VISUAL INSPECTION OF CONNECTING RODS

Alessio Comparini, Alice Gibellato

28th April 2022

1 Introduction

We chose to develop the project ”Visual Inspection of Motorcycle Connecting Rods”. Given a series of different images, for each image the program is able to identify all the information required for the scope of the project. To achieve the goal, we used computer vision techniques to separate the joined rods, eliminate noise from the images and ignore objects that should not be analysed. The programme is able to identify the following information for each image:

- Type of rod
- Position and orientation
- Length, Width, Width at the barycentre
- For each hole, position of the centre and diameter size.

2 Performance of the project

2.1 Image segmentation

The first part of the project focuses on the segmentation of the image to separate objects from the background. This segmentation is done to distinguish the information part related to the rods even though the images were taken with the backlight technique and the rods are easily distinguishable from the background. Binarization techniques allow to separate the object from the background and then analyse only the pixels of the object. Considering the various techniques with which segmentation can be performed, we decided to use the Otsu algorithm already implemented in the OpenCv library. We use this algorithm because it automatically determines the threshold value. Otzu is advantageous because it allows a different threshold for each image and adapts the threshold more easily to variations in illumination.

2.2 Removing of scattered iron powder

To remove the iron dust present in some of the images in the dataset, we decided to use the median filter. In particular, *medianBlur()* provided by the OpenCv library has been used.

In the program the filter is applied only to those images that present some noise, and, in order to distinguish them from those that don't have noise, has been counted the number of objects recognized by *findCountour()*: if it's < 10 the image is not considered noisy, otherwise it is. We have chosen the median filter because it is very advantageous in eliminating impulse noise. This non-linear filter works very well on this type of image because this powder introduces outliers into the image. The median filter was applied by a cascade method because some noise components were not easily removed. Exactly we applied the median filter 4 times in order to have a total removal of noise on all the images involved.

2.3 Extraction of connective rods

After removing the iron dust from the images and after implementing the image segmentation, the *connectedComponentsWithStats()* function was used to complete the extraction of the objects and isolate the connecting rods. Connecting rods are classified according to the number of holes they have and we decided to use the *findContours()* function available in the OpenCV library. This function extracts the contours of the objects in the image and hierarchizes them into a tree. In the tree generated by the function, each node corresponds to a contour identified in the image. If in the image a contour K is immediately contained in a contour L, then in the tree node K is a child of node L. In this way, after identifying the contour of a connecting rod, it is sufficient to recognize in the corresponding hierarchy how many children it has: 1 child only means it has only 1 hole and the rod is type A, 2 children means it has 2 holes and the rod is type B. This algorithm is implemented in the aorb() function.

2.4 Joined connective rods

In some images 2 or more connecting rods have connection points, which is a problem because the *connectedComponentsWithStats()* function would have identified the connected connecting rods as a single object. To solve this issue first it was necessary to identify the objects to be separated. The method adopted was to extract the contours of the segmented image and calculate the area of each one. If the area is greater than an arbitrarily fixed value (in our case 6000 was chosen because the largest connecting rods have an area of about 5000) then the function *findHotspots()* is called passing as argument the contour in question. This function has as main objective to identify the convex points within the contour which are then used by the function *separateRods()* to draw a background line that connects them two by two in order to separate the connecting rods from each other. The method used to identify convex points is to scan the entire contour in groups of 5 adjacent pixels, and calculate the distance between the two farthest ones. In the presence of convex points, in fact, the distance between these two pixels will assume a smaller value than the distance in the absence of a convex point, and assuming as threshold a distance of 4 they are recognized without problems. After identifying the convex points, they are paired 2 by 2 according to their distance, and a straight background line is drawn between them. Finally, since some connecting rods have a very narrow margin around their holes, simply drawing a straight line would correspond to the action of opening a hole to the background, regardless of the width of the straight line. To solve this last problem, it was sufficient to draw, for each background line, also two other foreground lines, shifted respectively of +0.3 and -0.3, in order to preserve the holes inside each connecting rod after separation.

2.5 Position and orientation

To calculate the position of the connecting rods, we found the barycentre of each connecting rod. The function *getOrientation()* finds the barycentre and uses the Principal Component Analysis algorithm to find the orientation of the connecting rod. This algorithm takes as input the vector of foreground points of the connecting rod and projects these points onto two new reference axes. The first axis represents the highest variance and the second axis represents the lowest variance. We then calculated eigenvectors and eigenvalues of the covariance matrix to identify the orientation of the major and minor axis relative to the horizontal axis.

2.6 Length and width

The *getLenghtWidth()* function finds the length and width of each connecting rod. To find this information we used the *minAreaRect()* function available in the OpenCV library. This function takes the foreground points as input and finds the enclosing rectangle of minimum size. We found the length and the width of the connecting rod by taking the length and the width of the enclosing rectangle drawn around the connecting rod.

2.7 Width at the barycentre

In order to find the width at the barycentre of each connecting rod, we used information already calculated in the previous points, i.e. connecting rod contour, position of the barycentre and angle of inclination of the connecting rod. The *getWidthAtBarycenter()* function gets as input the position of the barycentre and the angle of the major axis. The function finds two points in particular by finding the intersection of the line passing through the barycentre, perpendicular to the major axis of the connecting rod, and the points of the connecting rod contour. Then the width at the barycentre is equal to the distance between these two points.

2.8 Position of the center and diameter size for each hole

Once the contours have been extracted with the *findContours()* function,in particular the ones of the holes, since the holes have a circular shape, the center of each hole correspond to the barycenter of its contour. To find the diameter we calculated the distance between a random contour point and the centre of the hole. The *holeAnalysis()* function takes as input a connecting rod contour with its hierarchy and returns its position and diameter.