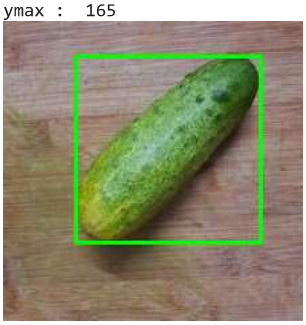
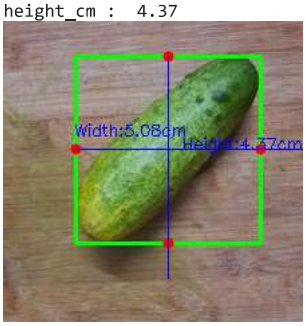
```
import xml.etree.ElementTree as ET
import cv2
import math
from google.colab.patches import cv2_imshow

lower_color = (0, 100, 100)
upper_color = (10, 255, 255)
img = cv2.imread("/content/0017.jpg")

# Convert the image to the HSV color space
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Threshold the image to get the binary mask
mask = cv2.inRange(hsv_img, lower_color, upper_color)

# path to the XML file
xml_file = "/content/0017.xml"

# parse the XML file
tree = ET.parse(xml_file)
root = tree.getroot()

# iterate through all object elements in the XML file
for obj in root.findall('object'):
    # get the bounding box coordinates
    xmin = int(obj.find('bndbox/xmin').text)
    ymin = int(obj.find('bndbox/ymin').text)
    xmax = int(obj.find('bndbox/xmax').text)
    ymax = int(obj.find('bndbox/ymax').text)
    print("xmin : ",xmin,"\nymin : ",ymin,"\nxmax : ",xmax,"\nymax : ",ymax)
    cv2.rectangle(img, (xmin, ymin), (xmax,ymax), (0, 255, 0), 2)

    # Display the image with bounding box
    cv2_imshow(img)

center_x = (xmin + xmax) // 2
center_y = (ymin + ymax) // 2
print("center_x : ",center_x,"\ncenter_y : ",center_y)

# Mark the particular pixel points used for calculating the height and width

#cv2.circle(img, (xmin,center_x), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (xmax, center_y), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (center_x,ymin), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (xmin, center_y), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (center_x,ymax), radius=4, color=(0, 0, 255), thickness=-1)
#cv2.circle(img, ((xmin + xmax), center_y), radius=4, color=(0, 0, 255), thickness=-1)

# Print the pixel coordinates used for calculating the width and height
print("Coordinates used for width calculation: ({}, {}), ({}, {})".format(xmin, center_y, (xmin + xmax), center_y))
print("Coordinates used for height calculation: ({}, {}), ({}, {})".format(center_x, ymin, center_x, (ymin + ymax)))

# Calculate the width and height using Euclidean distance
width = round(math.sqrt(((xmin+xmax) - xmin)**2 + (center_y - center_y)**2), 2)
height = round(math.sqrt((center_x - center_x)**2 + ((ymin+ymax) - ymin)**2), 2)
print("width : ",width,"\nheight : ",height)

# Draw lines to mark the width and height distance

cv2.line(img, (xmin, center_y), (xmin + xmax, center_y), (255, 0, 0), 1)
cv2.line(img, (center_x, ymin), (center_x, ymin + ymax), (255, 0, 0), 1)


num_pixels = cv2.countNonZero(mask)

# Calculate the pixels per millimeter (PPM) value
#1 pixel (X)  0.0264583333 cm
#1 pixel (X)  0.2645833333 mm

# Convert pixel measurements to millimeters
width_cm = round((width*0.0264583333),2)
print("width_cm : ",width_cm)

height_cm = round(height*0.0264583333, 2)
print("height_cm : ",height_cm)
```

```
# Annotate the width and height measures on the image


cv2.putText(img, "Width:{}cm".format(width_cm), (center_x - 70, center_y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 0, 0), 1)
cv2.putText(img, "Height:{}cm".format(height_cm), (center_x + 10, center_y), cv2.FONT_HERSHEY_SIMPLEX,  0.4, (255, 0, 0), 1)



# Display the image
cv2_imshow(img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
xmin :  54
ymin :  26
xmax :  192
ymax :  165
```



```
center_x :  123
center_y :  95
Coordinates used for width calculation: (54, 95), (246, 95)
Coordinates used for height calculation: (123, 26), (123, 191)
width :  192.0
height :  165.0
width_cm :  5.08
height_cm :  4.37
```




```
import xml.etree.ElementTree as ET
import cv2
import math
from google.colab.patches import cv2_imshow

lower_color = (0, 100, 100)
upper_color = (10, 255, 255)
img = cv2.imread("/content/0068.jpg")

# Convert the image to the HSV color space
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Threshold the image to get the binary mask
mask = cv2.inRange(hsv_img, lower_color, upper_color)

# path to the XML file
xml_file = "/content/0068.xml"

# parse the XML file
tree = ET.parse(xml_file)
root = tree.getroot()

# iterate through all object elements in the XML file
for obj in root.findall('object'):
```

```
        # get the bounding box coordinates
        xmin = int(obj.find('bndbox/xmin').text)
        ymin = int(obj.find('bndbox/ymin').text)
        xmax = int(obj.find('bndbox/xmax').text)
        ymax = int(obj.find('bndbox/ymax').text)
        print("xmin : ",xmin,"\nymin : ",ymin,"\nxmax : ",xmax,"\nymax : ",ymax)
        cv2.rectangle(img, (xmin, ymin), (xmax,ymax), (0, 255, 0), 2)

        # Display the image with bounding box
        cv2_imshow(img)

    center_x = (xmin + xmax) // 2
    center_y = (ymin + ymax) // 2
    print("center_x : ",center_x,"\ncenter_y : ",center_y)

    # Mark the particular pixel points used for calculating the height and width

    #cv2.circle(img, (xmin,center_x), radius=4, color=(0, 0, 255), thickness=-1)
    cv2.circle(img, (xmax, center_y), radius=4, color=(0, 0, 255), thickness=-1)
    cv2.circle(img, (center_x,ymin), radius=4, color=(0, 0, 255), thickness=-1)
    cv2.circle(img, (xmin, center_y), radius=4, color=(0, 0, 255), thickness=-1)
    cv2.circle(img, (center_x,ymax), radius=4, color=(0, 0, 255), thickness=-1)
    #cv2.circle(img, ((xmin + xmax), center_y), radius=4, color=(0, 0, 255), thickness=-1)

    # Print the pixel coordinates used for calculating the width and height
    print("Coordinates used for width calculation: ({}, {}), ({}, {})".format(xmin, center_y, (xmin + xmax), center_y))
    print("Coordinates used for height calculation: ({}, {}), ({}, {})".format(center_x, ymin, center_x, (ymin + ymax)))

    # Calculate the width and height using Euclidean distance
    width = round(math.sqrt(((xmin+xmax) - xmin)**2 + (center_y - center_y)**2), 2)
    height = round(math.sqrt((center_x - center_x)**2 + ((ymin+ymax) - ymin)**2), 2)
    print("width : ",width,"\nheight : ",height)

    # Draw lines to mark the width and height distance

    cv2.line(img, (xmin, center_y), (xmin + xmax, center_y), (255, 0, 0), 1)
    cv2.line(img, (center_x, ymin), (center_x, ymin + ymax), (255, 0, 0), 1)


    num_pixels = cv2.countNonZero(mask)

    # Calculate the pixels per millimeter (PPM) value
    #1 pixel (X)    0.0264583333 cm
    #1 pixel (X)    0.2645833333 mm

    # Convert pixel measurements to millimeters
    width_cm = round((width*0.0264583333),2)
    print("width_cm : ",width_cm)

    height_cm = round(height*0.0264583333, 2)
    print("height_cm : ",height_cm)


    # Annotate the width and height measures on the image


    cv2.putText(img, "Width:{}cm".format(width_cm), (center_x - 70, center_y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 0, 0), 1)
    cv2.putText(img, "Height:{}cm".format(height_cm), (center_x + 10, center_y), cv2.FONT_HERSHEY_SIMPLEX,  0.4, (255, 0, 0), 1)



    # Display the image
    cv2_imshow(img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

```
xmin :  24
ymin :  18
xmax :  189
ymax :  206
```

```
center_x :  106
center_y :  112
Coordinates used for width calculation: (24, 112), (213, 112)
Coordinates used for height calculation: (106, 18), (106, 224)
width :  189.0
height :  206.0
width_cm :  5.0
height_cm :  5.45
```

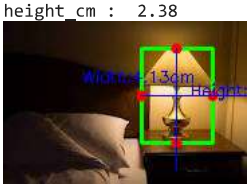


```
import xml.etree.ElementTree as ET
import cv2
import math
from google.colab.patches import cv2_imshow

lower_color = (0, 100, 100)
upper_color = (10, 255, 255)
img = cv2.imread("/content/5.jpg")

# Convert the image to the HSV color space
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Threshold the image to get the binary mask
mask = cv2.inRange(hsv_img, lower_color, upper_color)

# path to the XML file
xml_file = "/content/5.xml"

# parse the XML file
tree = ET.parse(xml_file)
root = tree.getroot()

# iterate through all object elements in the XML file
for obj in root.findall('object'):
    # get the bounding box coordinates
    xmin = int(obj.find('bndbox/xmin').text)
    ymin = int(obj.find('bndbox/ymin').text)
    xmax = int(obj.find('bndbox/xmax').text)
    ymax = int(obj.find('bndbox/ymax').text)
    print("xmin : ",xmin,"\nymin : ",ymin,"\nxmax : ",xmax,"\nymax : ",ymax)
    cv2.rectangle(img, (xmin, ymin), (xmax,ymax), (0, 255, 0), 2)

    # Display the image with bounding box
    cv2_imshow(img)

center_x = (xmin + xmax) // 2
center_y = (ymin + ymax) // 2
print("center_x : ",center_x,"\ncenter_y : ",center_y)

# Mark the particular pixel points used for calculating the height and width

#cv2.circle(img, (xmin,center_x), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (xmax, center_y), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (center_x,ymin), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (xmin, center_y), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (center_x,ymax), radius=4, color=(0, 0, 255), thickness=-1)
#cv2.circle(img, ((xmin + xmax), center_y), radius=4, color=(0, 0, 255), thickness=-1)

# Print the pixel coordinates used for calculating the width and height
print("Coordinates used for width calculation: ({}, {}), ({}, {})".format(xmin, center_y, (xmin + xmax), center_y))
print("Coordinates used for height calculation: ({}, {}), ({}, {})".format(center_x, ymin, center_x, (ymin + ymax)))
```

```
# Calculate the width and height using Euclidean distance
width = round(math.sqrt(((xmin+xmax) - xmin)**2 + (center_y - center_y)**2), 2)
height = round(math.sqrt((center_x - center_x)**2 + ((ymin+ymax) - ymin)**2), 2)
print("width : ",width,"\nheight : ",height)

# Draw lines to mark the width and height distance

cv2.line(img, (xmin, center_y), (xmin + xmax, center_y), (255, 0, 0), 1)
cv2.line(img, (center_x, ymin), (center_x, ymin + ymax), (255, 0, 0), 1)


num_pixels = cv2.countNonZero(mask)

# Calculate the pixels per millimeter (PPM) value
#1 pixel (X)    0.0264583333 cm
#1 pixel (X)    0.2645833333 mm

# Convert pixel measurements to millimeters
width_cm = round((width*0.0264583333),2)
print("width_cm : ",width_cm)

height_cm = round(height*0.0264583333, 2)
print("height_cm : ",height_cm)


# Annotate the width and height measures on the image


cv2.putText(img, "Width:{}cm".format(width_cm), (center_x - 70, center_y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 0, 0), 1)
cv2.putText(img, "Height:{}cm".format(height_cm), (center_x + 10, center_y), cv2.FONT_HERSHEY_SIMPLEX,  0.4, (255, 0, 0), 1)



# Display the image
cv2_imshow(img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
xmin :  103
ymin :  20
xmax :  156
ymax :  90
```

```
center_x :  129
center_y :  55
Coordinates used for width calculation: (103, 55), (259, 55)
Coordinates used for height calculation: (129, 20), (129, 110)
width :  156.0
height :  90.0
width_cm :  4.13
height_cm :  2.38
```



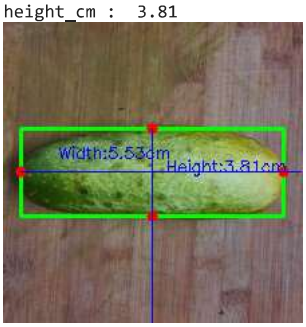

```
import xml.etree.ElementTree as ET
import cv2
import math
from google.colab.patches import cv2_imshow

lower_color = (0, 100, 100)
upper_color = (10, 255, 255)
img = cv2.imread("/content/0214.jpg")

# Convert the image to the HSV color space
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

```
# Threshold the image to get the binary mask
mask = cv2.inRange(hsv_img, lower_color, upper_color)

# path to the XML file
xml_file = "/content/0214.xml"

# parse the XML file
tree = ET.parse(xml_file)
root = tree.getroot()

# iterate through all object elements in the XML file
for obj in root.findall('object'):
    # get the bounding box coordinates
    xmin = int(obj.find('bndbox/xmin').text)
    ymin = int(obj.find('bndbox/ymin').text)
    xmax = int(obj.find('bndbox/xmax').text)
    ymax = int(obj.find('bndbox/ymax').text)
    print("xmin : ",xmin,"\nymin : ",ymin,"\nxmax : ",xmax,"\nymax : ",ymax)
    cv2.rectangle(img, (xmin, ymin), (xmax,ymax), (0, 255, 0), 2)

    # Display the image with bounding box
    cv2_imshow(img)

center_x = (xmin + xmax) // 2
center_y = (ymin + ymax) // 2
print("center_x : ",center_x,"\ncenter_y : ",center_y)

# Mark the particular pixel points used for calculating the height and width

#cv2.circle(img, (xmin,center_x), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (xmax, center_y), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (center_x,ymin), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (xmin, center_y), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (center_x,ymax), radius=4, color=(0, 0, 255), thickness=-1)
#cv2.circle(img, ((xmin + xmax), center_y), radius=4, color=(0, 0, 255), thickness=-1)

# Print the pixel coordinates used for calculating the width and height
print("Coordinates used for width calculation: ({}, {}), ({}, {})".format(xmin, center_y, (xmin + xmax), center_y))
print("Coordinates used for height calculation: ({}, {}), ({}, {})".format(center_x, ymin, center_x, (ymin + ymax)))

# Calculate the width and height using Euclidean distance
width = round(math.sqrt(((xmin+xmax) - xmin)**2 + (center_y - center_y)**2), 2)
height = round(math.sqrt((center_x - center_x)**2 + ((ymin+ymax) - ymin)**2), 2)
print("width : ",width,"\nheight : ",height)

# Draw lines to mark the width and height distance

cv2.line(img, (xmin, center_y), (xmin + xmax, center_y), (255, 0, 0), 1)
cv2.line(img, (center_x, ymin), (center_x, ymin + ymax), (255, 0, 0), 1)


num_pixels = cv2.countNonZero(mask)

# Calculate the pixels per millimeter (PPM) value
#1 pixel (X)    0.0264583333 cm
#1 pixel (X)    0.2645833333 mm

# Convert pixel measurements to millimeters
width_cm = round((width*0.0264583333),2)
print("width_cm : ",width_cm)

height_cm = round(height*0.0264583333, 2)
print("height_cm : ",height_cm)


# Annotate the width and height measures on the image


cv2.putText(img, "Width:{}cm".format(width_cm), (center_x - 70, center_y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 0, 0), 1)
cv2.putText(img, "Height:{}cm".format(height_cm), (center_x + 10, center_y), cv2.FONT_HERSHEY_SIMPLEX,  0.4, (255, 0, 0), 1)



# Display the image
cv2_imshow(img)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
xmin :  13
ymin :  79
xmax :  209
ymax :  144
```

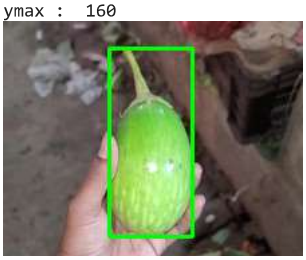
```
center_x :  111
center_y :  111
Coordinates used for width calculation: (13, 111), (222, 111)
Coordinates used for height calculation: (111, 79), (111, 223)
width :  209.0
height :  144.0
width_cm :  5.53
height_cm :  3.81
```




```
import xml.etree.ElementTree as ET
import cv2
import math
from google.colab.patches import cv2_imshow

lower_color = (0, 100, 100)
upper_color = (10, 255, 255)
img = cv2.imread("/content/0131.jpg")

# Convert the image to the HSV color space
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

# Threshold the image to get the binary mask
mask = cv2.inRange(hsv_img, lower_color, upper_color)

# path to the XML file
xml_file = "/content/0131.xml"

# parse the XML file
tree = ET.parse(xml_file)
root = tree.getroot()

# iterate through all object elements in the XML file
for obj in root.findall('object'):
    # get the bounding box coordinates
    xmin = int(obj.find('bndbox/xmin').text)
    ymin = int(obj.find('bndbox/ymin').text)
    xmax = int(obj.find('bndbox/xmax').text)
    ymax = int(obj.find('bndbox/ymax').text)
    print("xmin : ",xmin,"\nymin : ",ymin,"\nxmax : ",xmax,"\nymax : ",ymax)
    cv2.rectangle(img, (xmin, ymin), (xmax,ymax), (0, 255, 0), 2)

    # Display the image with bounding box
    cv2_imshow(img)
```

```
center_x = (xmin + xmax) // 2
center_y = (ymin + ymax) // 2
print("center_x : ",center_x,"\ncenter_y : ",center_y)

# Mark the particular pixel points used for calculating the height and width

#cv2.circle(img, (xmin,center_x), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (xmax, center_y), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (center_x,ymin), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (xmin, center_y), radius=4, color=(0, 0, 255), thickness=-1)
cv2.circle(img, (center_x,ymax), radius=4, color=(0, 0, 255), thickness=-1)
#cv2.circle(img, ((xmin + xmax), center_y), radius=4, color=(0, 0, 255), thickness=-1)

# Print the pixel coordinates used for calculating the width and height
print("Coordinates used for width calculation: ({}, {}), ({}, {})".format(xmin, center_y, (xmin + xmax), center_y))
print("Coordinates used for height calculation: ({}, {}), ({}, {})".format(center_x, ymin, center_x, (ymin + ymax)))

# Calculate the width and height using Euclidean distance
width = round(math.sqrt(((xmin+xmax) - xmin)**2 + (center_y - center_y)**2), 2)
height = round(math.sqrt((center_x - center_x)**2 + ((ymin+ymax) - ymin)**2), 2)
print("width : ",width,"\nheight : ",height)

# Draw lines to mark the width and height distance

cv2.line(img, (xmin, center_y), (xmin + xmax, center_y), (255, 0, 0), 1)
cv2.line(img, (center_x, ymin), (center_x, ymin + ymax), (255, 0, 0), 1)


num_pixels = cv2.countNonZero(mask)

# Calculate the pixels per millimeter (PPM) value
#1 pixel (X)    0.0264583333 cm
#1 pixel (X)    0.2645833333 mm

# Convert pixel measurements to millimeters
width_cm = round((width*0.0264583333),2)
print("width_cm : ",width_cm)

height_cm = round(height*0.0264583333, 2)
print("height_cm : ",height_cm)


# Annotate the width and height measures on the image


cv2.putText(img, "Width:{}cm".format(width_cm), (center_x - 70, center_y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.4, (255, 0, 0), 1)
cv2.putText(img, "Height:{}cm".format(height_cm), (center_x + 10, center_y), cv2.FONT_HERSHEY_SIMPLEX,  0.4, (255, 0, 0), 1)



# Display the image
cv2_imshow(img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
xmin :  79
ymin :  20
xmax :  141
ymax :  160
```




```
center_x :  110
center_y :  90
Coordinates used for width calculation: (79, 90), (220, 90)
Coordinates used for height calculation: (110, 20), (110, 180)
width :  141.0
height :  160.0
width_cm :  3.73
height_cm :  4.23
```