Circle Recognition - Color Image

Circle Recognition - Color Image

- Example Results
- Code Overview
  - Importing Modules
    - Setting the image size
    - Initialize the camera (RGB888 format)
    - Initialize the display module
    - Initialize the media manager and start the camera
    - Setting Hough Circle Detection Parameters
    - Image Processing and Hough Circle Detection
    - Resource release
  - Parameter adjustment instructions

Example Results

Run this section's example code [Source Code/06.cv_lite/4.rgb888_find_circles.py]

In this section, we will use the `cv_lite` extension module to implement Hough circle detection in RGB888 format on an embedded device.

Code Overview

Importing Modules

```
import time, os, sys, gc
from machine import Pin
from media.sensor import * # Import the camera interface
from media.display import * # Import the display interface
from media.media import * # Import the media manager
import _thread
import cv_lite # Import the cv_lite extension module
import ulab.numpy as np # MicroPython class NumPy library
```

Setting the 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 later.

Initialize the camera (RGB888 format)

```
sensor = Sensor(id=2, width=1280, height=960, fps=90)
sensor.reset()
sensor.set_framesize(width=image_shape[1], height=image_shape[0])
sensor.set_pixformat(Sensor.RGB888) # Set RGB888 pixel format / Set RGB888 pixel
format
```

- Initialize the camera, set the resolution to 1280x960 and the frame rate to 90 FPS.

> Note: Setting the camera resolution to 1280*960 can effectively suppress display distortion, but there is a chance of a purple screen when running this code.

- Resize the output frame to 640x480 and set it to RGB888 format (three-channel color image, suitable for circle detection in color images).

Initialize the display module

```
Display.init(Display.ST7701, width=image_shape[1], height=image_shape[0],
to_ide=True, quality=100)
```

Initialize the display module, using the ST7701 driver chip, with a resolution consistent with the image size. `to_ide=True` indicates that the image will be simultaneously transmitted to the IDE for virtual display, and `quality=100` sets the image transmission quality to a higher level.

Initialize the media manager and start the camera

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

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

Setting Hough Circle Detection Parameters

```
dp = 1 # Inverse ratio of accumulator resolution to image resolution
minDist = 30 # Minimum distance between detected centers
param1 = 80 # Higher threshold for Canny edge detection
param2 = 20 # Threshold for center detection in accumulator
minRadius = 10 # Minimum circle radius
maxRadius = 50 # Maximum circle radius
clock = time.clock() # Start FPS timer
```

- `dp`: Accumulator resolution ratio. A value of 1 indicates the same resolution as the image. Larger values result in faster detection but potentially 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 to detect image edges. A higher value results in fewer edges being detected.
- `param2`: The threshold for circle center detection in the Hough transform. A lower value results in more circles being detected (but may contain noise).
- `minRadius` and `maxRadius`: Limit the radius of the detected circles, detecting only those that fall within this range.
- `clock`: Used to calculate 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 an RGB888 ndarray reference

    # Call the cv_lite extended Hough circle detection function, returning a list
of circle parameters [x, y, r, ...]
    circles = cv_lite.rgb888_find_circles(
        image_shape, img_np, dp, minDist, param1, param2, minRadius, maxRadius
    )

    # Iterate over the detected circles and draw a circle frame
    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, 0, 0), thickness=2)  # Red circle

    # Display image with drawn circles
    Display.show_image(img)

    # Garbage collection
    gc.collect()

    # Print FPS
    print("findcircles:", clock.fps())
```

- **Image capture**: Acquire an image frame using `sensor.snapshot()` and convert it to a NumPy array reference using `to_numpy_ref()`.
- **Hough circle detection**: Call `cv_lite.rgb888_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 red circular outline (`color=(255, 0, 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()
```

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 detects fewer edges. If the environment is noisy, increase this value appropriately.
- `param2` : Hough transform threshold. A lower value detects more circles but may include noise; a higher value makes detection stricter but may miss detections.
- `minRadius` **and** `maxRadius` : Adjust the range based on the size of the target circle to ensure that only the expected circles are detected.