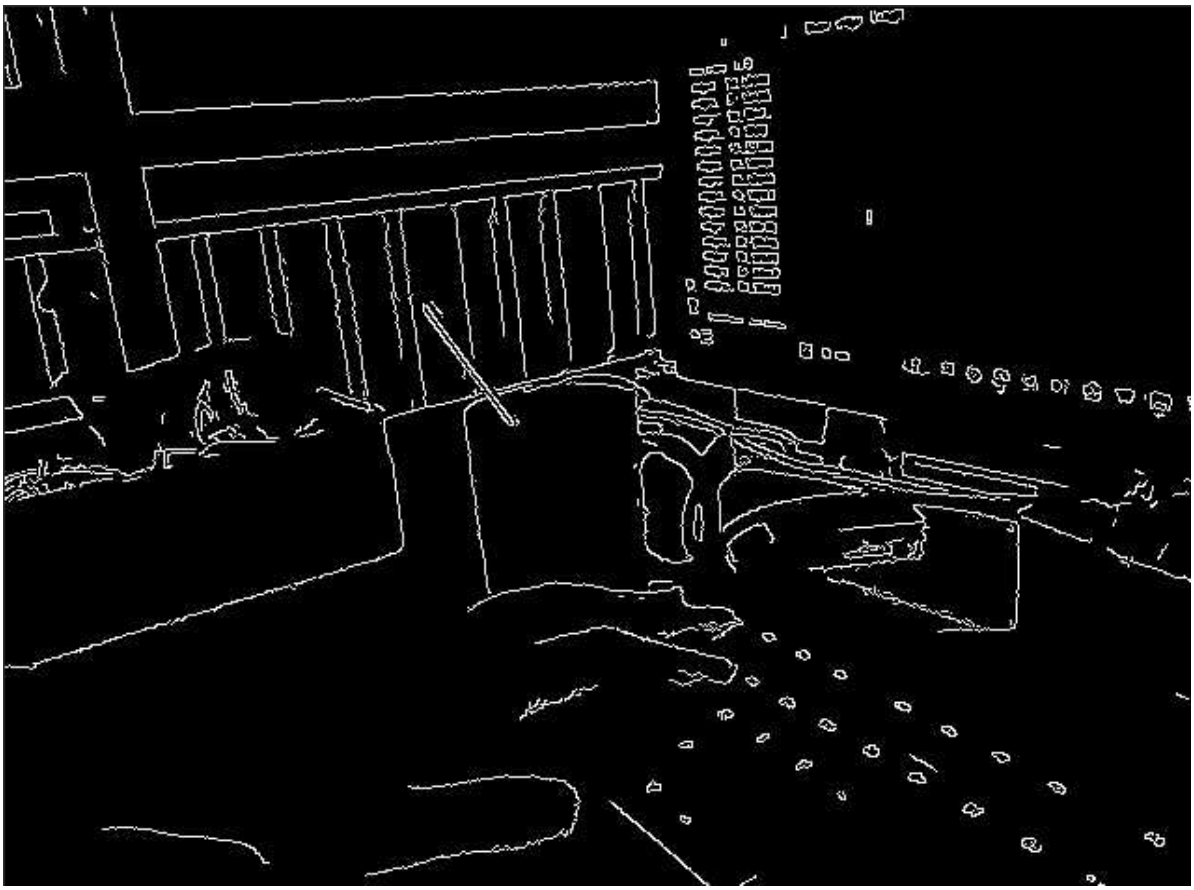
Edge Recognition - Grayscale Image

Edge Recognition - Grayscale Image
- Example Results
- Code Overview
  - Importing Modules
  - Setting the Image Size
    - Initialize the camera
    - Initialize the display module
    - Initialize the media manager and start the camera
    - Set edge detection parameters
    - Image processing and edge detection
    - Resource release
  - Parameter adjustment instructions

Example Results

Run the example code in this section [Source Code/06.cv_lite/7.grayscale_find_edges.py]

In this section, we will use the `cv_lite` extension module to implement grayscale image edge detection (Canny algorithm) on an embedded device.



Code Overview

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 (including edge detection)
import ulab.numpy as np # NumPy-like ndarray for MicroPython
```

Setting the Image Size

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

Define the image resolution to 480x640 (Height x Width). width), the camera and display module will be initialized based on this size later.

Initialize the camera

```
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.GRAYSCALE) # Set to grayscale image output /
Grayscale mode
```

- Initialize the camera, set the resolution to 1280x960 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 edge 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, 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=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 capture the image stream.

Set edge detection parameters

```
threshold1 = 50 # Lower threshold
threshold2 = 80 # Upper threshold
clock = time.clock() # Start FPS timer
```

- `threshold1`: The lower threshold for Canny edge detection, used to control the detection of weak edges.
- `threshold2`: The upper threshold for Canny edge detection, used to control the detection of strong edges. A larger value results in fewer edges being detected.
- `clock`: Used to calculate the frame rate.

Image processing and edge detection

```
while True:
    clock.tick()

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

    # Perform edge detection by calling the cv_lite extension
    # Returns edge image ndarray
    edge_np = cv_lite.grayscale_find_edges(
        image_shape, img_np, threshold1, threshold2)

    # Wrap ndarray as image for display
    img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
                          alloc=image.ALLOC_REF, data=edge_np)
    # Show edge image
    Display.show_image(img_out)

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

- **Image Capture**: Acquire an image frame using `sensor.snapshot()` and convert it to a NumPy array reference using `to_numpy_ref()`.
- **Edge Detection**: Call `cv_lite.grayscale_find_edges()` to perform Canny edge detection, returning a NumPy array of grayscale edge images.
- **Image Packaging and Display**: Pack the edge detection result into an image object and display it on the screen or in an IDE virtual window.
- **Memory Management and Frame Rate Output**: Call `gc.collect()` to clean up memory and print the current frame rate using `clock.fps()`.

Resource release

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

Parameter adjustment instructions

- `threshold1`: Low threshold. A smaller value can detect more weak edges, but may introduce noise. It is recommended to start adjusting from 50.
- `threshold2`: High threshold. A larger value can detect fewer strong edges, which is suitable for noisy environments. Increasing this value can reduce false edges.