ROBOT GRASP SYNTHESIS AS A PATTERN RECOGNITION PROBLEM

SIMULATION ENVIRONMENT

How to install the system

Simply unzip the file **"code.zip"**; the following folders will be created:

- **main:** contains the four main Matlab functions that can be run from the command window.
- **functions:** contains auxiliary Matlab functions.
- **objects:** contains the objects database as Matlab .mat files (38 objects).
- **results:** contains temporal result files.
- **weka:** contains the necessary Weka code.

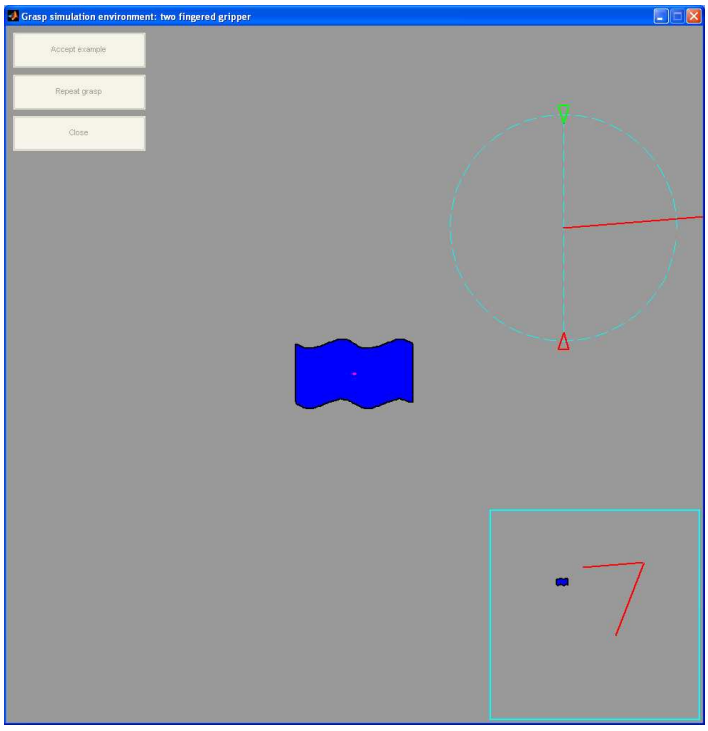
Both Matlab and java must be installed in your computer in order to be able to run the simulation environment.

Running a training session

Running a training session (that is, giving grasp examples to the system) is simple:

1. From the Matlab command window, use **cd** to change to the **main** folder.
2. Type **train_grasps([1:10])**. This command will allow you to input grasp examples for the objects 1 to 10 of our database, and will create a model of your grasping behaviour. You can select a different set of objects for training if you prefer.

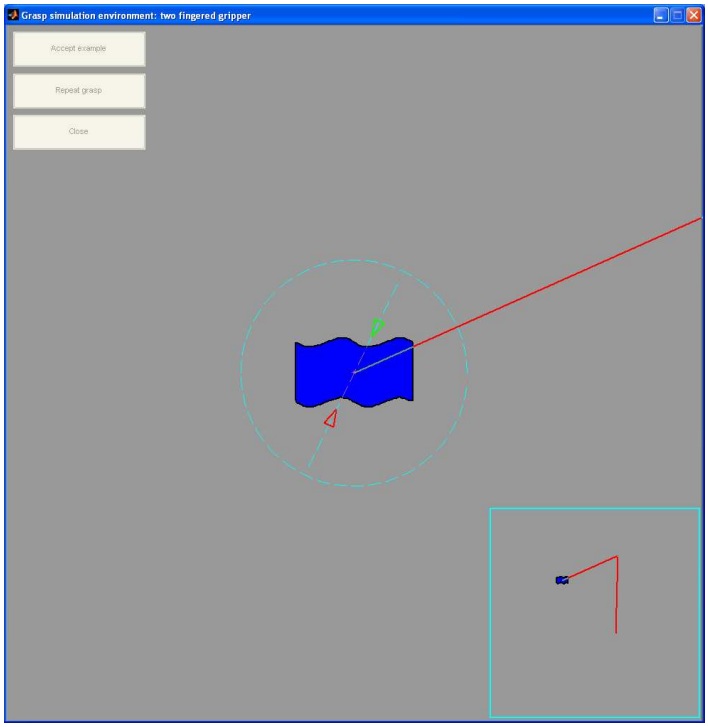
For each object, the following Matlab figure will appear:



Such figure shows the object to be grasped and the initial location of the robot gripper (the complete robot arm is shown scaled in the lower right corner). You have to move the robot gripper to the desired grasping points using the mouse:

- Left clicking on any location will displace the centre of the gripper to such location.
- Right clicking on the blue dashed circle will rotate the gripper fingers.
- Right clicking on the blue dashed line will open or close the robot fingers.

The goal is to place the gripper fingers close to the desired grasping points; for example, they could be placed like the following figure shows:



Once the gripper fingers have been correctly placed, pressing ENTER will close them, and the two contact points will be displayed. You will be asked to either accept the grasp example or repeat the grasp (you must click on one of the upper left buttons). You can also click on close, if you don't want to input more grasp examples.

After the 10 grasp training examples are entered, the system displays the following messages:

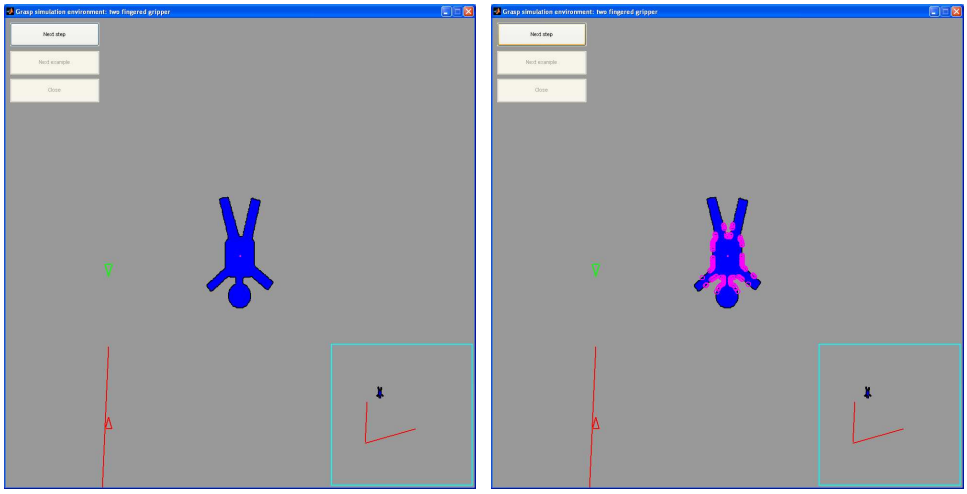
```
>> train_grasps([1:10])
Training grasp for object 1...
Training grasp for object 2...
Training grasp for object 3...
Training grasp for object 4...
Training grasp for object 5...
Training grasp for object 6...
Training grasp for object 7...
Training grasp for object 8...
Training grasp for object 9...
Training grasp for object 10...
Creating full model ...
>>
```

The last message indicates that the model has been successfully created.

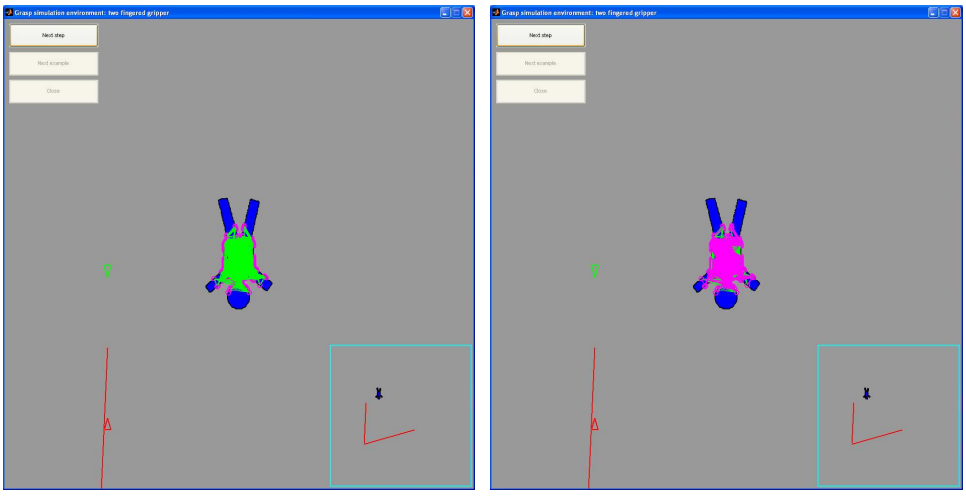
Running an autonomous grasping session

Once the system has been trained, it is possible to check whether it has learned to grasp new objects or not. Simply type **grasp_auto([11:12], 0)**, and the system will grasp autonomously objects 11 and 12 of the database (of course, you can select any other set of objects). The information that will appear on screen is the following:

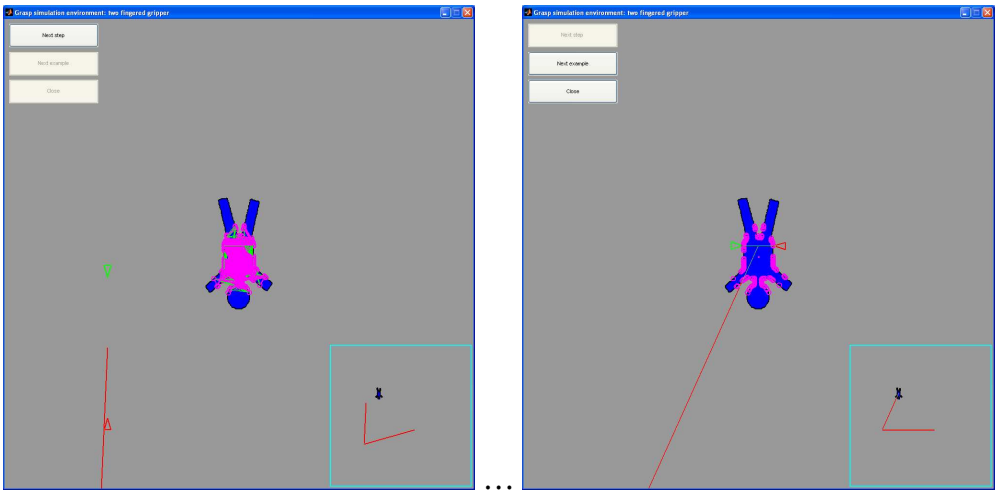
1. First, the object to be grasped will appear on screen. After you click on **Next step**, the contour points that the system has considered valid for placing a robot finger on them (according to the examples given) will be shown:



2. Clicking **Next step** twice more will display further information: the possible pairs of contact points reachable by the robot gripper (in green) and, among them, the ones that the system considers valid, according to the examples (in pink).



3. If you keep on clicking **Next step**, the ideal set of two contact points will be displayed (in blue) and the robot gripper will move towards its final location and grasp the object.



After the grasp has been completed you can click on **Next example** to continue checking the behaviour of the system or you can click on **Close** to end the application.

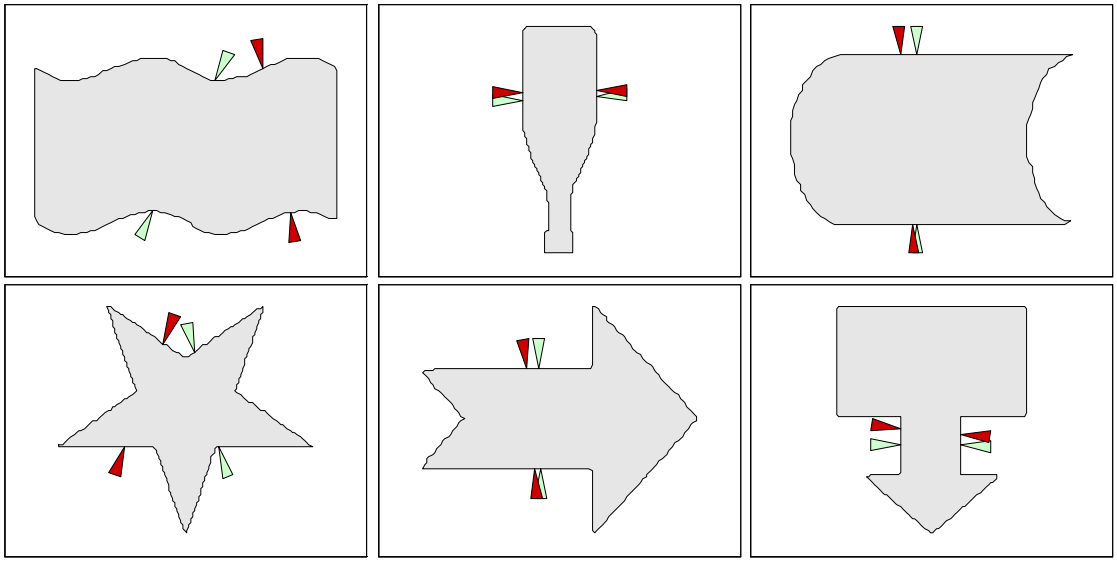
Running a leave-one-out experiment

A better way of checking the behaviour of the system is by running a leave-one-out experiment. Give grasp training examples for a set of objects (say, 6 objects) and then check the grasp that the system chooses autonomously for each of these 6 objects when the remaining 5 objects are used as training examples.

Just type **leave_one_out([1:6])** from the command window to run the experiment with the 6 first objects of the database (a different set of objects can be chosen). After giving all training examples to the system, the system will start creating partial models (discarding one of the objects) and computing autonomous grasps for such objects. This is the information that will be displayed on the command window:

```
>> leave_one_out([1:6])
Training grasp for object 1...
Training grasp for object 2...
Training grasp for object 3...
Training grasp for object 4...
Training grasp for object 5...
Training grasp for object 6...
Creating full model ...
Creating model discarding object 1...grasping object 1...
Creating model discarding object 2...grasping object 2...
Creating model discarding object 3...grasping object 3...
Creating model discarding object 4...grasping object 4...
Creating model discarding object 5...grasping object 5...
Creating model discarding object 6...grasping object 6...
>>
```

And the grasping results will be displayed as next figure shows:



In the previous figure, the light green arrows correspond to the training grasp given as an example, and the red arrows correspond to the grasp computed by the system using the other 5 objects as examples. Trying with different grasping behaviours will allow you to check whether the system imitates such behaviours or not.

Supervised grasping

System performance can be improved by adding user supervision. The main idea is to check the autonomous grasps performed by the system in a leave-one-out experiment and to tell the system which grasps have been performed poorly. These grasps will be added to the database as examples of invalid grasping behaviour (without user supervision, invalid grasping examples are generated randomly).

You can try user supervision by typing, for example, **leave_one_out_loop([1:6])**. Such a command will make the system run a leave one out experiment with the first 6 objects of the database, similar to that of the previous section. After presenting the results on screen, you will be prompted for the list of grasps that you consider inappropriate. Then, the system will be retrained adding these new invalid examples and the results will be presented again. Hopefully, the results will improve iteration after iteration.

An example is shown next: First, we type:

```
>> leave_one_out_loop([1:6])
```

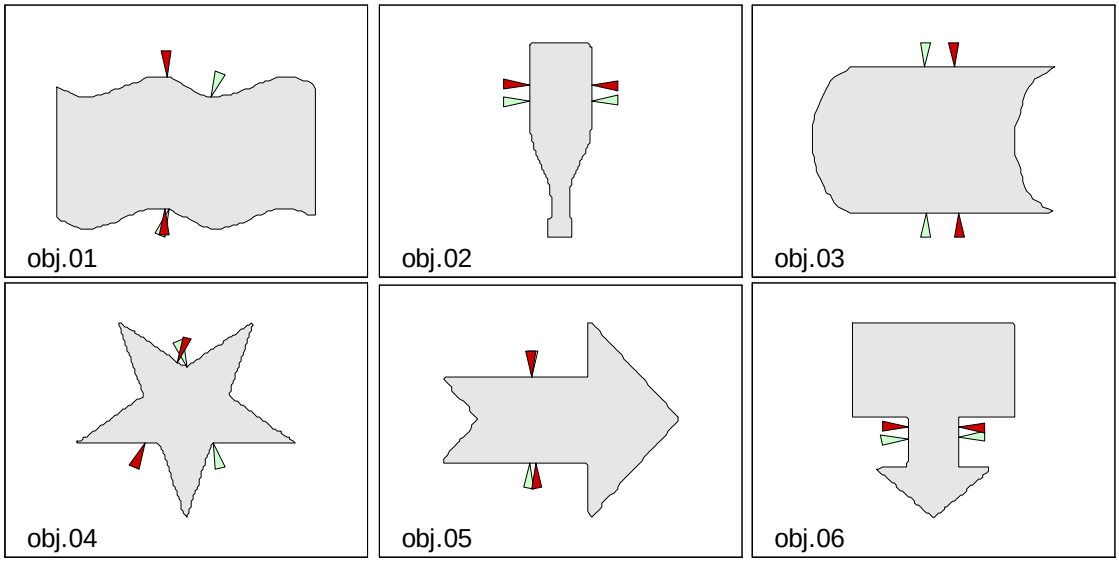
Then, we enter grasp examples for the 6 objects using the graphical interface. Once the examples are entered, the system will start processing:

```
Training grasp for object 1...
Training grasp for object 2...
Training grasp for object 3...
Training grasp for object 4...
Training grasp for object 5...
Training grasp for object 6...
```

```
Creating full model ...
Creating model discarding object 1...grasping object 1...
Creating model discarding object 2...grasping object 2...
Creating model discarding object 3...grasping object 3...
Creating model discarding object 4...grasping object 4...
Creating model discarding object 5...grasping object 5...
Creating model discarding object 6...grasping object 6...

PLEASE INPUT WRONG GRASPS OR '0' TO END
Wrong graps:
```

At this point, the grasping results are shown and we are asked to input the autonomous grasps that we consider incorrect in this first iteration. Next figure shows the results obtained in this example run:



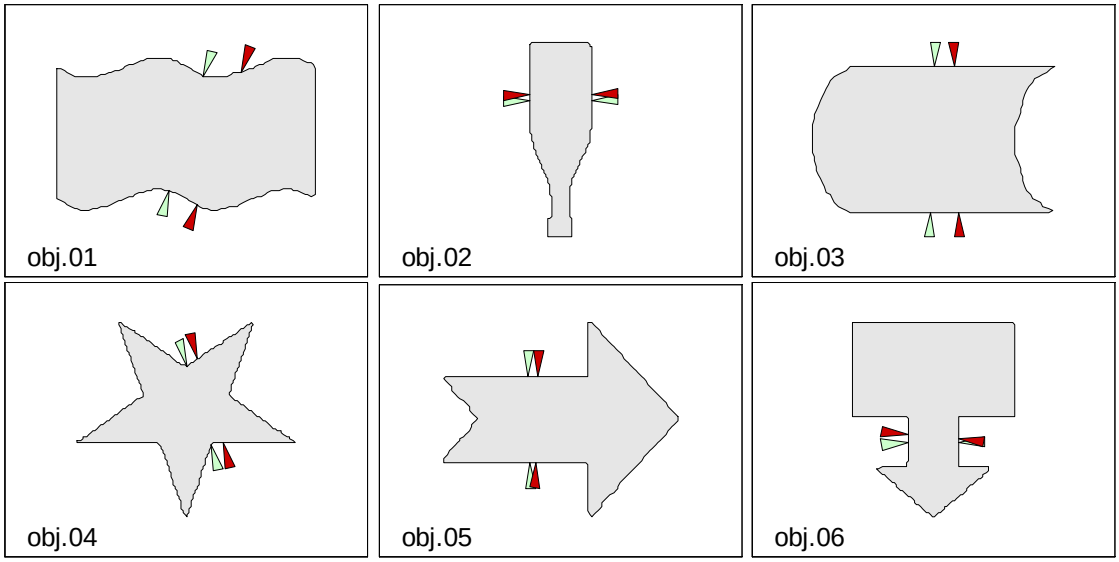
We may say that the grasp of object #01 is not what we expected (one of the contact points is located on a convex area). We should enter "1" and the system will start processing again, considering the autonomous grasp of object #01 as an invalid grasping example:

```
PLEASE INPUT WRONG GRASPS OR '0' TO END
Wrong graps: 1
Autonomous grasps of figures: [ 01 ] will be considered wrong
graps examples
Recomputing...

Creating model discarding object 1...grasping object 1...
Creating model discarding object 2...grasping object 2...
Creating model discarding object 3...grasping object 3...
Creating model discarding object 4...grasping object 4...
Creating model discarding object 5...grasping object 5...
Creating model discarding object 6...grasping object 6...

PLEASE INPUT WRONG GRASPS OR '0' TO END
Wrong graps:
```

At this point, new results are shown and we are asked again for the incorrect grasps. Next figure shows the results obtained in the second iteration:



The grasp of object #01 is now correct, and the other grasps are also correct (please note that they are not exactly the same grasps of the first iteration). So we may enter “0” to end the training loop:

```
PLEASE INPUT WRONG GRASPS OR '0' TO END
Wrong graps: 0
No wrong examples added
>>
```

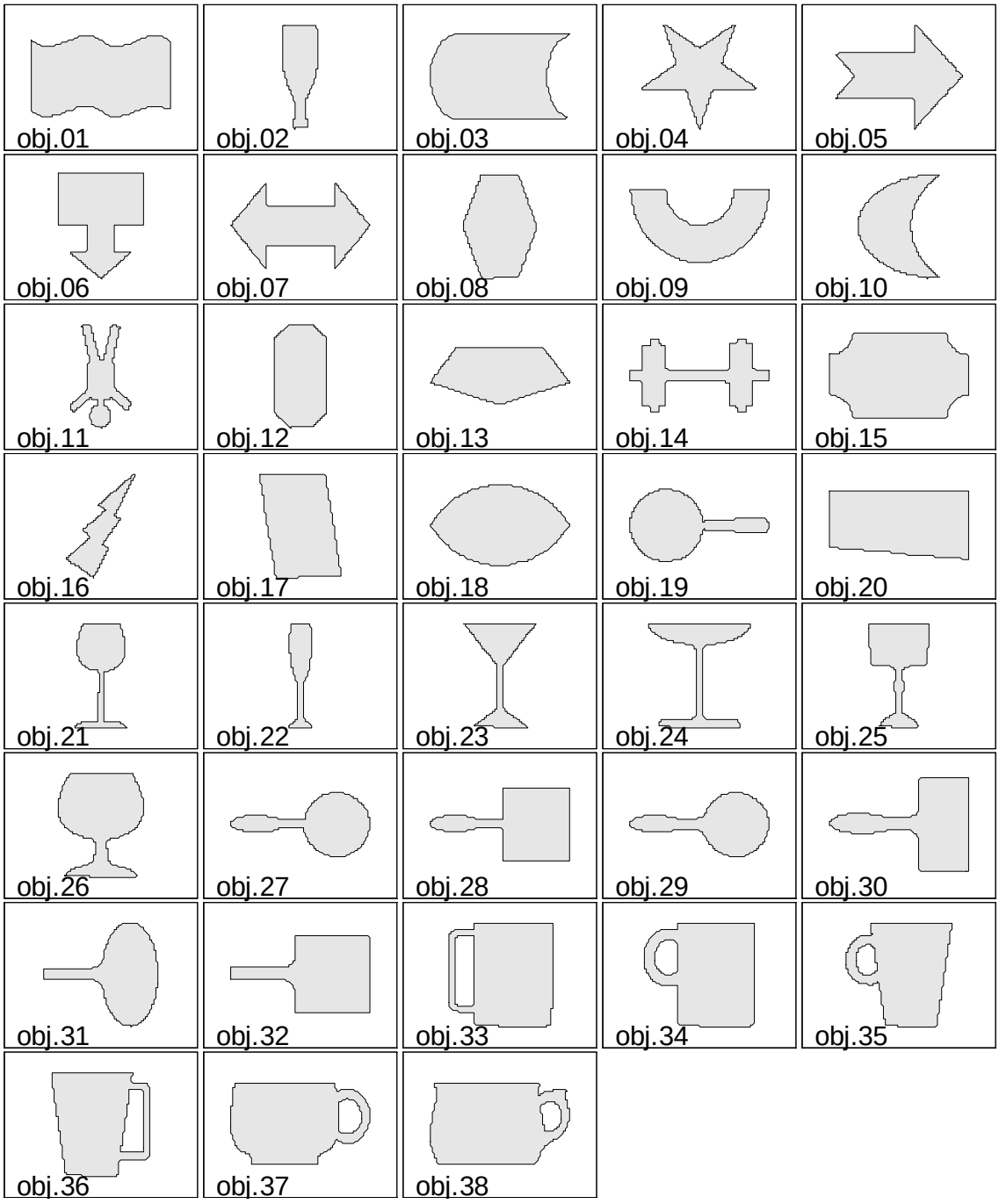
Usually, the number of iterations required for a visible change in the systems behaviour is higher than 2. Feel free to experiment the system and check the results obtained.

Object database

There are 38 different objects in our database. For each object, there are two files:

- A Matlab “*.mat” file containing all the object contours in Cartesian coordinates (x, y variables).
- A txt “*.cent” file which contains the coordinates of the centre of gravity of the object.

New objects can be added to the database just by creating both the *.mat and *.cent files. Next figure shows the 38 objects currently present in the database:



Further information

From the Matlab command window, type **help train_grasps**, **help grasp_auto**, **help leave_one_out** or **help leave_one_out_loop** in order to find more information about these functions; or just edit their codes (please feel free to modify them).