Final Project
Image processing and computer vision

D'Ortona Cristian† - Lella Luigi†

July 29, 2022

† These authors have equally contributed to this project

Abstract

The carried out project's aim is to analyze the features of motorcycle rods, which are later on classified in order to allow a Robot to pick them up according to their characteristics.

1 Rod Characteristics

The scope of the project was to calculate, for each rod, the following characteristics:

- Type
- Orientation
- Barycenter
- Length
- Width
- Barycenter Width
- Number of Holes and their diameter and barycenter

In order to tackle this problem we have implemented in Python an object named *Rod* which lets us keep track of all these aforementioned characteristics. In the provided images, there are two different kind of rods which we will label as A and B according to the number of holes they have: the former one has one hole, whereas the latter has two. All the information regarding the analyzed rods are saved in a CVS file.

2 First Task

2.1 Inspection Workflow

The inspection of each image has been performed following a common workflow generally employed in Machine Vision applications as illustrated in the Figure 1.

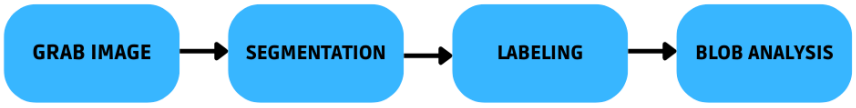

Figure 1: Inspection Workflow

The first step is opening the under analysis image in our Python script with the aid of the OpenCV library and then convert it in a gray-scale picture. The pictures have been taken in a controlled environment where the back-lighting technique has been used, hence their histogram appears inherently binary with two distinct modes which is essential for the algorithm later on used. The second step is *segmentation* where we wish to acquire semantic knowledge from the scene; in particular, we can define a vector-valued function p(x,y) whereby we define a set of image properties. In this machine vision scenario, however, such function will only contain information regarding the intensity of the pixels. Regarding segmentation, we wish to acquire two disjoint homogeneous regions that have in common the intensity; such regions are classified as background and foreground and as already stated they bring semantic knowledge about the scene.

Since the histogram is bimodal it is possible to define a threshold which tells apart the two modes, however it is common that images might be affected by different light conditions hence resulting in the inability of using a fixed threshold. Therefore to tackle this issue we have implemented the binarization using the Otsu's algorithm which automatically computes the threshold for each picture given as input to our script, although it is computationally less efficient since it has to compute the threshold. The Figure 2 illustrates the picture in gray scale and the respective binarized one.

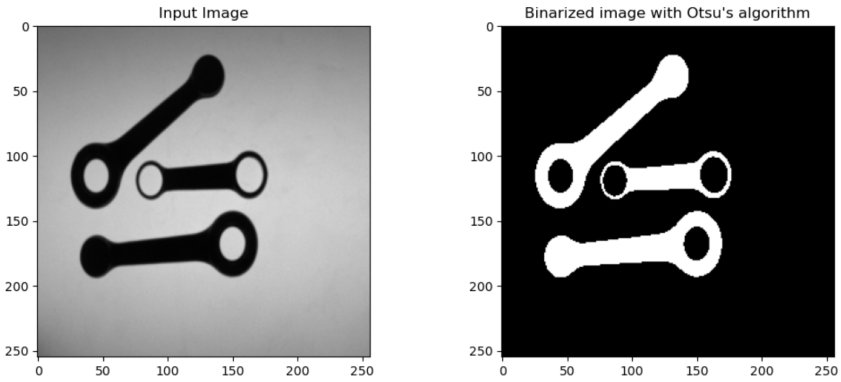

Figure 2: Binarization

The rods which are the objects we are interested in analyzing share the same intensity level equal to 255, hence they are part of the foreground; whereas the black pixels denote the background.

The foreground contains all the rods present in the image, however, in order to further process them and extract the needed information from each one of them, a *Connected Component Labeling* has to be performed. This task is carried out in the third part of our workflow and it is essential in order to tell apart the different objects, in this case rods, that are part of the foreground.

In openCV the function *connectedComponentsWithStats()* has been used and its output are the number of labels found and a matrix which has the same size of the input image and contains different integers to identify the different connected components(rods). Moreover, in order to better visualize the blobs, a RGB mask has been implemented and the final result is illustrated in the Figure 3.

Lastly, the fourth phase of our workflow involves the analysis of each connected component with the aim of obtaining the needed features of each rod; this process is generally referred to as *Blob Analysis*.

2.2 Blob Analysis

The first feature we wish to compute is the type of each rod which depends, as already stated, on the number of holes: type A has one hole, whilst type B has two.

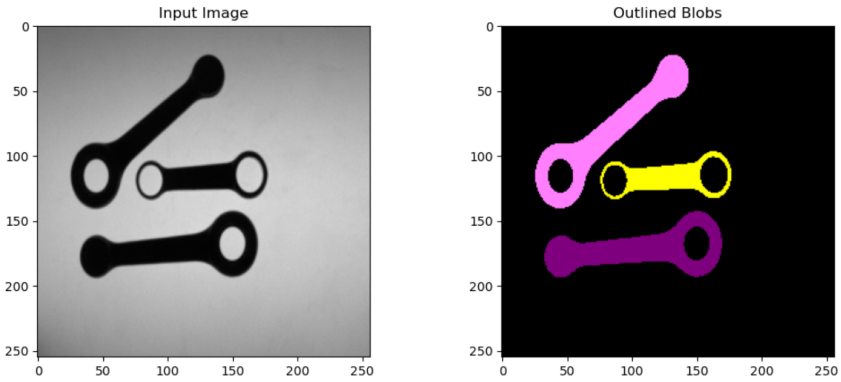

Figure 3: Connected Components Labeling

This task is accomplished during the connected components labeling phase where each blob detected is checked in order to make sure it is a rod (it will be further explained in the Chapter 3). To accomplish this task the openCV function *findContours()* has been used; it has been called for each connected component found and the outputs are a list of 3D Matrices, which stores in each channel pixel coordinates for each point of the contour, and a hierarchy matrix. The hierarchy is essential in our case, it is shaped as a 3D matrix which has a number of rows equal to the number of contours that are found in the analyzed connected component, whereas the number of columns is fixed and equal to four. Each element of the column provides a different information regarding the found contours, these are as follow: next, previous (they respectively refer to the next and previous contours of the same hierarchy level), child and parent (which are not used in our case).

We analyzed the hierarchy matrix in order to define its number of rows which corresponds to the number of contours detected and they are the external one outlining the rod itself and the inner contours of the internal holes. The trivial formula has been used to define the type of the rod according to the number of holes:

$$num\_holes = hierarchy\_rows - 1$$

As stated in the Chapter 1 we need to compute for each hole its diameter and barycenter, hence we first have to compute the contours'

features needed such as the moments and perimeter. The moments are employed in order to compute the barycenter using the following trivial formula:

$$c\_x = \frac{M_{10}}{M_{00}} \qquad c\_y = \frac{M_{01}}{M_{00}}$$

Furthermore, the diameter of each circle is computed using the inverse formula of the circle's perimeter.

Unlike the previous case, from now on we will be using the features obtained from the connected components in order to compute the remaining rod's characteristics. The first feature to be computed is the orientation of each rod which is defined as the angle between the major axis and the horizontal axis. The major axis is the line through the barycenter with the least moment of inertia, hence it can be seen as an optimization problem where we wish to minimize the moment of inertia. Since the moment of inertia is a function of the angle θ then we find that the angle which minimizes such quantity has the following formula:

$$\theta = -\frac{1}{2}\arctan\frac{2M_{11}}{M_{02} - M_{20}}$$

The found θ can either be referred to the orientation of the minor or major axis, hence in order to tell them apart we computed the second derivative and analyzed the sign.

Now we can compute the equation of the major and minor axis in the camera reference frame after finding the value of α and β which are the projection of those axis on the two horizontal and vertical axis.

An example of the orientation of the major axis is depicted in the Image 4

Besides, the width at the barycenter can be computed by calculating the distance between the two points which are the least distant from the minor axis and they will respectively be either on the left or right of the major axis according to the sign of the distance from the major axis.

The last two rod's characteristics which are the width and length are computed as features of the *MER (Minimum Enclosing Rectangle)*. In order to define the MER the contour of each component is needed, therefore we have used the morphology operator *Erosion* and the resulting image has been subtracted by the original image obtaining the external contour. Then four points are computed, which refer

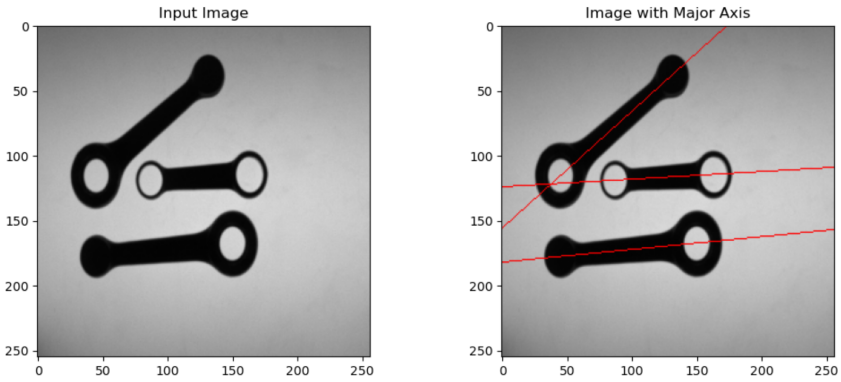

Figure 4: Rods with their major axis

to those pixels which are the further away from the major and minor axis.

We then trace four lines going through the four points and they will intersect in four additional points which are the vertexes of the MER. The width and length are calculated by computing the distance of two collinear vertexes.

3 Second Task

The second task of the project faced the handling of different problems one might encounter when working in a machine vision environment. The workflow has been updated as illustrated in the Figure 6.

3.1 Distractors Handling

So far the images we have dealt with only contained motorcycle rods, however it is not uncommon to have images where there are other objects that are not of interest and we do not wish to further analyze. In our project the distractors we have to filter out are illustrated in the Figure 7 and are of two kinds: washers and screws

In our program right after the connected component labeling, while we perform the rod identification as described in the Chapter 2.2, we check each computed component to asses whether it is a blob of interest or not.

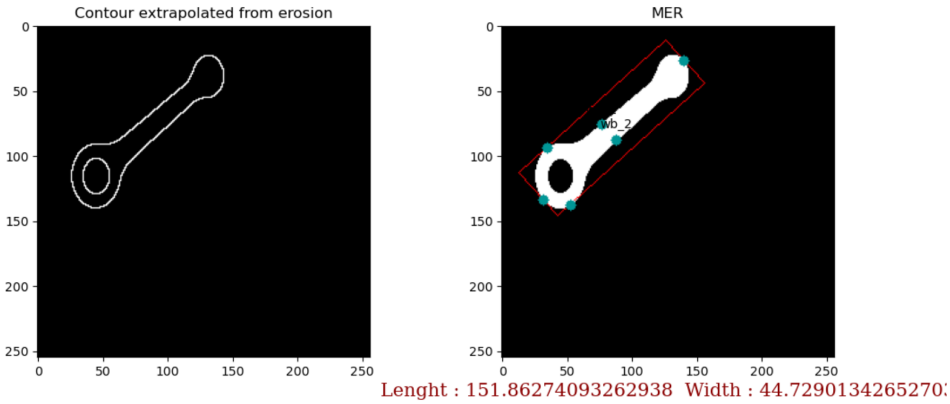

Figure 5: Contour and MER are computed for each rod

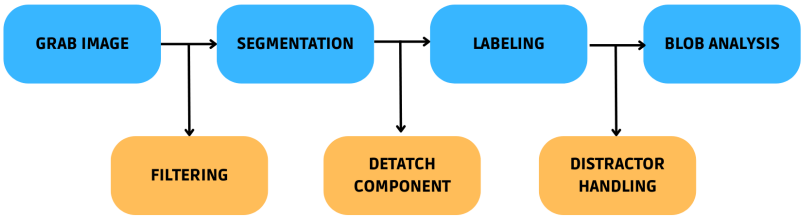

Figure 6: Updated Workflow

Regarding the screws, due to their shape, they are inherently easy to filter out: we use the Hierarchy matrix which we get as an output from the *findContour()* function and since the number of rows are equal to the number of contours detected for each blob, if we only have one row then the connected component under analysis must be a screw.

Regarding the washers, they have a circular shape and also contain a hole, hence the approach used to deal with screws can not be used as the number of rows of the hierarchy matrix is equal to two and it would be wrongly classified as a rod of type A. Instead we decided to compute the circularity of each blob which is used to detect circles; theoretically the value would be equal to 1 in the case of a circle, however we can not have perfect circles hence we defined a threshold

equal to 1.16 so that any object which has a circularity smaller than the threshold is classified as a washer. The formula for computing the Circularity is the following:

$$circularity = \frac{perimeter^2}{4 * area * \pi}$$

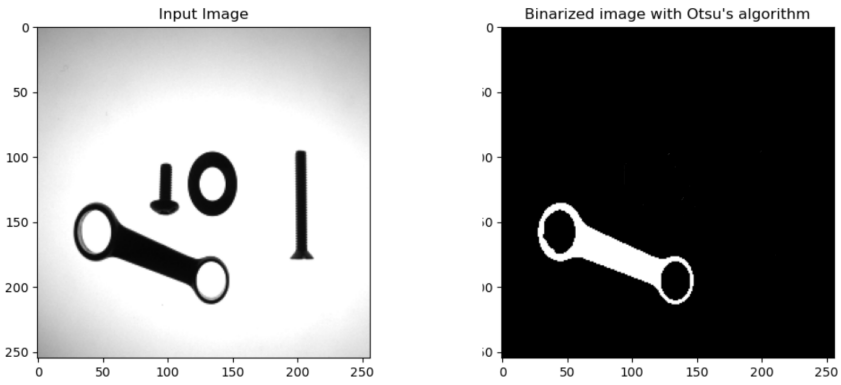

Figure 7: Example of an image with distractors

Once the distractors have been cut out, the resulting blobs are the correct rods which are appended to a list of rods.

3.2 Contact Points Handling

Some of the pictures which were given to analyze presented an issue where one or two rods where touching with one another. This problem hindered the connected component labeling since the attached rods are labeled as an unique blob, whereas we are interested in distinct connected components. Due to the shape of the rods, we deemed that the attached rods can be told apart by calculating the convex hull that encloses the attached connected component and then analyze the convexity defects. Therefore in order to compute the convex hull we have used the OpenCV function *convexHull(Contour contour, returnPoints = False)* while the convexity defects are computed by the function *convexityDefects(Contour contour, ConvexHull hull)*. The defects are stored in a 3D matrix which has a number of channels equal to the number of defects found in the convex hull, moreover each channel is a matrix which has one row and four columns. This matrix contains

the indices used in order to index the contour matrix, in particular the four columns refer to: starting and end point of the convexity defect, the point of the contour the furthest from the convex hull, the distance from this point and the line connecting the start and end point of the convexity defect.

We have created a list which contains a a pair of points that are the coordinate of the point belonging to the contour the furthest from the convex hull and their distance. Then we have sorted this list in an ascending order according to the distance. From a qualitative analysis we have noted that the contacting points have distance from the convex hull greater than the others, hence we are taking into account only the last four points of the list. There might be several ways to connect these four points with one another, despite this, we wish to connect together with a line only two two which are the closest with each other because they refer to the same contact point.

In order to deal with this, we check that the distance between any of the two points is smaller than 30, in case this condition is not fulfilled we switch to another pair of points until the condition is satisfied. Once the two points are found, we trace a black line in the binarized image in order to single out the rods.

The results are illustrated in the Figure 8.

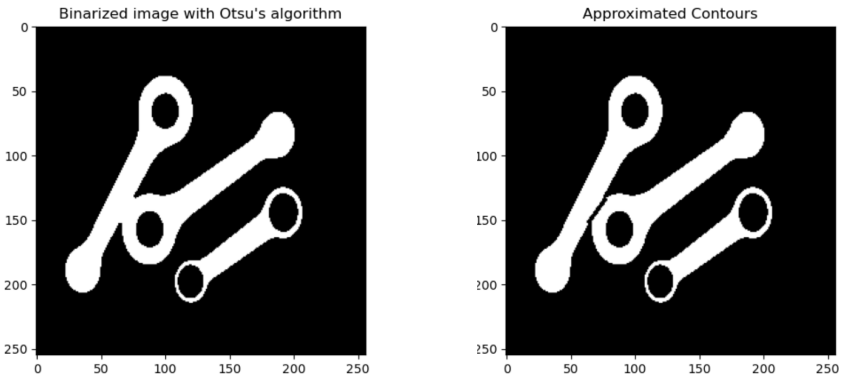

Figure 8: Rods may have contact points

In some cases the adopted solution fails as it wrongly classifies rods; this is because we trace a line connecting points with no awareness of the object's shape. An example of this issue is depicted in the Image

9, where the rod of type B will be wrongly classified as a rod of type A in the connected component labeling phase.

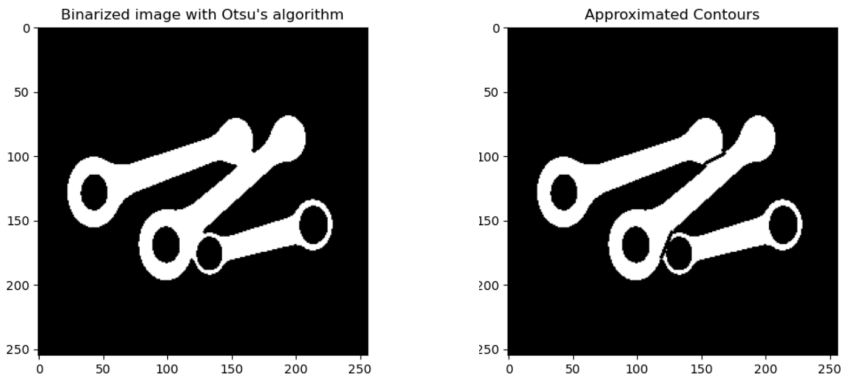

Figure 9: In some cases misclassifications may occur

3.3 Iron Powder Handling

When working in an industrial scenario images are often affected by production waste such iron powder in our case. Whenever we binarize an image which is affected by such defects, as illustrated in the Figure 10, the small particles of powder are classified as foreground and this leads to have many ill-defined blobs for each iron powder particle. In order to tackle this problem, the image must be filtered before the connected component labeling with the aim of removing the powder which can hinder the analysis. Since the iron powder can be modeled as salt and pepper noise, we deemed that the use of a median filter with a 3x3 kernel, iterated four times, would have been best. The result is illustrated in Figure 10.

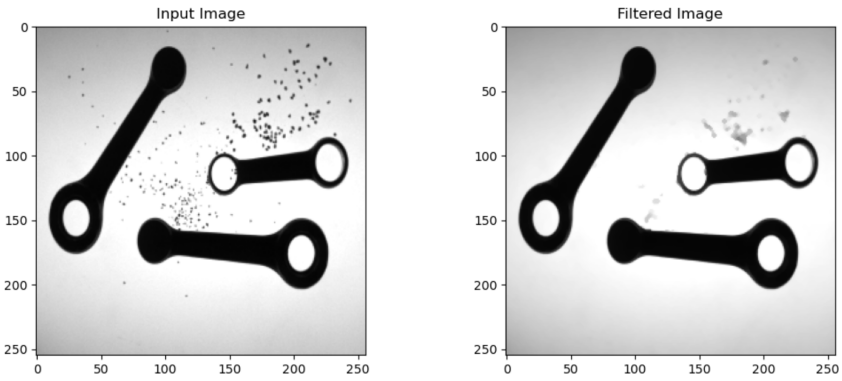

Figure 10: The iron powder problem is solved by applying a median filer