

Report of Homework of Mattia Sanchioni

A.Y. 2018/2019

In this homework we have to apply the techniques learned during the lectures in order to complete some tasks.

Given two images of two different cars we have to find some important features such edges, ellipses or symmetric point. With this information, later, we calibrate the camera (zero-screw, but not natural), localize it and find the position of some points.

Let's analyze tasks point-wise.

Extraction of image features

First of all images are uploaded, the photo used in the homework is '*Image1*'.



After that it is transformed in grey scaled:

```
% both transformations are equivalent  
rgb2gray(image1);  
0.299*images(:,:,1) + 0.587*images(:,:,2) + 0.114*images(:,:,3);
```

Now we can start with the first point in which we have to select the most important features useful for the following points.

The approach, that I have used to extract information from image, is to select a restricted area in image.

In order to do this I wrote the function `selectRegion()` that, given an image, allows to select a section and return the selected region and the coordinates of the rectangle, necessaries to translate the features selected in the region and then visualize in the original image.

After that the selection is derived for having a better visualization of the edges. During the fourth laboratory professor Boracchi showed us the code to make edge detection. I used this code adding the possibility to chose different threshold such that binary or hard threshold.

The edge detection is implemented in the function `findEdges()` in which it is possible to choose *binary*, *hard* or *Canny* thresholds.

All these functions are used in `findConic()` with which it is possible to detect the conics that represent the wheels. The detection is made with 5 points, because the automatic implemented in `findEllipses()` requires some filters applied to the image as erosion and dilation (not done due to lack of time), because it works well but, because of the noise, a lot of ellipses are detected but not the wheels.

To calculate the conic matrix from 5 points belonging the conic, we used the right null space that is the vector that right multiplied return the null vector. In my code the matrix A represents the system of equations of points belonging the conic.

$$\mathbf{x}^T * \mathbf{C} * \mathbf{x} = 0$$

$$\mathbf{A} * \mathbf{N} = \underline{0}$$

\mathbf{A} is a 5×6 matrix, and \mathbf{N} is the RNS of \mathbf{A} ($\mathbf{N} = \mathbf{RNS}(\mathbf{A})$), more precisely it is a vector with the coefficients that satisfies this equation:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

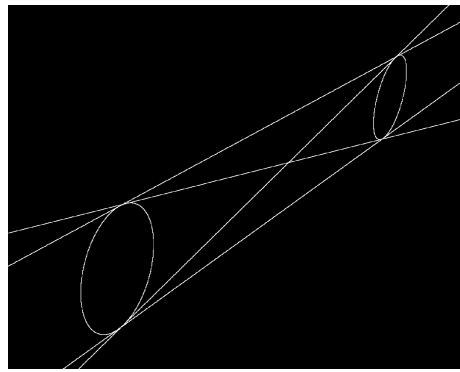
$$\mathbf{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$$

To give the possibility to visualize on the original image the conics just detected I implemented the function `fromConicToProfile()` that return a binary image with all zero except the points of the conic. Moreover `showProfile0pt()` return ad image as the original except in the point of the conic that are white. I optimized this function reducing its computational time of a tenth. Originally I use two *for loops* linked and spent 3/4 seconds, after changing approach I used matrix and logical operations that are more efficient and optimized.

Determine the ratio between diameter and wheel-to-wheel distance

Using the C matrices calculated in the previous point, I calculated the lines tangent both conics. To do this I used the dual conic, the set of line such that

$$\begin{aligned} & \underline{l} \cdot \mathbf{C}^* \cdot \underline{l} \\ & \left\{ \begin{array}{l} \underline{l} \cdot \mathbf{C}_1^* \cdot \underline{l} \\ \underline{l} \cdot \mathbf{C}_2^* \cdot \underline{l} \end{array} \right\} \implies 4 \text{ lines} \end{aligned}$$



After that I have to calculate the back transformation matrix necessary to calculate the ratio between diameter and distance wheel-to-wheel.

As seen during a lecture we are able to calculate the matrix with the image of two pair of lines and a conic.

First of all I calculated the pair of parallel lines `line1 & line2` and `line3 & line4`. Also from line we find vanishing points that represent the direction of parallel lines, respectively `vpoint1 & vpoint2`.

Secondly I found the line at infinity, that is the set of vanishing points.

Moreover, we studied that every conic intersect the line at infinity in two points `I, J`, called circular point, because every conic intersect the line at infinity in this two point.

$$I = \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} \quad J = \begin{bmatrix} 1 \\ -i \\ 0 \end{bmatrix}$$

So, I calculate the image of circular points:

$$\begin{cases} l_\infty \underline{x} = 0 \\ \underline{x}^T C \underline{x} = 0 \end{cases} \implies I', J'$$

The image of absolute conic at infinity is defines as:

$$C_\infty' = I' J'^T + I'^T J'$$

With the SVD decomposition we are able to have three matrices `U, S, V` such that:

$$svd(C_\infty') = U * S * V'$$

In addition we know that we can calculate the image of absolute conic at infinity with the similarity `Hr`. Knowing this, and the transformation rule of conic we can write

$$\begin{aligned} C_\infty^* &= Hr \cdot C_\infty'^* \cdot Hr^T \\ C_\infty'^* &= Hr^{-1} \cdot C_\infty^* \cdot Hr^{-T} \\ C_\infty^* &= \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \\ Hr^{-1} &= U \quad C_\infty^* = S \end{aligned}$$

In my case the matrix S isn't the matrix of absolute conic at infinity, so I decomposed it into three matrices

$$S = T C_{\infty}^* T$$
$$Hr = (UT)^{-1}e$$

Applying the back transformation to the intersection points $v1, v2$ (intersection between tangents and conics, found previously), we can calculate the diameter of wheels and the distance between wheel-to-wheel (from the center of the wheels).

```
>> ratio = diameter/distancebtow  
ratio =  
0.1813
```