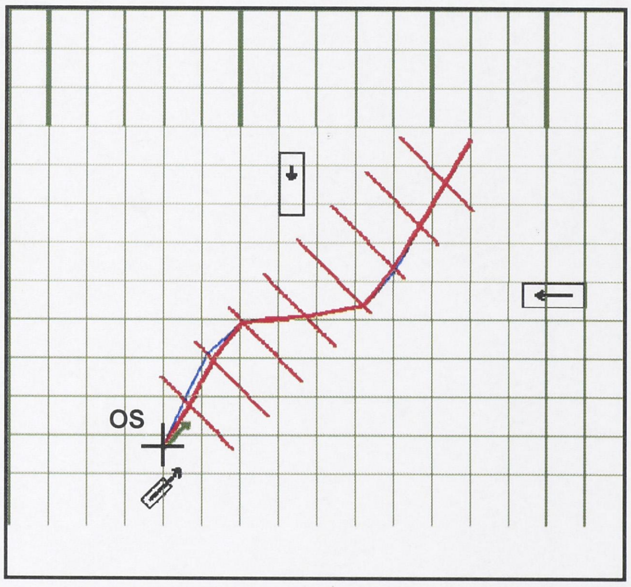


Fig. 4.23d Best Solutions after 100 Runs

The scenario serves to prove the resistibility and the robustness of the algorithm for different encountering cases at sea.

- The algorithm can still produce a collision-avoiding route while the algorithm using Dynamic-Programming fails.
- Ant colony converges promptly to the region of the globally optimum solution.
- The strategy is acceptable, from the rule application point of view.
- Best solutions found almost coincide with the optimal solution after 100 runs (Fig. 4.23d).

(See 3.5.6 and 5.4.6 for result comparisons).

4.5 Conclusions

The chapter is to propose an Ant Colony Optimization Algorithm (ACO) for generating the collision-avoiding route for the Own Ship. The route-generating algorithm is based on the assumption that Target Ships do not change their course and speed, and the collision is avoided by action of OS alone.

In comparison with ACO Meta-heuristic, our proposed ACO algorithm is modified in the following aspects, taking into consideration the nature of marine traffic

- A local search (post-processing) mechanism is applied to increase the search efficiency and the convergence speed.
- Only the ants (Np out of Na ants) that produce better routes lay trail-pheromone. Then unpromising regions of the search-space can be avoided.
- A scheme to limit pheromone-level is used to control the search.
- The searching-algorithm is globally supervised by the use of the best route that has been found at each run.

It has been shown that ACO is a suitable and very effective approach for the route-producing problem, taking into account the dynamic nature of the constraints. In comparison with Dynamic-Programming method, a sharp advantage of ACO is that the rules of the road can be actively taken into account, just by modifying the route-cost function (values of K).

With suitable choice of designing parameters as shown, the nearly-optimal route can be produced with in a short period of time (less than 15 seconds). The algorithm therefore meets the requirement of real-time application.

A deterrence of the route-producing algorithm is that its performance is rather sensitive to the variation of parameters. Changes in pheromone-delivering amount (4.9) or evaporation coefficient (4.8) may severely affect the convergence of the algorithm, especially the early convergence to the local optimums.

References

1. G. Ma, H. Dual, S. Liu, "Improved Ant Colony Algorithm for Global Optimal Trajectory Planning of UAV under Complex Environment", International Journal of Computer Science & Applications Vol.4, pp.57-68, 2007
2. J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem", Advanced Engineering Informatics, Vol. 18, pp. 41–48, 2002
3. M. Dorigi, M. Birattari, T. Stutzle, "Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique", IEEE Computational Intelligence Magazine, pp.28-40, 2006
4. M. Dorigo and T. Stützle, "Ant Colony Optimization", Massachusetts Institute of Technology, 2004
5. M. Dorigo, G. Di Caro, L. M. Gainbardella, "Ant algorithms for discrete optimization", Artificial Life, Vol. 5, No. 2, pp. 137-172, 1999
6. M. Dorigo, V. Maniezzo, A. Colorni, "Ant system: optimization by a colony of

cooperating agents", IEEE Transactions on Systems, Man and Cybernetics-Part B, Vol. 26, pp.29-41, 1996

7. M. Guntsch, M. Middendorf, and H. Schmeck, "An Ant Colony Optimization approach to Dynamic TSP", Proceedings of the Genetic and Evolutionary Computation Conference 2001, Morgan Kaufmann Publishers, pp. 860–867, 2001
8. M. Plucinski, "Application of the Ant Colony Algorithm for the Path Planning", Enhanced Methods in Computer Security, Biometric and Artificial Intelligence Systems, Chapter 3, pp.345-352, 2005
9. R. C. Eberhart and Y.H. Shi, "Particle swarm optimization: Developments, applications and resources", Proceedings of the IEEE congress on evolutionary computation, pages 81–86, 2001
10. R. Tinos and S. Yang, "Genetic algorithms with self-organized criticality for dynamic optimization problems", Proceedings of the 2005 IEEE Congress on Evolutionary Computation, IEEE Press, Vol. 3, pp. 2816–2823, 2005
11. T. Stutzle and H.H. Hoos, "Improving the Ant System: A detailed report on the MAX–MIN Ant System," FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep. AIDA-96-12, Aug. 1996
12. T. Stutzle and H.H. Hoos, "MAX–MIN Ant System," Future Generation Computer Systems, Vol. 16, No. 8, pp. 889–914, 2000
13. Y. Wang, J. Y. Xie, "An adaptive ant colony optimization algorithm and simulation", Journal of System Simulation, Vol. 14, pp.31-33, 2002

Chapter 5 Collision-Avoiding Route Generation by Adaptive Bacterial Foraging Optimization Algorithm

5.1 Introduction

As stated in Chapter 4, the route-producing algorithm based on Ant-Colony Optimization is a promising and effective tool to generate the collision-avoiding route for the ship in diversified maritime-traffic environments. However, the application of ACO algorithm itself copes with several difficulties that must be further improved:

- Firstly, the algorithm is very sensitive to the choice of parameters. An inappropriate choice of these parameters results in the very poor performance and the optimal route may never been reached at all.
- Secondly, the convergence of the ant-colony is rather slow in some cases.

A better algorithm therefore in considered in this study that possesses the pros of the ACO algorithm and overcome its limitations at the same time.

First introduced by Passino [6][11] in 2002, Bacterial Foraging Optimization Algorithm (BFOA) has been the subject of many researches in the last several years. Inspired by the bacteria forage over a landscape of nutrients, BFOA has been generally considered a promising solution for a variety of distributed optimization. The algorithm is a population based numerical optimization method which is simple but powerful and has been applied successfully to a wide range of engineering problems, including the optimal control, machine learning, harmonic estimation etc.

Being a population based bio-mimetic algorithm, BFOA is characterized by the following properties which make itself a more robust and effective method for optimal searching, in comparison with other gradient-based optimization methods:

- The individuals are distributed and autonomous. There is no central control and the failure of some individuals therefore can not influence the solving of the whole problem i.e. other individuals can still keep on seeking for the optima themselves independently. As a result, they tend to be more robust than other numerical algorithms.
- As the collaboration (swarming) is through indirect information communication, the algorithm is extensible. For a simple problem, a population of few bacteria can perform the search effectively. However, if the complexity of the problem increases, a larger population must be employed. Due to the extensibility of BFOA, it is easy to increase population size to achieve solution to these more complicated problems.
- The algorithm concerns only basic mathematical operations thus it can be simply and efficiently implemented on computer.
- The assumptions of differentiability, convexity and other mathematical conditions are not required. Hence, the algorithm is highly viable for a vast range of problems.

Then, to overcome the limitations of previously-mentioned algorithms for route-producing, in this chapter, an Adaptive-BFOA for generating the optimal route or an approximation of the optimal route to avoid collision for the Own Ship (OS) will be proposed, given the Target Ship (TS) motions, OS maneuvering characteristics and environmental constraints. It will be shown later that the Adaptive-BFOA is very efficient in timely route-producing and allows the realization of maritime traffic rules as stated in International Convention for Preventing Collision at Sea. It also removes the limitation of Dynamic Programming Algorithm in treating the problem as time-invariant. With suitable choice of designing parameters, the algorithm has admirable exploration, exploitation and convergence properties in comparison with the ACO based algorithm.

In this chapter, BFO fundamentals and a classical BFOA will be described in Section 5.2. Then the Adaptive-BFOA for producing collision-avoiding route of OS will be discussed in details in section 5.3. Computer simulation results will be shown and analyzed in section 5.4. Lastly, conclusions relating to the Adaptive-BFOA application will be stated in section 5.5.

As mentioned earlier, the collision-avoiding route will be produced with the following 2 assumptions:

- Target Ships do not change their speeds and courses.

- Collision avoidance is the duty of our Own Ship alone, even for the cases where OS is a stand-on vessel.

5.2 Bacterial Foraging Optimization Fundamentals and Classical Algorithm

5.2.1 Bacteria Foraging Optimization Fundamentals

Suppose that it is necessary to find the minimum of a function Q(S) defined in a certain domain which is often called the Solution-space or the Search-space. Additionally, a solution S must satisfy some constraints. In many practical applications, the problem is so complicated that it is impossible or too costly to have a measurement or analytic description of the gradient $\nabla Q(s)$. It is therefore referred to as a non-gradient optimization problem which can not be solved analytically. Rather than the exact solution of the optimal, a close approximation of it is applausive. An effective search for such an approximation is the target of numerical optimization method, including nature-inspired algorithms.

Being a bio-mimetic algorithm, BFOA is an iteration based optimization tool using an initial set of solutions which is generated randomly. Each solution is represented by the position of a bacterium. Then, throughout this chapter, the terms position of bacterium, bacterium and solution are used interchangeably. The iteration is the development of the population in a life-time of the bacteria. It undergoes the following stages:

- Each bacterium adapts itself to the environment and grows.
- Fitness of each solution is measured. The fittest solutions retain and reproduce while the less fit ones are removed from the population.
- Remaining solutions swarm.

The new solutions after each iteration are fitter than the original ones. As a matter of fact, they are nearer to the optimal which must be found.

The following nomenclature will be used throughout this section to illustrate a classical BFOA and its general modifications.

$S = [p(1), p(2), ..., p(i), ..., p(N)]$ *a solution (position of bacterium in search space)*

where $p(i)$ *: the* i^{th} *coordinate*

N *: Dimension of the search space*

$V = V[v(0), v(1), ..., v(i), ..., v(N)]$ *a vector representing a moving direction*

where $v(i) = 0$ *or* $v(i) = 1$

D *: Distance to move (i.e. Swim length)*

Q*: the fitting index (sometimes refered to as Quality – Index or Cost – Index)*

dQ*: added fitness due to bacterial communication*

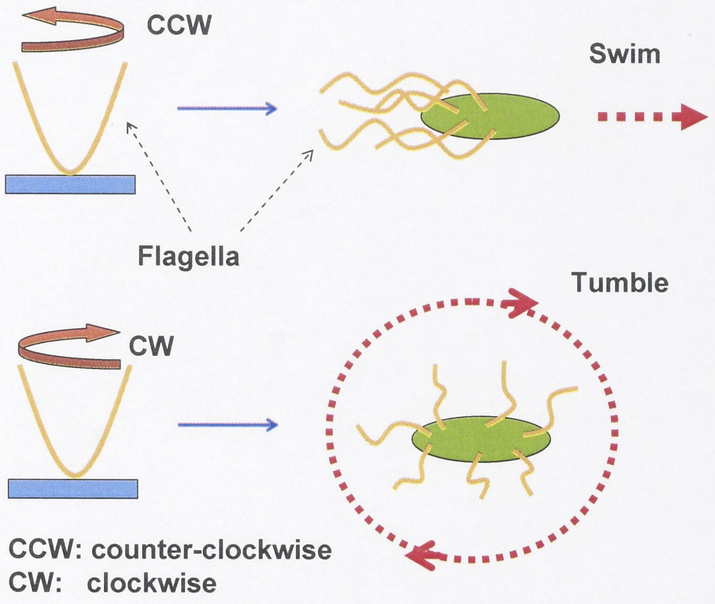


Fig. 5.1 Chemotaxis of Bacterium

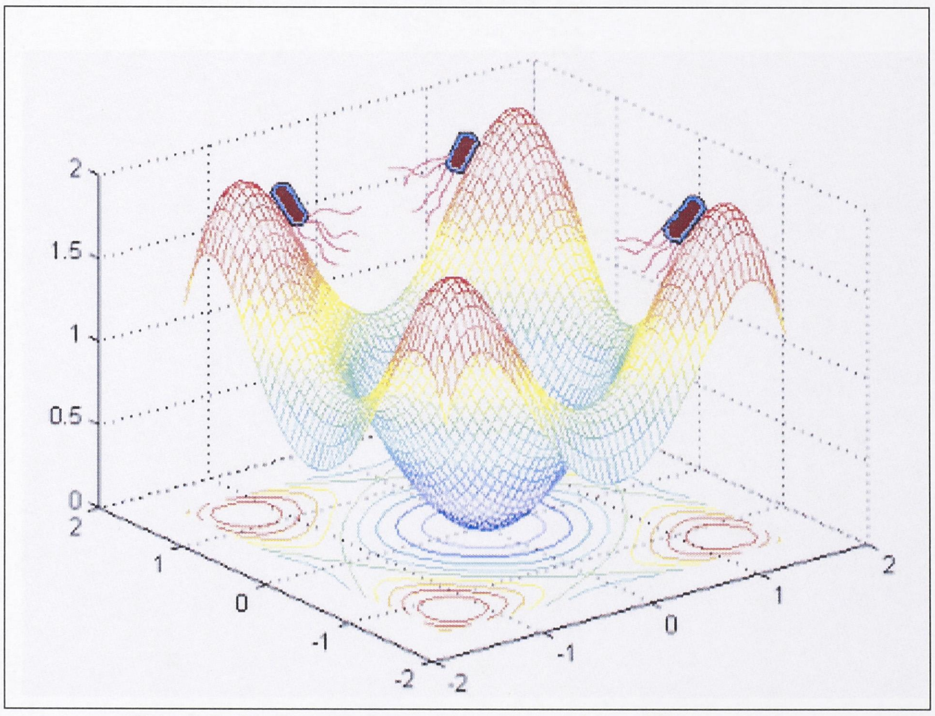


Fig. 5.2 Optimum Searching Behavior of Bacteria [4]

5.2.1 Classical BFOA

In short, a classical BFOA consists of the following 3 steps:

Step 1: Initialization. Initialize the bacteria of the population. Each bacterium is allocated randomly over the search-space.

Step 2: Evolution. Recursively manipulate the population to develop gradually by a 3-step procedure:

- Chemotaxis and swarming

- Reproduction
- Elimination and Dispersal

Step 3: Termination. Return the solution accompanying with the fittest bacterium in the population.

As mentioned in step 2, in nutrient foraging and evolving, bacteria population recursively goes through a process of 3 stages, namely the chemotaxis and swarming, reproduction, elimination and dispersal. The process enables the bacteria to gradually aggregate in the most favorable region or the region of highest concentration of nutrient. Imitating foraging behavior of E.coli bacteria, BFOA seeks the optimum of a function through the search-space by conducting the local search through the bacteria chemotaxis, distributing local search by swarming, intensifying search in promising region by reproduction, and avoiding traps of local optimums by elimination and dispersal.

5.2.2.1 Chemotaxis

Chemotaxis is the phenomenon in which bacteria, including E. coli, direct their movements according to the existence of certain chemicals in their environment. It is important for bacteria to move to areas of higher food concentration or to flee from poison. Chemotaxis is either positive or negative. In positive chemotaxis, the movement of bacteria is toward the positions of higher concentration of the chemical concerned. Conversely, negative chemotaxis causes bacteria to move away from them.

The movement of bacteria is the result of alternating tumble and swim phases. These 2 phases, performed through their entire lives, direct bacteria to search for the position or region where the food concentration is richest.

Swimming is the straight run (sliding) motion of bacteria in a pre-chosen direction. It is the result of rotating their flagella counter-clockwise. In BFOA, swimming can be considered as the straight motion of a bacterium up hill, toward the position of local optimum. If the distance of swim is long, bacteria move quickly toward the optimum. However, it may also cause the search to fluctuate around this point. A small swim-distance allows the bacteria to slide slowly but steadily to region of high nutrient concentration. The expenses accompanying with it are that it requires more time for the bacteria to reach optimum is long and bacteria are incapable of jumping out of the attractive area of a local optimum.

Tumbling, on the other hand, is the turning motion of bacteria by rotating their flagella clockwise. It is the action of a bacterium to change its moving direction while seeking for food. In BFOA, it is equivalent to starting the search in a new direction.

The 2 basic motions of a bacterium are illustrated in Fig. 5.1. Combination of the 2 chemotactic motions is rather random. A tumble may be followed by a tumble or a swim and vice verse. In fact, E.coli bacteria are unable either to decide the direction in which they swim or to swim in a straight line for more than a few seconds due to rotational diffusion. In other words, bacteria "forget" the direction in which they are going. By repeatedly evaluating their course, and adjusting if they are moving in the wrong direction, bacteria can direct their motion to find favorable locations with high concentrations of attractants (usually food) and avoid repellents (usually poisons).

$$S^i(j+1) = S^i(j) + V(j)\ x\ D \qquad (5.1)$$

where

i : *index of bacterium in the population*

j : *chemotatic step in bacteria life time*

The chemotactic (5.1) is applied if the following fitness inequality is satisfied.

$$Q(S^i(j+1)) \ < \ Q(S^i(j)) + dQ(S^i(j+1)) \qquad (5.2)$$

where dQ represents the total attractive / repellent forces between individuals

The overall motivation for mimicking bacteria's foraging behavior is illustrated in Fig. 5.2 for a search in 2-dimension space. By repeatedly undergoing chemotactic motions, bacteria gradually climb up the hills to approach the optimums, locally or globally.

5.2.2.2 Swarming

Swarming is an interesting aggregation behavior of bacteria swarm. This is achieved by a primary communication mechanism between bacteria in close proximity.

Due to the premature structure of bacteria cells, a bacterium can exchange signal with just the bacteria nearby. The signaling scheme results in an extra factor basing on which bacteria decide their chemotactic moves. Unfortunately, the mechanism behind this phenomenon is very complicated and has not yet been fully explainable. In BFOA, it is simulated by the attractive or repellent forces exerted by a bacterial cell on the other. A number of parameters must be chosen to express this effect when designing BFOA. In brief, it can be simply referred to as a process in which bacteria are attracted by those within a certain range from their positions and to be repelled when distance between them falls below a limit. If properly defined, it redistributes the search in a promising local region around the optimum and to draws bacteria from other regions to enter the most attractive region of the current search.

$$dQ(S^i(j+1)) \ = \ \sum_{k=1}^{Ns} dQ(S^i(j+1), S^h(j)) = \sum_{k=1}^{Ns}\left[-d_{attract}\exp(-w_{attract}\left\|S^i(j+1) - S^k(j)\right\|^2)\right]$$
$$+\sum_{k=1}^{Ns}\left[-h_{repellent}\exp(-w_{repellent}\left\|S^i(j+1) - S^k(j)\right\|^2)\right] \qquad (5.3)$$

where

$\|S\|$: *norm of vector S*

$d_{attract}, w_{attract}, h_{repellent}, w_{repellent}$: *coefficients representing range and magnitude of attractive / repellent force between bacteria*

5.2.2.3 Reproduction

After a number of chemotactic steps, some bacteria are in regions of better nutrient content than the other. Those bacteria therefore gain more food and become stronger (i.e. healthier/longer). The healthier bacteria have more chance to reproduce. Contrarily, the less healthy bacteria will eventually die before reproducing.

In BFOA, the healthier bacteria are commonly those being in better regions of the search-space, while the weaker ones are lying in poor regions of the function values. The reproduction of healthier bacteria therefore increases the number of bacteria in more favorable regions so as to intensify the search in these regions. The number of bacteria to reproduce is chosen to be the same with number of those who die to keep the population size unchanged. The process therefore increases the optimum seeking speed in more promising regions and gives up the less favorable ones.

A disadvantage of this mechanism is that bacteria which are actually in the region containing the global optimum may be killed before they can approach the optimum so as to gain nutrient in large volume. Accordingly, the region is therefore not properly searched. This must be carefully taken into consideration in designing BFOA.

5.2.2.4 Elimination and Dispersal

In nature, due to the sudden regional changes of the environment, e.g. temperature variation, a number of bacteria in a certain region of the search-space die. On the other hand, with favorable conditions, new bacteria arise in other region. Inspired by the phenomenon, BFOA undergoes an elimination and dispersal process in each generation.

Elimination is the act of removing bacteria. A number of bacteria except several healthiest one are selected randomly and removed from the bacteria population (elimination). They are replaced by the same number of bacteria randomly dispersed around the search region (dispersal). Thence, the total number of bacteria in the population remains unchanged. As stated above, premature convergence, or the convergence to local optimum rather than the global one is a deterrence of almost all numerical optimization methods currently in use. For BFOA, due to the randomness of the bacteria elimination and dispersal process, the possibility for every local region to be searched can be significantly increased.

The elimination and dispersal may destroy the chemotactic progress, but it also can be of help in improving speed of chemotaxis because the bacteria may be placed in a better region of nutrient concentration. By this diversification of search, the ability for some bacteria to reach the global optimal can be significantly strengthened. In this study, an appropriate elimination scheme will be proposed to ensure that every local region have been thoroughly exploited it is abandoned.

5.2.3 BFOA Limitations and Modifications

Experiments with complex and multimodal bench mark functions have revealed that the convergence property of the classical BFOA is poor and its performance heavily decreases with the growth in dimensionality of the search-space and the problem complexity.

To overpass these limitations, different variants of BFOA have been proposed to improve its searching performance.

Tripathy et al. [7] proposed an algorithm in which 2 modifications are applied:

- The minimum value of quality function at each position is used instead of the average value of all chemotactic steps so that the convergence speed is increased.
- The distance to the position of globally optimal bacterium is considered as a factor for modifying chemotactic move at any step, rather than distances to all other bacteria.

Mishra [8] suggested a fuzzy inference scheme to select the optimal chemotactic step size in BFOA.

C. Ying et al. [3] proposed a scheme in which the swarming pattern is inspired by that of a particle swarm optimization algorithm. The position of each bacterium after every move in the resulting algorithm (called a Fast Bacteria Swarming Algorithm – FBSA) is then decided as followings

$$\begin{aligned}&\textit{if } Q(S^b(j) < Q(S^i(j+1)) \textit{ then}\\ &S^i_{new}(j+1) = S^i_{old}(j+1) + C_{cc}\ x\ (S^b(j) - S^i(j)) \qquad (5.4)\\ &\textit{where b is the best bacterium in previous chemotactic step, } C_{cc} \textit{ is an attraction factor}\end{aligned}$$

Furthermore, the step length is gradually reduced after completing a cycle of iteration.

Similarly, H. Chen, Y. Zhu and K. Hu [4] developed the Adaptive Bacteria Foraging Optimization Algorithms (ABFA$_{0,1}$) in which the chemotactic step size is adjusted according to the required accuracy at each stages, namely the exploration and the exploitation. In exploration, a large swim-length unit is employed to explore the previously un-scanned regions. For exploitation, on the other hand, a small swim-length unit is applied to exploit a promising region.

Other modifications of BFOA can be found in [1][4][9] etc. in which the algorithm is combined with other optimization methods so as to improve its performance

Those variants however, are application-oriented and the choice of parameters is inconsistent. In this study, the author is to propose an algorithm with an adaptive chemotactic step size taking into consideration the fact that there are regions the ship can not interfere. The repellent forces are exerted on neighboring bacteria. The proposed algorithm will then be verified with a set of study cases to prove the required convergence property.

5.3 Collision-Avoiding Route Generation System Based on BFOA

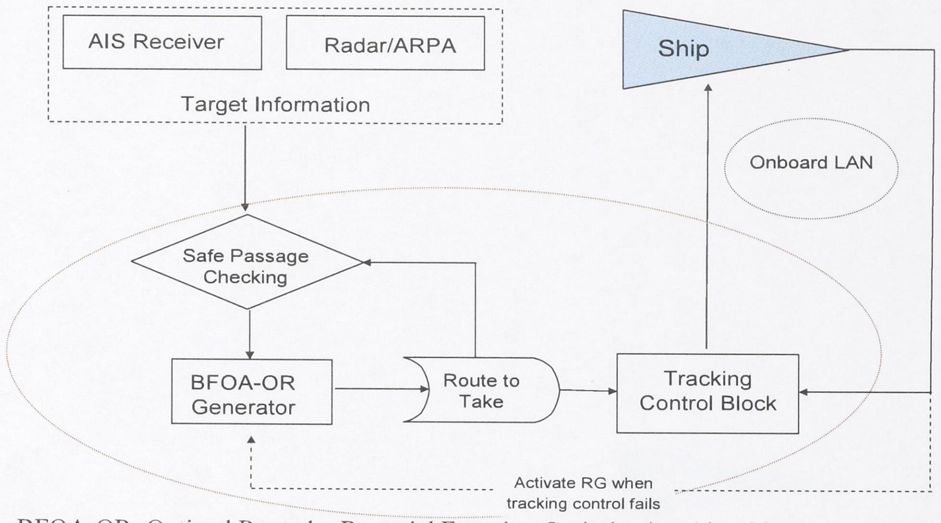
5.3.1 System Overview

The overall system configuration is illustrated in Fig. 5.1, similar to those defined in Chapter 3 and Chapter 4. The only difference is that the collision-avoiding route for the ship is produced by an Adaptive-BFOA-based route generator. As previously described, the inputs to the generator are still the TS information, OS maneuvering characteristics and the environmental constrains. The optimal route which is the aim of the searching algorithm proposed in this chapter is not the minimum time route (Chapter 3) but the safe route, i.e. free from collision with all TS and environment constraints, that requires minimum cost like that in Chapter 4.

The formula for route cost calculation will be described thoroughly in section 5.3.2. The realization of the route thereby generated is the duty of a tracking control block connected to OS control system though the network.

It should be noted again that the overall system is a real-time, autonomous process which is keeping watch for the safe passage of the OS at all time. A new collision-avoiding route is generated whenever there arises any potential endangerments due to one or some of the following reasons:

- OS seriously deviates from the generated route
- An existing TS alters course or changes speed
- A new TS interferes OS safe passage, etc.



BFOA-OR: Optimal Route by Bacterial Foraging Optimization Algorithm
RG: Route Generator

Fig. 5.3 System Structure Overview

The optimization problem which BFOA has to solve is a combination of points of the produced grid, one point on each line, following which, OS will neither be in risk of collision with TS nor violate environmental constraints. Simultaneously, the route should be as short as possible and satisfy the traffic regulations the most.

The route-producing procedure is illustrated in the following flow chart (Fig. 5.4). It should be noted again that a grid system with around 10 lines will be applied. The distance between lines is 1700 – 2000m. The problem is therefore a high-dimensional and non-continuous problem. The Adaptive-BFOA must be properly designed to achieve good convergence for this optimization function form.

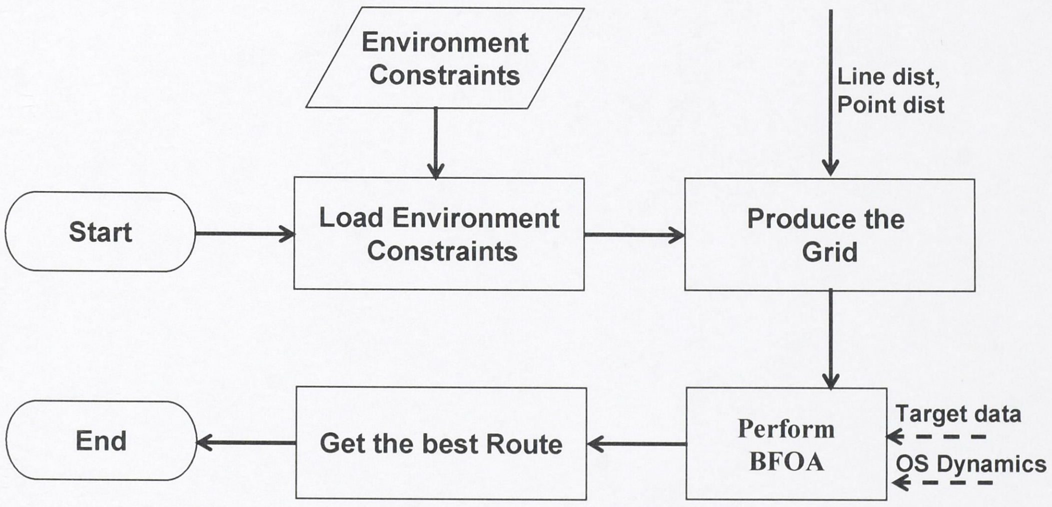


Fig. 5.4 General Route-Producing Procedure by BFOA

5.3.2 Route Cost Function

In this chapter, the cost of a route (or a Strategy/Solution interchangeably) is evaluated in the same manner with that used in Chapter 4. However, it is still mentioned here for the continuity of the chapter contents.

Route-Cost evaluation:

$$Q = \left(\sum_i T_i\right) \times \left(1 + \sum_j K_j\right)$$

$$\begin{aligned} &i = 1 \text{ to Number of Connections on the Route} \\ &j = 1 \text{ to Number of TS involved in the situation} \\ &T_i : \text{Time required to travel connection } i^{th} \text{ on the Route} \\ &K_j : \text{Additional cost due to rule infringement when avoiding TS } j^{th} \end{aligned} \qquad (5.5)$$

The choice of coefficient K in (5.5) for different encountering situation between the OS and a certain TS is defined as the followings

(1) Passing a Head-on TS on OS Starboard side: K = 0.2.
(2) Passing a TS Crossing from Starboard side on OS Starboard side (OS give-way): K = 0.05.
(3) Turning to Port while the TS is crossing from Port side (OS Stand-on): K = 0.1.
(4) Turning to Starboard while the TS is behind the OS traverse axis on starboard side and overtaking: K = 0.2.
(5) Turning to Port while the TS is behind the OS traverse axis on port side and overtaking: K = 0.2.
(6) Otherwise, K = 0.0.

5.3.3 Optimization Problem Modeling

Basing on the TS motions and environmental constrains, a suitable grid system could be built in the navigable water area for the OS to pass safely. The task assigned for the optimal route generator is to determine the best route, i.e. the minimum cost route for the OS. Every route can be represented by a vector containing indices of points on the grid lines as the followings:

$$\begin{aligned} &S = [p(1), p(2), ..., p(i), ..., p(N)] \\ &\text{where} \\ &\quad p(i) \in [1 \text{ to Number of points on grid line } i^{th}] \\ &\quad N: \text{Number of grid lines} \end{aligned} \qquad (5.6)$$

The proposed Adaptive-BFOA is to find the best strategy for the OS, i.e. a strategy $S_0 = [p_0(1), p_0(2), ..., p_0(i), ..., p_0(N)]$ that requires the minimum cost.

With the route cost value as defined in 5.3.2, the task is a global optimization searching problem in a space of N dimensions.

$$Q_0 = Min\ Q(S) = Q(S_0) \qquad (5.7)$$

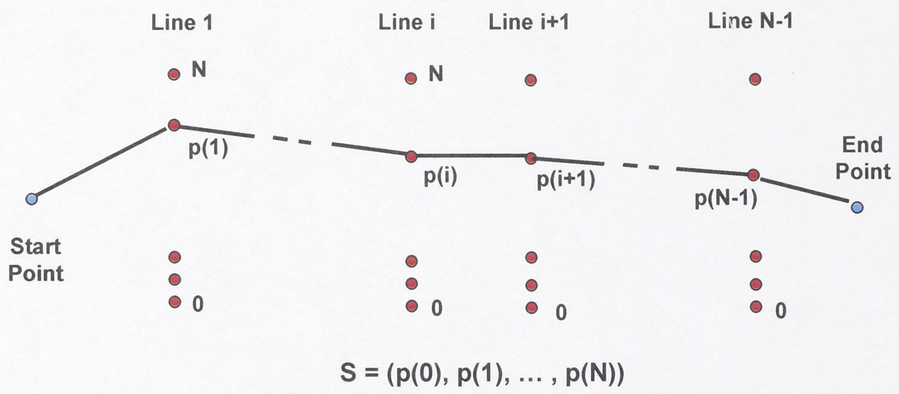


Fig. 5.5 Bacterium Position or OS Collision-Avoiding Route

Taking as an example the simple Head-On encounter situation, with the assumption that only the OS takes actions to avoid collision, it is easily seen that the set of possible strategies for the OS includes strategies in which the OS alters first to starboard and those in which the OS changes its course to port. Among those, there is at least a starboard minimal cost route and a port minimal cost route which are approximately illustrated in Fig. 5.3. The problem is therefore a multi-optimum global optimization. As it is highly nonlinear, an analytical method can not be easily applied. BFOA (or other approximation searching methods) is thus recalled naturally.

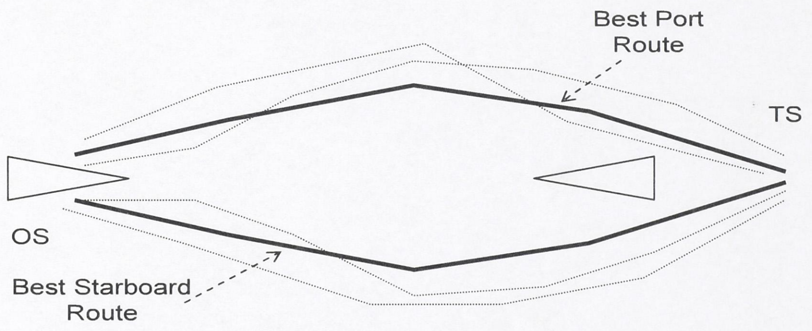


Fig. 5.6 Sample Collision-Avoiding Strategies in Head-on Encounter

5.3.4 Bacteria Position Initialization (Solution Initialization)

The procedure is to initialize positions of bacteria randomly. As mentioned earlier, a bacterium position (or a solution) S is a combination of points on the grid lines. Thus, the task of initializing process is to seek a random combination of these points that makes the solution viable i.e. a safe route for OS.

The algorithm for solution initializing is illustrated by the follow chart in Fig. 5.7, where it is driven by 2 designing parameters:

- Kmax: Maximum total trials on all components of the solution.
- Rmax: Maximum trials on a component, where a component is a point (p(i), i = 1 to N).

The variable l is used to count the component index i.e. the line number on the grid, and is therefore initially set to 0.

From a point p(l) on line lth, a point p(l+1) on line (l+1)th is chosen randomly and safety criterion is applied to check if the latter point can be accepted as a solution component i.e. OS can reach p(l+1) safely from p(l).

If it is safe, p(l+1) is appended to the solution: S(l+1) = p(l+1).

Otherwise, another point is tried until r reaches Rmax.

If it is unable to find the component p(l+1) after Rmax trials, the initializing process is shifted back several steps (3 steps in this study) and to continue from there.

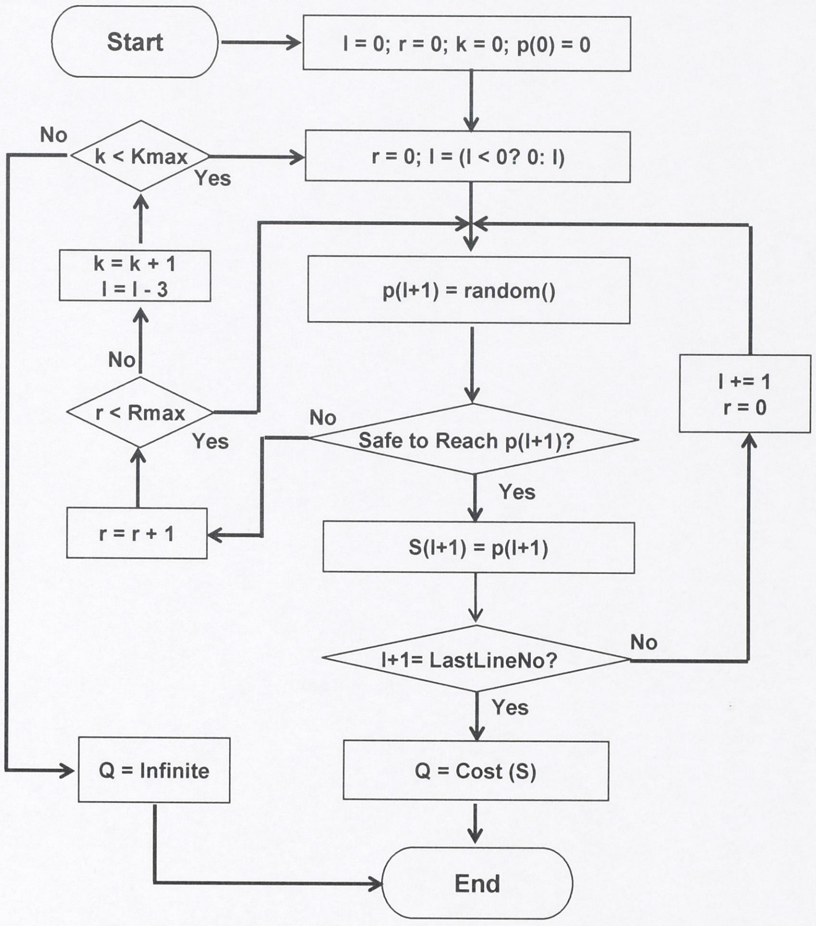


Fig. 5.7 Collision-Avoiding Route Initialization

If it is possible to find the whole combination, the solution is assigned for the bacterium. Solution cost, or the bacterium heath equivalently, is calculated accordingly. Otherwise, the cost is set to infinite to denote that an appropriate position for the bacterium has not yet been determined.

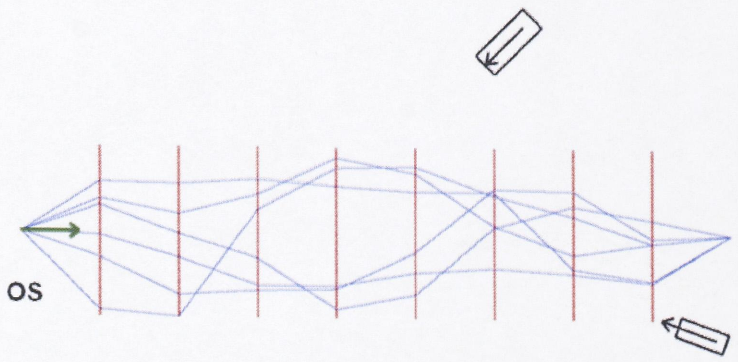


Fig. 5.8 Random Initial Solutions – 6 Solutions

In Fig. 5.8, the above solution initializing procedure is applied to disperse 6 bacteria, given the TS sizes and motions. It is obvious that due to the randomness of the initialization, solutions are scattered over different regions of the search-space, including solutions in which one among the following strategies is used:

- OS passes both TS on their port side.
- OS passes both TS on their starboard side.
- OS passes upper and lower TS on the port side of the former and starboard side of the latter.

Then, it is guaranteed that the overall route searching algorithm is to explore the entire search-space to achieve the global optimum.

5.3.5 Bacteria Chemotaxis Procedure

Starting from their initial positions, bacteria exploit the area around them for better solution. The local area searching process is performed by bacteria's chemotaxis, including the swimming and the tumbling and is illustrated by the flow chart in Fig. 5.9.

A tumble is represented by a vector V denoting the direction to which a bacterium is seeking. V is defined as

$$\begin{aligned}&V = [v(1), v(2), ..., v(i), ..., v(N)]\\&\textit{where}\\&\quad v(i) = 0 \textit{ or } v(i) = \pm 1\\&\quad N\textit{: Number of grid lines}\end{aligned} \tag{5.8}$$

Due to the high dimensionality of the problem, i.e. large N, there are a huge number of combinations of 0 and 1 in vector V. In this study, the choice of V is limited to the following combinations

$$\begin{aligned}&V_1 = [0,..., 0,1,0,...,0]\, i.e.\ v(i) = 1 \textit{ if } i = k;\ v(i) = 0 \textit{ otherwise}\\&V_2 = [0,..., 0,1,1,0,...,0]\, i.e.\ v(i) = 1 \textit{ if } k \le i \le k+1;\ v(i) = 0 \textit{ otherwise}\\&V_3 = [0,..., 0,1,1,1,0,...,0]\, i.e.\ v(i) = 1 \textit{ if } k \le i \le k+2;\ v(i) = 0 \textit{ otherwise}\\&V_4 = [0,..., 0,-1,0,...,0]\, i.e.\ v(i) = -1 \textit{ if } i = k;\ v(i) = 0 \textit{ otherwise}\\&V_5 = [0,..., 0,-1,-1,0,...,0]\, i.e.\ v(i) = -1 \textit{ if } k \le i \le k+1;\ v(i) = 0 \textit{ otherwise}\\&V_6 = [0,..., 0,-1,-1,-1,0,...,0]\, i.e.\ v(i) = -1 \textit{ if } k \le i \le k+2;\ v(i) = 0 \textit{ otherwise}\\&V_7 = [0,..., 0,1,0,...,0,-1,0,...,0]\, i.e.\ v(i) = 1 \textit{ if } i = k_1; v(i) = -1 \textit{ if } i = k_2;\ v(i) = 0 \textit{ otherwise}\\&\textit{where } k, k_1, k_2 \textit{ are random value produced at each tumble}\end{aligned}$$

Then, a tumble is a probabilistic choice of a vector V from the above set, where V_1, V_4 are chosen more frequently than V_2 and V_5. In turn, the later vectors are used slightly more often than the rest.

After each tumble, the bacterium undergoes one or several swims, depending on the success of the search in that direction. A new solution is produced from the current one using vector V and swim-length D (see section 5.3.6):

$$S' = S + V.D$$

The new solution (S') is first compared with the current one. If it is shorter, it might be an improvement solution for the current bacterium position. In that case, the safety of that solution is checked, using the appropriate criterion of collision risk assessment. If the solution is also viable, its cost is compared to the current route. Note here that an additional cost value dQ is included to express the influence of the communication between bacteria (see following section).

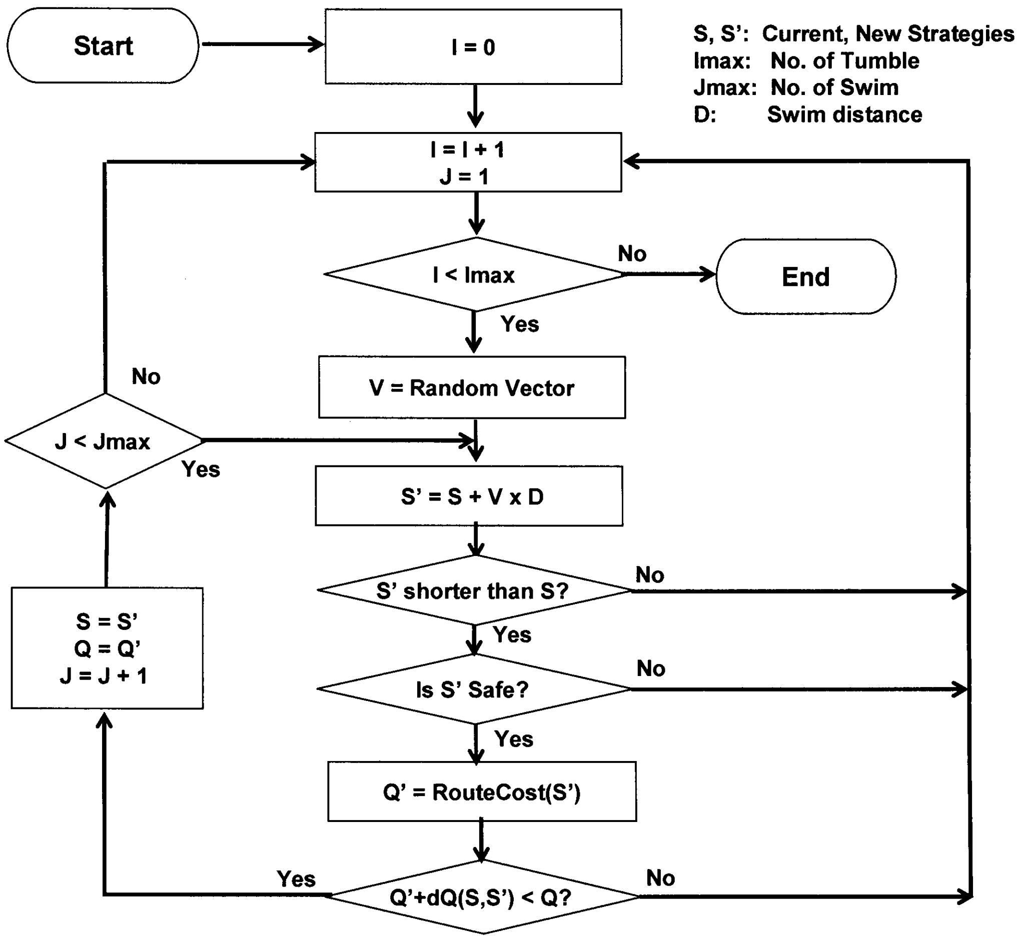


Fig. 5.9 Bacteria Chemotaxis Procedure

If the new solution satisfies all the requirements, bacteria move to this new position and the chemotatic process is to continue from there. The process is repeated until a pre-defined number of chemotatic moves (both tumbling and swimming) have been tried.

To illustrate the effect of the local searching process by chemotaxis of bacteria, the improved solutions from the initial ones which are shown in Fig. 5.8 are presented in Fig. 5.10. In this figure, it is clearly seen that the solutions converge nicely to their respective local optimums. Each region corresponds to a strategy the officer may use: port-to-port for both TS, starboard-to-starboard for both TS, port-to-port for one TS and starboard-to-starboard for the other.

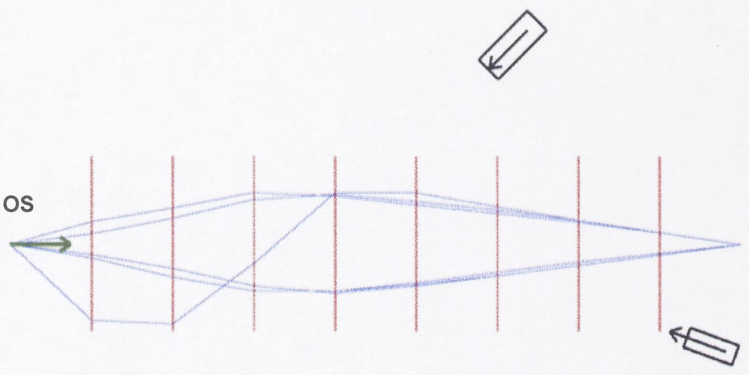


Fig. 5.10 Improved Solutions after Chemotaxis – 6 Solutions

5.3.6 Modifications to BFOA to Enhance Performance

5.3.6.1 Swim-Length Adapting Mechanism

The choice of move-length (or swim-length) has been recognized to be the most decisive factor controlling the performance of BFOA.

A long swim allows the bacterium to move quickly toward the area around the optima (local or global). However, the further use of long swim results in the fluctuation of the around an optima. A short move, on the other hand, enables a slow but steady approach toward the optima. Another shorting coming of short move is that the bacterium can not jump out of the attraction area of a local optimum. Then, many different adaptive schemes therefore have been proposed by researchers working on the field.

In this study, a scheme of adapting the bacteria's move-length is applied. The adapting mechanism is as followings:

- Step 1: The move-length is initially set rather long (d_{long}) to allow the bacterium to swim quickly toward promising regions.
- Step 2: If the solution cost is not reduced for a certain number of consecutive chemotatic moves, the move-length is reduced to d_{medium}.
- Step 3: If the move-length in step 2 fails in producing better solution for the bacterium after a number of consecutive chemotatic moves, the move-length is further reduced to d_{small}.
- Step 4: If there is no quality increase after a number of consecutive moves, the swim-length is reset back to that in Step 1.

Due to this adapting mechanism, the algorithm has its name: Adaptive-BFOA for producing the collision-avoiding route. The mechanism allows the route searching process to approach the optimum quickly and steadily. The swim-length reset action in Step 4 enables the bacterium to try solutions in other area of the search-space rather than being trapped in the current local optimum.

5.3.6.2 Cell to Cell Communicating Mechanism

The aim of this cell to cell (or bacterium to bacterium) communicating mechanism is to allow the bacteria to coordinate in a way that improves the efficiency of the searching process.

In this study, the distance between 2 bacteria (solutions) is defined by:

$$S1 = [p1(1), p1(2), ..., p1(i), ..., p1(N)]$$
$$S2 = [p2(1), p2(2), ..., p2(i), ..., p2(N)]$$
then
$$d_{S1-S2} = \sqrt{\sum_{i=1}^{N} (p1(i) - p2(i))^2}$$

The neighbors of a bacterium are bacteria which are located with in a limit distance from the former bacterium. It is desirable for the searching algorithm that two or more bacteria do not assume the same position or too close from each other, so that the local search is more effective. Then, an additional cost (dQ in Fig. 5.9) is used by the bacterium to decide whether it should move from its current solution (S) to a new solution (S'). dQ is calculated as followings:

Let

$\{S_1, S_2, ..., S_M\}$ *to be the set of neighbor bacteria of S*

$\{d_1, d_2, ..., d_M\}$ *are distances from S to its neighbor bacteria*

$\{Q_1, Q_2, ..., Q_M\}$ *are routed*$_1$*, d*$_2$*,..., d*$_M$*} are costs coresponding to S*$_i$

Define
$$dQ(S) = \sum_{k=1}^{M} \left[-Q_{expellant} \left(1 - \frac{d_k}{d_{\max}} \right) \right]$$
Similarly,
$$dQ(S') = \sum_{k=1}^{M'} \left[-Q_{expellant} \left(1 - \frac{d'_k}{d_{\max}} \right) \right]$$

Then
$$dQ(S,S') = dQ(S') - dQ(S)$$
$$\textit{if } (Q_{S'} < Q_S) \textit{ and } (Q_{S'} < Q_i (\textit{for all } i = 1 \textit{ to } M)) \textit{ then } dQ(S,S') = 0$$

The parameter d_{max} denotes an expellant region (neighborhood) around a bacterium, and $Q_{expellant}$ is the maximum expelling force. This communicating mechanism allows the bacteria to scatter themselves in the area around the optimum so that the optimum can be found quickly and bacteria are not all trapped in a local optimum.

5.3.6.3 Multi-Steps Searching Algorithm

The main deterrence of the BFOA is its poor performance in problem of high dimensionality. The randomness in selecting a search direction (tumble) makes the bacteria's search meticulous and time consuming. However, the ship officer normally uses just several courses (say 4 courses) in actual encounters at sea. Then, to improve the search speed at initial stage, it is wise to combine the initial solutions as determined by the above process (5.3.4) with a set of reduced initial solutions containing just 3 intermediate way-points:

$S = [u,...,u, p(i), u,...,u, p(j), u,...,u, p(k),...,u]$

where

u: undefined points

i,j,k: random integer in the range [1,N – 1]

The algorithm for producing the reduced solution is principally similar to that shown in Fig. 5.7. As the solution is defined by just 3 way-points, the tumbling vector V can be limited to the followings:

$V = [0,...,0, v(i), 0,...,0, v(j), 0,...,0, v(k),...,0]$

where v(i), v(j), v(k) are chosen from the set $\{-1, 0, 1\}$

The undefined components are later determined from the intermediate way-points for the reduced solution after a number of chemotatic moves. Then, it can be treated as normal initial solution.

A suggestion to the combination of the reduced solution and full solution is 1 to 1 ratio.

5.3.7 Overall Adaptive-BFOA for Route-Producing

This section is to summarize the procedures described earlier in the route-producing algorithm basing on Adaptive-BFOA. For simulation studies, the algorithm is coded in VB programming language for easy debugging and modification. Calculation speed is not the major subject of this study, as far as the optimal route or a route very close to the optimal can be achieved in a short period of time (around 10 seconds).

The overall route-producing algorithm can be illustrated by the following pseudo-code:

```
A. Initialization
  Initialize_Grid(N_line, N_point, D_point);

  For bac = 1 to Ns
      Initialize_Bacterium(B(bac));
  Next bac

B. Evolution

  For cycles = 1 to N_cyc
      For bac = 1 to Ns
          For chemo = 1 to Nc
              Perform_Chemotatic_Move(B(bac));
          Next chemo

          If(Number_of_ Unsuccessful_ Move > N_size_converted_to_[large/medium/small]) then
              Convert_move_length_from_[large/medium/small]_to_[medium/small/large]();
          End if
      Next bac

      Sort_the_Bacteria_and_Cost_Arrays_by_Ascending_RouteCost(B(Ns), Q(Ns));
```

```
    For die_no = 1 to Nr
       If ( Chemotatic_Move_of_Bacterium_Count(B(die_no))> N_steps_to_die ) then (*)
          Kill_bacterium(B(die_no));
          B(die_no)= Reprocude_Bacterium(B(Ns - die_no));
       End if
    Next die_no

    For disperse_no = 1 to Nd
       rand = produce_random_interger()
       Initialize_Bacterium(B(rand));
    Next disperse_no
  Next cycles
```

C. Termination

```
  Sort_the_Bacteria_and_Cost_Arrays_by_Ascending_RouteCost(B(Ns), Q(Ns));
  Return B(1);
```

Where the variables and designing parameters are defined as:

N_line: Number of lines on the grid
N_point: Number of points on a grid line
D_point: Distance between points on a line
N_cyc: Number of cycles in the algorithm i.e. number of generations of the bacteria population.
Ns: Number of bacteria in the population
B(Ns): Bacteria population (bacteria set)
Q(Ns): Cost array of the bacteria
Nr: Number of bacteria died/reproduced in a cycle
Nd: Number of bacteria eliminated/dispersed in a cycle
Nc: Number of chemotatic steps of a bacterium in a cycle
N_size_converted_to_large: Number of unsuccessful chemotatic move before converting the move-length from small to large
N_size_converted_to_medium: Number of unsuccessful chemotatic move before converting the move-length from large to medium
N_size_converted_to_small: Number of unsuccessful chemotatic move before converting the move-length from medium to small
N_steps_to_die: Number of chemotatic moves of bacteria before maturing

Note here that a bacterium dies ONLY IF it has matured (*). This prevents the algorithm from removing bacteria which are actually lying in the attractive region of the global optimal but have not yet explored the area thoroughly enough to produce good solutions.

5.4 Simulation Studies

The Adaptive Bacterial Foraging Optimization Algorithm (BFOA) is applied for a set of scenarios to verify its efficiency. The scenarios are the same as those used in simulation studies with route-producing algorithms by Dynamic Programming (Chapter 3) and Ant Colony Optimization Algorithm (ACO – Chapter 4) so as to enable a cross checks of the algorithm validity and to reveal its advantages as well as disadvantages. The same OS model with that described Section 3.3.3 (Chapter 3) is also used in this chapter. The details of own ship and target positions, together with their speeds and courses over ground are therefore not listed here to avoid repetition. This information can be found in corresponding section in Chapter 3.

The Adaptive-BFOA is designed with a population of 100 bacteria (Ns). The swim-lengths are chosen to be 6 points, 3 points and 1 point respectively for d_{long}, d_{medium}, d_{small}. The neighborhood distance is 6 points. Number of bacteria dispersed after each generation is 10 (Nd), the number of bacteria reproduced/eliminated is 5 (Nr).

In all the scenarios, the solutions accompanying with the 70 healthiest bacteria of the population are shown for algorithms with 3, 6 and 9 generations (cycles) respectively. Then, the best route that the bacteria swarm has found for the algorithms with different number of cycles, starting from 3 cycles, are depicted, with the best route found by the algorithm of 9 cycles drawn in red.

5.4.1 Scenario 1

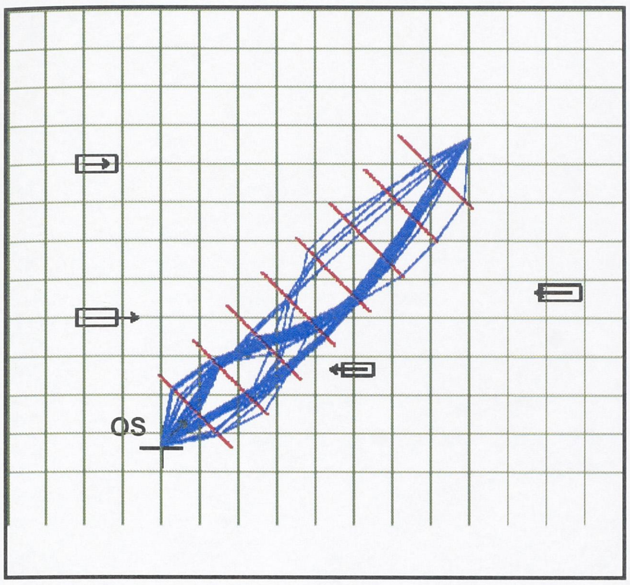


Fig. 5.11a Bacteria Positions after 3 Cycles

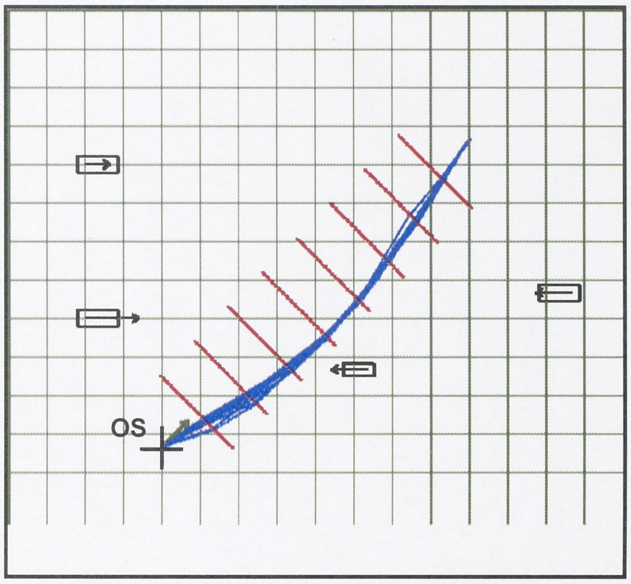


Fig. 5.11b Bacteria Positions after 6 Cycles

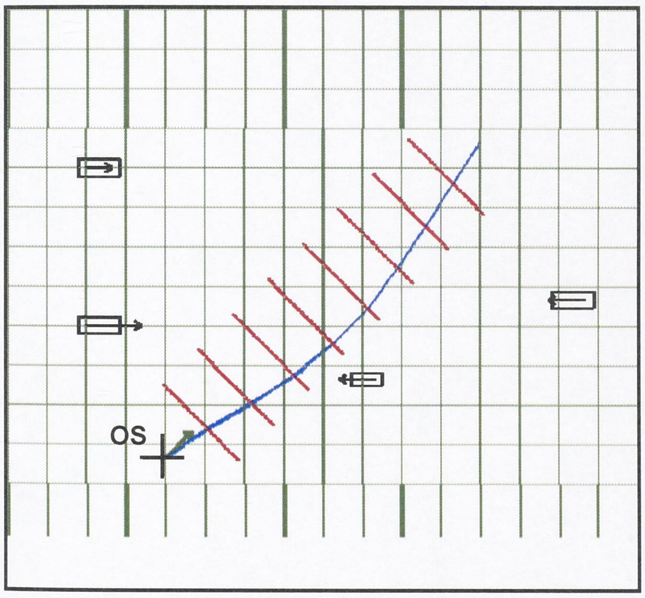


Fig. 5.11c Bacteria Positions after 9 Cycles

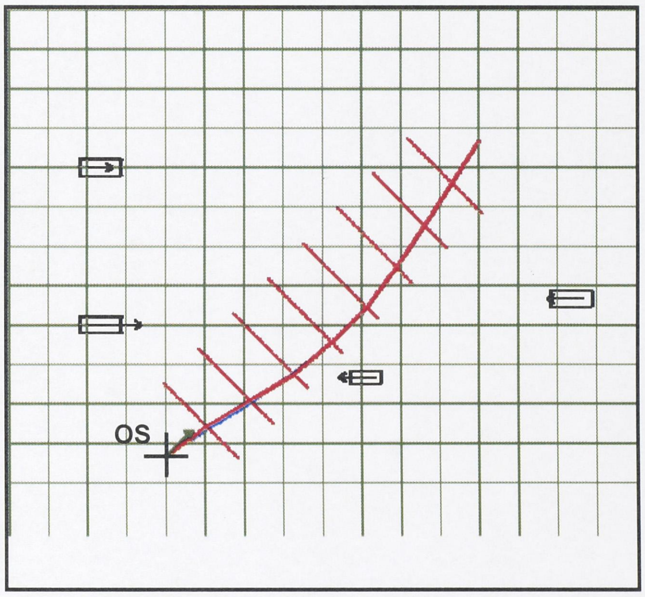


Fig. 5.11d Bac Best Positions in Each Cycles

Comments on the scenario:

- The search is initially scattered around the grid (Fig. 5.11a), thus the whole solution-space is explored thoroughly.
- The swarm converges promptly to a region of good collision-avoiding strategy.
- Regulations of the road are properly satisfied (turning to starboard, passing starboard crossing targets on OS port side).
- Best solutions found by algorithms with more than 3 generations are very close to the optimal solution (Fig. 5.11d). The optimal strategy is more appropriate than that produced in 3.5.1 and similar to that of 4.4.2.
- Convergence property is better than that of ACO algorithm.

(See 3.5.1 and 4.4.1 for result comparisons)

5.4.2 Scenario 2

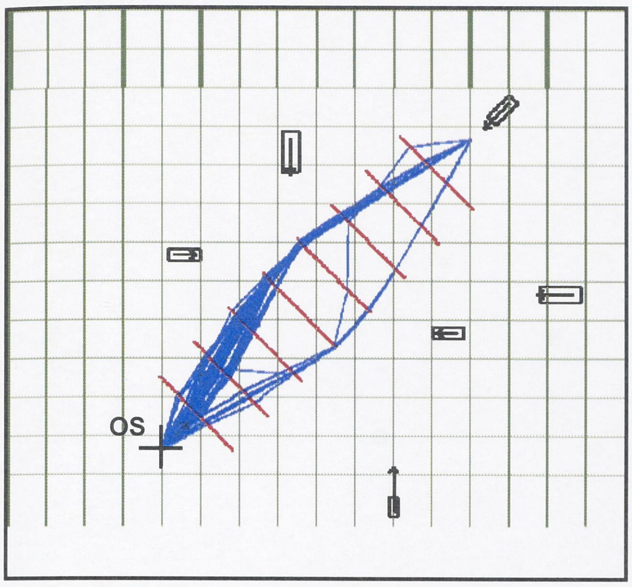


Fig. 5.12a Bacteria Positions after 3 Cycles

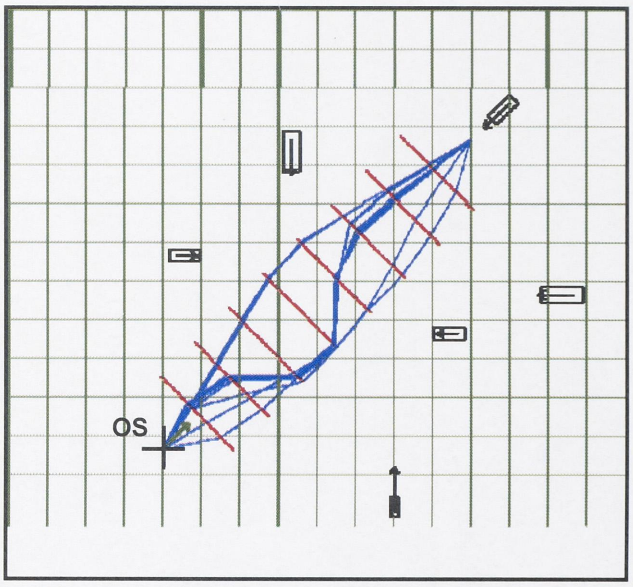


Fig. 5.12b Bacteria Positions after 6 Cycles

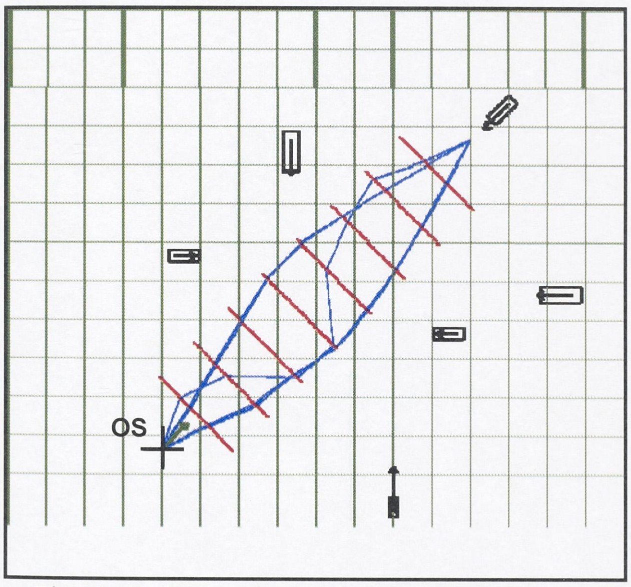


Fig. 5.12c Bacteria Positions after 9 Cycles

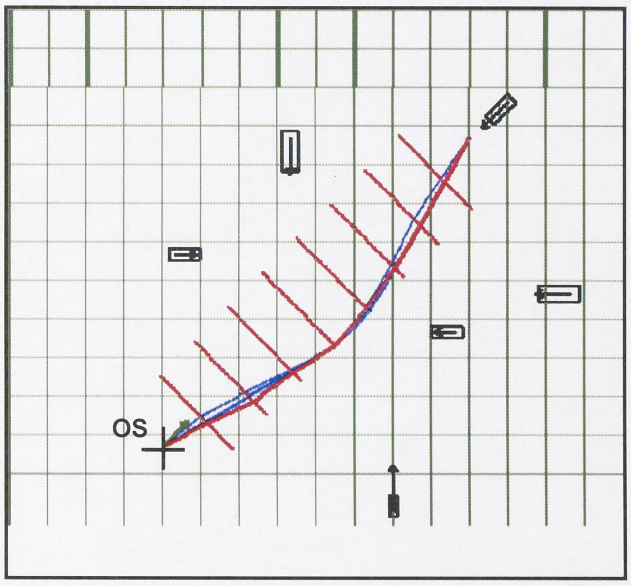


Fig. 5.12d Bac Best Positions in Each Cycles

Comments on the scenario:

- The search is initially scattered around the grid (Fig. 5.12a), then the whole solution-space is explored thoroughly.
- The swarm converges promptly to a region of good collision-avoiding strategy.
- Regulations of the road are properly satisfied (turning to starboard, passing starboard crossing targets on OS port, passing head-on target on port).
- Best solutions found by algorithms with more than 3 generations are very close to the optimal solution (Fig. 5.12d). The optimal strategy is more appropriate than that produced in 3.5.2 and similar to that of 4.4.2.
- Convergence property is better than that of ACO algorithm.

(See 3.5.2 and 4.4.2 for result comparisons)

5.4.3 Scenario 3

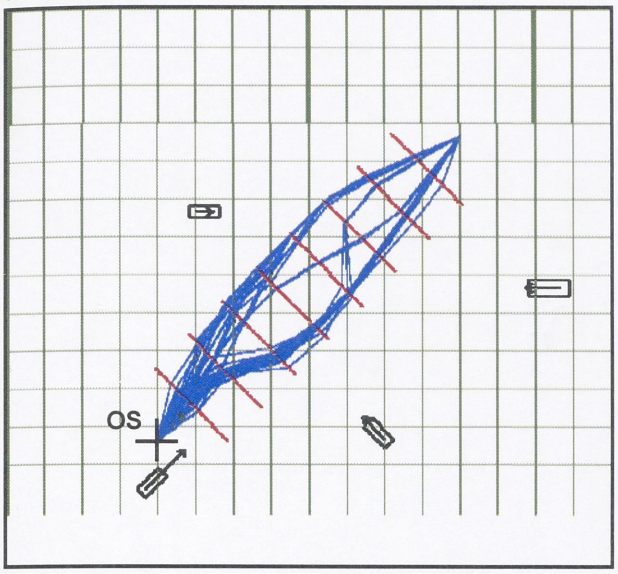


Fig. 5.13a Bacteria Positions after 3 Cycles

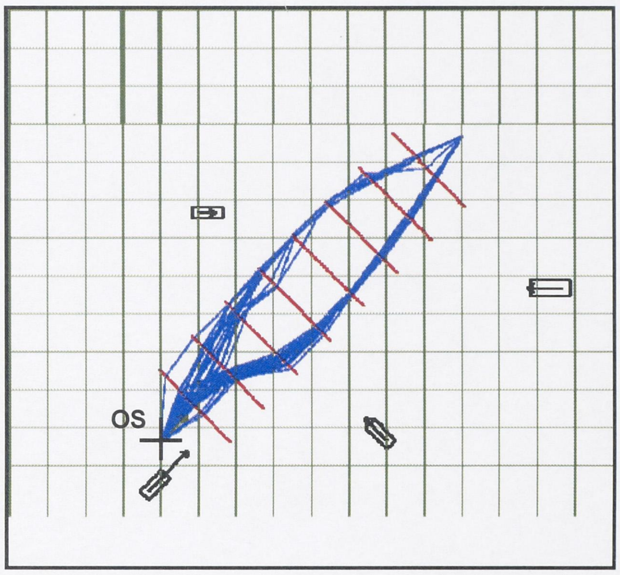


Fig. 5.13b Bacteria Positions after 6 Cycles

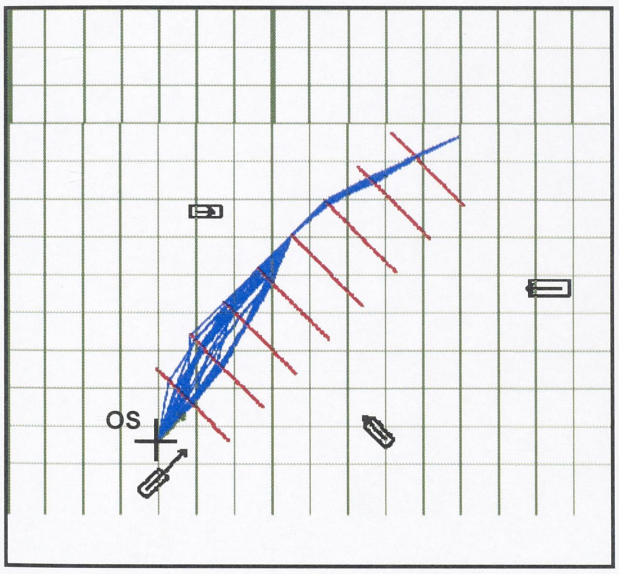


Fig. 5.13c Bacteria Positions after 9 Cycles

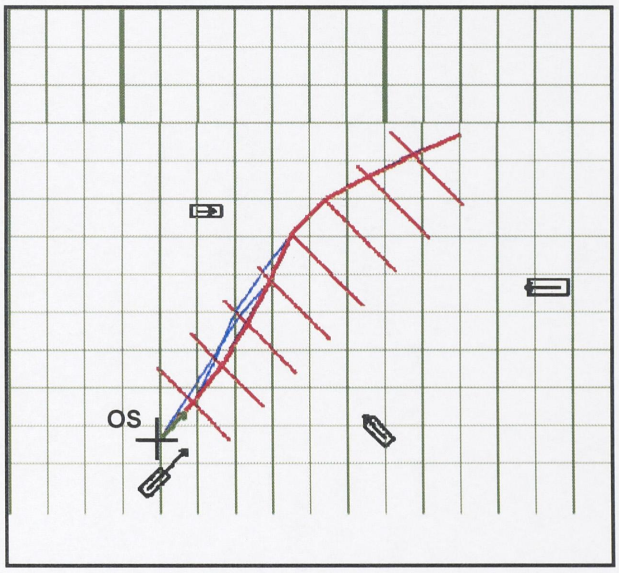


Fig. 5.13d Bac Best Positions in Each Cycles

Comments on the scenario:

- The search is initially scattered around the grid (Fig. 5.13a)

- The swarm first converges to 2 different regions as the route qualities relating to these 2 regions are more or less similar, and then focuses on the region of OS turning to port.

- Regulations of the road are satisfied to an acceptable extent. Although OS alters course to starboard and therefore passing starboard crossing targets on starboard, it does not cause any misunderstanding from TS side. Even if the targets actually judge the case wrongly, it is still safe as far as they follow rules of the road.

- Best solutions found by algorithms with more than 3 generations are very close to the optimal solution (Fig. 5.13d) and are similar to those produced by ACO (in 4.4.3).

- Convergence property is slightly better than that of ACO algorithm.

(See 3.5.3 and 4.4.3 for result comparisons)

5.4.4 Scenario 4

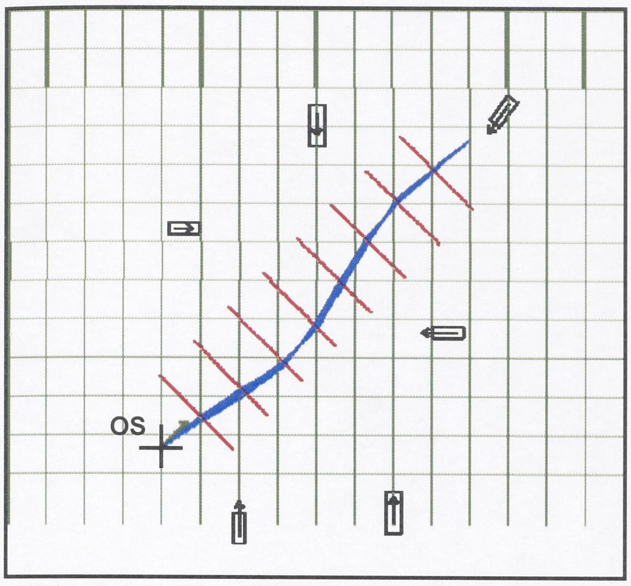


Fig. 5.14a Bacteria Positions after 3 Cycles

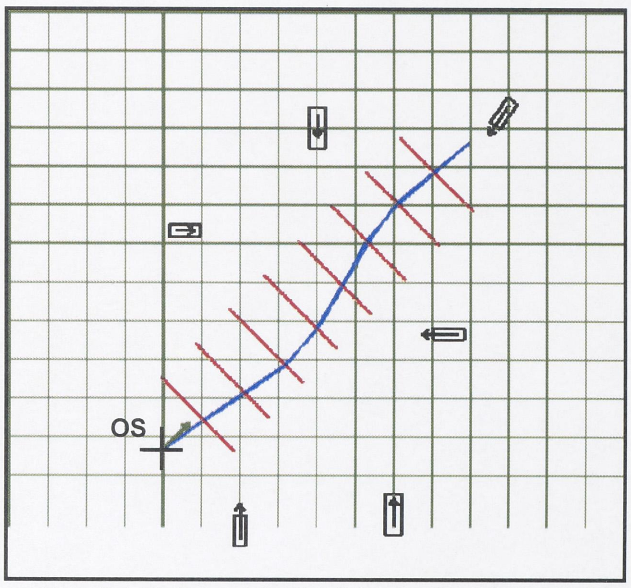


Fig. 5.14b Bacteria Positions after 6 Cycles

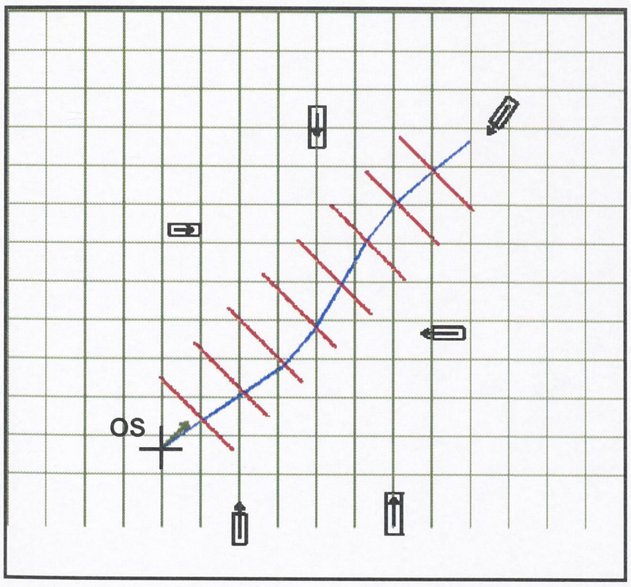


Fig. 5.14c Bacteria Positions after 9 Cycles

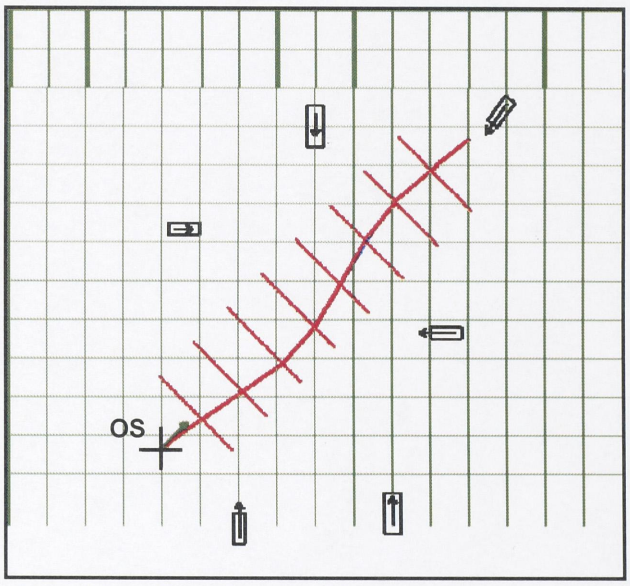


Fig. 5.14d Bac Best Positions in Each Cycles

Comments on the scenario:

- The swarm converges promptly to a region of good collision-avoiding strategy because the route quality corresponding to this region is far better than those of other regions.
- Regulations of the road are properly satisfied (turning to starboard, passing starboard crossing targets on OS port as far as the situation allows).
- Best solutions found by algorithms with more than 3 generations are very close to the optimal solution (Fig. 5.14d) and are similar to those produced by ACO (in 4.4.4)

(See 3.5.4 and 4.4.4 for result comparisons)

5.4.5 Scenario 5

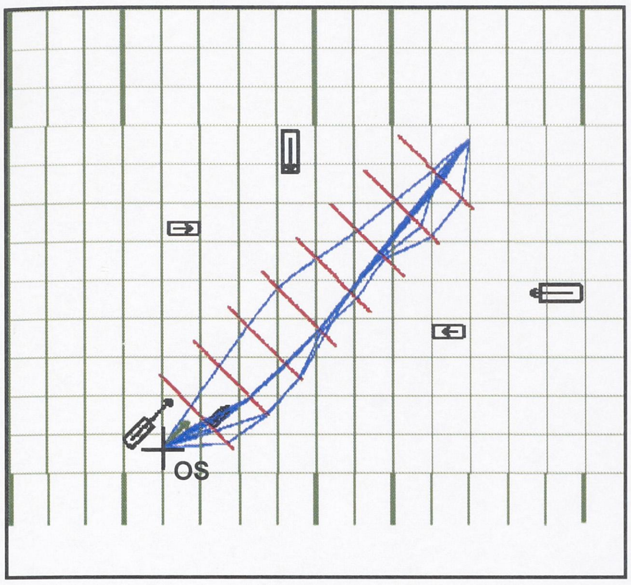


Fig. 5.15a Bacteria Positions after 3 Cycles

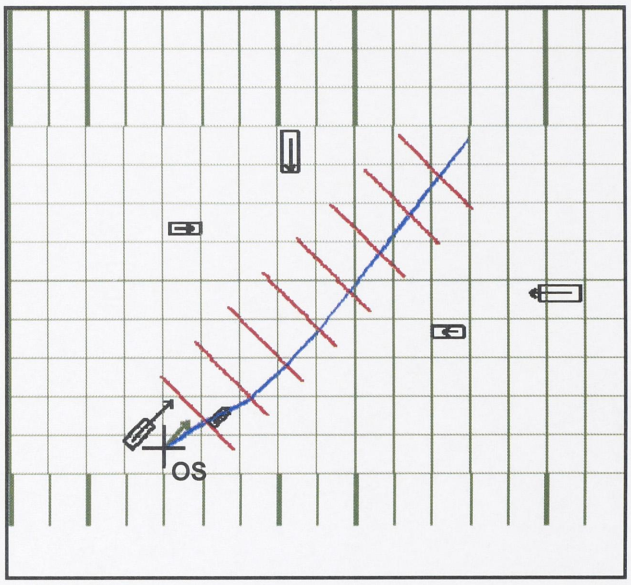


Fig. 5.15b Bacteria Positions after 6 Cycles

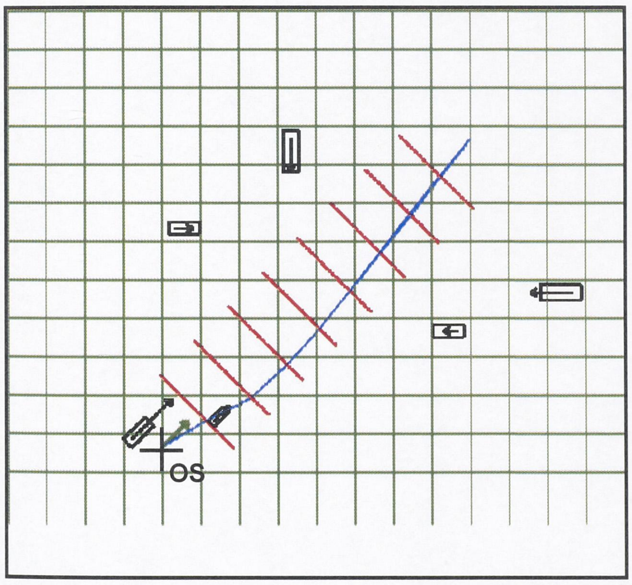


Fig. 5.15c Bacteria Positions after 9 Cycles

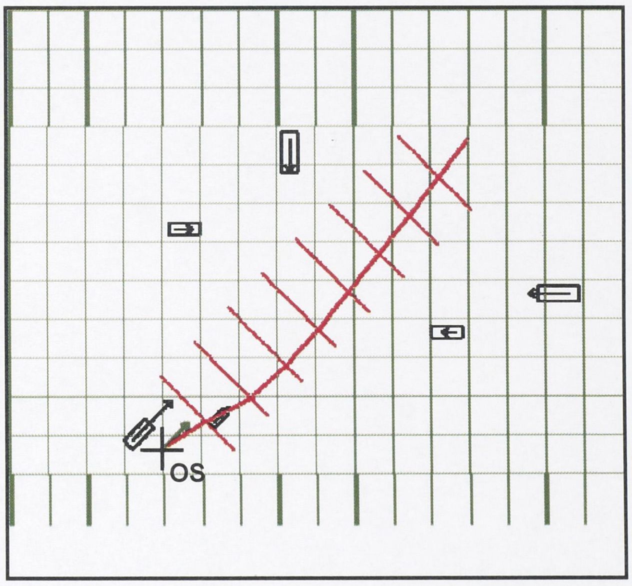


Fig. 5.15d Bac Best Positions in Each Cycles

Comments on the scenario:

- The search is initially scattered around the grid (Fig. 5.15a), then the whole solution-space is explored thoroughly.
- The swarm converges promptly to a region of good collision-avoiding strategy.
- Regulations of the road are perfectly satisfied.
- Best solutions found almost coincide with the optimal solution (Fig. 5.15d). The optimal strategy is more appropriate than that produced in 3.5.5 and is similar to that produced by ACO (in 4.4.5).
- Convergence property is slightly better than that of ACO algorithm.

(See 3.5.5 and 4.4.5 for result comparisons)

5.4.6 Scenario 6

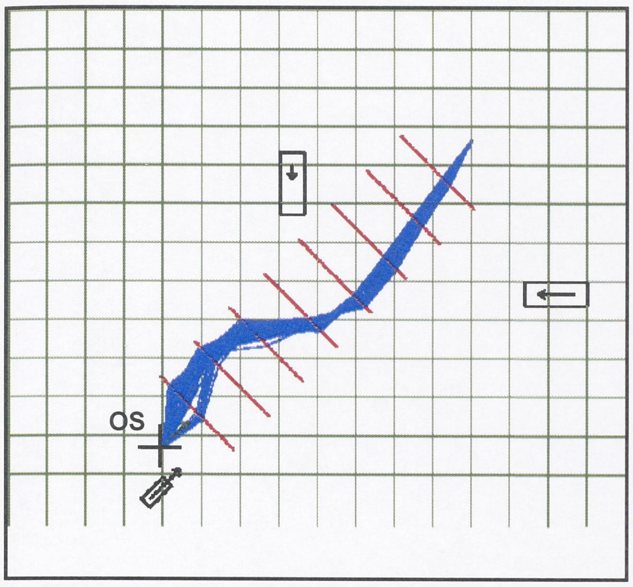


Fig. 5.16a Bacteria Positions after 3 Cycles

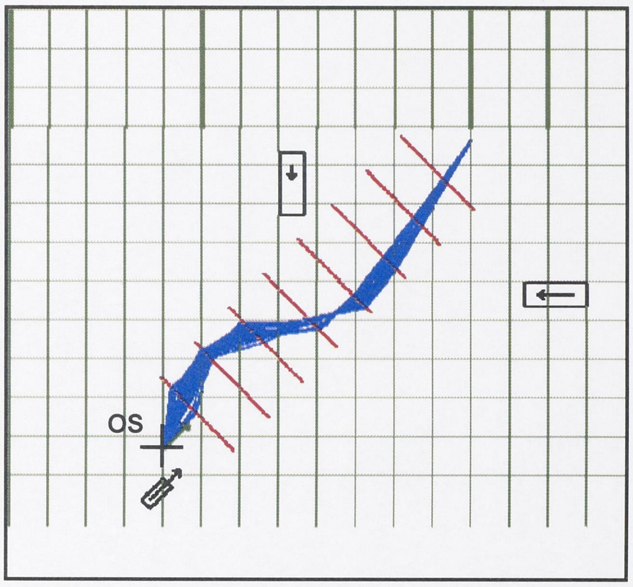


Fig. 5.16b Bacteria Positions after 6 Cycles

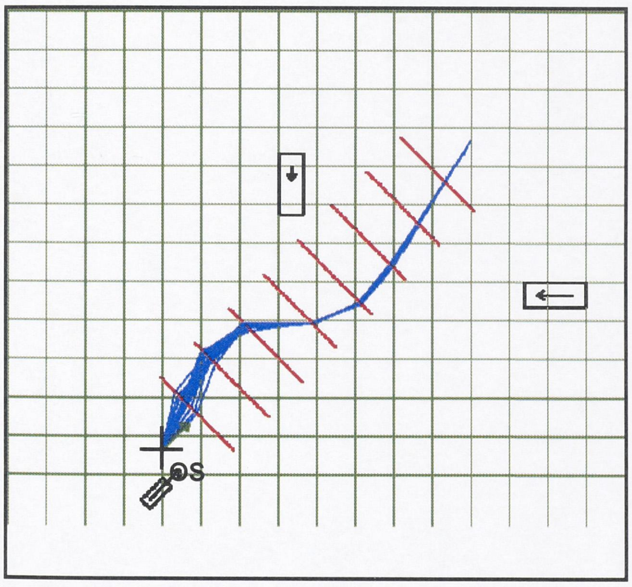


Fig. 5.16c Bacteria Positions after 9 Cycles

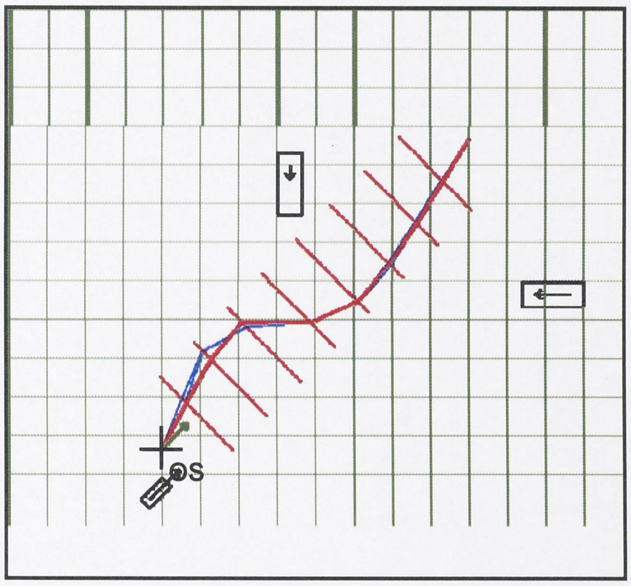


Fig. 5.16d Bac Best Positions in Each Cycles

Like that of section 4.4.6, the scenario serves to prove the resistibility and the robustness of the algorithm for different encountering cases at sea.

- The algorithm can still produce a collision-avoiding route while the algorithm using Dynamic Programming fails.
- The swarm converges promptly to a region of good collision-avoiding strategy.
- The strategy is acceptable, from the rule application point of view.
- Best solutions found by algorithms with more than 3 generations are very close to the optimal solution (Fig. 5.16d) and are similar to those produced by ACO (in 4.4.6).
- Convergence property is slightly better than that of ACO algorithm.

(See 3.5.6 and 4.4.6 for result comparisons).

5.6 Conclusions

In this chapter, an algorithm for producing the collision-avoiding route has been proposed. It is an Adaptive Bacteria Foraging Optimization Algorithm. The algorithm is based on assumption that TS do not change their courses and speeds, and the collision risk is avoided by OS action alone. Using Adaptive-BFOA, the solution is searched on a grid constructed from environmental constraints and other parameters decided by the ship officer.

To improve the efficiency of the searching algorithm, several modifications have been applied to classical algorithms suggested by other authors working on BFOA, taking into consideration the nature of marine traffic:

- A move-length adapting algorithm is suggested to improve the convergence speed and capacity of the bacteria to jump out of attractive region of local optimum.
- Bacteria are not replaced (die) before a certain number of chemotatic moves have been tried for them. The algorithm therefore does not miss a promising region.

The proposed BFOA is efficient for route-producing purpose. By the use of the route cost definition, the rules of the road can be properly taken into account, like that in the ACO algorithm. The solution is available within an acceptable time limit. Then, it is possible to apply in real time.

The Adaptive-BFOA overcomes the limitation of ACO algorithm in the aspect that the algorithm is much less sensitive to the choice of parameters. Simulations have also shown that the Adaptive-BFOA is very robust and reliable, given the diversity of marine traffic environment.

References

1. A. Biswas, S. Dasgupta et al., "Synergy of PSO and Bacterial Foraging Optimization-A Comparative Study on Numerical Benchmarks", available at "http://www.softcomputing.net/hais07_2.pdf"
2. C. Ying et al., "A Fast Bacterial Swarming Algorithm for high-dimensional function optimization", IEEE World Congress on Computational Intelligence, pp. 3135-3140, 2008
3. D. H. Kim, A. Abraham, and J. H. Cho, "A hybrid genetic algorithm and bacterial foraging approach for global optimization", Information Sciences, 177, pp.3918–3937, 2007
4. H. Chen, Y. Zhu, K. Hu, "Adaptive Bacterial Foraging Optimization", available at "http://www.hindawi.com/journals/aaa/2011/108269/"
5. H. Shen et al., "Bacterial Foraging Optimization Algorithm with Particle Swarm Optimization Strategy for Global Numerical Optimization", Available at "http://www.deepdyve.com/lp/association-for-computing-machinery/bacterial-foraging-optimization-algorithm-with-particle-swarm-FtBBdfh3qD"
6. K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control", IEEE Control Systems Magazine, Vol. 22, pp.52–67, 2002
7. M. Tripathy, S. Mishra, et al., "Transmission loss reduction based on FACTS and bacteria foraging algorithm", Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN '06), Vol. 4193 , pp. 222–231, 2006
8. S. Mishra, "A hybrid least square-fuzzy bacterial foraging strategy for harmonic

estimation," IEEE Transactions on Evolutionary Computation, Vol. 9, No. 1, pp. 61–73, 2005

9. S. Panikhom, N. Sarasiri, S. Sujitjorn, "Hybrid Bacterial Foraging and Tabu Search Optimization (BTSO) Algorithms for Lyapunov's Stability Analysis of Nonlinear Systems", International Journal of Mathematics and Computers in Simulation, Vol. 4, 2010
10. W. J. Tang, Q. H. Wu, and J. R. Saunders, "Bacterial foraging algorithm for dynamic environments", IEEE Congress on Evolutionary Computation, pp.1324–1330, 2006
11. Y. Liu and K. M. Passino, "Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviors", Journal of Optimization Theory and Applications, Vol. 115, pp. 603–628, 2002.

Chapter 6 Collision Avoiding Strategy in Critical Cases by Game Theory

6.1 Introduction

In chapter 3, 4 and 5, three different algorithms have been proposed to produce collision-avoiding route for the Own Ship (OS) with the assumption that target ships (TS) must not take adverse actions and there is ample time for OS to perform the necessary maneuver properly.

However, the marine traffic condition at sea is not always such cooperative. Due to various reasons, TS might misapprehend OS strategy and its counter-action is therefore inappropriate, i.e. the TS action might cause OS and TS to be in a highly risky encounter. The situation is extremely dangerous when the ships are in congested waters as the passing distances between them are usually small.

If TS intention is clear, the earlier route generating algorithms may be employed to provide the optimal collision-avoiding strategy for OS. However, the degree of danger increases sharply if the intention of the nearby target ships is doubtful or if these ships change courses abruptly and unexpectedly in a manner that may result in an imminent collision. Unfortunately, at this critical point the available route-producing algorithms do reveal their shortage. In this study, these cases are referred to as Critical Cases (see Fig. 6.1) for which the last-minute collision-avoiding strategies must be considered.

In critical cases, the following vitally important points must be noted:

- Firstly, the time allowed for judging the cases and taking actions is limited due to the small remaining distance between OS and TS.
- Secondly, the TS intention, i.e. TS intended course, is unclear.

Then, for the safety of the OS, the ship officer should prepare for the worst scenario, i.e. the scenario in which TS is trying to collide with the OS. This is exactly what happens in a pursuit-evasion game in which a player assuming the role of the pursuer exploits a strategy to catch another player, namely the evader. The evader, on the other hand, tries to avoid the collision (or the capture).

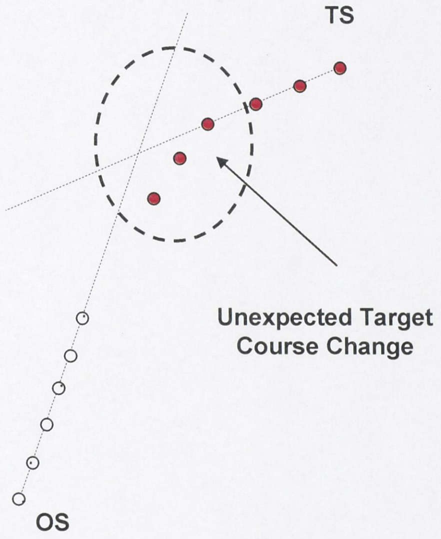


Fig. 6.1 Critical Case

Motivated by this perception, the chapter is a study aiming at providing the ship officer with a decision supporting means in the critical cases. This is a supplement to the route generating algorithms proposed in previous chapters in an effort to provide the OS officer with a recommended collision-avoiding decision in all navigational conditions from a departure point to the destination. The decision must ensure that the passage of the OS is safe and economic, and must be generated automatically.

The overall idea behind this collision-avoiding support system for critical case is described in Fig. 6.2. TS motions can be acquired by various observing aids, including the AIS receiver, Radar/ARPA or a camera system. It should be noted here that unlike the route generating algorithms for common encountering situations (at longer distances), camera images can be used in the algorithm of this collision-avoiding support for critical case. Because of the small remaining distances between vessels involving in the case, TS is in the effective range of the observation system based on cameras. Although the accuracy of the observation by camera is less than that provided by either AIS or Radar, camera may enable the detection and tracking of objects that do not possess an operating AIS receiver or that can not be seen on the Radar screen.

TS information and the OS planned route are used to assess the risk of collision. If collision risk arises due to a hasty and unexpected maneuver of the TS; and furthermore, the collision is imminent, the "critical case collision avoidance" module will be activated to generate a collision-avoiding strategy for the OS. The output of this collision-avoidance block is a sequence of rudder commands to navigate OS out of dangers while still, as far as possible, maintain a small deviation from the pre-planned route.

On the other hand, if the situation is NOT critical (i.e. collision is not imminent and/or there is no uncertainty in target maneuver), the normal route-generating algorithms may be activated to produce a collision-avoiding path for the OS. Then, the block for route-tracking control is to realize this path by sending the control system of the ship a suitable rudder command sequence.

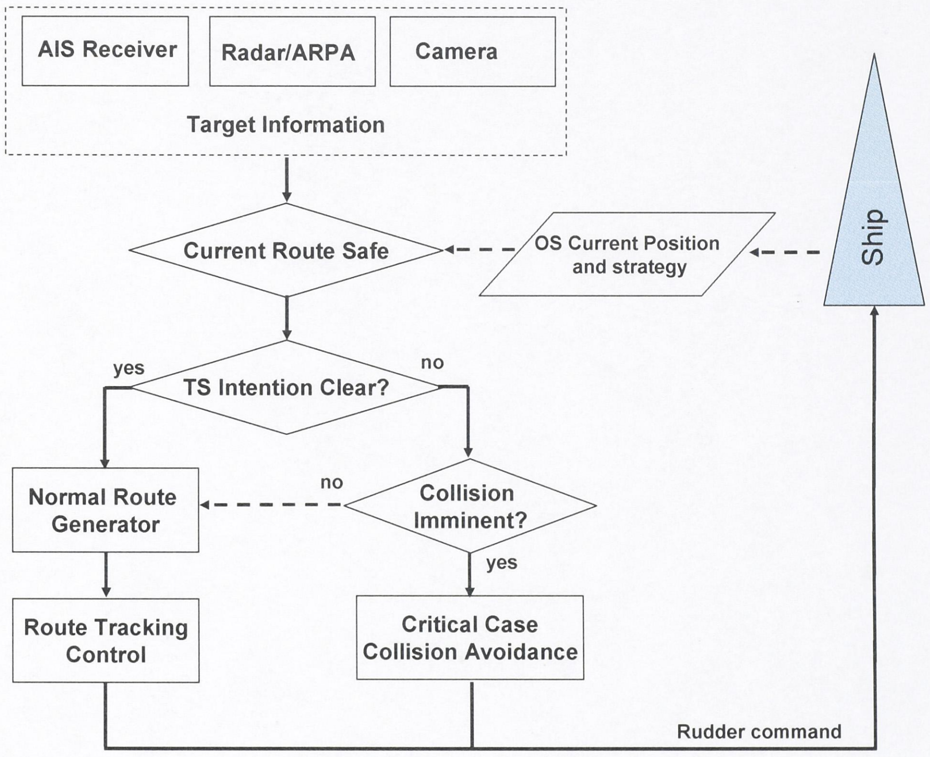


Fig. 6.2 System Structure Over View

Applying the Game Theory, the generation of collision-avoiding strategy for OS in critical cases is achieved by solving the following 2 tasks:

- Modeling the collision-avoiding problem as a pursuit-evasion game, with each ship having its individual goal (payoff function). The payoffs are defined so as to express the preferences of the ships in such cases.

- Solving the above game for the optimal OS strategy. As the problem is highly nonlinear, an adaptive Bacterial Foraging Optimization Algorithm (BFOA) is applied to seek the solution of the game approximately, i.e. to search for a sequence of rudder commands for the OS that maximizes its payoff function.

The chapter will be arranged as followings: Section 6.2 gives a brief introduction of the Game Theory in general and the pursuit-evasion game in particular. The modeling of the collision-avoiding problem in critical cases as a pursuit-evasion game is the subject of Section 6.3. In Section 6.4, an Adaptive-BFOA will be proposed to solve the optimization problem arising in the game. Then, some simulation results will be presented in Section 6.5. The conclusions of the chapter will be summarized in Section 6.6.

It should be noted that the algorithm is constructed with the assumption that the OS will not change its engine state while taking maneuver. The rudder is therefore the ONLY actuator.

6.2 Game Theory and the Pursuit-Evasion Game

6.2.1 Definitions and Classifications

Game theory is the branch of decision theory that concerns with the interdependent decisions. It typically involves several participants, each with her individual objective. The objectives of the participants may be conflicting and therefore the scenario is competitive.

In everyday life sense, a game is commonly defined as a competitive activity in which the players compete with each other according to a set of rules.

Then, the competitive situations mentioned above can be naturally considered games and the participants are referred to as the players. In fact, it is not even necessary that the situation is competitive; the game theory hence simply deals with any scenarios in which the action of a player depends on actions of other players. Game theory finds its application in a wide range of real life problems such as firms competing for business, political candidates competing for votes, animals fighting over the prey etc.

Like other sciences, game theory consists of a collection of models where a model is simply an abstraction used to explain one's observations and experiences about the nature or social phenomena. A component of many models in game theory is the theory of rational choice which states that [4]:

The action chosen by a decision-maker is at least as good, according to her preferences, as every other available action.

where an action is a possible decision the decision-maker can take, given her current state and the situation she is facing.

The 2 general types of game concerned in our study are the Strategic Game and the Extensive Game (or Extensive Form Game).

According to Osborne [4], a Strategic Game is a model of interacting decision makers. The decision makers are the players of the game. Each player has a set of possible actions. The model realizes the interaction between the players by allowing each player to be affected not only by her own action but by all other players' actions as well. Then, a strategic game consists of:

- A set of players
- For each player, a set of actions
- For each player, preferences over the set of action profiles

Unlike the strategic games, an Extensive Game enables the explicit representation of other aspects of the competing process such as the sequence of possible actions of the players, the players' choices at decision-points, the information a player has about other players at these points, etc. In its simplest instance, an extensive game can be seen as the spread of a strategic game with time. The following terms are commonly used in an extensive game:

- A move: a player's action at certain decision-point
- A strategy: a sequence of moves taken by a player

- The optimal strategy of a player: the strategy that produces the best outcome for the player.

However, the separating boundary between the above 2 game types is unclear. It has been stated that every extensive game has its equivalent strategic game form.

Basing on their rules, games can be classified into one of the following 2 kinds:

- Sequential Game: In a sequential game, the players make moves alternatively and therefore one player may have advantage of the knowledge (or information), i.e. her i^{th} move can be decided with the information of the i^{th} move of the other player that has been done.
- Simultaneous Game: In a simultaneous game, players make their moves simultaneously and thus are fair in terms of situation perception.

Basing on the relation between players' payoffs and their contributions to the total outcome of the game, games can be classified as either cooperative or non-cooperative.

- Cooperative game: A game is cooperative if the players in the game can be divided into different groups and players in each group share cooperative behaviors i.e. they form binding commitments. This can only be realized if the players have persistent communication links with each other and they are obligated to behave as promised. The game is then the competition between the player groups.
- Non-cooperative game: In a non-cooperative game, the players make decision independently.

Being a subject of intensive researches in the last several decades, huge volume of theories and models has been developed for the games as well as their applications. Interested readers may refer to textbooks and papers specifically published for the purpose [5][6].

6.2.2 Nash Equilibrium

One of the most important contributions to the development of the game theory is that proposed by J.F. Nash in his PhD thesis and therefore named after him: the Nash Equilibrium.

Assume that a strategic game is defined with n players. Let p be an action profile in which the action of each player j is p_j. Let p'_j be any action of player j that may be equal to p_j or different from it. Denote by (p'_j, p_{-j}) the action profile in which all players k except j choose their action p_k as specified in the profile p while j chooses p'_j instead of p_j. Then:

The action profile p is a Nash Equilibrium in a strategic game with ordinal preferences (denoted by p*) if for every player j and every action p'_j of j, p* is at least good according to preferences of player j as the action profile (p'_j, p^*_{-j}).

Equivalently, for every player j,

$$U_j(p^*) \geq U_j(p'_j, p^*_{-j}) \quad for\ j = 1\ to\ n$$

$$where\ U_j\ is\ a\ payoff\ function\ representing\ player\ j's\ preferences$$

Then, a Nash Equilibrium correspond to a steady state of players' actions in which no players can get benefit from changing his action while other players keep their actions unchanged.

To be applicable to other game models, several refinements have been proposed for the Nash Equilibrium. Among them, the Sub-game Perfection Nash Equilibrium is commonly studied in extensive games. Details of this can be found in suitable textbooks on the extensive game and its representation in tree form (or graph decision nodes).

6.2.3 Pursuit-Evasion Game

Among the most popular games, the pursuit-evasion game has been the subject of many different researches. It involves 2 or more players of conflicting interests in which:

- A player assumes the role of an evader trying to escape from the pursuer.
- Other players are the pursuers who try to capture the evader as quickly as they can.

Examples of this type of games are the Homicidal Chauffeur game, the Dubin Cars game, the Lion and Man game, etc. [1][2][5]. To elucidate the terms and definitions in previous sections, a simple pursuit-evasion game is raised and modeled here, both as a strategic game and as an extensive game.

A. Pursuit-Evasion Strategic Game

The game is defined with 2 players: a pursuer and an evader.

The Pursuer :

– *Initial position* (X_0^p, Y_0^p)

– *Move step size* V^p

– *Set of actions* $(MoveWithCourse(\psi^p))$

– *Payoff function* $U^p(\psi^p, \psi^e) = -d$

The Evader :

– *Initial position* (X_0^e, Y_0^e)

– *Move step size* V^e

– *Set of actions* $(MoveWithCourse(\psi^e))$

– *Payoff function* $U^e(\psi^p, \psi^e) = d$

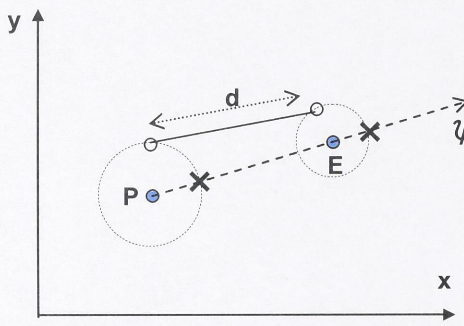


Fig. 6.3 Pursuit-Evasion Strategic Game

It is clearly seen from the definition that the players' payoff functions are expressions of their preferences. The pursuer wishes to get close to the evader the most (maximize his payoff -d) while the evader tries to get away from the pursuer as far as possible (maximize his payoff d).

An action profile is a couple of directions in which the players move: (ψ^p, ψ^e)

A Nash Equilibrium of the game is the action profile: (ψ^0, ψ^0) i.e. $(\psi^p = \psi^0, \psi^e = \psi^0)$ where ψ^0 is as shown in Fig. 6.3. The clue of this equilibrium is obvious: If the pursuer chooses ψ^0, the only action the evader can take to achieve the best payoff is to move at direction ψ^0. Conversely, if the evader chooses ψ^0, the best action of the pursuer is also ψ^0.

B. Pursuit-Evasion Extensive Game

A finite extensive pursuit-evasion game is formed by spreading (or spanning) the game A above in time. In Fig. 6.4, the game is played in 3 steps. Still, the pursuer's target is to approach the evader as quickly as possible while the later aims at escaping from the former.

A pursuer's strategy is the sequence $(\psi_1^p, \psi_2^p, \psi_3^p)$ of moving directions at each time step.

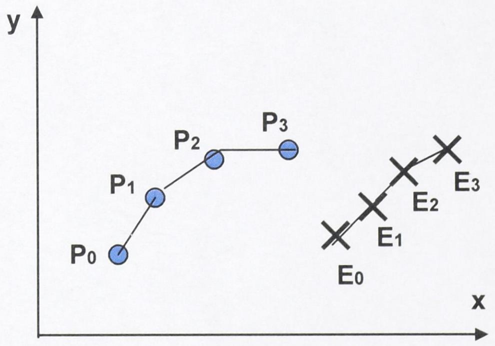


Fig. 6.4 Pursuit-Evasion Extensive Game

Similarly, a strategy of the evader is the moving direction sequence ($\psi_1^e, \psi_2^e, \psi_3^e$)

The game is called sequential game, with the pursuer having information advantage if the order of moves of the 2 players is as followings:

$\psi_1^e \to \psi_1^p \to \psi_2^e \to \psi_2^p \to \psi_3^e \to \psi_3^p$, and an equilibrium is the set of moves in the direction ψ^0 for both players ($\psi_1^e = \psi_1^p = \psi_2^e = \psi_2^p = \psi_3^e = \psi_3^p = \psi^0$).

6.3 Collision-Avoiding Problem as a Pursuit-Evasion Game

In critical cases, the TS intention is unclear and TS is approaching the OS on a collision course. Additionally, the time allowed for assessing the situation and maneuvering is terribly short.

Thence, to ensure the OS safety, the OS officer has to prepare for the worst case: TS is trying to collide with OS. The nature of the encounter is therefore similar to the behaviors of players in a pursuit-evasion game. From this perception, in this study, the collision-avoiding problem in critical cases is modeled as a pursuit-evasion game in which:

- TS: the pursuer, having the sole aim of catching the OS (getting closer to the OS) as quick as possible.

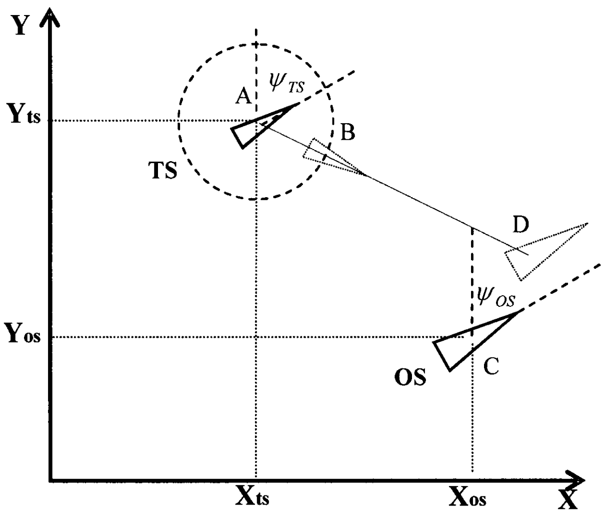


Fig. 6.5 TS and OS Motion Coordinates

- OS: the evader which reasons between the following 2 targets simultaneously: avoiding the capture by TS (or the collision) and retaining itself in the proximity of its original path, i.e. maintaining a small deviation from its planned route.

6.3.1 Own Ship and Target Ship Motion Models

The OS and TS maneuvering models are used to determine the possible actions they, as players, can take at any moment, i.e. the positions they can assume in the following interval, given the current states.

6.3.1.1 Own Ship Model (Evader)

For the OS, the maneuverability should be expressed accurately and simply so as to reduce calculation time while still ensuring the calculation accuracy. Then, in this study, a simple first derivative model of motion (i.e. 1st order Nomoto's T-K model for the yaw rate and a model of similar form for sway velocity) is used to present the OS maneuverability:

$$\dot{v} = -\frac{1}{T_v}v + \frac{K_v}{T_v}\delta$$

$$\dot{r} = -\frac{1}{T_\psi}r + \frac{K_\psi}{T_\psi}\delta$$

$$\text{where}: \qquad (6.1)$$

$$r, v, \delta : \textit{yaw rate, sway velocity, rudder angle}$$

$$T_{v,\psi}, K_{v,\psi} : \textit{coefficien ts of the model}$$

The model coefficients can be determined from full- scale experiments of the OS. In practice, it is generally difficult to determine coefficients Tv and Kv of sway velocity dynamics. Then, this part might be omitted from the ship model (6.1) ($v = \dot{v} = 0$) at the price that there will be an additional mismatch between the calculated and the actual ship tracks while maneuvering the OS with the calculated strategy.

Then, in discrete form, dynamics of the evader can be approximated as followings (see Fig. 6.5):

$$\begin{aligned}
&\delta(k+1) = C_{OS} \\
&\Delta v(k+1) = -\frac{1}{T_v} v(k) + \frac{K_v}{T_v}\delta(k) = a_v v(k) + b_v \delta(k) \\
&\Delta r(k+1) = -\frac{1}{T_\psi} r(k) + \frac{K_\psi}{T_\psi}\delta(k) = a_r v(k) + b_r \delta(k) \\
&u(k+1) = u(k) \\
&v(k+1) = v(k) + \Delta v(k+1) \\
&r(k+1) = r(k) + \Delta r(k+1) \\
&\psi(k+1) = \psi(k) + r(k+1) \\
&X_{OS}(k+1) = X_{OS}(k) + u(k+1) Sin(\psi(k))\,\Delta t + v(k+1) Sin(\psi(k) + \pi/2)\Delta t \\
&Y_{OS}(k+1) = Y_{OS}(k) + u(k+1) Cos(\psi(k))\Delta t + v(k+1) Cos(\psi(k) + \pi/2)\Delta t \\
&\quad where: \\
&\qquad \psi : OS\ heading \\
&\qquad u : OS\ forward\ speed \\
&\qquad X_{OS}, Y_{OS} : OS\ position\ \ coordinates \\
&\qquad C_{OS} : OS\ Rudder\ Command\ Input \\
&\qquad \Delta t:\ time\ interval
\end{aligned} \tag{6.2}$$

The coefficients for OS model which is used throughout this chapter have been calculated for different rudder angles, by Least Square Method. The result is as shown in the following table.

Rudder Angle	a_v	b_v	a_r	b_r
5^o	-0.0176099	-0.10115194	-0.02558338	0.0031756
10^o	-0.02333969	-0.08794908	-0.0335784	0.0029240
15^o	-0.0345990	-0.0593977	-0.04927108	0.0023004

Table 6.1 Coefficients of the OS First Derivatives Model

6.3.1.2 Target Ship Model (Pursuer)

In the critical cases, the TS motion is unpredictable. By navigation aids, just the target position at each sampling interval can be observed with certainty. TS velocity can be determined from long-period observation of TS motion and is assumed to be unchanged during the collision-avoiding procedure. Expectably, TS will, at least, not increase its speed even when there is human error in maneuvering.

As there is very little or no information about TS maneuverability, to ensure the safety it should be assumed that the TS could change its course as quickly as it desires. It is a reasonable assumption because in the critical cases, the target is changing its course fast.

Then, the pursuer's dynamics in discrete form can be approximated by the following set of formulae (see Fig. 6.5).

$$
\begin{aligned}
&\psi_{TS}(k+1) = C_{TS} \\
&V_{TS}(k+1) = V_{TS}(k) \\
&X_{TS}(k+1) = X_{TS}(k) + V_{TS} Sin(\psi_{TS}(k))\Delta t \\
&Y_{TS}(k+1) = Y_{TS}(k) + V_{TS} Cos(\psi_{TS}(k))\Delta t \\
&where: \\
&\quad \psi_{TS} : TS\ heading \\
&\quad V_{TS} : TS\ speed \\
&\quad X_{TS}, Y_{TS} : TS\ position\ coordinates \\
&\quad C_{TS} : Course\ (heading\ command)\ Input
\end{aligned}
\tag{6.3}
$$

6.3.2 Player Payoffs and Equivalent Games

6.3.2.1 Collision Avoidance as a Strategic Game

The collision-avoiding maneuver in critical cases should be calculated for a span of time ahead to ensure the safety of the OS for, at least, this interval before the wrong action could be detected and corrected accordingly. In this study, it is assumed that the game is played in Tf [sec], starting from the time of calculation.

The starting position of OS and TS is defined as OS_s and TS_s respectively. After Tf[sec], OS reaches position OS_f and TS assumes position TS_f.

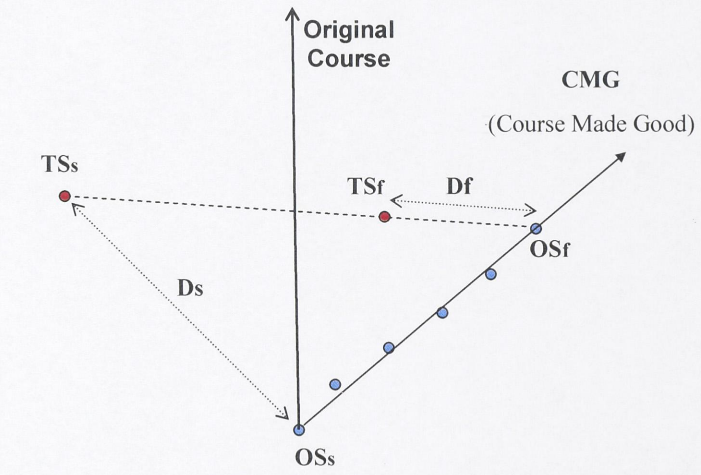


Fig. 6.6 OS Strategy and Countering TS Optimal Strategy

Modeling as a strategic game, the positions of OS and TS in the middle of the progress are blind to the other. Then, the action of TS is a sequence of control command to navigate the TS to TS_f and that of OS is a sequence of rudder-commands to bring OS to OS_f.

As mentioned earlier, it has been assumed that TS has advantage of information i.e. TS takes action after OS has already finished its move. Thence, the optimal or the most preferred action of TS, which has a single aim of catching OS as quickly as possible, is to take the course C_{TS} to assume the position TS_f where:

$$C_{TS} = direction(TS_s, OS_f)$$
$$\left|TS_s TS_f\right| = V_{TS} \times Tf \qquad (6.4)$$

On the other hand, the objective of the OS is to avoid collision and to try to keep its deviation from the original course as small as possible. Hence, the objective-function of the OS in this game is chosen as the followings:

$$Q = Cos(CMG) - k_0 \times \left(\frac{Ds - Df}{\Delta D}\right)^{\alpha} - k_1 \times \sum_{j=1}^{n} \frac{C_{OSj}}{C_{OS\max}} \qquad (6.5)$$

$$C_{OS} = \{C_{OS1}, C_{OS2}, \ldots\ldots C_{OSn}\}\ \textit{a sequence of control rudder for OS, each of length}\ \Delta t_{OS}$$

Df *final distance to TS*

Ds *start distance to TS*

$k_0, k_1, \alpha, \Delta D$: *coefficients to be chosen*

CMG: OS Course Made Good

The quality function (6.5) contains 3 components. The first one expresses the deviation of OS from its original path. The second component represents the distance reduction after players have taken their actions. The last component takes into consideration the control effort of OS. The coefficient k_1 should be small.

Then, the strategic game is defined with the following components:

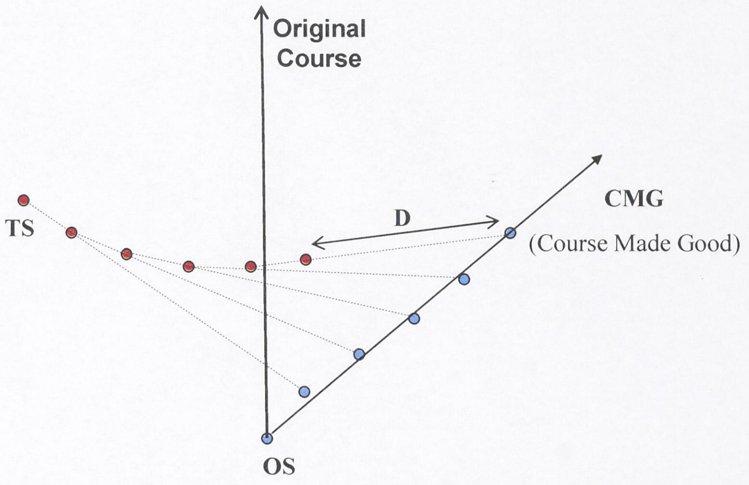


Fig. 6.7 OS Strategy and Countering TS Optimal Strategy

- Players: Evader (OS) and Pursuer (TS).
- Evader's actions: A sequence of rudder-commands in Tf[sec], using (6.2).
- Pursuer's actions: A sequence of course-commands in Tf[sec], using (6.3).
- Evader's payoff function: defined by (6.4)
- Pursuer's payoff function: -Df

The strategy of OS for collision-avoiding is determined by solving the following optimization problem (6.6)

$$Q_0 = Max_{(C_{OS})}(Optimal\ Target\ Strategy_{(C_{TS})}(Q)) \qquad (6.6)$$

where the optimal strategy of TS is defined in (6.4).

The solution, if properly found, is a Nash Equilibrium of the game as neither TS nor OS wish to deviate from this profile if the other follows the action assigned for it in the profile.

6.3.2.2 Collision Avoidance as a Finite Extensive Game

Unlike the above strategic game, in this finite extensive game, the calculating time Tf is split into finite time-intervals and the players make their moves alternatively in each time interval. In that manner, TS can renew its strategy when the OS motion information, i.e. its position, is updated and vice verse.

Then, the optimal strategy for TS at each sampling point is as illustrated in Fig. 6.5: it is to choose a course to reach point B, knowing that OS is to reach point D. Our game should be considered as a sequential game in which TS the information advantage. The course-command for TS at each sampling point, therefore, is

$$\begin{aligned} &C_{TS} = direction(AD) \\ &AB = V_{TS} \times \Delta t \qquad\qquad (6.7) \\ &where\ \Delta t\ is\ the\ sampling\ interval \end{aligned}$$

The optimal strategy of TS, the optimal sequence of courses, for a random strategy of OS is illustrated in Fig. 6.7.

In this game, the objective of the OS is to avoid collision and to keep its deviation from the original course as small as possible. Hence, the objective function of the OS can be chosen as the following:

$$Q = Cos(CMG) - k_0 \times \sum_{i=1}^{m} \left(\frac{D(0) - D(i)}{i \times \Delta D_{\max}} \right)^{\alpha} - k_1 \times \sum_{j=1}^{n} \frac{C_{OSj}}{C_{OS\max}} \qquad (6.8)$$

$$\begin{aligned} &C_{OS} = \{C_{OS1}, C_{OS2}, \ldots\ldots C_{OSn}\}\ a\ sequence\ of\ control\ rudder\ for \\ &\qquad\qquad\qquad Own\ Ship,\ each\ of\ length\ \Delta t_{OS} \\ &C_{TS} = \{C_{TS1}, C_{TS2}, \ldots\ldots C_{TSm}\}\ a\ sequence\ of\ control\ heading\ for \\ &\qquad\qquad\qquad target,\ each\ of\ length\ \Delta t_{TS} \\ &\Delta t_{TS} \times m = \Delta t_{OS} \times n = T_f \\ &T_f\ \ final\ avoidance\ time \\ &D(i)\ \ distance\ to\ TS\ at\ sample\ i \\ &k_0, k_1, \alpha, \Delta D_{\max} : coefficients\ to\ be\ chosen \\ &CMG: OS\ Course\ Made\ Good \end{aligned}$$

The problem to be solved in this game is then to find a sequence of OS rudder-angles to maximize its payoff, given TS strategy as described in (6.7).

$$Q_0 = Max_{(C_{OS})}(Optimal\ Target\ Strategy_{(C_{TS})}(Q)) \qquad (6.9)$$

6.4 Game Solution by Adaptive-BFOA

The optimization problems in the previous games are highly nonlinear and therefore almost impossible to be solved analytically. Fortunately, an Adaptive-BFOA similar to that described in Chapter 5 can be employed to seek an approximation of the optimal.

Due to the similarity of the 2 algorithms, principles of BFOA are not discussed in details in this chapter. Instead a brief description of the algorithm for generating initial solutions and that for imitating the bacteria foraging behavior, the chemotactic move, is mentioned here. Then, a pseudo-code algorithm similar to that in Chapter 5 is introduced for the whole procedure of seeking the optimal solution.

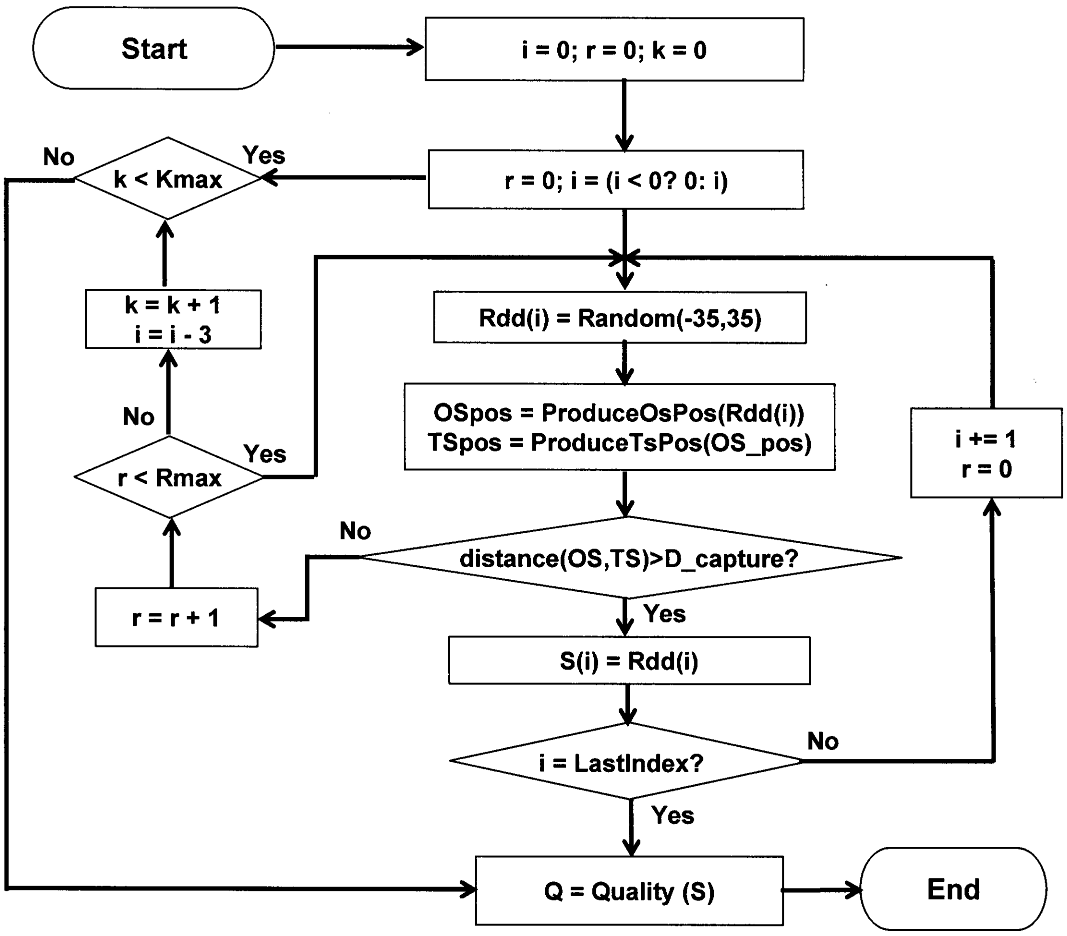
6.4.1 Bacteria Position Initialization (Solution Initialization)

The position of a bacterium in the population or a solution equivalently is a sequence of rudder-commands for the OS, each with a length of Δt_{OS} [sec]

$$S = [Rdd(1), Rdd(2), \ldots, Rdd(i-1), NA, \ldots, NA]$$

$$LastSucessIndex(S) = i - 1$$

In this algorithm, a capture-distance D_capture is defined for the game. If the distance between OS and TS falls below D_capture, the collision is deemed to have occurred. The total length in time of the rudder-command sequence (Tf) is chosen to be 180[sec]. However, it is not always possible for OS to avoid collision till Tf. Then, accompanying with each solution is its LastSuccessIndex which is the largest sampling index at which the distance between OS and TS is still above D_capture. After this index, the rudder-command is marked with NA (No Answer).



$$S = [Rdd(1), Rdd(2), \ldots, Rdd(i-1), NA, \ldots, NA]$$

Fig.6.8 Bacteria Position Initializing Procedure

A component of the solution is a rudder-angle in the range [-35°, 35°]. At each sampling interval, this angle is randomly set and the OS position at the following sampling time is calculated accordingly (ProduceOsPosition). TS position is then produced and used to check whether the capture has occurred or not. If OS has been actually captured, another rudder-angle is chosen instead. Otherwise, the search-procedure continues at the next sampling interval.

The initializing process is repeated at each sampling interval for a maximum of Rmax times. If it still fails, the process is shifted back 3 steps (intervals) and resumed from there.

Later, the payoff function for OS strategy is calculated by (6.5) and (6.8) depending on the type of game employed.

6.4.2 Bacteria Chemotaxis Procedure

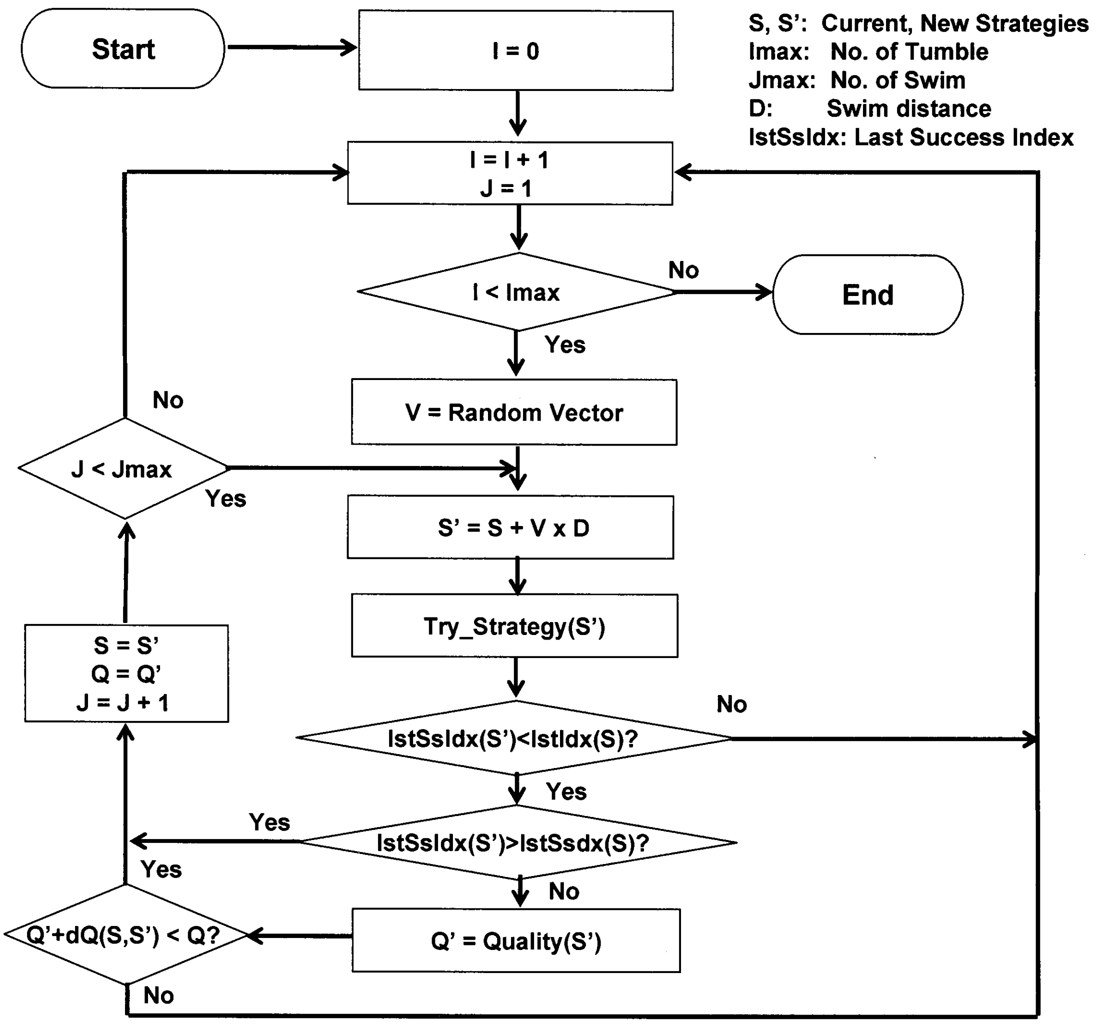


Fig.6.9 Bacteria Chemotaxis Procedure

The process is a local-search mechanism by imitating the behavior of bacteria in foraging. Like that in Chapter 5, it is the combination of the tumbles denoted by vector V and swims with swim-length D. It is the process of looking for an improvement to the current solution.

As shown in Fig. 6.9, a try is successful if at least one of the following two conditions is met:

- Last Success Index of a trial (new) solution is larger than that of the current solution.

- Last Success Indices of the 2 solutions are the same and the payoff of the trial solution, after adjusting due to bacteria communication effects, is better than the payoff of the current solution.

If a move is successful, the current solution is replaced with the trial solution.

The chemotactic process of a bacterium and its application in improving OS strategy are illustrated in Fig. 6.10 to Fig. 6.12. In this example, original positions of OS and TS are:

OS: (X[m], Y[m], ϕ [deg], Sog[m/s]) = (0, 0, 0, 7.0)

TS: (X[m], Y[m], Sog[m/s]) = (850, 850, 6.5)

Starting with an initial OS strategy called 1st track, with a rudder-command sequence shown in Fig. 6.10, the OS track obtained by this strategy is as shown in Fig. 6.12 (OS 1st track). Given OS strategy, TS countering strategy is TS 1st track in Fig. 6.12. If the capture-distance is set to be 700 [m], TS is to catch OS in 70[sec].

Improving the initial strategy through chemotaxis, by varying several rudder-commands, a new strategy (2nd track) can be obtained. Using this improved strategy, OS is safe for 160[sec], even if TS choose the most dangerous move available.

The variation of distance between OS and TS is shown in Fig. 6.11. It is easily seen here that the limit distance of 700[m] is maintained up to 160 seconds for the modified control sequence. This is obviously a much better strategy for the collision-avoiding purpose of OS as formulated in section 6.3.

OS and TS tracks for the initial and improved strategies are also illustrated in Fig. 6.12. Sampling interval is set to 10 [sec]. It should be kept in mind that TS speed is slightly slower than OS speed in this example.

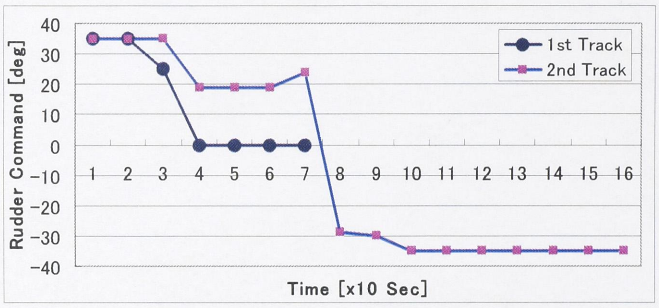


Fig.6.10 Initial and Improved OS Strategies

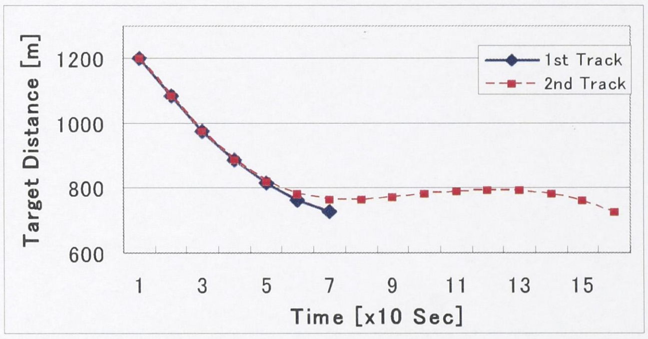


Fig.6.11 Distance between TS and OS

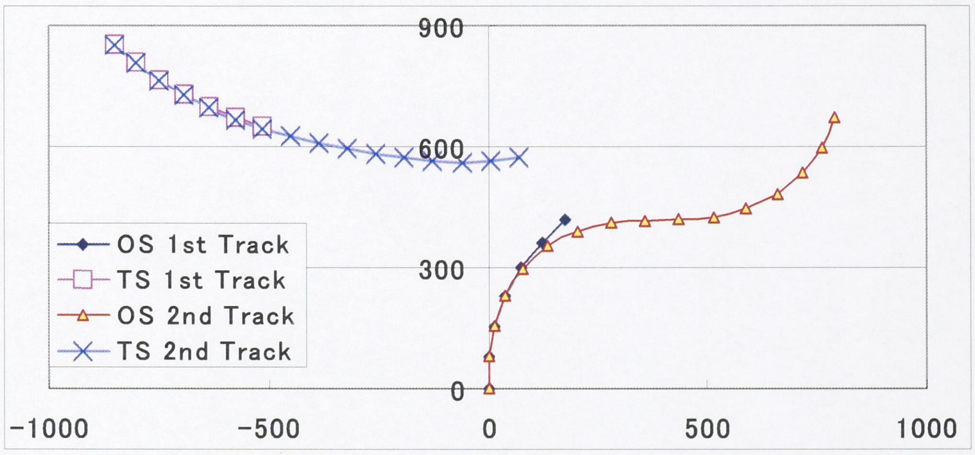


Fig.6.12 TS and OS Initial and Renewed Tracks

6.4.3 Overall Adaptive-BFOA for Producing Collision-Avoiding Strategy

The overall strategy-generating algorithm can be illustrated by the following pseudo-code:

```
A. Initialization
   Initialize_Rudder_command_Space(Rdd_max, Rdd_min, Rdd_step);

   For bac = 1 to Ns
       Initialize_Bacterium(B(bac));
   Next bac

B. Evolution

   For cycles = 1 to N_cyc
       For bac = 1 to Ns
           For chemo = 1 to Nc
               Perform_Chemotatic_Move(B(bac));
           Next chemo

           If(Number_of_ Unsuccessful_ Move > N_size_converted_to_[large/medium/small]) then
               Convert_move_length_from_[large/medium/small]_to_[medium/small/large]();
           End if
       Next bac

       Sort_the_Bacteria_and_Payoff_Arrays_by_Descending_Payoff(B(Ns), Q(Ns));

       For die_no = 1 to Nr
          If ( Chemotatic_Move_of_Bacterium_Count(B(die_no))> N_steps_to_die ) then
             Kill_bacterium(B(die_no));
             B(die_no)= Reprocude_Bacterium(B(Ns - die_no));
          End if
       Next die_no

       For disperse_no = 1 to Nd
          rand = produce_random_interger()
          Initialize_Bacterium(B(rand));
       Next disperse_no
   Next cycles

C. Termination
   Sort_the_Bacteria_and_Payoff_Arrays_by_Descending_Payoff(B(Ns), Q(Ns));
   Return B(1);
```

Where the variables and designing parameters are defined as:

Rdd_max: Maxium rudder angle
Rdd_mint: Minimum rudder angle

Rdd_step: Rudder angle step
N_cyc: Number of cycles in the algorithm i.e. number of generations of the bacteria population.
Ns: Number of bacteria in the population
B(Ns): Bacteria population (bacteria set)
Q(Ns): Cost array of the bacteria
Nr: Number of bacteria died/reproduced in a cycle
Nd: Number of bacteria eliminated/dispersed in a cycle
Nc: Number of chemotatic steps of a bacterium in a cycle
N_size_converted_to_large: Number of unsuccessful chemotatic move before converting the move-length from small to large
N_size_converted_to_medium: Number of unsuccessful chemotatic move before converting the move-length from large to medium
N_size_converted_to_small: Number of unsuccessful chemotatic move before converting the move-length from medium to small
N_steps_to_die: Number of chemotatic moves of bacteria before maturing

6.5 Simulation Studies

To verify the viability of the method, computer simulations have been carried out using an MMG model for the OS for different encounters i.e. various TS position and velocity. Later on, simulations are conducted with real tracks of ships navigating at sea. The data is acquired by Radar at 1-minute interval. The MMG model of OS is that of a domestic container ship of 100m long. Coefficients of the model are shown in Table 6.1.However, only the T-K model of the yaw-rate is used for the simulations.

TS, without recognizing the OS existence, changed courses dangerously. As a result, the OS is in a very serious encounter-case because the distance between the 2 ships is small. It is even more risky as the officer in charge of the OS is not certain about TS intention. Then, the strategy-generating algorithm based on game theory is applied to find a collision-avoiding strategy for OS.

6.5.1 Simulation Studies – Evading Strategy for Different Encounter Cases

The aim of this part is to test the strategies for OS to avoid collision in various critical encounter cases.

As mentioned earlier in section 6.3, the problem can be modeled as either a Strategic Game (SGame) or an Extensive Game (EGame). For each type of game, a strategy is produced accordingly. A population of 70 bacteria is used for solution searching. The population undergoes 7 generations before the optimal strategy is extracted. The capture-distance (D_capture) is set to 600[m] and calculation time (Tf) is 180[sec].

Scenario 1 (Fig. 6.13):

OS: (X[m], Y[m], ϕ [deg], Sog[m/s]) = (0, 0, 0, 7.0)
TS: (X[m], Y[m], Sog[m/s]) = (-850, 850, 6.5)

Scenario 2 (Fig. 6.14):

OS: (X[m], Y[m], ϕ [deg], Sog[m/s]) = (0, 0, 0, 7.0)
TS: (X[m], Y[m], Sog[m/s]) = (-850, 850, 7.5)

Scenario 3 (Fig. 6.15):

OS: (X[m], Y[m], ϕ [deg], Sog[m/s]) = (0, 0, 0, 7.0)

TS: (X[m], Y[m], Sog[m/s]) = (-1089.5, 508.0, 7.5)

Scenario 4 (Fig. 6.16):

OS: (X[m], Y[m], ϕ [deg], Sog[m/s]) = (0, 0, 0, 7.0)

TS: (X[m], Y[m], Sog[m/s]) = (-1089.5, 508.0, 8.5)

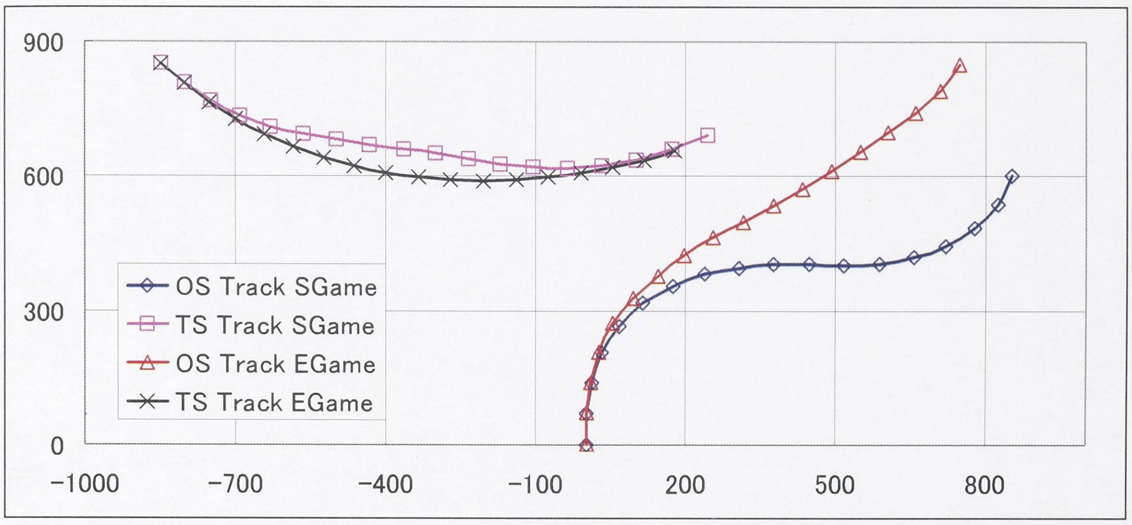


Fig.6.13 Evading Strategy of OS in Different Games, TS_Speed = 6.5m/s

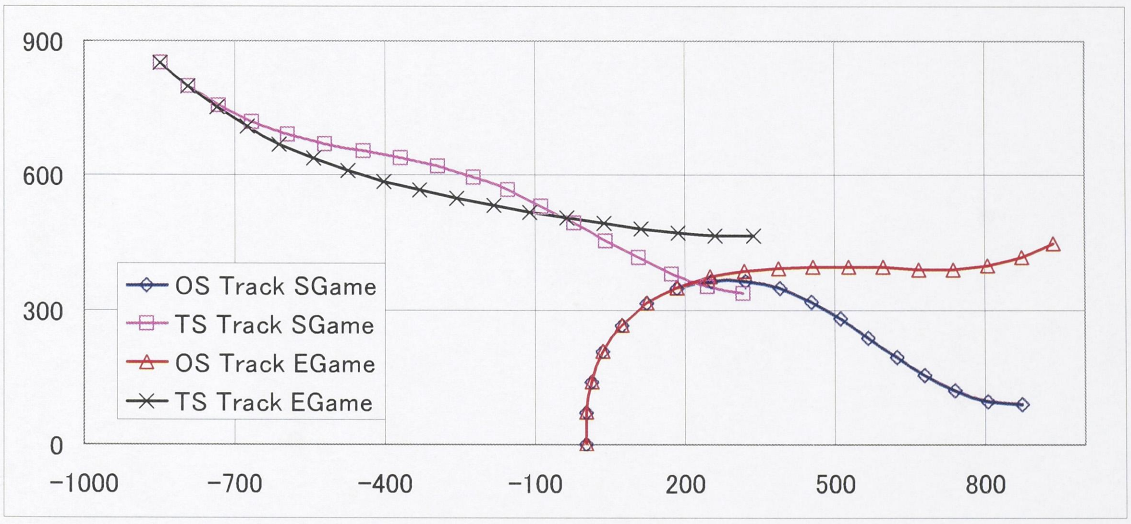


Fig.6.14 Evading Strategy of OS in Different Games, TS_Speed = 7.5m/s

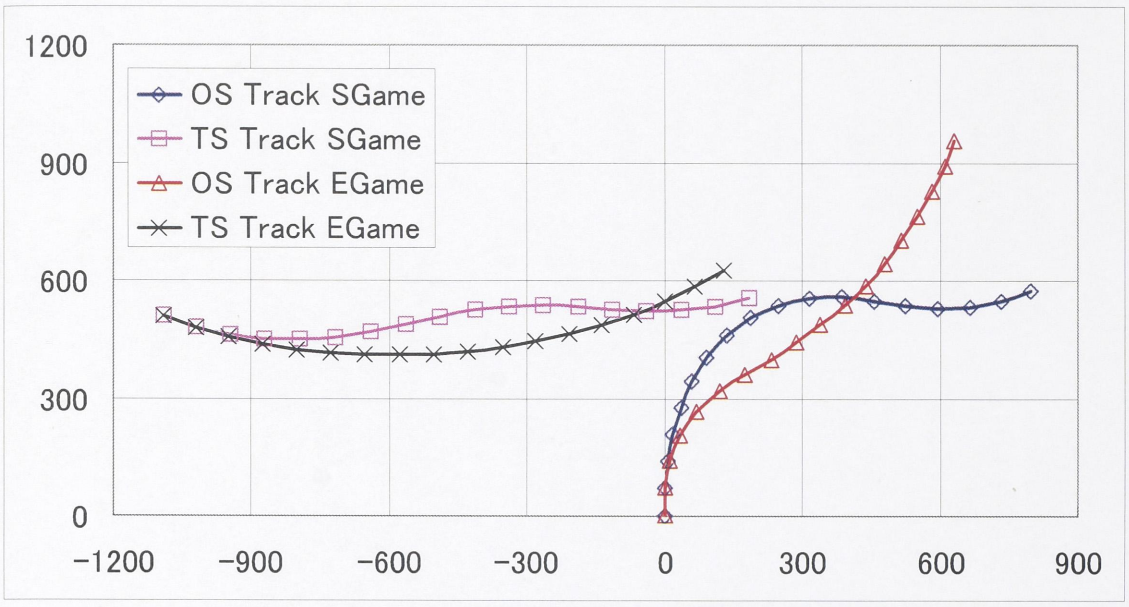


Fig.6.15 Evading Strategy of OS in Different Games, TS_Speed = 7.5m/s

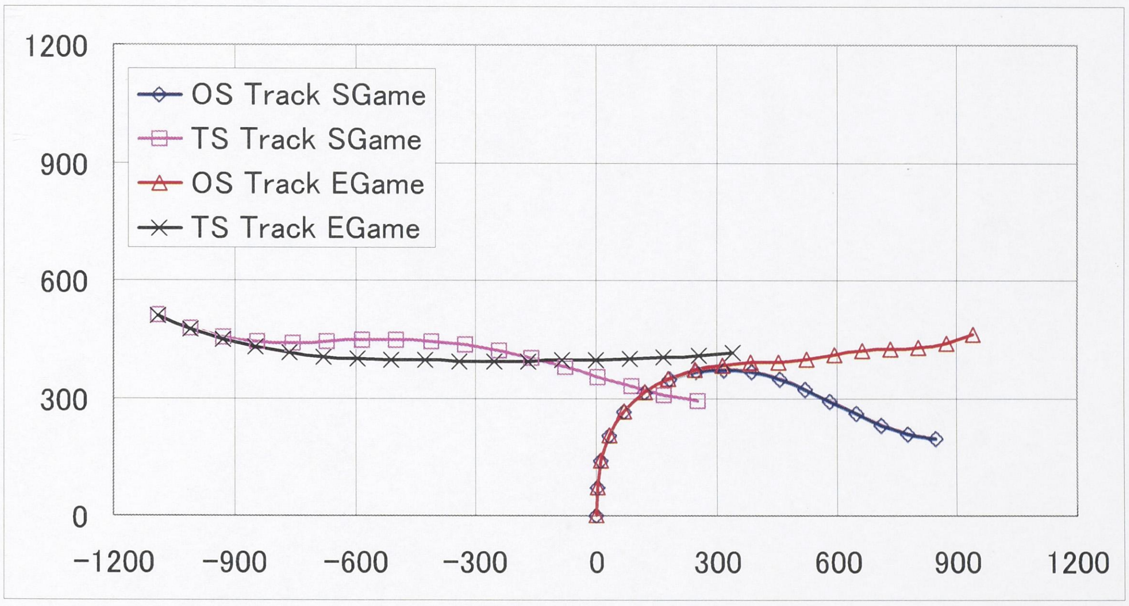


Fig.6.16 Evading Strategy of OS in Different Games, TS_Speed = 8.5m/s

Discussion:

- In all simulation scenarios, the collision-avoiding strategy for the OS is produced timely and it is getting along well with the experienced seamanship.
- Collision-avoiding strategy produced by EGame allows the OS to keep closer to the original route than that produced by SGame.
- If SGame is used, the strategy is safe and is the optimal strategy for the whole calculation time span Tf, given the payoff functions used. Then, the OS officer does not have to watch TS

motion closely for this whole period except that he wants to update the strategy, knowing that TS actually did not choose its optimal pursuing strategy.

- If EGame is applied, the OS officer has to watch TS motion continuously. It should be noted that the solution is NOT equilibrium of the problem. The OS might be in danger if the TS does not strictly follow its assumed strategy while the former commits to the strategy produced for it by solving the game. An example of this limitation is a scenario in which:

TS keeps on navigating on the course connecting its initial position with the OS final position. The final distance between the ships might therefore falls below the capture distance. This distance is always less than the calculated final distance at least.

Fortunately, as the TS motion is observed continuously, the collision-avoiding strategy for the OS can be updated accordingly whenever a deviation of the TS from its assumed track is detected. Another positive point of the strategy is that the distance between the ships at each sampling point is not less than that calculated. Hence, the update strategy is normally not worse than the initially produced strategy.

6.5.2 Simulation Studies – Recursively Updated Strategies against Radar Targets

In this subsection, the algorithm for producing strategies is tested with real TS motion-tracks. In fact, the TS is not pursuing OS. However, it is assumed that the TS does not detect the OS existence and therefore navigates unmannerly. The simulation is to show how the OS strategy is updated, given the changes in the TS strategy. The Extensive Game form is used, with calculation time Tf set to be 3[min]. Sampling interval is 1[min], equaling to TS position updating interval.

Scenario 1:

In this scenario, the TS is moving faster than the OS. When distance between them is just over 2000m, the TS alters course dangerously to starboard. To avoid collision and to keep small deviation from its original route simultaneously, the OS turns hard to starboard. The process of strategy-generating is repeated whenever the TS position is updated. Then, a new collision-avoiding strategy is applied. The whole collision-avoiding route of the OS and the TS track are shown in Fig. 6.17a.

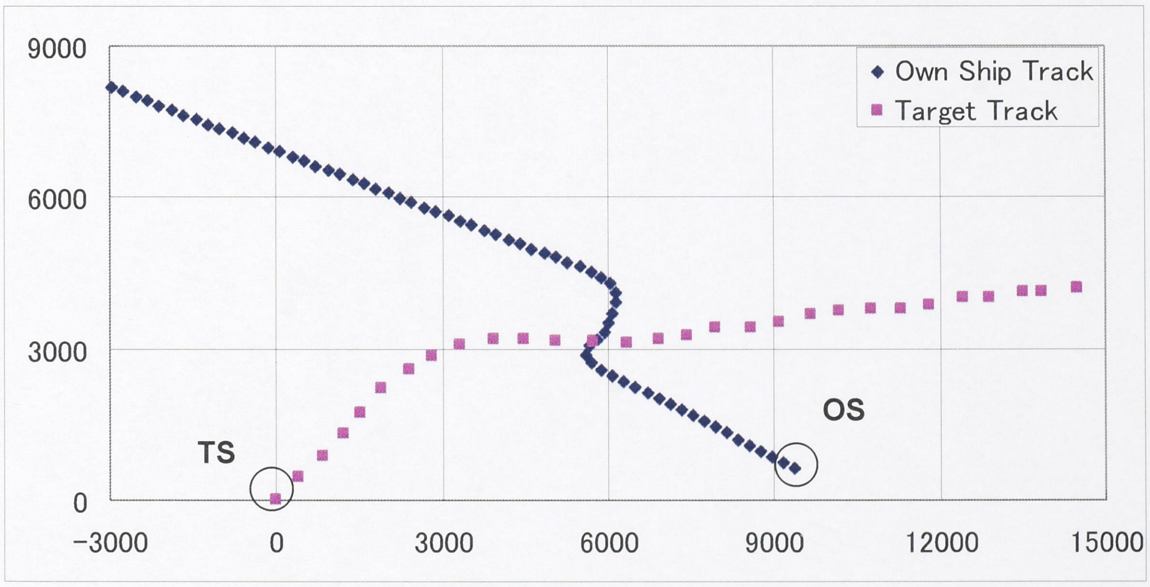


Fig.6.17a Actual Tracks of TS and OS

The OS track calculated at each sampling point as well as its actual track is drawn in Fig. 6.17b. The strategy updating process enables the elimination of the accumulated deviation of the OS position from its calculated track due to the modeling error, i.e. the mismatch between T-K model and MMG model.

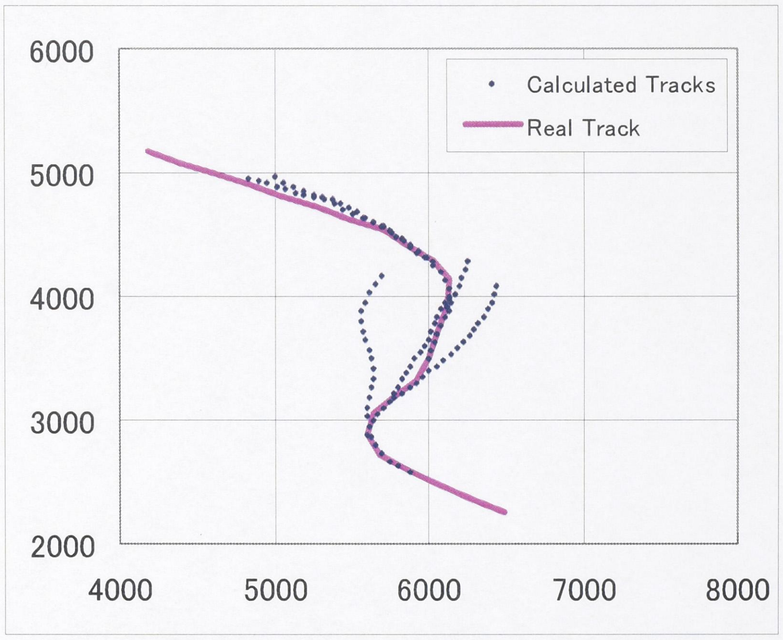


Fig.6.17b Calculated and Actual Tracks

In Fig. 6.17c, distance between the ships during the encounter is shown. The distance at the closest point of approaching is around 500 [m].

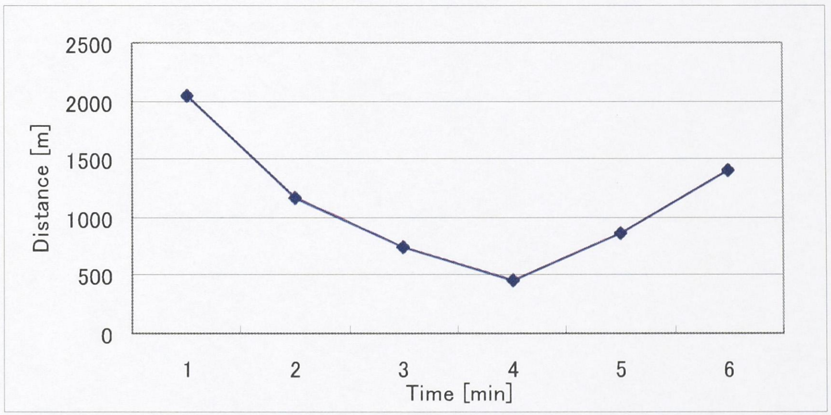


Fig.6.17c Distance to TS during Maneuver

Scenario 2:

In this scenario, the TS alters course dangerously to port when the distance to the OS is around 2300 [m]. The generated strategy for the OS is to turn hard to port. The whole collision-avoiding process between the OS and the TS is shown in Fig. 6.18a.

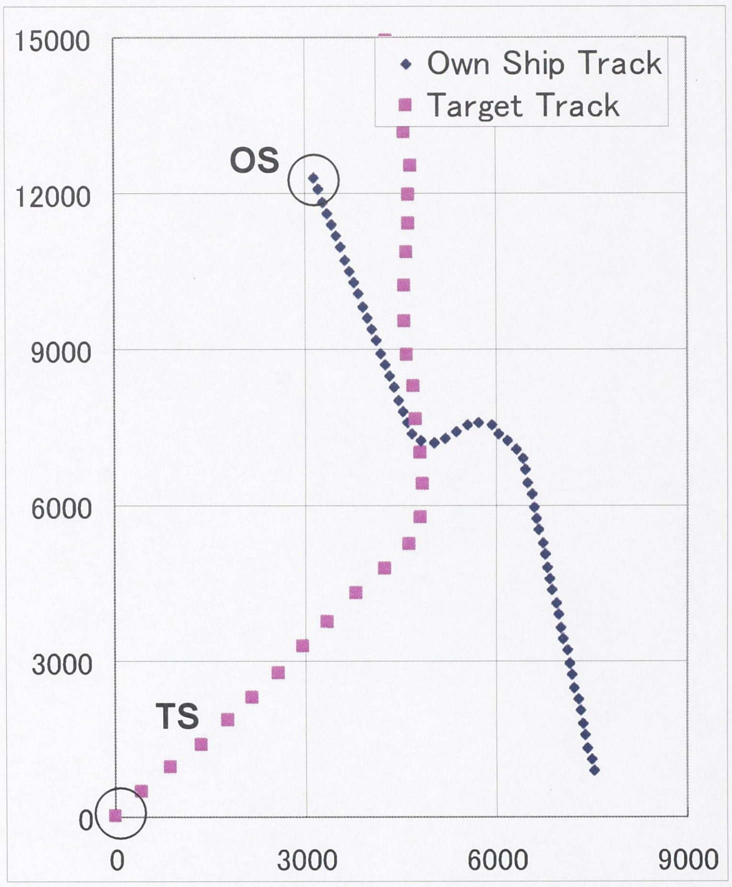


Fig.6.18a Actual Tracks of TS and OS

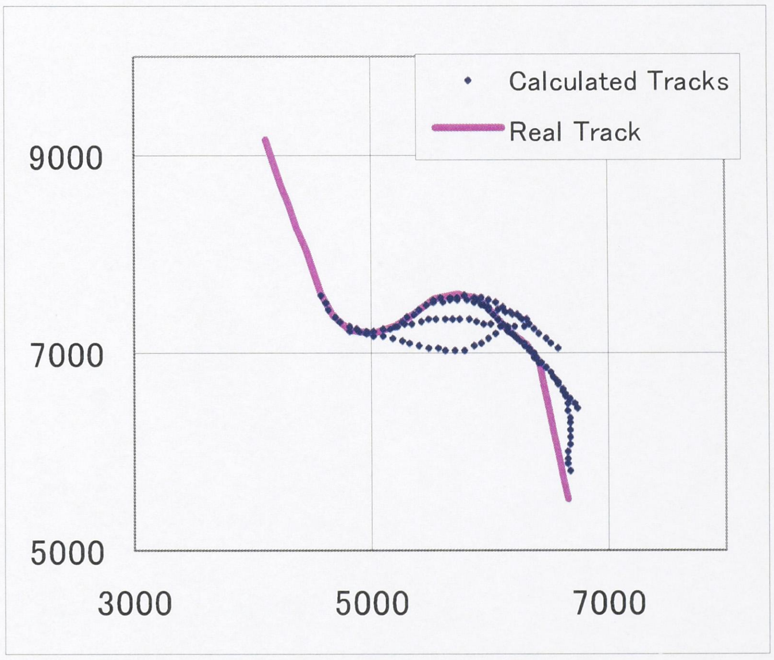


Fig.6.18b Calculated and Actual Tracks

It can be seen obviously in Fig. 6.18b that there is a mismatch between OS actual track and its calculated track for each sampling interval. However, due to the repetition of calculation, the error is retained under an acceptable limit.

Distance between the ships during the collision-avoiding process is shown in Fig. 6.18c. It can be seen that even with the deviation due to the model simplification, the OS is still kept at a safe distance from the TS.

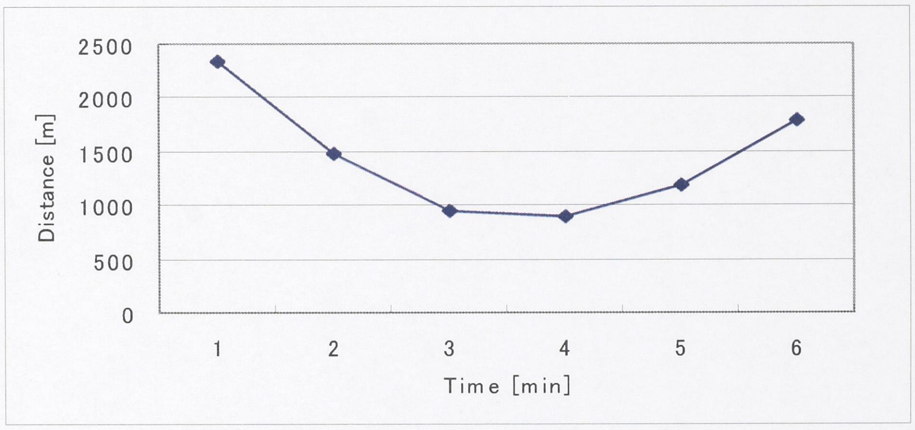


Fig.6.18c Distance to TS during Maneuver

6.6 Conclusion

In this chapter, the collision-avoiding problem of the OS in critical cases is analyzed as a pursuit-evasion game. The OS maneuverability has been properly taken into consideration by applying the T-K model to make the strategy realizable. The optimal strategy of the TS is assumed in advance, that is a strategy to cause a quick collision, and the optimal strategy of OS is determined accordingly by searching an approximation of the optimum of the OS payoff function. From computer simulations, it has been shown that:

- With a proper choice of OS payoff function, the collision-avoiding strategy for OS is appropriate.
- The Adaptive-BFOA algorithm is very efficient for solving the optimization problems arising in the games.
- The Extensive Game is more suitable for the application as the strategy thereby produced keeps the OS closer to its original route. However, the TS motion should be observed continuously and it may be necessary to update the OS frequently.
- The strategy produced by Strategic Game causes the OS to deviate largely from its original route. However, it ensures the safety of the OS for the whole calculation period.
- The mismatch between the OS actual position (MMG model) and its calculated position (T-K model) should be taken into consideration properly.
- Further study should be done on the collision-avoiding strategy by reducing/increasing the OS speed, in combination with its course.

References

1. J. Sgall, "Solution of David Gale's lion and man problem," Theoretical Computer Science, Vol. 259, No. 1-2, pp. 663–670, 2001

2. L. J. Guibas, J.C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, “A visibility-based pursuit-evasion problem,” International Journal of Computational Geometry and Applications, Vol. 9, no. 4/5, pp. 471-494, 1999
3. M. J. Osborne, "An introduction to game theory", available at “http://www.economics.utoronto.ca/osborne/igt/”
4. P. Cheng, "A Short Survey on Pursuit-Evasion Games", online available at "http://msl.cs.uiuc.edu/~pcheng1/courses/cs497/report.pdf"
5. R. Isaacs, "Differential Games", Dover, 1965
6. T. Basar and G. J. Olsder, "Dynamic Noncooperative Game Theory", SIAM, 1998
7. T. Miloh, S.D. Sharma, "Maritime Collision Avoidance as a Differential Game", 4th Ship Control System Symposium, 1975

Chapter 7 Conclusion

7.1 Conclusions

The dissertation is a structured study on the exploitation of the available navigational aids and computer calculating capacity to support the ship officers in observation and decision making. Its overall contents can be temporarily divided into the following 3 main parts:

- Observation support.
- Route-producing support for collision-avoidance in common navigating situations.
- Strategy-producing support for collision-avoidance in critical cases.

To verify the efficiency of the proposed supporting-algorithms, simulation studies have been performed with the MMG model of a container ship playing the role of the Own Ship for a collection of various encountering cases at sea.

From the experiments and simulation studies, the following pros and cons as well as the application notes on the above subjects can be deduced:

7.1.1 Conclusions on the Observation Support

Radar/ARPA and AIS are effective methods of target observation. The information acquired by these observing methods, needless to say, is an important and helpful reference for decision-making. However, there are still limitations accompanying with these 2 observing tools. As a supplement to them and to fulfill their limitations in some cases, an All-Time All-Weather Observing System based on cameras has been introduced, together with a floating-object detecting and tracking computer program.

- Camera (especially IR camera) observation is an effective tool for floating-object detecting and tracking purpose. It enables the detection of small objects not very far from the ship position.
- The observation by camera system allows the accurate and reliable tracking of targets at less than 2000 [m] distance from the cameras. The further the target is, the less accurately the object position can be estimated. However, the accuracy can be improved by applying a suitable fitting-algorithm (least mean square, e.g.)
- A very efficient sea-horizon detecting algorithm has been proposed basing on variation at different frequencies of the camera image pixels on the vertical direction. The performance of the algorithm is well over that of available edge-detecting algorithms.
- The proposed object-detecting algorithm provides an acceptable performance in a range of weather conditions frequently witnessed at sea.
- The effectiveness of the camera observing system is seriously reduced in bad weather due to the variation of camera height above the sea level and the errors in determining the sea-horizon line as well as the water-line of the object.
- The effective range of the observation system by camera is insufficient for collision-avoiding decision at longer distance (common situations) but can be an appropriate reference for decision-making in critical cases (shorter distance).
- The observation system using cameras is a highly potential for rescue mission application where human can appear sharply on the IR images due to the temperature discrepancy with his surrounding environment.
- Suitable choices of parameters in the algorithms have been discussed in respective sections of Chapter 2.

7.1.2 Conclusions on the Route-Producing Algorithms for Common Situations

The aim of these route-producing algorithms is to provide the ship officer a recommended collision-avoiding strategy both in the congested waters and at the open sea. To ensure the safety of the passage, different collision-risk accessing criteria may be applied. These have been thoroughly described in Chapter 3.

The construction of the grid and a suitable choice of the ship model have also been analyzed in details in this chapter.

In this study, 3 different route-generating algorithms have been proposed, including the algorithm basing on Dynamic Programming (DP), the algorithm basing on Ant Colony Optimization (ACO) and the algorithm using Bacteria Foraging Optimization (BFOA).

DP algorithm

- The algorithm allows the route to be produced quickly (it requires minimum calculation effort among the 3 algorithms).
- It enables the generation of route for situations in which it is difficult for the ship officer to determine a collision-avoiding strategy alone.
- DP is just an approximation method as it has treated the problem as time-invariant while it is in fact varying with time, the route generated is therefore an approximation of the optimal route or just a locally optimal route.
- The optimization function used in the algorithm is the minimum time route.
- The application of the traffic laws in this algorithm is complicated.

ACO algorithm

An ACO algorithm has been proposed for route-producing purpose, taking into consideration the nature of marine traffic. The algorithm differs from traditional ACO algorithms constructed by other authors in the following points:

- A local search (post-processing) mechanism is applied to increase the efficiency and the convergence rate of the search.
- Only the ants (say 70% of ants) that produce better routes lay trail-pheromone. Then, unpromising regions of the search-space can be ignored.
- A pheromone-manipulating algorithm (deamon actions) is used to drive the search.
- The searching algorithm is globally supervised by the use of the best route that has been found after each run (each ant generation).

In terms of route-producing capacity:

-The algorithm is able to generate a collision-avoiding route close to the optimal one in a short period of time.

- It allows the application of the maritime traffic laws (the rules of the road) so that the generated route is more appropriate from the experienced seamanship view point. This also reduces the possibility that target ships misunderstand the own ship intention and therefore act awkwardly.
- The algorithm is very flexible and robust i.e. it enables the route-producing even for situations in which the DP algorithm fails.
- A limitation of the ACO algorithm is that its performance is strongly influenced by the choice of parameters. Inappropriate parameter choice may drive the algorithm into an early convergence to the local optimums. A set of parameters has been suggested in Chapter 4.

Adaptive-BFO algorithm

Also, an Adaptive-BFOA has been introduced in this study for route-producing purpose. In comparison with other BFO algorithms, our proposed algorithm is different in the following aspects:

- A swim-length adapting mechanism is employed in this algorithm.
- It allows the bacteria to fully mature before they die. The algorithm therefore does not miss a promising region for optimality.

About the route-producing capacity:

- It is able to produce a collision-avoiding route very close to the optimal route in a short period of time.
- The search is very robust and flexible; a suitable route can be produced for extremely difficult situation where DP algorithm may fail.
- The rules of the road can be properly taken into account, like the ACO algorithm.
- The Adaptive-BFOA has better convergence property than the ACO algorithm.
- The choice of designing parameters is more flexible than that of the ACO algorithm.

From these perceptions, the Adaptive-BFOA can be considered as the most suitable algorithm for producing the collision-avoiding route for the ship. Thank to its flexibility and acceptable calculation time, it enables the real-time application onboard merchant ships.

7.1.3 Conclusions on the Route-Producing Algorithm for Critical Cases

The critical cases collision-avoiding problem is extremely important but is still insufficiently studied so far. To fulfill this gap, in this study, the problem is analyzed as a pursuit-evasion game in which the own ship plays the role of the evader while the target ship is the pursuer. T-K model is used to express maneuverability of the own ship to make the strategy appropriate (i.e. viable). It has been shown from the simulation studies that:

- By using a suitable payoff function for the own ship, the collision-avoiding strategy is appropriate.
- The Extensive Game model is more suitable for the application in critical cases as the trajectory of the ship it produces is closer to its original route.
- Strategic Game strategy causes the own ship to deviate largely from its original route. However, it ensures the ship safety for the whole calculation period i.e. it allows the system to calculate once and forget.
- The Adaptive-BFOA is an efficient algorithm for solving the optimization problems arising in the games.
- Due to the simplification of the motion model for the own ship (the use of T-K model), actual ship track deviates from its calculated track. This should be accounted for by adding an extra distance to the capture-distance chosen.
- A limitation of the study is that the speed-reducing strategy is not yet included due to the complexity of the resulting dynamic model. Further study therefore should be conducted on the matter.

7.2 Future Studies

Apart from the contributions of the study, several limitations remain and require more studies. Furthermore, other relating problems are also promising subjects for later researches.

7.2.1 Route Production under Wind, Wave Disturbances

The route-producing algorithm in this study is based on the dynamics of the ship (speed changing, course changing characteristics) in ideal weather condition. Unfortunately, the ship-dynamics is heavily influenced by wind and waves (drifting wave). This causes the ship to deviate from its calculated trajectory during the collision-avoiding process unless the counter effort is exercised. Unlike the course deviation which can be easily eradicated by rudder action, the speed increase/decrease draws more concerns as the engine revolution rate should not be adjusted too much often. To solve the problem, the ship dynamics model should be modified to include the effect of wind (and waves) in different direction. The modified model is then used for route-producing to ensure that the deviation of the actual route of the ship from the calculated one is kept under a limit. The problem needs more studies in the future.

7.2.2 Combination with Weather-Routing Algorithm

The route-producing algorithms basing on Ant Colony Optimization and Bacteria Foraging Optimization are strong tools for route-producing problem in our application. Furthermore, they are also potential solution for other optimization problems, including the weather-routing problem. For safer and more efficient navigation, the routing problem should be solved in the following 2 scales:

- Large scale routing: Producing the ocean crossing optimal route (weather route) for the ship.
- Small scale routing: Producing the collision avoiding route for the ship on every small part of the large scale route.

With the increasing understandings of the weather evolution and the accuracy of the weather forecast, weather-routing problem has received a lot more attention recently. It may be a subject of our later study.

7.2.3 Non linear Tracking-Control of Ship

An immediate task following route-producing is the realization of that route i.e. the route tracking. The tracking problem has been intensively studied in the last several decades but still not yet reached its mature. Almost all the tracking-control mechanisms are based on a linear model of the ship. The ship model however is not easily determined and there are also problems relating to the stabilities in real-time applications. A better tracking-control method should be proposed that enables real-time application in various weather conditions.

7.2.4 Cooperative Collision-Avoiding

The current researches on collision-avoiding are conducted with the assumption that the ships involved in the encountering situation work independently. With the rapid technology advancement, the communication between ships becomes much faster and more reliable.

Then, is it possible to establish a mechanism in which the ships cooperate to maximize the safety and navigation efficiency of the whole group? A first step might be the intention sharing through AIS receiver/transmitter. This possibility should be thoroughly studied in the future.

Acknowledgements

This Dissertation is to be submitted to the Tokyo University of Marine Science and Technology (TMUSAT) to fulfill the requirements for the degree of Doctor of Engineering in the field of Maritime Technology – Applied Environmental Studies.

First and foremost, I would like to express my sincere gratitude and respect to my supervisors, Professor OHTSU Kohei at the Marine System Laboratory and Associate Professor TAMARU Hitoi at the Route Planning Laboratory, TUMSAT for their continual supports, invaluable guidance and kind encouragements, both in research work and daily life in Japan.

My further gratitude and respect must go to Dr. SASANO Masahiko, Dr. KIRIYA Nobuo, Master IMASATO Motonobu, Dr FUKUTO Junji and others at Japan National Maritime Research Institute for granting me the chance and their generous guidance during my time taking a part in their project, from which I have been able to learn a lot.

It could have been extremely hard for me to complete the thesis without a very warm research environment, the close cooperation and helpful advice of Associate Professor SHOJI Ruri, Ms ODA Michiko and my friends, especially those in Route-Planning Lab. Their invaluable supports have always been an additional thrust-force for me to overpass difficulties.

I cannot end without expressing my thanks and dearest love to my family, on whose constant encouragement and love I have relied throughout my time studying in Japan and forever. Their love will always be the motivation inspiring me on my way to the land of hope and achievements.

Appendix I AIS Position Messages

The information necessary for collision avoiding decision is available in the position reports (dynamic messages) of AIS and partly from static messages (for Target Length).

For Class-A AIS, the Position Reports are Message 1, 2 and 3 in which the ship dynamic data are arranged as followings:

Parameter	Number of bits	Description
Message ID	6	Identifier for this message
Repeat Indicator	2	Indicate how many times msg has been repeated
User ID	30	MMSI Number
Navigation Status	4	0 = underway using engine 1 = at anchor
Rate of turn	8	0...± 126 (turning right/left)
SOG	10	Speed over ground in 0.1 knot step
Position accuracy	1	1 = high; 0 = low
Longitude	28	Longitude in 1/10000 min (negative for West)
Latitude	27	Latitude in 1/10000 min (negative for South)
COG	12	Course over ground in 1/10 degrees
True Heading	9	Degree (0-359)
Time stamp	6	UTC second when the report was generated
Reserved for regional application	4	
Spare	1	Not used. Should be set to zero
RAIM_Flag	1	Receiver Autonomous Integrity Monitoring flag of Electronic Position Fixing Device
Communication State	19	
Total number of bits	168	

For Class-B AIS, the ship dynamic data are encoded in Message 18 and Message 19 as listed in the table below (Message 18). The dynamic data part of Message 19 is exactly the same with that of Message 18 and therefore not to be shown to avoid repetition.

Parameter	**Number of bits**	**Description**
Message ID	6	Identifier for this message
Repeat Indicator	2	Indicate how many times a message has been repeated
User ID	30	MMSI Number
Reserved for regional or local applications	8	Reserved for definition by a competent regional or local authority. Should be set to zero, if not used for any regional or local application. Regional applications should not use zero.
SOG	10	Speed over ground in 0.1 knot step
Position accuracy	1	1 = high; 0 = low
Longitude	28	Longitude in 1/10000 min (negative for West)
Latitude	27	Latitude in 1/10000 min (negative for South)
COG	12	Course over ground in 1/10 degrees
True Heading	9	Degree (0-359)
Time stamp	6	UTC second when the report was generated
Reserved for regional application	4	
Spare	4	Not used. Should be set to zero
RAIM_Flag	1	Receiver Autonomous Integrity Monitoring flag of Electronic Position Fixing Device
Communication State Selector Flag	1	0 = SOTDMA Communication State follows 1 = ITDMA Communication State follows
Communication State	19	
Total number of bits	**168**	**Occupies one slot**

Appendix II Genetic Optimization Algorithm

```
Set  $Na$: number of agents in the population
     $N\_cyc$: Number of evolving generations.
     $Q(Na)$: Temporary cost array
     $P(Na)$: Temporary probability array
     $s^i$: position of agent $i^{th}$
     $s^*$: best position of the population
     $tmp\_s^i$: temporary position of agent $i^{th}$


1. Initialization
     $For\ i = 1\ to\ Np$
         $Initialize(s^i);$
     $Next\ i$
2. Evolution
     $For\ k = 1\ to\ N\_cyc$
       $For\ i = 1\ to\ Na\ (Q(i) = \cos t(s^i));$
       $For\ i = 1\ to\ Na\ \ (P(i) = Set\_\Pr obability(Q));\ (e.g\ P(i) = \frac{Q(i)}{\sum Q(j)})$
       $For\ i = 1\ to\ Na\ (tmp\_s^i = Evolution\_Operator(P, s));$

       $For\ i = 1\ to\ Na$
         $IF \cos t(tmp\_s^i) < \cos t(s^i) THEN\ s^i = tmp\_s^i;$
         $IF \cos t(s^i) < \cos t(s^*) THEN\ s^* = s^i;$
       $Next\ i$
     $Next\ k$
3. Termination
     $Return(s^*);$
```

Evolution Operation: (e.g. for the collision avoiding route producing purpose, where s = [p(1),p(2),...,p(N)], s(i) = p(i))

```
$probabilistic\_choice(mutation, recombination);$

IF mutation
    $generate(v = [v(1), v(2), ...., v(N)]\ where\ v(i) = 0\ or\ v(i) = 1);$
    $k = choose\_k\_probabistically(P);$
    For $j = 1$ to $N$
        IF $v(j) = 1$ THEN
            $temp\_s^i(j) = s^k(j)$
        ELSE
            $temp\_s^i(j) = random\_point();$
        END IF
    Next $j$
ELSE
    $k1 = choose\_k\_probabilistically(P);$
    $k2 = choose\_k\_probabilistically(P);$
    $flip = random\_choice(1, N);$

    For $j = 1$ to $flip$
        $temp\_s^i(j) = s^{k1}(j);$
    Next $j$

    For $j = flip$ to $N$
        $temp\_s^i(j) = s^{k2}(j);$
    Next $j$
END IF
```

Appendix III Particle Swarm Optimization Algorithm

```
Set  $Np$ : number of particles in the swarm
     $k_{MAX}$ : maximum number of repeatation
     $s_k^i$ : position of particle $i^{th}$ at time interval $k$
     $v_k^i$ : velocity of particle $i^{th}$ at $k$
     $p_k^i$ : best remembered position of particle $i^{th}$ till $k$
     $p^*$ : best remembered position of swarm till $k$
     where : $i = 1$ to $Np$; $k = 1$ to $k_{MAX}$
     $c_1, c_2, c_3$ : cognitive and social parameters
     $r_1, r_2$ : random number in $[0,1]$
```

```
1. Initialization
     Set_Parameters $(k_{MAX}, c_1, c_2, c_3)$;
     $k = 1$;
     For $i = 1$ to $Np$
         Initialize $(s_1^i, v_1^i)$;
         $p_1^i = s_1^i$;
         IF $\cos t(p_1^i) < \cos t(p^*)$ THEN $p^* = p_1^i$;
     Next $i$
2. Evolution
     For $k = 2$ to $k_{MAX}$
        For $i = 1$ to $Np$
           Generate$(r_1, r_2)$;
           $v_k^i = c_3 \times v_{k-1}^i + c_1 \times r_1 \times (p_{k-1}^i - s_{k-1}^i) + c_2 \times r_2 \times (p^* - s_{k-1}^i)$
           $s_k^i = s_{k-1}^i + v_k^i$;
           IF $\cos t(s_k^i) < \cos t(p_k^i)$ THEN $p_k^i = s_1^i$;
           IF $\cos t(p_k^i) < \cos t(p^*)$ THEN $p^* = p_k^i$;
        Next $i$
     Next $k$
3. Termination
     Return$(p^*)$;
```

For collision avoiding route producing task of this study, the position of a particle represents a possible solution (a safe route) that can be initialized by the solution initializing procedure in Chapter 5: s = [p(1), p(2), ..., p(N)] where p(i) denotes the point on line ith of the grid through which the route passes.

Appendix IV Kalman Filtering Algorithm

Kalman Filter has long been regarded as the optimal solution to many tracking and states predicting problems. It is a recursive filter that estimates the states of a dynamic system, basing on the linear model of the system and noise measurement of system output.

Consider a dynamic system with discrete state space model as followings

$$X_k = A \times X_{k-1} + B \times U_k + W_{k-1} \qquad (1)$$

$$Z_k = H \times X_{k-1} + V_k \qquad (2)$$

Where A, B, H are transition matrix, control matrix and observation matrix respectively. State variables and control input at time step k are expressed by vectors X_k and U_k.

Zk is a vector containing output measurements of the system, called observation vector.

The random matrix variables W_k and V_k represent the process and measurement noise respectively and are assumed to be independent of each other, white, zero mean with normal probability distributions.

$$p(W) \cong N(0, Q), \qquad Q = E[W \times W^T] \qquad (3)$$

$$p(V) \cong N(0, R), \qquad R = E[V \times V^T] \qquad (4)$$

In (3) and (4), process noise covariance Q and measurement noise covariance R are assumed to be constant.

Then, the system's state variables can be estimated by operating the Kalman filter recursively through 2 steps, namely "Predict" and "Correct":

Time update equations ("Predict"):

$$\hat{X}_k^- = A \times \hat{X}_{k-1} + B \times U_k \qquad (6)$$

$$P_k^- = A \times P_{k-1} \times A^T + Q \qquad (7)$$

Measurements update equations ("Correction"):

$$K_k = P_k^- \times H^T \times (H \times P_k^- \times H^T + R)^{-1} \qquad (8)$$

$$\hat{X}_k = \hat{X}_k^- + K_k \times (Z_k - H \times \hat{X}_k^-) \qquad (9)$$

$$P_k = (I - K_k \times H) \times P_k^- \qquad (10)$$

K is called Kalman gain matrix and P is covariance of estimate errors.

$$P_k = E[(X_k - \hat{X}_k) \times (X_k - \hat{X}_k)^T],$$

and $\hat{X}$ is the estimated state vector.

Appendix V Least Mean Square Algorithm

Least Square Method (Least Mean Square) has been commonly known as a standard approach to the approximate solution of over-determined system where the number of equations available exceeds that of variables (parameters) to be solved. Among LMS applications, the most popular and important one is, perhaps the data fitting. The aim of LMS is to determine a set of parameters so as to minimize the overall square error of the data set (measurements or observations).

LSM algorithms fall into one among the following 2 categories: Linear LMS (also known as ordinary LMS) in which the fitting model is a linear combination of the unknown parameters and Non-linear LMS.

For simplification, suppose that we have a data set of input and output data points $(x_i, y_i)\, where\, i = 1\ to\ n$

The model $y = f(x,\beta)\, where\, \beta = [\beta_1, \beta_2, ..., \beta_m]$ is used to fit the above data set.

1. Linear Least Mean Square Algorithm

The linear model is chosen as following:

$$f(x,\beta) = \sum_{j=i}^{m} \beta_j \times \phi_j(x)$$

where the coefficients of the models $\phi_j(x)$ are functions of x, and β_j are parameters to be determined. The following equation can be deduced:

$$X_{ij} = \frac{\partial f(x_i,\beta)}{\partial \beta_j} = \phi_j(x_i),\, where\, i = \overline{1,n}, j = \overline{1,m}$$

Then, the unknown parameters can be determined to minimize total square errors by:

$$\hat{\beta} = (X^T X)^{-1} X^T y \quad where\, y = [y_1, y_2, .., y_n]$$

2. Non-Linear Least Mean Square Algorithm

In the non-linear data fitting, $f(x,\beta)$ is a non-linear function of β_i . These unknown parameters are determined numerically by a successive approximating process:

$$\beta_{k+1} = \beta_k + \Delta\beta\, where\, \Delta\beta = [\Delta\beta_1, \Delta\beta_2, ..., \Delta\beta_m]$$

Where k is the iteration number and $\Delta\beta$ is the vector of increments that is often known as the shift vector.

Define the Jacobian matrix J with components J_{ij} as following:

$$J_{ij} = \frac{\partial f(x_i,\beta)}{\partial \beta_j},\, where\, i = \overline{1,n}, j = \overline{1,m}$$

Then, the vector of increments at each iteration is calculated by:

$$\Delta\beta = (J^T J)^{-1} J^T \Delta y \quad where\, \Delta y = [\Delta y_1, \Delta y_2, .., \Delta y_n]$$
$$\Delta y_i = y_i - f(x_i, \beta_k)$$

A problem usually faced in Non Linear Least Square Method is the convergence of the recursive algorithm. If the unknown parameters are properly initialized, the algorithm will converge rapidly to the exact values. Otherwise, the convergence is not guaranteed.