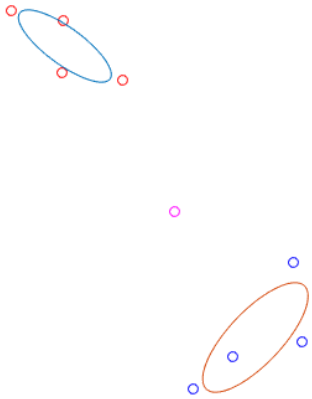
Dpto. Ingeniería de Sistemas y Automática
Universidad de Málaga

COMPUTER VISION

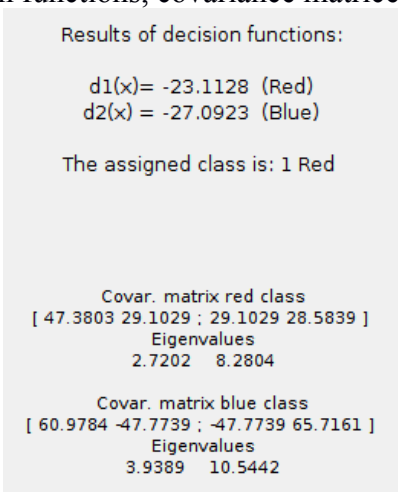
EXERCISE 6a: Object recognition with mVision

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:

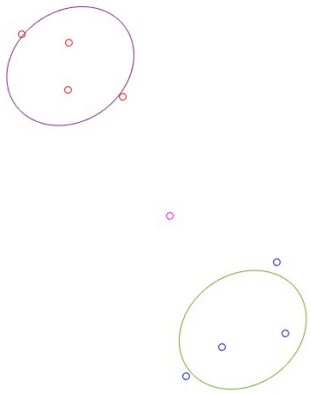


2. **Recognition:** Push the **Recognize** button and interpret the results obtained on the right part of the GUI (decision functions, covariance matrices, etc.).



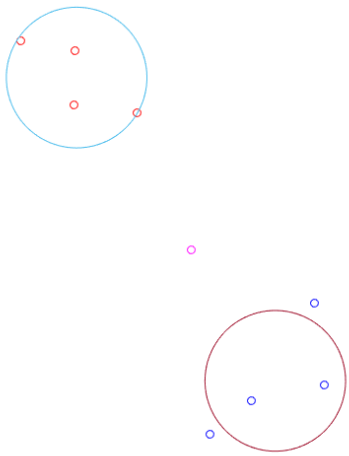
mVisions have classified our object as an red object. *Due to the position of the object to recognize, both decission functions are very similar, so we are not too sure about the result.*

3. **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.



Now both classes have the same covariance matrices. This makes our object now to be assigned to the blue class.

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



Now it's classified as a blue class object, as it is closer using Euclidean distance instead of Mahalanobis distance.

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

As explained before, we used first Mahalanobis distance firstly and then Euclidean.

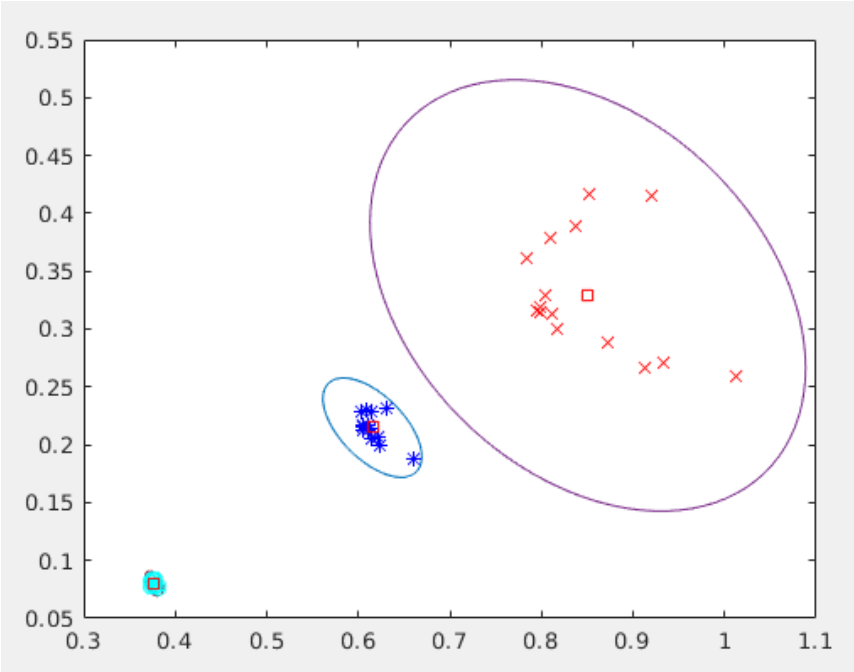
EXERCISE 6b: Implementing a classifier
Concepts: Bayesian Classifier

1. **Centroid and covariance computation:** Develop a 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. Use the function calculating the Hu moments from previous exercises.

   I develop this exercise using a for loop which made the same calculations for each of the three bottle categories. At first, I stored everything into a 3-dimensional array, but then I realised that every iteration was independent, and I could just store the needed values for the current iteration, which simplified things a lot.

2. **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: you can multiply the covariance matrices by a scalar for improving their visibility.*

   I've also plotted the Hu moments of each image onto the covariance ellipses, so it looks prettier. Furthermore, I've added some colours so each category can be easily difference.



3. **Bayesian classifier:** Design a classifier that takes as input the image of a bottle and assigns it to its corresponding class. For doing so:

a. Implement a function called **decision_function** that computes the expression $\mathbf{d_k(x)}$ shown below. The input of this function should be: the 7 Hu moments of the initial image, and the centroid [redacted] ^{k} and covariance [redacted] ^{k} of the class C^i.

$$d_k(\mathbf{x}) = \ln P(C_k) + \ln p(\mathbf{x}/C_k) = \ln P(C_k) + \ln \frac{1}{(2\pi)^{n/2}|\Sigma^k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}^k)^T(\Sigma^k)^{-1}(\mathbf{x}-\boldsymbol{\mu}^k)}$$

b. Write a function called **belonging_class** that calculates and displays the values of the decision function for the three different classes, and returns the belonging class of the input image. Use the **max** function to determine the class.

The code developed for this task was done with the help of my college Ricardo Holthausen. The first script simply takes the central moments of an image and the covariance matrix of a class. The prior probability is the same for each class, one third.

The second script simply gets an image as input. Afterwards, it computes the central Hu moments of the image and then it finds out which class has the highest output from the decission function script. That is the supposed class to which the bottle belongs.

4. **Classifying**: Extend the previous script with a loop reading each of the 5 unused images of the three bottle classes, and:
   a. represents in the figure the two first Hu moments of each image,
   b. shows the assigned class. Use the function **text** to display a text in the (x,y) coordinates (show the image). *Note: don't forget the pause and delete functions.*
   c. Comment the obtained classifications.

This is a simple function that just loops over the 15 test images, compute it's Hu moments for display purposes and also a little tag with the class that the script assigned to the image.