



Computer Vision: Shape Categorization Final Report

Prof. Dr.-Ing. Rainer Herpers

Senthilkumar Sockalingam Kathiresan, Matrikel Nr. 9030382
Yannick Weitz, Matrikel Nr. 9030385

Contents

1	Introduction	2
2	Material and methods	2
2.1	Description of the setup	2
2.2	PPT platform details:	2
2.3	Object details:	3
2.4	Cavity tiles details:	3
2.5	Camera details:	3
3	Approach	5
3.1	Getting rid of Unnecessary information	5
3.2	Cavity tile extraction	8
3.3	Comparison	9
4	Evaluation and results	11
4.1	Used for evaluation	11
5	Results	13
5.1	Results for normal cases	13
5.2	Results for special cases	13
6	Conclusion:	14

1 Introduction

The computer vision project "Shape Recognition" is based on a task "Precision Placement Test" given in the robocup@work competition rulebook. According to this task, a robot should be able to fit a three-dimensional object (fig) in a two-dimensional cavity tile (). The tiles will be placed in the work table called PPT (Precision Placement Test) platform (Figure1). To succeed at this task, the robot should have robust perception and manipulation capabilities.

The project "Shape Recognition" deals with the perception requirement for the precision placement task. By this project, the robot should be able to find the objects and their orientation that will fit in the cavities of the tiles, given the image of the PPT platform.

A series of digital image filters such as adaptive thresholding, canny edge filter, etc., are applied to the input image. From the processed image the tiles are extracted. The extracted cavity tiles are then cropped from the original image. Homography is done over the cropped images to get the top view of the cavity tiles. This extracted top view of the cavity tiles is then compared with the cavity tiles stored in the database to identify the best match. In addition to the images stored in the database, details on the orientation and the object fitting into the cavity are stored.

2 Material and methods

2.1 Description of the setup

From the rule book:

- A single youBot is used.
- The robot is placed by the team freely within the arena.
- The objective of the task is to pick the objects which are placed on one service area and make a precise-placement in the corresponding cavity at the service area with the special PPT platform (an example configuration is illustrated in Figure 1).

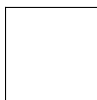


Figure 1: The PPT platform including five cavity tiles

- The task consists of multiple grasp and place operations, possibly with short base movement in between.
- The task is finished once the objects are picked up and placed in the corresponding cavities or when the time foreseen for the run ends.
- Note that the placement of the object in the cavity is finished when the object is fallen into the cavity (i.e. at least some part of the object has to touch ground floor underneath the cavity).
- All objects to be transported in a run of a team and the corresponding cavities share the same orientation, either horizontal or vertical. This may vary between different teams and different runs.[1]

2.2 PPT platform details:

- The color of the PPT platform is a light grey.
- There will always be five tiles (Figure 1)
- There will be no gap between the tiles (Figure 1)

2.3 Object details:

1. Large aluminium profile
2. Small aluminium profile
3. Large nut
4. Small nut
5. Screw
6. Cylinder



Figure 2: six objects which has to be placed in the cavity tiles [2]

2.4 Cavity tiles details:

- There are 11 tiles that will be used in the competition
- Each tile measures 14 x 14 cm
- The color of the tile is close to white
- The material is 3D printed

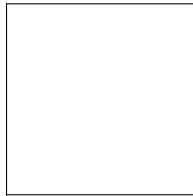


Figure 3: Cavity tiles for the objects

2.5 Camera details:

- Camera model fixed to the robot is **asus xtion pro live** (Figure 4)
- **Intrinsic parameters**
 - height: 480
 - width: 640
 - distortion model: plumb bob
 - D: [0.0, 0.0, 0.0, 0.0, 0.0]
 - K: [570.3422241210938, 0.0, 319.5, 0.0, 570.3422241210938, 239.5, 0.0, 0.0, 1.0]
 - R: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
 - P: [570.3422241210938, 0.0, 319.5, 0.0, 0.0, 570.3422241210938, 239.5, 0.0, 0.0, 1.0, 0.0]

- binning x: 0
- binning y: 0
- roi:
- x offset: 0
- y offset: 0
- height: 0
- width: 0
- do rectify: False



Figure 4: Asus xtion pro live

Prerequisites

For the precision and placement test the youBot will get the command to pick up a particular part from a toolbox, from the referee unit, and then drive back to the table, where the cavities are stored. This is where the project begins. The camera of the youBot is pointing at the table. Five out of twelve plates will be placed in a cavities inside the table, arranged in a row (fig. 5). The background of the cavities inside the tiles will thus be dark and the full tile will always be visible. The tiles themselves have a nearly white color and will be produced by a 3D printer. The size is fixed to $14cm^2$, with a thickness of $1cm$. The object, that the youBot is holding is already recognized and known to the bot. There are 5 different objects available.

The cavities are always centered in the middle of the tile and will only correspond to only one particular orientation of the object.

The youBot is using a asus xtion pro live for this task. The camera will be positioned in front of the tiles, with different possible angles, always capturing all five tiles.

There will be no additional light source attached to the youBot, so shadows are possible.

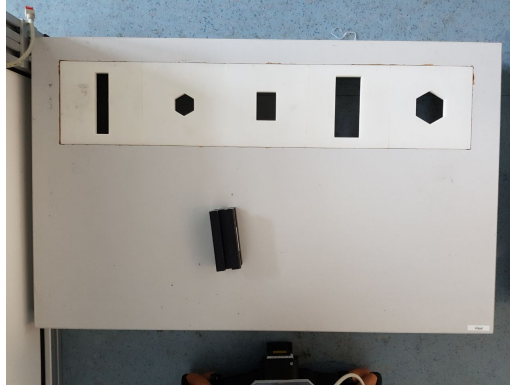


Figure 5: Example of a set of five tiles arranged like they will be in the Robocup@Work competition.

3 Approach

As a preprocessing step the image input has to be normalized. Since it can't be assumed that the youBot will always provide a 90° top-down view of the tiles, the input image should be transformed using homography. The new image after this step should only show an approximate of what the row of tiles would look like from the 90° top-down view. The result will then be further decomposed into the single tiles.

To classify the extracted tiles, a comparison between the tiles and a database is performed. The database consists of example images showing all the tiles in all possible configurations.

The project is divided into three phases:

1. Getting rid of Unnecessary information
2. Cavity tile extraction
3. Comparison

3.1 Getting rid of Unnecessary information

In order to normalize the input image and rectify the row of tiles, the first step is to crop the image and get rid of unnecessary information (fig. 6).

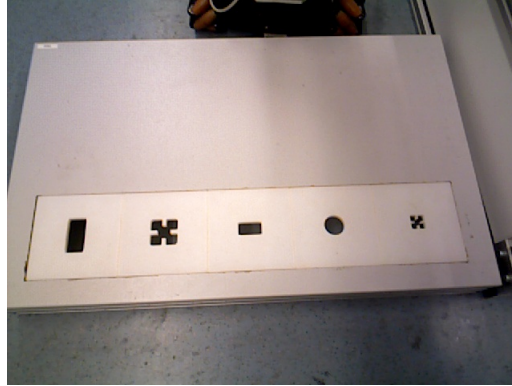


Figure 6: The Original image containing irrelevant information, like the table or small parts of the robot.

Edge filter

so at first the area that should be cropped has to be determined. Since the images are in general a lot brighter than the cavities, an edge filter is applied (fig. 7). Threshold and adaptive threshold have been tested but didn't deliver good results.

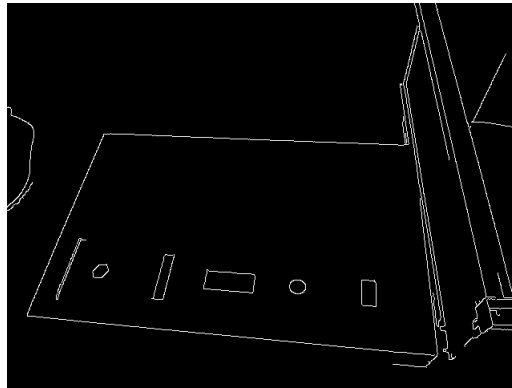


Figure 7: The image after a canny edge filter has been applied.

Contour extraction

Using the findContours method all contours of the preprocessed image are extracted and the pixel coordinates of the center points are stored (fig. 8). Since all contours should lie in the middle of the tiles, the centers of the cavities should lie in a line consisting of five equally spaced points.

The first attempt to this problem was to use blob detection in order to extract dark areas in the image. This method failed due to the multitude in cavity shapes and many other dark regions in the image. Also the datastructure of the blobs made it hard to enter the information.

Finding the center of the cavities

To find the five points that correspond to these centers, a line is drawn between all extracted centers from substep one. From all the lines where at least five points lie closely on the same line, the one is chosen where the error between the points and the line is the least (fig. 9). This has proven to give the correct line for all tested images. Since we search for the minimum distance, lines with five points are preferred over lines with more points, so by default lines with too many points get ruled out.

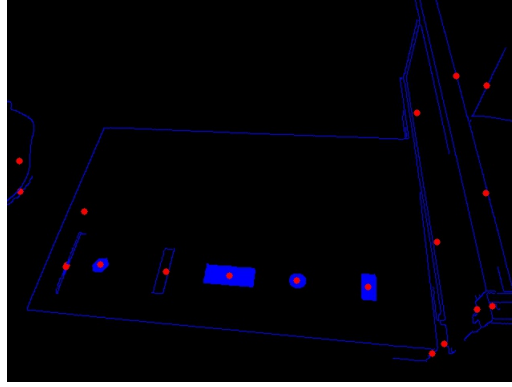


Figure 8: All the extracted contours from the edge detection image.

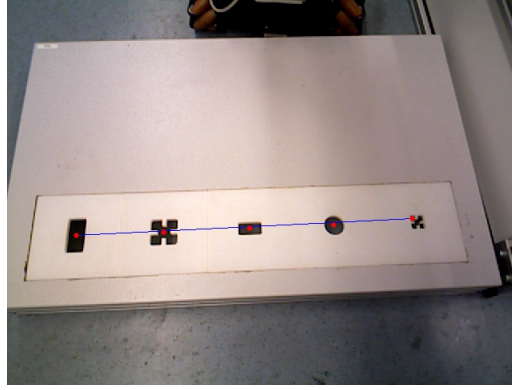


Figure 9: The five points that lie closest to a line have been extracted.

Cropping out unnecessary information

We know that all tiles have the same dimensions. Thus, the next step aims to find a correspondence between the pixels and the distance, or pixels per centimeter. The center points of the cavities are expected to lie in the center of the tile. It follows, that the distance between two neighboring centers is equal to $14cm$. Hence, The approximate pixel equivalent to $14cm$ is calculated by getting the mean distance between two neighboring tile centers.

Now the image can be cropped by using (fig. 10):

$$cropped\ image = image[\min(x) - \frac{\mu_{dist}}{2} : \max(x) + \frac{\mu_{dist}}{2}, \min(y) - \frac{\mu_{dist}}{2} : \max(y) + \frac{\mu_{dist}}{2}] \quad (1)$$

Where:

$\frac{\mu_{dist}}{2}$ = Pixel per 7 Centimeter, plus some additional term to compensate the angular orientation of the five tiles

$\min(x)$ = Minimum x value of all center points

$\max(x)$ = Maximum x value of all center points

$\min(y)$ = Minimum y value of all center points

$\max(y)$ = Maximum y value of all center points



Figure 10: A lot of irrelevant information has been cropped out of the image.

3.2 Cavity tile extraction

Corner detection

To extract the corners of the five tile row, the adaptive threshold filter is applied to the cropped image, since this has proven to give a good differentiation between the five tile row and the background. Again the contours of the image are extracted, this time with a limitation of the area, that the contour should enclose (fig. 11). The original approach used the `goodFeaturesToTrack` method. Because of the low resolution of the image the algorithm detects too many corners. To overcome this a method of choosing the correct corners is needed, but after multiple attempts, this approach was replaced by hough line transformation. The idea was to get the points, where the hough lines meet. However, the algorithm didn't extract good lines, that fit the sides of the contour and was thus discarded.



Figure 11: The contour of the image that corresponds to the biggest area of the cropped image.

The low resolution of the image makes it hard to extract the four corners immediately. The Harris corner detection is used to extract the four corner points, since these are a little misplaced the `cornerSubPix` method as described in [2] is used to extract the corners more precisely (fig. 12).



Figure 12: The two extracted corners are shown, red shows the Harris corner, while green shows the corner with sub pixel precision.

Homography

The four points are used to transform the perspective of the tiles to create an artificial top-down image containing only the five tiles (fig. 13).



Figure 13: An artificial top-down view of the five tiles.

Cropping cavity tiles

The resulting image is then divided into five equal parts and the images are stored (fig. 14).

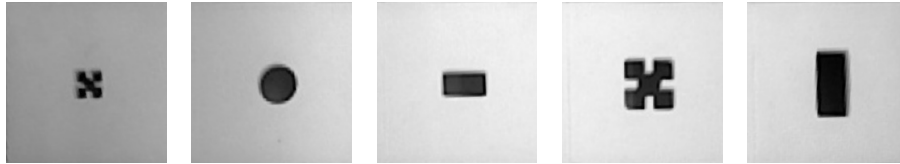


Figure 14: The five cropped individual images of the tiles.

3.3 Comparison

The last phase of the project is to find the object that will fit into the cavity tile that is extracted from the input image. This last phase is approached by comparing the extracted image with all the possible orientations of the tile stored in the database.

creating Database

- There are eleven tiles used in the competition.
- In the eleven tile
 - five tiles have only one possible orientation

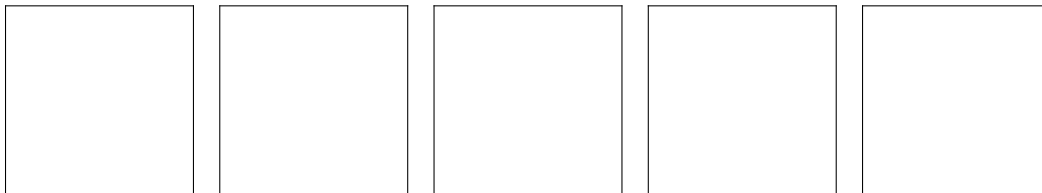


Figure 15: Tiles with one possible orientation

- Cavity tiles have two possible orientation

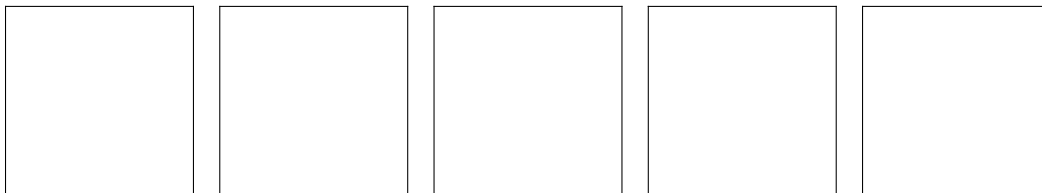


Figure 16: Cavity tiles with two possible orientations

- One tile has four possible orientation



Figure 17: Single cavity tile with four possible orientations

- Images of all cavity tiles in all possible orientations [total 19 images] are stored in the database
- The title for each stored image consist of the details of the object (that will fit in the cavity tile), object's orientation and cavity tile's orientation



Name: **Screw_Horizontal_270**
Meaning of the name:
Object that fits the cavity tile: **screw**
Orientation of the object: **Horizontal**
Orientation of the cavity tile: **180 degree**

Name: **SmallNut_Horizontal_00**
Meaning of the name:
Object that fits the cavity tile: **SmallNut**
Orientation of the object: **Horizontal**
Orientation of the cavity tile: **0 degree**

Name: **SmallNut_Vertical_0**
Meaning of the name:
Object that fits the cavity tile: **SmallNut**
Orientation of the object: **Vertical**
Orientation of the cavity tile: **0 degree**

Mean Square Error(MSE)

The Mean Square Error (MSE) image comparison method is based on the formula:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2 \quad (2)$$

With I and K being the two compared images.

In MSE, both images are converted into the same size. Then, the difference (error) between each pixel of both the images are taken and squared. The summation of all the error square will give the MSE value. The lower the error, the more similar are the images.

There is another technique for image comparison called Structural Similarity Measure (SSM).

Structural Similarity Index Measure (SSIM)

The SSIM models the changes in the structural information of the images. While MSE models the error.

It is based on the formula [3]

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_1)} \quad (3)$$

where, μ – mean σ – variance c – covariance

This is used to compare two windows rather than two images

The output value ranges from -1 to 1, where 1 represents a perfect match



Figure 18: From the left. First image shows the extracted tile image, that is given as an input to the MSE or SSIM comparison algorithm. The second image is the best match found in the database using MSE method. The third image shows the best match found using the SSIM method.

For better efficiency

For the comparison techniques seen above perform well, the images are preprocessed:

- The image is converted into **gray scale**.
- Both images to be compared are **resized** to an equal size.
- A **binary threshold** is applied to the image, in order to increase the contrast between cavity and tile, to normalize the colors of the images (get rid of shadows and non-homogeneous lighting), but mainly to provide a clear shape of the cavity (Fig:19)
- **Median blur** is applied to get rid of the salt and pepper noise (Fig:19)



Figure 19: From the left. First, the image after applying a threshold. Second, after using the median blur.

Optimisation:

Not yet implemented

When the difference between the best match and second best match is very low If in MSE the difference between the top two best matches is very low, then the SSIM result is considered.

When best match of MSE and SSIM are different From the input image the contour is detected and the area is calculated. This area is compared with the area of the cavities calculated from the database cavity tiles. The best match by area comparison is used to decide on the database image.

4 Evaluation and results

4.1 Used for evaluation

- Eight images have been used for evaluation purpose (fig. 20).
- All images have the PPT platform in it but from different angles with different cavity tiles.
- Five of the images show five tiles, placed in the PPT platform, with regular lighting, no occlusion and no other objects.
- The three remaining images have been created to test extreme conditions.

- For the five regular images, some tiles have specifically been chosen to check whether the system is capable of differentiating the size, while the rest is selected arbitrarily.
 1. The first image consists of simple shapes, where the focus is placed on differentiating between the same shape with a different size, which could give problems due to the similarity of the two shapes (fig. 20, top, most left).
 2. The second image is introducing more complex shapes, that could be harder to classify and recreate while using homography (fig. 20, top, second from the left).
 3. In the third image only small similar looking shapes have been chosen to test the algorithm for a number of similar shapes (fig. 20, top, second from the right).
 4. The fourth image shows four variations of the same 2 dimensional shape, that fit to different 3 dimensional objects (fig. 20, top, most right).
 5. The fifth image is used to determine how the algorithm works for similar shapes with different orientations (fig. 20, bottom, most left).
- For the three remaining sets the focus is set to special conditions rather than testing the algorithm for the distinction between different shapes.
 1. In the sixth image, the number of tiles has been reduced to four in order to test border cases (fig. 20, bottom, second from the left), the algorithm is not expected to solve this problem, it is just an additional case.
 2. The Seventh image shows images that correspond to a problem that occurs if other areas in the image have the same color and shape of the cavities (fig. 20, bottom, second from the right).
 3. In the last image, the light-source has been switched of to create a darker image (fig. 20, bottom, most right).
- It can easily be seen, that each example image from the eight datasets has different areas of lighting, thus no dataset testing for shadows was necessary.

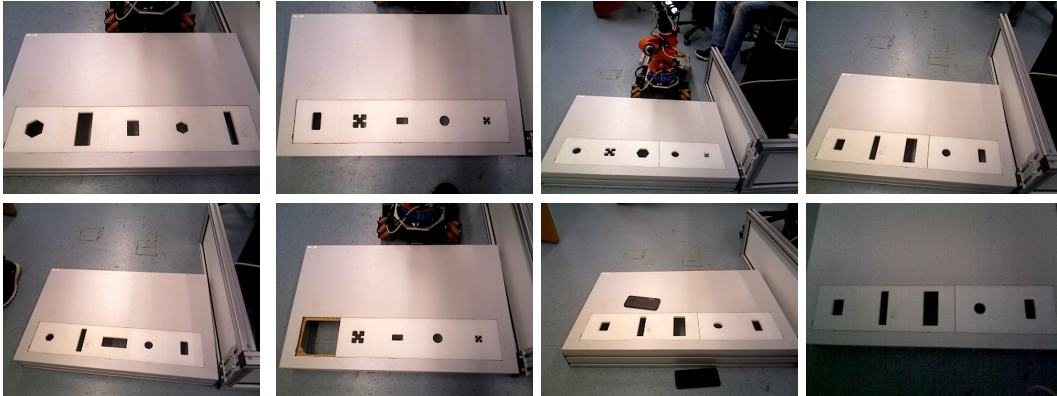


Figure 20: Example images for the eight datasets, starting with one in the top left corner, to eight in the lower right.

5 Results

5.1 Results for normal cases

Operations	Image01	Image02	Image03	Image04	Image05
Center of cavity detection	✓	✓	✓	✓	✓
Cropping	✓	✓	✓	✓	✓
Detecting the five tiles	✓	✓	✓	✓	✓
Detecting the corners	✓	✓	x	x	✓
Homography	✓	✓	x	x	✓
Number of cavity tiles extracted	5	5	0	0	5
Number of objects matched	5	5	0	0	4

Summary

- The algorithm is able to robustly detect the center of the cavity for all five tiles in the normal test cases
- When there is a slight gap between the cavity tiles in the PPT platform, the contour detection finds two contours
- Hence the corner detection also finds the corners of only one set of cavity tiles
- The corner detection on the cropped image is not working correctly for each image
- The comparison of images using MSE was able to find the correct match for 14 out of 15 tiles during the test run
- When the clarity of the image is too poor the comparison technique fails
- As shown in the Fig:21, the object for the first tile is supposed to be small nut, but the algorithm identifies it as a cylinder. This is because there is only a minimal difference between the hexagon and the circle.



Figure 21: The results for the fifth image. The top left image shows the original input image. the 5 remaining images show one of the extracted tiles along with the identified object and orientation shown in the captions.

5.2 Results for special cases

Operations	Image06	Image07	Image08
Center of cavity detection	✓	✓	x
Cropping	✓	✓	x
Detecting the five tiles	x	✓	x
Detecting the corners	x	x	x
Homography	x	x	x
Number of cavity tiles extracted	0	0	0
Number of objects matched	0	0	0

Summary

- The algorithm doesn't work properly on dark lighting

6 Conclusion:

References

- [1] Robocup @work rulebook, p. 35. Robocup-rulebook2017
- [2] Webpage, last viewed 13.12.17 20:25, https://docs.opencv.org/3.1.0/dc/da5/tutorial_py_drawing_functions.html
- [3] Comparing images with ssim and mse
<https://www.pyimagesearch.com/2014/09/15/python-compare-two-images/>