Image Binarization - Grayscale

Example Results

Run the example code in this section [Source Code/06.cv_lite/13.grayscale_threshold_binary.py]

In this section, we will use the `cv_lite` extension module to implement grayscale image binarization 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 binarization interface)
import ulab.numpy as np # MicroPython NumPy library
```

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 (grayscale mode)

```
sensor = Sensor(id=2, width=1280, height=960, fps=30)
sensor.reset()
sensor.set_framesize(width=image_shape[1], height=image_shape[0])
sensor.set_pixformat(Sensor.GRAYSCALE) # Grayscale format
```

- Initialize the camera, set the resolution to 1280x960 and the frame rate to 30 FPS.
- Resize the output frame to 640x480 and set it to grayscale mode (single-channel image, saves memory and computing resources, suitable for binarization processing).

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 binarization parameters

```
thresh = 130 # Threshold value
maxval = 255 # Maximum value for white pixels after binarization
clock = time.clock() # Start FPS timer
```

- `thresh`: Binarization threshold. Pixels below this value will be set to 0 (black), while pixels greater than or equal to this value will be set to maxval (white).
- `maxval`: Maximum pixel value after binarization, typically set to 255 to represent white.
- `clock`: Used to calculate frame rate.

Image processing and binarization

```
while True:
    clock.tick()

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

    # Call cv_lite extension for binarization
    # Returns binary image ndarray
    binary_np = cv_lite.grayscale_threshold_binary(image_shape, img_np, thresh,
maxval)

    # Construct grayscale image object for display
    img_out = image.Image(image_shape[1], image_shape[0], image.GRAYSCALE,
    alloc=image.ALLOC_REF, data=binary_np)

    # Display binarization result / Show binary image
    Display.show_image(img_out)

    # Clear memory and print frame rate / Garbage collect and print FPS
    gc.collect()
    print("binary:", clock.fps())
```

- **Image capture**: Acquire an image frame using `sensor.snapshot()` and convert it to a NumPy array reference using `to_numpy_ref()`.
- **Binarization**: Call `cv_lite.grayscale_threshold_binary()` to perform binarization, returning a NumPy array of the binarized grayscale image.
- **Image packaging and display**: Package the binarization 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 clear 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

- `thresh`: Threshold. A higher value requires brighter pixels to turn white. Adjust based on image brightness. It is recommended to test from 100 to 200 to filter noise or highlight the target.
- `maxval`: Maximum value. Usually kept at 255. If you need to customize the output intensity, you can adjust it, but generally it does not need to be changed.