Faculty of Engineering, Ain Shams University

Image processing, spring 2019

“Converting images to AutoCAD drawings”

Final Report

Student Names and IDs:

43721 -	احمد محمد احمد هيكل
43777 -	دعاء كامل عبدالمنعم على
43821 -	عمر اشرف صبحى محمد
43875 -	محمد محمود فرج احمد

Project Idea description:

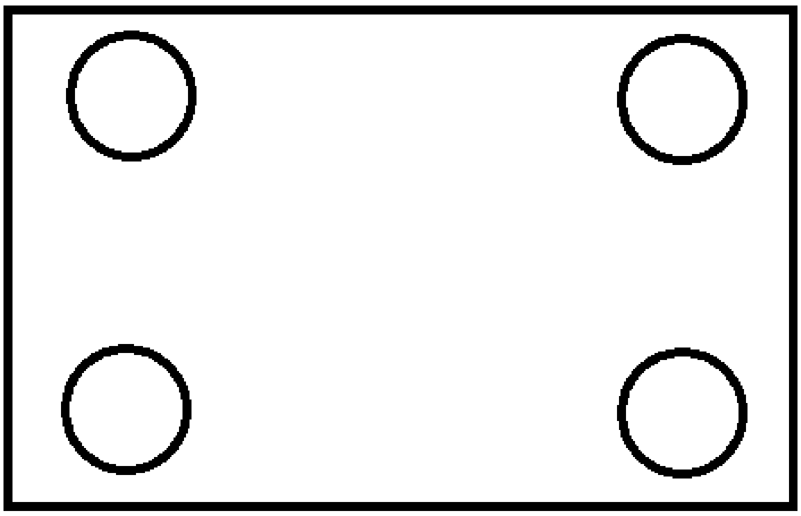
Desktop application for drawing shapes in AutoCAD.

The application takes image contains shapes like circles and rectangles as an input, process this image then outputs the same shapes into AutoCAD project.

The project is running through three main steps:-

1- Preprocessing and detection:

Our algorithm first reads image like this one



It contains a rectangle and four circles inside it,

We detect the rectangle by detecting every line of it by using openCV's function HoughLinesP and detect the four circles by openCV's function HoughCircles.

We also can detect half circles (arcs) as we will show examples of it in the video.

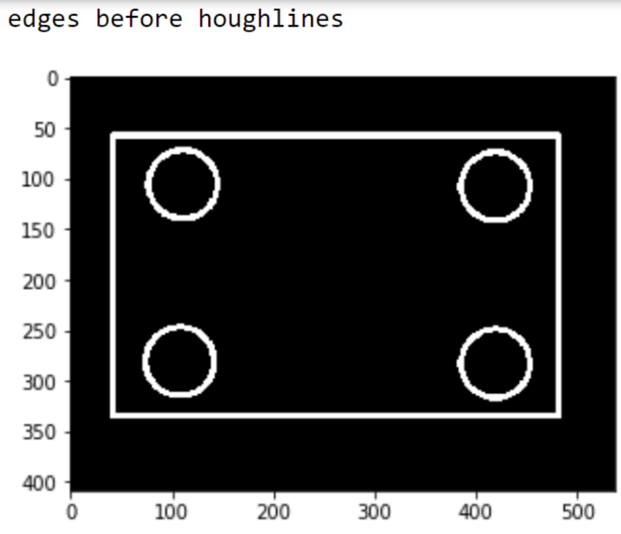
First we convert the original image to gray scale using this opencv’s function

```
gray = cv2.cvtColor(test, cv2.COLOR_BGR2GRAY)
```

Then we move to the lines detection step, to prepare the image to HoughLinesP function we apply canny edge detection to it and convert it to binary image, we also find out that using closing technique will make results better.

```
    edges = cv2.Canny(gray, 50, 100)                         # get the edges

    #convert to binary image
    ret,edges = cv2.threshold(edges,40,255,cv2.THRESH_BINARY)



    kernel=cv2.getStructuringElement(cv2.MORPH_RECT,(7,7))
    edges = cv2.morphologyEx(edges,cv2.MORPH_CLOSE,kernel)
HoughLinesP
    plt.imshow(edges, cmap='gray')
    plt.show()
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, 80, minLineLength=0, maxLineGap=0)
```

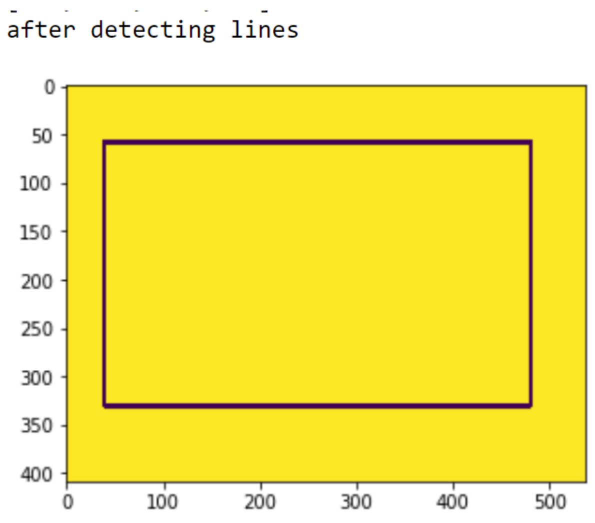
The image after preparing it to enter HoughLinesP function:



Unfortunately; HoughLinesP function outputs many lines that are similar to each other for every single line. So we wrote some lines of code to loop over these lines and come out with only a unique line for every single line in the image. We discarded many lines that seems to be redundant.

Every line is represented by a start point and an end point (x and y coordinates for each of them)

This is image of drawing lines after filtering on a blank image:

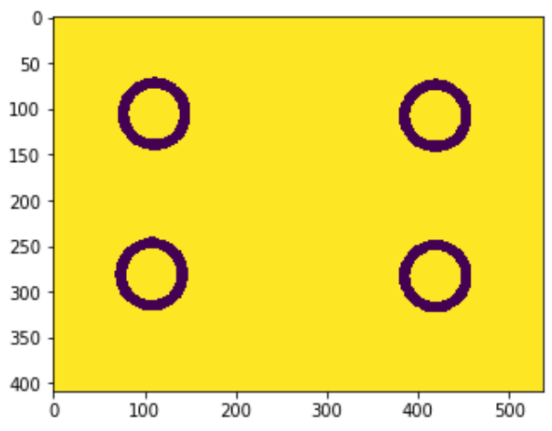


Then we loop over all lines and divide them into Vertical lines, Horizontal lines and general lines and put every genre in a list to prepare them to be drawn on AutoCAD later.

After that we deal with circles and arcs, we prepare the image to enter the HoughCircles function by deleting lines from the image and leaving only cirlces and arcs, we do that by subtracting image with lines only from the original image.

This will be the output of subtracting and it will be ready for HoughCircles function:

```
output circles only to be detected
```



After extracting the circles we put it in a list to be ready to be drawn on Autocad.

Each circle is represented with a center, and a radius.

Before using dxfwrite library; we loop over circles list to divide them into five classes by appending an extra element for each element in list_circles list which determines whether it is full circles, upper half circle, lower half circle, Right half circle or left half circle.

We make our reference of drawing (origin) x=0, y=0 at the bottom left point of the drawing.

2- Linking with AutoCAD

Now we will illustrate how the dxfwrite library work.

```
from operator import itemgetter
from dxfwrite import DXFEngine as dxf
from dxfwrite.dimlines import dimstyles, LinearDimension ,RadialDimension
```

Class shapes:

This class links python with AutoCAD. And generating dxf file.

It contains many functions to draw lines, circles and arcs without dimensions and also with appropriate dimensions.

It supports also writing text if you want on your drawing.

And here we will discuss these functions with details:

This function is used to write any text at any position and also you can choose one of the seven color.

```
def text(self,text,x,y,color = Foshia):
    text = dxf.text(text, (x, y), height=0.3, rotation=0)
    text['layer'] = 'TEXT'
    text['color'] = color
    self.drawing.add(text)
    self.drawing.save()
```

This function uses to draw a line with any slope with no dimension

```
def line_with_NO_dims(self,x1, y1, x2, y2,color = White):
    line = dxf.line((x1, y1), (x2, y2))
    line['color'] = color
    self.drawing.add(line)
    self.drawing.save()
```

We use this to draw horizontal line with under dimension

```
def Horizontal_line_with_under_dims(self,x1, y1, x2, y2,length,color = White):
```

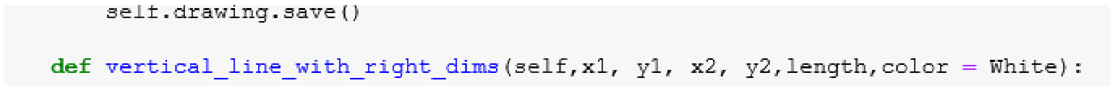
We use this to draw horizontal line with upper dimension

```
def Horizontal_line_with_upper_dims(self,x1, y1, x2, y2,length,color = White):
```

And this function for vertical lines only with left dimension

```
def vertical_line_with_left_dims(self,x1, y1, x2, y2,length,color = White):
```

This is the same, but with Right dimension



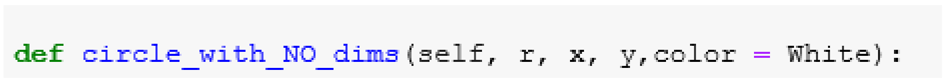
```
        self.drawing.save()

    def vertical_line_with_right_dims(self,x1, y1, x2, y2,length,color = White):
```

This function is used to draw a circle without dimensions

r = radius of the circle

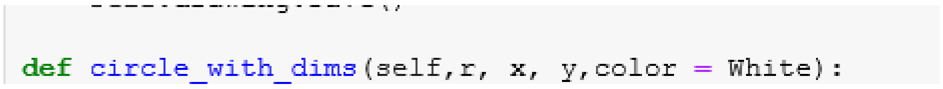
(x,y) = coordinates of the center

Color = used to determine color of circle



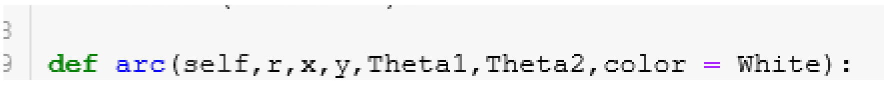
```
def circle_with_NO_dims(self, r, x, y,color = White):
```

The same as before but with dimensions



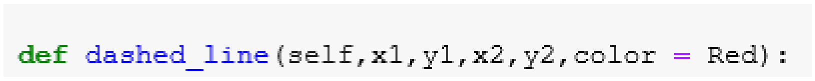
```
def circle_with_dims(self,r, x, y,color = White):
```

This used to draw an arc with no dimensions



```
def arc(self,r,x,y,Theta1,Theta2,color = White):
```

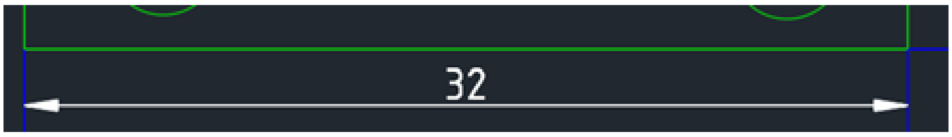
This used to draw dashed line



```
def dashed_line(self,x1,y1,x2,y2,color = Red):
```

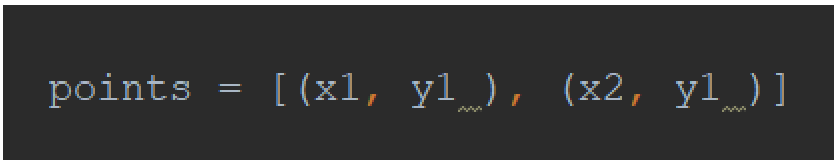
Dimensions:

A bit more details about how to control the Dimensions:



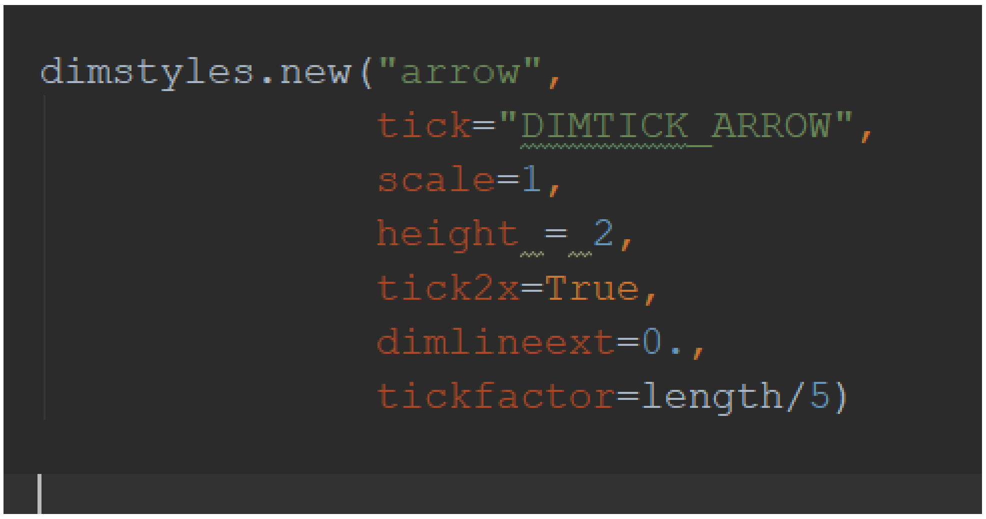
First we use dxfwrite.dimlines to draw Dimensions

1) Linear Dimensions :



```
points = [(x1, y1 ), (x2, y1 )]
```

we determine the x-position of the Dim line above. And the y-position determine the height of blue line.



```
dimstyles.new("arrow",
              tick="DIMTICK_ARROW",
              scale=1,
              height = 2,
              tick2x=True,
              dimlineext=0.,
              tickfactor=length/5)
```

These line of code create new dim line and determine some properties:
1st and 2nd parameter: choose the dim style.

3rd parameter: choose any scale you want (m , cm , mm , …)

4th parameter: determine the font of dimension written above the line.

5th parameter: it says true to be two arrows in the dimension.

6th parameter: determine the extension of line out of arrows.

7th parameter: determine the size of arrow itself.

```
self.drawing.add(
    LinearDimension((10, y1 - length/10), points, dimstyle='arrow', angle=Theta))
```

This line add this linear dim to AutoCAD file.

1st parameter: determines y-position of the dim line.

2nd parameter: is the points above.

3rd parameter: is the style of dim line (here it's arrow).

4th parameter: is the angle of rotation of this dim line.

We create under and upper dimensions for horizontal lines, also we create right and left dimensions for vertical lines.

We choose the appropriate parameters for each one.

2) Radial Dimensions

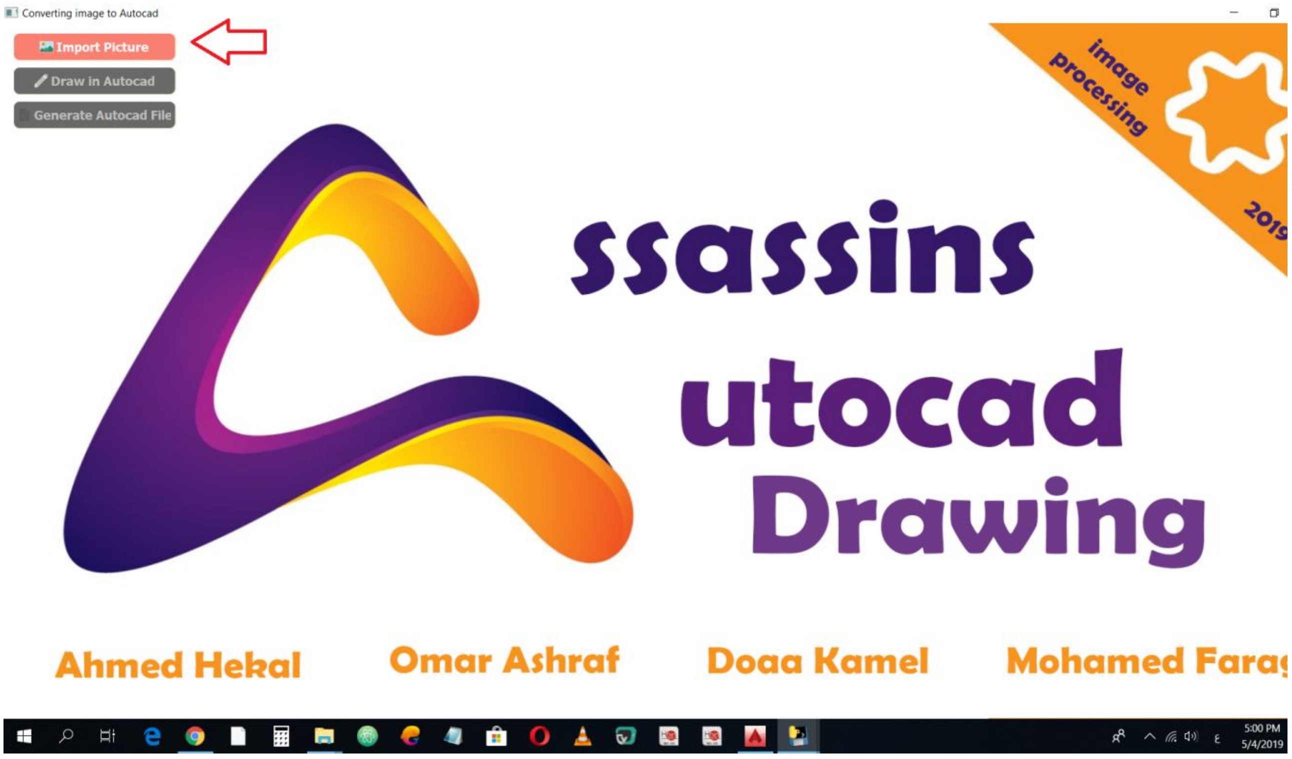
We use radial dimensions with circles

```
dimstyles.new("radius",scale=1, height=r/4, prefix='R=',tickfactor=r/1.5)
```

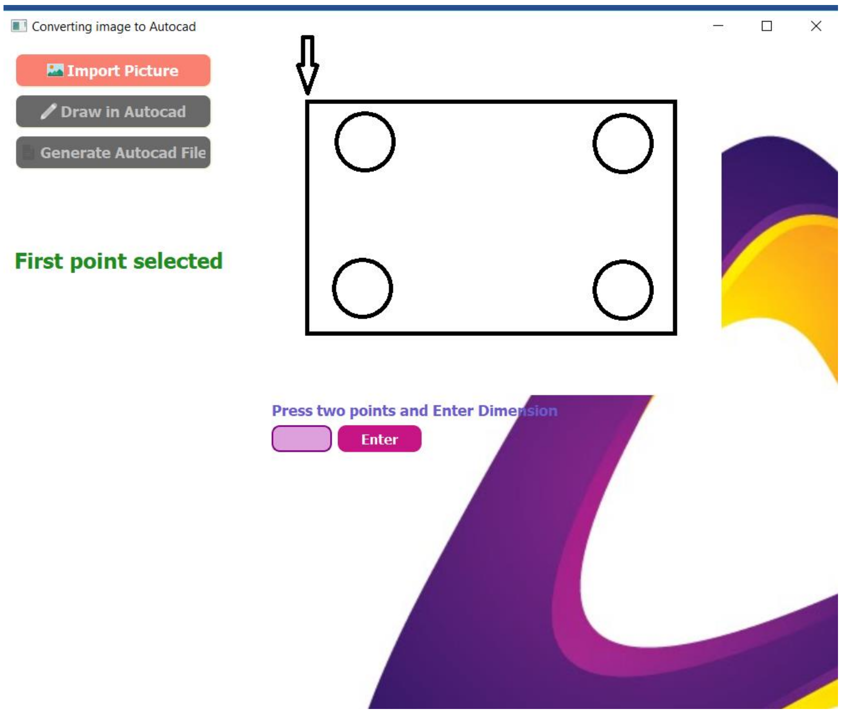
And we use the same parameters except for prefix which used to write "R=" before the radius.

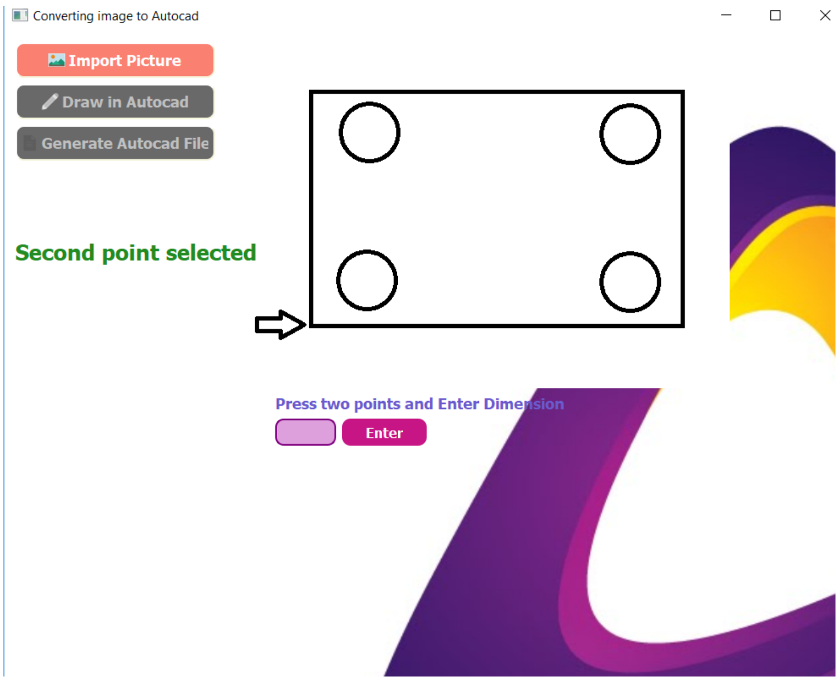
3- Merging the code with user interface (GUI):

First user will import the image by clicking on “import picture” button and choose an image



Then he should click on two points of any line like this:



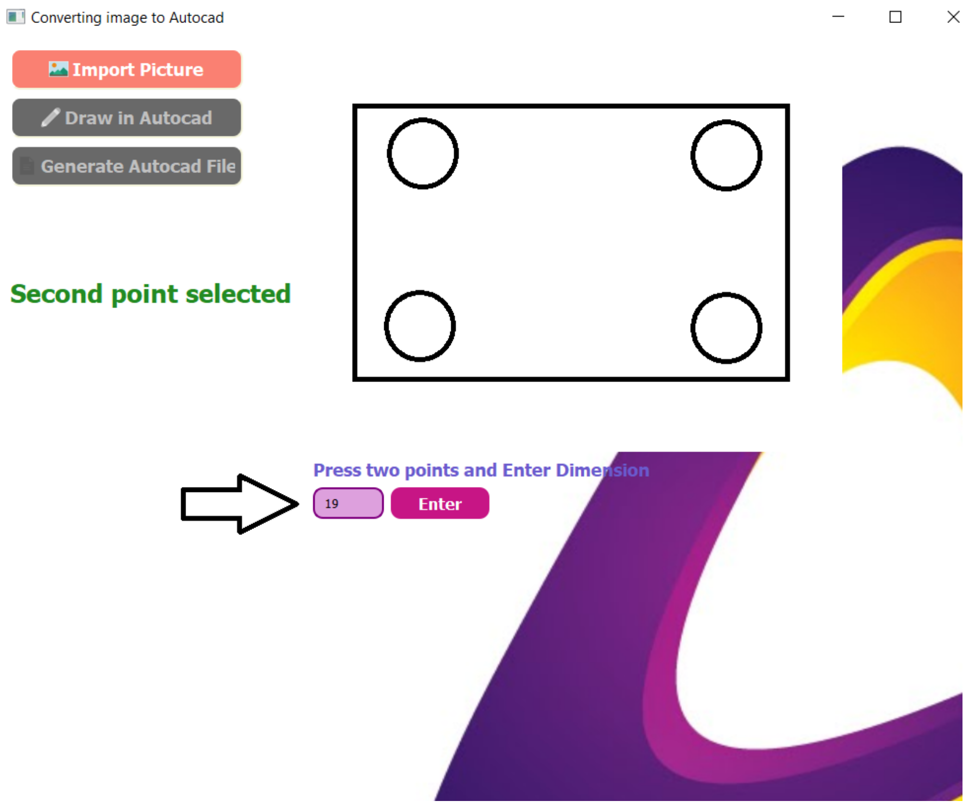


Second point selected

Press two points and Enter Dimension

Then user should enter the length of that line with cm

Then press “enter” button



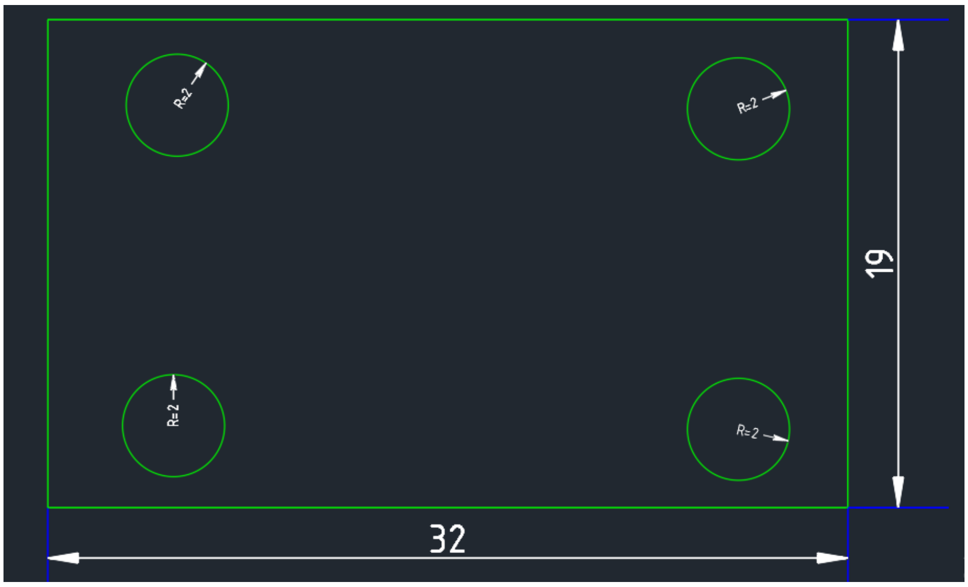
Second point selected

Press two points and Enter Dimension

19

Finally if the user has AutoCAD in his computer he can press “Draw in AutoCAD” to open an AutoCAD project and draw on it, otherwise he can press “generate AutoCAD file” to only generate dxf file containing the drawing.

The output in AutoCAD :



Thank you !