Object edge detection

Introduction to the experimental results of the routine

> **[Edge detection]** is a basic problem in image processing and computer vision. The purpose of [edge detection] is to identify points in digital images where the brightness changes significantly. Significant changes in image attributes usually reflect important events and changes in attributes.

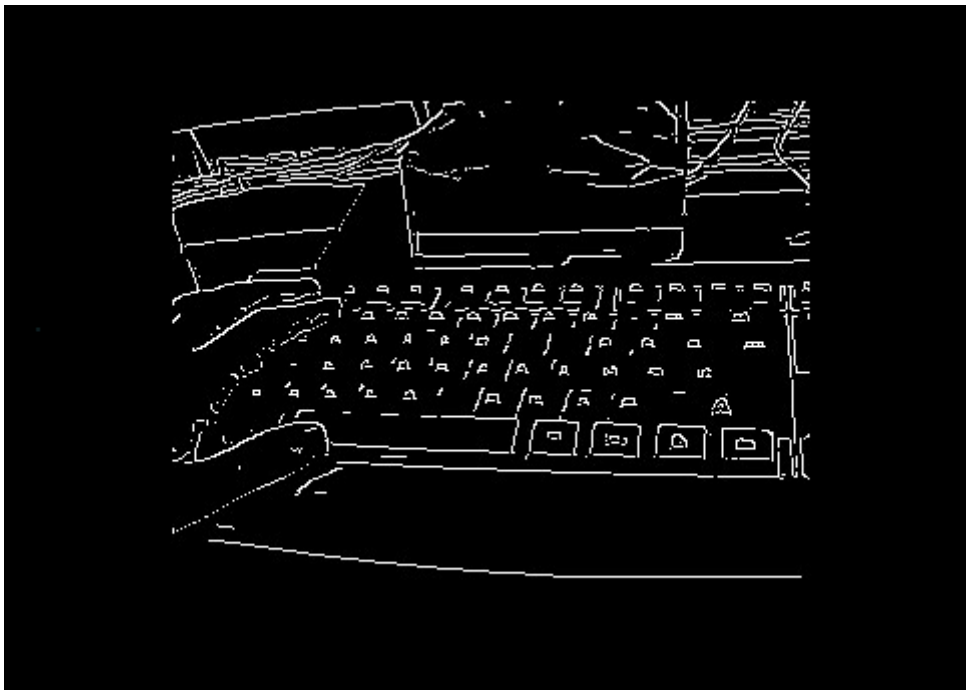
The routine code in this section is located at: [Source code/ 04.Detecting / 04.edges.py]

We use CanMV IDE to open the routine code and connect K230 to the computer via USB

Click the run button in the lower left corner of CanMV IDE,

Aim the camera of K230 at the surrounding objects

You can see that the edges of the objects will be marked with white lines on the screen



Code explanation

The peripherals we will use in this section are mainly camera modules

Line segment detection is implemented by the find_edges() method in K230, which belongs to the image module

Key code

```
# 初始化设备 / Initialize devices
sensor = init_sensor()
init_display()
MediaManager.init()
sensor.run()

# 初始化时钟 / Initialize clock
clock = time.clock()

# 计算显示偏移量以居中显示 / Calculate display offsets for center alignment
x_offset = round((DISPLAY_WIDTH - FRAME_WIDTH) / 2)
y_offset = round((DISPLAY_HEIGHT - FRAME_HEIGHT) / 2)

while True:
    clock.tick()  # 更新时钟 / Update clock

    # 捕获和处理图像 / Capture and process image
    img = sensor.snapshot()
    process_image(img)

    # 居中显示图像 / Display image in center
    Display.show_image(img, x=x_offset, y=y_offset)

    # 显示FPS / Display FPS
    print(clock.fps())
```

Code structure

Configuration section:

- Two sets of resolutions are defined: sensor capture resolution (1920x1080) and actual processed frame resolution (640x480). If you need to increase the frame rate, you can reduce the actual processed resolution.
- Set the display resolution to 640x480

Initialization function:

- `init_sensor()`: Configure the camera sensor, set the resolution and grayscale image format
- `init_display()`: Initialize the display

Image processing:

- `process_image()`: Use the Canny algorithm for edge detection, and set the threshold to 50-80

Main program flow:

- Initialize all devices (sensors, displays)
- Calculate the display offset to center the image on the screen
- Enter the main loop:

    - Capture the image
    - Perform edge detection processing
    - Display the processed image on the screen
    - Display the current FPS (frames per second)

Error handling:

- Contains a complete exception handling mechanism
- Resources will be cleaned up at the end of the program (stop the sensor, clean up the display, garbage collection)

Edge detection function

find_edge()

```
image.find_edges(edge_type[, threshold])
```

This function converts the image to black and white, keeping only white pixels at the edges.

- `edge_type` optional values include:
    - `image.EDGE_SIMPLE` - Simple threshold high-pass filter algorithm
    - `image.EDGE_CANNY` - Canny edge detection algorithm
- `threshold` is a two-element tuple containing a low threshold and a high threshold. You can control the edge quality by adjusting this value, and the default setting is `(100, 200)`.

Note: This method only supports grayscale images.