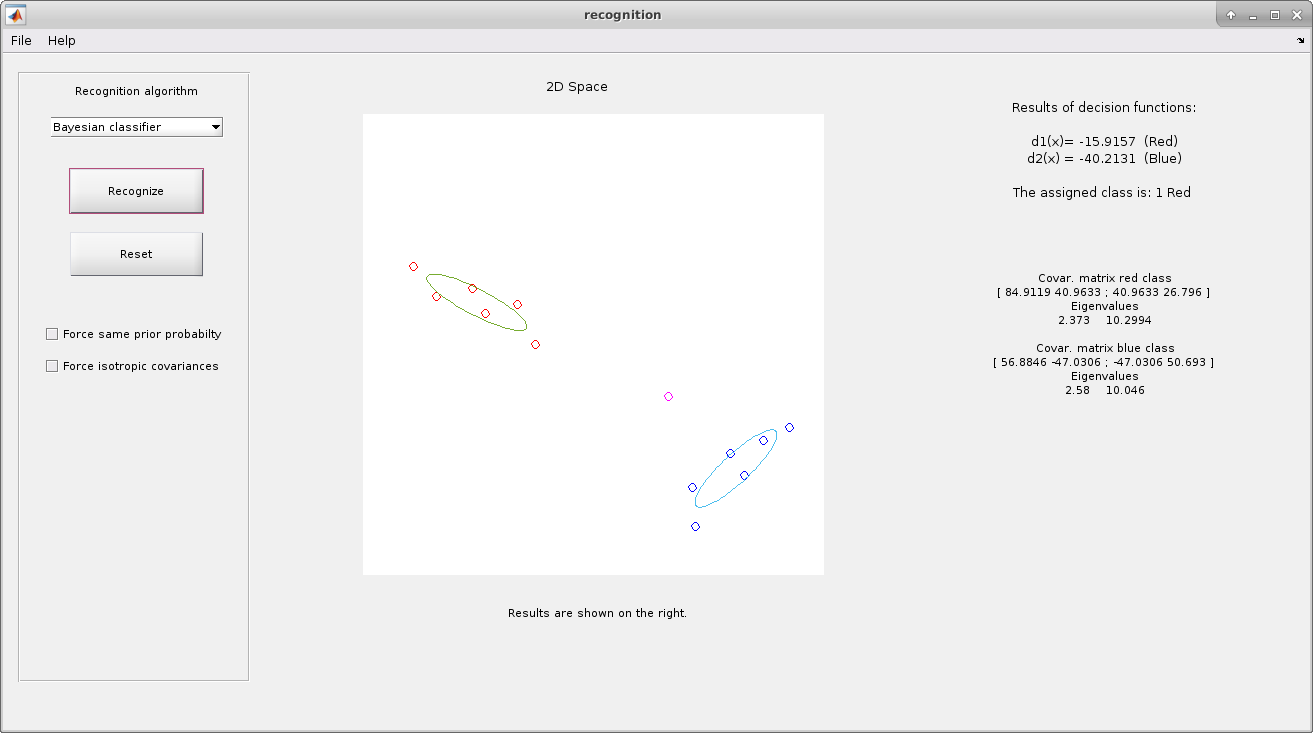
1. **COMPUTER VISION**

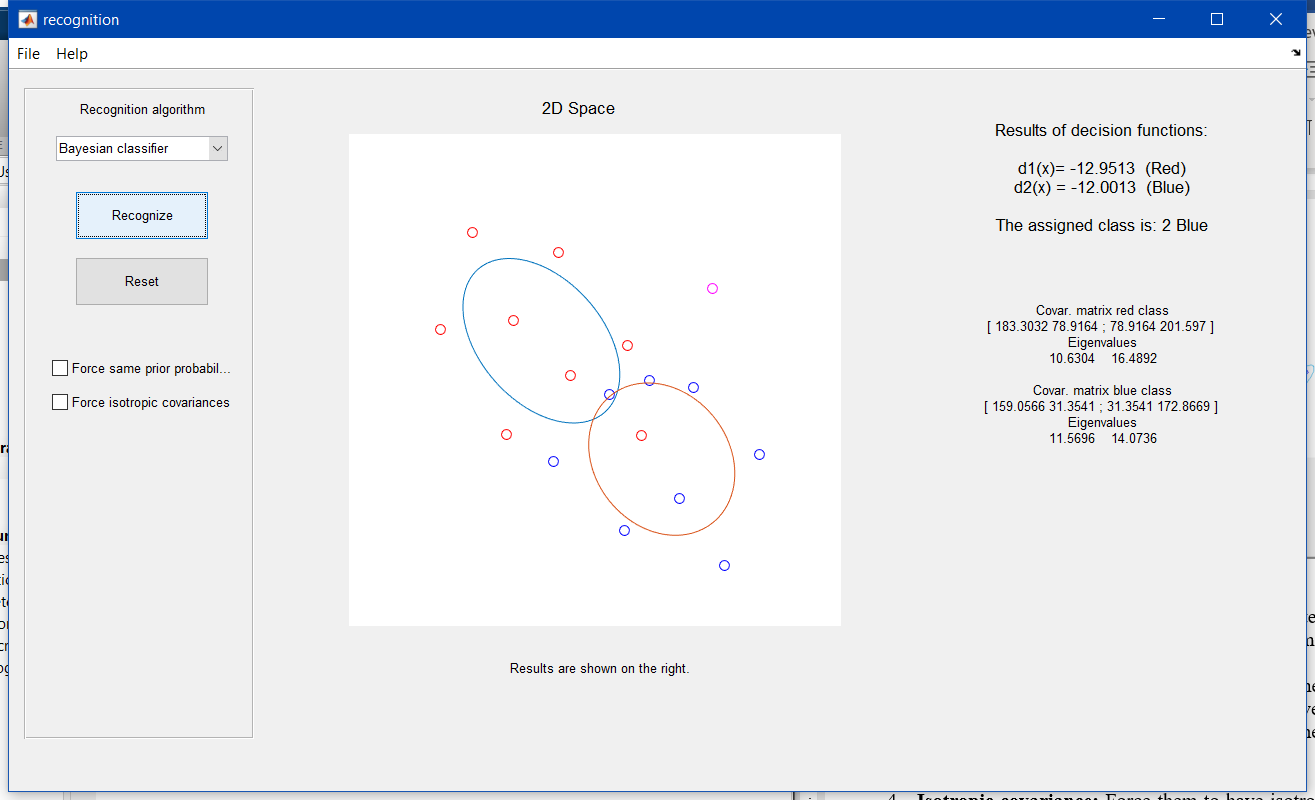
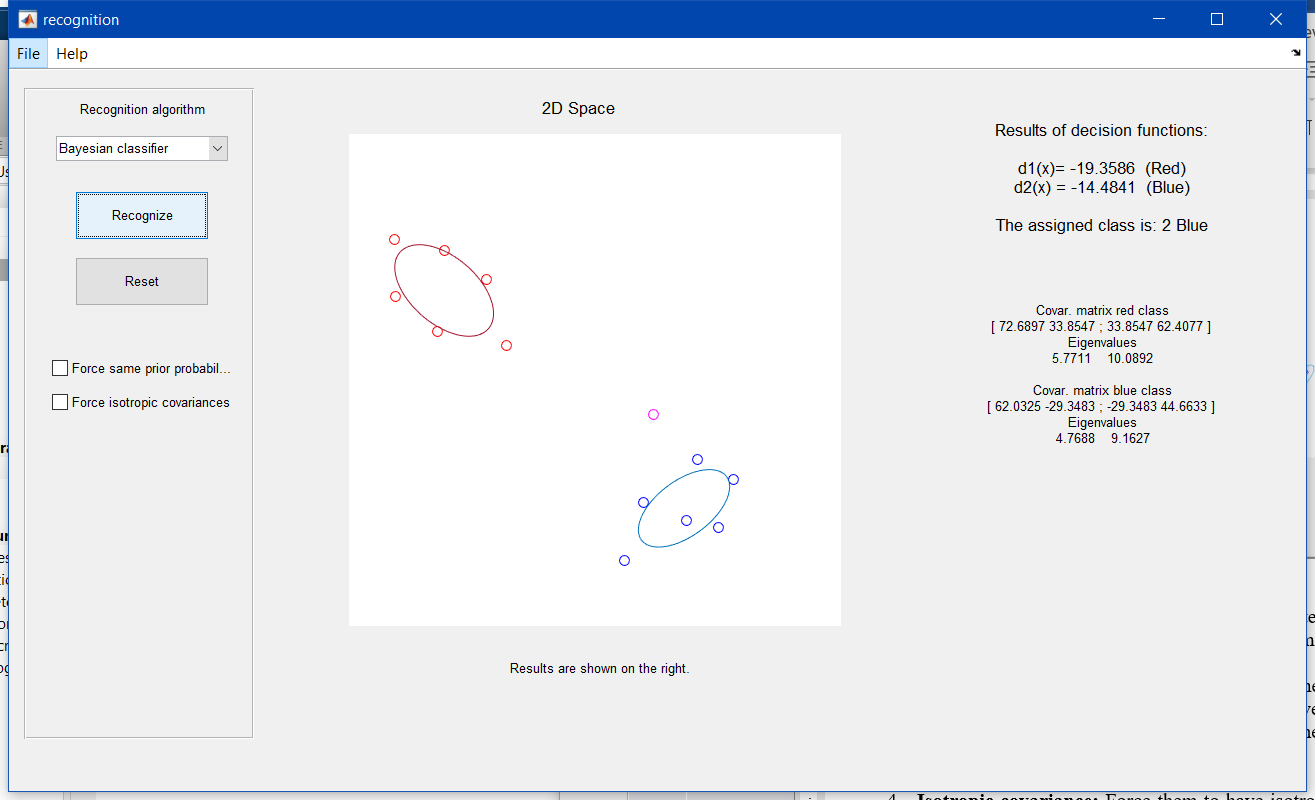
**EXERCISE 6a: Object recognition with mVision**

Concepts: Bayesian Classifier

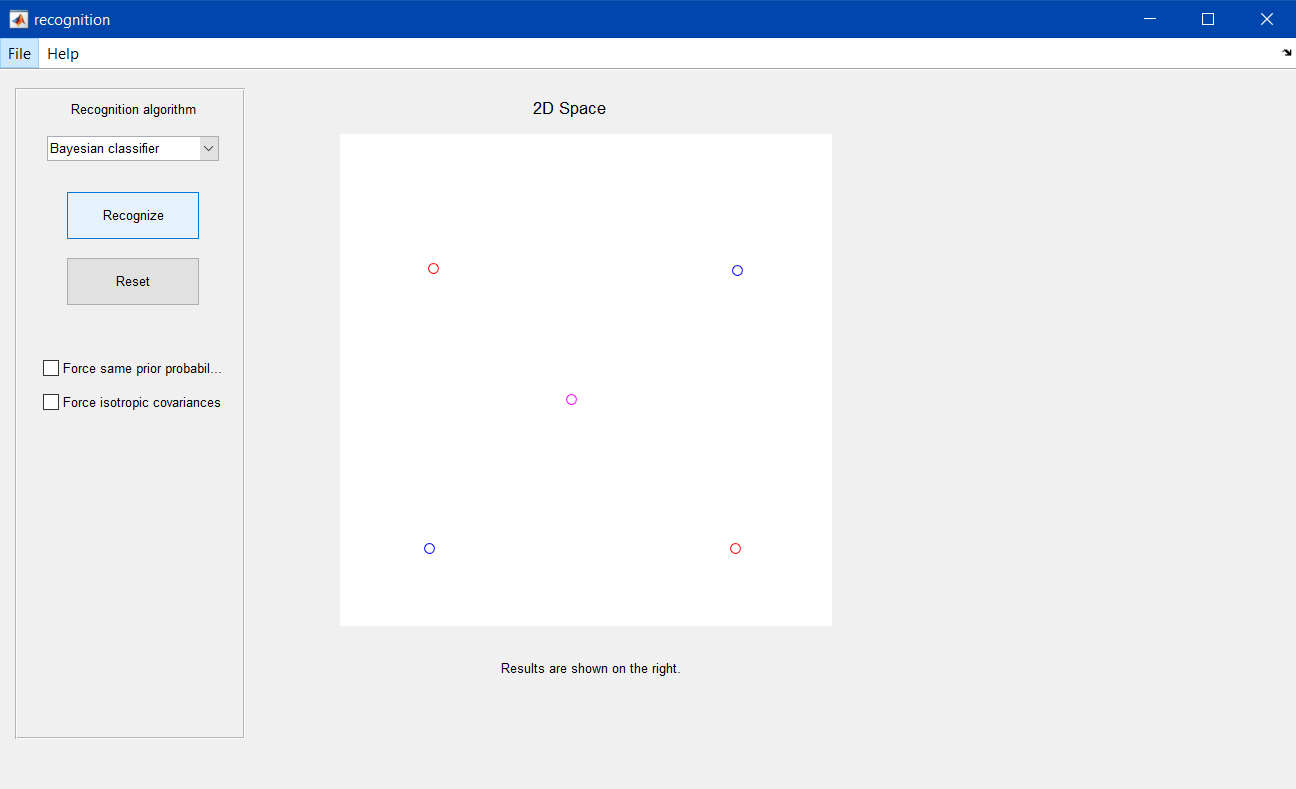
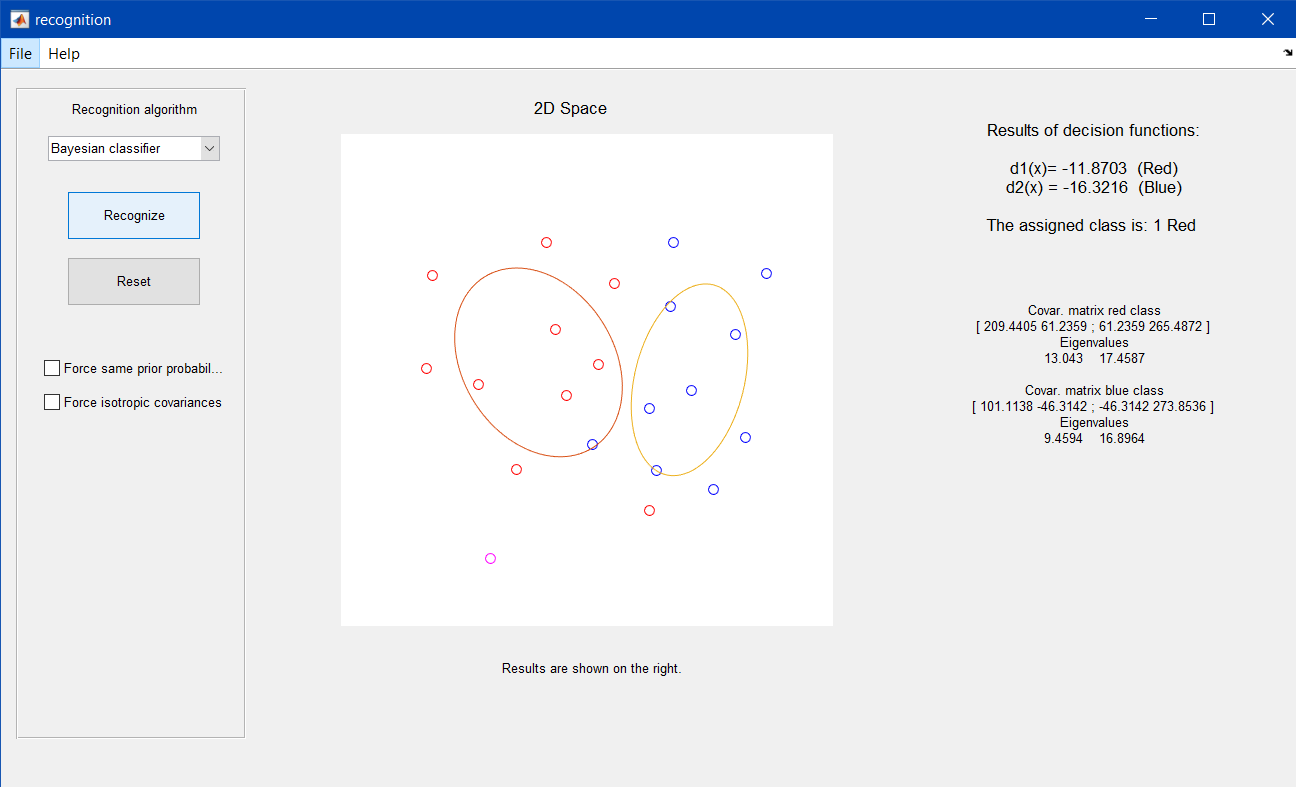
1. **Running the GUI:** Launch the **recognition** GUI of mVision, and follow the instructions on the bottom part of it to insert objects (points) in a similar way to how they are shown in the following figure (try to insert them in similar positions). The red points represent two features of objects of one class, the red ones features for other class, and the magenta point is the object to classify:



1. **Recognition:** Push the **Recognize** button and interpret the results obtained on the right part of the GUI (decision functions, covariance matrices, etc.).

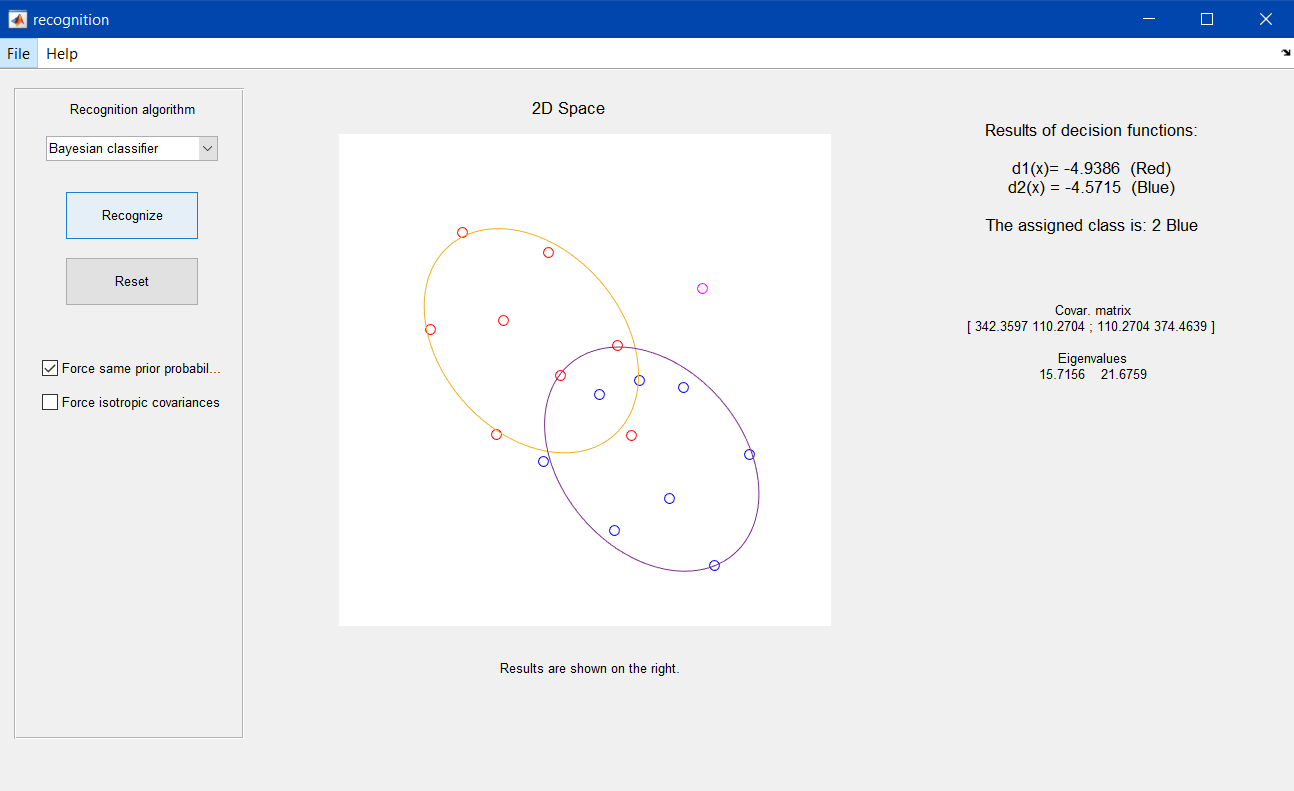
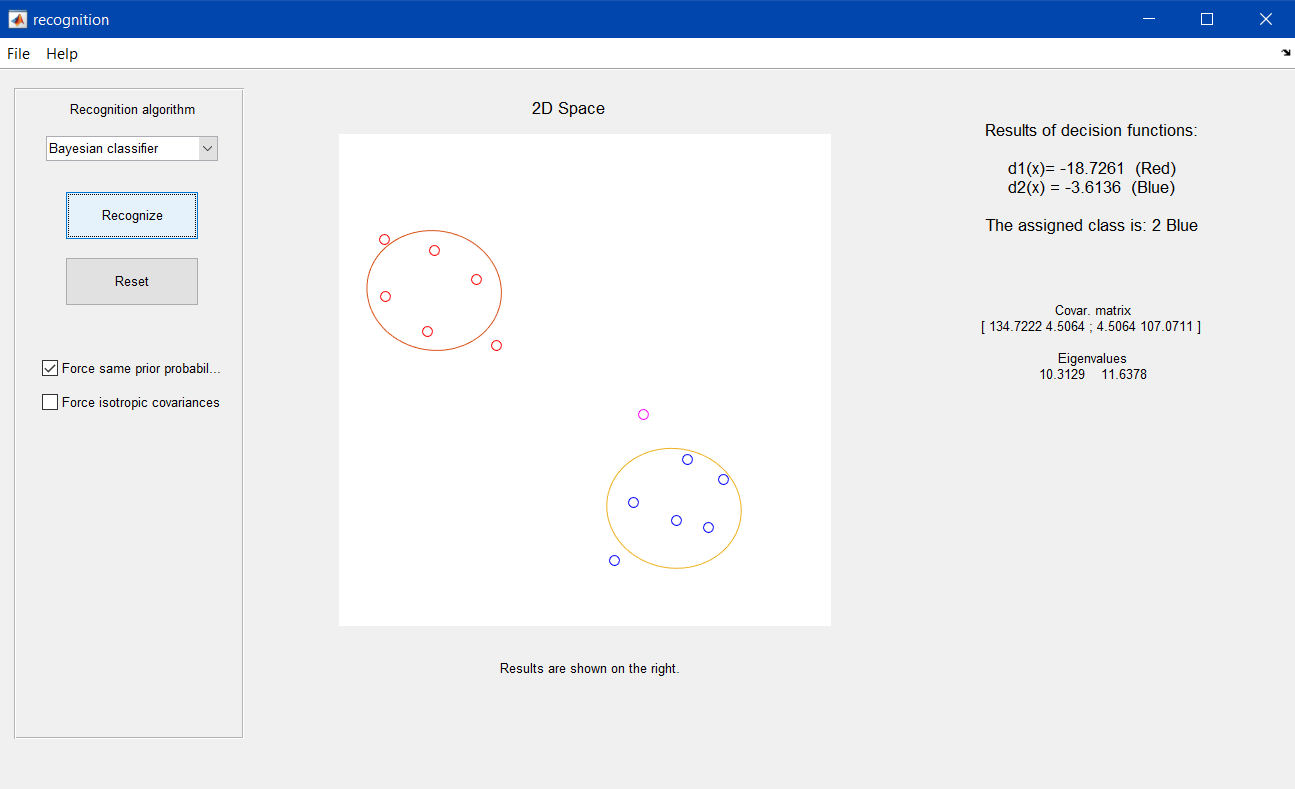


*Image 2 Image 3*

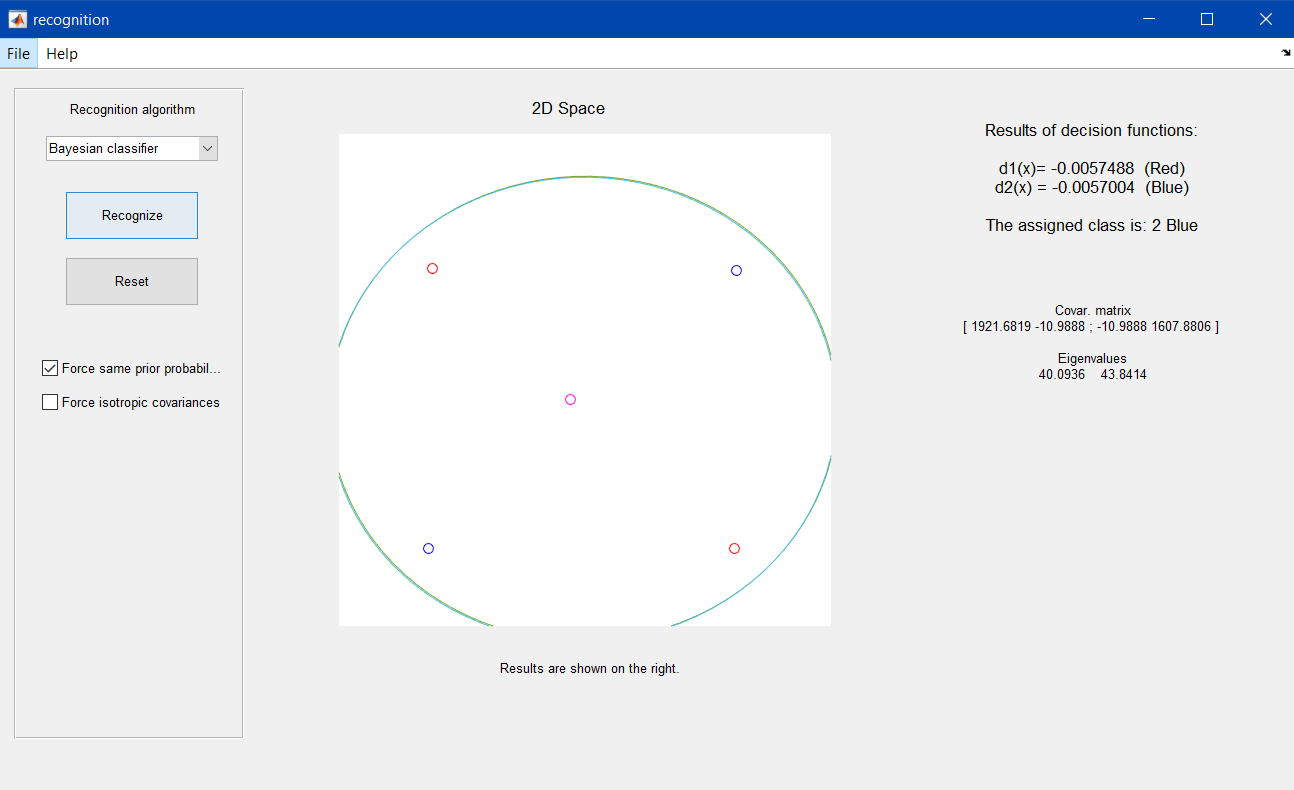
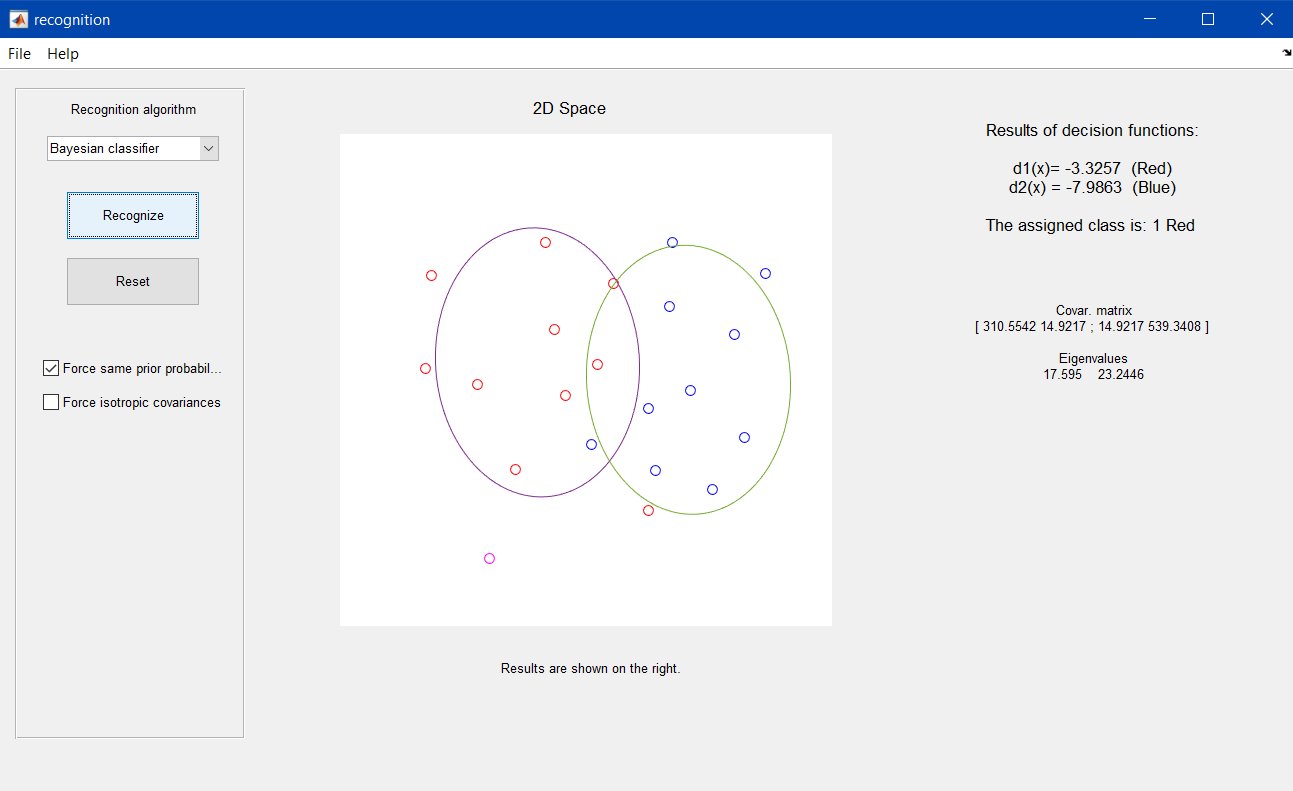


*Image 4 Image 5*

1. **Same probability and covariance:** Now force the classes to have the same probability. With that the GUI also forces the classes to have the same covariance matrices. This emulates situations where both classes have the same dispersion. Discuss the new results.

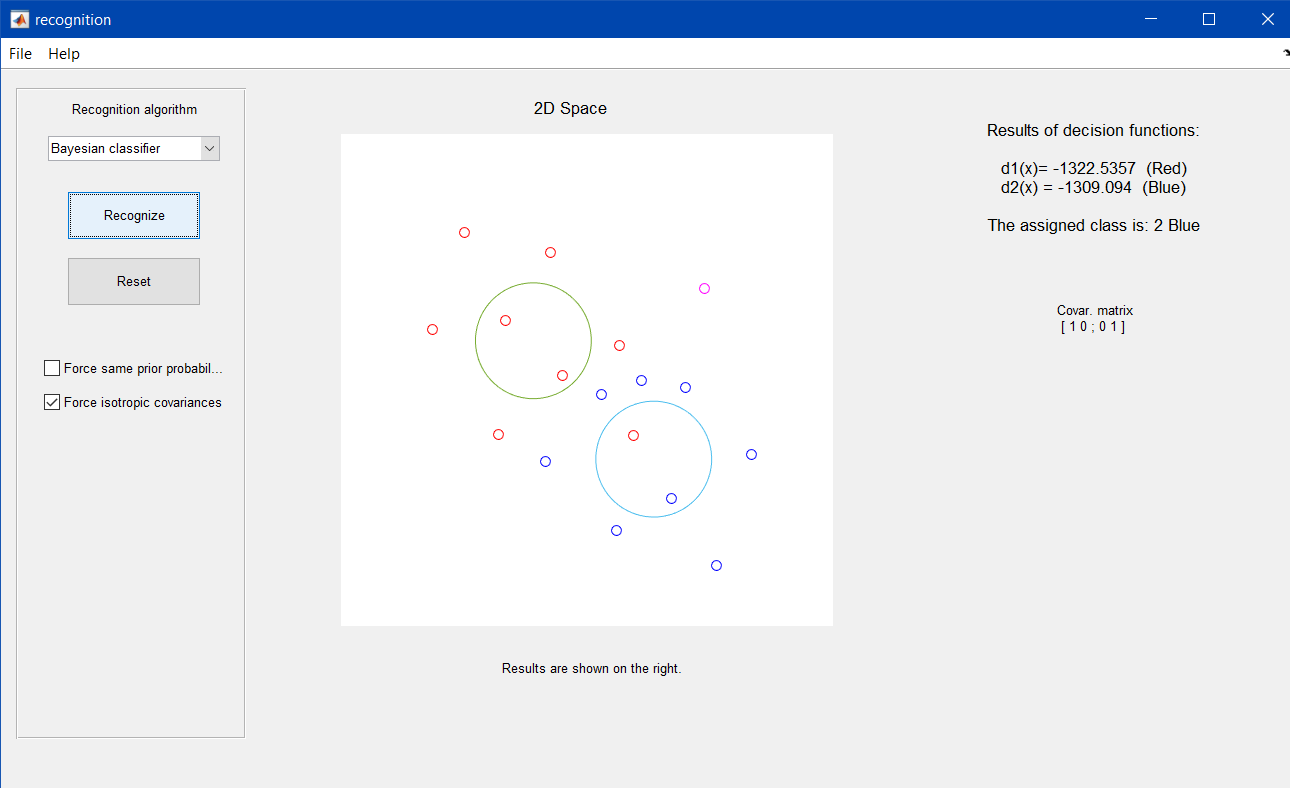
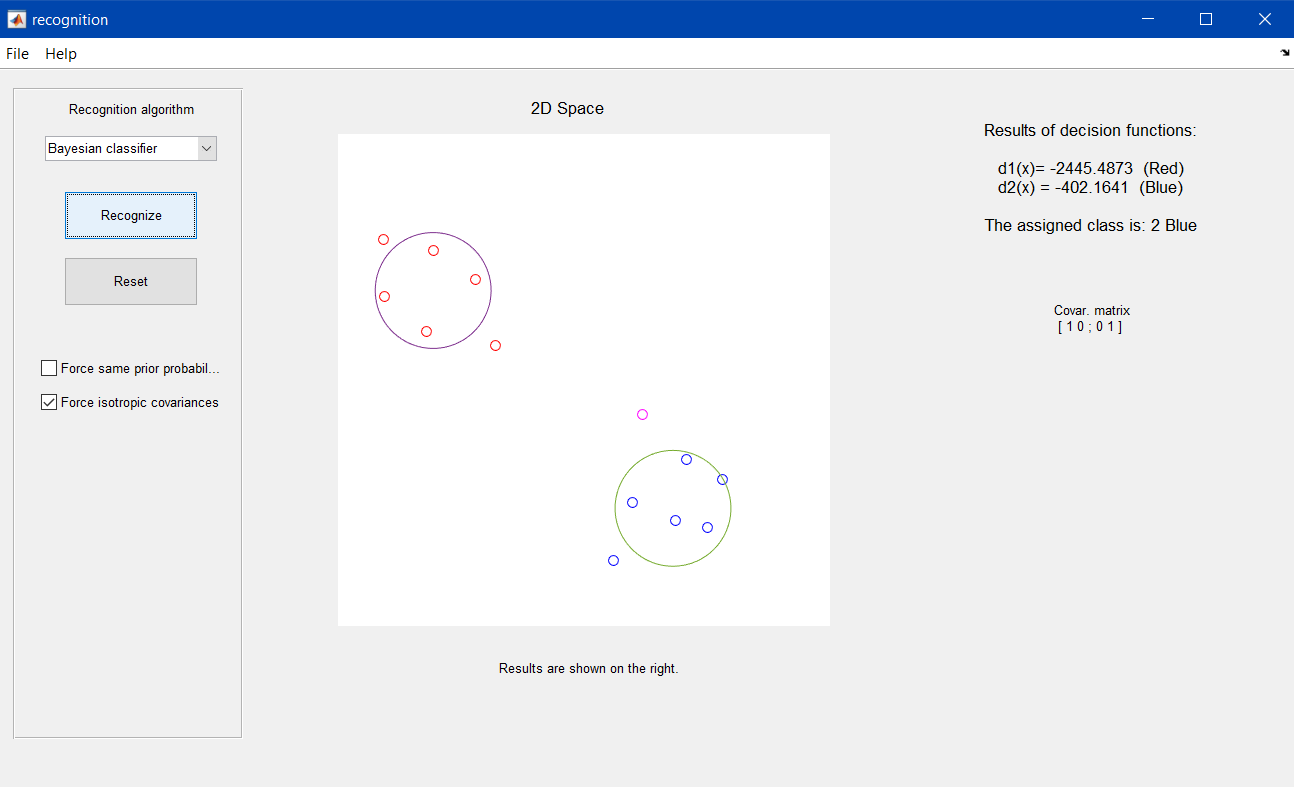


*Image 6 Image 7*



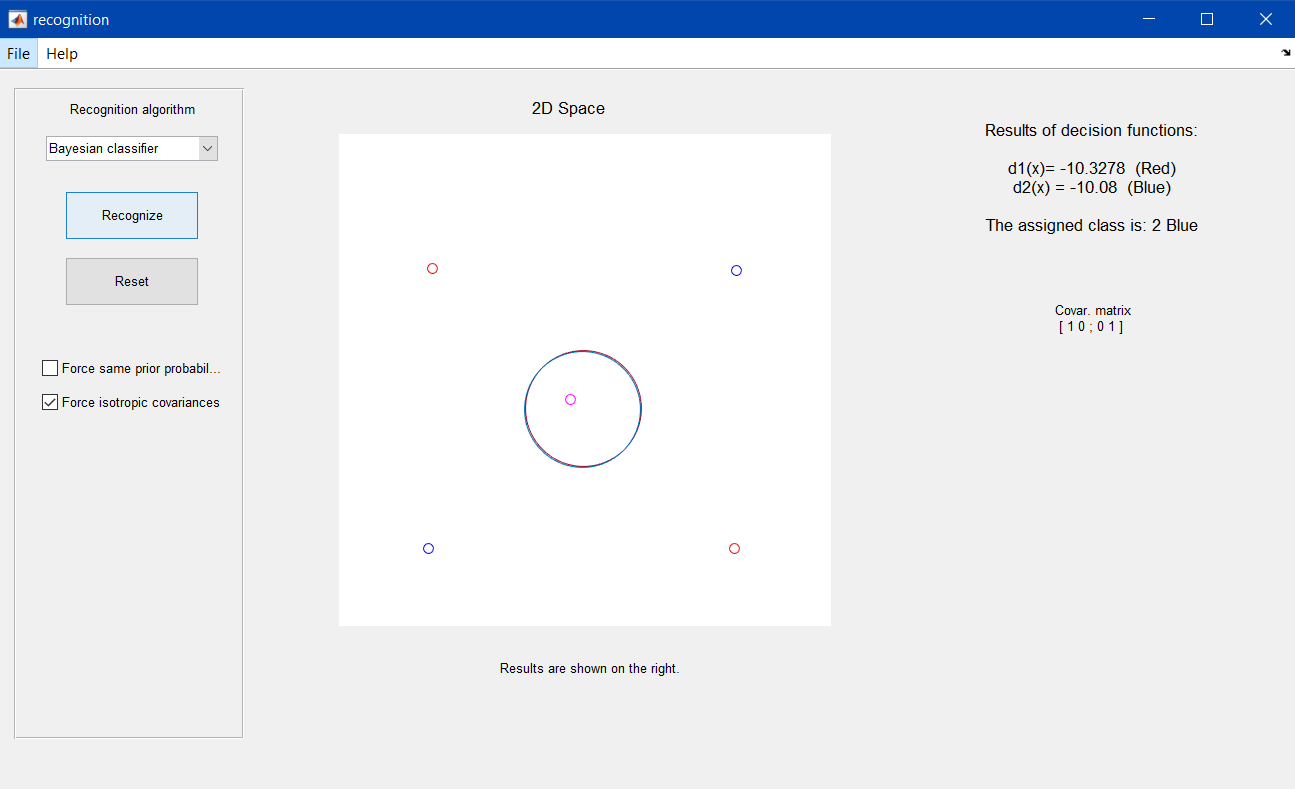
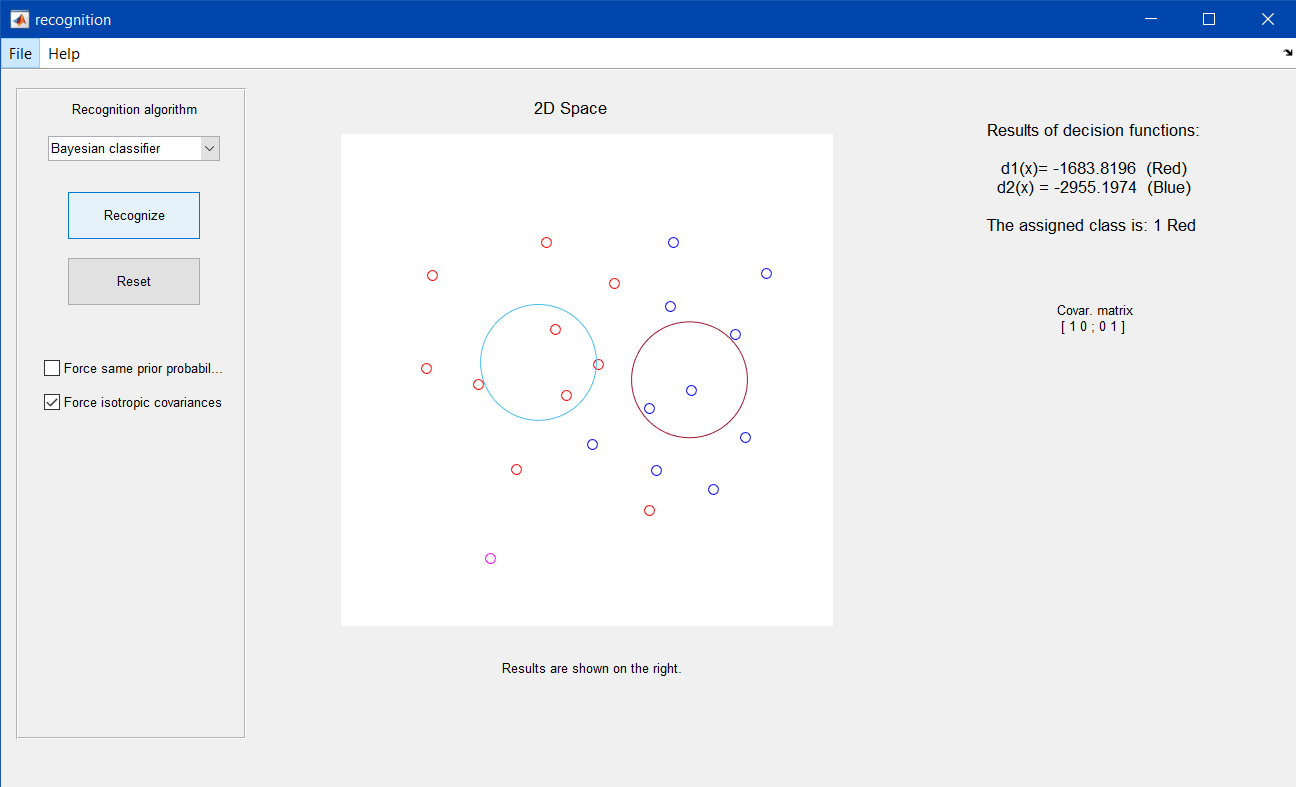
*Image 8 Image 9*

1. **Isotropic covariance:** Force them to have isotropic covariance. Discuss the obtained results.



*Image 10*

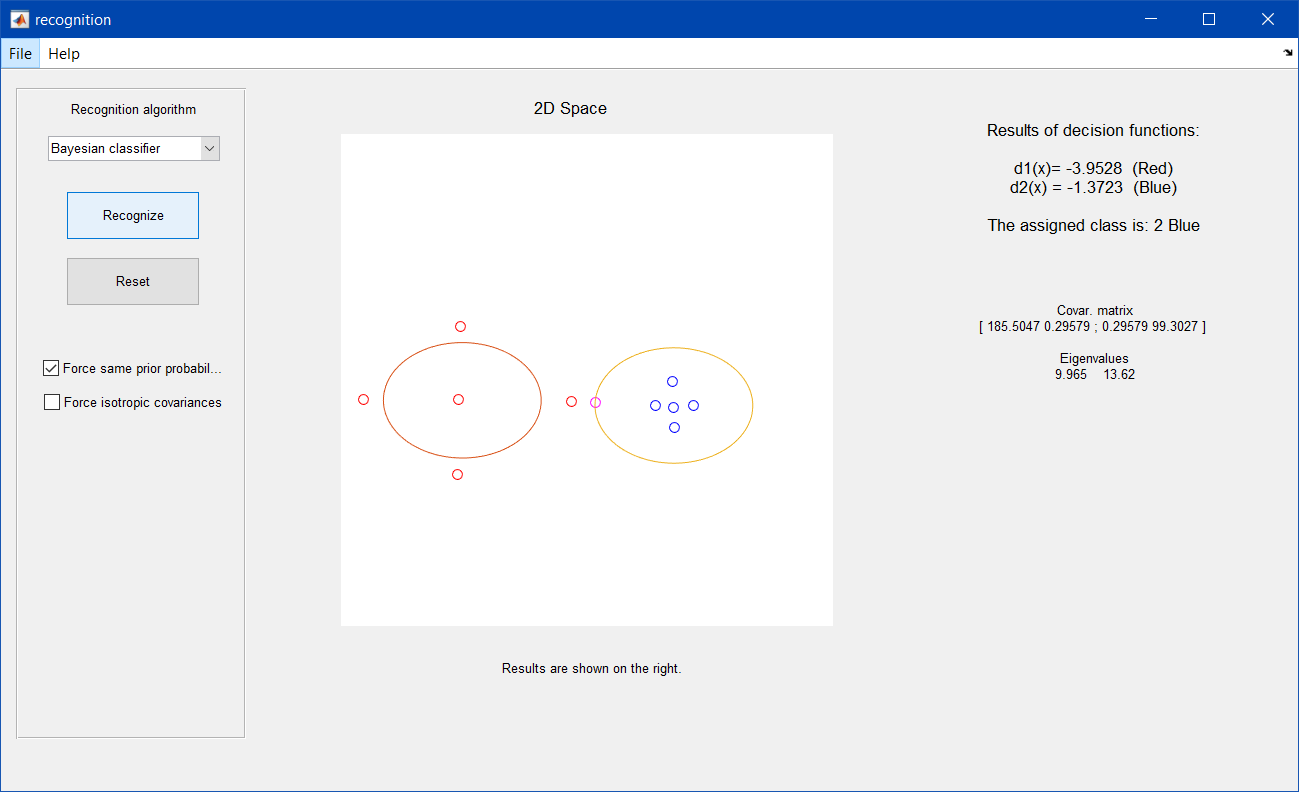
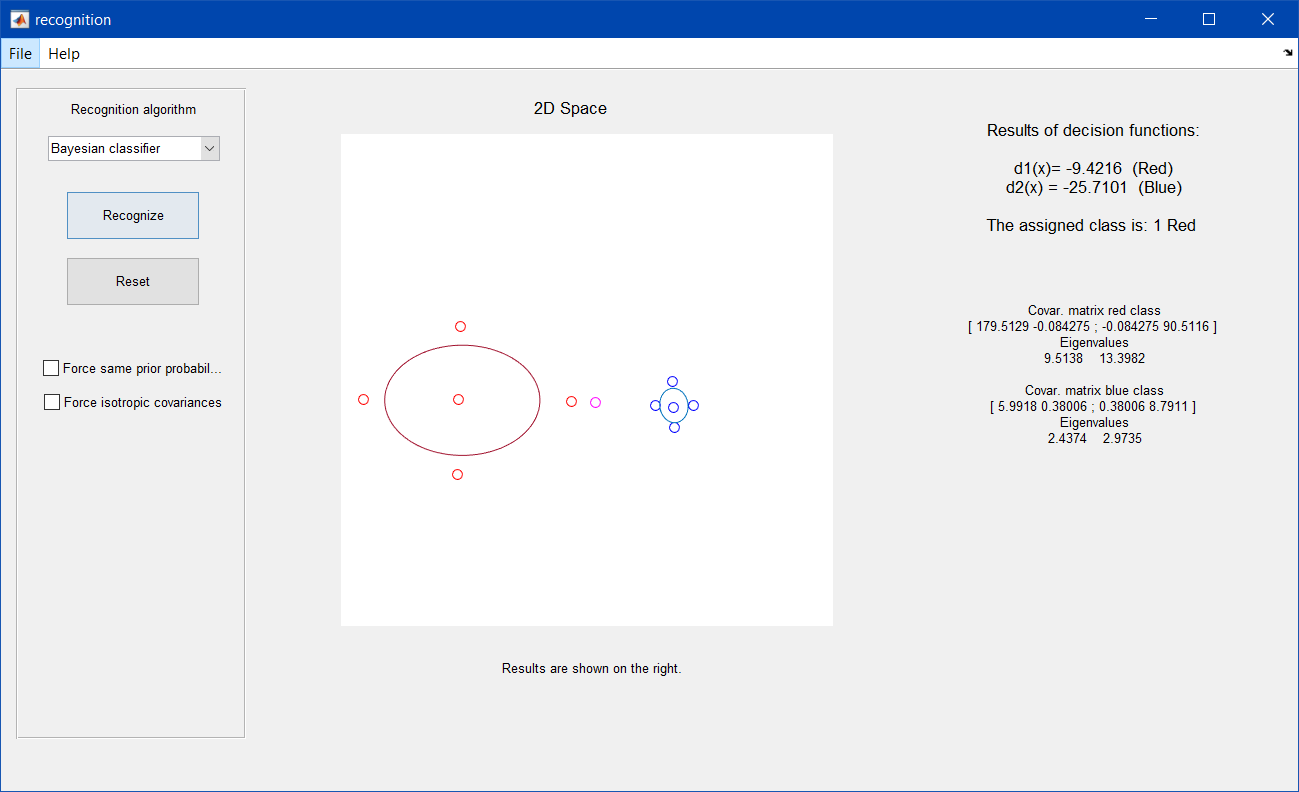
*Image 11*



*Image 12*

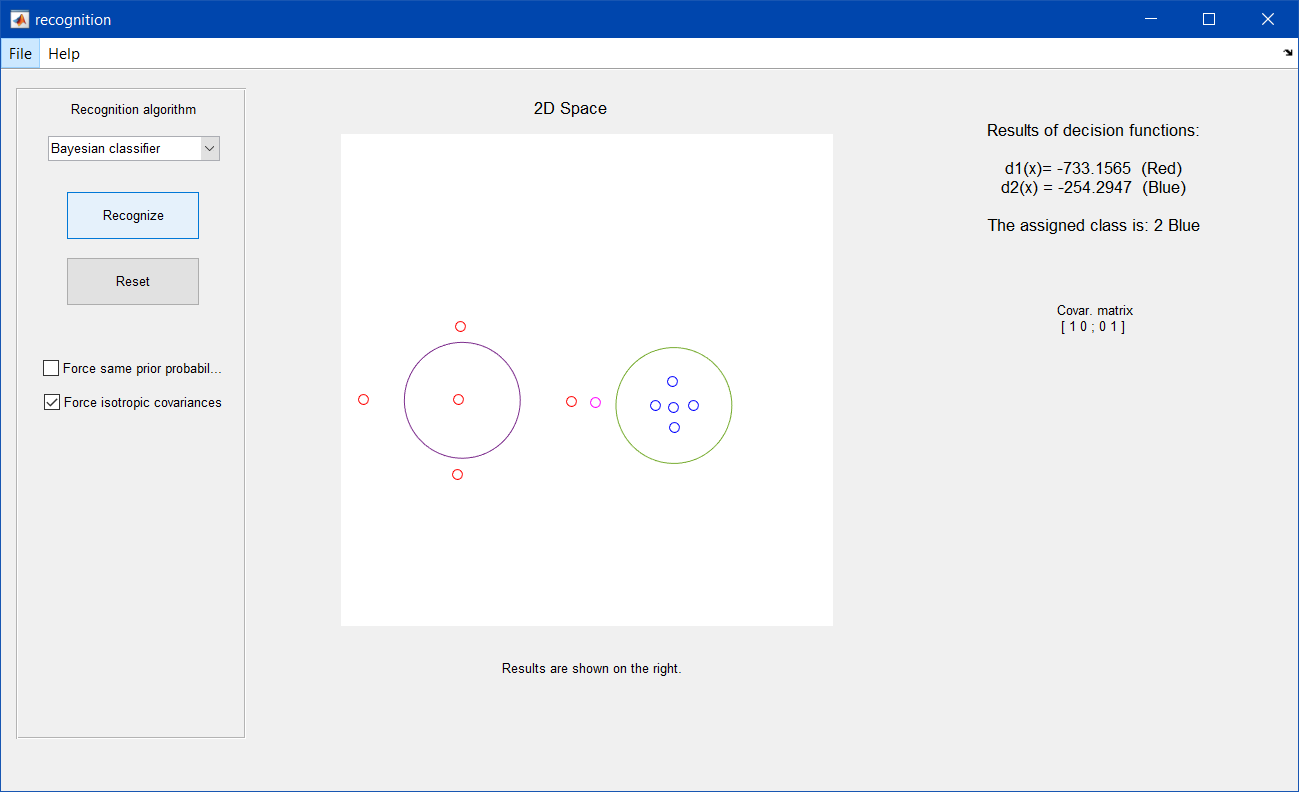
*Image 13*

1. **Distances:** Finally, comment which distances have been used in each section (2 to 4) to compute the decision functions: Mahalanobis or Euclidean?



*Image 14. Execution with the first configuration*

*Image 15. Execution with the second configuration*



*Image 16. Execution with the third configuration*

As we see in the images, with the first configuration the algorithm uses the Mahalanobis distance, whereas the second and third ones use the Euclidean distance. It uses the Mahalanobis because the covariances of each gaussian are not equals, and thus there is not equal probability between the two gaussian. In the other hand, the Euclidean is used because the covariances of each gaussian are equal, and thus the probabilities are equals.

**EXERCISE 6b: Implementing a classifier**

Concepts: Bayesian Classifier

The attached Matlab code partially implements a code for recognizing bottles using the Hu moments as descriptors and a Bayesian Classifier. To complete it:

1. **Centroid and covariance computation:** Develop the code that computes the statistical parameters (centroid and covariance matrix) of the Hu moments of a set of 15 images of three different bottle classes. That is, three centroids and covariance matrices are needed.

centroids = [mean(MHu(:,1,1)) mean(MHu(:,1,2)) mean(MHu(:,1,3));

mean(MHu(:,2,1)) mean(MHu(:,2,2)) mean(MHu(:,2,3));

mean(MHu(:,3,1)) mean(MHu(:,3,2)) mean(MHu(:,3,3));

mean(MHu(:,4,1)) mean(MHu(:,4,2)) mean(MHu(:,4,3));

mean(MHu(:,5,1)) mean(MHu(:,5,2)) mean(MHu(:,5,3));

mean(MHu(:,6,1)) mean(MHu(:,6,2)) mean(MHu(:,6,3));

mean(MHu(:,7,1)) mean(MHu(:,7,2)) mean(MHu(:,7,3))];

plot (centroids(1,1), centroids(2,1),'ks','MarkerSize',8,'MarkerFaceColor','b')

text (centroids(1,1)+0.02, centroids(2,1),'Centroid of bottle type A','Color','blue')

plot (centroids(1,2), centroids(2,2),'ks','MarkerSize',8,'MarkerFaceColor','r')

text (centroids(1,2)+0.02, centroids(2,2),'Centroid of bottle type B','Color','red')

plot (centroids(1,3), centroids(2,3),'ks','MarkerSize',8,'MarkerFaceColor','g')

text (centroids(1,3)+0.02, centroids(2,3),'Centroid of bottle type C','Color','green')

1. **Visualization:** Show graphically the centroid and covariance of each class (only the two first Hu moments). Use the function **error\_ellipse** to represent the covariance matrix. Which information could we get from that ellipses? *Note: we multiply the covariance matrices by a scalar (10) for improving their visibility.*

for bt = 1:1:N\_bottle\_types

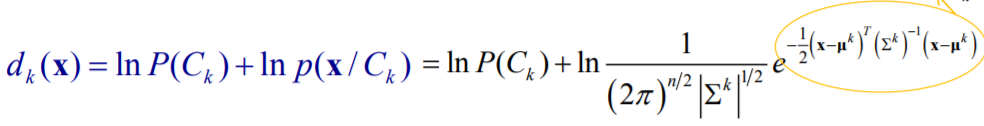
desv=MHu(:,:,bt)'-centroids(:,bt);

covars(:,:,bt) = desv\*desv'/N\_bottles;

error\_ellipse(covars(1:2,1:2,bt)\*10,centroids(1:2,bt));

end

1. **Classifying**: Complete the given loop that reads each of the 5 unused images of the three bottle classes and classify them. For that:
   1. Enable the classification part of the code with **execute\_classification.**
   2. Complete the function **evaluateDecisionFunction** that computes the expression **dk(x)** shown below. The input of this function should be: the 7 Hu moments of the initial image, the number of classes in the problem (number of bottle types), he centroid **k** and covariance **k** of the class Ci., and it’s a priori probability. *Note: The prior probability is the same for all the classes (p(Ci )= 1/3, i=1,2,3).*



function d = evaluateDecisionFunction(x, num\_classes, mu, covar, prior)

diff=x-mu;

B=-(num\_classes\*log(2\*pi)+log(det(covar))+diff'\*inv(covar)\*diff)/2;

d =A+B;

end

* 1. Implement the code that calculates and displays the values of the decision function for the three different classes, and obtains the belonging class of the input image. Use the **max** function to determine the class.

N\_starting\_bottle = N\_bottles+1;

N\_bottles = 5;

for type = 1:3

for i\_bottle = N\_starting\_bottle:N\_starting\_bottle+N\_bottles-1

im\_file\_name = strcat(path,Bottle\_types(type),int2str(i\_bottle),'.bmp');

im = imread(im\_file\_name{1});

MHu\_to\_classify(i\_bottle,:,type) = momentos\_Hu(im);

features = MHu\_to\_classify(i\_bottle,:,type)';

% Call the three decisions functions

d1 = evaluateDecisionFunction(features,N\_bottle\_types,centroids(:,1),covars(:,:,1),1/N\_bottle\_types);

d2 = evaluateDecisionFunction(features,N\_bottle\_types,centroids(:,2),covars(:,:,2),1/N\_bottle\_types);

d3 = evaluateDecisionFunction(features,N\_bottle\_types,centroids(:,3),covars(:,:,3),1/N\_bottle\_types);

% Get the winner!

[~,classified\_as] = max([d1 d2 d3]);

* 1. Represent in the figure the two first Hu moments of each image.
  2. Shows the assigned class side to such first two moments (**text** function).

% Show graphically the results

res = sprintf('Classified as: %c\n', types(classified\_as));

handler1 = plot(features(1),features(2),'dc','MarkerSize',10);

handler2 = text(features(1)+0.01,features(2)+0.01,res);

pause

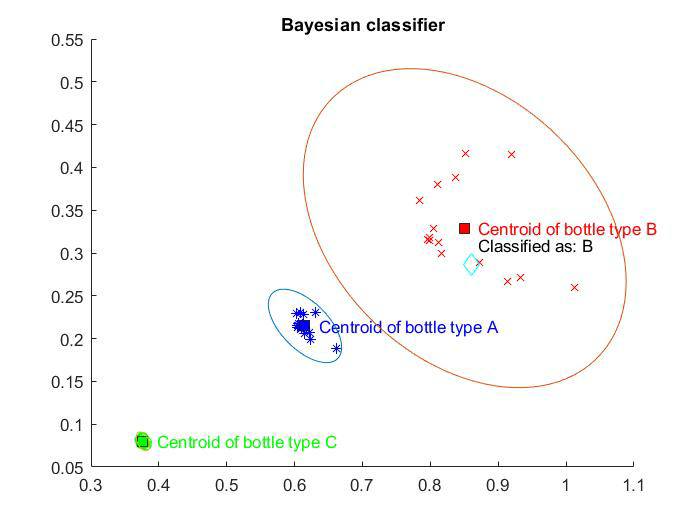
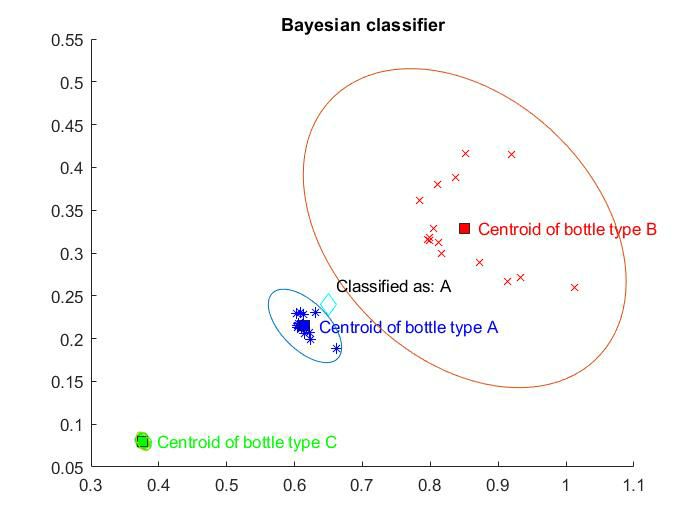
delete(handler1);

delete(handler2);

end

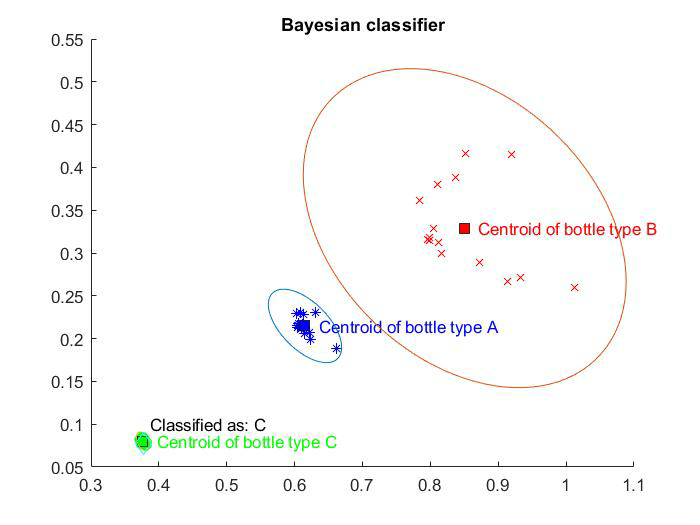
end

* 1. Comment the obtained classifications.



*Plot 1*

*Plot 2*



*Plot 3*

As we see in the images, all are classified correctly and uses the Mahalanobis distance.

1. **Decision boundaries.** Retrieve the decision boundaries between the pairs of bottle types and show them graphically over the figure using the given function **plotConic.m**. For that we are going to employ the two first Hu moments (it can not be represented using the 7 moments). Recall that the decision boundaries are obtained using the following expression, and that they look as conic sections:

where *dij* is the boundary between classes *i* and *j,* and is the decision function of the class *i* assuming that we only employ 2 Hu moments ( and ). In this way:

1. Enable the boundaries computation with **execute\_boundaries**.
2. Obtain the matrix representing the conic section of each decision function using **get\_conic\_matrix**.

conic\_matrix\_A = get\_conic\_matrix(covars(1:2,1:2,1),centroids(1:2,1)');

conic\_matrix\_B = get\_conic\_matrix(covars(1:2,1:2,2),centroids(1:2,2)');

conic\_matrix\_C = get\_conic\_matrix(covars(1:2,1:2,3),centroids(1:2,3)');

1. Get the three decision boundaries by subtracting the corresponding “conic matrix”.

decision\_boundaryAB = conic\_matrix\_A-conic\_matrix\_B;

decision\_boundaryAC = conic\_matrix\_A-conic\_matrix\_C;

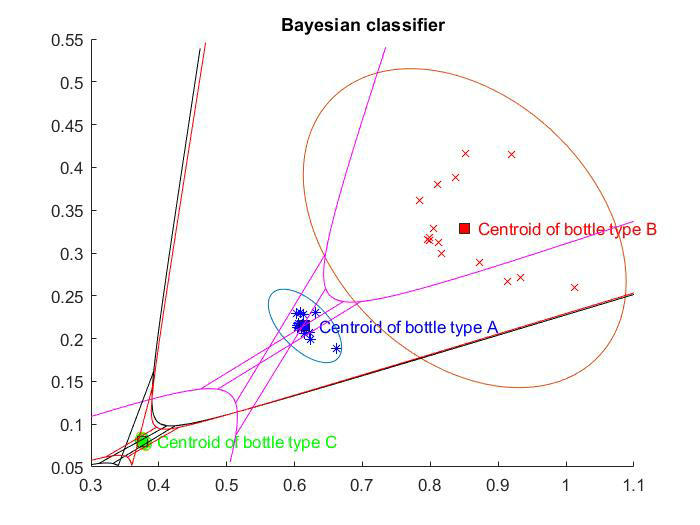
decision\_boundaryBC = conic\_matrix\_B-conic\_matrix\_C;

1. Plot them and discuss the results.

plotConic(decision\_boundaryAB, 'm');

plotConic(decision\_boundaryAC, 'k');

plotConic(decision\_boundaryBC, 'r');



*Plot 4*

**Attached code:**

%--------------------------------------------------------------------------

% EXERCISE: BAYESIAN CLASSIFIER

%--------------------------------------------------------------------------

**function** exercise\_bayesian\_clasifier

% Clean the workspace

close all**;**

clear variables**;**

% Control the execution of certain parts of the exercise

execute\_classification **=** 0**;**

execute\_boundaries **=** 0**;**

% Load bottle images from file and store their Hu moments

N\_bottles **=** 15**;**

N\_bottle\_types **=** 3**;**

Bottle\_types **=** **{**'botella\_A\_'**,**'botella\_B\_'**,**'botella\_C\_'**};**

types **=** **[**'A'**,**'B'**,**'C'**];**

path **=** 'imagenes botellas/'**;**

**for** type **=** 1**:**3

**for** i\_bottle **=** 1**:**N\_bottles

im\_file\_name **=** strcat**(**path**,**Bottle\_types**(**type**),**int2str**(**i\_bottle**),**'.bmp'**);**

im **=** imread**(**im\_file\_name**{**1**});**

MHu**(**i\_bottle**,:,**type**)** **=** momentos\_Hu**(**im**);**

**end**

**end**

% Show the first two moments for the images of each type

figure**()**

hold on**;**

title**(**'Bayesian classifier'**)**

plot **(**MHu**(:,**1**,**1**),**MHu**(:,**2**,**1**),**'b\*'**)**

plot **(**MHu**(:,**1**,**2**),**MHu**(:,**2**,**2**),**'rx'**)**

plot **(**MHu**(:,**1**,**3**),**MHu**(:,**2**,**3**),**'go'**)**

% Compute their mean and show the centroids of the two first Hu moments

centroids **=** --------**;**

plot **(**--------**,**--------**,**'ks'**,**'MarkerSize'**,**8**,**'MarkerFaceColor'**,**'b'**)**

text **(**--------**+**0.02**,** --------**,**'Centroid of bottle type A'**,**'Color'**,**'blue'**)**

plot **(**--------**,**--------**,**'ks'**,**'MarkerSize'**,**8**,**'MarkerFaceColor'**,**'r'**)**

text **(**--------**+**0.02**,** --------**,**'Centroid of bottle type B'**,**'Color'**,**'red'**)**

plot **(**--------**,**--------**,**'ks'**,**'MarkerSize'**,**8**,**'MarkerFaceColor'**,**'g'**)**

text **(**--------**+**0.02**,** --------**,**'Centroid of bottle type C'**,**'Color'**,**'green'**)**

% Compute their covariance matrices and show them

**for** bt **=** 1**:**1**:**N\_bottle\_types

--------

covars**(:,:,**bt**)** **=** --------**;**

error\_ellipse**(**--------**\***10,--------**]);**

**end**

**if** execute\_classification **==** 1

%Ok, now load the bottles not used for training, and classify them using

% a Bayesian classifier

N\_starting\_bottle **=** N\_bottles**+**1**;**

N\_bottles **=** 5**;**

**for** type **=** 1**:**3

**for** i\_bottle **=** N\_starting\_bottle**:**N\_starting\_bottle**+**N\_bottles**-**1

im\_file\_name **=** strcat**(**path**,**Bottle\_types**(**type**),**int2str**(**i\_bottle**),**'.bmp'**);**

im **=** imread**(**im\_file\_name**{**1**});**

MHu\_to\_classify**(**i\_bottle**,:,**type**)** **=** momentos\_Hu**(**im**);**

features **=** MHu\_to\_classify**(**i\_bottle**,:,**type**)';**

% Call the three decisions functions

d1 **=** evaluarFuncDec**(**--------**);**

d2 **=** evaluarFuncDec**(**--------**);**

d3 **=** evaluarFuncDec**(**--------**);**

% Get the winner!

classified\_as **=** --------

% Show graphically the results

res **=** sprintf**(**'Classified as: %c\n'**,** types**(**classified\_as**));**

handler1 **=** plot**(**--------**,**--------**,**'dc'**,**'MarkerSize'**,**10**);**

handler2 **=** text**(**--------**+**0.01**,** --------**+**0.01**,**res**);**

pause

delete**(**handler1**);**

delete**(**handler2**);**

**end**

**end**

**end**

**if** execute\_boundaries **==** 1

% Plot the decision boundary between all the possible pairs of bottle

% types: AB, AC and BC.

% First, get the matrix representing each conic section

conic\_matrix\_A **=** --------

conic\_matrix\_B **=** --------

conic\_matrix\_C **=** --------

% Get decision boundary between all the types (their conic section

% representation)

decision\_boundaryAB **=** --------

decision\_boundaryAC **=** --------

decision\_boundaryBC **=** --------

% Plot them using plotConic

plotConic**(**--------**,** 'm'**);**

plotConic**(**--------**,** 'k'**);**

plotConic**(**--------**,** 'r'**);**

**end**

**end**

**function** d **=** evaluateDecisionFunction**(**x**,** num\_classes**,** mu**,** covar**,** prior**)**

% Evaluate the gaussian deccision function of a vector of features given:

% x: vector of features

% num\_classes: number of classes in the problem

% covar: covariance matrix of the class

% prior: a priori probability of that class

d **=** --------

**end**

**function** conic\_matrix **=** get\_conic\_matrix**(**covariance**,**centroid**)**

% Returns the matrix representation of a conic section given the:

% covariance: covariance matrix of a two dimensional gaussian

% centoid: centroid [mu1 mu2] of that gaussian

syms x1 x2

X **=** **[**x1 x2**];**

independiente **=** **-**1**/**2**\*((**log**(**det**(**covariance**))+**centroid**\***covariance**.^(-**1**)\***centroid**'));**

lineal **=** X**\***covariance**.^(-**1**)\***centroid**';** % simplify?

cuadratico **=** **-**1**/**2**\***X**\***covariance**.^(-**1**)\***X**.';**

ecuacion **=** cuadratico **+** lineal **+** independiente**;**

**[**decision**,**terminos**]** **=** coeffs**(**ecuacion**);**

A **=** double**(**decision**(**find**(**terminos**==**x1**^**2**)));**

B **=** double**(**decision**(**find**(**terminos**==**x1**\***x2**)));**

C **=** double**(**decision**(**find**(**terminos**==**x2**^**2**)));**

D **=** double**((**decision**(**find**(**terminos**==**x1**))));**

E **=** double**(**decision**(**find**(**terminos**==**x2**)));**

F **=** double**(**decision**(**find**(**terminos**==**1**)));**

conic\_matrix **=** **[**A B**/**2 D**/**2**;** B**/**2 C E**/**2**;** D**/**2 E**/**2 F**];**

**end**