GotABot

User Guide version 3.1.0

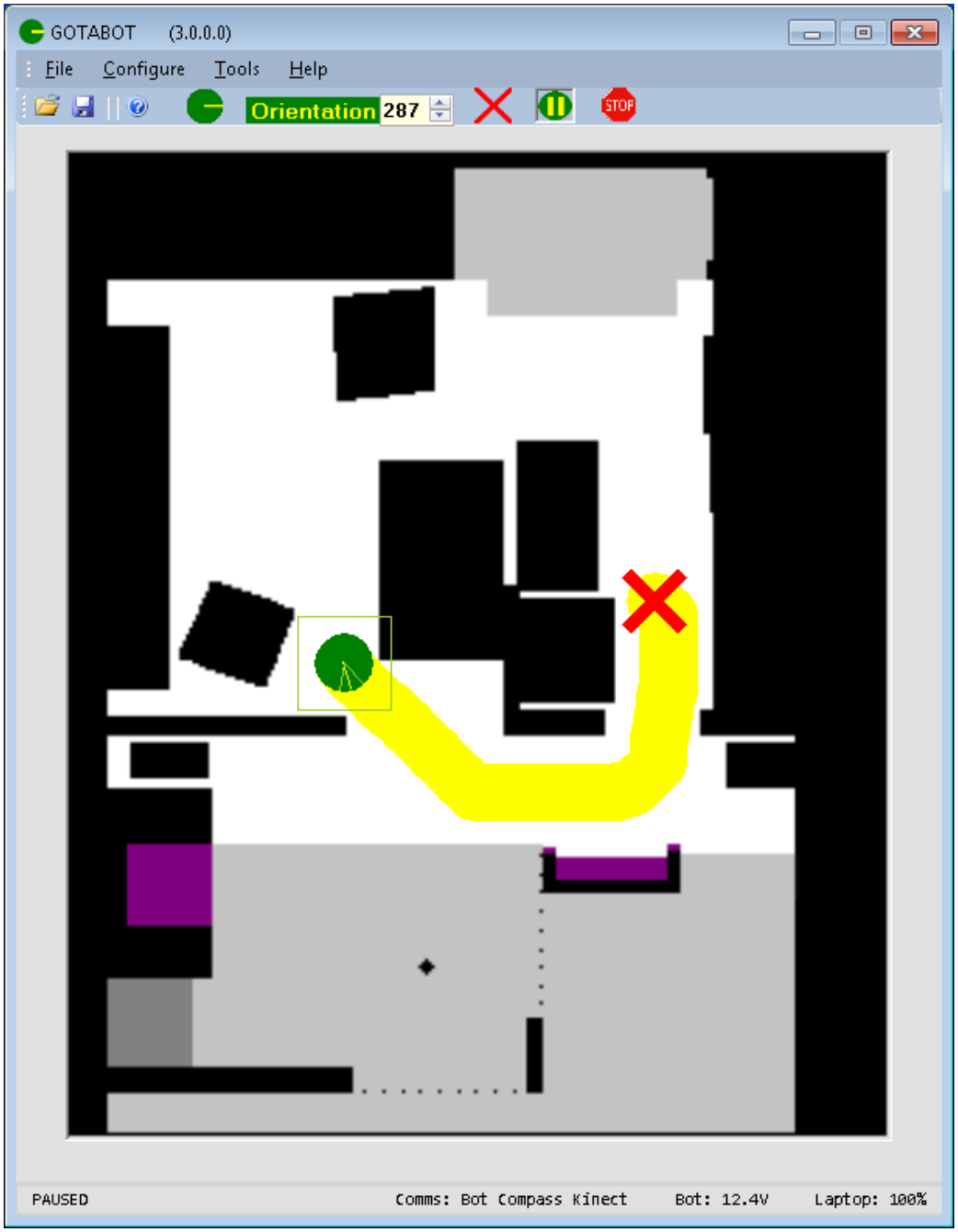


Table of Contents

1. Introduction

So you've got a robot and need some software to make it do stuff. Perhaps GotABot can help with that. With a user created map, it can plan a route and direct your robot to a desired destination. Using only odometry and a camera or Kinect, GotABot will track the robot and correct its position and trajectory so as to stay on course.

GotABot also supports optional enhancements such as bumper switches, compass , Kinect and Range Finders.

Additional information and the latest version of the software may be found on the website: http://gotabot.weebly.com/.

Important Safety Information

Although great effort has been taken to debug GotABot, it is a "Use at your own risk" type program. Please use common sense and follow all safety instructions provided by the robot manufacturer.

2. Getting Started

As an overview, the steps required to operate GotABot are:

1) Install
2) Create a map
3) Capture some Land Marks (if using a visual recognizer)
4) Enjoy

Place the robot on the map by:
1) Click this icon
2) Then click the map at the desired location

Make the robot go to a destination by:
1) Click this icon
2) Then click the map at the desired location

Pause the robot's movement. Re-click to resume movement.

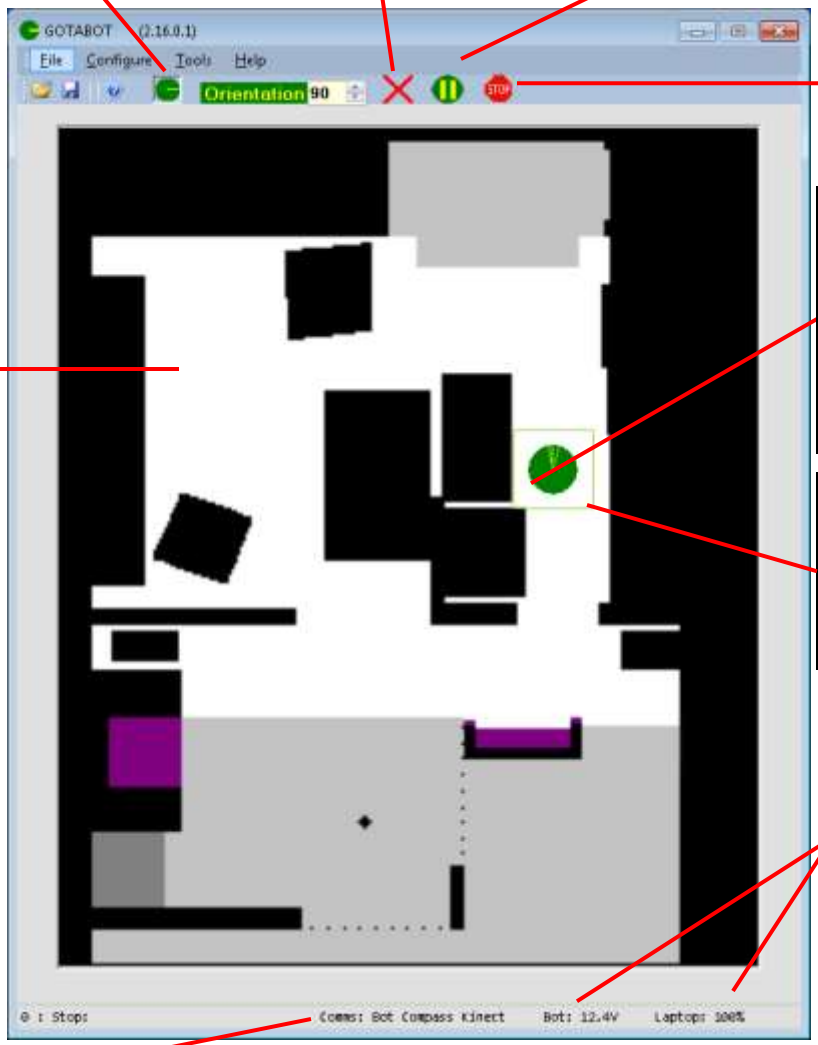
This is the map area.

Black indicates solid objects and areas where the robot can't go.

White indicates areas where the robot is free to move.

Grey indicates areas where the robot can try to go if necessary, but probably shouldn't.

Purple indicates areas which Range Finders or Kinect sensors indicate as empty but which are in fact solid objects or areas where the robot should not go.



Stop the robot's movement.

This is the robot. The bright yellow line indicates the direction, or orientation, it is facing.

Yellow-Green lines indicate the uncertainty region of the robot's pose.

Robot's battery status.

Communications established with these devices.

Figure 2-1 Overview

System Configuration and Installation of GotABot

System Configuration

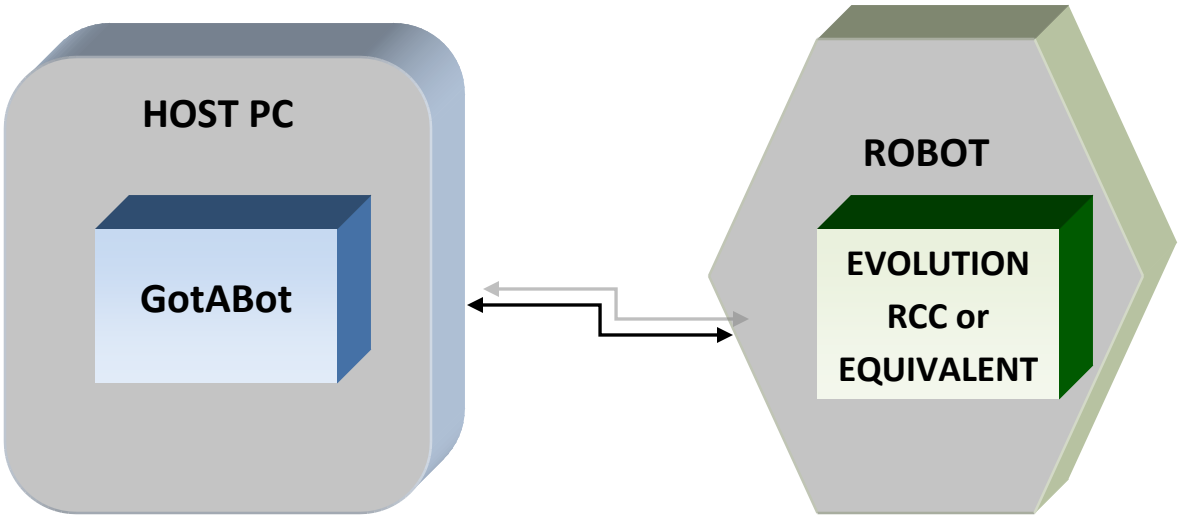
Before installing GotABot, one should first decide how to configure one's robotic system. There are a number of factors to consider such as:

1) The robot which is to be controlled
2) Will GotABot act as a standalone controller or will it be controlled by a higher level program.
3) Will the robot use a Kinect sensor or a webcam to determine its position.

These and other considerations will be discussed in subsequent chapters.

Simple System Configuration

The simplest system configuration is as shown below. The Host PC may be separate from the robot and communicate to it via a wireless link or it might be on or part of the robot itself. This is the easiest configuration and should, if one has an ER1 or equivalent robot, be used first if for no other reason than to familiarize oneself with GotABot operation.



Other System Configurations

Although the previous 'Simple System Configuration' works well, there are other configurations which may be more optimal for some applications:

1) If you wish to add additional behaviors, customize GotABot's operation, or allow a separate program to use all or portions of GotABot's functionality then see the chapter "API".
2) If your robot is not an ER1 then it might be controllable through the RoboRealm interface. More details are contained in a later chapter, "GotABot Control of a RoboRealm Robot".
3) If your robot does not have image processing capabilities or you wish to have faster Image Recognition capabilities or perhaps you need to reduce power consumption, then see the chapter "Local Image Recognition"
4) A Kinect sensor is easier to setup and can provide better performance than image recognition of Landmarks. For even more reliable results, both Kinect and Landmark recognition may be used simultaneously.

Installation

For most applications, GotABot is installed on a host computer which can communicate with your robot. The recommended minimum requirements of the computer are:

- Pentium® III class, Intel® Celeron®, or AMD processor, or better - 1000 MHz or faster
- Windows XP or better
- 256 MB RAM
- 50 MB of free Hard Disk Space

GotABot is a portable application and it may be unzipped or copied to any convenient directory.

3. Creating a Map

Maps are BMP files representing the environment the robot will operate in. They can be created by most graphics programs including Microsoft Paint.

- **Black** areas represent walls, furniture and "out of bound" areas.
- **Grey** areas represent areas which may be occupied or areas where the robot should not travel through unless it can find no other route. Lighter shades of grey indicate a less prohibited route than darker shades of grey.
- **White** areas represent open space through which the robot is free to move.
- **Purple** indicates areas which Range Finders or Kinect sensors indicate as empty but which are solid objects or areas where the Robot should not go.

Creating Maps With Microsoft Paint

1) *Measure the actual area you want mapped – be accurate.*
2) *Decide on a scale. The scale can be in units of inches or cm and each pixel may represent multiple inches or cm. With the tight spaces my robot would navigate in, I decided on a scale of 1 pixel = 1 inch.*
3) *Start Paint (Start -> Run..->'mspaint')*
4) *Set the Map Size (Press Ctrl+E then set the map width and height)*
5) *Enlarge the map (Press Ctrl+PgDn)*
6) *Show a grid (Press CTRL+G)*
7) *Using the grid and the mouse position indicator in the bottom-right, draw the map.*
8) *Save the map in the GotABot directory with the filename "GotABot_Map.bmp".*

Scale

The map's scale is set by using GotABot's Tool Bar's Map menu (*Click Configure->Map->Grid Size and Configure->Map->Grid Units*)

Due to a lack of rigorous testing, it is strongly recommended that the user choose a Scale of 1 pixel = 1 inch.

As an example, this is a map that I use:

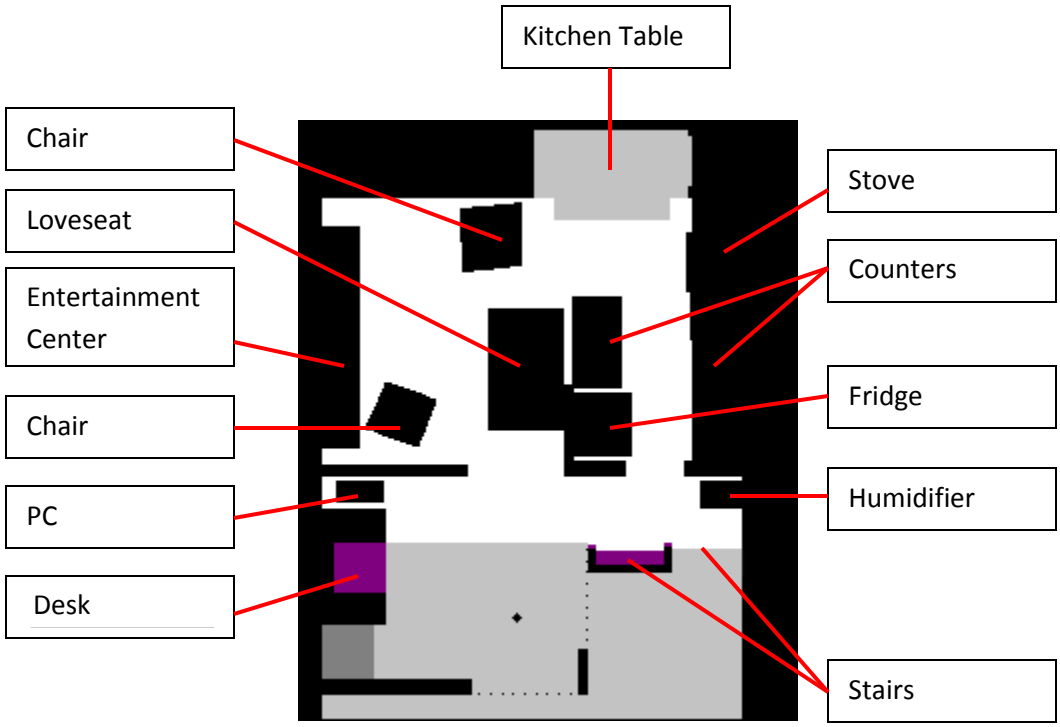


Figure 3-1 Example Map

…but shouldn't the robot make the map?

Well. It can (see the chapter on SLAM), but unfortunately, making a high accuracy map is a decidedly non trivial problem. Robot generated maps are especially problematic when the only measuring instruments are a cheap webcam, dodgy odometry and low cost range sensors. All in all, the best maps are drawn by old fashioned humans.

4. Kinect Configuration

GotABot provides an interface to a Microsoft Kinect sensor which allows for excellent localization and collision prevention.

Localization using the Kinect is in many ways superior to using visual Land Marks. It is easier to use, more accurate, localizes faster and has a microphone array for enhanced speech recognition. Highly recommended if your Robot can support a Kinect and has a laptop.

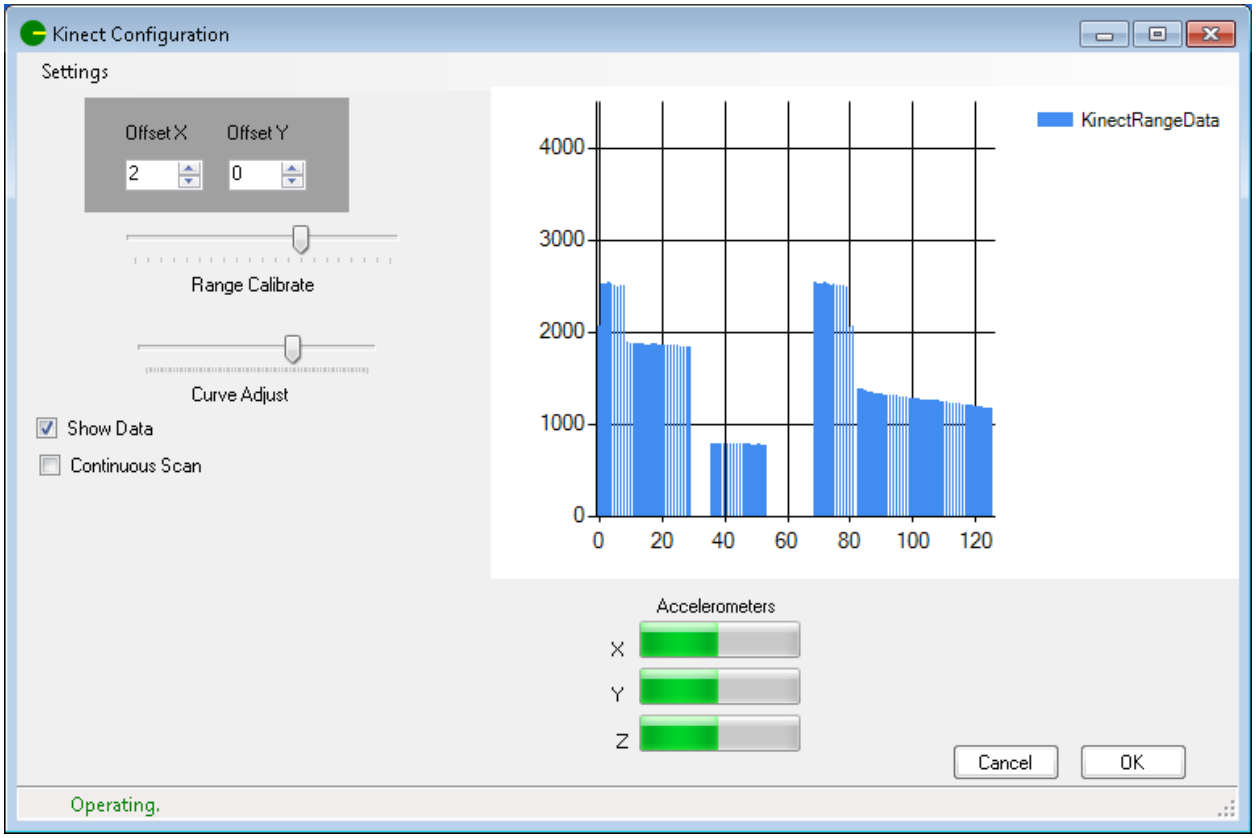


Figure 4-1 Kinect Configuration Window

Configuration

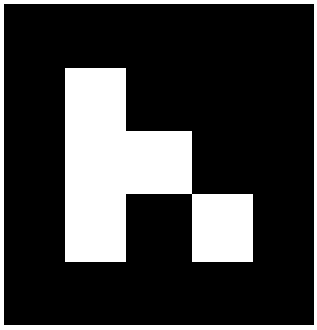
Configure Kinect operation by opening the Kinect Configuration window (click *Configure->Kinect*).

The Offset fields represent the Kinect's position on the robot relative to its center. The offset x and y coordinate system are the same as in the Ranger offset configuration. (Click *Configure->Bot->Range Finders* to see a diagram)

Adjust the Calibration sliders to correct any errors in the Kinect range measurements.

5. Landmark Configuration

In addition to the Kinect interface, Landmark recognition may be used for determining the robot's position. Landmarks are user selected images that the robot can recognize and measure range and direction to. Knowing the Land Mark's location and the estimated robot's position and orientation, GotABot uses simple geometry to determine where the robot is located.



5-1 Example of Glyph or Fiducial

For detection of Land Marks, GotABot provides interfaces to four image recognition programs. Any or all may be selected (Click *Configure->Recognizer* to select):

1) **GotAGlyph**

GotAGlyph is a Glyph recognition program provided with the GotABot package. Glyphs are a grid pattern of white squares on a black background. The high contrast between the white and black coupled with the square shapes and 2D planar nature yield an easy to identify image. This is the easiest, most convenient recognition program to start with. Unfortunately, it requires significant processing resources.

2) **RoboRealm**

RoboRealm is a commercial program which, among its many capabilities, can perform Fiducial recognition. These fiducials are highly similar to Glyphs.

3) **RCC**

The **recommended** image processor. This application is provided with the Evolution Robotics ER1 robot. Included in this application is a powerful image recognition utility suitable for identifying many types of images.

4) **AVM**

AVM is an optional plug-in for RoboRealm which can be used for image recognition. Difficult to configure the distance determination but useful for recognizing non 2D objects.

Figure 5-2 Example of a Highly Recognizable Land Mark

Establishing a Land Mark is realized by:

1) Select the Landmark
2) Capture an image of it
3) Enter it in the GotABot database

Selection of Land Mark Images

Without a doubt, the selection of good Land Marks is the most crucial component to ensuring adequate localization. A few well chosen Land Marks will be the equal of dozens of randomly chosen images of a typical home environment.

A good Land Mark will be:

1) Unique
2) Easily recognizable by the robot's image processing algorithm
3) Useable over a wide range of distance and angles

In practice, 2D Land Marks such as Fiducials, Glyphs, pictures, posters and playing cards are excellent choices. When first setting up, it would be well advised to temporarily attach a few of these to strategically located spots. Later, they can be replaced or augmented by a series of normal room images.

Tips For Good Land Marks

1) A good Land Mark should contain a number of high contrast features which the robot's algorithm can easily discern. Silhouette, or black and white, images are highly recommended!
2) Land Marks are most robust when they are in a single plane such as a picture, poster or playing card. Using such a Land Mark, the distance from robot to Land Mark can be estimated accurately over a wide range of distances.
3) If Landmark images containing various features at various distances must be used, the valid region over which distance can be accurately measured will be relatively shallow. In such situations, it is best to make multiple Land Mark images, each at a slightly different distance.
4) Land Marks will yield more accurate range information when the orientation of the robot to Land Mark is close to that of the original image. If the Land Mark will be seen over a wide range of angles then create multiple Land Marks by capturing images at different angles.
5) Images which primarily consist of repeating features such as sets of drawers, slatted doors, patterned wallpaper or furniture, railings, etc, do not make good Land Marks.
6) Land Mark images which contain unwanted objects may be edited in Paint. Usually, the desired details may be easily isolated by overwriting everything else with grey rectangles.
7) Good locations for Land Marks include ends of corridors, doorways and any place where careful maneuvering is required to pass through tight spaces.
8) Ideally, the Land Mark should be at the same height as the camera.
9) Landmarks may not be recognized if they are placed against a 'busy' background. In such cases, a blank border around the land mark will improve recognition (i.e. if your landmark is, say, a picture attached to a patterned background then insert a slightly larger blank sheet of paper between the picture and the background)

The Trouble With Land Marks

As a rule of thumb, there should be at least one Land Mark visible at any location the robot is likely to stop. With one Land Mark, the robot's pose (i.e. – its position and orientation) can usually be calculated accurately, but sometimes, especially when the robot's pose is not accurately known, errors will occur and the pose will be *shifted*. Usually, these errors will be corrected with time and further localizations, but sometimes the snowball accelerate downhill, the reported pose drifts ever further from reality and the robot runs into a wall.

The treatment for this error condition is simple – add more Land Marks. Make sure that wherever the robot is likely to look, there is a good Land Mark. If there are two Land Marks visible everywhere then so much the better. Errors introduced by one Land Mark will be corrected by the others.

Accuracy Counts!

When establishing a Land Mark, make sure that the center of recognition is accurately transcribed in the Land Mark Edit form. It is probably necessary and certainly advisable to check where the image recognition software thinks the center is.

Having established where the actual center is, make sure that the range to this point is also correct.

The Edit Land Mark Window

Land Marks are entered through the Edit Land Mark window (*Click Configure-> Landmark*). Remember to save your settings after making change s (*Click File->Save Settings*).

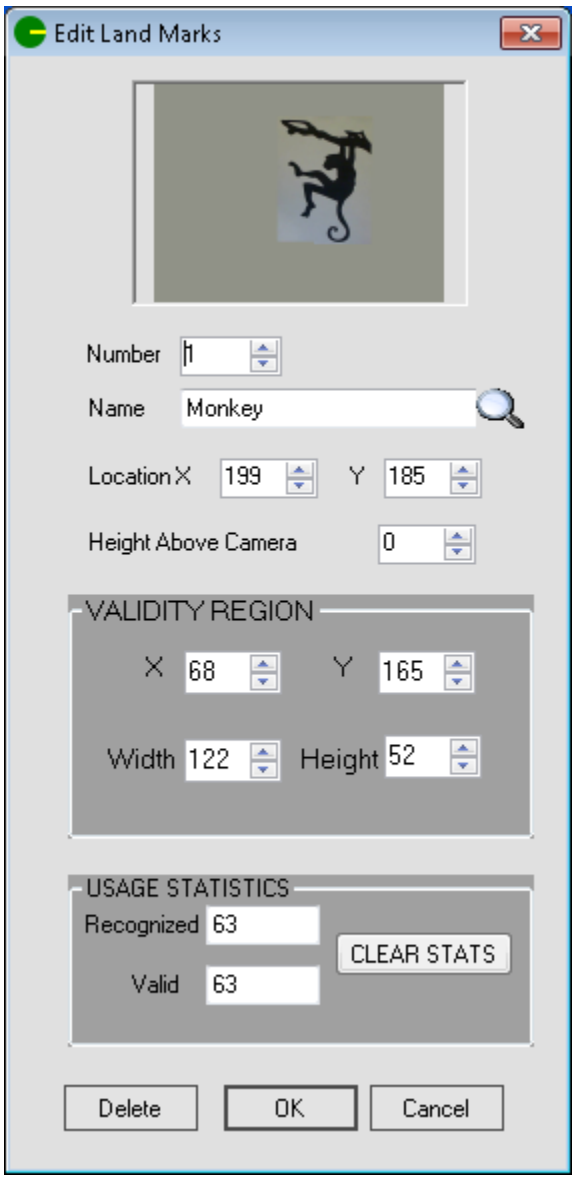


Figure 5-3 Edit Land Mark Form

Land Mark Number

Each Land Mark has a unique number. As the number field is changed, the corresponding Land Mark and its Validity Region are plotted on the map.

Land Mark Name

The Land Mark Name field must contain the exact name reported by the robot.

Search Land Mark Name

Pressing the Magnifying Glass icon beside the Name field allows a search of the Land Mark database by name.

Land Mark Location

This is the location of the center of the Land Mark. It may be entered by:

1) Click on either of the Location X or Y up / down entry box.
2) Click on the map at the location of the Land Mark.

Height Above Camera

This is an optional field, but it can help to calculate a more accurate range to a LandMark when the LandMark is not at camera level. This is typically useful when the camera is near ground level and the LandMark is, say, a picture hung at a much higher eye level.

Land Mark Validity Region

To lessen the chance of harm due to false Land Mark recognitions, a validity region is required for each Land Mark. This region restricts the area and robot orientation within which the recognition will be judged to be valid.

The validity area is a rectangle and is entered by:

1) Click on either of the Validity Region X or Y up / down entry box.
2) On the map, position the mouse at one of the Validity Region's corners, press the left button and draw a rectangle. Release the button.

Land Mark Usage Stats

These values are generated by the GotABot program and provide information on the usefulness of each Land Mark. Low Recognition or Validity numbers may be an indication that the Land Mark information is incorrect or that a better Land Mark is required.

6. Voice Commands

Voice commands are enabled by clicking *Configure->Options->Voice Command Enable*.

Examples of voice commands include:

COMMAND	RESULT
"*GotABot stop*"	Stop all motion
"*GotABot pausemode*"	Pause driving to destination NOTE: Say '*pausemode'* as one word
"*GotABot status*"	Report status
"*GotABot localize*"	Perform a localization operation
"*GotABot recharge*"	Go to dock location and recharge
"*GotABot destination three inches*"	Move forward three inches
"*GotABot destination minus fifteen inches*"	Move backwards fifteen inches
"*GotABot destination one three five degrees*"	Turn 135 degrees left
"*GotABot destination sixty five comma two four proceed*"	Go to map coordinate 65,24
"*GotABot destination right*"	Turn 45 degrees right
"*GotABot batchfile seven proceed*"	Execute the user defined batch file "kbatch7.bat", located in the GotAKinect directory NOTE: Say *' batchfile'* as one word
"*GotABot elevate 27 degrees*"	Rotate the Kinect sensor to 27 degrees
"*GotABot elevate up*"	Rotate the Kinect sensor up 5 degrees
"*GotABot elevate down*"	Rotate the Kinect sensor down 5 degrees
"*GotABot funny*"	Tell a joke
"*GotABot newscast* "	Start news feed
"*GotABot fox newscast* "	Start FOX news feed
"*GotABot bbc newscast*"	Start BBC news feed
"*GotABot npr newscast* "	Start NPR news feed
"*GotABot forecast*"	Get weather forecast
"*GotABot telltime*"	Tell the time (say 'telltime' as one word)
"*GotABot thankyou*"	Even a robot appreciates acknowledgement
"*GotABot Hello*"	More pleasantries
"*GotABot good bye*"	Even more pleasantries
~~"*GotABot LookAtMe*"~~	~~Stop scanning. Find and Track the nearest face NOTE 1: Say 'LookAtMe' as a single word NOTE 2: If GotABot cannot see your face because it is not pointed in your general direction, say "*GotABot*" and GotABot will attempt to locate your face using sound direction.~~

NOTE 3: say 'GotaBot Stop' or 'GotaBot Localize' to return to scan mode.

TIP: Voice Recognition Configuration

If your robot has a Kinect sensor, configure Windows to use the Kinect microphones for Voice Recognition. These microphones are highly recommended and likely far superior to the default laptop microphone.

7. Additional GotABot Configuration

A number of other parameters may also be set.

Robot Selection

Selection of a robot to control is achieved through this option (*Click Configure->Bot->Robot Select ->ER1 or Configure->Bot->Robot Select ->RoboRealm Robot*). See the chapter GotABot Control of a RoboRealm Robot for more details. See also GotATranslator for additional robot control options.

The *ER1* selection has two additional submenu choices, Enable Bumper Switches and Enable Evolution IRs.

IP Address and Port

The robot's IP address and port may be set with this option (*Click Configure->Bot->Robot IP Address and Configure->Bot->Port*)

Password

If a password is required to establish communications with the robot, it may be set with this option (*Click Configure->Bot->Password*).

Pose

The robot's pose may be set with this option (*Click Configure-> Pose*). A robot's pose is a series of parameters which describes the robot's position on a map, the direction it is facing, and the uncertainties in each of these parameters.

Settings

GotABot's configuration setting may be saved or reloaded by selecting the "File" menu Settings options.

Robot Diameter

The robot's diameter may be set with this option (*Click Configure->Bot->Robot Diameter*). The Robot Diameter not only determines the smallest openings the Robot can move through, but it also sets the distance the Robot will try to stay away from walls and other obstructions. Setting the diameter too large will restrict the Robot from certain openings and/or may cause the Robot to perform many small "stop, turn and go" corrections while navigating narrow spaces. Setting the diameter too small will result in the Robot colliding with the walls. It's a tradeoff.

Camera Resolution

This setting will define the camera's resolution (*Click Configure->Camera->Camera Resolution).* Resolution is entered in the form 800, 600.

Camera FOV

This setting will define the camera's horizontal FOV or Field Of View (*Click Configure->Camera->Camera FOV).*

Camera Offset

This setting will define how far the camera is from the *turning* center of the robot (a positive value indicates that the camera is closer to the front of the robot than the turning center) (*Click Configure->Camera->Camera Offset*). Note – the turning center is usually centered between the two drive wheels.

TIP: Camera Selection / Configuration

Get the very best camera you can afford.

a) Choose a camera with a wide Field Of View. A minimum 60 degree FOV is recommended. More is better.

b) Choose a camera with good low light capability.

c) Choose the highest resolution the camera, wifi link and PC speed will support. 1280 x 1024 works well. Even higher resolutions are better.

Range Finders

Range Finders are optional and are used to perform small pose corrections when near walls. Rangers are also used to provide a limited obstacle detection capability.

Velocity

The robot's linear velocity may be set with this option (*Click Configure->Bot->Velocity*)

Robot Acceleration

The robot's linear acceleration may be set with this option (*Click Configure->Bot->Acceleration*)

Robot Angular Velocity

The robot's angular velocity may be set with this option (*Click Configure->Bot-> Angular Velocity*)

Robot Angular Acceleration

The robot's angular acceleration may be set with this option (*Click Configure->Bot-> Angular Acceleration*)

Max Travel

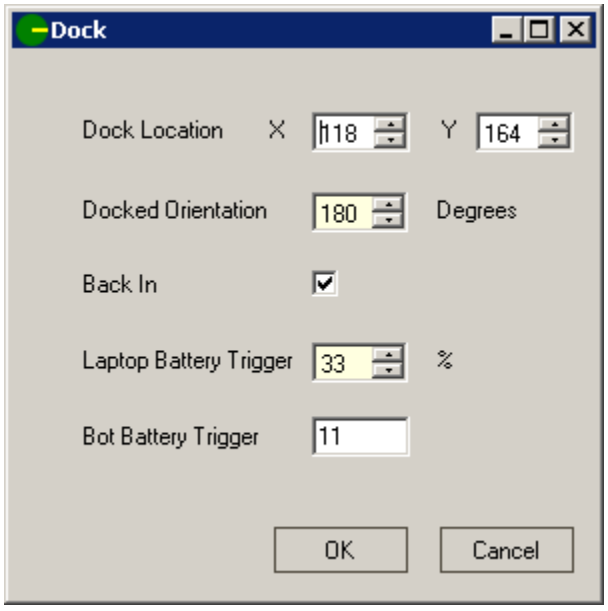
GotABot normally uses an old fashioned 'Sense / Plan / Act' algorithm and thus only checks corrects its position when stopped. Therefore we want the robot to stop frequently and acquire any visible land marks so GotABot can update its location and correct its path. The maximum distance the robot moves before stopping to re-localize may be set by selecting this option (*Click Configure->Bot->Max Travel*).

Docking Configuration

Docking is the process which allows the Robot to (automatically) recharge its batteries.

To configure automatic docking click *Configure->Dock:*

1) Dock Location is the ***Robot*** location when docked.
2) Docked Orientation is the Robot's orientation when docked.
3) Back In should be checked if the Robot must back into the Dock.
4) The Laptop Battery and Robot Battery Trigger fields will trigger an automatic docking operation when either of the respective battery levels falls below the trigger level



7-1 Docking Configuration Form

Localization Time

This setting may be used to control the length of time that GotABot will search for landmarks. Longer times will increase the number of times landmarks will be recognized (up to a maximum of 25 valid recognitions) thereby improving the localization accuracy. This setting may need to be increased if the image processing software is running on a slow PC or if other programs such as video servers are consuming the PC's time. (*Click Configure->Options->Visual Localization Time*)

Mute

This setting prevents GotABot from speaking. (*Click Configure->Options->Mute*)

Enable Robot RCC

This setting allows GotABot to identify Land Marks using a robot based RCC recognizer. (*Click Configure->Recognizer->Enable Robot RCC*).

Enable Local RCC

This setting allows GotABot to identify Land Marks using a host based RCC recognizer. (*Click Configure->Recognizer->Enable Local RCC*). See the chapter titled " Local Image Recognition " for a more complete description.

Enable Local RoboRealm

This setting allows GotABot to identify Land Marks using a host based RoboRealm application as a Fiducial recognizer. (*Click Configure->Recognizer->Enable Local RoboRealm*). See the chapter titled " RoboRealm Interface" for a more complete description.

Enable Y Rotation

This setting allows GotABot to use the RoboRealm Y Rotation variable as additional localization data (*Click Configure->Recognizer->Enable Local RoboRealm->Enable Y Rotation*). Enable this option only if you achieve consistently accurate results after setting RoboRealm's 'depth of field' .

Enable GotAGlyph

This setting allows GotABot to identify Land Marks using the GotAGlyph recognizer included with GotABot. (*Click Configure->Recognizer->Enable GotAGlyph*).

Allow RoboRealm to Control ER1

See Allow RoboRealm to Control ER1 for more details.

8. Operation

Software

By default, GotABot assumes that it is controlling an Evolution Robotics ER1 robot. As such the robot must run the Evolution Robotic's Robot Control Center (or RCC) software or equivalent. The RCC software must be configured to accept API control (*click Settings->Remote Control -> Allow API control of this instance*). See the ER1 User Guide - Accessing the API - for more complete instructions.

Other robots may be controlled with GotATranslator or RoboRealm.

Operation Controls

Destination

The robot may be commanded to go to a destination by:

1) Click on the Tool Bar's Destination Icon ( )
2) Click on the map at the desired position

A yellow path (follow the yellow brick road) will be drawn on the map leading from the robot to the destination and the robot will be commanded to follow it to the destination.

Robot Position

The robot's position on the map may be changed by:

1) Click on the Tool Bar's Robot Icon ( )
2) Click on the map at the desired position
3) Click on the Tool Bar's Robot Orientation up/down controls (Orientation 0)
4) Adjust to the desired orientation

OR – Click *Configure-> Pose* and adjust as required.

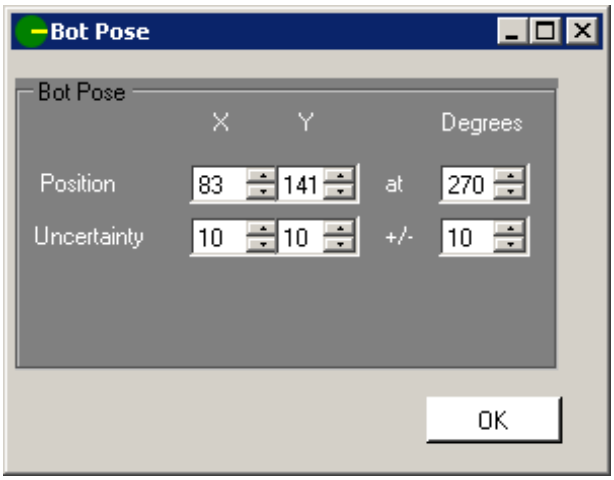


Figure 8-1 Robot Pose Form

Dock

Docking is the process which allows the Robot to (automatically) recharge its batteries.
Docking Configuration is described in Docking Configuration. To manually initiate a docking operation click *Tools->Dock* .

Localize

Selecting this option (*Click Tools->Localize*) will force GotABot to use Land Marks and / or Kinect sensor data to determine its position and orientation

Simulated Movement

Selecting this option (*Click Configure->Options->Simulated Movement*) will prevent GotABot from communicating with the robot. Instead, robot movement is simulated on the map.

Stop

Double clicking on this Tool Bar icon (

) will force the robot to stop.

Debugging

Two mechanisms are provided for debugging, a log file and a debug window. The log file, *GotABot_Log.txt*, may be consulted to review movement and Land Mark recognition events while the debug window (*click ViewTools-> Debug Info*) provides real time information.

TIP – Close the Debug Window

When not in use, the Debug Window should be closed. This will speed processing considerably.

Zoom

The map may be enlarged or shrunk by dragging the bottom right corner of the window with the mouse or by placing the mouse pointer anywhere within the map and using the mouse's scroll wheel.

Manual Controls

The robot may be controlled manually by using the keyboard's Arrow keys (Forward, Back, Turn Left, and Turn Right). WARNING: Use caution with these controls. Rapid changes in direction could conceivably cause stress to the robot's motor control electronics. Some robots (ER1s in particular) are under engineered in this area.

9. Compass

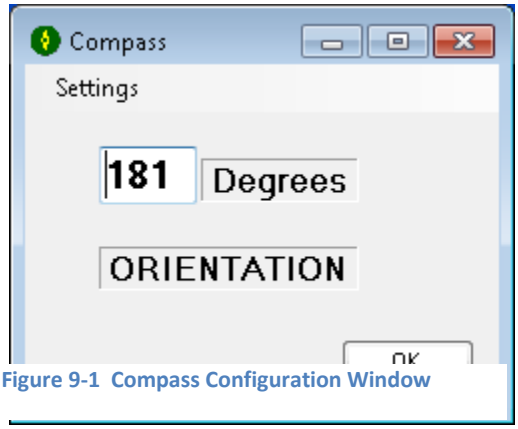


Figure 9-1 Compass Configuration Window

One of the most useful sensors for a robot is a magnetic compass. GotABot provides an interface for an optional Phidget Spatial 1044 sensor. Download and install the PhidgetSpatial drivers and calibration program before using this interface.

Configuration of the compass is accessed by using its Configuration Window (click *Configuration->Compass*). The only parameter which needs to be set is the 'Orientation Offset' which shifts the compass orientation so as to agree with GotABot's orientation (click *Settings->Orientation Offset*).

NOTE: The compass orientation is assumed to have an uncertainty of +/- 16 degrees. The robot's orientation will only use the compass orientation when the robot's pose has an orientation uncertainty of 16 degrees or greater.

10. Video

Video from the Robot may be remotely viewed using GotAGlyph (*click Configure->Recognizer->Enable GotAGlyph*). GotAGlyph can access the video either directly through a MJPEG video stream or through a Virtual Camera utility which allows the video stream to appear as a local webcam.

If not inherently provided by the robot, there are numerous video streaming programs which may be used. I use WebCam XP 5 (free edition) – it works well and the price is right, but there are many other alternatives as well. Yawcam, Skype, Microsoft Messenger are all free and suitable for streaming video

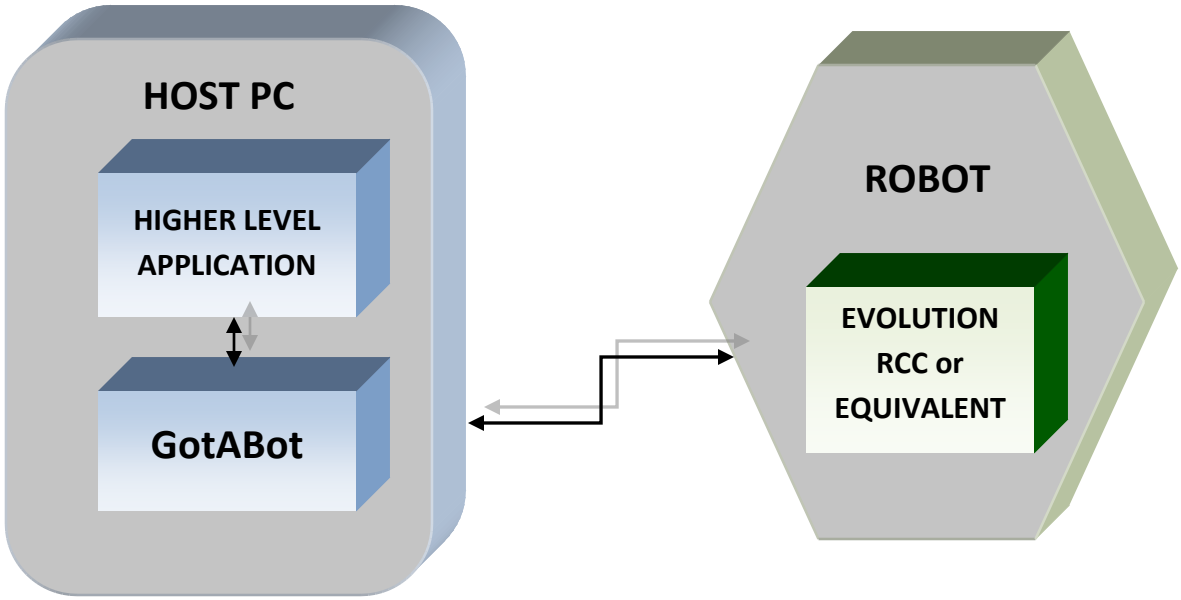
See the chapter Local Object Recognition for additional details on setting up remote viewing.

Note: When streaming video with a Robot based image recognizer, the localization time may have to be increased to account for the increased processing load on the robot's processor.

11. API

Up to this point, we have discussed how GotABot may be controlled through the GUI (Graphical User Interface). This chapter will explore an alternate control method, the API (Application Program Interface). The API is accessed by opening a TCP/IP socket to GotABot and using simple text commands to control GotABot and / or any robots connected to GotABot. This method of providing control is common and widely used by many robot control programs.

An API is especially useful when we have a higher level program or script which provides additional functions or behaviors for the robot. Theoretically, one could create a stack of API enabled programs, each relying and building on lower level functions to achieve new heights of robot behavior.



GotABot contains two identical APIs, either of which can be used by other programs or the user to pass commands or data to GotABot.

Accessing the API through Telnet

Accessing the API can be as simple as using a Telnet session. Although manual telnet control may have limited practical applications, it is very useful for debugging the commands and their sequences before implementing them in a script or program.

There are many different telnet programs but, as an example, here is how one could telnet to GotABot using the telnet program provided with Windows XP:

1) On the PC which will run the telnet program, click the **Start** button (lower left corner of your screen).
2) Click **Run** and type: *"telnet <ip address> <port>"*

   Where the <ip address> is the ip address of the PC running GotABot
   and <port> is the port number of the robot + 100 (API #1)
   or port number of the robot + 400 (API #2)
   An example could be "*telnet 192.168.1.150 9100*"
3) If the robot has a password, then the first step will be to login:

   Type: "*login <password>*" followed by pressing "*Enter*"
   Where <password> will be the same password used by the robot
   GotABot will respond with either "LOGIN ACCEPTED
   OK >"
   or "ERROR: INCORRECT PASSWORD >"
4) Once logged in, any of the API commands may be used.

TIP – Install Telnet Client for Windows 7 or 8

As explained in http://technet.microsoft.com/en-us/library/cc771275(v=ws.10).aspx, the telnet client is not installed by default for Windows 7 or 8.

TO INSTALL:

1. Open a command prompt window. Click **Start**, type **cmd** in the **Start Search** box, and then press **ENTER**.
2. Type the following command:

```
pkgmgr /iu:"TelnetClient"
```

3. If the **User Account Control** dialog box appears, confirm that the action it displays is what you want, and then click **Continue**.
4. When the command prompt appears again, the installation is complete

API Commands

GotABot supports the following API commands:

```
set diameter xxx
set ltime xxx
set size xxx
set units xxx
set velocity xxx
set angular velocity xxx
set pose xxx xxx xx xx xxx xx
set ranger xx xxx xx
set orientation xx xx
set laptop_battery xx
set destination xxx xxx
set simulated xx
set max xx
set offset xxx
dock
localize
stop
bumper xxx
get pose
get rangers
get status
get path xxx xxx
login xxx
load settings xxx
save settings
load map xxx
transmit xxxxxxx
receive
help
```

Set Diameter Command

This command is equivalent to the GUI's Robot Diameter command.

Example

COMMAND: "*set diameter 10*"

RESPONSE: "OK > "

ACTION: Set the Robot's diameter to 10 units.

Set ltime Command

This command is equivalent to the GUI's Localization Time command.

Example

COMMAND: "*set ltime 15*"

RESPONSE: "OK > "

ACTION: Set the robots localization time to 15 seconds.

Set Size Command

This command is equivalent to the GUI's Grid Size command.

Example

COMMAND: "set *size 2.5*"

RESPONSE: "OK > "

ACTION: Set the Map grid size to 2.5.

Set Units Command

This command is equivalent to the GUI's Grid Units command.

Example

COMMAND: "set *units cm*"

RESPONSE: "OK > "

ACTION: Set the Map units to centimeters.

Set Offset Command

This command is equivalent to the GUI's Camera Offset command.

Example

COMMAND: *"set offset 5"*

RESPONSE: "OK > "

ACTION: Set the camera offset to 5 units.

Set Destination Command

This command is equivalent to the GUI's Destination command.

Example

COMMAND: *"set destination 50 135"*

RESPONSE: "OK > "

ACTION: Instructs GotABot to plan a path from the Robot to map position 50 135, and then to issue driving directions to the Robot.

Set Simulated Command

This command is equivalent to the GUI's Simulated Movement command.

Example

COMMAND: *"set simulated true"*

RESPONSE: "OK > "

ACTION: Simulated movement enabled

Set Max Command

This command is equivalent to the GUI's Max Travel command.

Example

COMMAND: *"set max 48"*

RESPONSE: "OK > "

ACTION: Maximum travel distance = 48 units

Set Pose Command

This command allows the calling program to set the robot's location, orientation and uncertainties. The command format is:

set pose <location x> <location y> <uncertainty x> <uncertainty y> <orientation> <orientation uncertainty>

Example

COMMAND: "*set pose 150 202 8 9 90 10*"

RESPONSE: "OK > "

ACTION: This instructs GotABot that the robot is located at location 150, 202 with an 8 unit uncertainty in the X axis, 9 unit uncertainty in the Y axis, and with an orientation of 90 +/- 10 degrees.

Set Ranger Command

This command allows the calling program to inform GotABot of a short range Sensor (typically IR or ultrasonic) reading. The command format is:

set ranger <ranger number (15-64)> <relative orientation> <distance in cm> <Ranger's maximum range in cm (optional – default = 80 cm)>

Example

COMMAND: *"set ranger 15 90 32"*

RESPONSE: "OK > "

ACTION: This instructs GotABot that the robot's Range Sensor #15 is orientated at 90 degrees relative to the ranger location and has detected an object at a distance of 32 cm from the ranger.

Set Orientation Command

This command allows the calling program to set the robot's orientation and uncertainty. The command format is:

set orientation <orientation> <orientation uncertainty>

NOTE: This command will only update the orientation when the robot is stopped and the orientation uncertainty is less than the existing uncertainty.

Example

COMMAND: *"set orientation 90 10"*

RESPONSE: "OK > "

ACTION: This instructs GotABot that the robot's orientation is 90 +/- 10 degrees.

Set Bearing Command

This command allows the calling program to set the robot's orientation and uncertainty. The command format is:

set bearing <bearing> <orientation uncertainty>

NOTE: This command differs from 'set orientation' in that it conforms to the conventional magnetic bearing standards - i.e the angles increase in a clock wise direction.

Example

COMMAND: *"set bearing 90 10"*

RESPONSE: "OK > "

ACTION: This instructs GotABot that the robot's magnetic bearing is 90 +/- 10 degrees.

Set Laptop Battery Command

This command allows the calling program to inform GotABot of the robot's laptop battery charge status. The command format is:

set laptop_battery <percent charged>

Example

COMMAND: *"set laptop_battery .80"*

RESPONSE: "OK > "

ACTION: This instructs GotABot that the robot's laptop battery is 80% charged.

Get Pose Command

This command will instruct GotABot to respond with the robot's current pose in the format:

<location x> <location y> <uncertainty x> <uncertainty y> <orientation> <orientation uncertainty>

Example

COMMAND: "*get pose*"

RESPONSE: "POSE = *150 202 8 9 90 10*
OK > "

ACTION: This instructs GotABot to report the current Robot pose.

Get Rangers Command

This command will instruct GotABot to respond with the Ranger data in the format:

<ranger number> <range> <relative orientation> <X offset from robot center><Y offset from robot center>
<ranger number> <range> <relative orientation> <X offset from robot center><Y offset from robot center>
…

<ranger number> <range> <relative orientation> <X offset from robot center><Y offset from robot center>

Example

COMMAND: "*get rangers*"

RESPONSE: "1 25 60 4 8
2 15 -60 -4 8
6 9 0 0 8
11 6 180 0 -8
OK > "

ACTION: This reports that Ranger #1 has detected an object at 25 units, at an orientation of 60 degrees, and that the ranger is located at an XY offset of 4, 8 from the center of the robot. Similarly, all other enabled rangers (in this example 2, 6 and 11) are also reported

Get Status Command

This command will instruct GotABot to respond with the robot's current driving status (*"docked" or* "*moving*" or "*stopped*")

Example

COMMAND: "*get status*"

RESPONSE: "stopped
OK > "

ACTION: This instructs GotABot to report the Driving status.

Get Path Command

This command will instruct GotABot to plan a path from the robot to a destination, and then to respond with a series of locations indicating the steps required to reach that destination. The command format is: get path <location x> <location y> (where location x and location y are the desired destination)

Example

COMMAND: "*get path 150 55*"

RESPONSE: " Path =
171 188
178 181
178 177
180 175
180 160
181 159
181 157
182 156
182 125
176 119
176 77
154 55
150 55
OK >> "

ACTION: Plan a path to location 150, 55 and then report the steps to reach that destination.

Dock Command

This command is equivalent to the GUI's Dock command.

Example

COMMAND: *"dock"*

RESPONSE: OK > "
ACTION: This instructs GotABot to Dock with its charging station

Localize Command

This command is equivalent to the GUI's Localize command.

Example

COMMAND: *"localize"*

RESPONSE: " POSE = *150 202 8 9 90 10*
OK > "

ACTION: This instructs GotABot to perform a localize operation (look for Land Marks , deduce the Robot's position and orientation) and then to report the Robot's pose.

Stop Command

This command is equivalent to the GUI's Stop command. All active GotABot and robot activities should stop on receipt of this command.

Example

COMMAND: *"stop"*

RESPONSE: " OK > "

ACTION: Stop all movement and localization.

Bumper Command

This command allows the client program to report that a bumper switch has detected a collision. Four separate types of collisions may be reported:

bumper left
bumper right
bumper front
bumper rear

Upon receipt, GotABot will initiate a Collision Recovery sequence

COMMAND	RECOVERY SEQUENCE
bumper left	*stop, move backward, rotate right, move*
bumper right	*stop, move backward, rotate left, move forward*
bumper front	*stop, move backward*
bumper rear	*stop, move forward*

After completing the Recovery Sequence, GotABot will resume travel to its destination.

Example

COMMAND: "*bumper left*"

RESPONSE: " OK > "

ACTION: Stop all movement, then move backward, rotate right and finally move forward. Resume travel to destination.

TIP: Get Some Bumper Switches

Bumper Switches are such simple devices that their utility is frequently under appreciated. I would argue that given the inherent contrariness of a typical robot, bumper switches are well nigh indispensible.

Login Command

This command is required when GotABot's password is set.

Example

COMMAND: *"login alpha1"*

RESPONSE: " LOGIN ACCEPTED
OK > " or
"ERROR: INCORRECT PASSWORD > "

ACTION: Login to both GotABot and the Robot.

Load Map Command

This command is equivalent to the GUI's Load Map (*Click map->Open Map*) command.

Example

COMMAND: *"load map home.bmp"*

RESPONSE: " OK > "

ACTION: Instructs GotABot to load the map file 'home.bmp'.

Load Settings Command

This command is equivalent to the GUI's Load Settings command.

Example

COMMAND: *"load settings settings.xml"*

RESPONSE: " OK > "

ACTION: Instructs GotABot to load the settings file 'settings.xml'.

Save Settings Command

This command is equivalent to the GUI's Save Settings command.

Example

COMMAND: *"save settings"*

RESPONSE: " OK > "

ACTION: Instructs GotABot to save the settings file.

Transmit Command

This command allows the calling program to transmit a command directly to the robot.

Example: *"transmit gripper close"* (instructs)

Example

COMMAND: *"transmit gripper close"*

RESPONSE: " OK > "

ACTION: Instructs the Robot to close its gripper, assuming the robot has a gripper and understands the command 'gripper close'.

Receive Command

This command allows the calling program to receive data directly from the robot. This command must be reissued each time new data is required.

Example

COMMAND: *"receive"*

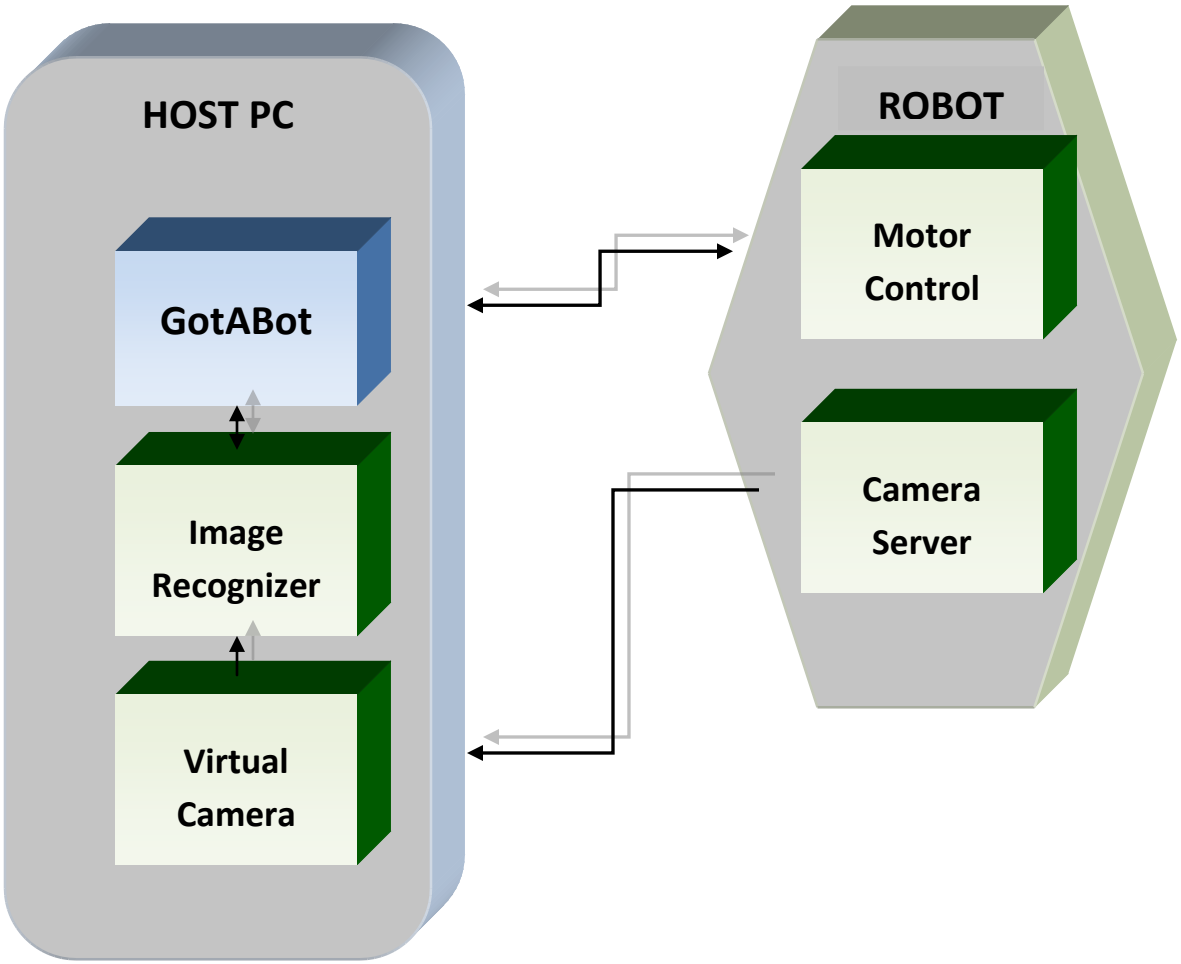
RESPONSE: " OK > "

ACTION: Instructs GotABot to relay data received from the Robot.

12. Local Image Recognition

GotABot's simple system configuration uses a robot based Image Recognition service. In other words, the robot is not only capable of moving on command but also of performing image recognition. An example of such a robot is the Evolution ER1, where the software (Evolution RCC) performs both motor control and image recognition.

An alternative approach moves the image recognition function from the robot to the host PC. This has a



number of advantages including:

i) The host PC is usually more powerful than the robot, hence able to process images faster.
ii) The robot will enjoy a longer battery life if it does not have to process images.
iii) Capturing images is more convenient on the host PC.
iv) Many robots, such as Rovios, Spykees, etc, do not have image recognition capabilities.
v) This configuration provides a method for viewing the Robot's video feed.

The principle disadvantage of this approach is the increased complexity of configuring and operating multiple programs.

Configuration

GotABot

GotABot may be configured to use a local image recognizer through its Configure Menu (*Click Configure-> Enable Recognizer-> Enable Local RCC or Tools-> Enable Recognizer -> Enable Local RoboRealm).* Note – all of the other software programs, motor control, camera server, virtual camera and image recognizers, should be loaded and started *before* starting GotABot.

Motor Control

Presently, GotABot directly interfaces with the Evolution ER1 robot only and the motor control software is their RCC application. Other Robots may be controlled via RoboRealm or through GotATranslator.

Virtual Camera

This software will convert the video stream provided by the Robot's Camera Server and make it appear as a normal web cam. This allows programs such as Evolutions RCC or RoboRealm to easily access the Robot's video.

There are a number of virtual camera applications available but I have had good luck with webcamXP5 free edition and webcamXP5 pro (60 day free trial then…pricey, but works well) for this function. Each will install a virtual camera driver which is listed as "*IP Camera (JPEG/MJPEG)*" and can be connected to as a regular webcam.

RoboRealm may also be used as as the Virtual Camera driver. Be sure to follow the setup instructions at http://www.roborealm.com/help/Virtual_Camera_Driver.php. The following screen shots show how RoboRealm is configured for my needs:

1) *Press Options->Video* and select IP Camera

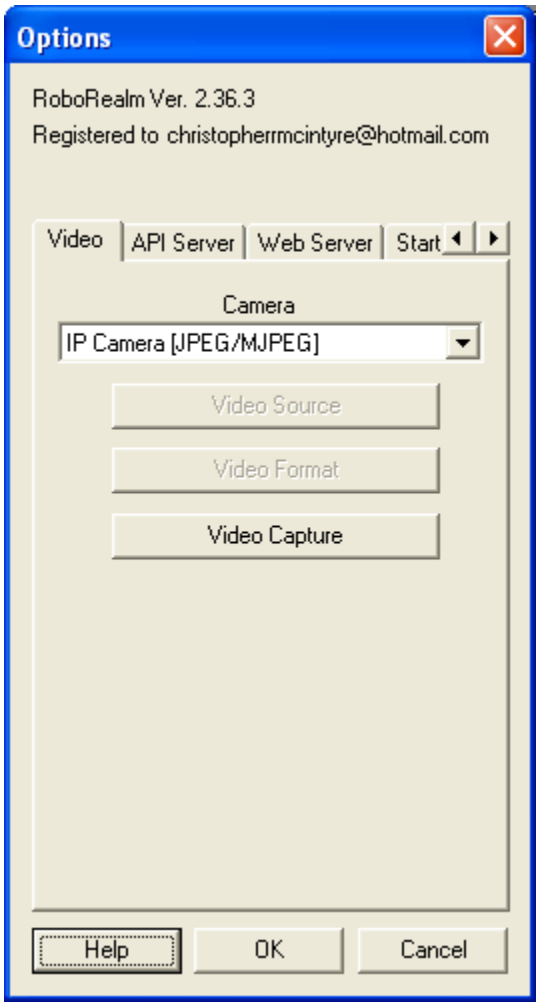


Figure 12-1 RoboRealm Camera Select

NOTE: For this example, we have selected the webcamXP5 IP camera interface.

2) Select 'Video Capture'

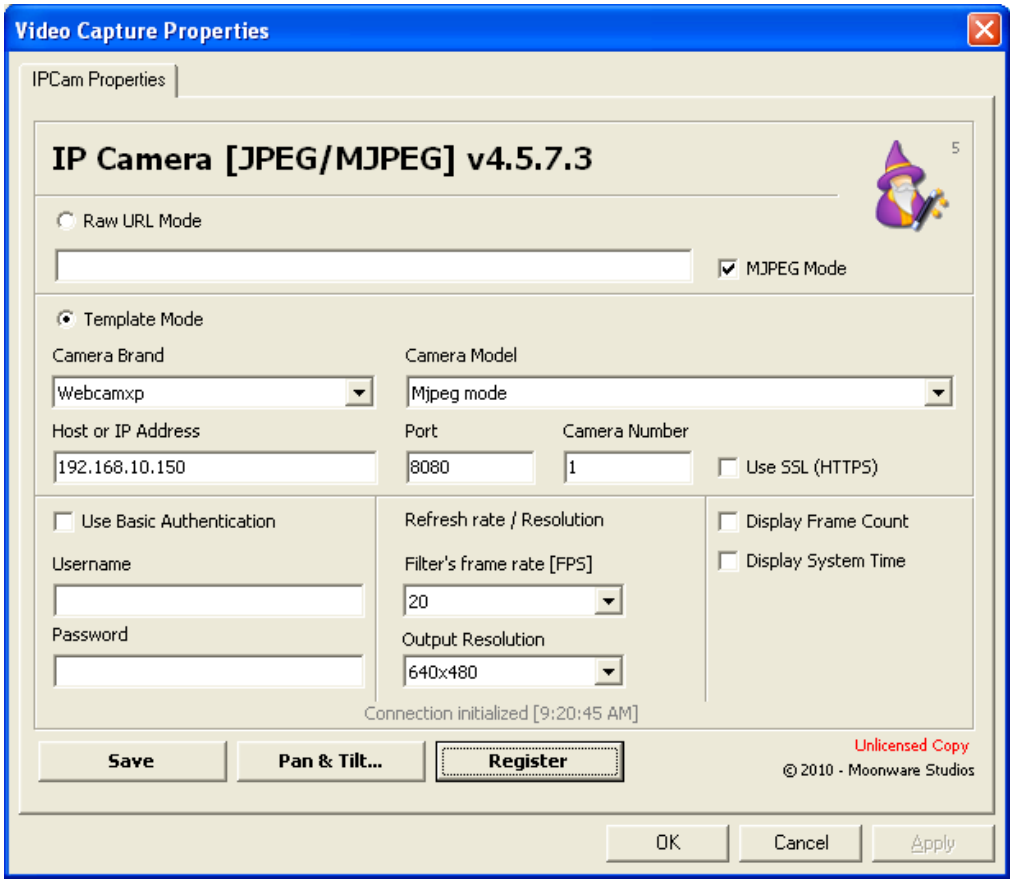


Figure 12-2 RoboRealm Video Capture

HINT: Remember to press '*Save'* when done

HINT: It may be necessary to restart RoboRealm (several times) before the video operates properly.

3) Press *Options->Other* and Check '*Virtual Camera'*. Choose '*Source'* in the *'Use Image' drop box.*

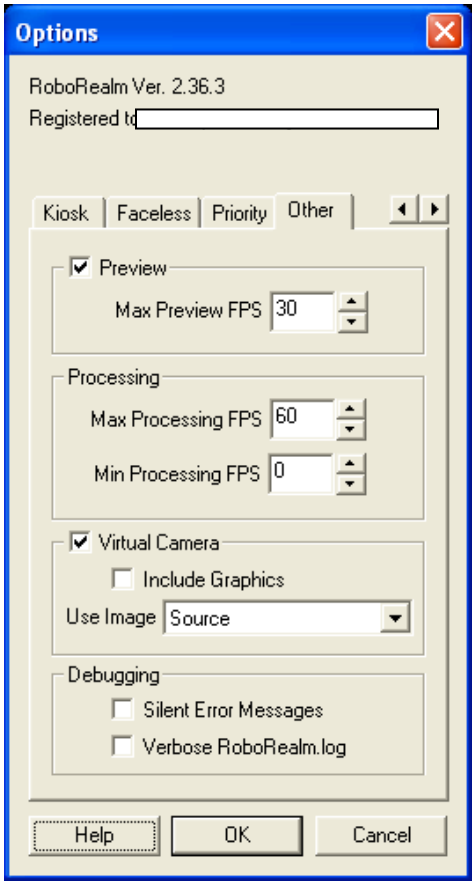


Figure 12-3 RoboRealm Virtual Camera

HINT: It may be necessary to save the configuration and restart RoboRealm before the Virtual Camera operates properly.

Image Recognizer

Evolution RCC Configuration

Use the Settings - > Camera tabs to select the virtual camera.

Use the Setting -> Remote Control tab to select "Allow API control of this instance". Make sure the port is set to the Robot's port +200, and make sure the Password is identical to the Robot's.

RoboRealm Configuration

(see Using RoboRealm to Recognize Fiducials)

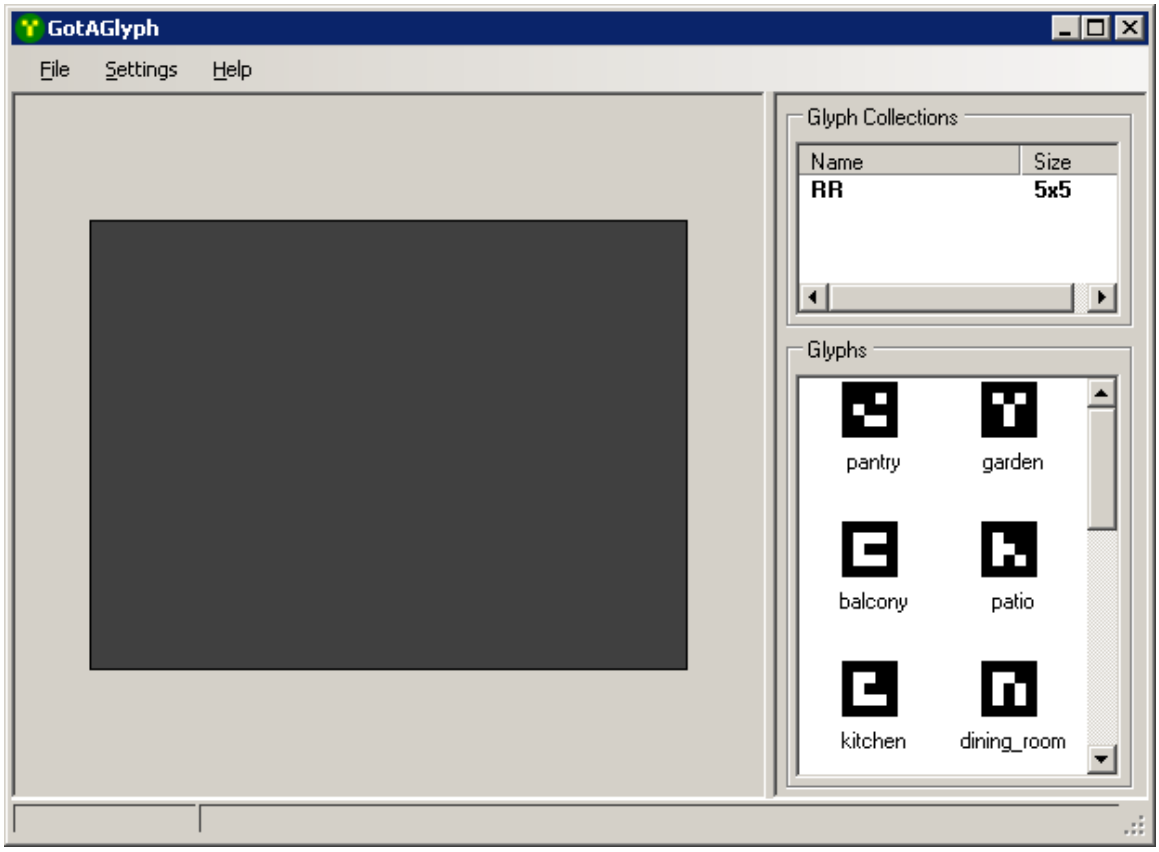
GotAGlyph Configuration

(see http://www.aforgenet.com/projects/gratf/)

Camera Server

I used WebCam XP 5 (free edition) – it works well and the price is right.

13. GotAGlyph



13-1 GotAGlyph Main Window

GotAGlyph is a Glyph recognition application provided with GotABot which can be used in a Local Image Recognition Configuration. This program is an adaptation of GRAFT (Glyph Recognition and Tracking Framework) with the principle difference being an added API for communication. Operational instructions and documentation is provided on the GRAFT homepage (http://www.aforgenet.com/projects/gratf/).

13-2Glyph Example

Glyphs are highly recognizable Land Marks and are a subset of the Fiducials which may be recognized by RoboRealm. Glyph Size and usage is the same as Fiducial usage as explained in the chapter "Using RoboRealm to Recognize Fiducials". (NOTE – Y Rotation is not yet implemented for GotAGlyph)

GotAGlyph is started by selecting *Configure->Recognizer->Enable GotAGlyph*. The camera's horizontal FOV (Field Of View) must be entered (*Click Settings->Options*).

14. RoboRealm Interface

RoboRealm is an application for use in computer vision, image analysis, and robotic vision systems. By combining RoboRealm with GotABot we can form the nucleus of a powerful Robot navigation system. This chapter explains some of the ways they can work together and how to integrate them via RoboRealm's API (In RoboRealm: Press *Options->API Server* and check the '*Activate RoboRealm API Server'*; Enter 6060 in the '*Port'* box).

NOTE: An example of a RoboRealm robo file, suitable for operation with GotABot, is included with GotABot's installation package.

Allow RoboRealm to Control ER1

Although RoboRealm contains interfaces to a number of robots, the Evolution ER1 robot is not one natively supported. Using GotABot however, RoboRealm can now command ER1 motor functions. (*Click Configure->Options->Allow RoboRealm to Control Robot*)

RoboRealm can control the ER1 via the variable '*move***'.**

Variable '*move*'

0	Stop
1	Drive Forward
2	Drive Back
5	Turn Left
6	Turn Right

GotABot polls this variable and directs the ER1 to responds as commanded. Additionally, GotABot writes the ER1's battery voltage to the RoboRealm variable *'battery'* and the ER1's raw position reading to the RoboRealm variable *'position'.*

GotABot Control of a RoboRealm Robot

GotABot can control several types of robots by accessing RoboRealm's interfaces to these robots. GotABot sets two RoboRealm variables, 'move' and 'command', either of which can be used by RoboRealm to control a robot.

Configuring GotABot for this is accomplished by pressing *Robot->Robot Select ->RoboRealm Robot*

NOTE: It is critical to set the Robot attributes, acceleration, velocity, move accuracy, etc to match the robot's characteristics.

Variable 'move' Control Method

This control mechanism works best (which is to say it will only work) if the robot has a relatively slow angular velocity– perhaps 45 degrees per second. The move control method has the advantage of being easy for RoboRealm to work with.

Variable '*move*'

0 = Stop
1 = Drive Forward
2 = Drive Back
5 = Turn Left
6 = Turn Right

RoboRealm must poll this variable and respond as directed. Additionally, GotABot reads the RoboRealm variable *'battery'* and displays it as a voltage in its GUI Status bar.

Variable 'command' Control Method

In this method, GotABot controls the type and magnitude of the move by writing two variables to RoboRealm, '*command*' and '*intDistance*'. This control mechanism is potentially more accurate than the move method but requires a script to be added within RoboRealm to translate the command to actions.

Examples of '*command' and intDistance:*

Variable	*'command'*	*'intDistance'*	
	move	10	= Drive forward 10 inches
	move	-10	= Drive backwards 10 inches
	turn	33	= Turn left 33 degrees
	turn	-125	= Turn right 125 degrees
	stop		= STOP

RoboRealm must poll this variable and respond as directed. When RoboRealm has completed the command, it must write '0' (zero) to the variable '*command*'.

Additionally, GotABot reads the RoboRealm variable *'battery'* and displays it as a voltage in its GUI Status bar.

Using RoboRealm to Recognize Fiducials

RoboRealm provides Fiducial Recognition, a powerful means of determining location based on known Fiducial locations. Enabling this capability is accomplished by loading the Fiducial module in RoboRealm and (in GotABot) by pressing *Configure->Recognizer->Enable Local RoboRealm.* Fiducial names, locations and validity regions are entered into GotABot's Land Mark data base in the same manner as are Land Marks.

Fiducial Size

Fiducial are normally a fixed size of 4.25" square:

|-------- 4.25"------|

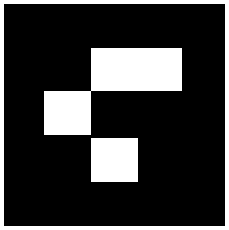


Figure 14-1 Fiducial Size

Naturally, one size does not fit all, and sometimes we need to use bigger or smaller Fiducials. For this, the Fiducial name must be modified so that GotABot can interpret the range information correctly. As an example, say that a Fiducial named "Cat" was not 4.25" square but double sized at 8.5" square. This requires that the Fiducial name "Cat" be changed to "Cat_!#$4". Other sizes are similarly used by adding similar suffixes to the Fiducial name:

Fiducial Size	Suffix
0.53"	_!#$0
1.06"	_!#$1
2.13"	_!#$2
4.25"	
8.5"	_!#$4
17"	_!#$5
34"	_!#$6
68"	_!#$7
136"	_!#$8
272"	_!#$9

Fiducial Y Rotation

RoboRealm provides a variable which reports the Y rotation angle of the Fiducial relative to the robot - very useful for localization (*Click Configure->Recognizer->Enable Local RoboRealm->Enable Y Rotation* to enable usage). Enable this option only if you achieve consistently accurate results after setting RoboRealm's 'depth of field' (see http://www.roborealm.com/forum/index.php?thread_id=4087# for one solution to Y Rotation inaccuracies).

15. AVM Interface

AVM is an optional plug-in for RoboRealm which can be used for image recognition. Compared to the other image recognition applications, range and bearing information is less precise, recognition is slower and setup more difficult. However, AVM can provide superior recognition of some types of objects.

AVM recognition is enabled by clicking *Configure->Recognizer->Enable AVM*.

AVM LandMark entry requires the following additional steps:

i) The LandMark name is of the form *AVM_1* where, in this case, the number '1' signifies that this LandMark is the first image (or *object* in AVM parlance) which the AVM application was trained to recognize.

ii) Range is calculated from the size of the rectangle drawn by AVM around the LandMark. This requires that each LandMark be individually calibrated (click Configure->Recognizer-*>Enable AVM->Range Calibration*). The *Correction Factor* may be calculated by:

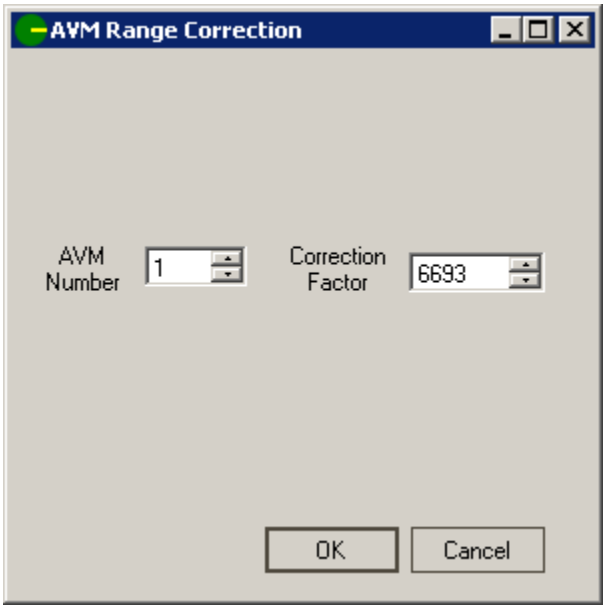
a. start the Debug Window (click *View->Debug Info*)

b. perform a localization (click *Tools->Localize*)

c. In the Debug Window's Debug Log, look for an entry of the form:

" ## "
" Localizing: 0: object ""AVM_1"" 10 30 910.5 100 344.5773"

In this example, the reported range is 344.5773 cm. Adjust the *Correction Factor* and repeat until the correct range is reported.



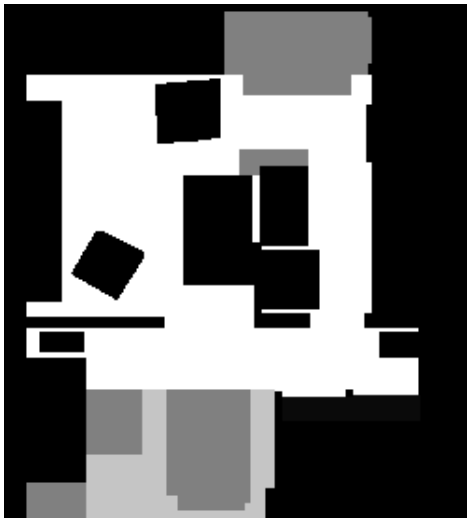
15-1 AVM Range Correction Form

16. GotASLAM

*(NOTE: With the addition of the Kinect interface and some programming changes, SLAM operation is considerably improved from that shown below. Nevertheless, GotABot's SLAM capabilities are still rudimentary and its use is **NOT RECOMMENDED**)*

SLAM, or Simultaneous Localization And Mapping, has long been one of the holy grails of the robot navigation world. After all, making a map by hand can be a fussy and aggravating exercise of measurement and transcription. Far better to let our clever robots handle these mundane details.

Over the years, many schemes utilizing various sophisticated algorithms and differing sensors have been developed and offered. Indeed, many academic careers have been built on this research. At times, SLAM has been declared a 'solved problem' but new information on inadequacies of existing methods and new methods of addressing these inadequacies continue to be developed.



16-1 Hand Drawn Map

GotASLAM Requirements

The most obvious requirement for SLAM operation is, of course, to have some method of determining empty from occupied space. GotABot employs the Kinect interface, Ranger API command and / or the ER1 IR Ranger IF to achieve this function.

16-2 Low Resolution GotASLAM Map

The Trouble With GotASLAM

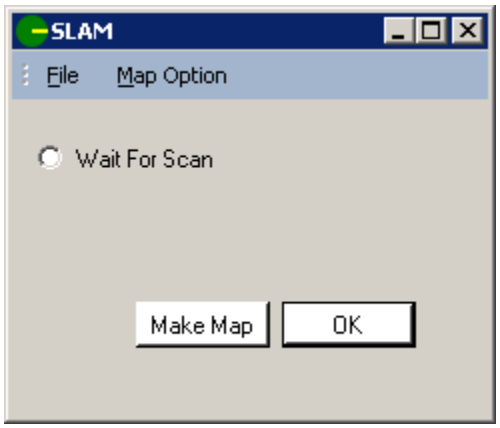
In a word, the trouble with GotASLAM can be summed up as 'resolution'. Unless the Robot's odometry or localization technique is exceptional and the ranging devices measure flawlessly, the resulting map will be less accurate, likely considerably less accurate, than a map generated by the user. For example, compare the user generated map with its equivalent GotASLAM map. The GotASLAM map was generated using short range (10-80 cm) IR sensors which necessitated that the Robot make many short movements in order to cover the area. This, of course, results in uncertainties in the Robot's location , which when combined with the low resolution distance measurements produces the low detail shown. Surprisingly thought, this poor excuse of a map is sufficient for many navigation tasks.

GotASLAM Operation

While map making, GotABot uses the following sequence:

i) Range Scan and map
ii) Move forward a small distance, or turn if there is an obstacle
iii) Repeat

To initiate and control the map making process, follow these steps:



16-3 SLAM Form

STEP 1) Open the SLAM form by clicking *Tools->SLAM.*

STEP 2) Select a SLAM map when prompted. An unexplored SLAM map is a uniform mid grey (one is included in the installation package).

STEP 3) Use the Robot Pose form to set the map pose to match the actual robot's pose.

STEP 4) Press *Make Map* to start the exploration process.

EITHER STEP 5A) In the unlikely event that a full set of Landmarks is defined that will allow near perfect localization or the Robot has perfect odometry then simply let the robot map and explore until the map is adequately explored or the battery is drained. It is good practice to periodically press Pause and save the map (Click *File->Save SLAM Map…)*

OR STEP 5B) The more likely scenario would be to employ a semi-automated approach:

i) When the actual robot's pose differs from that shown on the map or on the Robot Pose form press *Pause.*
ii) Adjust the robot or the reported pose to be equal and press Pause again to continue mapping.
iii) Now is a good time to save the map. Click *File->Save SLAM Map…*
iv) Repeat until the map is adequately explored or the battery is drained.

(OPTIONAL) STEP 6) The map may be enhanced by selecting one of the options under the *Map Option* menu. The map may also be edited in a graphics editor such as MSPaint.

NOTE: Select the *Wait For Scan* radio button when the range scans require extra time

APPENDIX A: Optional ER1 Hardware

The ER1 has the capability of accessing both digital and analog ports through its Robot Control Module (RCM). GotABot exploits this capability by employing them to access various sensors and devices which extend and enhance its basic operation.

Use of these capabilities is optional and not required for basic operation. Robot's other than the ER1 may enjoy similar capabilities through use of the appropriate API command.

ER1 Bumper Switches

Gotabot supports three Bumper switches; left, right and rear. Behavior due to a Bumper Switch activation is the same as per the Bumper Switch API command.

Bumper Switch Wiring

RIGHT SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DIO	21

LEFT SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DI1	21

REAR SWITCH

Switch Contact	*Signal Name*	*RCM J5-*
N.C.	+5V	1
N.O.	GND	25
Center	DI2	22

NOTE: Bumper switches are highly recommended - almost mandatory.

ER1 IR RangeFinders

Gotabot supports up to fourteen GP2Y0A21 Sharp IR Range Sensors (10-80cm detection distance). Behavior due to a Range Switch readings is the same as per the Ranger API command.

Ranger Wiring

J6 Connector

Pin	Signal	Ranger
1	+5V	Ranger VCC
2	+5V	Ranger VCC
3	A0	Ranger 0
4	A2	Ranger 2
5	A4	Ranger 4
6	A6	Ranger 6
7	A8	Ranger 8
8	A10	Ranger 10
9	A12	Ranger 12
10	A14	Ranger 14
11	GND	Ranger GND
12	GND	Ranger GND
13	GND	Ranger GND
14	+5V	Ranger VCC
15	+5V	Ranger VCC
16	A1	Ranger 1
17	A3	Ranger 3
18	A5	Ranger 5
19	A7	Ranger 7
20	A9	Ranger 9
21	A11	Ranger 11
22	A13	Ranger 13
23	GND	Ranger GND
24	GND	Ranger GND
25	GND	Ranger GND

NOTE: Add a bulk decoupling capacitor (100 uF is good) between +5V and GND.

Evolution IR Sensor Pack

GotABot also supports the Evolution IR Sensor Pack. These sensors are of an older technology, not nearly as advanced as the Sharp sensors. They do not provide a consistent range reading and are sensitive to the color of any detected objects. Nevertheless, they do provide some utility and GotABot can utilize them for collision detection, obstacle detection, or localization.

For bump detection, the sensors should be positioned such that:

Sensor 1 = left bumper
Sensor 2 = center bumper
Sensor 3 = right bumper

APPENDIX B: ER1 – Win 7 XP Mode Compatibility

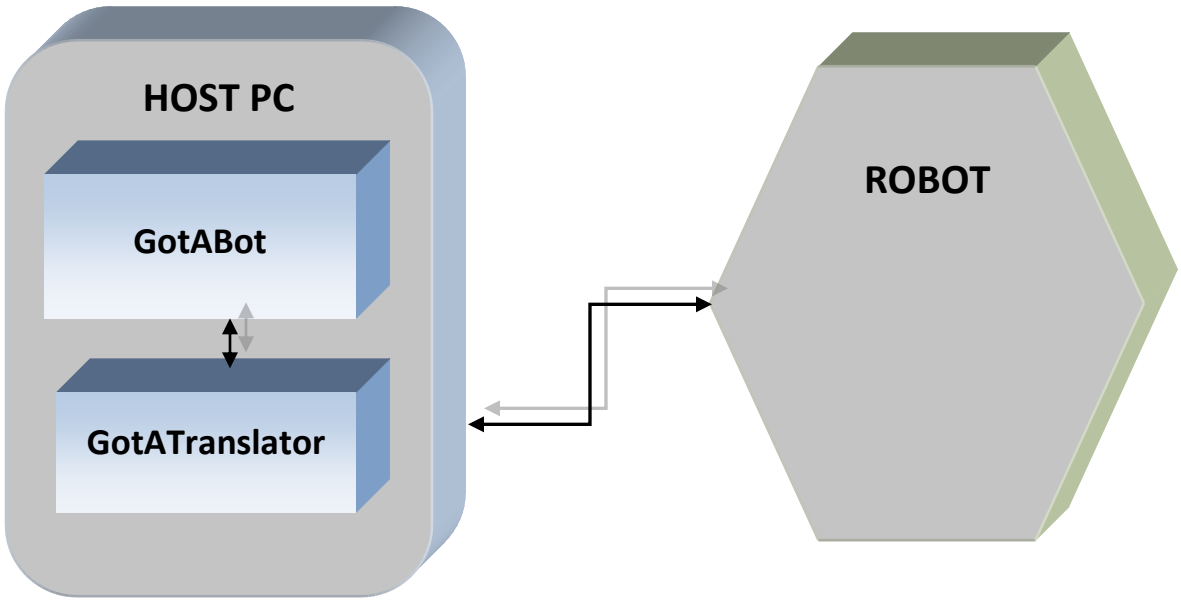
Although written long before Windows 7 was released, the ER1's RCC software can operate in this operating system. If the RCC is only to be used for image recognition then installation is easy and any of the Windows 7 variants (Starter, Home, Pro, Ultra, etc) may be used. However, if the RCM is used to control the ER1's RCC then Windows 7 Pro or better with XP Compatibility mode added must be used. The additional configuration steps required for compatibility mode are:

1) Start a Windows XP Virtual Machine
2) Configure the network
    a. Select your computers network adapter (on the toolbar, press Configure->Options->Networking)
    b. Use the Control Panel to set the IP address (press *Start->Control Panel->Network and Internet->* (etc))
3) Install the RCM software and drivers
4) Attach the RCM's USB port (on the toolbar, press *USB->ER1 Control Module v 1.0*)
5) It is possible that the RCM software will think that the computer is always being charged. In this case, add a file named *config.txt* to the ER1 directory. The contents of this file are:

```
[hardware]
disregard_charging_laptop=1
stop_when_charging=0
```

APPENDIX C: GotATranslator (or How To Interface With New Bots)

So, you've got a robot which is neither supported by GotABot nor by RoboRealm. Perhaps we can help with that. GotATranslator is a Visual Basic project which can translate GotABot output into a format your robot can understand (some assembly required).



STEP 1: Download GotATranslator from http://gotabot.weebly.com/

STEP 2: Download Visual Basic Express (its free) from http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-basic-express

STEP 3: In Visual Basic Express, open the GotATranslator project

STEP 4: Configure GotATranslator's interface with the robot:

i) In Visual Basic Express, open the module *modPublicDeclarations*

ii) Set the variable *boolPassThroughUnTranslated* to 'false'

iii) If your robot's interface is USB or RS232 then set *strComPort* to the name of the port your PC or laptop will use to communicate with the robot. Set the variable *boolComPort* to

'true'. Open the module *modBotIF* and modify the port settings in the function *ftnOpenComPort*.

iv) If your robot's interface is Socket Based then modify Robot_*IPAddress* and *Robot_Port* to be compatible with the Robot's interface.

STEP 5: Tell GotATranslator how to translate GotABot commands to your robot's format:

i) In Visual Basic Express, open the Module *modGotABotTranslate*

ii) In the Function *ftnOneLineTranslate* there is a CASE statement. Modify its code as directed by the embedded comments

iii) Minimally, only the "move" code needs to be modified

STEP 6: Run GotATranslator . Check the "Show Robot Communication" box

STEP 7: Start the Robot

STEP 8: Start GotABot. *Click Robot->Robot Select->ER1.* Make sure the IP address and port numbers are correct

STEP 9: Click on GotABot's Battery Status field (on GotABot's Status Bar – bottom of window – should say "—" before connecting). This will cause GotABot to try to connect with the Robot.

-Check GotABot and GotATranslator Comms status (on Status Bar – bottom of windows) to see if they are communicating.

-Check GotATranslator's Communications log to diagnose problems

STEP 10: Test / Debug / Repeat

GLOSSARY

API	Application Program Interface.
AVM	Optional RoboRealm plug-in for image recognition and other functions.
Bot	Short for Robot.
ER1	Robot manufactured by Evolution Robotics.
Fiducial	In the context of GotABot, a Fiducial image is a high contrast picture designed to be easily recognizable by a computer vision program.
FOV	Field Of View. The horizontal angle, in degrees, captured by the camera.
Glyph	In the context of GotABot, a Glyph is a high contrast image designed to be easily recognizable by a computer vision program. It is composed of a rectangular grid of white squares on a square black background, which in turn is on a larger white background.
Land Mark	User selected images that the robot can recognize and measure range and direction to.
Localization	Determination of a Robot's pose through the use of Land Mark recognition or other sensory input.
Pose	The robot's position in space, quantified in the x, y and theta dimensions.
RCC	Robot Control Center. Software for controlling the ER1 and image recognition.
RCM	Robot Control Module. Hardware component of the ER1.
Recognizer	A program which uses the robot's camera and performs object recognition on the image . This program must be able to determine the range to the object and relative angle. The Recognizer is integral to the Landmark recognition process.
RoboRealm	An application for use in computer vision, image analysis, and robotic vision systems.
RR	Short for RoboRealm
SLAM	Simultaneous Localization And Mapping
VGA	The VGA or Video Graphics Array standard specifies a resolution of 640 by 480 pixels.
YMMV	Your Mileage Might Vary.