Hyperbolic Position Location on Eurobot table

The Green robot

The green robot is also referred as "robot" or "our robot". The green robot carries the receiver beacon (beacon number 4).

In[1]:=
```
robot = {0.8, 1.5};
```

The Red robot

The red robot is also referred as the "enemy". This is the robot we are attempting to locate. The red robot carries the transmitter beacon.

In[2]:=
```
enemy = {1.2, 0.9};(* Red Robot position *)
```

Repeater Beacons

Repeater Beacons are modules which repeat a pulse when one is received. Their positon is fixed on the sides of the table. Minimum number of beacons for succesfull positioning is two. It is possible to place up to three beacons on the sides of the Eurobot table. Using three beacons makes the positioning system more robust and more accurate.

Beacon coordinates:

In[3]:=
```
bcng1 = {3.124, 0.0};(* Green Beacon 1 *)
bcng2 = {0.0, 1.100};(* Green Beacon 2 *)
bcng3 = {3.124, 2.212}; (* Green Beacon 3 *)
bg = {bcng1, bcng2, bcng3, robot};
% // MatrixForm

bcnr1 = {0.0, 0.0};(* Red Beacon 1 *)
bcnr2 = {3.124, 1.100};(* Red Beacon 2 *)
bcnr3 = {0.0, 2.212}; (* Red Beacon 3 *)
br = {bcnr1, bcnr2, bcnr3, enemy};
```

Out[7]//MatrixForm=

$$\begin{pmatrix} 3.124 & 0. \\ 0. & 1.1 \\ 3.124 & 2.212 \\ 0.8 & 1.5 \end{pmatrix}$$

How it works

A transmitter beacon is a beacon which transmits short ultrasound pulses (i.e. 2ms) at constant intervals (i.e. 200ms). The repeater beacons repeat these pulses. Receiver beacon receives all the pulses.

In this kind of setup a pulse has multiple paths to get to the receiving beacon. The shortest path is the straight line from the transmitter to the receiver, this is the first pulse to arrive to the receiver. We define this moment as $t_0 = 0$.

After t_0, pulses from beacons 1 through 3 arrive. We define these moments t_1, t_2 and t_3 respectively and call them the difference of time of arrival - or DTOA for short.

We multiply DTOA values by the speed of the signal $c_{\text{sound}} = 343\ \frac{m}{s}$ to get the distances ct_1, ct_2 and ct_3. We use these measured distances to calulate the Hyperbolic Position Location of the transmitting beacon.

It is necessary to know the location of the receiving beacon when calculating the location of the transmitting beacon. It is also possible to calculate the location of the receiving beacon, but then the location of the transmitting beacon must be known. The locations of the repeater beacons must always be known.

Following calculations give us accurate values for ct_1, ct_2 and ct_3. We use these calculated values to test our equations.

```
In[12]:= ct1 = 1.0 * (Norm[bcng1 - enemy] + Norm[bcng1 - robot] -
           Norm[enemy - robot] + RandomReal[NormalDistribution[0, 0.05]] * 0);
         ct2 = 1.0 * (Norm[bcng2 - enemy] + Norm[bcng2 - robot] - Norm[enemy - robot] +
           RandomReal[NormalDistribution[0, 0.05]] * 0);
         ct3 = 1.0 * (Norm[bcng3 - enemy] + Norm[bcng3 - robot] - Norm[enemy - robot] +
           RandomReal[NormalDistribution[0, 0.05]] * 0);
```

Mathematic solution

A hyperbola is defined as the locus of points $p = (x, y)$ where the difference of the distances to the focal points b_i and b_j is constant d_{ij}.

$$d_{\text{ij}} = \|b_i - p\| - \|b_j - p\| = \sqrt{(x_i - x)^2 + (y_i - y)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2} \tag{1}$$

By squaring equation (1) we get the relationship

$$d_{\text{ij}}^2 + \left(x_j^2 - x_i^2\right) + \left(y_j^2 - y_i^2\right) + \left(2\,x_i - 2\,x_j\right)\cdot x + \left(2\,y_i - 2\,y_j\right)\cdot y = -2\,d_{\text{ij}}\cdot\sqrt{(x_j - x)^2 + (y_j - y)^2}$$

where

$$\lambda_{\text{ij}} = d_{\text{ij}}^2 + \left(x_j^2 - x_i^2\right) + \left(y_j^2 - y_i^2\right) \tag{2}$$

Two hyperbolae that share a focal point, b_j, give the hyperbolic relationship

$$\begin{cases} \lambda_{ij} + 2\left(x_i - x_j\right)\cdot x + 2\left(y_i - y_j\right)\cdot y = -2\, d_{ij}\cdot\sqrt{\left(x_j - x\right)^2 + \left(y_j - y\right)^2} \\ \lambda_{kj} + 2\left(x_k - x_j\right)\cdot x + 2\left(y_k - y_j\right)\cdot y = -2\, d_{kj}\cdot\sqrt{\left(x_j - x\right)^2 + \left(y_j - y\right)^2} \end{cases}$$

Subtracting these equations give the relationship

$$2\left[d_{kj}\left(y_i - y_j\right) - d_{ij}\left(y_k - y_j\right)\right]\cdot y + \; 2\left[d_{kj}\left(x_i - x_{ij}\right) - d_{ij}\left(x_k - x_j\right)\right]\cdot x + \; d_{kj}\,\lambda_{ij} - d_{ij}\,\lambda_{kj} = 0$$

We can see that the solution lies on a straight $y = m\cdot x + b$ where

$$m = \frac{d_{ij}\left(x_k - x_j\right) - d_{kj}\left(x_i - x_j\right)}{d_{kj}\left(y_i - y_j\right) - d_{ij}\left(y_k - y_j\right)} \tag{3}$$

and

$$b = \frac{d_{ij}\,\lambda_{kj} - d_{kj}\,\lambda_{ij}}{2\,d_{kj}\left(y_i - y_j\right) - d_{ij}\left(y_k - y_j\right)} \tag{4}$$

In[15]:=
```
d1 = ct1 - Norm[bcng1 - robot];
d2 = ct2 - Norm[bcng2 - robot];
d3 = ct3 - Norm[bcng3 - robot];

dr = {d1, d2, d3};

d12 = d1 - d2;
d13 = d1 - d3;
d21 = d2 - d1;
d23 = d2 - d3;
d31 = d3 - d1;
d32 = d3 - d2;

(* This is our measurement matrix *)
d = {{0, d12, d13, d1}, {d21, 0, d23, d2}, {d31, d32, 0, d3}, {-d1, -d2, -d3, 0}};
% // MatrixForm

(* These functions are for lines defined by
 two hyperbolae that share a focal point (point j). *)
lambda[i_, j_] := (d[[i, j]]^2 + bg[[j, 1]]^2 - bg[[i, 1]]^2 + bg[[j, 2]]^2 - bg[[i, 2]]^2);

m[i_, j_, k_] := (d[[i, j]] * (bg[[k, 1]] - bg[[j, 1]]) - d[[k, j]] * (bg[[i, 1]] - bg[[j, 1]])) /
   (d[[k, j]] * (bg[[i, 2]] - bg[[j, 2]]) - d[[i, j]] * (bg[[k, 2]] - bg[[j, 2]])) ;

b[i_, j_, k_] := (d[[i, j]] * lambda[k, j] - d[[k, j]] * lambda[i, j]) /
   (2 * (d[[k, j]] * (bg[[i, 2]] - bg[[j, 2]]) - d[[i, j]] * (bg[[k, 2]] - bg[[j, 2]])));

(* This function defines a distance difference of point p to points b1 and b2. Set
  this function constant to define a hyperbola i.e. ddist[b1, b2, {x,y}] == d *)
ddist[b1_, b2_, p_] := Norm[b1 - p] - Norm[b2 - p];
```

Out[26]//MatrixForm=

$$\begin{pmatrix} 0 & 0.907542 & -0.204665 & 1.40298 \\ -0.907542 & 0 & -1.11221 & 0.495442 \\ 0.204665 & 1.11221 & 0 & 1.60765 \\ -1.40298 & -0.495442 & -1.60765 & 0 \end{pmatrix}$$

Graphics

These are some graphics objects and colors for plotting.

In[31]:=
```
(* Official Colors *)
green = RGBColor[101 / 255, 140 / 255, 68 / 255];
red = RGBColor[166 / 255, 43 / 255, 30 / 255];
blue = RGBColor[6 / 255, 65 / 255, 106 / 255];
brown = RGBColor[51 / 255, 35 / 255, 30 / 255];

grobot = Graphics[{green, PointSize[0.02], Point[{robot}]}];
genemy = Graphics[{red, PointSize[0.02], Point[{enemy}]}];

(* The Table *)
gTableBorder =
  Graphics[{Transparent, EdgeForm[Thin], Rectangle[{0.062, 0.050}, {3.062, 2.150}]}];
gGreenStrart = Graphics[{Transparent, EdgeForm[{Thin, green}],
    Rectangle[{0.062, 1.650}, {0.562, 2.150}]}];
gRedStrart = Graphics[{Transparent, EdgeForm[{Thin, red}],
    Rectangle[{2.562, 1.650}, {3.062, 2.150}]}];
gCenter = Graphics[{brown, EdgeForm[{Thin, brown}], Circle[{1.562, 1.100}, 0.15]}];
gBeacon = Rectangle[{-0.04, -0.04}, {0.04, 0.04}];
gBeaconGreen1 = Graphics[{green, EdgeForm[Thin], Translate[gBeacon, bcng1]}];
gBeaconGreen2 = Graphics[{green, EdgeForm[Thin], Translate[gBeacon, bcng2]}];
gBeaconGreen3 = Graphics[{green, EdgeForm[Thin], Translate[gBeacon, bcng3]}];
gBeaconRed1 = Graphics[{red, EdgeForm[Thin], Translate[gBeacon, bcnr1]}];
gBeaconRed2 = Graphics[{red, EdgeForm[Thin], Translate[gBeacon, bcnr2]}];
gBeaconRed3 = Graphics[{red, EdgeForm[Thin], Translate[gBeacon, bcnr3]}];
gTable = Show[gGreenStrart, gRedStrart, gTableBorder, gBeaconGreen1, gBeaconGreen2,
   gBeaconGreen3, gBeaconRed1, gBeaconRed2, gBeaconRed3, gCenter, PlotRange → Automatic];
```

Position Location of the Enemy robot

Method 1

In method 1 distance d_1 is defined as the distance difference from robot to enemy and beacon 1 to enemy. Distances d_2 and d_3 are defined in the same way with their respective beacons. The distances 1...3 can be calculated using measured DTOA values and the distances from robot to beacons 1...3.

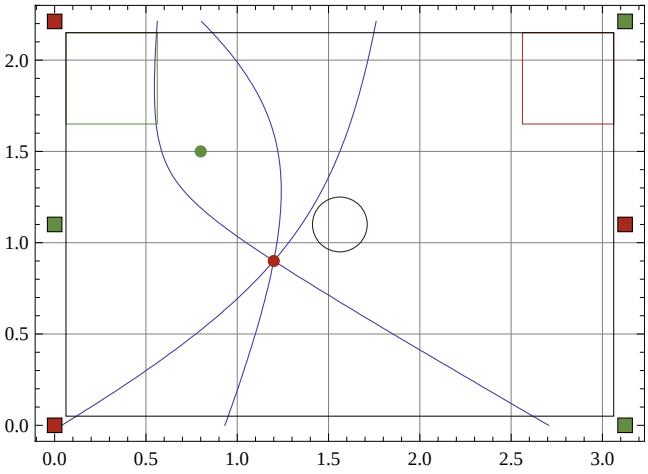
In[49]:=
```
d1 = ct1 - Norm[bcng1 - robot];
d2 = ct2 - Norm[bcng2 - robot];
d3 = ct3 - Norm[bcng3 - robot];
```

Contours 1...3 are the resulting hyperbolas (1). The location of the enemy robot is found on the intersection of these curves. In method 1 the other focal point of every hyperbola is the green robot and the other one is a beacon. The method 2 finds hyperbolas with focal points on beacons only.

In[52]:=
```
contour1 = ContourPlot[ ddist[bcng1, robot, {x, y}] == d1,  {x, 0, 3.124}, {y, 0, 2.212}];
contour2 = ContourPlot[ ddist[bcng2, robot, {x, y}] == d2,  {x, 0, 3.124}, {y, 0, 2.212}];
contour3 = ContourPlot[ ddist[bcng3, robot, {x, y}] == d3,  {x, 0, 3.124}, {y, 0, 2.212}];

Show[contour1, contour2, contour3, grobot, genemy, gTable,
  GridLines → Automatic , AspectRatio → Automatic, PlotRange → Automatic]
```

Out[55]=



Method 2

In method 2 we define hyperbolas with focal points on a pair of beacons only.

The distances are now:

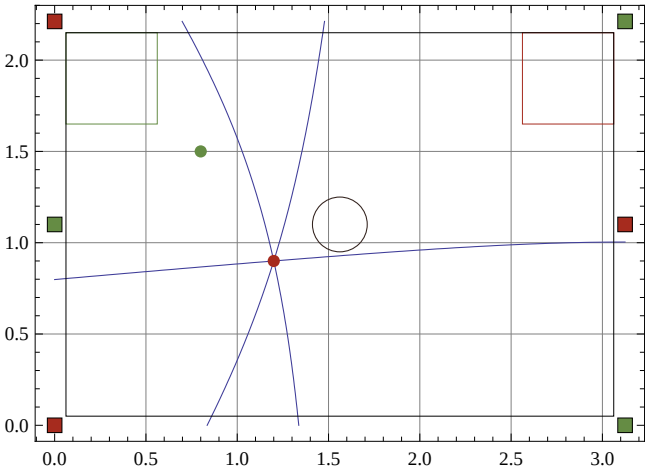
In[56]:=
```
d12 = d1 - d2;
d13 = d1 - d3;
d21 = d2 - d1;
d23 = d2 - d3;
d31 = d3 - d1;
d32 = d3 - d2;
```

And the hyperbolas (1):

In[62]:=
```
contour12 = ContourPlot[ ddist[bcng1, bcng2, {x, y}] == d12,  {x, 0, 3.124}, {y, 0, 2.212}];
contour13 = ContourPlot[ ddist[bcng1, bcng3, {x, y}] == d13,  {x, 0, 3.124}, {y, 0, 2.212}];
contour23 = ContourPlot[ ddist[bcng2, bcng3, {x, y}] == d23,  {x, 0, 3.124}, {y, 0, 2.212}];

Show[contour12, contour13, contour23, grobot, genemy, gTable,
 GridLines → Automatic, AspectRatio → Automatic, PlotRange → Automatic]
```

Out[65]=



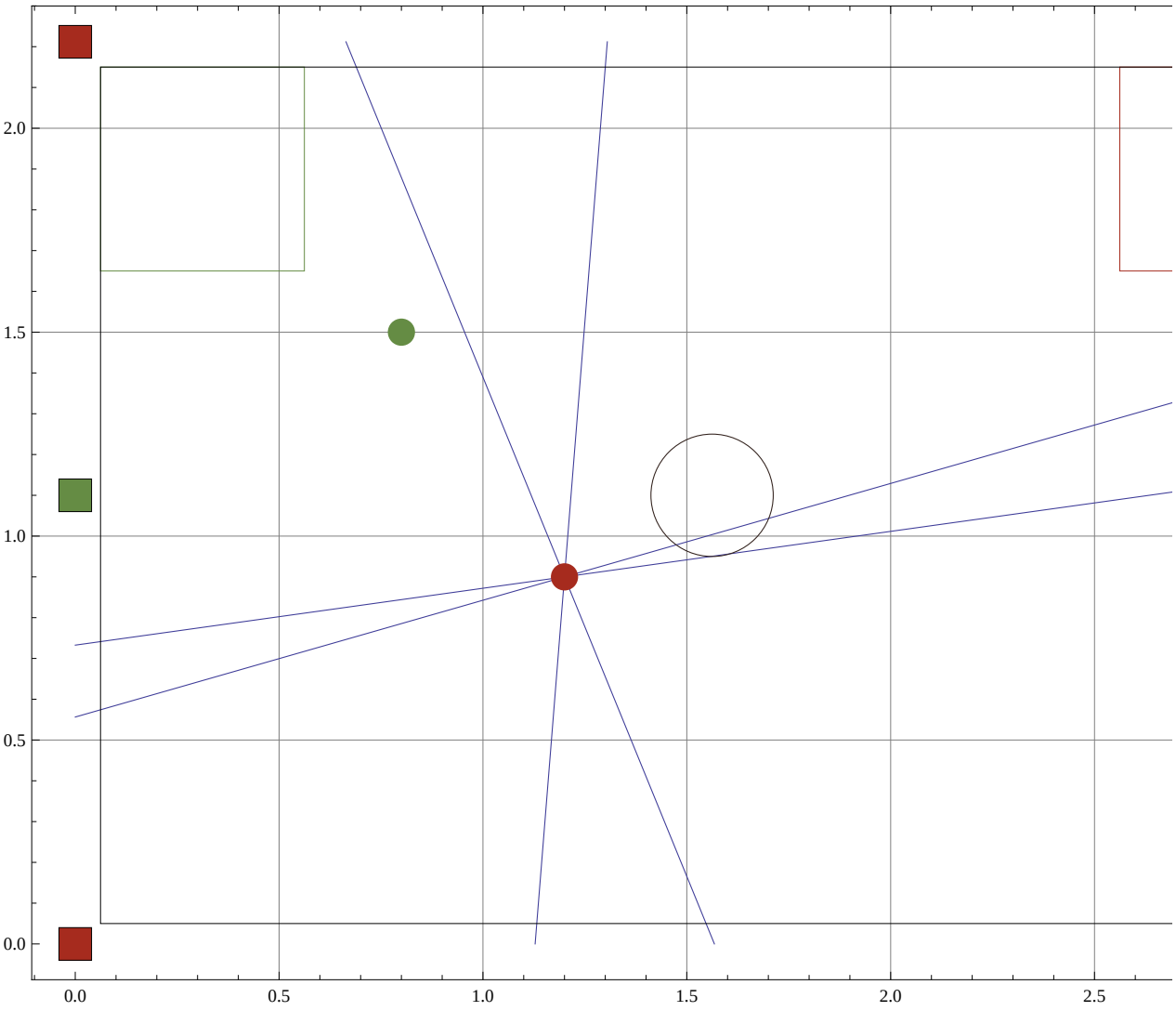
Solutions for Green team

Plots

In[81]:=
```
(* The unique lines *)
l123 =
  Graphics[ContourPlot[y - m[1, 2, 3] * x - b[1, 2, 3] == 0, {x, 0.0, 3.124}, {y, 0.0, 2.212}]];
l124 = Graphics[ContourPlot[y - m[1, 2, 4] * x - b[1, 2, 4] == 0,
    {x, 0.0, 3.124}, {y, 0.0, 2.212}]];
l134 = Graphics[ContourPlot[y - m[1, 3, 4] * x - b[1, 3, 4] == 0,
    {x, 0.0, 3.124}, {y, 0.0, 2.212}]];
l234 = Graphics[ContourPlot[y - m[2, 3, 4] * x - b[2, 3, 4] == 0,
    {x, 0.0, 3.124}, {y, 0.0, 2.212}]];
```

In[85]:=
```
Show[l123, l124, l134, l234, gTable, grobot, genemy, Frame → True, Axes → True,
 GridLines → Automatic, AxesLabel → {x, y}, AspectRatio → Automatic, PlotRange → Automatic]
```

Out[85]=



In[86]:=
```
(* All lines and hyperbolas *)
Show[l123, l124, l134, l234, gTable, contour1, contour2, contour3, contour12,
 contour23, contour13, grobot, genemy, Frame → True, Axes → True, GridLines → Automatic,
 AxesLabel → {x, y}, AspectRatio → Automatic, PlotRange → Automatic]
```

Out[86]=