# Ergo V.1.1 User's Guide


## John Anderson
**University of Manitoba**

**E2-476 EITC, Department of Computer Science**
**Winnipeg**
**Manitoba**
**R3T 2N2**
**Canada**
**andersj at cs dot umanitoba dot ca**

## Jacky Baltes
**University of Manitoba**

**E2-402 EITC, Department of Computer Science**
**Winnipeg**
**Manitoba**
**R3T 2N2**
**Canada**
**jacky at cs dot umanitoba dot ca**


**2008-12-01**

### Revision History

Revision 1.0 2008-12-01 Revised by: JEA
First official release


Ergo is a Linux-based open-source vision server, ideal for tracking robots. It is stable, easy to calibrate, and operates well under a wide range of lighting conditions, because it is indifferent to colour. It has been used for a wide array of robot soccer applications, as well as in experimental settings requiring global object tracking. Ergo broadcasts descriptions of tracked objects over ethernet, allowing clients interested in information from the visual feed to independently receive this information. This document discusses how to install and configure Ergo in a typical Linux environment. Examples are shown running on Ubuntu, but installation for and operation on other distributions is similar. After installation and configuration, we describe camera calibration and the use of this software.

# 1. Introduction

This user's manual describes the installation, calibration, and use of Ergo. Ergo is a colour-indifferent video capturing and object tracking program especially targeted towards real-time domains.

Ergo provides the following features:

- Fast colour processing routines
- Indifference to colour when tracking robots, allowing operation under variable lighting conditions
- 50Hz(PAL)/60Hz (NTSC) operation in field mode
- Tracks objects in 2 or 3 dimensions
- Flexible architecture that allows the user to easily add new objects to track

## 1.1. Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples and information at your own risk. There may be errors and inaccuracies that could be damaging to your system. Proceed with caution, and although this is highly unlikely, the author(s) do not take any responsibility.



## 1.2. Credits / Contributors

In this document, I have the pleasure of acknowledging:

- John Anderson < `andersj at cs dot umanitoba dot ca` >
- Jacky Baltes < `jacky at cs dot umanitoba dot ca` >
- Paul Furgale < `paul.furgale at robotics dot utias dot utoronto dot ca`>
- Michael de Denus <`mdedenu at cs dot umanitoba dot ca`>
- Jeffrey Allen < `jallen at cs dot umanitoba dot ca` >

## 1.3. Feedback

Feedback is most welcome. Send your additions, comments and criticisms to : < andersj at cs dot umanitoba dot ca >.

# 2. Installation

This section describes the installation of Ergo on a Linux system. It shows where to obtain Ergo itself and details dependencies on other software.

## 2.1. Before Installing Ergo

Ergo requires a number of other packages before installation can begin. In particular, it requires that QT4 and its associated libraries (including *libqt4-dev*) exist on your system. Use the normal package update process on your flavour of Linux to obtain and install these (e.g. Synaptics on Ubuntu). The instructions that follow assume you have QT4 installed, and that you have a working driver for the video capture device (webcam, capture card) you are using. Getting any particular camera operating on your system is beyond the scope of this document.

## 2.2. Obtaining Ergo

Ergo may be obtained from the software repository on the University of Manitoba Autonomous Agents lab website. Look for the downloads link on the top-level lab website (http://aalab.cs.umanitoba.ca/), then go to the *Vision Software* section. The download link will access a bzipped tarfile (.tbz) containing Ergo source and related software. The same software repository also contains a number of utilities useful for Ergo, such as tools for logging and viewing logs, that may be of interest once you have Ergo running. The repository also contains tools that you can interface with Ergo, such as a world server for mixed reality applications.

Begin by untarring the compressed ergo download:

```
tar -xjvf ergo.tbz
```

This will create an `ergo-vision-server` directory. Now move into this directory:

```
cd ergo-vision-server
```

## 2.3. Installing Ergo

There are two software packages that must be installed before Ergo itself. These packages are included in the Ergo distribution. The first of these is *dotconfpp*, a configuration file parser.

Assuming you are at the top level of the ergo directory, to install dotconfpp.0.0.5, do the following:

```
tar -xzvf dotconfpp-0.0.5.tar.gz
cd dotconfpp-0.0.5
./configure --prefix=/usr
make
sudo make install
```

Ergo looks for dotconfpp in the */usr* directory, requiring the *--prefix* option above. As usual, root access is required to perform a **make install**. Throughout this document, we assume the Ubuntu security model and prefix these with **sudo**. The common alternative to this is to become root (**su**) before issuing the **make install** command. Substitute becoming root anywhere you see **sudo** in this document if you so desire.

Next, libdc1394 *Version 2, not Version 1* must be installed. This is required to build Ergo whether or not you intend to use a 1394 (Firewire) camera. Again, a suitable version is supplied with the Ergo distribution. You need to move back up to the main Ergo directory and then perform a similar untar and build to the one just performed:

```
cd ..
tar -xzvf libdc1394-2.0.2.tar.gz
cd libdc-1394-2.0.2
./configure --prefix=/usr
sudo make install
```

You are now ready to install Ergo. Move to the *src* directory, and perform a similar build:

```
cd ../src
qmake
make
```

It is important that you *do not* supply the *-project* option on **qmake**. The default **qmake** will use the *ergo.pro* file supplied with the distribution, which is what you want.

Assuming this runs without errors, you have built Ergo. Any errors should be the result of the dependencies noted above. If errors occur, perform a **make clean**, install the missing component, and try again. After building Ergo, you may want the binary to be somewhere accessible to all users. If this is desired, copy the binary to that location:

```
cp ergo /usr/bin
```

# 3. Running Ergo

Ergo requires a configuration file (`ergo.conf`) to run. This file provides specifications for the objects to be tracked by the system, along with many alterable system parameters, and exists in a configuration directory that is also used to hold files that are created when Ergo is calibrated in a specific setting. A number of example configuration directories are provided with the system. *Config*, *ConfigDC1394*, and *ConfigTv* provide example configurations for an attached USB camera, 1394 (firewire) camera, and tv capture card respectively. Each of these contains settings for particular cameras or capture cards we tend to use in our own work (e.g. the *ConfigDC1394* directory is set up for the Point Grey Scorpion camera). For your system, you will likely want to copy the directory that is closest to what you have and make alterations from there. While doing so, you may wish to place the configuration directory in the location/account from which you intend to run Ergo, or to somewhere on the system accessible by all accounts. Since other files will be created within the configuration directory when you run Ergo, the current user needs to have write access to files in the configuration directory. If this is a problem, you need to change to the configuration directory (or copy you have made) and make things writeable:

```
cd  insert_path_to_configuration_directory_here
chmod u+w *
```

Inside the configuration directory, the `ergo.conf` file contains many settings that you will eventually want to customize. You need to change only two of these to be able to run the software and ensure your installation is correct. Open the `ergo.conf` file in your favourite text editor. Existing values in the `ergo.conf` file can be commented out using **#** if you want to ensure they are there to restore changes from later.

Look for the **CONFIG_DIR** variable in the file: it specifies the path to the configuration directory containing `ergo.conf` and dependent configuration files. This must be changed to reflect the location of the configuration directory you are using. This can be specified as an absolute path or a relative path from where you will be invoking Ergo. The former of these is likely preferred, since the latter limits where Ergo can be run from.

Below this definition in the `ergo.conf` file is a set of <videostream> parameters that specify parameters for the camera you are using. If you do not know appropriate settings for these parameters given the camera you are using, running Ergo and allowing it to fail with your camera given the default parameters should result in a display of the available modes for your camera as Ergo tries to use them. For example, the scorpion camera default mode in Ergo is 1280x960, which is the setting in `ergo.conf` in the `ConfigDC1394` directory. If your camera cannot sustain this resolution, Ergo will ultimately fail with this setting, but error lines showing the camera attempting the resolutions it can work with should be displayed. You can then set the height and width to useable alternatives. In addition to the height and width, there is a *Bayer* option that is used to indicate whether your camera uses a Bayer colour filter array. If the colour in the display seems wrong when you run Ergo, try setting Bayer to N (or comment it out) to turn this setting off.

If you are using a firewire camera and have previously attempted running Ergo with one camera and change to another, you may have to unplug the new camera and plug it back in order for the camera driver to release the firewire bus. This may also occur anytime Ergo quits abruptly or fails to start.

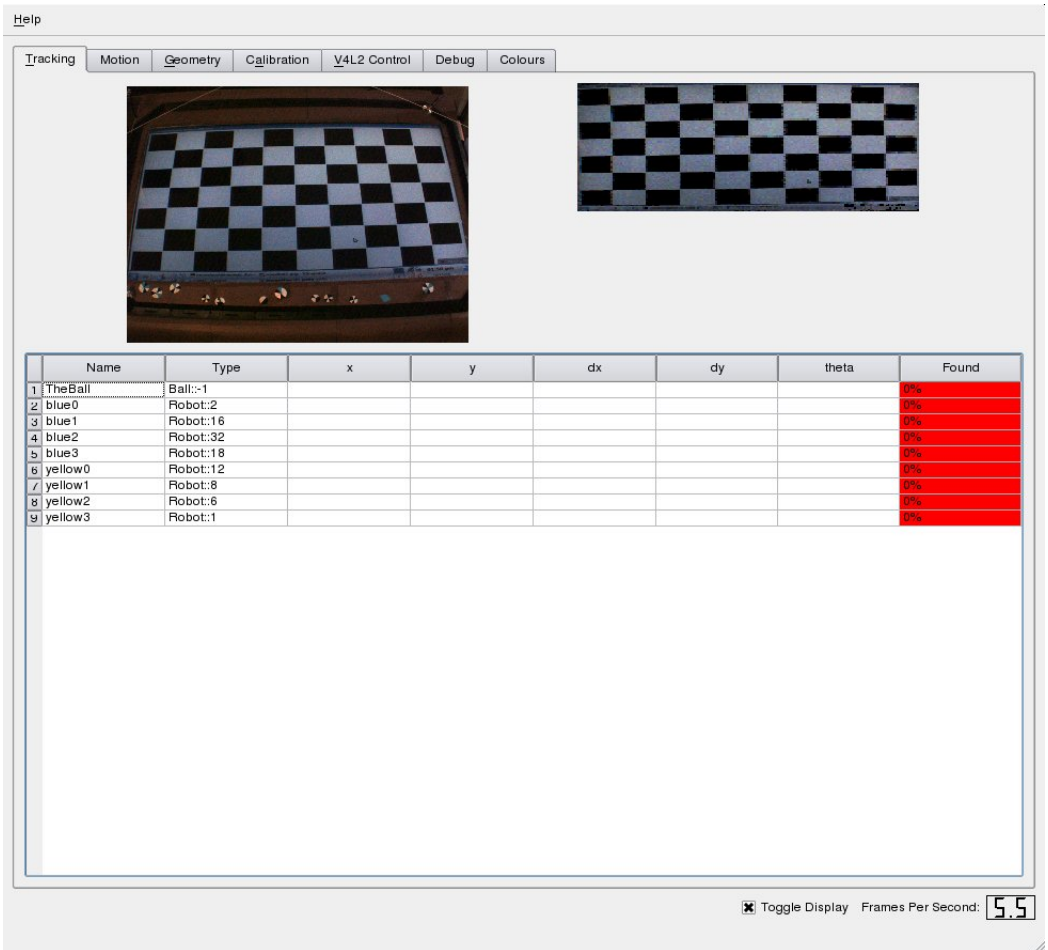
Now that you are ready to run Ergo, you need to specify the location of the appropriate `ergo.conf` file on the command line. For example, if you have a firewire camera connected to the system, and you are in the ergo/src directory, ergo can be run using the following command:

```
./ergo -c ../ConfigDC1394/ergo.conf
```

If you've moved or copied ergo to */usr/bin* or similar public directory, you no longer need to be in the ergo directory to run the system, but you must still supply the location of the configuration files. In this case you would use:

```
ergo -c insert-path-to-configuration-directory-here/ergo.conf
```

**Figure 1. Initial View**



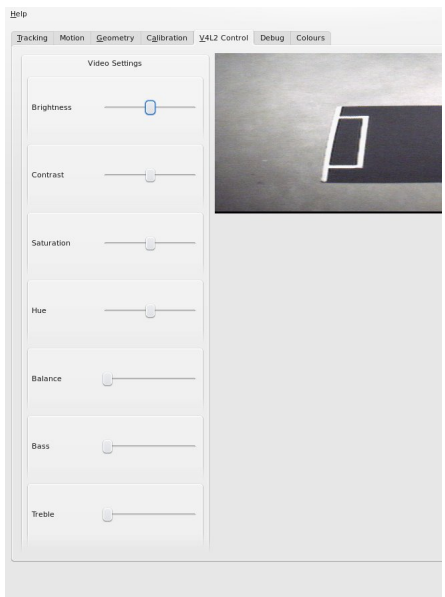
In either case, the result should be an Ergo window similar to that of Figure 1. The window presents a tabbed interface, and the **Tracking** tab is selected by default. The left side of the window shows your camera's output (assuming your camera is properly connected, its driver is working, and the camera's settings have been entered into `ergo.conf` as described above). The right side of the window provides space for two images. The first is intended to show motion in your camera's image, and below that, a detailed view of any selected recognized object (no object is selected initially, so there is no image here initially). Below these images is a list of the objects to be recognized: each of these corresponds to a description in the *ergo.conf* file indicating what the system is supposed to be tracking. Here, no objects are successfully recognized because nothing in the image corresponds to any object the system's default configuration file states it should be looking for. Altering this configuration file to look for your own objects of interest will be covered in Section 5.1.

# 4. Calibration

Now that you have Ergo running, you need to set up the system for your current camera drivers, calibrate the system for the current camera position and rotation, and then supply it with information about colour (to the degree that colour is relevant in your application, since the system is indifferent as far as tracking robots). This section describes the details of this process.

If you are using an IEEE1394 (FireWire) camera, then you will have a direct feed with no camera drivers, and you can move straight to calibration. If not, you will likely want to change some of the driver settings to improve the camera image under you current lighting conditions before calibrating. Click on the **V4L2 Control** tab. You will see a display similar to that of Figure 2, where basic features such as the brightness, contrast, colour saturation and hue, etc. can be altered. When you are satisfied with the quality of the image produced by your camera, you can begin calibrating the system.

**Figure 2. Camera Options**



## 4.1. Camera Calibration

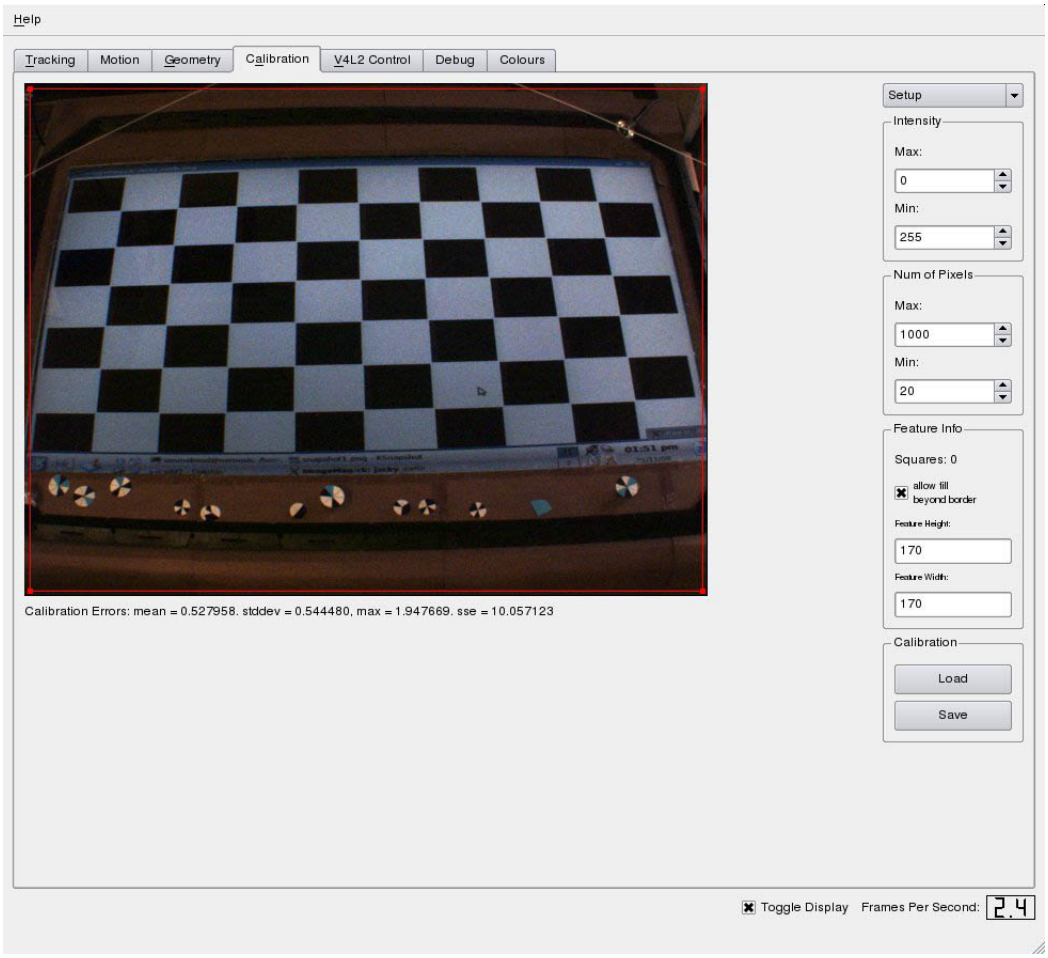
Rather than specifying the precise placement of the camera ahead of time, Ergo relies on determining a run-time mapping between the pixels in the image and the actual physical space in the area being captured by the camera. To allow Ergo to determine this mapping, you will need a calibration image or carpet made up of rectangles repeated at precise intervals. The figures shown in this section all illustrate a carpet presented as a screen image displayed on a large horizontally mounted LCD (the image shown in Figure 1), since we currently use Ergo mainly for mixed reality applications. A sample image is included with the ergo distribution (`calibration.png` at the top level), and a python script for generating such images is also included (`make_calibration.py` in the `tools/` directory). A physical carpet made up of a regular matrix of rectangles temporarily laid over your playing field is also suitable to use, though it is much more work to create than a simple graphic image. Previously, we have used everything from cloth to taped printed paper successfully. The image shown in these examples is also using only black and white, but a carpet made of regularly spaced coloured rectangles is also possible, provided the colours can be distinguished from one another by intensity rather than color difference (i.e. light vs. dark colours).

If you are using a physical calibration carpet, remember that accuracy in consistent matrix cell size is extremely important. It is tedious doing very precise cutting and pasting when making up a calibration carpet, but it makes a huge difference in the quality of the visual recognition you will get from Ergo. The calibration carpet is what Ergo relies upon for accuracy in spatial measurements, so even being a few millimeters off over the course of the whole carpet can significantly lower the effectiveness of this software. The ease with which a calibration image can be generated virtually is just one of the many advantages to using a large screen LCD for a robot playing field (if you are using our world server software for mixed reality applications, Ergo will send a message over ethernet automatically instructing the world server to bring up the calibration carpet, for example).

Irrespective of whether your calibration matrix is physical or virtual, it is important that the matrix cell size not be too small, so that the program gets a good picture of spatial proportions over the whole visual area. Too many selected rectangles can also lead to overfitting: as a rule of thumb, you want around fifteen to be in the selection area (our example uses eighteen).

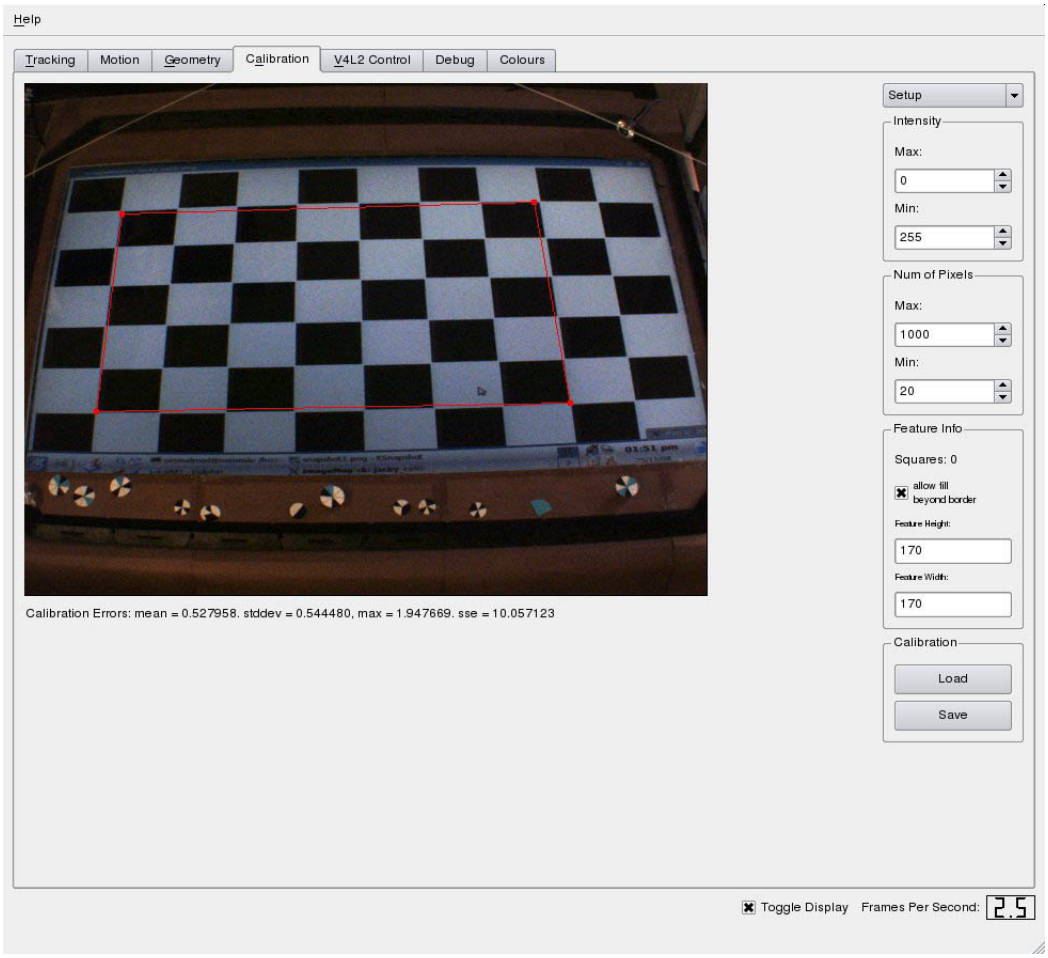
To begin calibrating Ergo, click on the **Calibration** tab. The window (Figure 3) will display the camera image on the left, some statistics below it, and parameter settings on the right.

**Figure 3. Initial View**



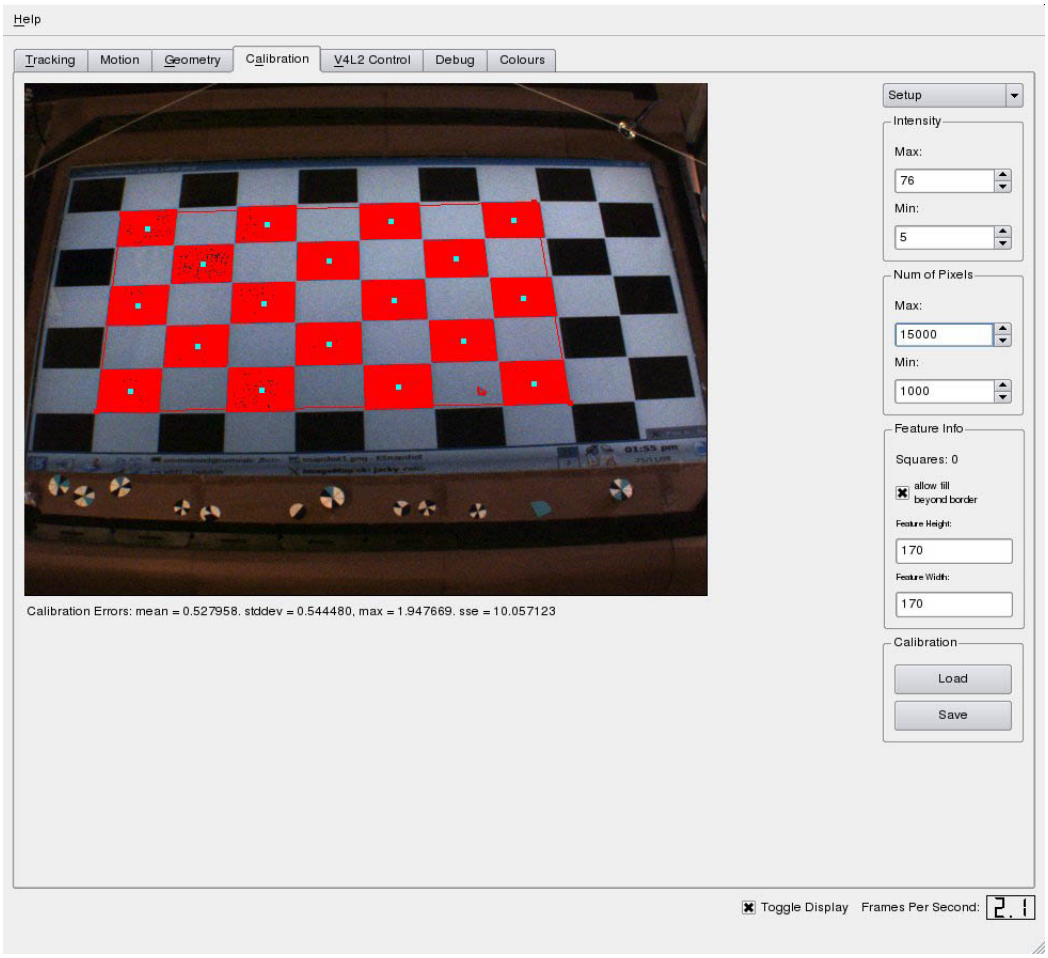

The first step in calibrating Ergo is to provide the software with data that will allow it to determine the relationship between pixels in the visual image and actual distances in the real world (i.e. allow the system to extrapolate the camera position and angle from the image). You will need to select an area of your calibration image/carpet with an appropriate number of rectangles to allow Ergo to make these calculations. Figure 3 shows a red rectangular frame with four red handles in the corners along the outside edge of the camera image. Click and drag these, one by one, to form a bounding box around a set of rectangles in the calibration image/carpet (noting the size guidelines mentioned earlier). The corners of the selected grid of rectangles should all be the same colour of rectangle you are going to use for the calibration, and the four corners must themselves form a rectangle in the physical world (block coordinates are initially assigned along straight lines between these points). Figure 4 shows the four handles dragged to suitable positions in the image.

**Figure 4. Defining A Bounding Box**




Now that you have selected an area on which to base the calibration, you need to make the system aware of the rectangles within that area by selecting all those of one colour. Right-clicking on a pixel in the image adds the colour of that pixel to the selected colour range, and shades the corresponding colours in the camera image. Your goal here is to select one of the corresponding sets of rectangles in the calibration image/carpet, across the range defined by the bounding box. You will see the values in the *max* and *min* text areas under the **Intensity** label on the right change as a result of selecting pixels. You can also manually change the values in this box to select or deselect a larger colour range at once (e.g. you might lower the minimum intensity directly if you accidentally select too large a range of pixels and end up shading more than just the rectangles in the image - this is easy to have happen if you do not use a black and white carpet/image, since the rectangles are less easily distinguished from one another). As potential rectangles in the calibration image/carpet are recognized (i.e. the colour range increases enough to capture most of any given rectangle), they are labeled with a dot in the image displayed. If the rectangle is properly recognized, the dot will be in the *center* of the rectangle, as shown in Figure 5. If the dots are not in the center, the rectangles are not being properly recognized.

**Figure 5. Recognizing One Set of Rectangles in the Bounding Box**



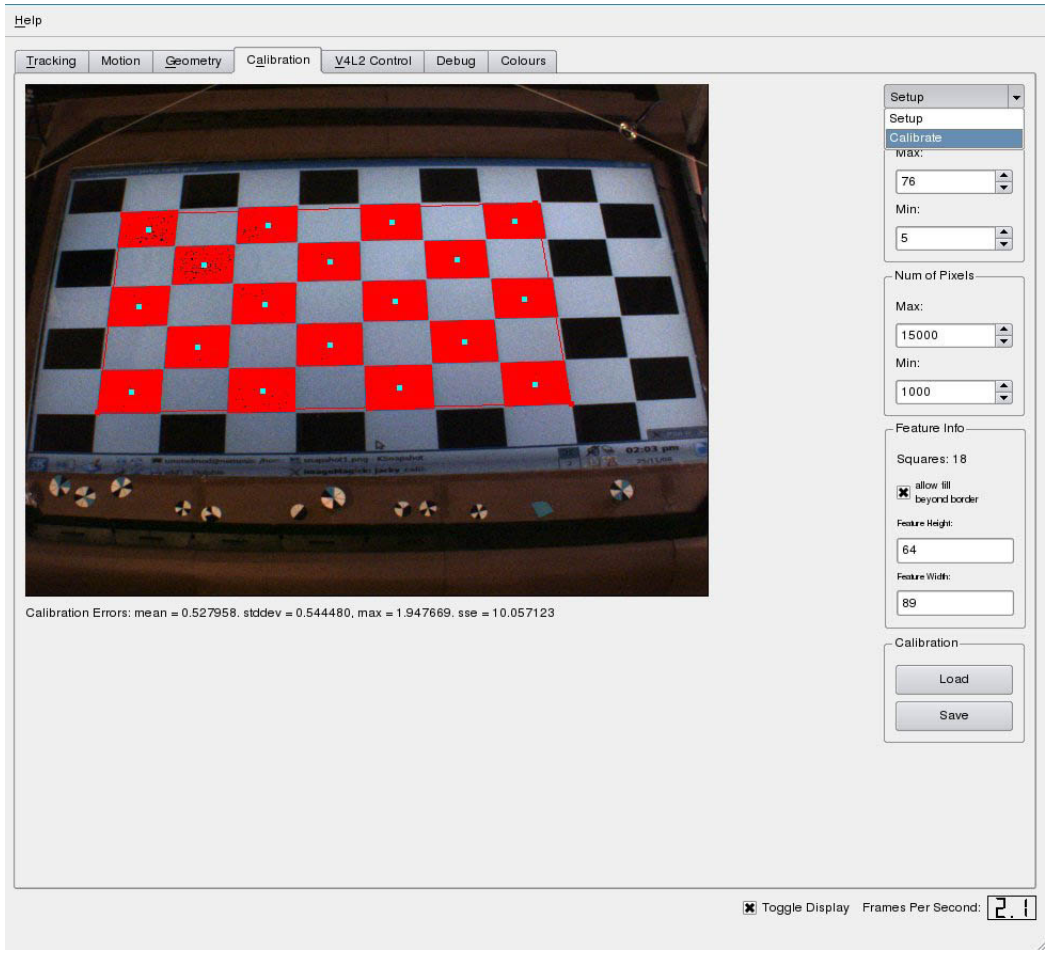

Below the **Intensity** values on the right of the screen are a pair of values labeled **Num of Pixels**. Note also that both of these values has been altered between the images shown in Figure 4 and Figure 5. These values represent the maximum and minimum sizes of a blob in the image that Ergo should consider as a potential rectangle during the calibration process. The purpose of these parameters is to make the search for rectangles easier, by giving the system their approximate size, and also to eliminate noise: if any size of area could be a rectangle, there would be many potential false positives. Chances are when selecting the pixels that make up a set of rectangles, you will have many such small stray areas that happen to be the same colour in the image: appropriate values for these parameters let the system ignore these. You will want to alter these values to fall somewhere close to the range of pixels expected in a rectangle, given your camera, your calibration image/carpet, and the distance between them. These values will vary widely between installations, as the change in values between Figure 4 and Figure 5 shows. In a 1280x960 image with large calibration image/carpet rectangles, for example, 10,000-15,000 pixels can make up a rectangle very easily, but this would be overwhelmingly large for a 640x480 image with small calibration rectangles. The same difference would occur for a camera positioned very close to vs. very far away from the calibration image/carpet. The configuration shown in these figures is for a firewire camera, and so the maximum size of a rectangle was significantly increased from the default values. If

the values chosen here are not appropriate, you may have the rectangles properly selected in the previous step, but still have no rectangles noted by the system (i.e. some or all centered dots missing). The maximum valid size value accepted is 15,000, so consider this when making up the calibration image/carpet you will use for your camera.

In addition to changing the anticipated number of pixels in rectangles through these values, further down the right side of the window is a *Fill Beyond Border* check box. This is checked by default, which allows pixels to be shaded beyond the width of your bounding box (increasing forgiveness if your bounding box selection is a little off, and allowing tolerance for distortion by the camera lens). In the event you want no exceptions outside the bounding box, this can be unchecked.

Now that Ergo knows the centers of each of the rectangles in the area you've selected in the calibration image, it needs to know how these translate to physical distance on the image/carpet. You need to measure the x and y distances *between the same corners of two rectangles of the same colour* on the image/carpet. We find this is most easily done by measuring the distance between the same corner of two rectangles of the same colour (e.g. the upper-left of one black rectangle and the next one) and then dividing that distance by two. This is of course the same distance as the distance between centers - corners are just much easier to get an accurate measure between. Obtain these measures (in millimeters) in each dimension and enter them in the *Width* and *Height* locations respectively, labeled under **Feature Info**.

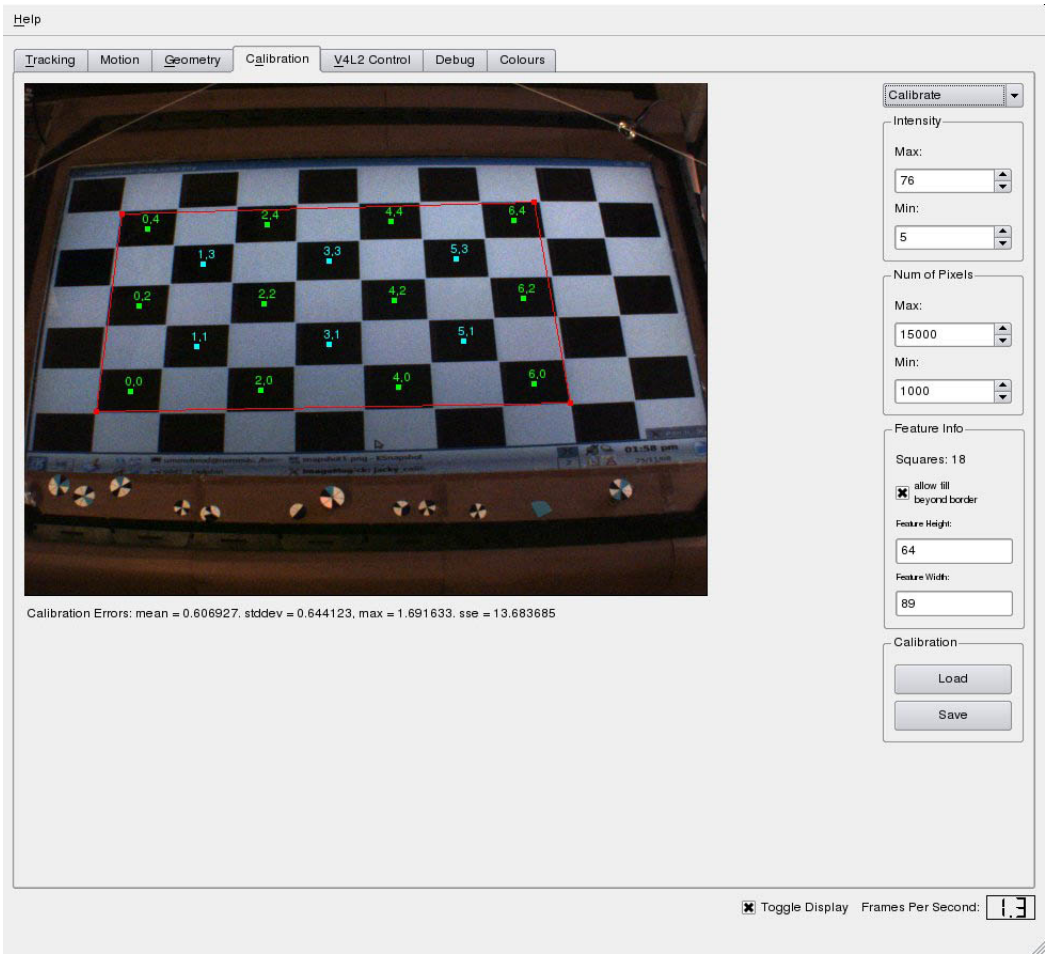
**Figure 6. Choosing Calibrate**




Now everything is set for the system to calibrate itself. Go to the top right of the tab, where you will see the label **setup**. This is a drop-down menu, and clicking it will allow you to choose *calibrate* (Figure 6), which will perform a calibration based on the values you have selected and entered.

Once the calibration is complete, you will see a grid of numbers labelling the X and Y locations of the rectangles that have been calibrated (Figure 7). You should see that they begin at **0 0** in the lower left corner (they appear to the upper right of each recognized rectangle), increase by one as you go right (or up), and are alternating blue and green. If the numbers don't alternate blue and green, if some are missing, or if they do not increase by one for each row or column, you will need to perform the calibration again to get a more accurate recognition. Slightly altering the rotation of the camera or the handles you originally chose for the calibration area may help this.

**Figure 7. Calibration Quality**




Along the bottom of the image is a set of statistics measuring the quality of the calibration. The value labeled *max* indicates the maximum number of millimeters of error (in theory) that any point in the calibrated area will have. The example value shown in Figure 7 (less than 2) is very small, but this is partly because it is a small (40") field and a close-set camera: the scale of values would differ significantly if this were a 3m field width and camera distance. If the value is relatively poor given your field size and camera distance, try the calibration again, adjusting the many options (not only in the system, but including camera placement and calibration image/carpet size) described previously. In any case, the maximum error value should be as small as you can possibly get it (but it will never be zero).

When you are happy with your calibration, you need to save it via the **Save** button in the lower portion of the menu. There is a delay when saving when saving in calibration mode (especially with the large volume of data provided by a firewire camera) because this causes the system to re-calibrate before saving. Saving a calibration you are happy with will be much faster if you use the **set up/calibrate** drop-down menu to move back to **set up** mode first. So, make that menu choice and then click the **Save** button. When you choose Save, you will get a default filename of `calibration_points.dat`. You can save the calibration as whatever name you want (and load one of many existing calibrations you've made

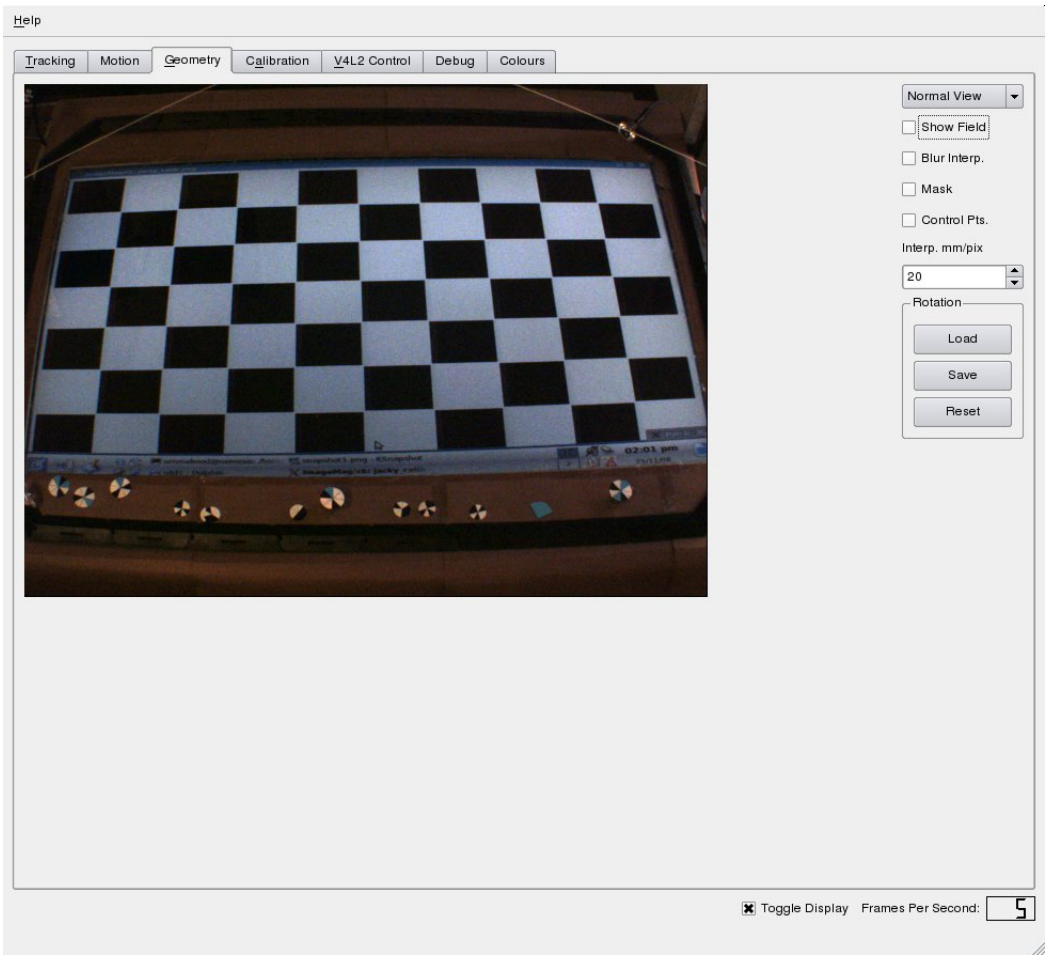
through the **Load** button), but the default calibration the system will look for is the filename specified in the `ergo.conf` file, which is this default name. Edit `ergo.conf` if you wish to have a different default calibration points file loaded automatically.

## 4.2. Calibrating the Rotation Matrix and Associated Geometry

In conjunction with the defined calibration points you have just created, you must also create a rotation matrix that completes the definition of a coordinate plane for the visual area. Managing rotation separately from the camera position allows small field rotations to be corrected independently from major camera positioning changes and keeps the overall calibration process simpler.

To define a rotation matrix, click on the **Geometry** tab. Figure 8 shows the initial display that will result.

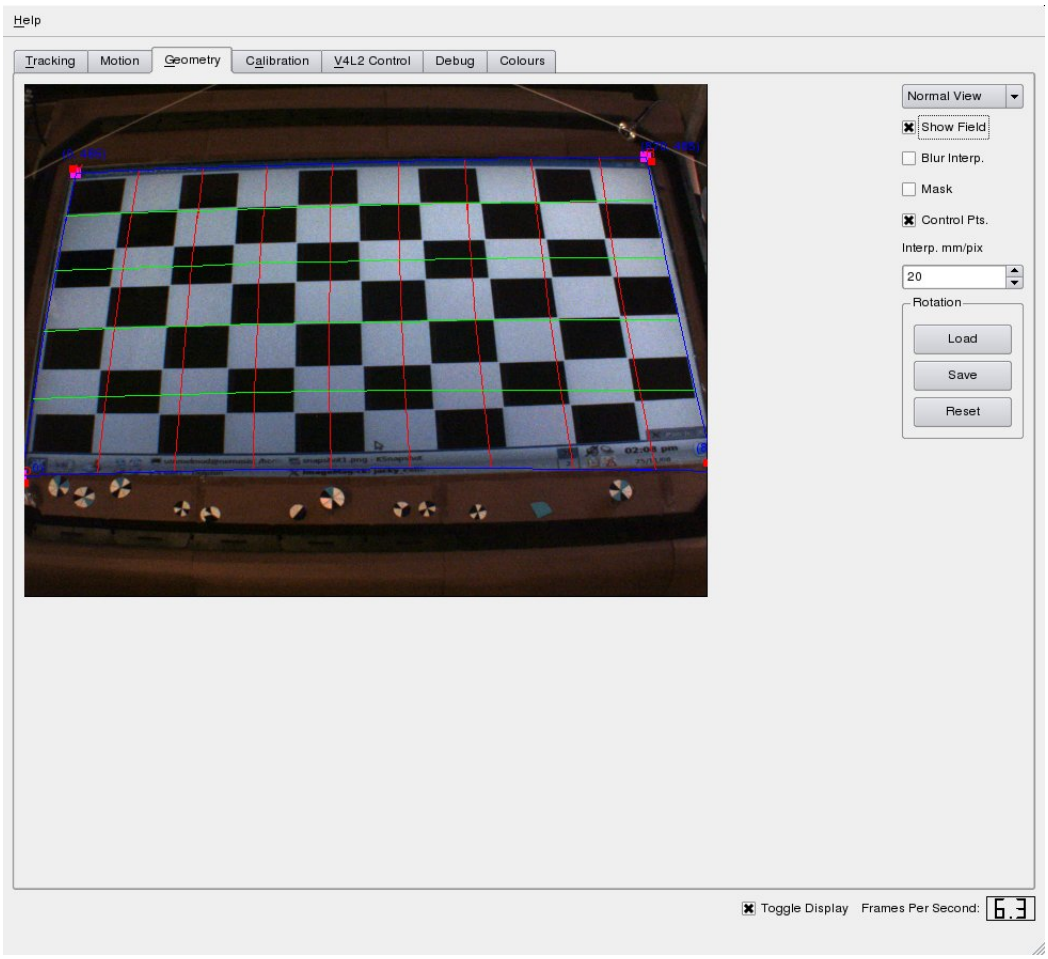
**Figure 8. Geometry Tab**




Clicking on the *Show Field* and *Control Pts.* check boxes, respectively, will show the existing rotation

matrix and the handles for resizing this. Each handle is in fact a pair of markers: a red marker that is draggable for resizing, and a pink following marker that indicates the calculated position of the rotation matrix extrapolated to the plane of the playing field. Drag each of the handles to the appropriate corner of the actual playing field in the visual image. You will end up with something that looks like Figure 9. If you are unhappy or otherwise wish to try again, click the **Reset** button and try again (you need to do a reset each time you change the rotation matrix).

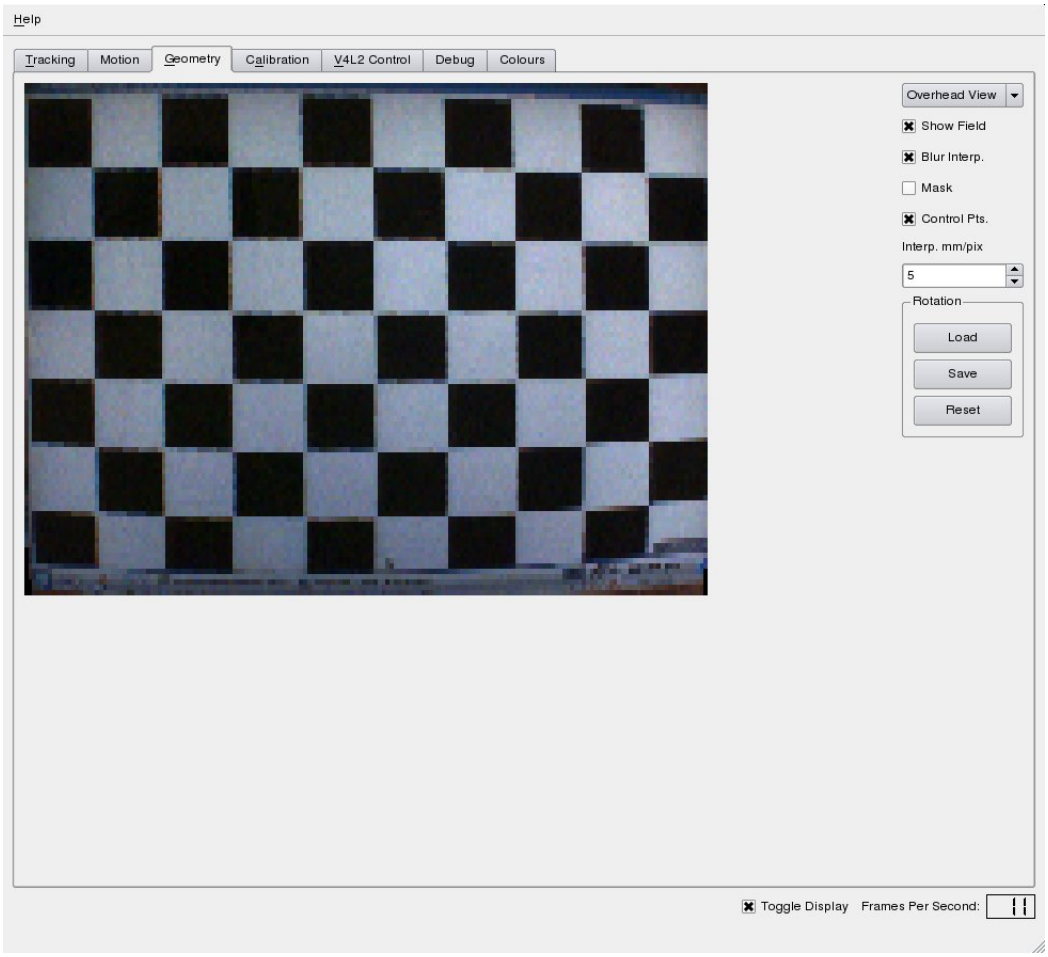
**Figure 9. The Aligned Rotation Matrix**




**Normal View** in the upper-right of the window is a drop-down menu, allowing you to choose **Overhead View** as an alternative. Choosing this shows you a real-time interpolated image (i.e. a mathematical translation, using your calibration, of what the displayed field image would look like if were to be viewed perfectly overhead). Such an interpolation is shown in Figure 10. The interpolation will look much coarser than this by default, because the area represented by any pixel will be a reasonably large value. In Figure 10, the *Interp. mm/pixel* has been lowered to a value of 5 from 20, and the *Blur Interp.* option has been selected to smooth it to a near-perfect view. Yours should look similarly good under these same settings. The framerate will drop drastically when you do this, since the interpolation is much more detailed. Changing this parameter here changes only how this interpolation appears momentarily, so that you can check if any apparent inaccuracy is due to a bad calibration/rotation matrix, or if it simply

appears that way because of a low-resolution interpretation. Changing the resolution of the interpretation used internally is also possible (and sometimes necessary), but must be done by editing the configuration file (Section 5.1).

**Figure 10. Interpolated Overhead View**

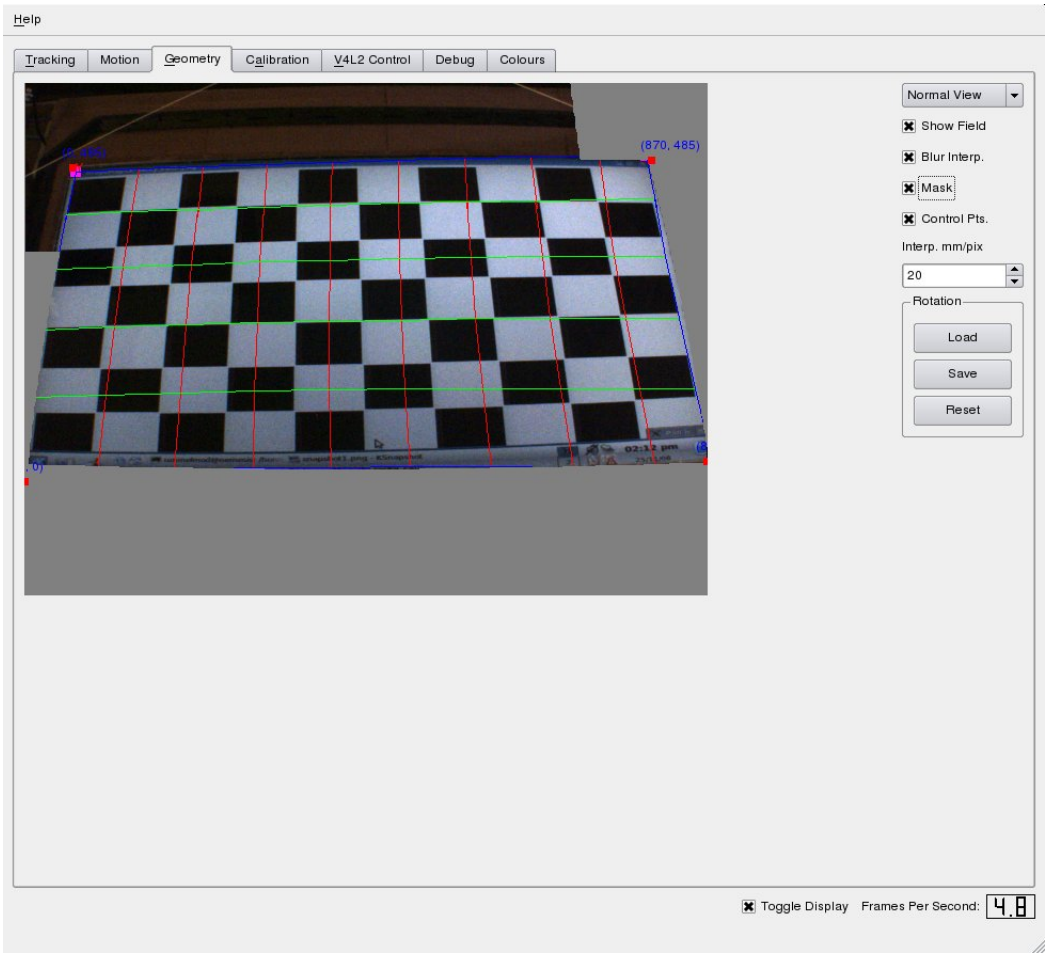



The quality of the interpolated image (outside of the blur and mm/pixel options) is a reflection of the quality of the calibration and can be used to judge if you are ready to save this (or if you want to further alter the rotation matrix). When you are satisfied, click the **Save** button. The default filename is `rotationmatrix.dat`. You can again choose any other filename you want, but the default rotation matrix will be loaded from this filename as specified in the `ergo.conf` file, from the same directory as `ergo.conf`. If you want this rotation matrix to be automatically loaded when you run Ergo, you will want to ensure the filename matches the specified name in `ergo.conf`.

There is one more option on this tab, a **Mask** checkbox. Now that you have defined the field area through the rotation matrix, the system can mask off parts of the image where objects will not have to be tracked, improving efficiency by ignoring these areas. This mask is not precisely the area you selected when specifying the rotation matrix, however. Since Ergo operates in 3-d, the height of the objects to be

tracked must be considered when setting this mask. Ergo obtains the maximum possible height from the descriptions of objects to be tracked in the `ergo.conf` file (see Section 5.1), and uses this to calculate a mask automatically. Clicking the **Mask** option displays this mask. You cannot edit this mask yourself, but viewing it is a good way to diagnose some recognition problems (e.g. if it is not seeing objects at the top of the field, perhaps you have mismeasured the height of the objects to be tracked, and these need to be altered in `ergo.conf`). The mask for the calibration we have been performing in these example figures is illustrated in Figure 11.

**Figure 11. Automatically Calculated Mask**



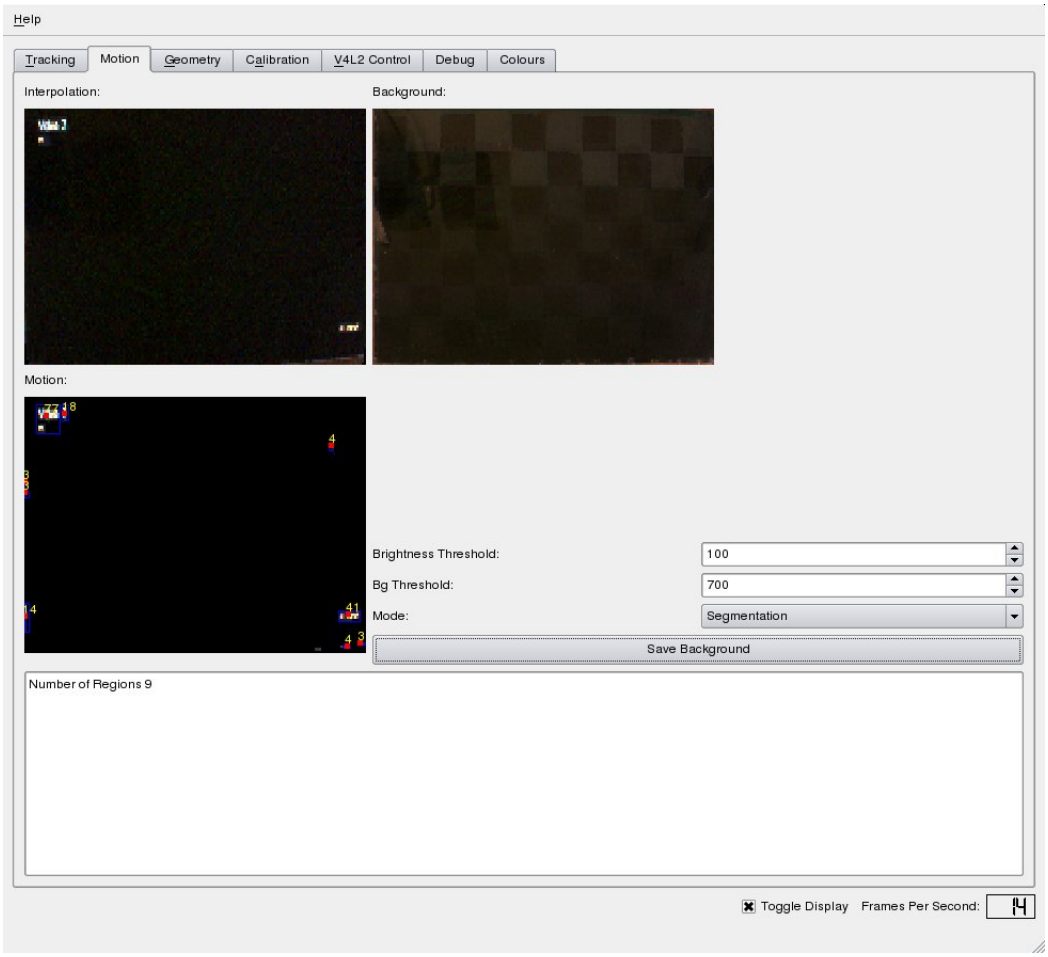

At this point your camera calibration is complete, and you can put away your calibration image/carpet.

## 4.3. Gathering a Background

This version of Ergo uses background differencing (to determine motion) as the primary means of tracking objects. Now that you have the camera geometry calibrated, you must capture the background that will be used as a base to distinguish motion (if the system seems inaccurate after some time and

appears to need re-calibrating, this is the simplest and most common starting point, unless you know the camera or the field has been shifted). Begin by clicking the **Motion** tab. You will see a display like that in Figure 12.

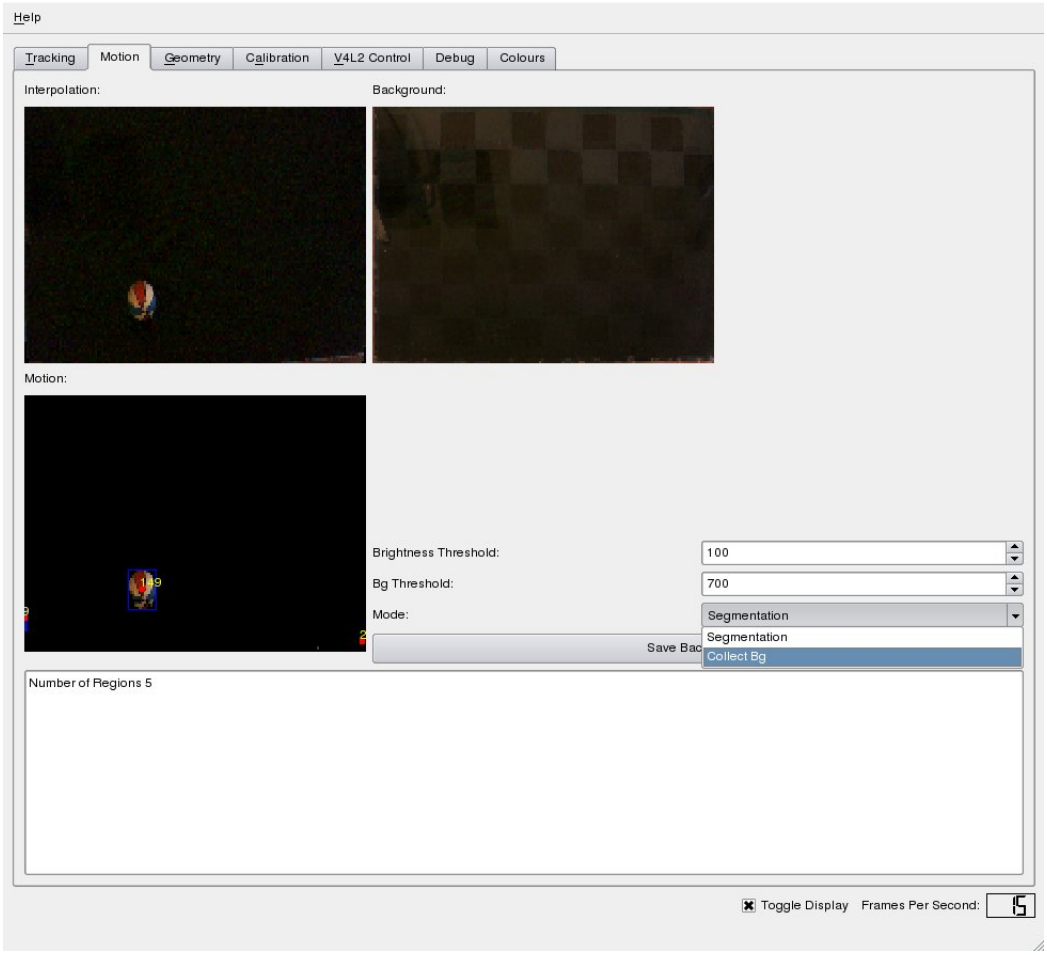
**Figure 12. The Motion Tab**




There are three image areas on this display. The upper left shows the interpolated overhead view of the field (this is just dark in the image because the calibration image has been removed and the screen is black). To the right of this is the current background, which is black because no background has been set. Below the interpolated image is the motion that is observed (numbered regions) given the currently set background and camera capture. This initially just noise, since the background is void. Below these image areas is a text box displaying the total number of regions of motion detected.

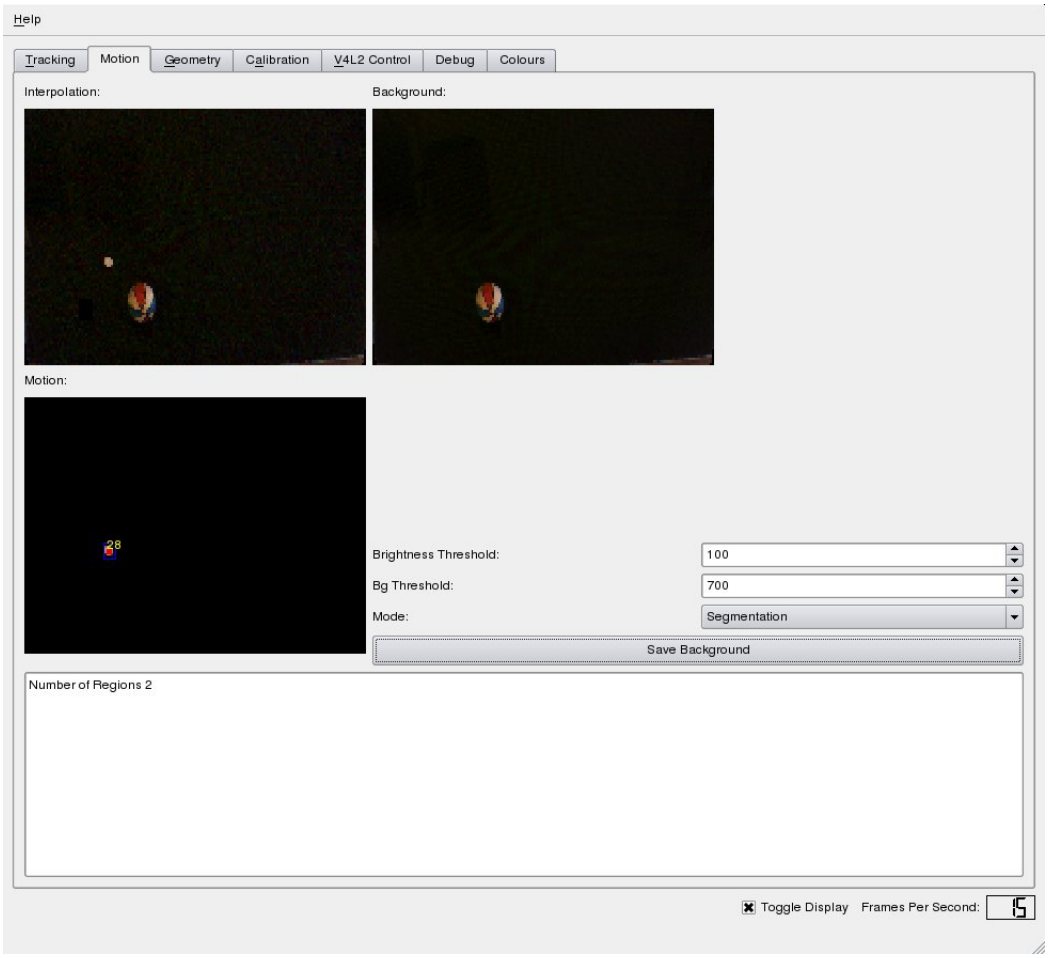
Now you will want to gather a background. Remove all the objects to be tracked and anything else that might move during the course of play (i.e. ensure the camera is capturing only what you want to define as the legitimate background). Then click the **Mode** menu (currently showing *Segmentation*) and choose *Gather BG* as shown in Figure 13 to move to background gathering mode, and click **Save Background** to save a background image.

**Figure 13. Gathering a Background**



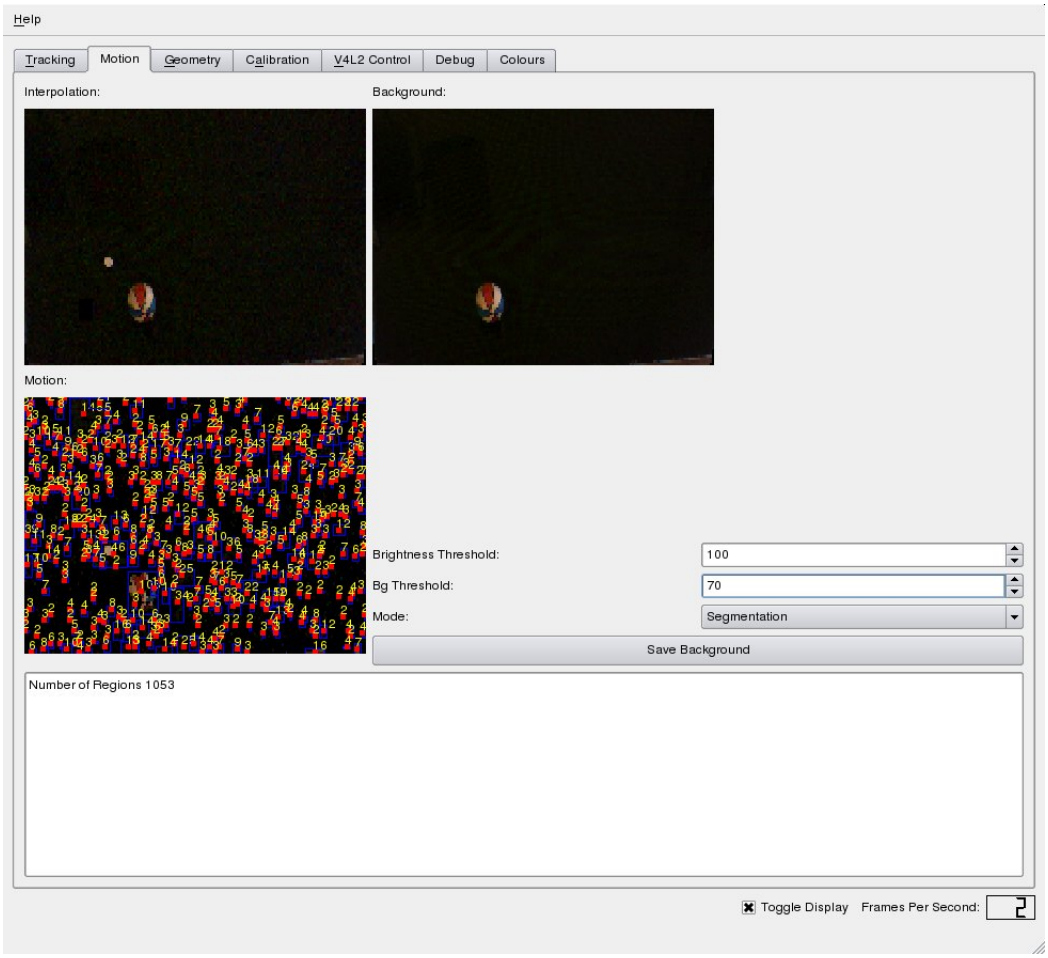

In Figure 13, a large multicoloured ball has been placed in the field of view here to be gathered as part of the background, just to show up against the dark screen - it is highly unlikely you would want such a ball as part of your actual background, since it would be prone to movement). Now that you have captured a background, you can then choose *Segmentation* from the **Mode** menu to get out of background gathering mode. You will see that the image you captured is now in the labeled background area. If you place any new objects in the camera field, they will show up as potential areas of motion because of background differences. In Figure 14, we placed a small ball to the upper right of the ball that was gathered as part of the background, and two separate regions are shown as possible areas of motion because of it.

**Figure 14. Background Differencing**



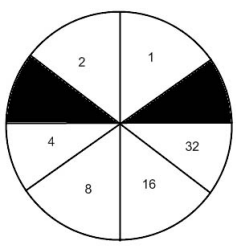

There are two parameter values you can set on this tab, both of which affect sensitivity of object recognition. For an area of the captured image to be considered as a tracked object, it must be different from the background, and have pixels that are non-black. The **Bg Threshold** parameter directly sets the sensitivity for determining how different something must be to be considered a background difference. Altering this to too high a value will cause (potentially a great deal of) noise to be considered as potential motion based on background differences (e.g. Figure 15 shows 1053 regions of motion given the same camera capture shown in Figure 14). Similarly, the **Brightness Threshold** defines the limit at which pixels are considered non-black for the purposes of background differencing. Making this value too sensitive will introduce similar noise.

**Figure 15. Too High a Background Sensitivity**



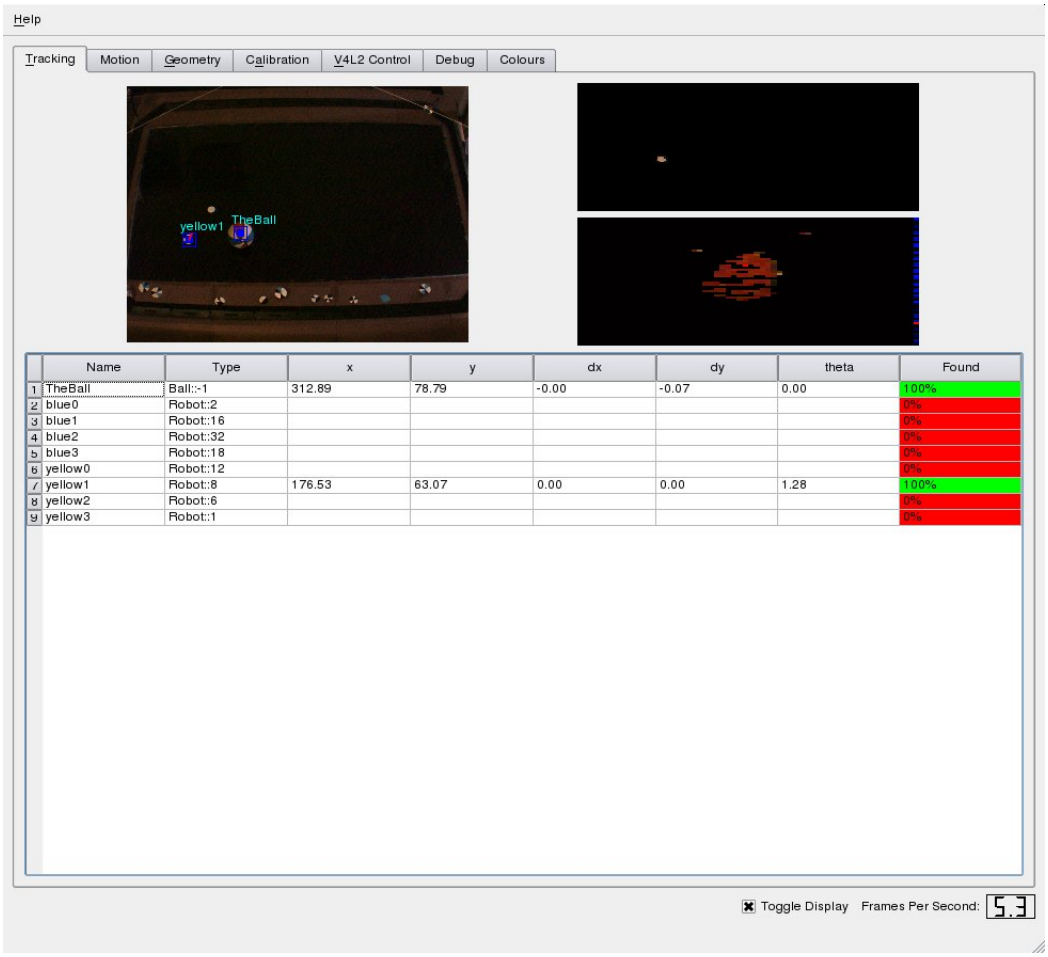

At this point you can now recognize robots based on background difference and motion. Ergo uses a circular marker placed on top of each robot (commonly called a hat). This has two black wedges that are used to determine orientation, and wedges of white and *any colour other than black and white* to represent zeroes and ones in wedges that correspond to powers of 2, forming an id value for the robot. The format of this hat (and the wedges corresponding to each power of 2) is shown in Figure 16.

**Figure 16. Ergo Hat**



The non-black and non-white colour that is used requires no calibration (it is recognized based on noting differences from white and black), and so simply placing an example hat on the playing field (corresponding to a robot that Ergo is interested in tracking, see Section 5.1) should allow recognition at this point. The result of placing such a hat somewhere in the field and then clicking the **Tracking** tab is shown in Figure 17

**Figure 17. Recognizing a Robot**




Here, the robot with a coloured wedge in position 8 and white wedges everywhere else (other than the fixed black orientation markers) corresponds to the identity *yellow1* in a line in the `ergo.conf` file. Seven other robots are also described in this file, producing seven other entries for robots to be tracked (all of which are unrecognized). The right hand side of the display shows a percentage estimating the quality of recognition of each individual object, set in an appropriately-coloured bar. Each recognized object also displays its X and Y locations on the field, any change in motion in the X and Y dimensions, and its orientation angle theta.

In the event you place a robot or robot hat on the field and it is not recognized, there are a number of things to check. First, you will want to check that the hat you have placed does indeed correspond to one

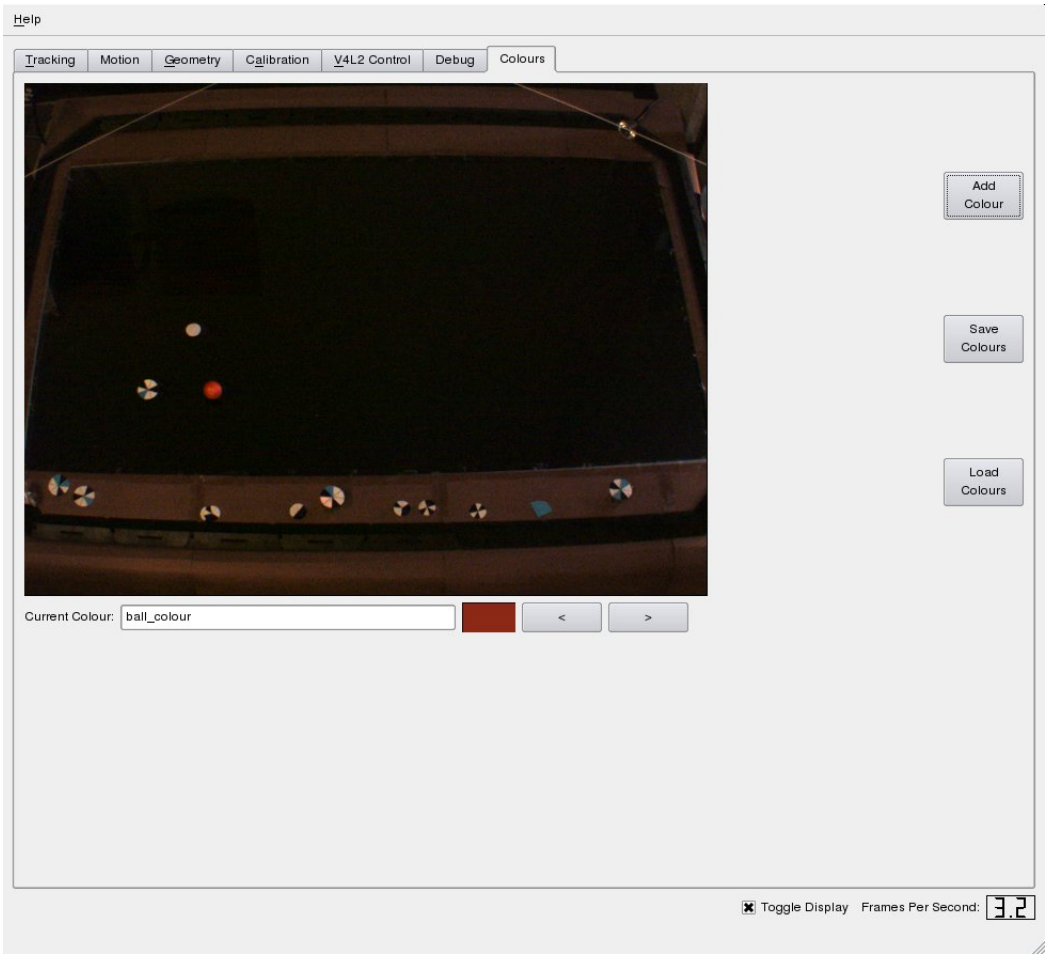
of the entries in the `ergo.conf` file. That is, you will need to double check that the coloured markings on the hat wedges actually do match those that Ergo is looking for. The structure of how these descriptions are laid out in the `ergo.conf` file is described in Section 5.1. Assuming the hat pattern is correct, you will then want to consider the global resolution setting for the interpolation. It may be that given the size of the hat image in the captured visual stream, the interpolation Ergo performs is not of a high enough resolution for the wedges in the hat to be discernable in the interpolated image. You should be able to discern these in the interpolated image yourself, as well. If the hat wedges are not discernable in the interpolated image, there is a setting in the `ergo.conf` file that can be edited to increase or decrease interpolation resolution (see Section 5.1).

Note that the figure also shows a second object (*TheBall*), also recognized and highlighted in the camera image as a portion of the large ball we placed in field when gathering the background. Recognizing the ball is done by colour rather than pattern (since a moving ball has no easily-discernable pattern), and in this case one of the stripes in the ball we placed was an appropriate size and colour match to approximate what is intended to be a ball given the description in the `ergo.conf` file. To get the system to recognize the intended ball, we need to calibrate the colour we intend to use for it, which is the last step in the calibration process.

## 4.4. Colour Calibration

In a soccer application, the ball is the only colour we care to recognize directly (though in other applications there are many other potential uses for tracking colour). These exceptions are the only need for colour in Ergo, and so this step is left for last. To perform colour calibration, click the **Colours** tab. You will see a display similar to that in Figure 18.

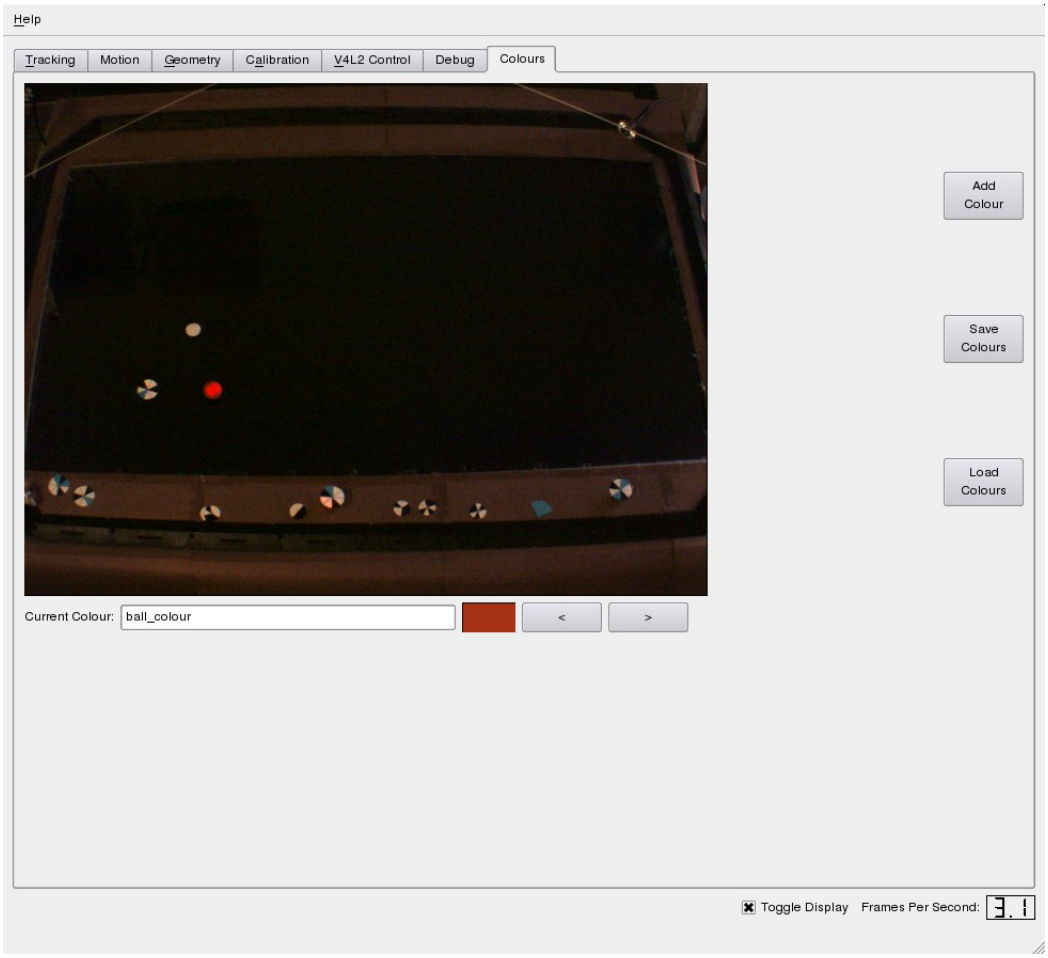
**Figure 18. Colours Tab**




Any application will use a set of zero or more named colours. These names and their values are ultimately stored in a file that you are creating using this tab. The **Add Colour** button lets you define and name a new colour, and the buttons labeled with less-than and greater-than signs let you step through the currently defined colours. At this point, the current set of colours will have been automatically loaded using the filename specified in the `ergo.conf` file, which includes a ball colour in the sample configurations provided. For Figure 18, we substituted the large multicoloured ball previously used to illustrate background differencing with a smaller orange ball that we want to have tracked. **ball_colour** has been selected as the colour to calibrate (just rotate through the defined colours if it is not there, and if you don't find it, check that you are using one of the sample `ergo.conf` files provided with Ergo).

To change the definition of a colour, left-click to add the current pixel to the colour's range, and right-click to reset the colour. Figure 19 illustrates the range of colour occupied by the ball we want to track gathered through successive left-clicks.

**Figure 19. Selecting a Colour Range**

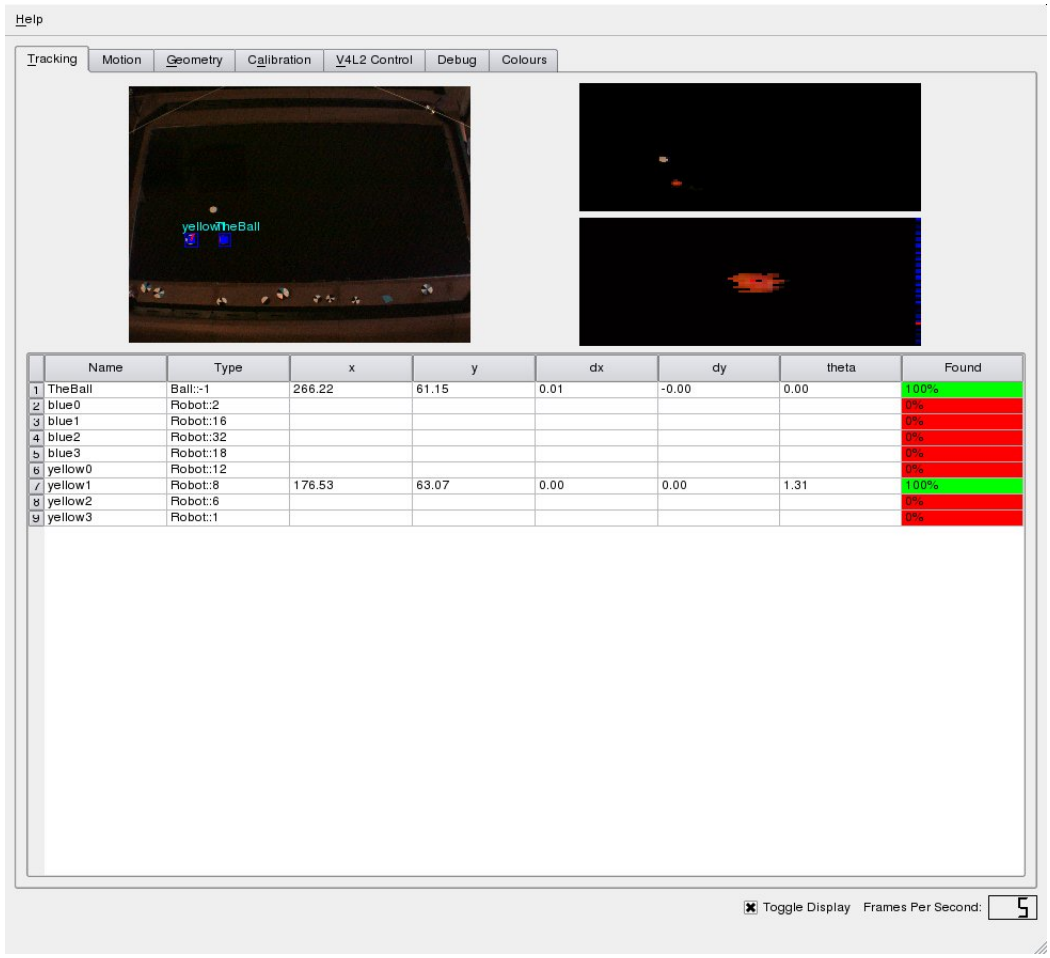



Just like the other areas of calibration, you need to save the defined colours once you are happy adding to or changing them. You will get a default filename (`colours.dat`) that matches the filename that will be loaded automatically, as specified by your `ergo.conf` file. You can again change this to whatever you want, but you will need to edit `ergo.conf` to reflect that new filename if you want it automatically loaded when you run Ergo.

# 5. Using Ergo

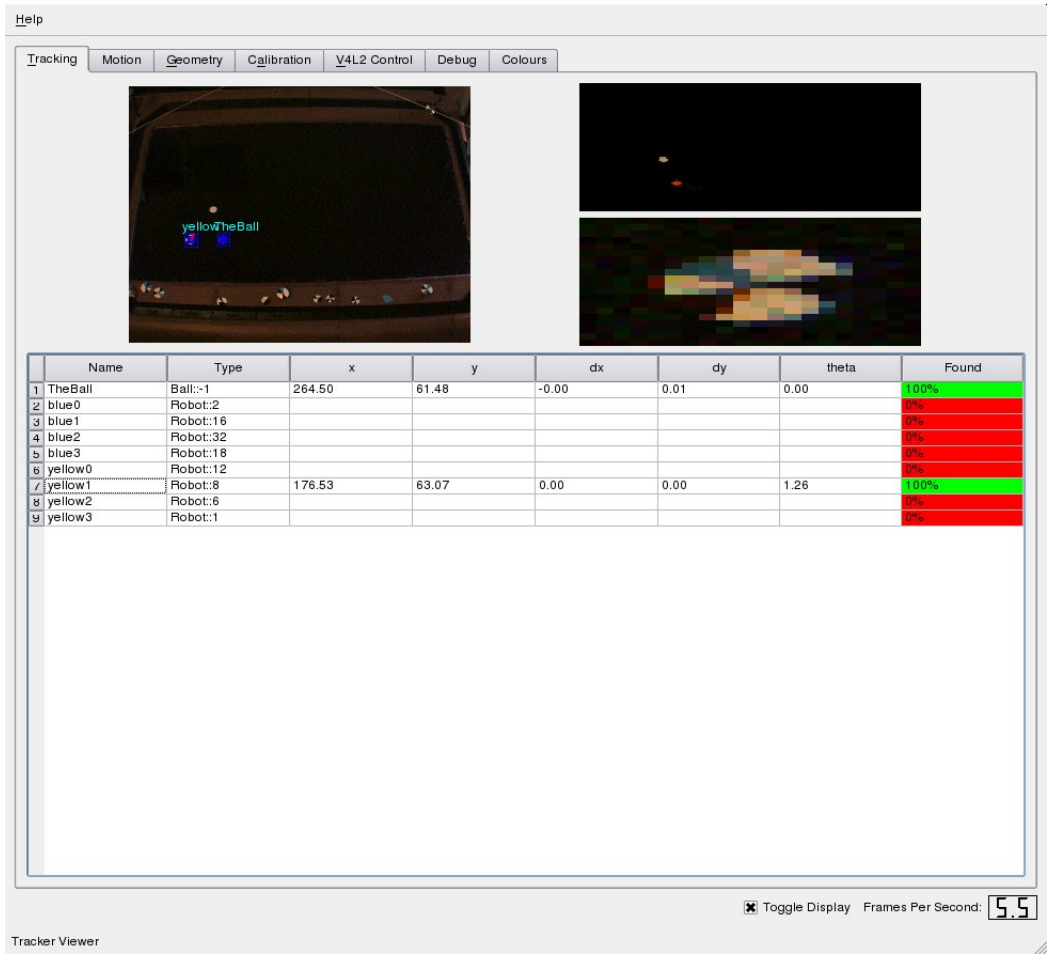
You have now performed all the necessary calibrations and are ready to run Ergo with your application. Click back to the **Tracking** tab and you will see the new ball being properly recognized following the calibration of its colour, as shown in Figure 20.

**Figure 20. Successful Ball Recognition**



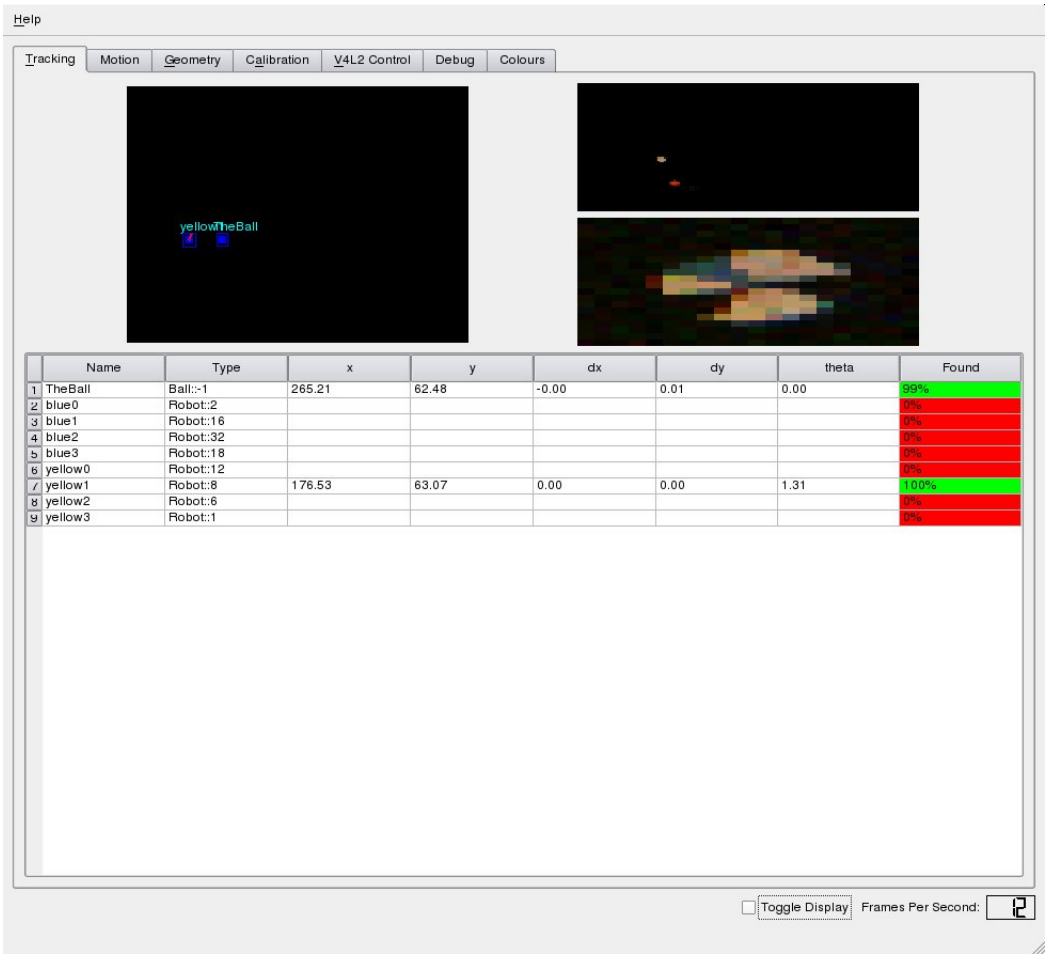

To get a closer look at any object, simply click on it and it will be displayed in a magnified and interpolated form in the lower-image on the right hand side. This is useful for diagnosing recognition problems if they occur. For example, clicking on the robot being tracked moves its magnified image into this area (Figure 21).

**Figure 21. Zooming in on an object**




One other useful feature is the ability to stop the system from updating the display. This can save system resources and greatly increase the achievable frame rate. In the bottom right of the Ergo window (on any tab), you will see a **Toggle Display** checkbox. Clicking this stop the system from updating the camera display, but in this case more than doubles the frame rate of the system (Figure 22). Note that in this case, the frame rate is low to begin with because of the high-resolution of the firewire camera - using a lower resolution will allow for much faster frame rates than are typical of the images you see here.

**Figure 22. Toggle Display**




## 5.1. Ergo Messages and Editing ergo.conf

Ergo runs by tracking objects and broadcasting descriptions of these via ethernet. These messages can then be picked up by client programs interested in the visual feed. Each message from Ergo will be similar to the following example:

```
9 12697 91979 870 485 0
-1237.12 1280.55 658.859
1 TheBall F 265.631 61.6534 20 0 0.00678911 -0.0113096
0 blue0 N 0 0 20 1.34827 0.000270165 0.000376505
0 blue1 N 0 0 20 0 0 0
0 blue2 N 0 0 20 0 0 0
0 blue3 N 0 0 20 0 0 0
0 yellow0 N 0 0 20 0 0 0
0 yellow1 F 178.2 63.7678 20 1.34827 0 0
0 yellow2 N 0 0 20 0.287979 0.00626374 0
0 yellow3 N 0 0 20 1.58389 0.00542921 0.00285079
```

Here, the first integer on the first line represents the number of objects tracked, corresponding to the number of lines to be expected beyond two-line header. The other integers on the first line represent, respectively, the frame number, the time difference from the previous message, the field width, the field length, and the field height. The next line contains three values, representing the X, Y, and Z, coordinates of the camera, extrapolated with respect to the 0,0 point in the calibration.

Following this, each line represents one object. The first integer on the line represents the type (1 = ball, 2 = robot, 3=spot/obstacle), the second item is the symbolic name of the object, and this is followed by a flag indicating whether the object was found or not (each object may not be found in every frame). Following this are the X, Y, and Z coordinates of the object, and then the angle of orientation in radians from the X axis. Following these are the velocity of the object in the X and Y dimensions.

The `ergo.conf` file defines additional parameters beyond those set through the interface and described in this document. One of the most critical of these is the resolution of the interpolated image used by Ergo in tracking objects. The actual process of creating an interpolation and using it is described in a number of papers on Ergo (available from the Autonomous Agents Laboratory's website) that are well beyond the scope of this document. Suffice it to say, however, that constructing an interpolated image in real time can use up a great deal of system resources. The resolution of the interpolated image, in terms of the size of the physical world that is encompassed by a pixel, greatly affects this. This same density also greatly affects the size of objects that can be recognized by Ergo: lowering the resolution will make the interpolation process more efficient, but will also mean that any patterns to be recognized (most importantly, the wedges in a robot hat, but also the size of a round object such as the ball) will have to be large enough that the pattern will still be visible at that resolution. The `ergo.conf` file has a **MM_PER_PIX** parameter that controls the resolution of the interpolation used internally (this is distinct from the parameter on the **Geometry** tab, which is used only for displaying the interpolation to check the accuracy of the rotation matrix and other calibration elements, and does not affect the actual resolution used when tracking objects). The smallest valid value to use here is 1 mm/pix, but you will see that lowering the default value significantly affects the framerate, since the interpolation is much more fine-grained. For our work with microrobots (approximately 1.5" across, running on a 42" surface with the camera approximately the same distance away), we typically use a resolution of 5 mm/pixel. The suitable value for any application depends on the size of the objects to be recognized in the camera image, which is affected by both the scale of the object, the distance of the camera, and the focal length of the lens employed. You want to have a value no smaller than necessary, for reasons of efficiency. If the value you choose is too large, the robot hats will not be recognized when they are placed on the field (Section 4.3), and you can then decrease this value until recognition is achieved.

The `ergo.conf` file also describes the objects to be tracked, and to alter Ergo for your own environment the next step is to edit this file to describe your own objects. You will see that the file consists of a set of XML tags, and in particular, one section looks as follows:

```
<Tracker>
    # The list of items to be tracked.
    # types are Robot, Ball, or Spot
    # case sensitive.
    #
```

```
mm_per_pixel ${MM_PER_PIX}
<Trackable>
    type Ball
    name TheBall
    height ${BALL_HEIGHT}
    radius ${BALL_RADIUS}
    colour ball_colour
</Trackable>
<Trackable>
    type Robot
    name blue0
    height ${BOT_HEIGHT}
    subtype 2
    radius ${BOT_RADIUS}
</Trackable>
<Trackable>
    type Robot
    name blue1
    height ${BOT_HEIGHT}
    subtype 16
    radius ${BOT_RADIUS}
</Trackable>
...
```

Many of these items refer to constants also described in this file. For example, you can edit the **BALL_HEIGHT** or **BOT_HEIGHT** to appropriate values for the size of objects you are using. The **subtype** specified for any robot defines the unique integer value that its hat pattern should have. For example, subtype 6 should haveO coloured wedges in the spaces labeled 4 and 2 in Figure 16. With fixed or moving spots, robots, and a ball as built-in trackable object types, you can define a wide range of environments. For example, a ball doesn't need to be a physical ball, just a circular area of one color. To move beyond this, you can explore Ergo's source code, examine the definitions of these object types, and extend them to new types of your own devising.

# 6. Copyright and License



## 6.1. GNU Free Documentation License

Version 1.1, March 2000

### 6.1.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 6.1.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

### 6.1.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

### 6.1.4. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

### 6.1.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher

of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

### 6.1.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

### 6.1.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

### 6.1.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

### 6.1.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License

provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

### 6.1.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

### 6.1.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

### 6.1.12. How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) 2008 John Anderson and Jacky Baltes. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.