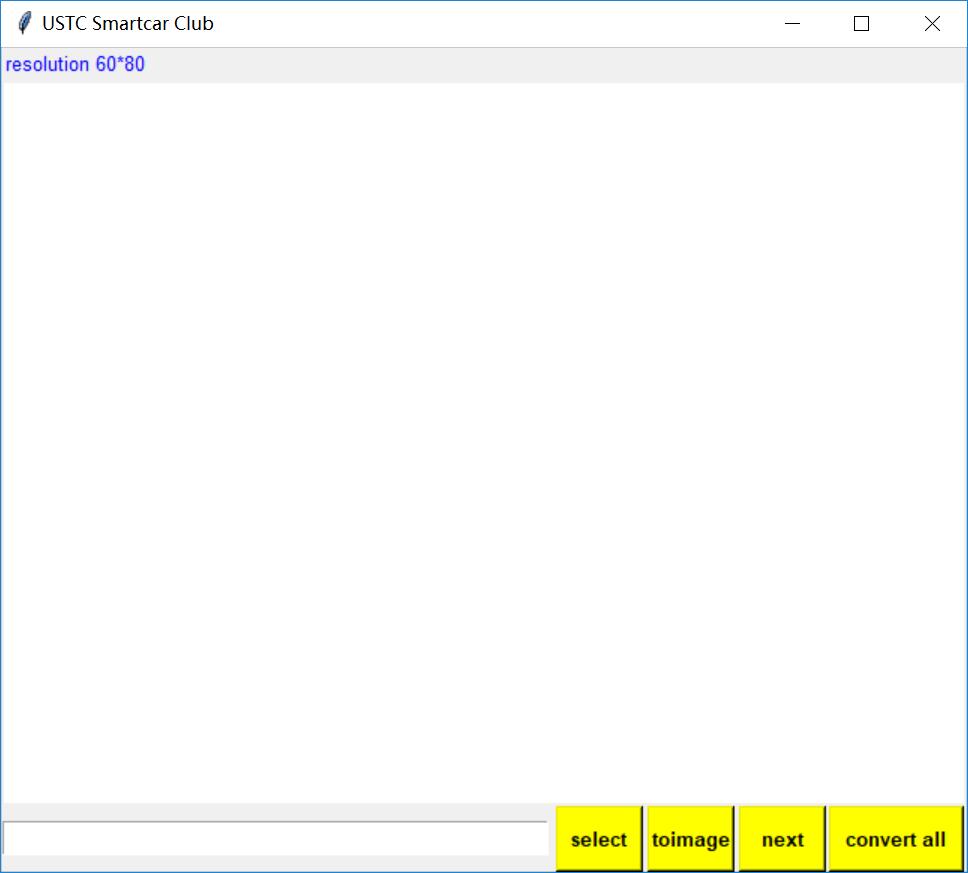
**循迹小助手V1.0使用教程**

1. **简介**

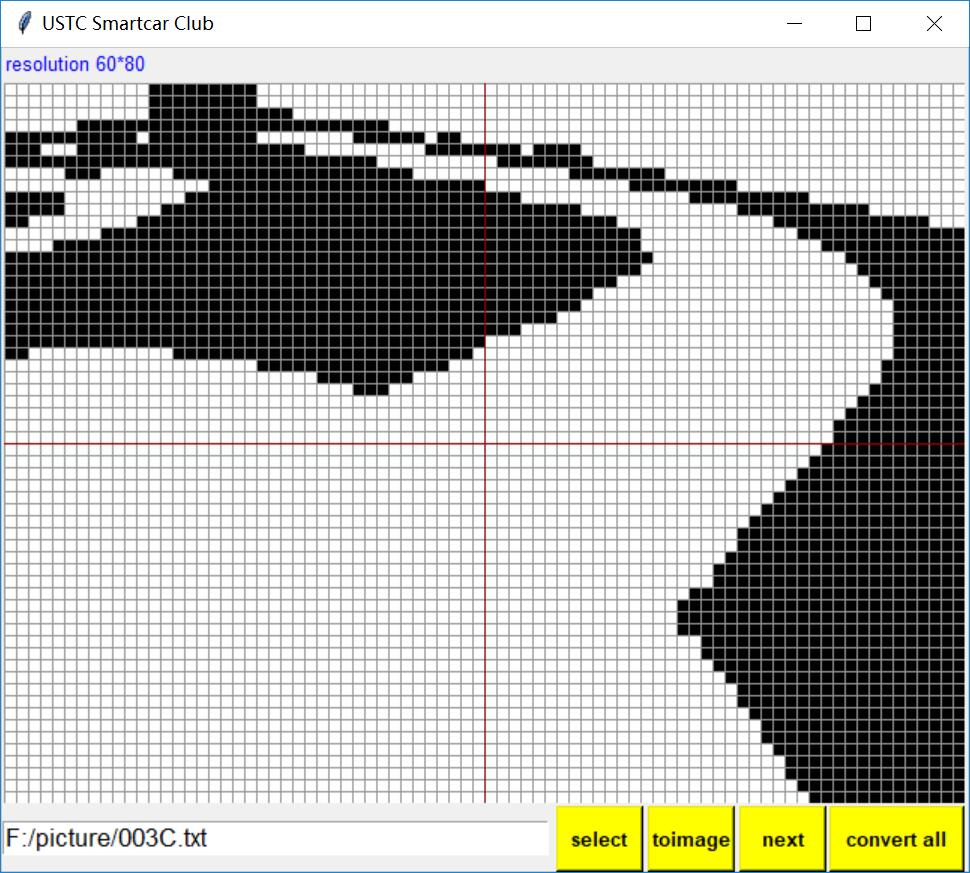
前几天有同学问能不能拿车回家调试，因为实验室的东西不能带出实验室是实验室的基本规则之一，所以请大家遵守规定！但是没车就算回家写了识别赛道的算法也无法验证自己写的对不对，所以我写了个循迹小助手帮助大家验证循迹算法，也就是验证你找到的左右边界和中线对不对。

先看下软件界面，界面很简单（丑-\_-）

初次打开界面是这样的：



加载图片后是这样的：



1. **功能介绍：**

使用循迹小助手前要做点准备，通过QQ之类的可截图的软件截取山外调试助手显示的赛道图片如图，截图最好是刚好和那个框框一样大



然后保存在一个文件夹（文件夹目录不要带中文，以下假设这个文件目录为dir）里，第一次帮你们准备好了图片，以后可根据需要自行截图。如果不想截图的话也可以，将赛道信息打印出来保存到dir中，文件名为XXXC.txt（XXX可表示000~999，以下同）。循迹小助手可以将截取的图片转换成XXXC.txt，XXXC.txt里面保存的就是原始赛道信息，共60行，每行80个数字，全是0和1，0表示黑，1表示白。具体用哪种方法你们可自行选。

下面一一介绍四个按钮的功能。

**convert all**

* 为了便于后续处理会将你截图保存的图片重命名为XXXA.jpg，仅支持将原来后缀名为\*.jpg ，\*.jpeg ，\*.png的图片重命名。点击这个按钮之后，选择dir中的任意一个文件，就会将dir中的所有那几种图片重命名。如果你是第一次使用小助手或者在dir中添加了新的截图，可以先使用这个按钮，这时不仅会将图片重命名，还会生成对应的XXXC.txt文件，以及由XXXC.txt生成的带网格的赛道图片XXXB.jpg。
* 得到XXXC.txt文件之后你们就可以用其中保存的原始赛道信息进行处理了，你们自己写个C语言程序，处理XXXC.txt得到XXXD.txt，在附录1中我给出了一个代码框架，你们在框架里写就行了。XXXD.txt中对应左右边界和中线的位置数值设置为2，其余部分与XXXC.txt相同。如果dir中存在XXXD.txt，你再次点击convert all选中dir中任意一个文件，就会在dir中生成由XXXD.txt得到的XXXB.jpg，原来的XXXB.jpg被覆盖了，XXXD.txt中值为2的位置在XXXB.jpg中显示为红色，也就是红色线就是你的左右边界和中线。附录2给出了赛道中线提取的一个思路。

**select**

* 在dir中选择一个.txt文件，也就是XXXC.txt或者XXXD.txt，选好之后在左下角的文本区中会显示出文件绝对路径名。设计这个按钮是因为用convert all按钮一次会把所有XXXD.txt转换为图片，如果数据量大可能需要比较久的时间，这个按钮支持选择一个txt文件进行转换。也可直接在文本区输入文件绝对路径名。

**toimage**

* 点击之后将文本区显示的XXXC.txt或者XXXD.txt文件转换成图片XXXB.jpg并在图片区显示出来。

**next**

* 转换下一张图片。如果当前文本区的显示文件是023C.txt，点击之后就会将024C.txt转换成024B.jpg并显示出来。对XXXD.txt也是这样。

**附录1 程序框架**

#include<stdio.h>

#include<process.h>

#include<string.h>

#define N 100//N为含有未处理过的赛道信息的txt文件数量

void initCD(int i,char \*s,char \*TextC,char \*TextD);

void read(char\*);

void write(char\*);

int img[60][80];//保存从一个\*C.txt中读到的赛道信息

int imgnew[60][80];//保存处理过后的赛道信息，写入对应的\*D.txt

int main(){

char dir[50]="F:\\picture\\";//s为文件夹目录,视具体情况而定

char TextC[N][50];//用来保存你读取的文本文件名，这些文件必须为\*C.txt形式

//每个文件中含有60行，每行80个数字,全是0和1，为原始赛道信息

char TextD[N][50];//用来保存你处理后含有左右边界和中线信息的文本文件名

//这些文件必须为\*D.txt形式，每个文件中含有60行，每行80个数字

int i;

for(i=0;i<N;i++){

initCD(i,dir,TextC[i],TextD[i]);

read(TextC[i]);//从文件TextC[i]中读取出赛道信息到img[60][80]中

/\*\*\*\*\*\*\*\*\*\*\*\*\*

处理img[60][80]得到左右边界和中线，进而得到imgnew[60][80]

imgnew[60][80]中对应左右边界和中线的点值设为2，其余同img[60][80]

这是大家自己处理的主要部分，有较大难度,一定要有耐心

\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

write(TextD[i]);//将处理后得到的imgnew[60][80]写入TextD[i]中

}

return 0;

}

char s[4];

char\* num2str(int i)

{

if(i<10){

s[0]='0'; s[1]='0'; s[2]='0'+i;

}

else if(i<100){

s[0]='0'; s[1]='0'+i/10; s[2]='0'+i%10;

}

else{

s[0]='0'+i/100; s[1]='0'+(i/10)%10; s[2]='0'+i%10;

}

s[3]='\0';

return s;

}

void initCD(int i,char \*s,char \*TextC,char \*TextD)

{

strcpy(TextC,s);

strcat(TextC,num2str(i));

strcat(TextC,"C.txt");

strcpy(TextD,s);

strcat(TextD,num2str(i));

strcat(TextD,"D.txt");

}

void read(char\* file)

{

FILE \*fp;

int i,j;

char c;

if((fp=fopen(file,"r"))==NULL)

{

printf("cannot open file %s\n",file);

return ;

}

for(i=0;i<60;i++){

for(j=0;j<81;j++){

c=fgetc(fp);

if(c!='\n')

{

img[i][j]=c-48;

}

}

}

return;

}

void write(char \*file)

{

FILE \*fp;

int i,j;

if((fp=fopen(file,"w"))==NULL)

{

printf("cannot open file:%s\n",file);

return ;

}

for(i=0;i<60;i++){

for(j=0;j<80;j++)

fputc(imgnew[i][j]+48,fp);

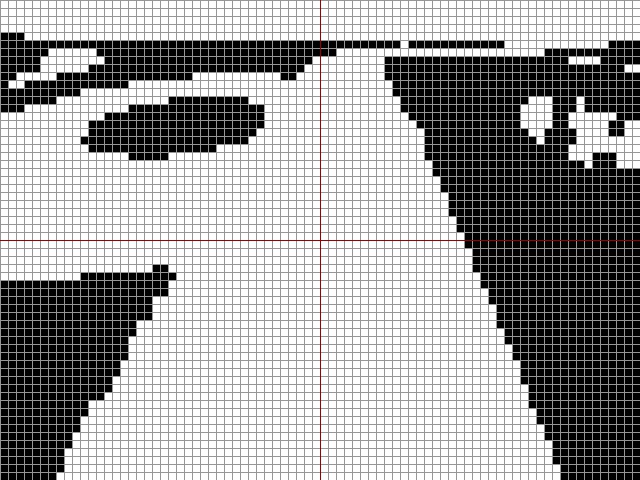
fputc('\n',fp);

}

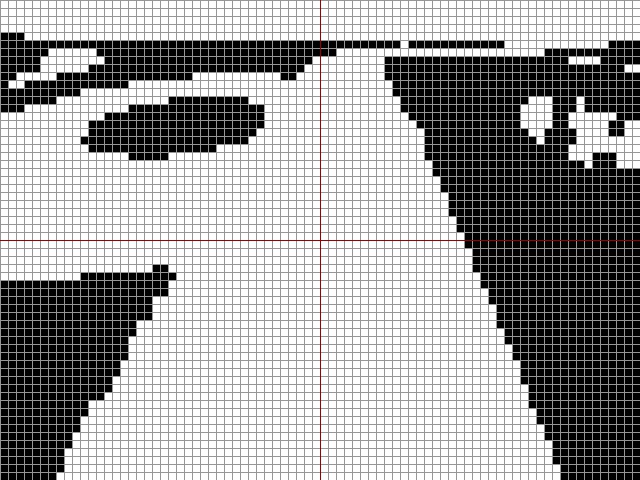
return ;

}

**附录2 特殊赛道元素中线提取思路**



以这张图为例，这是进环岛



首先你可以先找到上面的红线，根据其特点判断出左前方是环岛，当然具体怎么判断需要你们自己去做。总之你要先判断出来前面是什么赛道。

上面找到的左边界是不对的，你需要通过补线的方法找出正确的左边界

补线需要找到两个点，比如这幅图是图中标出的两个蓝色点，然后把这两个点连起来作为左边界，这样你就可以根据右边界和修正的左边界算出中线。

不同的赛道元素（十字，障碍物，起跑线等）补线方法可能不一样，你的目标就是找出正确的左右边界。循迹小助手可以帮助你们可视化自己找到的左右边界和中线。