python

Copy code

```
import cv2 import numpy as np import math import time # initialize the camera camera = cv2.VideoCapture(0)
```

- We start by importing the necessary libraries: OpenCV (`cv2`), NumPy (`numpy`), and `math`. We also import the `time` module for later use.
- Then, we initialize the camera using `cv2.VideoCapture(0)`. This will open the default camera (usually the built-in camera on a laptop or the first camera connected to a desktop) and set it as the source for capturing video frames.

python

Copy code

```
def get_distance_to_object(): # capture a frame from the camera ret, frame = camera.read() if not ret: return None # convert the frame to grayscale gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

- Next, we define a function called `get_distance_to_object()` that will capture a frame from the camera and calculate the distance to an object in the frame.
- We use `camera.read()` to capture a frame from the camera. The `ret` variable will be set to `True` if the frame was successfully read, or `False` if there was an error (for example, if the camera is disconnected). The `frame` variable will contain the captured frame as a NumPy array.
- If `ret` is `False`, we return `None` to indicate that there was an error.
- We then convert the frame to grayscale using `cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)`. This will make it easier to find the object in the frame later.

python

Copy code

```
# apply Gaussian blur to reduce noise blurred = cv2.GaussianBlur(gray, (5, 5), 0) # perform edge detection using Canny edges = cv2.Canny(blurred, 50, 150, apertureSize=3)
```

- We apply a Gaussian blur to the grayscale image to reduce noise. This is done using `cv2.GaussianBlur()`.
- We then perform edge detection using the Canny algorithm. This is done using `cv2.Canny()`. The second and third arguments specify the lower and upper thresholds for the

edges. Any edges with intensity gradient less than the lower threshold are discarded, and any edges with intensity gradient greater than the upper threshold are considered strong edges. Edges with intensity gradient between the two thresholds are considered weak edges if they are connected to strong edges. The `apertureSize` parameter specifies the size of the Sobel kernel used for edge detection.

python

Copy code

```
# find contours in the edge map contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) # find the contour with the largest area max_area = 0 max_contour = None for contour in contours: area = cv2.contourArea(contour) if area > max_area: max_area = area max_contour = contour # if no contour is found, return None if max_contour is None: return None
```

- We use `cv2.findContours()` to find contours in the edge map. The first argument is the input image (the edge map), and the second and third arguments specify the contour retrieval mode and contour approximation method, respectively. We use `cv2.RETR_EXTERNAL` to retrieve only the external contours (contours that form the outer boundary of an object) and `cv2.CHAIN_APPROX_SIMPLE` to approximate the contours by their