Trikebot Graphical User Interface Manual

by Andrés Santiago Pérez-Bergquist <aspb@mapache.org>

Managing the connection

The connection window is where sessions with the robot begin and end. It also displays the status of the most recently taken action, and closing it will quit the entire program. To begin, enter the IP address of the iPaq into the field labeled IP. (If the iPaq has been set up with a domain name, then this may be used as well.) Then, after having made sure that the iPaq is running the TrikePaq program, click the Connect button. If it succeeds, the status will change to "Connected", otherwise it will change to "No Connection!". (All status messages ending in an exclamation point indicate an error of some sort.)

The first thing that should be done after connecting to the robot is to click the Init Robot button until the status reads "Success". At this point, you may command the robot.

When you are done, you may either click Disconnect or Kill iPaq. Disconnect closes the TCP/IP connection to the iPaq, while Kill iPaq explicitly tells the TrikePaq program to terminate itself. Recent versions of TrikePaq terminate themselves when they lose the connection, though this may at some point be changed back to the earlier behavior of attempting to revert to a listening state identical to that entered upon running TrikePaq. For now, the two buttons are functionally identical.

Directly commanding the Trikebot

The Joystick window contains controls for manually commanding the robot to drive and look around. The set of coordinate axes on the left is for driving, and the one on the right is for moving the head.

On the drive axes, the vertical is the speed (going from 100% forward to 100% reverse) and the horizontal is the wheel angle (going from –90° to +90°). To drive, click and hold the mouse down. The red dot will follow the mouse pointer, and the robot will drive at the speed and wheel angle indicated. Releasing the mouse causes the robot to stop. The text fields below the axes update to match the position of the dot. Typing a value in one of these fields followed by a carriage return will send that command to the robot as well (and move the dot). Doing so for the speed field commands the robot to drive at the specified speed, using the last used angle; it will continue driving until the motor timeout is invoked (by default 5 seconds; see the Motor window for how to change it). The Recenter button stops the robot and straightens the wheel.

On the look axes, the vertical is the tilt angle (going from –45° to +90°) and the horizontal is the pan angle (going from –90° to +90°). Clicking anywhere will cause the head to move to that position, and dragging will cause the head to follow the mouse pointer. The text fields below reflect the changing values, and values can be typed into the fields followed by a return to send the command. Recenter positions the camera facing straight ahead.

Motor feedback

The Motor window provides feedback on the voltage differential across the motor and on the amount of current it is being fed. The voltage should lie between about –800 and –200. Voltage is correlated with the speed of the motor, with around –800 being full reverse, around

–200 full forward, and around –500 being full stop. The current is correlated with the effort required to achieve the present velocity. Negative numbers are forward, positive numbers are backwards. This is a 16-bit signed value, and values at the extreme end of the range (beyond ±32,000) typically indicate pushing against an obstacle, although they can appear transiently when getting the motor up to a particular speed.

As previously referred to, the iPaq implements a motor timeout. After it receives a command to drive, it will continue to turn the wheel at the specified speed until it receives a new drive command or the specified time period has elapsed. This period can be set by entering a value in seconds (between 0 and 255) in the Seconds field and clicking the Set Timeout button.

The range-finder

The Range Finder window displays output from and performs functions involving the infrared range finder on the robot. The current range finder reading is displayed in the Range box at the left of the window, and is updated when any motion command is sent to the Trikebot. You can also force it to update at any time by clicking the Refresh button. (This will also refresh other returned values from the robot, such as motor current and voltage.)

The range-finder can only sense obstacles between about 5 to 45 cm from the sensor. Closer objects return higher readings. When facing open space, the reading will be be somewhere around 70 or 80 for most sensors. For a close obstacle, the reading will probably be in the 160 to 200 range. The grey semicircle gives a graphical view of the reading. The blue line is correlated with distance (its actual length is proportional to 255–reading). The window is designed under the assumption that the range-finder will be mounted on the drive wheel. Thus, the angle of the line tilts to match that of the wheel. Since the head is another popular mounting point for the range-finder, this window should be updated to support that mode of operation as well.

Sweeping is a complex behavior the takes multiple readings. When you click the sweep button, it moves the wheel from the start to the end angle specified, taking a reading every step degrees. When it’s done, it displays a blue line for every angle at which a reading was taken. (When this function is used programmatically, it returns an array of readings.) Note that this takes several seconds and should not be attempted while the robot is in motion.

Lastly, the Set Safety button and the Reading box next to it are designed to set a safety threshold by instructing the iPaq to stop the robot immediately if it sees a range-finder reading higher than the specified one. This was designed to allow low-latency crash-avoidance. Unfortunately, due to the limited range of the sensor, this safety feature would only work reliably at low speeds, where it is least needed. Thus, it has never been implemented on the iPaq side. Attempting to invoke it will only result in a status message indicating the packet type was not understood.

Calibration

The calibration window consists of two columns of buttons with associated text boxes. The left column lets you send raw input values to the servos and the motor. The right column lets you chose what user-understandable values those raw values will map to and from. When you first open the window, all the fields will contain the current values of the corresponding constant. Before using the robot, you will have to tune these values and update the source code so the default values match your Trikebot.

The raw servo values have to be integers between 0 and 255. Clicking any of the first six buttons on the left will cause that servo to move to the position in the text field immediately to the right of that button. Be careful not to command the servo to move to a position that could damage the robot. Gradually decrease the minimum value and click the button after each change until you either reach zero or further motion would cause the moving part of the robot to hit the rest of the robot. Now, take a protractor and carefully measure (in degrees) the angle of the moving part. For the pan and steering servos, straight ahead is zero, left is negative, and right is positive, and the range should be close –90° to +90°. For the tilt servo, zero is straight ahead, down is negative and up is positive, and the range should be close to –45° to +90°. Enter your measured degree reading in the field on the far right of the same row that you used to issue the movement command, then click the button next to it. This changes the actual constants. Repeat with the maximum values (except that you stop increasing the maximum value when you reach 255 or the maximum safe position).

For the motor, the four buttons on the left again send raw speed commands, which are in the range –128 to +127, and the right buttons set the corresponding constants. When trying the minimum forward and reverse speeds, you are looking for the speed that just barely causes the robot to start creeping in that direction. For the maximum speeds, be reasonable. You can't change the user-understandable minimum forward and reverse speeds; they are always zero. You can change the user-understandable values that the maximum speeds map to, but it is recommended to leave them at ±100, so that the speeds you enter will represent a percentage of your chosen safe maximum speed.

When you are done calibrating, write down the values you chose for reference, and click the Set As Default button; this will actually update the source to CalibrationConstants.java. After you recompile, your program will now default to your calibrated values. Also, you can use the Save As… and Open… buttons to bring up dialogs that let you save calibrations to an arbitrary file (whose format is identical to CalibrationConstants.java) and load them again from such files, for switching between multiple robots. Note that if you hand-edit calibration files, you must not change the spacing or the ordering of the individual constants, or it will not be loaded by the parser!

Recording and playing back sequences of actions

The Macro window allows you to record sequences of commands sent to the robot, and then play them back at a later time, complete with the timing originally used. To use, first click the Record button; it will now change to Stop. Any commands that you send to the robot will now be recorded. When you are done, click Stop. You will be asked to enter a name for that particular macro, or you can click Cancel to not keep it. To perform that sequence of actions again, select the macro's name in the list, and click Play. The Trikebot will go through the sequence of actions again.

By selecting a macro name and clicking Save As…, you can save the macro to a file of your choosing. Later, you can click Open… and find the file to load the macro again (you will be prompted to name it again). Of particular value is the ability to record macros by hand, save them, and then play them back from inside a program.

When recording macros, you should limit yourself to sending simple motion commands to the Trikebot (by using the Joystick window), as more complex commands (such as most involving the camera) require interactions between the Java GUI and the iPaq that are not recorded by the macro mechanism.

Tuning motor performance

The motor makes use of a PID control system to attempt to achieve the velocities it is commanded to drive at. The PID window lets you adjust these values to try and get smoother motion out of the robot. The defaults should be adequate for the standard motor.

Additionally, this window lets you set the period and latency of the PWM signal used to control the motor. These should be even less likely to require changing.

Working with the camera

The Camera window controls most of the basic functionality of the CMUcam. Prior to doing anything with the camera, you should click the Reset Camera button (and keep doing so until you get a Success as your status). The Get Version button simply queries the camera to find out what version of its firmware is installed; it is primarily used as an easy way to determine if you are communicating with the camera in a coherent manner.

Below that are sliders that let you adjust the brightness and the contrast of the camera. To do so, move the slider then click the corresponding set button. (No commands are sent to the camera until you click the button). Below that are two pairs of buttons that let you turn on and off the Auto White Balance and Auto Gain, which make the camera try and keep the image of the scene constant under changing illumination. See the CMUcam manual for more details on how these features work.

Below that is the Set Window button and four fields in which you can enter the size of the window you wish the camera to use. The x values must be between 1 and 80, and the y values must be between 1 and 143.

Below that is the Get Mean button. It obtains the mean color of the currently set window. (Note that it does not set the window; to use a nonstandard window, first enter its bounds, then click Set Window, then click Get Mean.) The red, green, and blue values of the mean color are returned in the three text fields next to the Mean label, and that color is displayed in the box next to them. (Each color component is an unsigned 8-bit value.) The standard deviations of the color components are displayed below them numerically and as a color as well. This color display is substantially less useful, but just remember that dark colors mean a fairly constant color field while brighter colors indicate more variability.

The last button is called Grab Window, and instructs the robot to the set the window as specified above, and then send the image the camera currently sees to the Java GUI. The image will appear in the Camera View Window. The Camera View window has another button called Grab Frame that is similar, except that is always takes a picture of the entire frame. Note that this will reset the window to the frame, overwriting any previously set window. In either case, the frame grab will take up to 5 seconds to complete (smaller windows are faster), and the display will update itself periodically throughout the process. The image returned is not a single picture, but rather a series of vertical lines taken one frame at a time from right to left. As a result, if the scene is moving, the image will be all blurred. Additionally, the robot cannot do anything else while it is taking a frame grab; all other commands will be ignored until it is done. Frame grabs should not be attempted while the robot is in motion.

Once you have a frame grab, the Camera View window provides additional features. Moving the mouse over the image will cause the current (x, y) position of the mouse pointer to be displayed (the origin is at the bottom left), as well as the RGB values of the currently pointed-to pixel and a sample swatch of that color. The Keep button causes a new window to be created with a copy of the current image. There is currently no functionality to save the images, but a screenshot utility can be used to capture the displayed image(s). (Use command-shift-3, command-shift-4, or the Grab utility on a Mac, the PrintScreen key under Windows, or XV under X-Windows.)

Tracking objects

The Tracking window allows you to track objects with the camera. Using the fields labeled Min and Max for each of the three colors, you can enter a range of colors that you would like to track. For instance, setting Min R = 100, Max R = 255, Min G = 0, Max G = 100, Min B = 0, and Max B = 80 would enable you to track bright red objects. Clicking Track Color Once will search the currently visible scene for objects of that color and return information on them if found.

X and Y are the position of the center of mass of the tracked object. Values of zero indicate no such object was found. Width and Height are the dimensions (in pixels) of bounding box around the tracked object. Pixels is its size in pixels (capped at 255). Conf is a measure of the object's density, with higher values indicating a solid clump of pixels, and low values indicating sparse or poorly detected objects. While the conf can jump to over 150, values of 40 are already very good, and anything over 8 or so is quite decent.

The Track Window Once button finds the largest solid-colored object in the scene, and then returns information on it as if a Track Color Once had been performed on that color.

Active Tracking is a behavior where the iPaq will continuously track an object and move the Trikebot's head to follow the object. Active Color Tracking uses the values entered in the window. Active Window Tracking causes the camera to enable Auto White Balance and Auto Gain, wait five seconds to adapt to the current scene, and then lock onto the largest colored object in view. (To use this, click the button, wait four seconds, and then place the object you wish to track so that it takes up most of the camera's field of view).

Once Active Tracking is enabled, the camera will follow the object, and the tracking info will update continuously. You will not be able to command the pan and tilt servos or communicate with the camera while Active Tracking is on. If the camera loses the object, it will stand still, and you should be able to command the pan and tilt servos until it finds the object again. By default, Active Tracking takes command of both the pan and tilt servos. Using the checkboxes, you can disable either or both servos, so that the camera will only track horizontally or vertically, or not move at all. Any servo that is not being commanded by Active Tracking can be commanded by the Java GUI. (Disabling both is generally not very useful from the interface, but can be useful inside programs where you wish to get continuous feedback about whether or not you are looking at an object while moving the head yourself.)

To end Active Tracking, merely click Stop Active Tracking. Note that Auto White Balance and Auto Gain will be off afterwards, regardless of their state beforehand.

Adding your own functionality

Lastly, the User Window contains ten buttons and twenty text fields that do nothing. You can program them yourself to run programs of your own devising involving the Trikebot. Please see the Tutorial.pdf file for more information on how to get started programming for the Trikebot.