In [ ]:

```
import numpy as np
import cv2
import  imutils

# Read the image file
image = cv2.imread('bmw-car-front-png-30.png')

# Resize the image - change width to 500
image = imutils.resize(image, width=500)

# Display the original image
cv2.imshow("Original Image", image)

# RGB to Gray scale conversion
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("1 - Grayscale Conversion", gray)

# Noise removal with iterative bilateral filter(removes noise while preserving edges)
gray = cv2.bilateralFilter(gray, 11, 17, 17)
cv2.imshow("2 - Bilateral Filter", gray)

# Find Edges of the grayscale image
edged = cv2.Canny(gray, 170, 200)
cv2.imshow("4 - Canny Edges", edged)

# Find contours based on Edges
(new, cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cnts=sorted(cnts, key = cv2.contourArea, reverse = True)[:30] #sort contours based on th
eir area keeping minimum required area as '30' (anything smaller than this will not be co
nsidered)
NumberPlateCnt = None #we currently have no Number plate contour

# loop over our contours to find the best possible approximate contour of number plate
count = 0
for c in cnts:
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.02 * peri, True)
        if len(approx) == 4:  # Select the contour with 4 corners
            NumberPlateCnt = approx #This is our approx Number Plate Contour
            break


# Drawing the selected contour on the original image
cv2.drawContours(image, [NumberPlateCnt], -1, (0,255,0), 3)
cv2.imshow("Final Image With Number Plate Detected", image)

cv2.waitKey(0) #Wait for user input before closing the images displayed
```

In [ ]:

```
```