```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
import zipfile

# Path to your dataset zip file in Google Drive
zip_file_path = '/content/drive/MyDrive/honeybeeSeg/HoneyBee.v1i.yolov8.zip'  # Change this path

# Directory where you want to extract the dataset
extract_dir = '/content'

# Unzipping the dataset
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

# Check if files are extracted
print("Files extracted to:", os.listdir(extract_dir))
```

```
Files extracted to: ['.config', 'train', 'drive', 'data.yaml', 'README.dataset.txt', 'README.roboflow.txt', 'valid', 'sample_data']
```

```
!pip install ultralytics
```

```
Show hidden output
```

```
data_yaml = """
train: /content/train  # Path to training images
val: /content/valid      # Path to validation images

nc: 2  # Number of classes
names: ['eggs', 'larves']  # List of class names
"""

# Writing to a YAML file
with open('/content/data.yaml', 'w') as file:
    file.write(data_yaml)


from ultralytics import YOLO

# Load the YOLOv8 segmentation model
model = YOLO('yolov8n-seg.pt')
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov8n-seg.pt to 'yolov8n-seg.pt'...
100%|██████████| 6.74M/6.74M [00:00<00:00, 114MB/s]
```

```
# Train the model
model.train(
    data='/content/data.yaml',
    epochs=50,
    imgsz=640,
    batch=16,
    name='yolov8_custom_seg'
)
```

```
0.99522,     0.99523,     0.99525,     0.99527,
            0.99528,     0.9953,     0.99531,     0.99533,     0.99534,     0.99536,     0.99537,     0.99539,      0.9954,
0.99542,     0.99543,     0.99545,     0.99547,     0.99548,      0.9955,     0.99551,     0.99553,     0.99554,     0.99556,
0.99557,     0.99559,      0.9956,     0.99562,
            0.99563,     0.99565,     0.99567,     0.99568,      0.9957,     0.99571,     0.99573,     0.99574,     0.99576,
0.99577,     0.99579,      0.9958,     0.99582,     0.99583,     0.99585,     0.99586,     0.99588,      0.9959,     0.99591,
0.99593,     0.99594,     0.99596,     0.99597,
            0.99599,      0.996,     0.99602,     0.99603,     0.99605,     0.99606,     0.99608,      0.9961,     0.99611,
0.99613,     0.99614,     0.99616,     0.99617,     0.99619,      0.9962,     0.99622,     0.99623,     0.99625,     0.99626,
0.99628,     0.99629,     0.99631,     0.99633,
            0.99634,     0.99636,     0.99637,     0.99639,      0.9964,     0.99642,     0.99643,     0.99645,     0.99646,
0.99648,     0.99649,     0.99651,     0.99653,     0.99654,     0.99656,     0.99657,     0.99659,      0.9966,     0.99662,
0.99663,     0.99665,     0.99666,     0.99668,
            0.99669,     0.99671,     0.99672,     0.99674,     0.99676,     0.99677,     0.99679,      0.9968,     0.99682,
0.99683,     0.99685,     0.99686,     0.99688,     0.99689,     0.99691,     0.99692,     0.99694,     0.99696,     0.99697,
0.99699,       0.997,     0.99702,     0.99703,
            0.99705,     0.99706,     0.99708,     0.99709,     0.99711,     0.99712,     0.99714,     0.99716,     0.99717,
0.99719,      0.9972,     0.99722,     0.99723,     0.99725,     0.99726,     0.99728,     0.99729,     0.99731,     0.99732,
0.99734,     0.99735,     0.99737,     0.99739,
            0.9974,     0.99742,     0.99743,     0.99745,     0.99746,     0.99748,     0.99749,     0.99751,     0.99752,
0.99754,     0.99755,     0.99757,     0.99759,      0.9976,     0.99762,     0.99763,     0.99765,     0.99766,     0.99768,
0.99769,     0.99771,     0.99772,     0.99774,
            0.99775,     0.99777,     0.99778,      0.9978,     0.99782,     0.99783,     0.99785,     0.99786,     0.99788,
0.99789,     0.99791,     0.99792,     0.99794,     0.99795,     0.99797,     0.99798,       0.998,     0.99802,     0.99803,
0.99805,     0.99806,     0.99808,     0.99809,
            0.99811,     0.99812,     0.99814,     0.99815,     0.99817,     0.99818,      0.9982,     0.99822,     0.99823,
0.99825,     0.99826,     0.99828,     0.99829,     0.99831,     0.99832,     0.99834,     0.99835,     0.99837,     0.99838,
0.9984,      0.99841,     0.99843,     0.99845,
            0.99846,     0.99848,     0.99849,     0.99851,     0.99852,     0.99854,     0.99855,     0.99857,     0.99858,
0.9986,      0.99861,     0.99863,     0.99865,     0.99866,     0.99868,     0.99869,     0.99871,     0.99872,     0.99874,
0.99875,     0.99877,     0.99878,      0.9988,
            0.99881,     0.99883,     0.99884,     0.99886,     0.99888,     0.99889,     0.99891,     0.99892,     0.99894,
0.99895,     0.99897,     0.99898,       0.999,     0.99901,     0.99903,     0.99904,     0.99906,     0.99908,     0.99909,
0.99911,     0.99912,     0.99914,     0.99915,
            0.99917,     0.99918,      0.9992,     0.99921,     0.99923,     0.99924,     0.99926,     0.99927,     0.99929,
0.99931,     0.99932,     0.99934,     0.99935,     0.99937,     0.99938,      0.9994,     0.99941,     0.99943,     0.99944,
0.99946,     0.99947,     0.99949,     0.99951,
            0.99952,     0.99954,     0.99955,     0.99957,     0.99958,      0.9996,     0.99961,     0.99963,     0.99964,
0.99966,     0.99967,     0.99969,     0.99971,     0.99972,     0.99974,     0.99975,     0.99977,     0.99978,      0.9998,
0.99981,     0.99983,     0.99984,     0.99986,
            0.99987,     0.99989,      0.9999,     0.99992,     0.99994,     0.99995,     0.99997,     0.99998,           1,
1,           1,           1,           1,           1,           1,           1,           1,           1,           1,
1,           1,           1,           1,
```

```
# Evaluate the trained model
metrics = model.val()
print(metrics)
```

```
Ultralytics YOLOv8.2.82 🚀 Python-3.10.12 torch-2.4.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8n-seg summary (fused): 195 layers, 3,258,454 parameters, 0 gradients, 12.0 GFLOPs
val: Scanning /content/valid/labels.cache... 74 images, 0 backgrounds, 0 corrupt: 100%|██████████| 74/74 [00:00<?, ?it/s]
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incompatible with multi
  self.pid = os.fork()
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50
                   all         74         74      0.999          1      0.995      0.952      0.999          1      0.995      0
                larves         74         74      0.999          1      0.995      0.952      0.999          1      0.995      0
Speed: 0.9ms preprocess, 19.6ms inference, 0.0ms loss, 6.8ms postprocess per image
Results saved to runs/segment/yolov8_custom_seg2
ultralytics.utils.metrics.SegmentMetrics object with attributes:

ap_class_index: array([1])
box: ultralytics.utils.metrics.Metric object
confusion_matrix: <ultralytics.utils.metrics.ConfusionMatrix object at 0x7fcb7c273e50>
curves: ['Precision-Recall(B)', 'F1-Confidence(B)', 'Precision-Confidence(B)', 'Recall-Confidence(B)', 'Precision-Recall(M)', 'F
curves_results: [[array([          0,    0.001001,    0.002002,    0.003003,    0.004004,    0.005005,    0.006006,    0.007007,
         0.024024,    0.025025,    0.026026,    0.027027,    0.028028,    0.029029,     0.03003,    0.031031,    0.032032,    0
         0.048048,    0.049049,     0.05005,    0.051051,    0.052052,    0.053053,    0.054054,    0.055055,    0.056056,    0
         0.072072,    0.073073,    0.074074,    0.075075,    0.076076,    0.077077,    0.078078,    0.079079,     0.08008,    0
         0.096096,    0.097097,    0.098098,    0.099099,      0.1001,      0.1011,      0.1021,      0.1031,      0.1041,
          0.12012,     0.12112,     0.12212,     0.12312,     0.12412,     0.12513,     0.12613,     0.12713,     0.12813,
          0.14414,     0.14515,     0.14615,     0.14715,     0.14815,     0.14915,     0.15015,     0.15115,     0.15215,
          0.16817,     0.16917,     0.17017,     0.17117,     0.17217,     0.17317,     0.17417,     0.17518,     0.17618,
          0.19219,     0.19319,     0.19419,      0.1952,      0.1962,      0.1972,      0.1982,      0.1992,      0.2002,
          0.21622,     0.21722,     0.21822,     0.21922,     0.22022,     0.22122,     0.22222,     0.22322,     0.22422,
          0.24024,     0.24124,     0.24224,     0.24324,     0.24424,     0.24525,     0.24625,     0.24725,     0.24825,
          0.26426,     0.26527,     0.26627,     0.26727,     0.26827,     0.26927,     0.27027,     0.27127,     0.27227,
          0.28829,     0.28929,     0.29029,     0.29129,     0.29229,     0.29329,     0.29429,      0.2953,      0.2963,
          0.31231,     0.31331,     0.31431,     0.31532,     0.31632,     0.31732,     0.31832,     0.31932,     0.32032,
          0.33634,     0.33734,     0.33834,     0.33934,     0.34034,     0.34134,     0.34234,     0.34334,     0.34434,
          0.36036,     0.36136,     0.36236,     0.36336,     0.36436,     0.36537,     0.36637,     0.36737,     0.36837,
          0.38438,     0.38539,     0.38639,     0.38739,     0.38839,     0.38939,     0.39039,     0.39139,     0.39239,
          0.40841,     0.40941,     0.41041,     0.41141,     0.41241,     0.41341,     0.41441,     0.41542,     0.41642,
          0.43243,     0.43343,     0.43443,     0.43544,     0.43644,     0.43744,     0.43844,     0.43944,     0.44044,
          0.45646,     0.45746,     0.45846,     0.45946,     0.46046,     0.46146,     0.46246,     0.46346,     0.46446,
          0.48048,     0.48148,     0.48248,     0.48348,     0.48448,     0.48549,     0.48649,     0.48749,     0.48849,
           0.5045,     0.50551,     0.50651,     0.50751,     0.50851,     0.50951,     0.51051,     0.51151,     0.51251,
          0.52853,     0.52953,     0.53053,     0.53153,     0.53253,     0.53353,     0.53453,     0.53554,     0.53654,
```

```
0.55255,     0.55355,     0.55455,     0.55556,     0.55656,     0.55756,     0.55856,     0.55956,     0.56056,
0.57658,     0.57758,     0.57858,     0.57958,     0.58058,     0.58158,     0.58258,     0.58358,     0.58458,
 0.6006,      0.6016,      0.6026,      0.6036,      0.6046,     0.60561,     0.60661,     0.60761,     0.60861,
0.62462,     0.62563,     0.62663,     0.62763,     0.62863,     0.62963,     0.63063,     0.63163,     0.63263,
0.64865,     0.64965,     0.65065,     0.65165,     0.65265,     0.65365,     0.65465,     0.65566,     0.65666,
0.67267,     0.67367,     0.67467,     0.67568,     0.67668,     0.67768,     0.67868,     0.67968,     0.68068,
 0.6967,      0.6977,      0.6987,      0.6997,      0.7007,      0.7017,      0.7027,      0.7037,      0.7047,
0.72072,     0.72172,     0.72272,     0.72372,     0.72472,     0.72573,     0.72673,     0.72773,     0.72873,
0.74474,     0.74575,     0.74675,     0.74775,     0.74875,     0.74975,     0.75075,     0.75175,     0.75275,
0.76877,     0.76977,     0.77077,     0.77177,     0.77277,     0.77377,     0.77477,     0.77578,     0.77678,
0.79279,     0.79379,     0.79479,      0.7958,      0.7968,      0.7978,      0.7988,      0.7998,      0.8008,
0.81682,     0.81782,     0.81882,     0.81982,     0.82082,     0.82182,     0.82282,     0.82382,     0.82482,
0.84084,     0.84184,     0.84284,     0.84384,     0.84484,     0.84585,     0.84685,     0.84785,     0.84885,
0.86486,     0.86587,     0.86687,     0.86787,     0.86887,     0.86987,     0.87087,     0.87187,     0.87287,
0.88889,     0.88989,     0.89089,     0.89189,     0.89289,     0.89389,     0.89489,      0.8959,      0.8969,
0.91291,     0.91391,     0.91491,     0.91592,     0.91692,     0.91792,     0.91892,     0.91992,     0.92092,
0.93694,     0.93794,     0.93894,     0.93994,     0.94094,     0.94194,     0.94294,     0.94394,     0.94494,
```

```
# Save the model to Google Drive
model.save('/content/drive/MyDrive/honeybeeSeg/yolov8_custom_seg.pt')
```

```
# Evaluate the model
metrics = model.val()
```

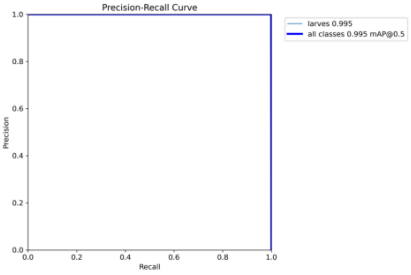
```
Ultralytics YOLOv8.2.82 🚀 Python-3.10.12 torch-2.4.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
val: Scanning /content/valid/labels.cache... 74 images, 0 backgrounds, 0 corrupt: 100%|██████████| 74/74 [00:00<?, ?it/s]
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incompatible with multithr
  self.pid = os.fork()
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95
                   all         74         74      0.999          1      0.995      0.952      0.999          1      0.995      0.96
                larves         74         74      0.999          1      0.995      0.952      0.999          1      0.995      0.96
Speed: 0.4ms preprocess, 10.1ms inference, 0.0ms loss, 8.2ms postprocess per image
Results saved to runs/segment/yolov8_custom_seg3
```

```
print("Performance Metrics:")
for key, value in metrics.results_dict.items():  # Access results_dict as an attribute
    print(f"{key}: {value:.4f}")
```

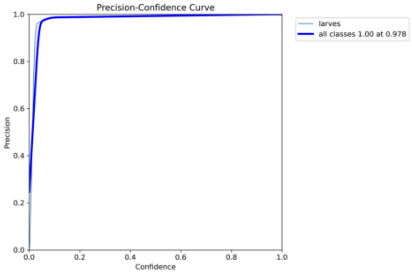
```
Performance Metrics:
metrics/precision(B): 0.9992
metrics/recall(B): 1.0000
metrics/mAP50(B): 0.9950
metrics/mAP50-95(B): 0.9520
metrics/precision(M): 0.9992
metrics/recall(M): 1.0000
metrics/mAP50(M): 0.9950
metrics/mAP50-95(M): 0.9669
fitness: 1.9260
```

```
import os
import cv2
import matplotlib.pyplot as plt

def display_segmented_outputs_as_subplots(results_dir, columns=3, image_size=(10, 10)):
    """
    This function displays the segmented images saved in the results directory as subplots.

    :param results_dir: Path to the directory containing the segmented images
    :param columns: Number of columns in the subplot grid
    :param image_size: Size of each subplot image
    """
    # Get a list of image files in the results directory
    image_files = [f for f in os.listdir(results_dir) if f.endswith(('.png', '.jpg', '.jpeg'))]

    # Determine the number of rows based on the number of images
    rows = (len(image_files) + columns - 1) // columns

    # Create a subplot figure
    fig, axes = plt.subplots(rows, columns, figsize=(columns * image_size[0], rows * image_size[1]))

    # Flatten axes for easy iteration
    axes = axes.flatten()

    # Display each segmented image in a subplot
    for i, image_file in enumerate(image_files):
        image_path = os.path.join(results_dir, image_file)

        # Load the image using OpenCV
        image = cv2.imread(image_path)

        # Convert BGR (OpenCV default) to RGB for correct color display
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        # Display the image on the corresponding subplot axis
        axes[i].imshow(image_rgb)
        axes[i].set_title(f"{image_file}")
        axes[i].axis('off')  # Hide axes

    # Hide any remaining empty subplots
    for j in range(i + 1, len(axes)):
        axes[j].axis('off')

    # Adjust layout
    plt.tight_layout()
    plt.show()

# Example usage
display_segmented_outputs_as_subplots('/content/runs/segment/yolov8_custom_seg3', columns=3, image_size=(
```

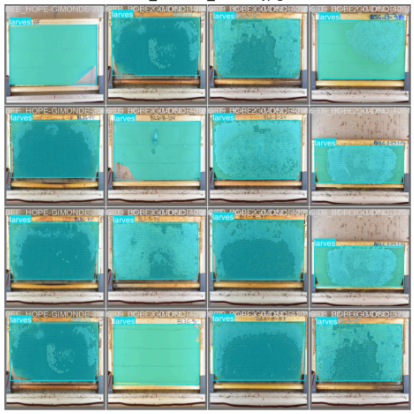
MaskPR_curve.png



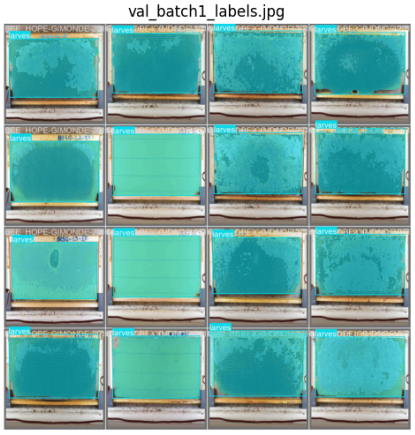
BoxP_curve.png



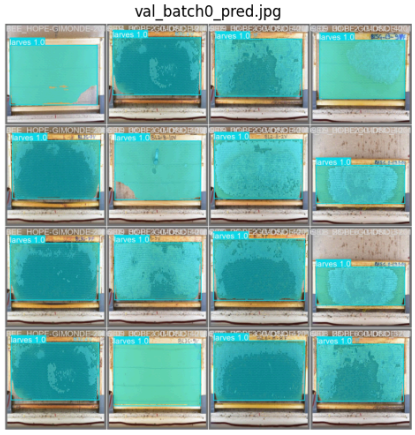
val_batch0_labels.jpg



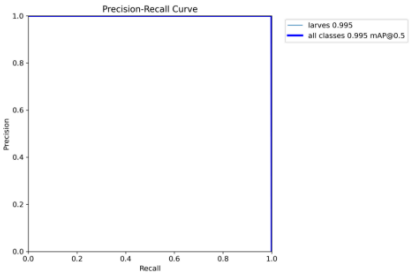
val_batch1_labels.jpg



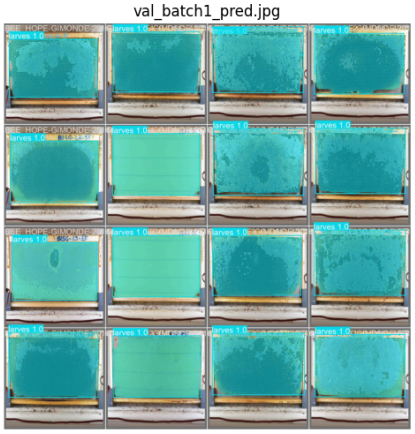
val_batch0_pred.jpg



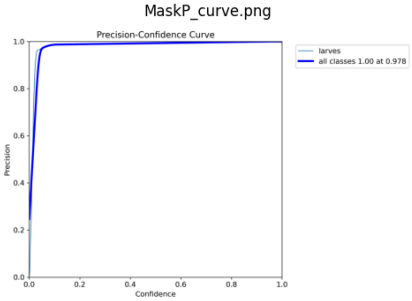
BoxPR_curve.png



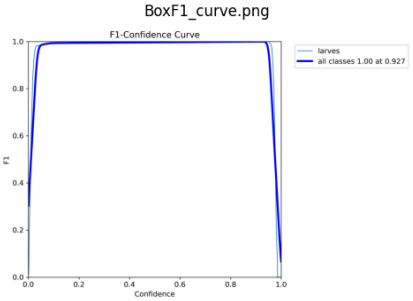
val_batch1_pred.jpg



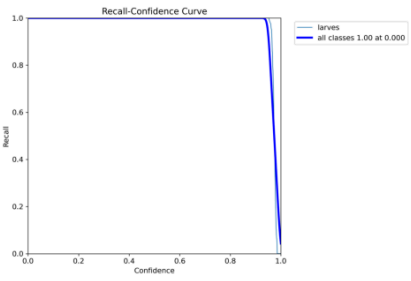
MaskP_curve.png



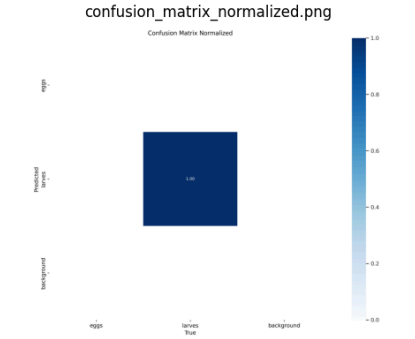
BoxF1_curve.png



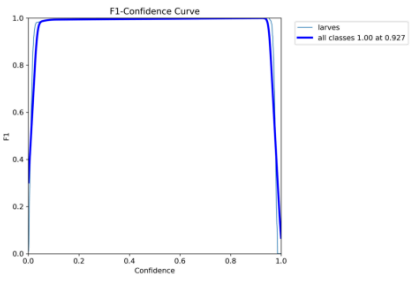
MaskR_curve.png



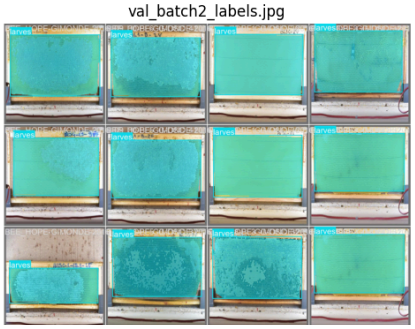
confusion_matrix_normalized.png



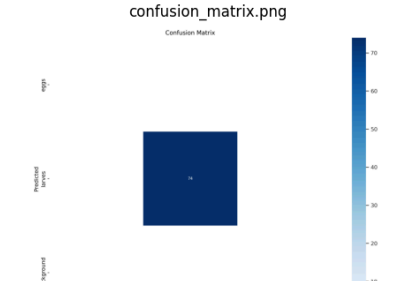
MaskF1_curve.png



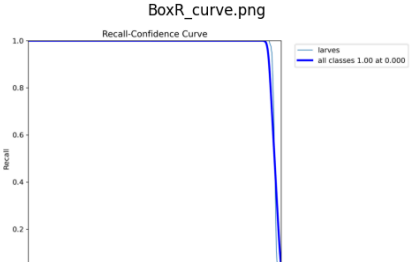
val_batch2_labels.jpg

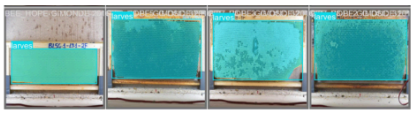


confusion_matrix.png

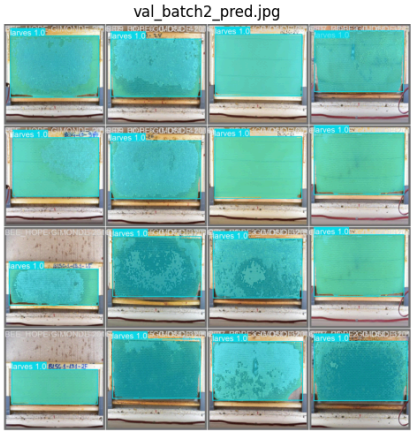


BoxR_curve.png





val_batch2_pred.jpg



```
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

# Load the YOLOv8 segmentation model
model = YOLO('/content/yolov8n-seg.pt')  # Replace with your model path

def segment_image(image_path):
    # Perform segmentation
    results = model.predict(source=image_path, save=False)

    # Extract the segmented image
    segmented_image = results[0].plot()

    # Convert the image from BGR to RGB for matplotlib
    segmented_image_rgb = cv2.cvtColor(segmented_image, cv2.COLOR_BGR2RGB)

    # Display the segmented image
    plt.figure(figsize=(10, 10))
    plt.imshow(segmented_image_rgb)
    plt.axis('off')  # Hide axes
    plt.show()

# Example usage
segment_image('/content/BEE_HOPE GIMONDE 2016_07_28 BL1_G DSC_3153.JPG')  # Replace with the path to your
```

```
image 1/1 /content/BEE_HOPE GIMONDE 2016_07_28 BL1_G DSC_3153.JPG: 448x640 (no detections), 11.8ms
Speed: 3.2ms preprocess, 11.8ms inference, 0.5ms postprocess per image at shape (1, 3, 448, 640)
```