

Open MV 视觉识别

作者：032310212 郑林郁

1、技术原理

1、设置对比度

```
1 | sensor.set_contrast(1)
```

对于模板匹配来讲，图像太亮或太暗都会导致匹配效果变差，而对比度正是影响亮暗的主要参数之一。这里，设置对比度为1，使得图像更加清晰，又不会使图像失真。变换公式如下， α 为对比度， I 为原图像。

$$I_{\text{new}} = \alpha \cdot (I - 128) + 128$$

2、摄像头保持稳定

```
1 | sensor.skip_frames(time = 1000)
```

摄像丢弃 1s 的初始帧，待其稳定之后再进行处理。

3、关闭白平衡

```
1 sensor.set_auto_whitebal(False)
```

摄像头在不同光源下，会自动调整红绿蓝三个通道的增益比例，让图像看起来“自然”，在灰度图像下，虽然颜色本身不重要，但是白平衡仍可能通过传感器信号的增益分配去影响灰度值，影响模板匹配。所以，当我们处理对图像特点有严格要求的任务时，关闭即可。

4、设置曝光

```
1 Exposure_scale = 1 # 曝光调节尺度
2 print(f"< 最初曝光 >
   {sensor.get_exposure_us()}")
3 sensor.skip_frames(time = 1000)
4 print(f"< 当前曝光 >
   {(current_exposure:=sensor.get_exposure_us())}")
5 sensor.set_auto_exposure(False, exposure_us
   = int( current_exposure* Exposure_scale))
```

曝光时间越长，摄像头感光越多，图像就越亮，反之就越暗。选择合适的曝光，会使图像亮度整体合适。我在源代码上进行了优化，使代码简洁且 `pythonic`。

5、图像格式选择

常见的图像格式有很多，在这里，我们只探讨 `pgm`、`bmp`、`jpg`、`png` 这四种格式的特点和使用场景，以及为什么我们选择 `pgm` 作为模板匹配的图像保存格式。

`pgm`: 只能存储灰度图，文件体积大，但是加载速度快，读入图片后能直接使用，适合模板匹配。

`bmp`: 文件体积大，支持灰度、`RGB`等格式，适合需要简单读写的任务，一般用于调试和存储中间结果。

`jpg`: 有损压缩，会造成细节损失，适合精度要求不高的任务，一般用于展示给人看。

`png`: 无损压缩，适合于需要保真但文件不能太大的图像，可以作为模板匹配，但是解码速度不如 `pgm`。

6、模板匹配

```
1 R = img.find_template(template, 0.8,  
    step=1, roi=(50, 0, 60, 50),  
    search=SEARCH_EX)
```

`template`是用于匹配的模板，`0.8`是阈值，较高的阈值可以降低错误识别的概率，但是也会降低检测率，较低的阈值则相反。`roi`是感兴趣区域，`step`表示在查找模板时要跳过的像素数，可以加快算法速度，默认为2。`search`是模板匹配的搜索算法，具体介绍如下。

- `SEARCH_EX` 穷举搜索,精度高, 速度慢
- `SEARCH_DS` 随机搜索, 首先把模板和`roi`缩小, 找到大致范围再精细搜索

7、串口接收

```
1 | uart = pyb.UART(3, 115200, timeout_char =  
   | 1000) # 初始化串口  
2 | data[i] = uart.readchar()
```

串口定义中，参数 `timeout_char` 表示，接收字节的时间为 `1000 ms`，超时会认为无数据发送。并且要注意的是，串口接收函数返回的是一个十进制值（`int`），表示接收到的单个字节的 ASCII 值。

8、ROI 误区

```
1 | sensor.set_windowing((0, 50, 160, 50))
```

该函数会把摄像头捕捉到的图片裁剪到该函数，之后的函数，用到的 `roi` 都是在这个基础上限定感兴趣区域的。

9、增益

```
1 sensor.set_gainceiling(16) # 设置自动增益
2 .....
3 sensor.set_auto_gain(False) # 关闭自动增益
```

打开摄像头后，开启自动增益，使其根据环境自动找到合适的参数，之后我们关闭它以防止干扰图像。

10、详解任务一

```
1 # 识别目标数字
2 if Find_Task == 1:
3     temp=[FirstFindTemplate(img) for img
4         in template_base]
5     if not any(temp): continue
6     [(roi,idx)]=[(returned_roi,i) for
7         i,returned_roi in enumerate(temp) if
8         returned_roi]
9     FirstFindedNum(roi,idx+1)
```

本代码使用了大量的列表推导式，首先进行了模板匹配，并把结果存储在 `temp` 变量中。之后判断 `temp` 是否都是 `None`，如果是的表示并未匹配到模板，直接退出即可。之后对 `temp` 进行了解析，找出有效 `roi` 和他的索引，因为我在代码的开头对模板进行了排序，所以此时的索引+1 就是真实的数字。之后调用函数，发送给主控，并框出该数字。

11、详解任务二

```
1 # 任务2 :
2 elif Find_Task == 2:
3     # Target_Num是目标数字
4     if LoR==0:
5
6         ROI_L=FindTemplateL(template_L[Target_Nu
7 m-3])
8
9         ROI_R=FindTemplateL(template_L[Target_Nu
10 m-3])
11
12         ROI_LL=FindTemplateL(template_L[Target_N
13 um-3])
14
15         ROI_RR=FindTemplateL(template_L[Target_N
16 um-3])
17
18         if not LoR and any(ROI:=
19 [ROI_L,ROI_R,ROI_LL,ROI_RR]):
20             Target_ROI=[roi for roi in ROI if
21 roi][0]
22             FindedNum(Target_ROI,Target_Num)
```

本代码对源代码进行了优化，使得代码简洁，逻辑清晰。首先，判断LoR是否为0，如果是的话再进行模板匹配。因为之前对模板进行了排序，使得我们可以直接用 `Target_Num-1` 表示目标数字对应的索引，进而直接找到特定模板。然后，我们判断是否匹配到模板，也就

是判断是否 ROI 非空，之后用列表推导式找到目标 ROI，把信息发送给主控。

2、代码部分

```
1  import time,
    image,sensor,math,pyb,ustruct
2  from image import SEARCH_EX, SEARCH_DS
3  from pyb import Pin, Timer,LED
4  import os
5
6
7  # part1
8  #----- 基本配置 -----
    -----
9  sensor.reset() # 重置摄像头到初始配置
10 sensor.set_contrast(1) # 设置对比度
11 sensor.set_gainceiling(16) # 设置自动增益
12 sensor.set_framesize(sensor.QQVGA) # 160
    x 120 ，减小计算量
13 sensor.set_pixformat(sensor.GRAYSCALE)
    # 模板匹配要求灰度图
14 sensor.set_windowing((0, 50, 160, 50))
    # 直接设置感兴趣区域，减小计算量
15 sensor.set_auto_gain(False) # 关闭自动增
    益
```



```
16 sensor.set_auto_whitebal(False) # 关闭白
    平衡
17
18 Exposure_scale = 1 # 曝光调节尺度
19 print(f"< 最初曝光 >
    {sensor.get_exposure_us()}")
20 sensor.skip_frames(time = 1000)
21 current_exposure=sensor.get_exposure_us(
    )
22 print(f"< 当前曝光 > {current_exposure}")
23 sensor.set_auto_exposure(False,exposure_
    us=int(current_exposure*
    Exposure_scale))
24
25 uart = pyb.UART(3, 115200, timeout_char
    = 1000) # 初始化串口
26
27 #----- 基本配置 -----
    -----
28
29
30 # part2
31 #----- 参数初始化及读取模板
    -----
32
33 Find_Task = 2
34 Target_Num = 4
35 find_flag = 0
36 x = 0
37 data = [0x00]*8
```

```

38 LoR=0
39 dir=r"/Template"
40
41 # 图片读入
42 template_base,template_L,template_R,template_LL,template_RR=[],[],[],[],[]
43 for img_path in
sorted(os.listdir(dir),reverse=False):
44     value=img_path.split(".")[0]
45     value=value if len(value)==1 else
value[1:]
46     img=image.Image(dir+"/"+img_path)
47     index=-1
48     if value in ["L","R","LL","RR"]:
49         index=
["L","R","LL","RR"].index(value)
50     else:
51         index=-1
52
[template_L,template_R,template_LL,template_RR,template_base][index].append(img)
53
54 #----- 参数初始化及读取模板 -----
55 # 检验代码
56 # for tem in
[template_L,template_R,template_LL,template_RR,template_base]:
57 #     print(f"{len(tem)}")
58

```

```
59
60 # part3
61 #----- 功能函数定义 -----
   -----
62
63 def UartReceiveDate():
64     '''
65         UART接收函数,同时对收到的数据进行解码,返回任务和目标数字
66     '''
67     global Find_Task
68     global Target_Num
69     global x
70     global data
71
72     for i in range(8):data[i]=
uart.readchar()
73
74     if x < 5 and data[x] == 42 and
data[x+3] == 38:
75         Find_Task = data[x+1]
76         Target_Num = data[x+2]
77         Find_Task = Find_Task - 48
78         Target_Num = Target_Num - 48
79         x = 0
80     elif x >= 5: x = 0
81     x+=1
82     # print(f"\n< 任务码 > {Find_Task}")
83     # print(f"< 目标数字 >
{Target_Num}\n")
```

```
84
85 def FirstFindTemplate(template):
86     '''
87     寻找目标数字,并返回数字边框
88     '''
89     return img.find_template(template,
90                               0.8, step=1, roi=(50, 0, 60, 50),
91                               search=SEARCH_EX)
92
93 def FirstFindedNum(R, Finded_Num):
94     '''
95     画出数字矩形框,并把【目标数字,方向控制,识别标志,当前任务】发送给主控
96     '''
97     global Find_Task
98     global find_flag
99     img.draw_rectangle(R, color=(225, 0,
100                                0))
101     find_flag = 1
102     Num = Finded_Num
103     FH = bytearray([0x2C, 0x12, Num, LoR,
104                     find_flag, Find_Task, 0x5B])
105     uart.write(FH)
106     print("目标病房号: ", Num)
107
108 def FindTemplateL(template):
109     '''
110     左侧模板识别,并返回数字框
111     '''
```

```
108     return img.find_template(template,
0.5, step=1, roi=(0, 0, 70, 50),
search=SEARCH_EX)
109 def FindTemplaterR(template):
110     '''
111     右侧模板识别，并返回数字框
112     '''
113     R = img.find_template(template, 0.5,
step=1, roi=(90, 0, 70, 50),
search=SEARCH_EX)
114     return R
115 def FindTemplateLL(template):
116     '''
117     左左侧模板识别，并返回数字框
118     '''
119     R = img.find_template(template, 0.5,
step=1, roi=(0, 0, 70, 50),
search=SEARCH_EX)
120     return R
121 def FindTemplateRR(template):
122     '''
123     右右侧模板识别，并返回数字框
124     '''
125     R = img.find_template(template,
0.79, step=1, roi=(90, 0, 70, 50),
search=SEARCH_EX)
126     return R
127 def FindedNum(R, Finded_Num):
128     global Find_Task
129     global find_flag
```

```

130     '''
131     调整小车方向
132     '''
133     img.draw_rectangle(R, color=(225, 0,
0))
134     if R[0] >90:
135         LoR = 2
136     elif 0< R[0]<60:
137         LoR = 1
138     else:
139         LoR= 0
140     find_flag = 1
141     Num = Finded_Num
142     if LoR>0:
143         FH = bytearray([0x2c,0x12,Num,
LoR, find_flag, Find_Task,0x5B])
144         uart.write(FH)
145         print("识别到的数字是：", Num, "此数
字所在方位：", LoR)
146     #----- 功能函数定义 -----
-----
147
148
149     # part4
150     #----- 主函数 -----
-----
151
152     while (True):
153         img = sensor.snapshot()
154         UartReceiveDate()

```

```
155     # 任务0 : 直走
156     if Find_Task == 0:
157         LoR=0
158     # 任务1 : 识别目标数字
159     if Find_Task == 1:
160         temp=[FirstFindTemplate(img) for
img in template_base]
161         if not any(temp): continue
162         [(roi,idx)]=[returned_roi,i)
for i,returned_roi in enumerate(temp) if
returned_roi]
163         FirstFindedNum(roi,idx+1)
164     # 任务2 :
165     elif Find_Task == 2:
166         print(LoR)
167         # Target_Num是目标数字
168         if LoR==0:
169
170             ROI_L=FindTemplateL(template_L[Target_N
um-3])
171
172             ROI_R=FindTemplateR(template_R[Target_N
um-3])
173
174             ROI_LL=FindTemplateLL(template_LL[Targe
t_Num-3])
175
176             ROI_RR=FindTemplateRR(template_RR[Targe
t_Num-3])
177
178             ROI=[ROI_L,ROI_R,ROI_LL,ROI_RR]
```

```

174         if not LoR and any(ROI):
175             Target_ROI=[roi for roi in
ROI if roi][0]
176
FindedNum(Target_ROI,Target_Num)
177
178 #----- 主函数 -----
-----
179

```

3、主要步骤

1. 初始化摄像头和基本参数，**有序** 读取图像模板。设置图像对比度，设置自动增益之后关闭（原理见[链接](#)），减小分辨率，设置灰度图，手动划定窗口以减小计算量。
2. 串口接收并进行解码，得到任务码，并选择不同的处理方式。
 1. **任务0**。保持现状
 2. **任务1**。识别找到目标数字，具体方法如下。将帧图像与模板进行比较，如果都没有匹配到，就直接退出；如果匹配到，就找出目标所在的感兴趣区域和对应的真实数字，经过处理后发送给主控。
 3. **任务2**。判断 LoR，只有当其为0，也就是小车直走的时候再进行模板识别。接着，我们让帧图像和各个模板比较，找到目标区域，并把信息发送给主控。

4、效果展示

