



Electronic Maneuvering Board And Dead Reckoning Tracker

Documentation

Revision: 6/22/2007



Table of Contents

Introduction.....	1
Quick Start	1
What is MoBo?	1
Moboards and DRTs	2
Plotting with MoBo	3
Units.....	3
Entering Unit Data	4
Unit Menu	4
Connecting Units	6
The Map.....	8
Standard Moboard.....	8
Importing Screen Images	9
Transparency Settings	10
MoBo Toolset	10
Compass.....	10
Dials	10
Connections or Bearing Lines.....	11
Movement	12
Show	12
Unit	12
Vector.....	14
Keyboard Commands	14
Standard Keyboard & Mouse Functions	14
Keyboard Shortcuts.....	15
Quick Tips.....	15
Simple Plotting Examples.....	16
Angle on Bow (AoB)	16
Determining Intercept Course	18
Adjusting Scale	20
Time-Speed-Distance (TSD).....	23
Advanced Plotting.....	26
Converting Relative to True Motion	26





Vector Addition	26
Target Motion Analysis (TMA)	30
Advanced TMA	34
General Moboard Terms and Definitions	35
Relative Movement (RM)	35
Direction of Relative Motion (DRM).....	35
Measurement of Relative Motion (MRM)	35
Speed of Relative Motion (SRM)	35
Closest Point of Approach (CPA)	35
M Observations	35
Time Speed Distance (TSD)	36
Nomogram	36
Relative Bearing (RB).....	36
True Bearing (TB).....	36
Target Angle or Angle on Bow (AoB).....	36
A Challenge... ..	37
About.....	38





Intentionally Blank





Intentionally Blank





Introduction

Quick Start

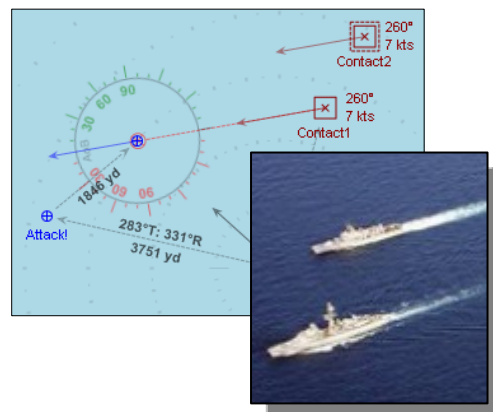
If you try to launch right into MoBo you probably won't get too far unless you at least understand the following:

- You can add units by double-clicking anywhere on the MoBo display
- To move a unit, left-click to activate the "Move" tool
- Right-click on units to display the Unit Menu of tools available
- The MoBo Cursor will always give an indication of what tool is active
- You must right-click to disable any active tool

What is MoBo?

MoBo is an electronic version of a Maneuvering Board or "moboard" for short. A moboard is a navigational charting tool used to solve relative motion problems involving ships at sea. Moboards have been around and in use for a long time. Long before computers were around moboards were used to answer questions like:

- *Can I intercept the contact?*
- *What is the target speed?*
- *What is the target heading?*
- *What is the Angle on Bow?*
- *How should I maneuver for a good attack position?*



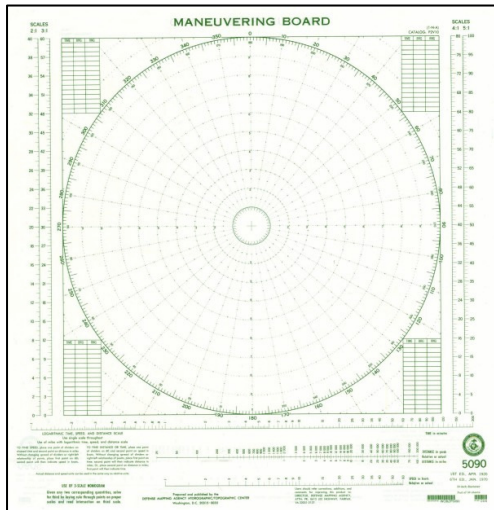
MoBo allows you to plot relative positions and perform the same types of calculations that you would on a manual moboard only it does a lot of the more tedious work for you. You get to plot the positions, and MoBo does the measuring. You'll find that MoBo is easier to use and a lot more flexible than a traditional moboard.

MoBo is different than traditional moboards in the following ways. First, you don't have to plot OwnShip in the center (but you can if you want to.) Instead, you can think of it this way; every unit plotted is like its own little moboard. Also, MoBo works seamlessly with either relative or true bearings; whereas, a traditional moboard would want everything converted to true. You'll also notice a number of different compasses, dials, and tools that are available to make your job easier and help you with many different types of visualizations.



Moboards and DRTs

MoBo can be used just like a traditional moboard. However, it also has another function; it can be used like a DRT (Dead Reckoning Tracker). I started building MoBo after many hours of playing a game called Silent Hunter 3. In SH3, I found that the navigation tools in the game



Typical Maneuvering Board – “Moboard”

were lacking certain features that I thought would surely be available to a submarine tracking party. Granted, they didn’t have laptops on WW2 U-boats; but the trig calculations for plotting intercept courses and so forth were certainly well known and could be figured out with a slide rule and moboards were also around and in widespread use.

I needed a tool that would allow me to take measurements based on my SH3 nav map observations. I started by just using the screen capture feature (Alt-PrnScrn) to copy images into other applications that I could use to make these measurements. Although it worked, it was a clumsy process and copy/pasting things to MS Excel was kind of taking me out of the whole gaming simulation experience.

What I really wanted was a virtual desktop where I could draw on maps with my own tools. And probably just like you, I wanted to be able to cruise quietly at 50-100m listening to that convoy in the distance; record bearings over time and be able to determine from those readings everything I needed to know to plan my attack. When I started out I knew nothing of maneuvering boards. In the quest to find answers to the problems I needed to solve I learned that the application I was building was very much like a tool that the US Navy refers to as a Dead Reckoning Tracker (DRT).

I was perfectly content with building a DRT-style tool. However, while developing and testing the software, relative motion began to slowly creep into my vocabulary. I realized that some of my plotting activities in DRT mode would be skewed by my OwnShip’s motion. I thought to myself, *“How do I compensate for OwnShip’s movement in my calculations?”*

That’s when I began to learn, in detail, about moboards and how they are used to solve relative motion problems. As I began to learn and incorporate the logic for relative motion calculations, my application evolved. I realized that I could use my tool to perform calculations exactly the same way that you would on a real-life moboard. In fact, I was plotting and solving problems directly from the US Navy Ops Spec Training Manual¹. What I really had in development was a stand-alone electronic moboard that could also serve a dual role as a dead reckoning tracker.

After you learn the basics of how the MoBo application works, I would highly recommend that you do as I did and print out [Chapter 11 of the US Navy Ops Spec NRTC](#)¹ (28 pages) and learn how moboards are used and maybe practice with some of the sample problems listed.

¹ US Navy Operations Specialist, Volume 01 NRTC: <http://www.globalsecurity.org/military/library/policy/navy/nrtc/14308.htm>







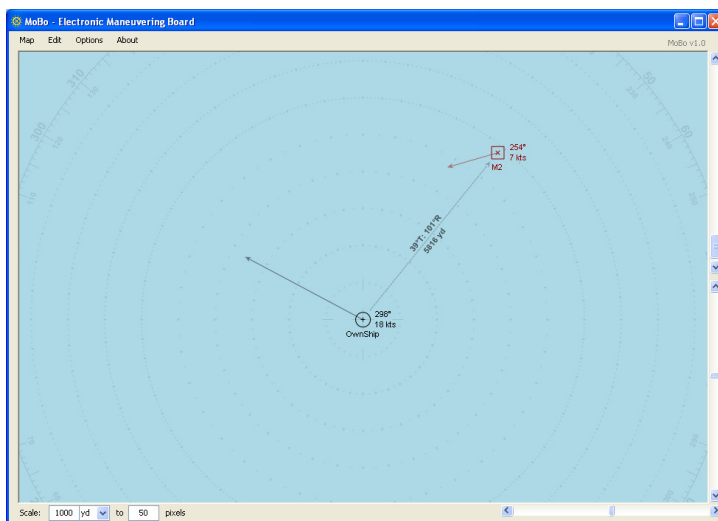
Plotting with MoBo

Units

MoBo is a special kind of plotter that allows you to work with Points and Lines on a two-dimensional Polar/Cartesian plane. A Line in MoBo (and mathematics in general) cannot exist by itself. Rather, a Line is defined as a connection between two points. The points that we'll be plotting in MoBo are special; they store information and have certain characteristics that we'll need to understand before moving on to forming line connections between units.

In MoBo, the "Points" are referred to collectively as "Units" and upon initial release; there are four different types of units available that you can plot:

1.  OwnShip
2.  Node
3.  Contact
4.  Add



The MoBo application display area with OwnShip and a Contact plotted.

When MoBo starts, the display area will be empty (no units plotted) and you'll see the default background map image of a moboard. The cursor will say "OwnShip" and you just left-click anywhere on the display to plot your own ship's position. Don't worry about precise positioning you can always move or nudge units around after plotting.

After you plot a unit, the type of the unit can also be changed at any time. Nodes can easily be converted to Contacts and vice versa. The only rule (that is enforced automatically) is that there can be no more than one instance of the OwnShip unit. You are allowed to delete OwnShip or to change OwnShip to a different type of unit and work with only Nodes and/or Contacts if you like but you can never have more than one OwnShip unit. At any time, if you find yourself without your OwnShip and needing it, just press the "S" key and you will bring up the OwnShip plotting cursor.

Each unit has characteristics that can be altered, viewed, or used in calculations. Some unit characteristics may include:

- | | | |
|------------|--------|--------------|
| • Location | • Time | • Notes |
| • Speed | • Type | • Connection |
| • Heading | • Name | |

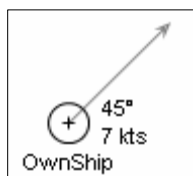




Entering Unit Data

MoBo was designed from the beginning to be a graphics based plotting tool. Top priority was given to reduce or eliminate keyboard entry of numerical data. The keyboard will be used mostly for hotkey shortcuts and occasionally entering observation times and notes. The majority of our time will be spent plotting and manipulating graphic representations of units. We get to focus on the fun stuff and let MoBo manage the slide-rule!

The units in MoBo are more than just display icons; they're little data storage devices. The more info you store in the unit, the more calculations you'll be able to perform. It's easy to enter the (x, y) location, in fact you'll do that without even thinking about it. Wherever the unit appears represents its location relative to the other units that you plot. Once a unit position is plotted you can then adjust the Unit Vector. A vector is a line that represents both magnitude and direction. By adjusting the length and direction of the Unit Vector you are supplying two key inputs: Heading and Speed.



The direction the vector is pointing represents the unit's heading and the length of the vector represents its speed. The Unit Vector serves two basic functions. First, it's a fast and easy way to enter heading and speed. Secondly, and probably most importantly, it gives you a strong visualization of the various units' relative motion. Since units with the same speed will have vectors of exactly the same length, you can instantly see which direction units are headed and which ones are moving faster or slower.

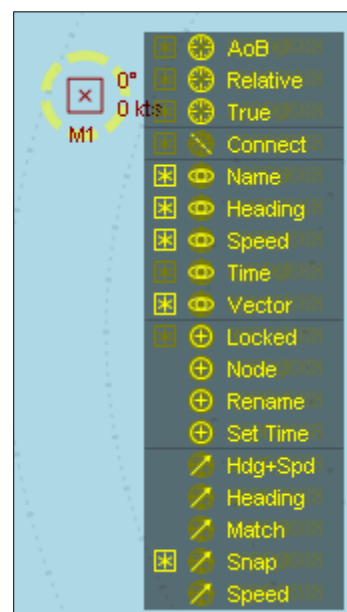
To adjust the Unit Vector, simply right-click the unit and select "Hdg+Spd" from the Vector tool section of the unit menu. Alternatively, you may also use the keyboard "V" shortcut by hovering your cursor over the unit until you see the "Select" cursor appear and then press "V" to enter the Unit Vector "Hdg+Spd" mode. Remember to right-click when you're done to disable the "Hdg+Spd" tool.

When adjusting the Unit Vector you'll notice that by default it will automatically *snap* to one knot and one degree increments. If you want greater resolution than one knot and one degree, you can pull up the unit menu and toggle off the "Snap" option for the Unit Vector.

Unit Menu

At first glance, MoBo may appear to be a very simple application. And actually, from a user's standpoint, it is indeed very easy to use. If you want to use MoBo for simple plots and determination of angles or basic time-speed-distance calculations; MoBo will happily do that for you. However, for those who want to plot more complex relative motion problems, you'll find a large toolset at your disposal.

Access to the MoBo Toolset is provided via the Unit Menu. To see the Unit Menu, simply right-click on any plotted unit. There are two things to keep in mind when using the Unit Menu. First, the menu displays different tools for different units. For instance, a Node unit will give you access to the "Note" tool for writing on the display,










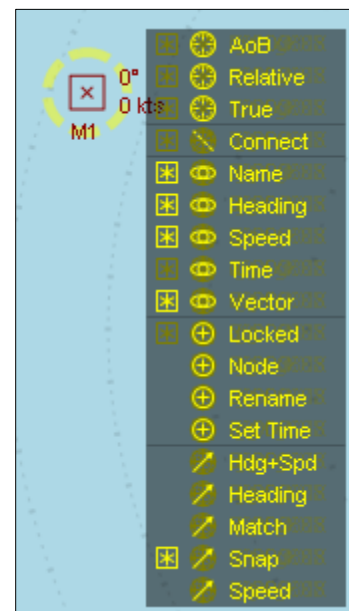


while the Contact unit gives access to the “Intercept” tool for assistance in determining a proper course to intercept. You’ll find that units do in fact share many of the same tools, but not all.

The second thing to keep in mind about the Unit Menu is that it is a *smart menu*. The menu will only display tools that you are able to use given the currently plotted data. Therefore, you will notice the number of tools available for each unit will grow as more information becomes available and is plotted. For instance, the Contact “Intercept” tool cannot determine a course of intercept unless it first knows about the motion of OwnShip and Contact. After you adjust the Unit Vectors for OwnShip and Contact, you’ll see “Intercept” appear as a usable tool on the Contact’s menu.

The Unit Menu includes categorization by tool type. Upon initial release, the tools all fall into one of the following seven categories:

1.  **Compass** Various types-styles of compasses
2.  **Dial** Any gadgets that move and rotate to help visualize data
3.  **Line.....** Connecting lines that use unit data to perform calculations
4.  **Movement...** Free and relative movement options
5.  **Show** Hide/show various bits of data
6.  **Unit.....** Unit operations, data entry, type conversion, renaming
7.  **Vector** Tools related to the Unit Vector



The Unit Menu

You’ll notice to the left of each tool there may or may not be an asterisk lit up. If you see an asterisk (lit-up or not) that means the tool can be toggled on/off. The asterisk light is your indication of what tools are presently active. In the menu displayed above right, notice that Name, Heading, Speed, and Vector are all visible and the Vector Snap tool is active for the currently selected unit.

To activate a tool simply left-click the appropriate menu item. Once a tool is active, it does not turn off unless you specifically tell it to do so by right-clicking. It doesn’t matter where you right-click, just right-click to deactivate. The default tool is the “Add Unit” cursor. Whenever the “Add Unit” cursor is visible you are in default mode operation and you may double-click anywhere on the MoBo display to add a Node-type unit.

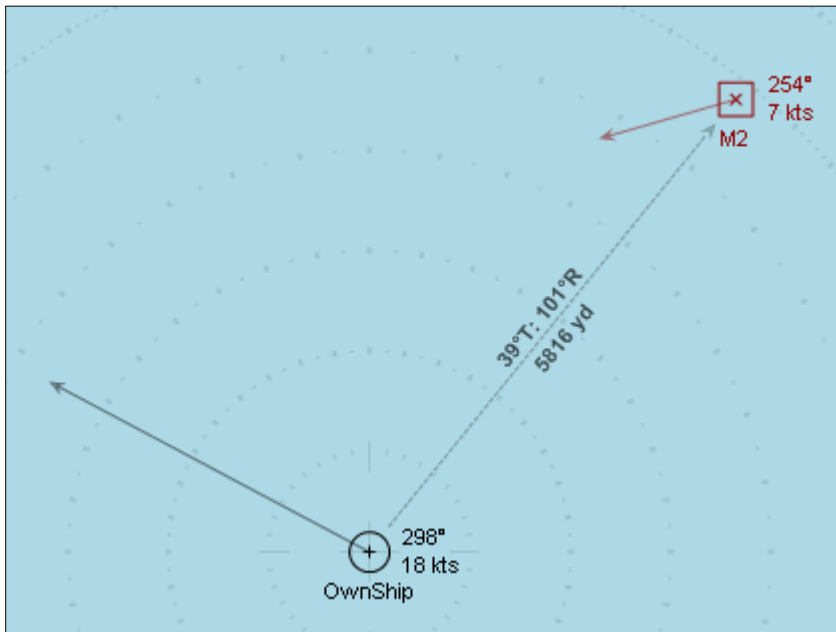


Connecting Units

If you think of the points or units as little data storage devices, then you can think of the lines that connect them as calculators. The connecting lines use the data in each unit to perform various types of calculations including: bearing, angles, elapsed time, speed, and distance.

For instance, the line can calculate the distance between two nodes because it knows the relative position of each node. If connected nodes have observed times, the line can tell you the elapsed time between the nodes.

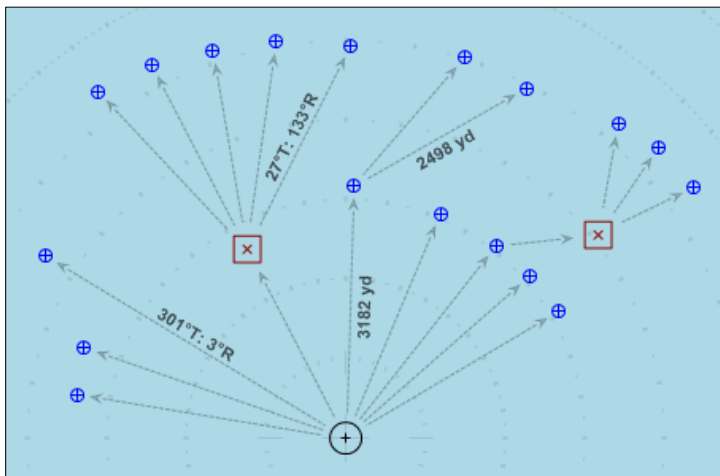
On the unit menu, under the "Line" series of tools you'll see "Connect" as an option. Just select Connect and then select the unit you'd like to form a connection with and the line is created.



The connected line arrow always points to the unit that originated the connection.

Alternatively, there is a simple shortcut key to form these connections quickly. Just hover the cursor over any unit and press the space bar to connect the unit to whatever unit is currently active. If no unit is selected the connection defaults to OwnShip. With a connection in place (remember, the line is like a mini-trig calculator) you can display various bits of information about the line; distance and bearing for instance.

This next part is important! Each unit is allowed to ORIGINATE only one active connection. The connecting line will ALWAYS show an arrow pointing to the unit that originated the connection.



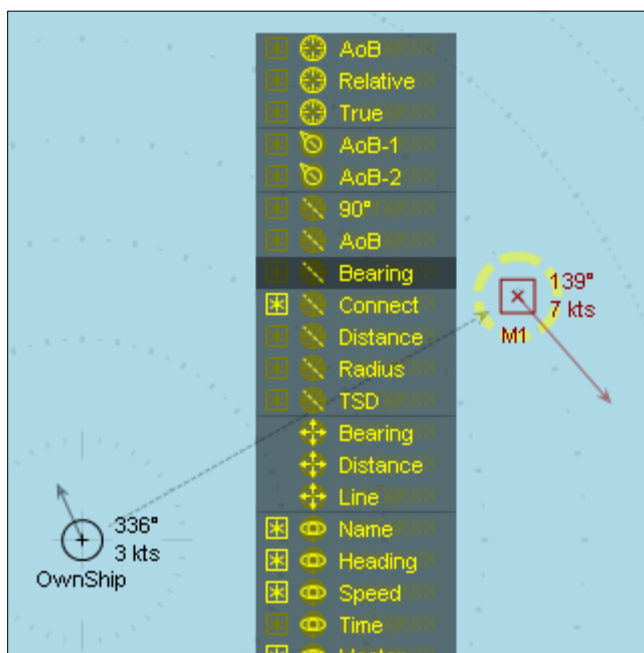
You can think of MoBo like a branching one-to-many database.

Although units are allowed to originate only one connection, there is no limit to the number of times a unit can be connected to. No matter how cluttered the display area gets, you can always tell which unit is associated with which connection by the arrow. The arrow of the line always points to the unit that originated the line. If you like to think in terms of databases, this system is an interesting graphic representation of a branching one-to-many database. So to quickly recap, Units store data; Lines are like



calculators – got it!

Knowing which unit originated the connection is very important. The tools associated with the connecting line will appear on the unit menu of the unit that originated the connection.



Display bearing on the connecting line that originated from Contact "M1"

So if I want to display the bearing on the line I would activate the appropriate unit menu and select "Bearing" from the line tools to calculate and display the bearing.

All connections, and therefore calculations, in MoBo are 'live'. That means once a connection is made, the units can be moved around and the lines will adjust accordingly. Calculations or items displayed on that line will also update automatically.

If I create a Contact and connect it to OwnShip, a line will be displayed from OwnShip to the Contact with the arrow of the line pointing at the Contact. If I display bearing for the line, it reads as bearing from OwnShip to Contact.

So the direction the arrow is pointing lets you know two things:

- 1) The arrow always points to the unit that originated the connection
- 2) Bearings and AoB are relative to the direction the arrow is pointing

It's important to remember these two things because if you want to display information on a line between two units, you need to select the unit that originated the line, and you need to know the orientation of the data (i.e. bearing to or bearing from). That's probably the most complicated aspect of the whole application; once you get the hang of that you're off and running!



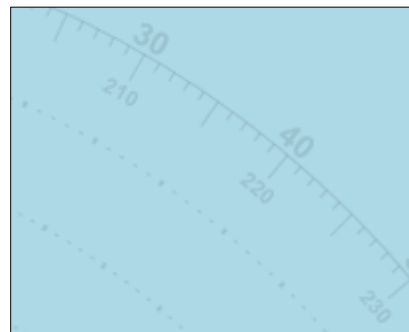


The Map

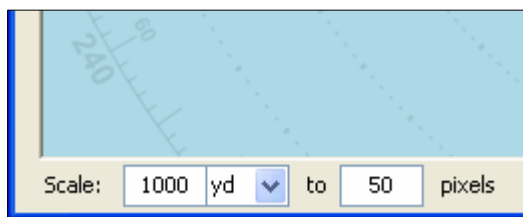
Standard Moboard

By default, MoBo loads with a standard maneuvering board image as the background. The image is scaled such that 50 pixels equates to 1000 yards. The moboard map has 10 rings and, by default, the rings are scaled to 1000 yards. So the default display with 10 rings is good for up to 10,000 yards.

On a real moboard you can choose to associate different scales



Default moboard-style image



MoBo scale settings

with the rings. In

MoBo, you are allowed to set the rings at any scale you'd like to work with and you can also choose between different units of measure in both the Imperial and Metric systems. As mentioned, the default setting for scale is set at 1000 yards to 50 pixels. That scale is always visible and can be adjusted in the lower left corner of the MoBo

application window.

So if I needed to work on a larger scale, I could just set the scale to 4000 yards to 50 pixels and easily convert the work area to cover 40,000 yards.

Users with higher resolution monitors may also wish to adjust the pixel setting so the moboard image takes up more space on their screens. By default, the background moboard image is drawn on a 1300x1300 resolution bitmap. You can change the size of the image to work on larger images if you like; just be aware that *very large* images can cause a noticeable slow down in the performance of the application.

It's important to remember that the rings of the standard moboard background image are tied to the pixel setting for scale. By default the rings are 50 pixels apart. If you change the pixel settings (and you will) the distance between the rings would also change. A user with a higher resolution monitor might want to change pixels to 75 to work with a larger image.



You can access the height and width settings for the background map from the Options menu.

You'll see examples later that show you how to adjust the scale to be in-sync with a game map, in which case, you may be using 240 or more pixels. If you loaded the moboard image with a 240 setting for pixels you might only see the first couple rings on the display. If you find yourself trying to load a moboard image and you don't see the rings just make sure the pixel count isn't too high.



Importing Screen Images



Import your favorite game map images directly from the PrtScn Buffer

map or you could clear the background image and work without all the background clutter. Alternatively, if you have a nice official looking image of a standard 5090 Maneuvering Board that you'd like to use as a background, you can do that too.

You can load and save images created in MoBo. So if you want to develop a tactical manual and you need to show positions or illustrate attack strategies, you can use MoBo to generate and share those bitmaps.

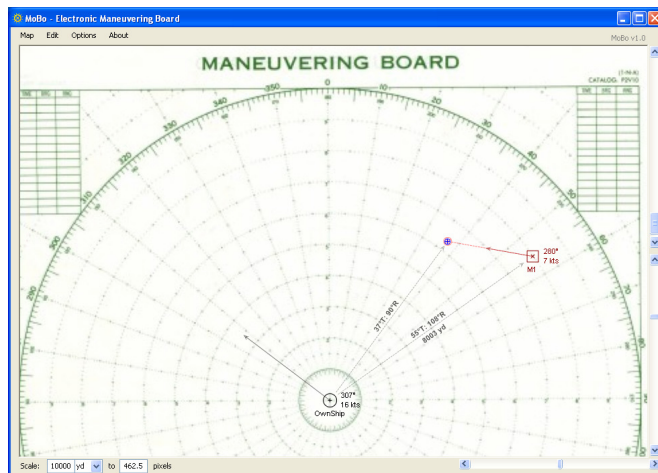
You can access load, save, and print screen options from the "Map" main menu bar at the top left of the MoBo application. Also, to quickly update images from the print screen buffer you can just press the "Insert" key on your keyboard.

Putting images in the print screen buffer to use with MoBo is very simple. While playing your favorite game just press the "Print Screen" button on your keyboard. If you're using multiple monitors remember to press "Alt-PrtScrn" to capture only the active game window or you might end up with a huge bitmap image that spans multiple monitors as your MoBo background.

MoBo is unique in that it allows you to instantly import images from your favorite game maps directly from the Windows Print Screen Buffer. You also have the option of loading any kind of bitmap you'd like to use as a background. All standard bitmap formats are supported.

So rather than having to carefully plot where you *think* units might be on the moboard, you can just pull the map image straight into MoBo and plot your units right on top of the game map image!

Once the units are plotted you have the option of continuing to work on the game



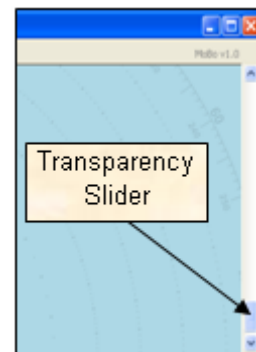
Load any kind of background image that you'd like to use

Transparency Settings

There are two additional options for plotting relative positions in MoBo. You can use the transparency slider on the top right-hand side of the MoBo application to allow see-thru functionality, or you can set the background map to be completely transparent.

There are times when I prefer to work on uncluttered background so adjusting the transparency gives me an alternative to use MoBo as an overlay. I adjust the transparency; plot my positions, then set transparency back to opaque to work on my plotting. This is only useful if you have games or applications that are capable of running in “windowed” mode.

As mentioned, you can also set the background map to just be completely transparent. You can access that feature from the main menu “Map” option under “Load”.

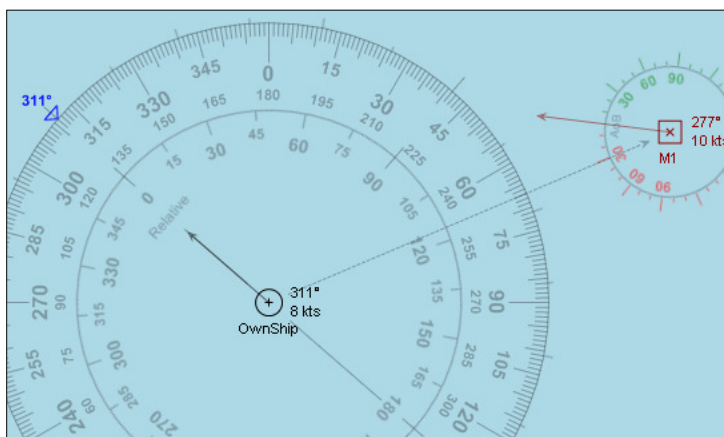


MoBo Toolset



Compass

Compasses are tools that provide a visual display of degrees for a specified unit. At the time of initial release, there are 3 different types of compasses that you can display for a unit: AoB, Relative, and True.



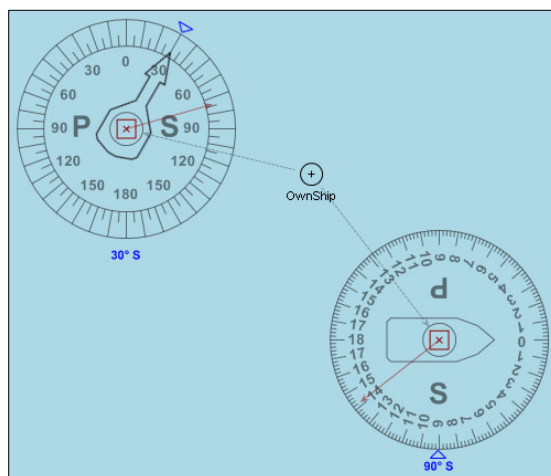
OwnShip Relative and True Compasses with AoB compass on the Contact



Dials

Dials are slightly different than compasses in that they have moving parts. When using dials you’ll notice that the dials change and rotate as relative positions of the units are changed.

The two dials that are presently available are related to the AoB dials that you might encounter while playing your favorite WWII sub game. Rather than just telling you what the AoB of a target is, MoBo is showing you exactly how to set a TDC dial to properly reflect the situation. Moving the units around and watching how the dials change also provides a helpful visual tutorial on AoB settings.

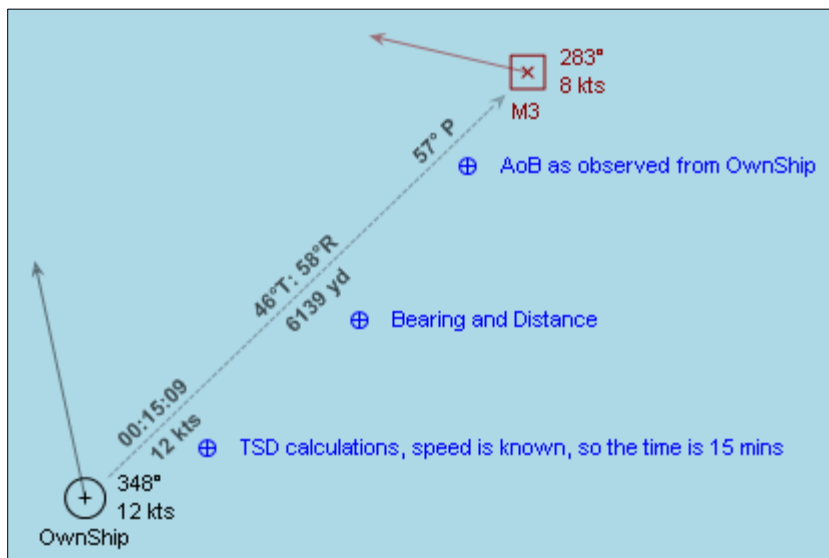


“OK, now I think I understand how those things work!”

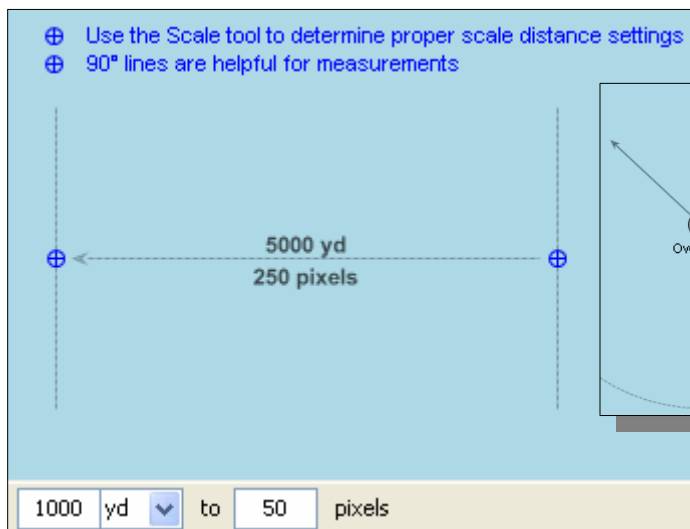
Connections or Bearing Lines

Once bearing lines are plotted, there are a number of different tools you can use to display various calculations related to the lines:

- 90°
- AoB
- Bearing
- Connect
- Distance
- Radius
- Scale
- TMA
- TSD

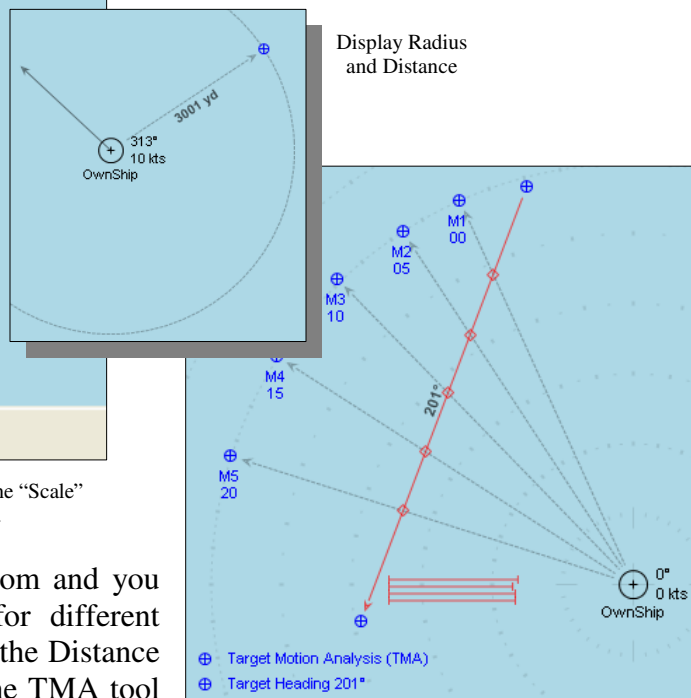


Lines are designed to always display information in the same place. Moving OwnShip or the Contact would cause this information to change dynamically as you move the units around.



Sometimes you'll need to "Connect" nodes to make use of tools. The "Scale" tool becomes an option for the line when you "Connect" two Nodes.

There are many different tools to choose from and you can use different combinations of tools for different results. You can display a Radius and show the Distance on the line; or connect two nodes and use the TMA tool to plot a course; then display the Bearing for the TMA line. Combining the different tools gives you great flexibility in solving all different types of relative motion problems.



Target Motion Analysis – see the example on "Target Motion Analysis" for more details



Movement

You've probably discovered on your own that, by default, if you left-click a unit the standard "Move" cursor will appear and allow you to click other locations on the map to move the unit around. If you hold down the left mouse button while the "Move" tool is active it allows you to drag units around the screen.

Remembering to right-click to disable the "Move" tool is a common mistake that results in moving a unit when you really didn't want to move it. If that happens to you, just remember there's an "Undo Last Move" feature on the main menu that will *usually* allow you to put the unit back where it belongs.

There are some situations where you'll find it helpful to be able to move the unit *relative* to a connected unit. For example, you may position a unit such that the connecting line shows a bearing of 315° and distance is displayed as 8150yds. You need to keep the heading at 315°, but extend the distance of the line to 9500yds. That's when the following list of relative Movement tools can be very helpful:

- **All** - move all units relative to OwnShip movement (OwnShip menu only)
- **Bearing** - hold distance constant and move bearing
- **Distance** - hold bearing constant and move distance
- **Line** - hold both bearing and distance constant

In actual practice, I find myself doing the following: I'll import a game map, do some plotting, return to game, make some course corrections, then re-import the game map, do some more plotting, return to game, make adjustments, re-import, and so on. During this process I may have to move the map around in the game. If I re-update the map in MoBo my units are out-of-sync with the map image in the background. This may suggest to you that you should not be moving around the in-game map image if you want to work with MoBo, but that's just not the case!

The "All" relative movement option allows you to easily realign your plotted units with changes that may have occurred in the background image. The "All" movement option is only available from the OwnShip menu. So just remember if your map isn't lining up with your plotting work, select "All" from the OwnShip menu and then move OwnShip to realign your work. The only thing you really have to avoid is changing the map scale. For fine tuning positions you are also allowed to use the arrow keys to "nudge" units pixel-by-pixel.



Show

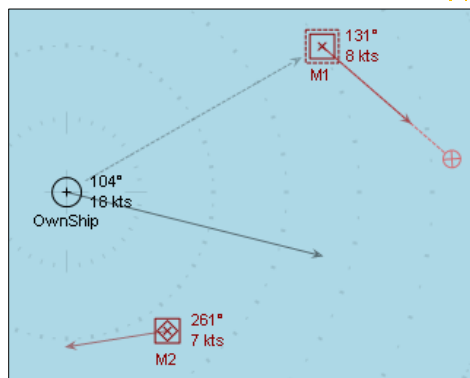
Very simple... the tool icon looks like an eyeball, so you guessed it, these are display toggles for various bits of information related to the unit. You can toggle the unit displays for: **Name, Heading, Speed, Time, and Vector**. Sometimes you'll just want to shut them off so you can see things better.



Unit

Unit tools allow you to make various types of adjustments to the unit, including: entering observation time, converting to different type of unit, locking unit position, renaming, and setting unit position as relative.

- Add - convert unit to an Add-type unit
- Contact - convert unit to a Contact-type unit
- Locked - prevent accidental movement
- Node - convert unit to a Node-type unit
- Note - enter notes (Node-type unit only)
- Relative - set position as being relative to another unit
- Rename - rename the unit
- Set Time - enter an observation time for the unit



Contact M1 is set to Relative, Contact M2 is Locked

If you choose to turn on the “Locked” status of a unit, you’ll notice a diamond shape will appear within or around the unit. When units are locked, you can not move them, but you can still enter information or change their vector settings. To unlock a unit just select “Locked” again from the unit menu to toggle off.

The “Relative” option only appears on the unit menu if the unit is connected to another unit. Turning on “Relative” means that the unit will maintain its relative position (the line bearing and distance does not change) to the connected unit if you move the connected unit. Setting a unit to Relative does not automatically lock the unit. You can still move around a relative unit if you select it. You can set a unit to both Locked and Relative if you want.

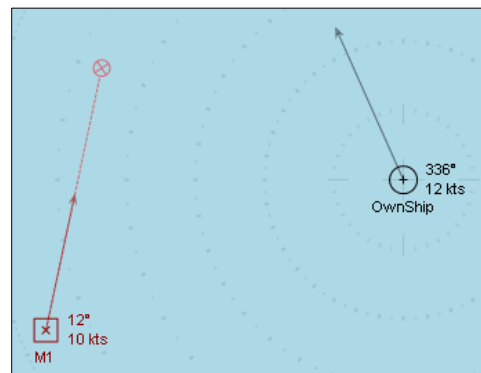
The “Set Time” option allows you to enter an observation time for the unit. This can be useful for various TSD-type (time-speed-distance) calculations. You’ll see an example later on how to determine speed given two unit observations with times entered.



Vector

Vector tools are related to adjustments and different calculations that can be performed using the units motion or Unit Vector. There are tools for adjusting the vector heading, speed, or both; as well as tools that use vector information to perform calculations. The following tools are associated with the Vector toolset:

- Hdg+Spd - adjust heading and speed
- Heading - adjust heading only
- Intercept - calculate intercept
- Match - set vectors equal
- Snap - 1 knot 1° increment (default = ON)
- Speed - adjust speed only
- Vector1 - assign Vector1 (Add-type unit only)
- Vector2 - assign Vector2 (Add-type unit only)



If OwnShip and Contact speed and heading are known an Intercept can be calculated (see “Determining Intercept Course” for more details)

Keyboard Commands

Standard Keyboard & Mouse Functions

The standard keyboard commands include the following:

- Esc - disable the active tool and return to default “Add Unit” cursor
- Arrow Keys - can be used in conjunction with “Move” to *nudge* units by 1 pixel
- Delete - delete the currently selected unit
- L-Mouse - left-mouse double click to add a Node-type unit
- N - add Node-type unit
- M - add Node-type unit with “M” designation visible
- C - add Contact-type unit
- S - activate OwnShip plotting cursor (if not already present)
- Insert - load map image from Windows Print Screen Buffer
- Numpad “+” - add an Add-type unit



Keyboard Shortcuts

The following keys can be used with plotted units to perform certain tasks quickly:

- With cursor over any unit...
 - Spacebar - create a connection with OwnShip or a selected unit
 - A - toggle AoB line text display
 - B - toggle Bearing line text display
 - D - toggle Distance line text display
 - I - toggle Intercept vector display
 - L - toggle unit Locked mode
 - T - enter observation time
 - R - toggle unit Relative mode
 - V - activate Vector “Set Hdg+Spd” mode
 - Y - toggle TSD line text display
 - = - set vectors equal (Vector Match shortcut)

Quick Tips

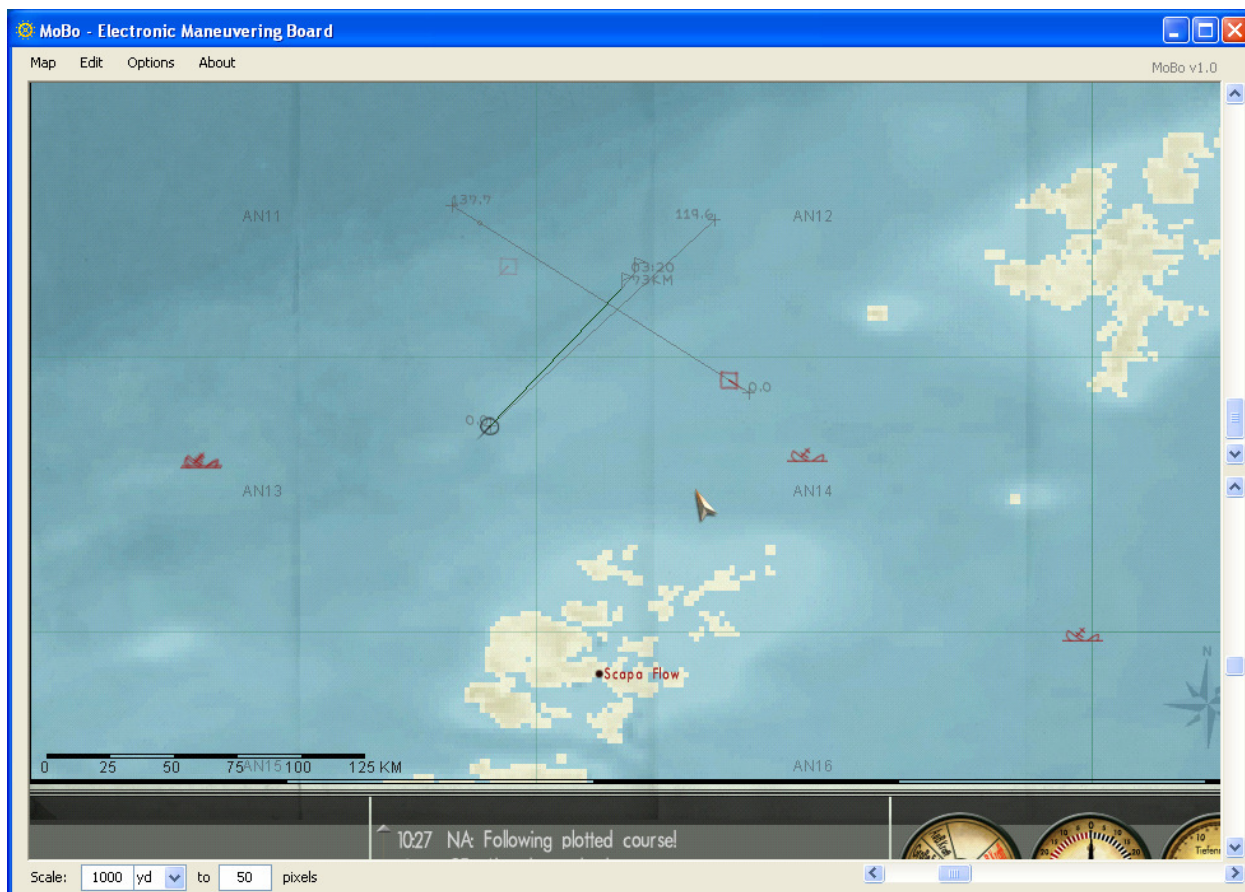
- You can quickly add connected Nodes or Contacts by pressing N, M, or C followed by the spacebar.
- The following sequence will add a Contact; connect it with OwnShip, and display bearing, distance, and AoB: C-Space-B-D-A
- You might not see an Intercept if OwnShip isn’t moving fast enough to catch the Contact.
- You don’t have to memorize all the shortcut keys to use MoBo. All of these commands are accessible via the right-click Unit Menu... but they are convenient shortcuts!
- Remember the Edit menu at the top of the MoBo application allows you to undo the last unit Move action.
- Units are allowed to overlay one another. If you find yourself needing an extra connecting line originating from the center of a Contact... BE CREATIVE! No one is stopping you from plotting or moving another node right on top of an existing unit.
- If at any time you find yourself *locked-up* in a tool, press the ESC key to deactivate the tool and return to default mode.



Simple Plotting Examples

Angle on Bow (AoB)

In this example we start with an in-game screen capture (Alt-PrtScrn) and import that to use as the MoBo background (press Insert key). So we start with the following image loaded in MoBo:



Granted, in this particular example, we're so far away from the contact that we wouldn't be concerned about AoB at this point, but that doesn't really matter. The same steps would be used regardless of range.

(see next page for step-by-step instructions)



Step 1: Plot OwnShip and Contact positions



Step 2: With cursor over Contact “M1” press the V key to activate Vector Set Hdg+Spd mode. Adjust the vector accordingly. We’ll assume the unit is moving at 7kts and the heading is 302°.



Step 3: With cursor over Contact “M1” press the Spacebar to create a bearing line connection with OwnShip.



Step 4: With cursor over Contact “M1” press the A key to display AoB for the bearing line

The solution for AoB is 43° Port. If you want to, you can right-click on the contact and display one of the AoB dials to show exactly how the TDC (Target Data Computer) dial would need to be set to reflect the 43° Port AoB solution.



Determining Intercept Course

For this example we'll use the same in-game map image from our AoB example above. The situation is, we're cruising along and we get a radio report for a contact. Our navigator has plotted the position on our in-game nav map and we know the contact is moving "slow" and the approximate heading. We are approximately 100 km from the contact's last known position and we need to plot a course to intercept given what information we have.



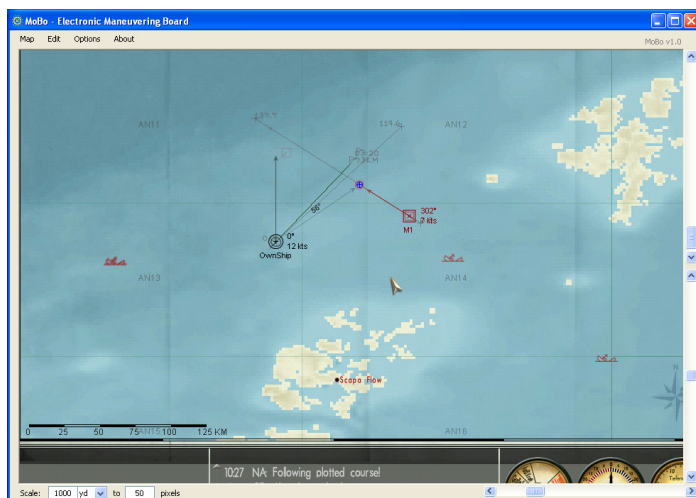
Step 1: Plot OwnShip and Contact positions



Step 2: Adjust the Vectors for Contact and OwnShip. Again, we'll assume the unit is moving at 7kts and the heading is 302°. OwnShip intercept speed set for 12kts.



Step 3: With cursor over Contact, press the I key to calculate and plot an intercept.



Step 4: Double-click on the intercept point to add a Node, then with cursor over the Node press Spacebar-B to draw the bearing line and show the heading.

The direction that OwnShip is heading is irrelevant; I just used 0° in my example above. We're trying to determine a course for intercepting the contact and our present course will change to the intercept course once we figure out what that is. At the end of Step 4 we know that the heading we need to take is 56° assuming we want to intercept at a speed of 12kts.



The speed of OwnShip doesn't always have to be greater than the speed of the contact to generate an intercept. However, if MoBo cannot determine an intercept, you just won't see an intercept plotted. Try increasing OwnShip speed or decreasing the Contact speed and the intercept plot will appear.

The intercept plot will automatically adjust to changes in OwnShip or Contact speed. By default, the intercept plot is in reference to OwnShip. If you connect the Contact to another unit, the intercept plot will be in reference to the connected unit's speed, not OwnShip's.

An intercept cannot be determined unless you have at least:

- 1. Speed and Heading for the Contact**
- 2. An intercept speed for OwnShip**

It's also worth noting that in this example the intercept course is listed simply as 56° . That's because I set OwnShip's heading at 0° . The heading of 56° is correct for either Relative or True course setting. If OwnShip's heading was 315° , MoBo would detect that and give you the bearing to the intercept point as: " 56°T : 101°R ".

I know at least in the Silent Hunter games, it's easy to order a course change on the bridge by using a sighting tool to point at a relative direction and press the "=" key to order the heading change, in this case 101°R .

The alternative would be to use the navigator tools in the game map to lay down a 56°T course line and then use the course plotting tools to set course. How you decide to order the course change is up to you.

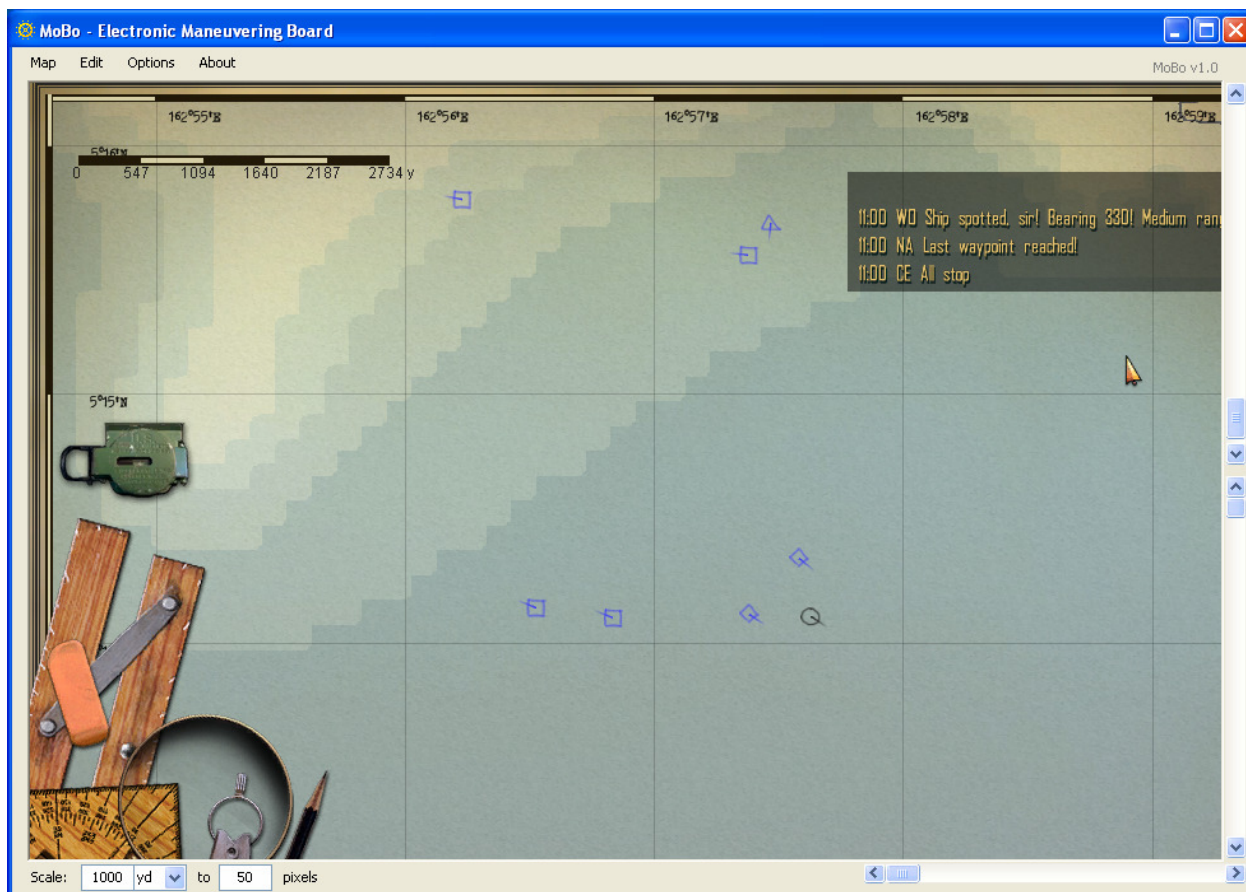




Adjusting Scale

If you want to do any kind of calculation that involves distances, you will need to make sure the map scale in MoBo is set appropriately. The good news is MoBo is very flexible and allows for unlimited scaling options in several different units of measure in both the Imperial and Metric systems.

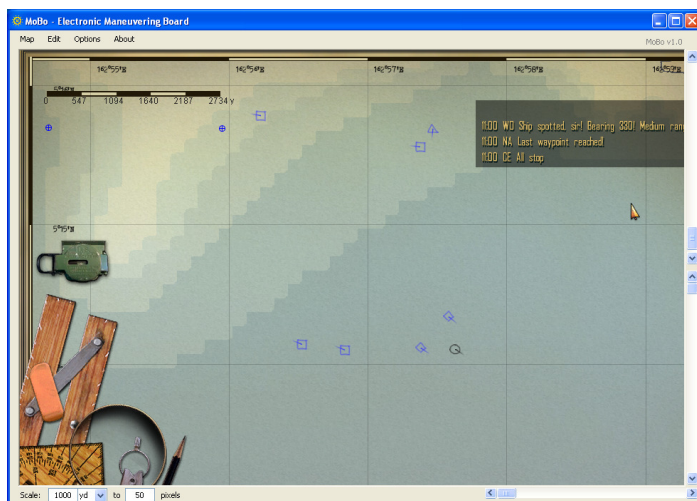
We start with the following screen capture:



In this case notice the default scale for MoBo is 1000 yards to 50 pixels. If I want to measure distances on the game map and have them be correct; I need to change the scale settings in MoBo to reflect the proper scale for the game map that you see in the upper left corner.

(see next page for step-by-step instructions)

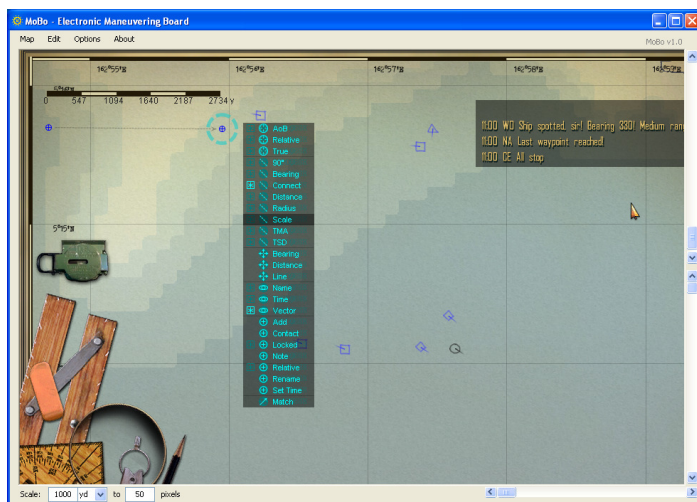




Step 1: We start by just plotting two nodes near each end of the scale. Just move the cursor near the spots and press the N key.



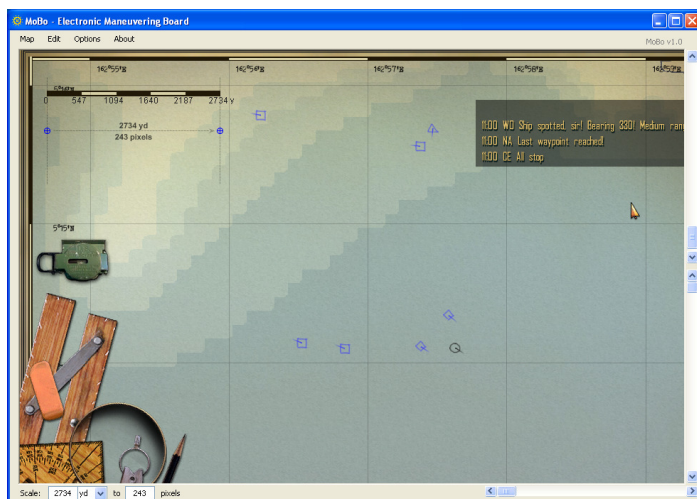
Step 2: Right-click on the node to the right and select “Connect” from the unit menu. Choose the other node as the target.



Step 3: With the connection in place right-click on the node to the right again. We now have expanded our toolset options to include “Scale” as a line tool option. Select it by left-clicking.



Step 4: The Scale tool automatically adds 90° bars to the ends of the line and displays pixel distance. We carefully position the nodes to line up with the map scale and we can see the pixel distance is 243 pixels and the game map has that distance as 2734 yards.



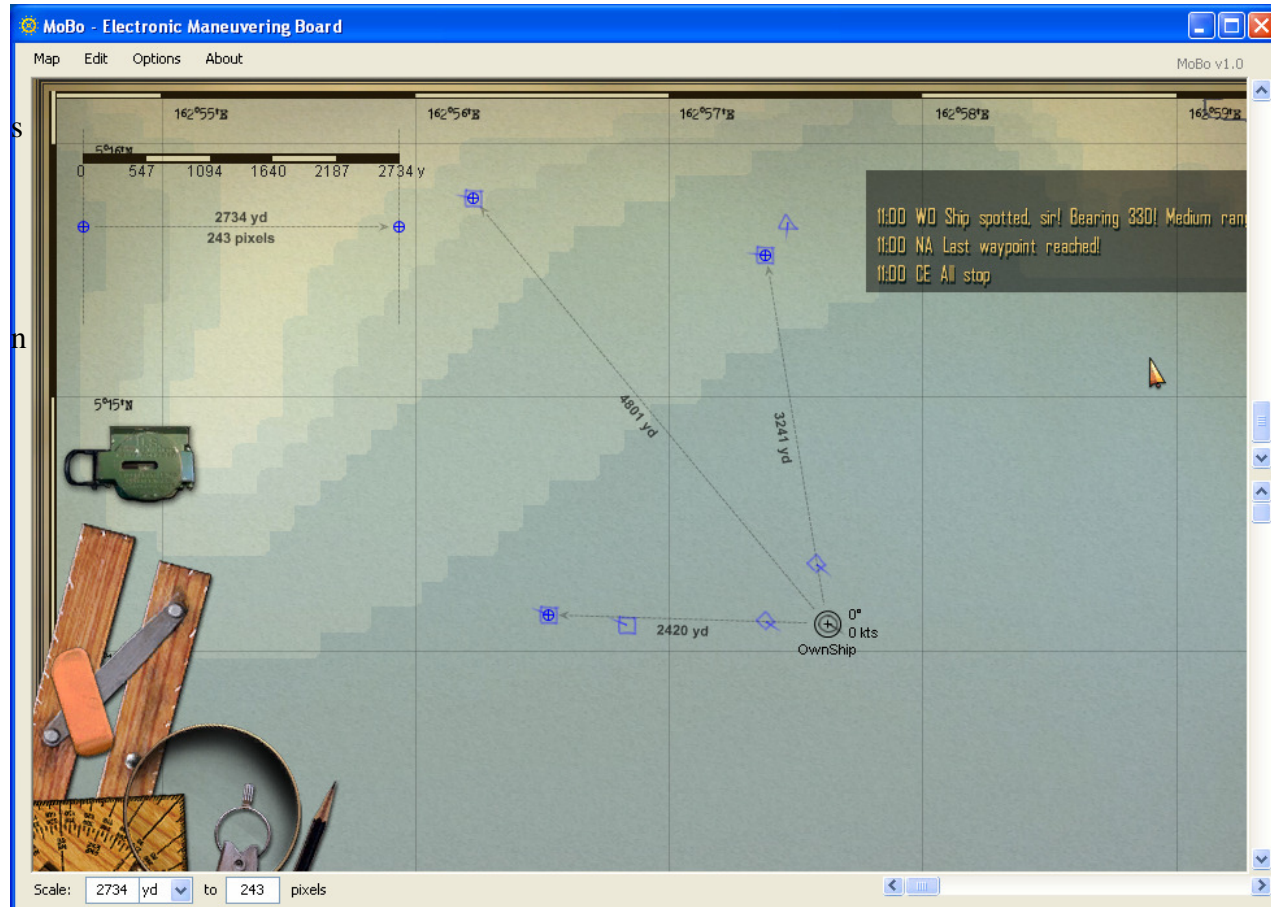
Step 5: We adjust the scale in MoBo by changing the scale input boxes in the lower left corner of the application to be in-sync with the game map. We enter 2734 yd to 243 pixels.

When entering values in those textboxes, be sure to keep the mouse hovering over them. If you move the mouse over the map display area it will automatically deactivate your text input.

When you update the values in the scale textboxes you’ll notice that your plotted scale line will read: “2734 yd, 243 pixels” and now you know the MoBo scale is in-sync with your game map.



Now that MoBo is in-sync with the game map we can plot nodes, form connections, and display distances on the lines. The distances are now an accurate measurement.



One nice thing about game maps in general is that within each game, the scale size in pixels tends to always be the same; it's just the numbers on the scale that change. That means we can leave the pixel setting at 243 and as we use different scales in the game, we just change the input for yards or meters or km or whatever to be in reference to the 243 pixels.

The important thing to remember is I don't always have to use the scale tool to re-measure the pixel distance every time I change my game map scale. If the pixel scale was 243 last time, it probably isn't going to change. If the pixels are known, all I have to do is change the inputs for the left side of the scale to re-sync the maps.





Time-Speed-Distance (TSD)

If we know how to adjust scale and can determine accurate distances between objects, it's very easy to calculate the time it would take to cover that distance. Alternatively, if we know the position of a unit at different times, we can enter observation times for the unit and determine the unit's speed.

On a real moboard you would use a nomogram to solve TSD problems. In MoBo, the nomogram is a function that's built into every connecting line. All you have to do is provide the inputs and display the TSD results. If your scale distances are correct (see previous Adjusting Scale example) you can accurately solve for unit speed, or time elapsed.

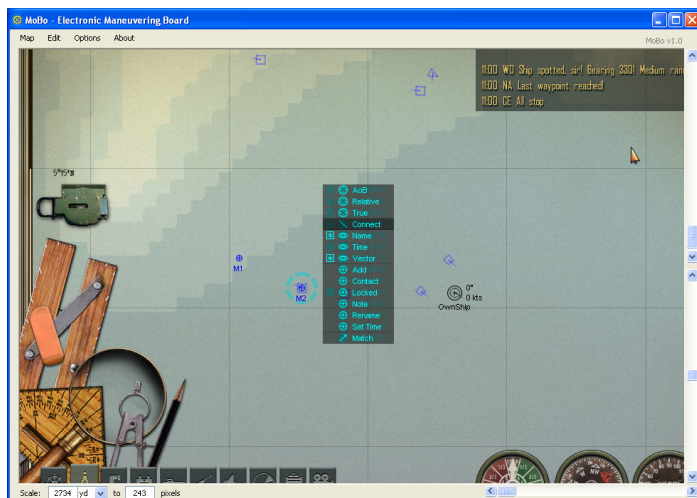
Assuming our scale is correct we start with the following:



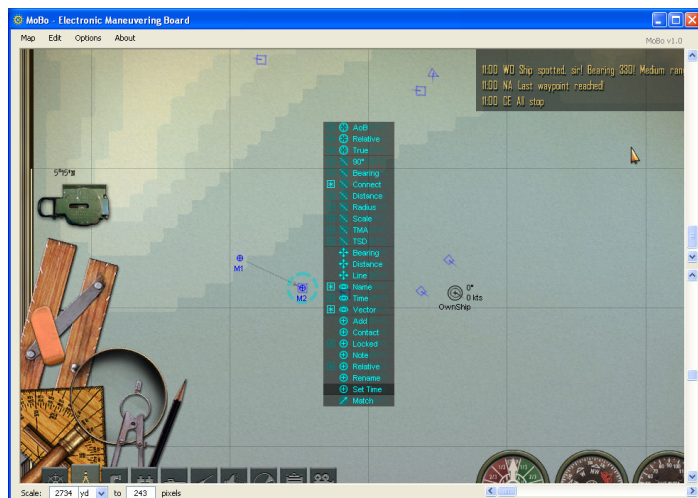
Here we've observed the movement of a unit over 5 minutes. M1 marks an observation of the unit at 1430 hours and M2 is the same unit at 1435.

(see next page for step-by-step instructions)

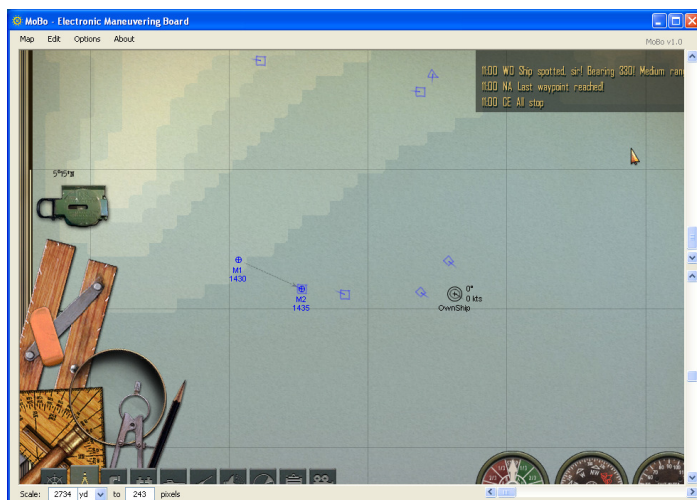




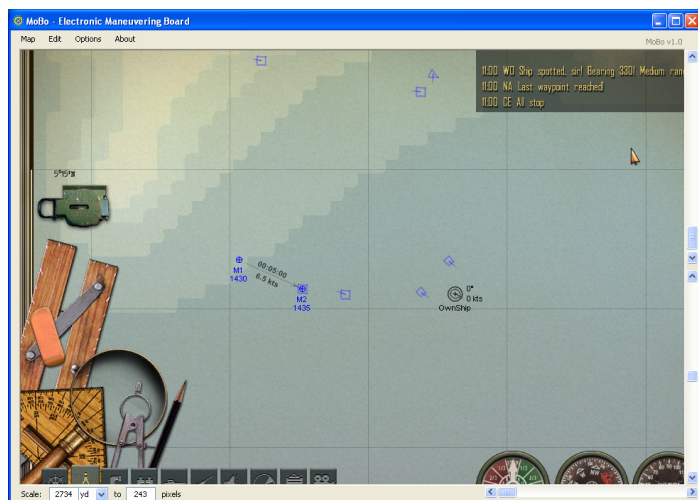
Step 1: Right-click the M2 node and select “Connect” choose unit M1 to connect to.



Step 2: Right-click the M2 node again and select “Set Time”



Step 3: Enter time for unit M2 as 1435. Move cursor over unit M1 and press T to enter 1430 as the time for M1.

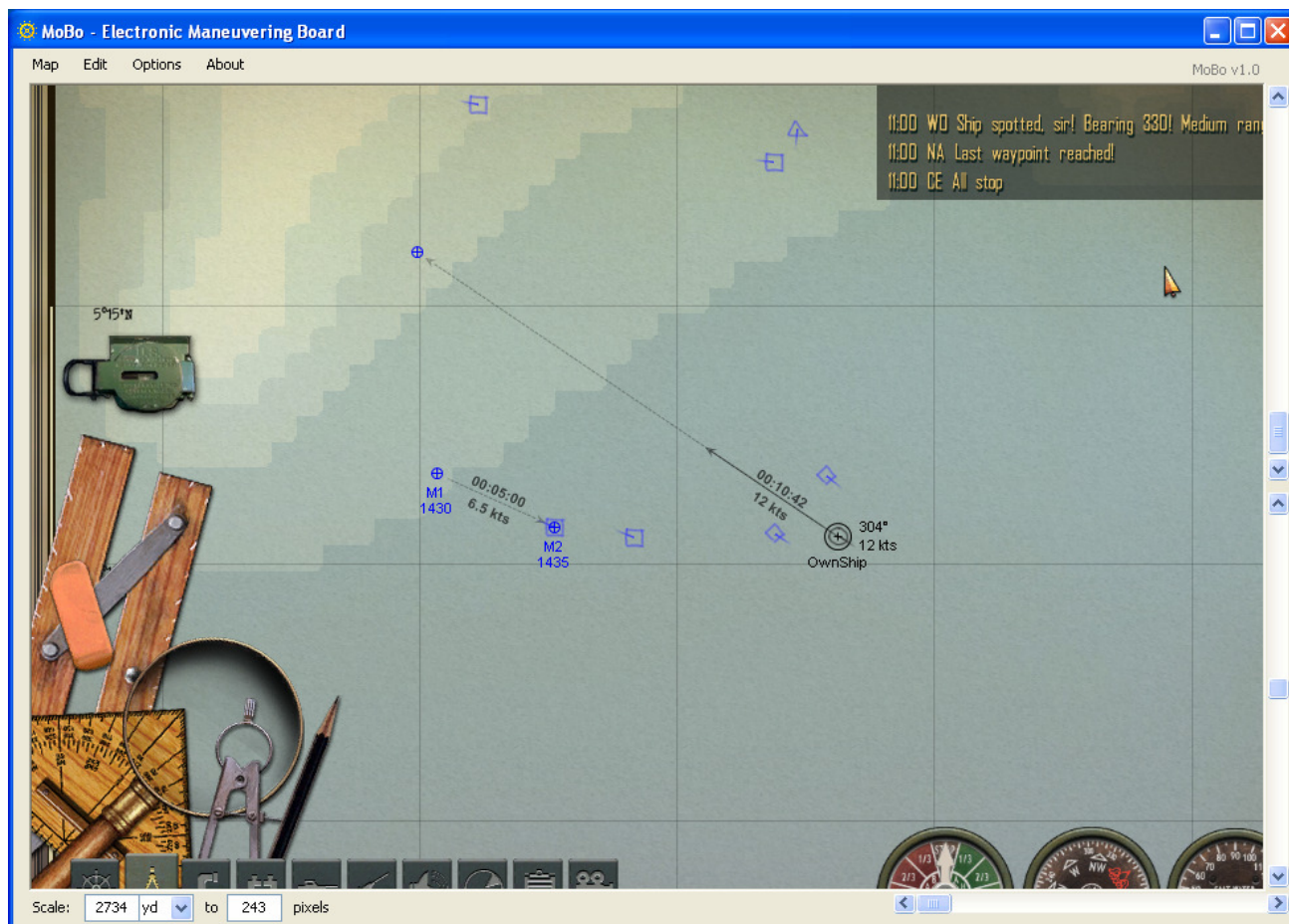


Step 3: With times entered for each observation, move cursor over unit M2 and press the Y key to display the TSD calculation for the line.

At the end of Step 3 we get a solution for the speed as 6.5 knots. Notice the elapsed time is also displayed as “00:05:00”. That’s the correct lapsed time in hh:mm:ss between observation M1 at 1430 and M2 at 1435.

MoBo understands lots of different formats for time entry. So don’t worry about if you enter “14:30” or 1430 or 05... chances are MoBo will understand what you meant. You’ll know MoBo understands because it will always display the elapsed time in “hh:mm:ss” format and you’ll be able to see if your time entry was properly interpreted.





MoBo understands that if times are given and speed is unknown the TSD calculation will solve for speed. If speed is known, MoBo returns the time it would take to traverse the distance.

Notice in the example above how the TSD calculation solves for speed if times are provided. However, if I plot another node on the map and connect it to OwnShip and set OwnShip speed to 12 knots, the TSD calculation tells you how long it will take to traverse the distance at 12 knots (10 minutes and 42 seconds).



Advanced Plotting

Converting Relative to True Motion

So far, in our simple plotting examples, we have been dealing with MoBo almost exclusively as a DRT (Dead Reckoning Tracker). We have been able to do this because we know the exact positions of targets on our map and can plot exactly where they are.

Let's assume for a moment OwnShip is a submarine. Our sub is submerged at 250 feet and not moving. We're listening carefully to a target that is moving somewhere nearby. From the bearings taken at regular intervals, we are able to determine the course of the target using our TMA tools. This all works fine, if our sub is not moving!

Now let's assume the sub is cruising at 250 feet and doing 5 knots. We can still use TMA tools to solve for a heading, but the heading is now skewed by the speed and direction of our own movement. Essentially we know the "Relative Motion" (RM) of the target with respect to our own movement, but we don't know the target's "True Motion" or "True Course" (TC).

To determine the True Course of a target when you know its Relative Motion you need to perform Vector Addition.

Vector Addition

So we're in a submarine cruising along at 250 feet and doing 5 knots. We've made several observations of a target and we think we have a solution for the target speed, distance, and heading. Now we plot the intercept and set course to meet up with that target.

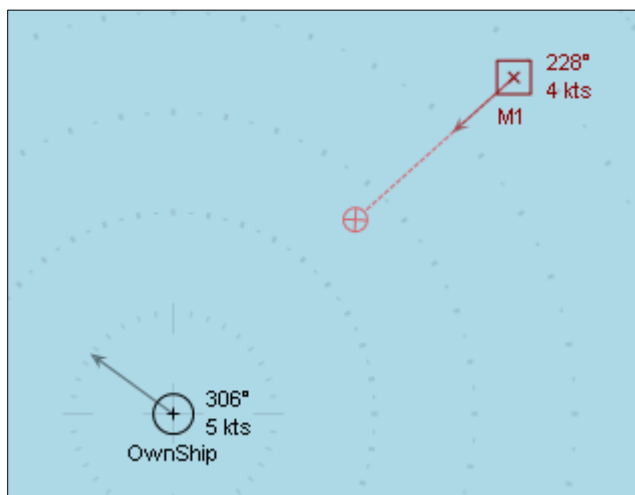
As we approach, we come to periscope depth and take a peek... nothing in sight... oh wait a minute... there it is, but the target is already well passed us on the port side and moving faster than 4 knots! We'll never catch em submerged.

Did the target detect us and change its course and speed?

What happened?

In fact, the target had no idea we were lurking nearby and never made any course or speed adjustments; however...

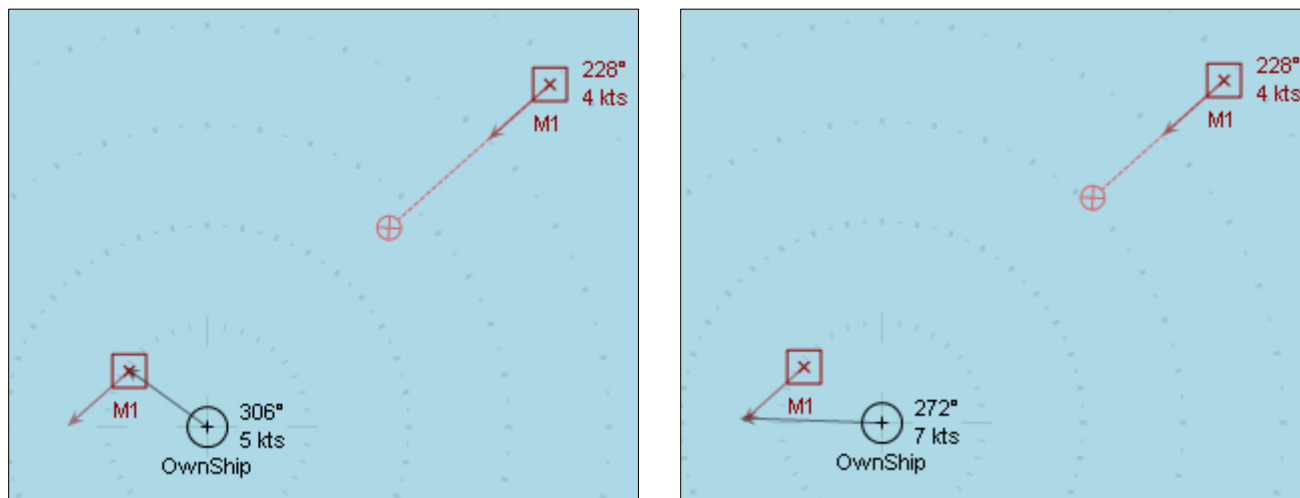
We failed to take into consideration our own movement!



The relative motion of M1 is headed straight for us. If I set course for 90°R, we should run smack into em! ...or will we?

Our solution for the relative motion for the target would've been fine if our sub was sitting completely still. However, since we were moving when we made our observations we must adjust the relative motion of the M1 contact by adding our own vector to the M1 vector.

Vector addition is very simple, and on a real moboard it would look something like this:



If you draw a new vector from the center of OwnShip to the terminal point of the M1 vector, that new vector is the vector addition result and represents the true course and speed vector for M1. I took a shortcut in the process by simply readjusting OwnShip's vector to terminate in the same spot as the M1 vector. I now see the result of the vector addition as 272° and 7 knots, which represents the true course and speed of the M1 contact.

Phew... That's a lot of tedious work! In the process of plotting all this you can quickly become overwhelmed. Now that you understand what's going on with vector addition, how about if we let MoBo do those calculations for us! To do that, we'll plot a new type of unit that we haven't seen before, the Add-type unit.

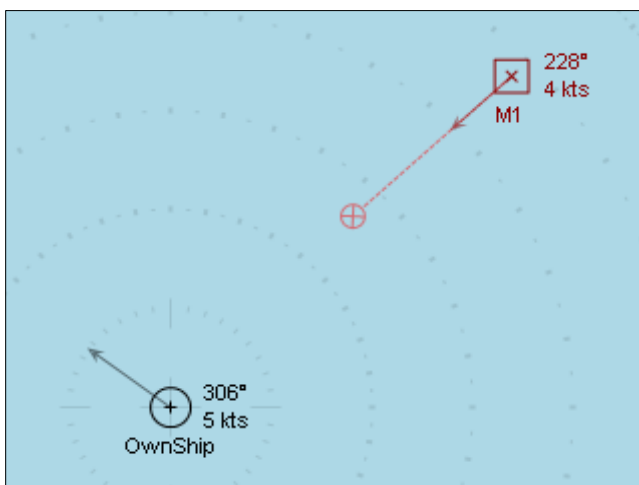
The Add unit appears as a green hexagon by default and is designed to *look at* two units, add their vectors together, and display the result. We can use the Add-type unit directly in our plotting or just leave it off to the side as a calculator and create other units based on the results of our vector addition.

Reposition the M1 vector at the terminal point of the OwnShip vector.

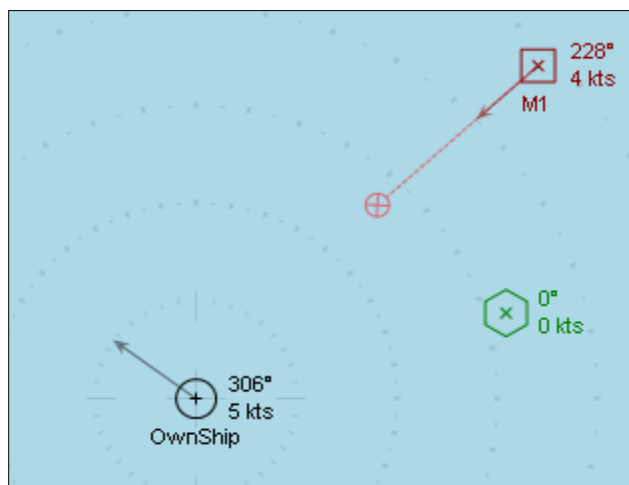
The terminal point of the repositioned M1 vector is the vector addition result and can be used to find the true motion vector.

(see next page for step-by-step instructions)

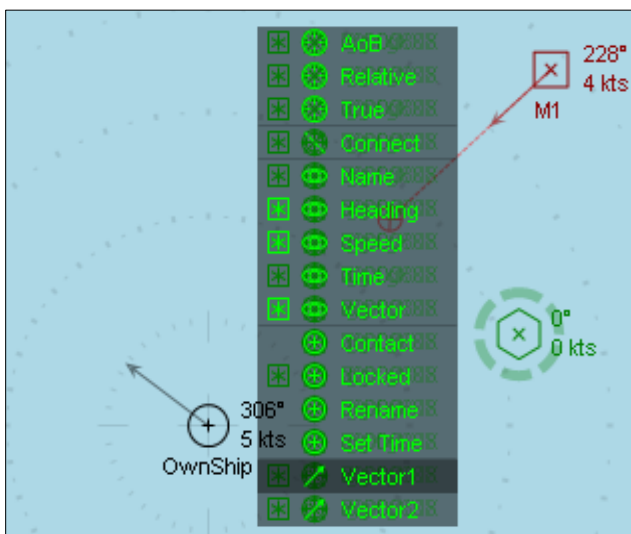
So back to our original problem...



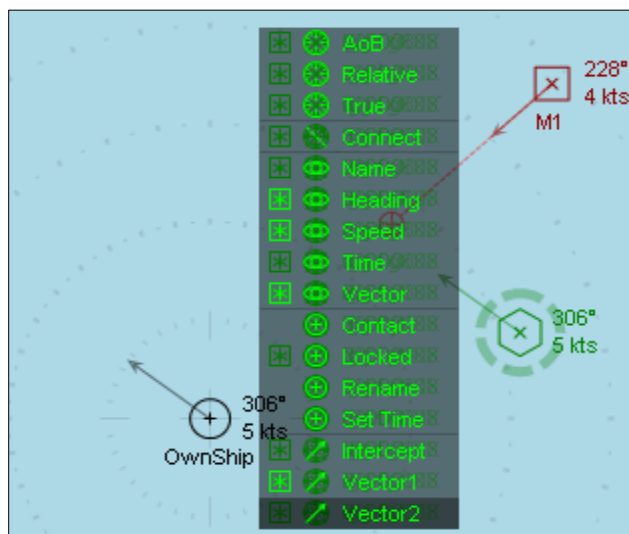
Challenge: Use vector addition to convert relative motion of M1 to True Motion



Step 1: Move cursor to a nearby location and press the keyboard “+” key to plot an Add-type unit



Step 2: Right-click the Add unit and select Vector1, select OwnShip as the Vector1 reference.



Step 3: Right-click the Add unit and select Vector2, select M1 as the Vector2 reference.

At the end of Step 3 you should see that the Add unit has plotted the vectors of OwnShip and M1 and calculated the new vector for true course and speed. The result of 272° and 7 knots is exactly the same as our result from doing it using the manual approach, but our workload is greatly decreased.

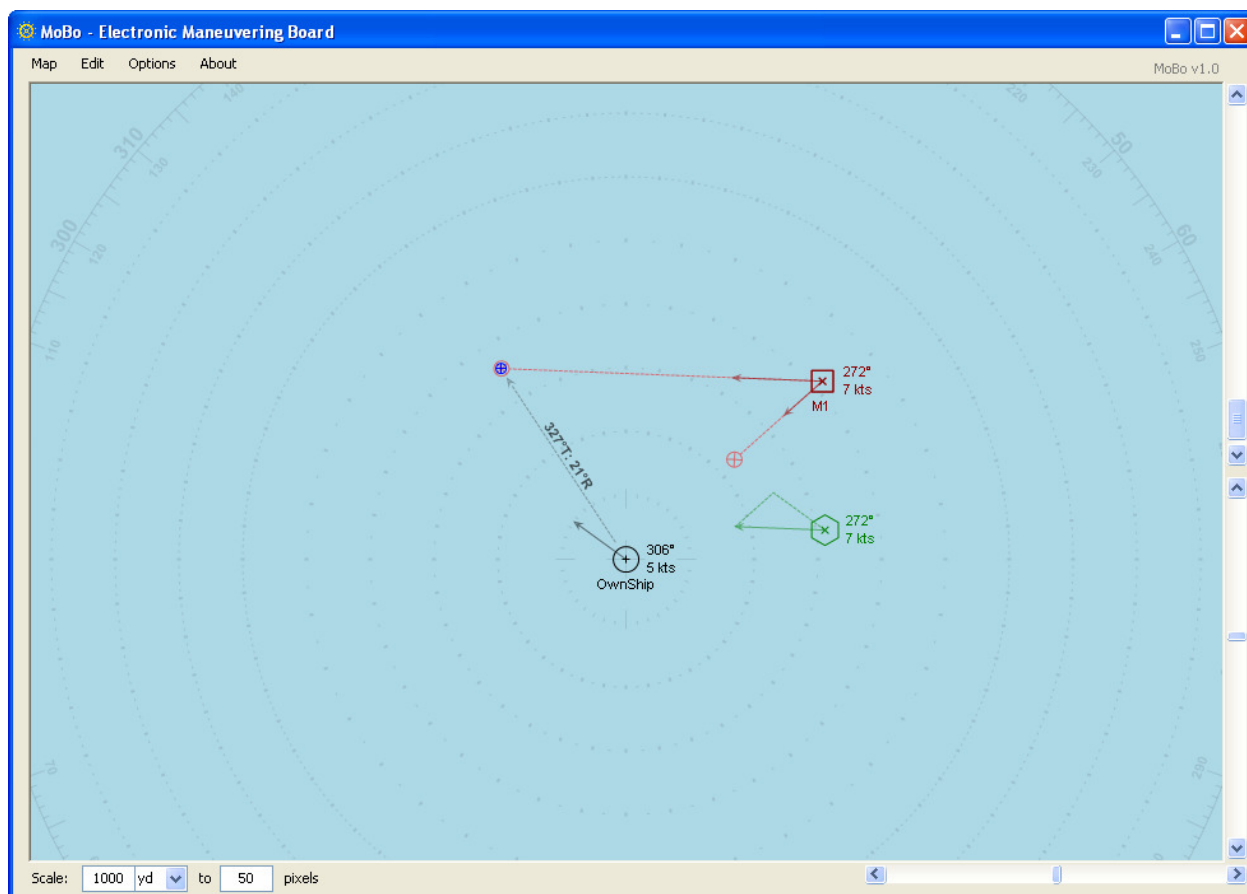


Vector Addition Result

(continued on next page)



So now if I use the results of the vector addition to create a new contact, and overlay the new contact on the original M1 unit...



...I can see the intercept for the true course and speed of M1 is FAR different than that relative motion intercept that I previously plotted. I had an intercept for a unit doing half the speed and heading straight for me, when in reality they were headed almost due West at 7 knots!

Well no wonder we missed em!

Let's not make that mistake again...

In my example above I also could've just displayed the intercept for the Add unit and positioned it atop the M1 contact. Instead I opted to add a contact (C-key) then I used the vector "Match" tool to set the new contact vector equal to the Add unit vector.

There's a lot of room for creativity in your problem solving. Don't think that just because my examples are done a certain way that it's the ONLY way!





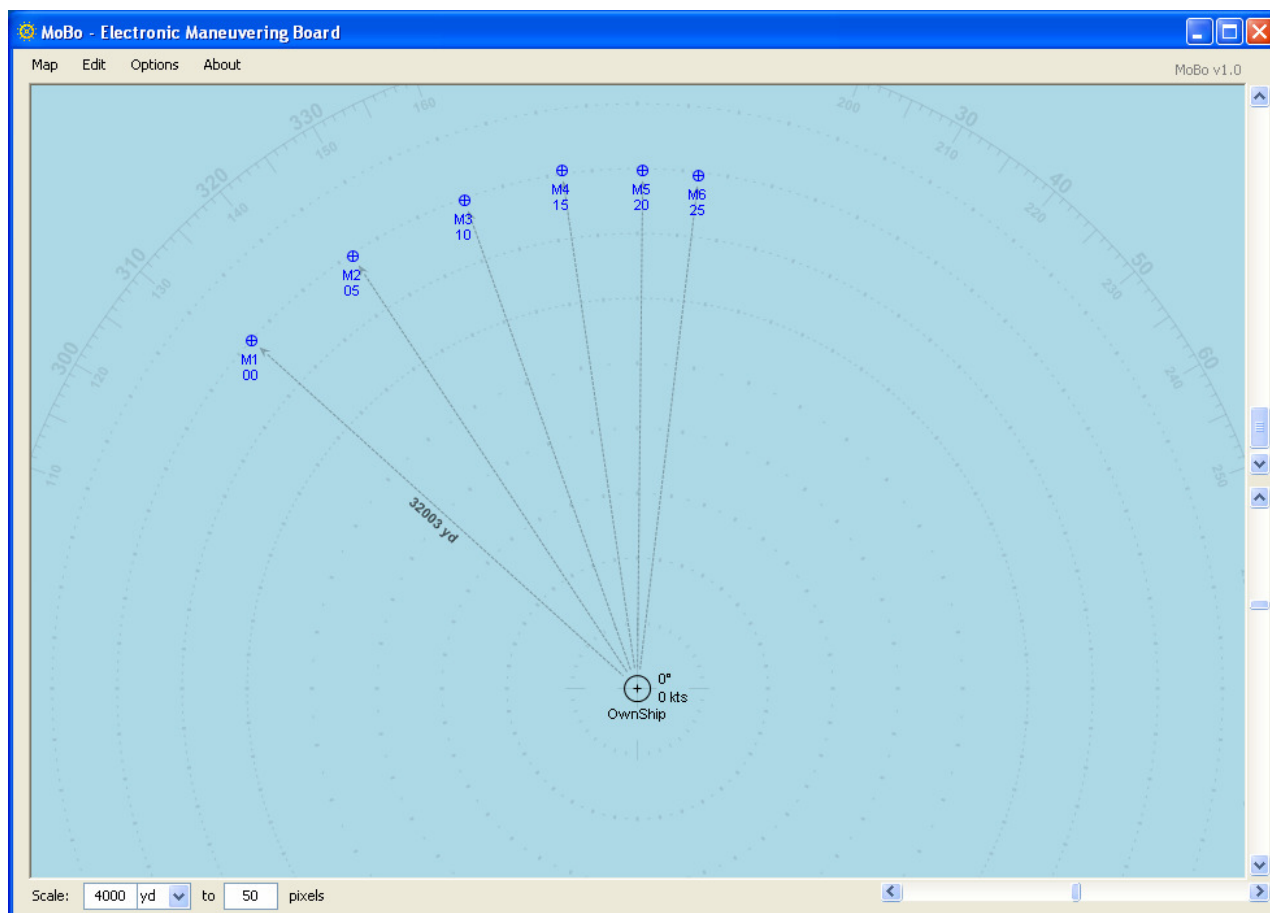
Target Motion Analysis (TMA)

In previous examples we talked about how relative motion needs to be converted to true motion and in those examples I alluded to this magical tool called “TMA” that will somehow tell us all about the relative motion of the contact, and maybe even the exact location. Better yet, we can get all this information while cruising at 250 feet and passively listening.

Sounds great! What is it? How does it work?

Let’s first take a simplified approach and assume our sub is submerged but stationary. Our hydrophone operator is ordered to follow the nearest contact. He reports the bearings of a slow moving target practically every minute. We’re going to plot the bearing he reports at even 5 minute intervals for the next 25 minutes. That should give us 6 bearings on the target assuming we also record the bearing at time zero.

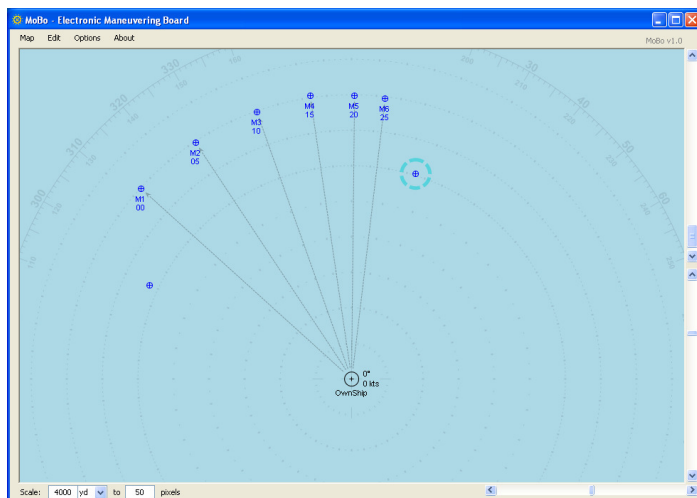
So we end up with a plot that might look something like this:



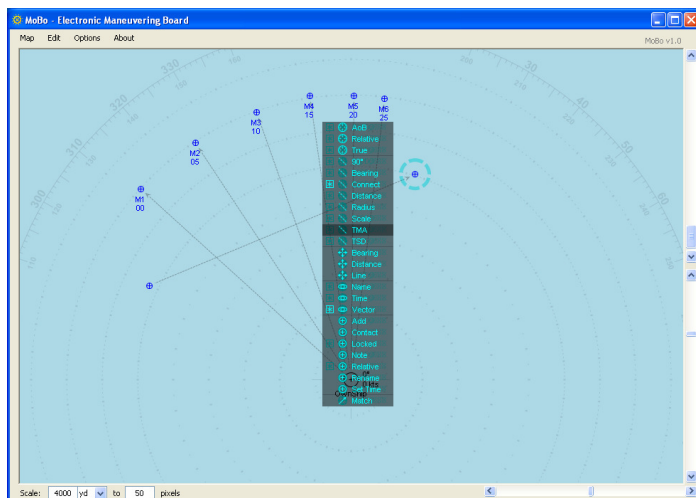
The idea behind the TMA tool is that if we make an assumption that the target is moving at a constant speed and heading, it is also correct to assume that the target will cover the same distance at each 5 minute interval. To figure out the target heading, all we need to do is figure out what line bisects these bearings such that each bisected segment of the line is equal in length.

That’s where the TMA line tool comes into play...

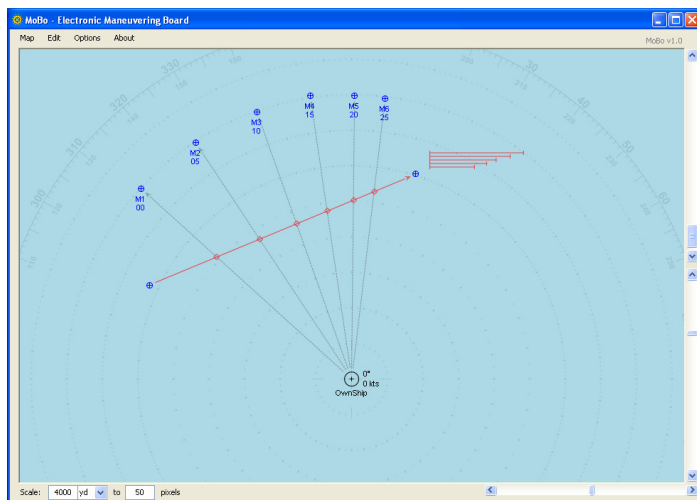




Step 1: We'll start by just adding two nodes; one on either side of our bearing line observations. We select the node on the right and connect it to the node on the left.



Step 2: We should now see a connection with an arrow pointing toward the node on the right. Next, right-click the node on the right and select "TMA" from the list of tools.



Step 3: The connecting line now appears as a solid red TMA line. On the line you'll see diamonds where the line is bisected by the bearing lines.

make those stacked lines all the same size. We can move the TMA line by moving one or both of its nodes.

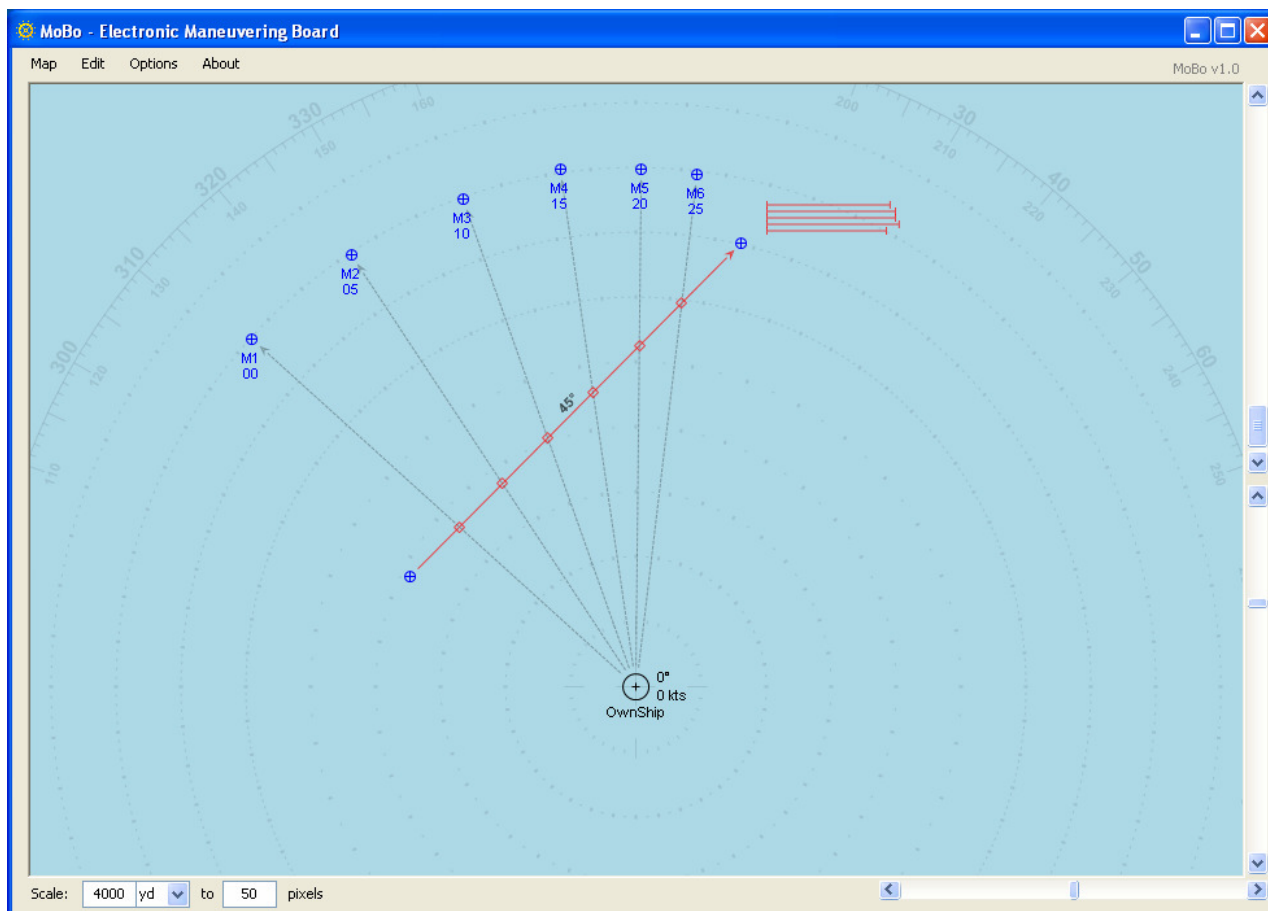
We know just from the gross left to right orientation of our bearings that the target's heading is in a general Easterly direction. We don't really know yet whether it's north or southeasterly, but we'll find out!

We need to adjust the position of the TMA line such that the distances between the diamonds on the red line are equal. Thankfully, we don't have to eyeball it!

At the terminal end of the TMA line you'll see a stack of lines. The stacked lines represent the distance between each diamond x2. In this case there are 6 bearing observations resulting in 5 line segments; the more bearings the more segments. All we need to do is adjust the position of the TMA line in such a way as to

(continued on next page)





After playing with the TMA line for a bit, I settle on a course that appears to have all the stacked lines close to the same size. Then I right-click the TMA node (the node the TMA line points to) and select “Bearing” from the line tools to see that my TMA line suggests a 45° course for the target.

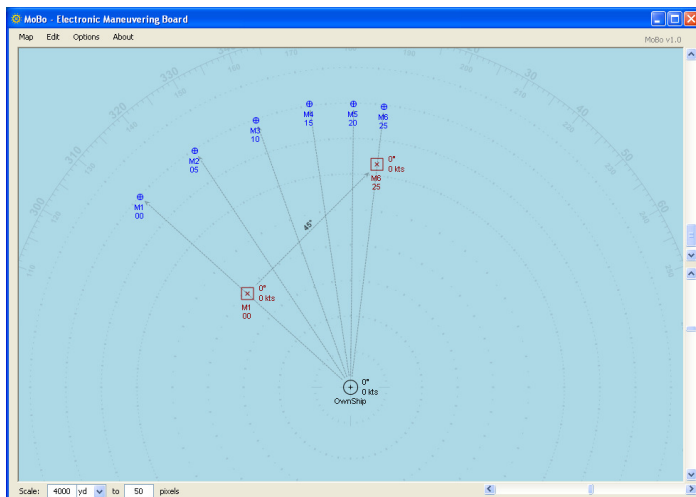
It’s important to keep in mind that all I have at this point is a heading for the target. I don’t know the speed or distance. In fact, I could move that TMA line closer or farther away and as long as the 45° heading for the line is unchanged there would be an infinite number of parallel line solutions all with 45° headings.

But, we still have a few tricks up our sleeve...

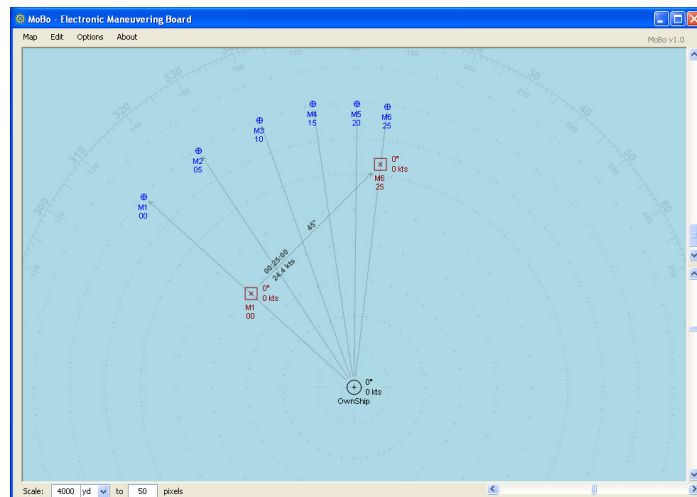
There are infinite solutions for distance depending on the speed of the target; but we know the target is not capable of infinite speed. If we have an idea of the speed, we can make an educated guess and just say, “*I think the target is moving at about 6-8 knots*” and find a range of possible distances.

(continued on next page)



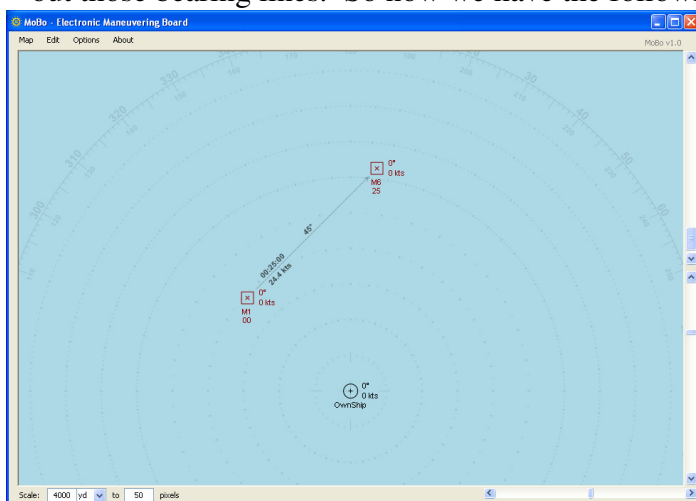


Step 1: Since I know the course is 45° I can get rid of the TMA line and plot two contacts for observation M1 and M6. I connect the contacts and make sure the bearing is 45° and each contact is on the appropriate bearing line.

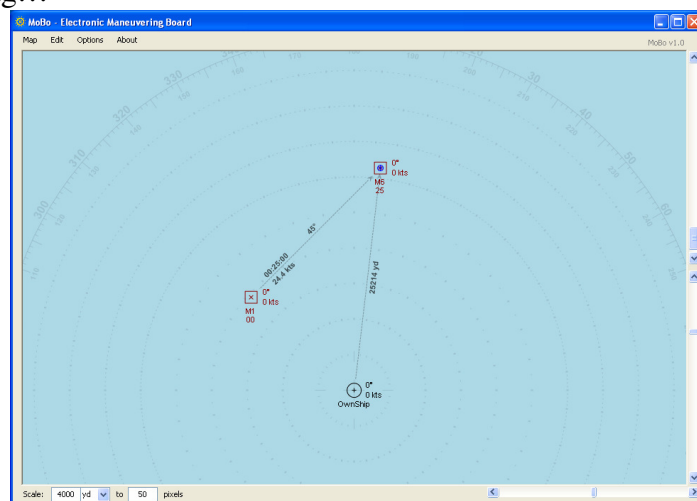


Step 2: After setting the time for contact M1 as 00 and M6 as 25, I use the TSD tool to find out that the speed solution for this distance is 24.4 knots.

Let's assume I want to find a solution for 7 knots. I could move around the line until I find the solution. However, adjusting the position of the two contacts does get a little bit tedious. Instead we'll manually adjust the scale until our speed solution is set for 7 knots. At this point, if I wanted to, I could even clear out those bearing lines. So now we have the following...



Step 3: My solution plot for target heading of 45° with TSD display showing a 24.4 knot speed solution.



Step 4: I add a node to the display and connect it to OwnShip. I display the distance on the line and carefully position the node on top of the M6 contact.

At the end of Step 4, I have a distance solution for the last known target position (M6). The distance reads as 25,214 yards; but that's the distance assuming the target is moving at 24.4 knots. I want to know the distance for the target if it's moving at 7 knots. After a few trial and error attempts I find that if I set my map scale to 1150 yards, I get a speed solution for 7 knots and the distance is displayed as 7249 yards. Plugging in a few more numbers I can give myself a range to work with:

- 6 knots = 6178 yards
- 8 knots = 8258 yards



Not too bad, I've given myself a range to work with based on my own speed guesstimates. Usually that's good enough to find the bad guys.

But what if I didn't want to guess at the speed?

Is there a way to pinpoint the target without making guesses?

Advanced TMA

The answer to that last question is **Yes...**

However, this is the turning point at which we may be spending more time plotting than actually playing the game. For the diehard sim fans who build their own slide-rule gadgets to figure out attack angles... **Hey, no problem!**

Casual users, on the other hand, might checkout at this stage. If you want to keep it simple, just stop your sub, do the TMA stuff in the previous example, make some guesses at speed and your probably close enough.

For those who really like this sorta stuff, I'll give you the theory and let you figure out how to plot it. Personally... Yes. I have done this, and it works.

Instead of guessing at the target speed, you can use the TMA tool to predict future target bearings along the projected TMA heading. Assume for a moment that you've plotted 6 bearing lines. You develop a TMA solution that fits the bearings. But rather than stop there, you plot 3 more bearings for times 5, 10, and 15 minutes into the future.

Since we already know the bearings for 15 minutes into the future from our current position (assuming we're not moving), there's no point in hanging around. We'll shoot off in a different direction for 15 minutes and then get a new bearing on the target from a different position at the 15 minute mark.

The intersection of your predicted bearing for 15 minutes and your new position bearing at the same 15 minute mark should pinpoint the target. You're effectively triangulating on the target without the aid of a second ship!

Something to keep in mind; I have used this triangulation technique and I know it works fine if the sub is stationary. You may confound things if you try to do it while your sub is in motion. Also in my previous *simple* TMA example I mentioned that the sub was not moving. If the sub was moving in the previous example what do you think I would need to do in order to convert the Relative Motion solution to True Course and Speed? Hmmm...





General Moboard Terms and Definitions

Relative Movement (RM)

All objects move relative to something else. Relative movement is motion measured with respect to a selected object (which may or may not have movement). The mo-board is used to find actual course and speed required to bring about a desired change in relative position to other ships being monitored.

Direction of Relative Motion (DRM)

The direction of relative motion in degrees true; or a line drawn between M1 and M2 with an arrow pointing in the direction.

Measurement of Relative Motion (MRM)

The distance “M” contact traveled relative to our ship.

Speed of Relative Motion (SRM)

The speed of the contact along the relative movement line; or the contact’s speed relative to our ship. Use distance and time to solve for speed.

Closest Point of Approach (CPA)

The closest point at which one ship will pass another; includes bearing, range, and time of occurrence. Expressed as a perpendicular line from your ship to the contact.

$$\text{CPA} = \text{DRM} \pm 90^\circ$$

M Observations

M1 = first observation, M2 = next observation, etc.





Time Speed Distance (TSD)

Calculations involving the formula for time-speed-distance calculations: $T \cdot S = D$, given any two variables of the TSD formula, you can solve for the third using a Nomogram, a calculator, or your head. The connection lines in MoBo will also do it for you!

Nomogram

A 3-scale Nomogram allows you to solve for a value on one of the scales if two values on the other scales are known. You simply draw a straight line between the scales that connect the two known values and where that line intersects the other scale is the solution for the unknown value. The scales can be expressed in either linear or log format. Log scales are probably the more useful and common. A Nomogram for TSD solutions usually appears at the bottom of a standard Mo-Board. MoBo doesn't display or need a Nomogram... the lines have a built-in TSD solution engine.

Relative Bearing (RB)

Measured in degrees from OwnShip's heading clockwise to the observed object. The course or heading of OwnShip will always affect relative bearing.

True Bearing (TB)

Measured clockwise from true north in degrees to line of sight. The heading of our ship does not affect true bearing.

$$TB = \text{OwnShip Heading} + RB$$

Target Angle or Angle on Bow (AoB)

The observed angle between a contact's heading or DRM and the line of bearing from OwnShip.

$$AoB = (\text{Contact TB} \pm 180^\circ) - \text{Contact True Course}$$





A Challenge...

Your sub is on course 160°T speed 10 kts. You sight a contact bearing 280°T, distance 16,500 yards at 1340. At 1405 the contact bears 330°T, range 14,000 yards. You're in enemy waters and your crew awaits your orders. You're not sure if this contact is a small merchant, or a destroyer. Your sound man is still recovering from the last DC episode and he can't hear the screws. Your XO asks, "Cap'n do you wish to intercept or evade?"

Your job is to find the following: DRM, MRM, SRM, and CPA

One member of the MoBo Development Team who you may recognize on the forums as "**Lurker_HLB3**" has a very interesting background:

Retired US Navy, Chief Operations Specialist (OSC), 23 Years, Served On USS Bausell DD845, USS England DLG22, USS Sterrett CG31, and USS Leahy CG16. Qualified as a Tactical Action Officer (TAO), Air Intercept Controller (AIC/AICS), Anti-Submarine Air Controller (ASAC).

I am currently a Software Troubleshooter / Systems Integrator that works on various military C4I Systems for DOD. I have used my Systems Integrator skills to merge various mod / super mods on my custom SH3 install, have also done the same for SH4.

Lurker says the answer is:

DRM = 45°

MRM = 12,954 yards

SRM = 15.4 kts

CPA = 315°T / 13,548 yards

True Course / Speed = 84° / 14.4 kts

Course to intercept 20° / 17 kts

...but can you do it?





About

The MoBo project was conceived by Aaron T. Blood mid-year 2006 and developed over the course of several months. MoBo v1.0 was first released in June of 2007. MoBo was developed exclusively with Microsoft's Visual Studio Express Edition (a freely available download from Microsoft).

All graphics displayed in MoBo are rendered at runtime using a vector-based graphics engine to manipulate the Visual Studio GDI (Graphics Development Interface). The internal design of the application features a retained mode graphics system that allows users to interact with graphic objects as data storage devices.

The language used within MS Visual Studio was VB.net 2005, and like all Visual Studio applications, the 2.0 .Net Framework is required to be installed on host machines. The .Net Framework comes pre-installed on Vista machines and is also a freely available download from Microsoft.

Many, many, many hours were put into the development, programming, documentation and testing of this program. If you like MoBo, stop by the MoBo forum at www.SubSim.com and let us know what you think. We're always looking for fresh ideas, new plotting examples, and new whiz-wheels or dials tools to add to the MoBo toolset.

MoBo is a free program distributed exclusively through Aaron's website:

www.XL-Logic.com

Commercial reproduction, distribution, bundling, or sale of MoBo without written consent of Aaron T. Blood is prohibited.



