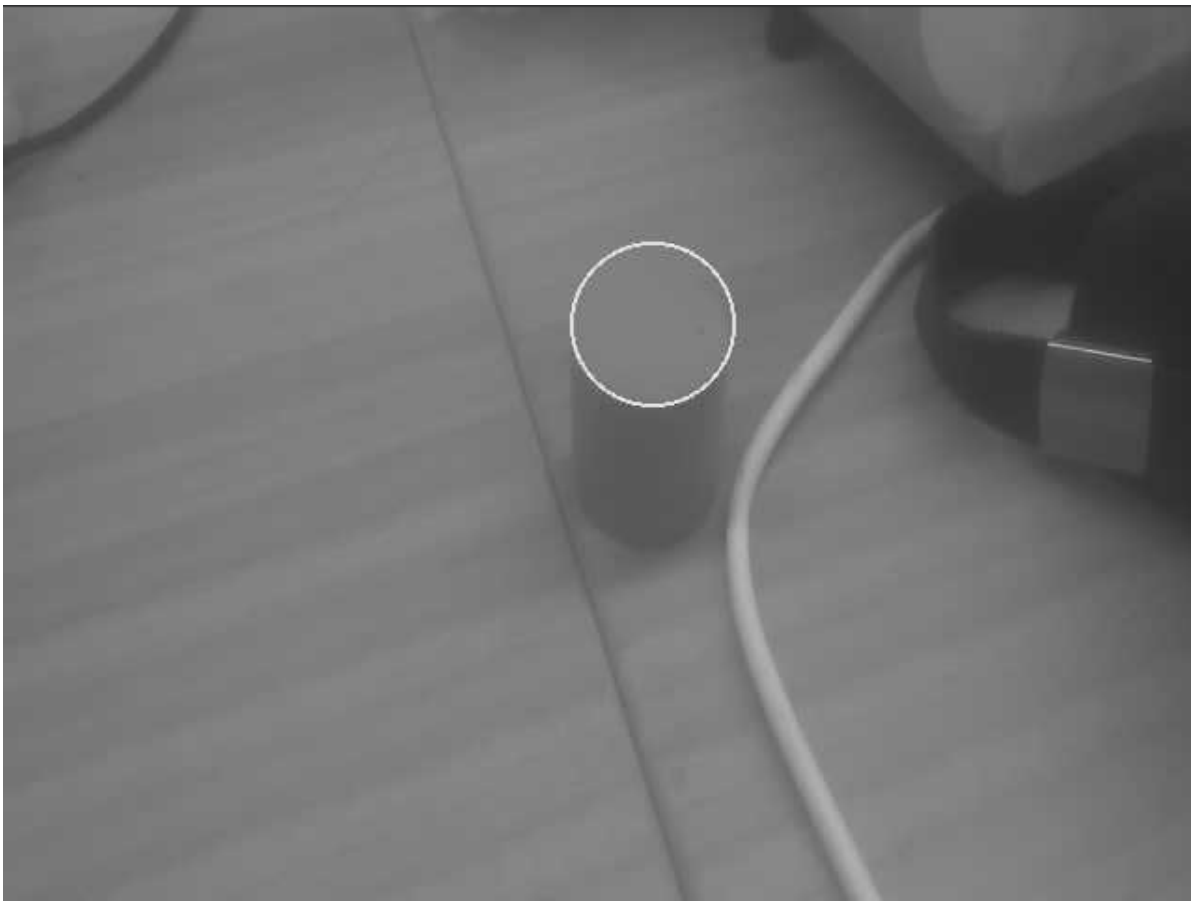
Circle Recognition - Grayscale Image

Example Results

Run this section's example code [Source Code/06.cv_lite/3.grayscale_find_circles.py]

This section uses the `cv_lite` extension module to implement circle detection. It captures images in real time using a camera and draws a circular outline for annotation after detecting circular objects.

> The average frame rate of this section's examples is around 15 frames per second. The specific frame rate depends on the actual detection scenario.



Code Explanation

Code Snippet Explanation

Importing Modules

```
import time, os, sys, gc
from machine import Pin
from media.sensor import * # Camera interface
from media.display import * # Display interface
from media.media import * # Media manager
import _thread
import cv_lite # cv_lite extension module (contains circle detection functions)
import ulab.numpy as np # NumPy-like ndarray for MicroPython
```

These modules provide the functionality required for camera control, image display, memory management, and Hough circle detection.

Set image size

```
image_shape = [480, 640] # Height x Width
```

Define the image resolution to 480x640 (height x width). The camera and display module will be initialized based on this size.

Initialize the camera

```
sensor = Sensor(id=2, width=1280, height=720, fps=90)
sensor.reset()
sensor.set_framesize(width=image_shape[1], height=image_shape[0])
sensor.set_pixformat(Sensor.GRAYSCALE)  # Set to grayscale image output /
Grayscale mode
```

- Initialize the camera, set the resolution to 1280x720 and the frame rate to 90 FPS.
- Resize the output frame to 640x480 and set it to grayscale mode (single-channel image, saves memory and computing resources, suitable for Hough circle detection).

Initialize the display module

```
Display.init(Display.ST7701, width=image_shape[1], height=image_shape[0],
to_ide=True, quality=50)
```

Initialize the display module, use the ST7701 driver chip, and the resolution is consistent with the image size. `to_ide=True` means that the image will be transmitted to the IDE for virtual display at the same time, and `quality=50` sets the image transmission quality.

Initialize the media manager and start the camera

```
MediaManager.init()
sensor.run()
```

Initialize the media resource manager and start the camera to start capturing the image stream.

Setting Hough Circle Detection Parameters

```
dp = 1 # Inverse ratio of resolution
minDist = 20 # Minimum distance between centers
param1 = 80 # Upper threshold for Canny edge detector
param2 = 20 # Accumulator threshold for center detection
minRadius = 10 # Minimum radius to detect
maxRadius = 50 # Maximum radius to detect
clock = time.clock() # Start FPS timer
```

- `dp`: Accumulator resolution ratio. A value of 1 means the accumulator resolution is the same as the input image. Larger values result in faster detection but lower accuracy.
- `minDist`: Minimum distance between detected circle centers, used to avoid repeated detection of the same circle.
- `param1`: The high threshold for Canny edge detection, used for edge extraction. A higher value results in fewer edges being detected.
- `param2`: The accumulator threshold, used for circle center detection. A lower value results in more circles being detected (but may include noise).
- `minRadius` and `maxRadius`: Limit the radius of the detected circles, detecting only those that fall within this range.
- `clock`: Used for calculating the frame rate.

Image Processing and Hough Circle Detection

```
while True:
    clock.tick()

    # Capture a frame
    img = sensor.snapshot()
    img_np = img.to_numpy_ref() # Get image data reference

    # Detect circles using Hough Transform
    # Return format: [x1, y1, r1, x2, y2, r2, ...]
    circles = cv_lite.grayscale_find_circles(
        image_shape, img_np,
        dp, minDist,
        param1, param2,
        minRadius, maxRadius
    )

    # Iterate over the circle information and draw each detected circle
    for i in range(0, len(circles), 3):
        x = circles[i]
        y = circles[i + 1]
        r = circles[i + 2]
        img.draw_circle(x, y, r, color=(255, 255, 0), thickness=2)  #  Yellow
circle

   # Show processed image
   Display.show_image(img)

   # Cleanup and print FPS
   gc.collect()
   print("findcircles:", clock.fps())
```

- **Image capture**: Get an image frame using `sensor.snapshot()` and convert it to a NumPy array reference using `to_numpy_ref()`.
- **Hough circle detection**: Call `cv_lite.grayscale_find_circles()` to detect circles and return a list of detected circle information (each circle occupies 3 elements: `[x, y, r]`, i.e., the center coordinates and radius).
- **Draw a circle**: Traverse each detected circle and draw a yellow circular outline (`color=(255, 255, 0)`) on the image to mark the target area.
- **Display image**: Display the processed image to the screen or IDE virtual window.
- **Memory management and frame rate output**: Call `gc.collect()` to clean up the memory and print the current frame rate through `clock.fps()`.

Resource release

```
sensor.stop()
Display.deinit()
os.exitpoint(os.EXITPOINT_ENABLE_SLEEP)
time.sleep_ms(100)
MediaManager.deinit()
```

When the program exits, stop the camera, release the display module resources, and clean up the media manager.

Parameter adjustment instructions

- `dp`: Adjust the accumulator resolution ratio. The larger the value, the faster the detection speed, but some circles may be missed. It is recommended to start the test from 1.
- `minDist`: Adjust based on the size and distribution of circles. If circles are close together, reduce this value; if duplicate circles are detected, increase it.
- `param1`: High threshold for Canny edge detection. A higher value results in fewer edges being detected. If the environment is noisy, increase this value appropriately.
- `param2`: Accumulator threshold. A lower value detects more circles but may include noise; a higher value results in stricter detection but may result in missed detections.
- `minRadius` **and** `maxRadius`: Adjust the range based on the size of the target circle to ensure that only the expected circles are detected.