6.3 画像処理入門

本節では、簡単な画像処理プログラムを作ってみよう。なお、先回作成したexport.c、imgutil.c、word.cを引き続き使う。

6.3.1 ビットマップ読み取りプログラム

ビットマップファイルを読み取るためのプログラムとして、import.cを用意した。このファイルには、readImage関数が実装されている。この関数の簡単な説明は以下のとおりである。

- 事前にビットマップファイルを、fopen関数を使って読み取りモードでオープンする。その際に得られるFILEポインタをこの関数は第一引数にとる。
- 第二引数はIMAGE型のポインタ[1]である。これは、ビットマップファイルを読み取って得られた情報をIMAGE型のポインタ変数として返すためのものである。
- 読み取りが成功すれば1を返し、失敗していれば0を返す。

ちなみに、この関数のプロトタイプ宣言はimport.hで行われており、その中身は以下のとおりである。readImage関数を使用するときには、このヘッダファイルをインクルードし、コンパイル時にはimport.cをリンクすべきファイルに含めること。また、この関数の実装のために、word-r.cの関数（対応するヘッダファイルはword-r.h）を利用している。必要に応じてこれもリンク（およびヘッダファイルのインクルード）すること。

```
#ifndef IMPORT_H_
#define IMPORT_H_
#include "imgutil.h"
#define MAXCOLORS 256

int readImage(FILE *fp, IMAGE *img);

#endif /* IMPORT_H_ */
```

[1] IMAGE構造体の定義は二回前の資料を見よ。

6.3.2 エッジ検出プログラム

位置(x, y)での赤色の値を$R(x, y)$と書くとすると、x方向の変化は$\frac{\partial}{\partial x}R(x, y)$、$y$方向の変化は$\frac{\partial}{\partial y}R(x, y)$として得られる。このことから、赤色の変化の大きさは、

$$\sqrt{\left(\frac{\partial}{\partial x}R(x, y)\right)^2 + \left(\frac{\partial}{\partial y}R(x, y)\right)^2} \tag{6.1}$$

で得られる。ただし、実際には赤色の関数$R(x, y)$は連続的な関数ではないため、実際にはこの式は次のように計算される[2]。

$$\delta R(x, y) = \sqrt{\left(\frac{R(x+1, y) - R(x-1, y)}{2}\right)^2 + \left(\frac{R(x, y+1) - R(x, y-1)}{2}\right)^2} \tag{6.2}$$

緑色、青色も同様である。

このことを用いると画像のエッジ（色が大きく変化する部分）を得ることができる。以上のことを利用して、画像のエッジを得るプログラムを作成せよ(edge.c)。以下の処理は引数なし、戻り値がvoidである関数`edge()`に実装せよ。

- venice.bmpを読み込みモードでオープンする。このためのFILEポインタをfpInとせよ。readImage関数を利用してこの画像ファイルのデータをIMAGE構造体imgInに読み込む[3]。

- エッジ検出用のデータを格納するためのIMAGE構造体imgOutを用意し[4]、imgInのwidth、height、depthの値をimgOutのwidth、height、depthの値にそれぞれ代入する。imgOutのpixelsに画素分の要素数をPIXEL構造体のサイズに乗じた分だけの容量をメモリ上に確保する。

- imgInのメンバーであるpixels配列の各要素とx方向・y方向にそれぞれ隣接する要素[5]が持つそれぞれのRGB値をもとに、式(6.2)の計算をR、G、Bの各色に対して行え[6]。これらの結果として得られる値

[2]いわゆる差分というやつだ。

[3]当然、使用前にはメモリ上に領域を確保すること。

[4]言うに及ばず、これもメモリ上に領域を確保しておく必要あり。

[5]getLabel関数を活用せよ。

[6]画像ファイルの上下左右の端にある画素については注意が必要。簡単のため、この課題では、そのような画素での色のの変化は0であるとしてしまおう。

は0以上255以下の範囲に入るため、imgOutのpixels配列の各要素がもつRGB値に代入する。なお、代入するときには、白黒反転させるため[7]、255から得られた値を引いたものをRGBのそれぞれの値に代入すること。

- venice-edge.bmpを書き込むためにオープンする。このためのFILEポインタをfpOutとする。saveImage関数を使って、imgOutに格納されている情報をビットマップファイルとしてこのファイルに書き込む。

- 使用したポインタを開放する。

このプログラムに対応するヘッダファイルedge.hは以下のとおり。

```
#ifndef EDGE_H_
#define EDGE_H_

void edge();

#endif /* EDGE_H_ */
```

main関数は以下のとおりedge関数を実行するだけの処理を行う。このプログラムをedgeMain.cとする。

```
#include <stdio.h>
#include "edge.h"

int main(void){
edge();
}
```

このような仕様で作られたedgeMain.cに対し、edge.c、export.c、imgutil.c、word.c、word-r.c、import.cとともにコンパイル・リンクを行う。実行ファイル名はedgeMainとする。実際にこれを実行して、生成されたビットマップファイルをペイントソフト等で開いてみよ。

[7]そのほうが結果が見やすいので。

6.3.3 太線化プログラム

エッジ検出を行ってできた画像は往々にして線が細い。そこでその線を太くしてみよう。前節同様、点(x, y)での赤色$R(x, y)$,緑色$G(x, y)$,青色$B(x, y)$を用いて説明する（簡単のため赤色で説明するが、緑色、青色も同様)。

基本的な考え方は、書き込み用の画像ファイルにおける点(x, y)での赤色の値を$R'(x, y)$とすると、いろいろな計算方法はあるが、

$$R'(x, y) = \min(R(x-1, y), R(x, y), R(x+1, y), R(x, y-1), R(x, y+1))$$

で求めることとしよう。そうすると、白以外の値がある場合、その両脇のピクセルも同じ色になる[8]。

以上のことを利用して、プログラムを作成せよ(thicken.c)。以下の処理は引数なし、戻り値がvoidである関数`thicken()`に実装せよ。

- venice-edge.bmpを読み込みモードでオープンする。このためのFILEポインタをfpInとせよ。readImage関数を利用してこの画像ファイルのデータをIMAGE構造体imgInに読み込む[9]。
- 太線化済みのデータを格納するためのIMAGE構造体imgOutを用意し[10]、imgInのwidth、height、depthの値をimgOutのwidth、height、depthの値にそれぞれ代入する。imgOutのpixelsに画素分の要素数をPIXEL構造体のサイズに乗じた分だけの容量をメモリ上に確保する。
- imgInのメンバーであるpixels配列の各要素が持つRGB値をもとに太線化を行い、imgOutのpixels配列の各要素がもつRGB値に代入する。
- venice-t.bmpを書き込むためにオープンする。このためのFILEポインタをfpOutとする。saveImage関数を使って、imgOutに格納されている情報をビットマップファイルとしてこのファイルに書き込む。
- 使用したポインタを開放する。

関連するヘッダファイルthicken.hは次の通り。

[8]通常の画像処理の扱いと違い、白黒反転（Rの値を$255 - R$としている）ことに注意。

[9]当然、使用前にはメモリ上に領域を確保すること。

[10]言うに及ばず、これもメモリ上に領域を確保しておく必要あり。

```
#ifndef THICKEN_H_
#define THICKEN_H_

void thicken();

#endif /* THICKEN_H_ */
```

main関数は次のとおりであり、ファイル名をthickenMain.cとする。

```
#include <stdio.h>
#include "thicken.h"

int main(void){
thicken();
}
```

このような仕様で作られたthickenMain.cおよびthicken.cに対し、export.c、imgutil.c、word.c、word-r.c、import.cとともにコンパイル・リンクを行う。実行ファイル名はthickenとする。実際にこれを実行して、生成されたビットマップファイルをペイントソフト等で開いてみよ。

6.3.4 レポートについて

第11回の授業にレポートを集める。必ず開始前に、提出できるよう準備をしておくこと。なお、以下について提出を行うこと。

- 第7回から第9回までで作った、export.c、imgutil.c、word.c、randomWalk.c、edge.c、thicken.cの各ファイル(今回はヘッダファイルは不要)。
- 実行ファイル simの実行結果および考察。
- 実行ファイル edgeMainの実行結果および考察。
- 実行ファイル thickenの実行結果および考察。

なお、レポート作成時に他の受講者と相談することは奨励するが、プログラムソースおよびレポートのデッドコピーの疑いがあるものは点をつけない。各自、レポートは自力で作成すること。加えて、プログラムソースや入出力のみで、苦労した点、工夫した点、考察等がないものについても減点する。必ず、これらについても記述すること。レポートはPDF形式での提出を求める。加えてソースファイルも提出すること(Scombで提出してもらう)。