

# BÀI TẬP

## Xử lý ảnh và video

**Yêu cầu:** Viết chương trình C thực hiện phân tích 5 - 10 ảnh khuôn mặt theo Edge Detection để lấy đặc trưng riêng từng ảnh. Dữ liệu đầu vào là 1 trong 5-10 ảnh khuôn mặt đó và kết quả ngõ ra là vị trí của ảnh ngõ vào trong 5-10 ảnh đó.

### 1. PHẦN XÂY DỰNG MÃ NGUỒN C

- Các thư viện sử dụng để nhập xuất, thao tác với file bộ nhớ, hàm toán học và tạo một cấu trúc dữ liệu đọc từ ảnh. Ngoài ra còn tạo hằng số là số ảnh đầu vào tối đa và giá trị Threshold (TH\_val).

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <dirent.h>
#include <conio.h>

#define NUM_IMAGES 10
#define TH_val 50
#pragma pack(push, 1)

typedef struct {
    unsigned short type;        //Loại ảnh
    unsigned int size;          //Kích thước ảnh
    unsigned short reserved1;
    unsigned short reserved2;
    unsigned int offset;        // Địa chỉ đầu
} BMPHeader;

typedef struct {
    unsigned int size;
    int w;
    int h;
    unsigned short planes;
    unsigned short bitsPerPixel;
    unsigned int compression;   // Phương pháp nén
    unsigned int imageSize;     // Kích thước hình ảnh
    int xPixelsPerMeter;
```

```
int yPixelsPerMeter;
unsigned int colorsUsed;      // Số màu
unsigned int importantColors;
} BMPInfoHeader;
#pragma pack(pop)
```

### - Hàm đọc ảnh bitmap (.bmp)

```
unsigned char* readBMPimg(const char *filename, int *w, int *h) {
    FILE *fp = fopen(filename, "rb");
    if (fp == NULL) {
        printf("Không thể mở file %s\n", filename);
        exit(1);
    }
    BMPHeader bmpHeader;
    BMPInfoHeader bmpInfoHeader;
    fread(&bmpHeader, sizeof(BMPHeader), 1, fp);
    fread(&bmpInfoHeader, sizeof(BMPInfoHeader), 1, fp);
    if (bmpHeader.type != 0x4D42) { //Mã ASCII hexa của M là 4D, của B là 42
        printf("Không phải file BMP\n");
        fclose(fp);
        exit(1);
    }
    *w = bmpInfoHeader.w;
    *h = bmpInfoHeader.h;
    unsigned char *image = (unsigned char *)malloc((*w) * (*h) * 3); // 3 bytes per
pixel (RGB)
    fseek(fp, bmpHeader.offset, SEEK_SET);
    fread(image, 3, (*w) * (*h), fp);
    fclose(fp);
    return image;
}
```

### - Hàm chuyển đổi từ ảnh màu RGB sang Gray

```
unsigned char* convertToGray(unsigned char *rgbImage, int w, int h) {
    unsigned char *grayImage = (unsigned char *)malloc(w * h * sizeof(unsigned
char));
    for (int i = 0; i < w * h; i++) {
        // Y' = 0.299 * R + 0.587 * G + 0.114 * B
        unsigned char r = rgbImage[i * 3];
        unsigned char g = rgbImage[i * 3 + 1];
        unsigned char b = rgbImage[i * 3 + 2];
        grayImage[i] = (unsigned char)(0.299 * r + 0.587 * g + 0.114 * b);
    }
    return grayImage;
}
```

- Hàm tạo lại ảnh bitmap từ dữ liệu được tạo ra với header được tạo từ các đặc tính tương ứng.

```
void writeGrayBMPimg(const char *filename, unsigned char *image, int w, int h) {
    FILE *fp = fopen(filename, "wb");
    if (fp == NULL) {
        printf("Không thể mở file %s để ghi\n", filename);
        exit(1);
    }
    BMPHeader bmpHeader;
    BMPInfoHeader bmpInfoHeader;
    bmpHeader.type = 0x4D42; // 'BM'
    bmpHeader.size = sizeof(BMPHeader) + sizeof(BMPInfoHeader) + w * h;
    bmpHeader.reserved1 = 0;
    bmpHeader.reserved2 = 0;
    bmpHeader.offset = sizeof(BMPHeader) + sizeof(BMPInfoHeader);
    bmpInfoHeader.size = sizeof(BMPInfoHeader);
    bmpInfoHeader.w = w;
    bmpInfoHeader.h = h;
    bmpInfoHeader.planes = 1;
    bmpInfoHeader.bitsPerPixel = 8;
    bmpInfoHeader.compression = 0;
    bmpInfoHeader.imageSize = w * h;
    bmpInfoHeader.xPixelsPerMeter = 0;
    bmpInfoHeader.yPixelsPerMeter = 0;
    bmpInfoHeader.colorsUsed = 256; // 2^8 bit = 256
    bmpInfoHeader.importantColors = 256;
    fwrite(&bmpHeader, sizeof(BMPHeader), 1, fp);
    fwrite(&bmpInfoHeader, sizeof(BMPInfoHeader), 1, fp);
    for (int i = 0; i < 256; i++) {
        unsigned char color[4] = {i, i, i, 0};
        fwrite(color, sizeof(color), 1, fp);
    }
    fwrite(image, 1, w * h, fp);
    fclose(fp);
}
```

- Hàm áp dụng đưa kernel vào lấy đặc trưng của ảnh là cường độ pixel (I) từ kernel(K<sub>d</sub>) và ảnh Gray(I) theo công thức dưới và dùng sao chép giá trị biên để tính phần rìa ảnh.

$$I_d(x, y) = \left| \sum_{i=-2}^2 \sum_{j=-2}^2 K_d(i, j) \cdot I(x + i, y + j) \right| \quad (1)$$

$$d \in \{H, P, V, M\}, \quad (2)$$

```
int applyKernel(unsigned char *image, int w, int h, int x, int y, int kernel[5][5]) {
    int I = 0;
```

```

for (int i = -2; i <= 2; i++) {
    for (int j = -2; j <= 2; j++) {
        //Sao chép giá trị biên để tính phần rìa ảnh
        int newY = y + i;
        int newX = x + j;
        if (newY < 0) newY = 0;
        if (newY >= h) newY = h - 1;
        if (newX < 0) newX = 0;
        if (newX >= w) newX = w - 1;
        I += abs(image[newY * w + newX] * kernel[i+2][j+2]);    //Định nghĩa
        lại vị trí từ kernel
    }
}
return I;
}

```

- Hàm khai báo giá trị các kernel lọc, thực hiện so sánh cường độ qua 4 kernel và với giá trị Threshold tại mỗi pixel và tạo các giá trị cờ cạnh theo mục 2.1 trong tài liệu.

$$F_d^*(x, y) = \begin{cases} 1, & \text{if } I_d(x, y) = \max_{d^* \in \{H, P, V, M\}} I_{d^*}(x, y) \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

$$F_d(x, y) = \begin{cases} F_d^*(x, y), & \text{if } I_d(x, y) > TH(x, y) \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

```

void Kernel(unsigned char *image, int w, int h, int *Fh, int *Fp, int *Fv, int *Fm) {
    int Kh[5][5] = {{0, 0, 0, 0, 0}, {1, 1, 1, 1, 1}, {0, 0, 0, 0, 0}, {-1, -1, -1, -1, -1}, {0, 0, 0, 0, 0}};
    int Kp[5][5] = {{0, 0, 0, 1, 0}, {0, 1, 1, 0, -1}, {0, 1, 0, -1, 0}, {1, 0, -1, 0, 0}, {0, -1, 0, 0, 0}};
    int Kv[5][5] = {{0, 1, 0, -1, 0}, {0, 1, 0, -1, 0}, {0, 1, 0, -1, 0}, {0, 1, 0, -1, 0}, {0, 1, 0, -1, 0}};
    int Km[5][5] = {{0, -1, 0, 0, 0}, {1, 0, -1, -1, 0}, {0, 1, 0, -1, 0}, {0, 1, 1, 0, -1}, {0, 0, 0, 1, 0}};
    int *Ih = (int *)calloc(w * h, sizeof(int));
    int *Ip = (int *)calloc(w * h, sizeof(int));
    int *Iv = (int *)calloc(w * h, sizeof(int));
    int *Im = (int *)calloc(w * h, sizeof(int));
    for (int y = 0; y < h ; y++) {
        for (int x = 0; x < w ; x++) {
            Ih[y * w + x] = applyKernel(image, w, h, x, y, Kh);
            Ip[y * w + x] = applyKernel(image, w, h, x, y, Kp);
            Iv[y * w + x] = applyKernel(image, w, h, x, y, Kv);

```

```

Im[y * w + x] = applyKernel(image, w, h, x, y, Km);

// Tìm max I(x,y)(Idh, Idp, Idv, Idm)
int max = Ih[y * w + x];
if (Ip[y * w + x] > max) max = Ip[y * w + x];
else if (Iv[y * w + x] > max) max = Iv[y * w + x];
else if (Im[y * w + x] > max) max = Im[y * w + x];
// Gộp công thức 3 và 7 so sánh và tạo direction map Fh, Fp, Fv, Fm
if (Ih[y * w + x] == max && Ih[y * w + x] > TH_val) {
    Fh[y * w + x] = 1; Fp[y * w + x] = 0;
    Fv[y * w + x] = 0; Fm[y * w + x] = 0;
} else if (Ip[y * w + x] == max && Ip[y * w + x] > TH_val) {
    Fp[y * w + x] = 1; Fh[y * w + x] = 0;
    Fv[y * w + x] = 0; Fm[y * w + x] = 0;
} else if (Iv[y * w + x] == max && Iv[y * w + x] > TH_val) {
    Fv[y * w + x] = 1; Fh[y * w + x] = 0;
    Fp[y * w + x] = 0; Fm[y * w + x] = 0;
} else if (Im[y * w + x] == max && Im[y * w + x] > TH_val) {
    Fm[y * w + x] = 1; Fh[y * w + x] = 0;
    Fp[y * w + x] = 0; Fv[y * w + x] = 0;
}
}
}
}

```

Với giá trị của 4 kernel theo Fig. 3

Horizontal	+45 degree	Vertical	-45 degree
0 0 0 0 0	0 0 0 1 0	0 1 0 -1 0	0 -1 0 0 0
1 1 1 1 1	0 1 1 0 -1	0 1 0 -1 0	1 0 -1 -1 0
0 0 0 0 0	0 1 0 -1 0	0 1 0 -1 0	0 1 0 -1 0
-1 -1 -1 -1 -1	1 0 -1 -1 0	0 1 0 -1 0	0 1 1 0 -1
0 0 0 0 0	0 -1 0 0 0	0 1 0 -1 0	0 0 0 1 0
<b>K<sub>H</sub></b>	<b>K<sub>P</sub></b>	<b>K<sub>V</sub></b>	<b>K<sub>M</sub></b>

- Hàm xây dựng và thực hiện thuật toán APED (Averaged Principal-Edge Distribution) từ giá trị các cờ cạnh tạo các bản đồ tính chất cạnh hướng H, P, V, M.

$$\begin{aligned}
 \mathbf{H}(a + 4b) &= \sum_{i=0}^{15} \sum_{j=0}^{15} F_H(16a + i, 16b + j) \\
 \mathbf{P}(a + 4b) &= \sum_{i=0}^{15} \sum_{j=0}^{15} F_P(16a + i, 16b + j) \\
 \mathbf{V}(a + 4b) &= \sum_{i=0}^{15} \sum_{j=0}^{15} F_V(16a + i, 16b + j) \\
 \mathbf{M}(a + 4b) &= \sum_{i=0}^{15} \sum_{j=0}^{15} F_M(16a + i, 16b + j) \\
 a &= 0, 1, 2, 3, \quad b = 0, 1, 2, 3.
 \end{aligned} \tag{8}$$

```

void APEDfunction(int *Fh, int *Fp, int *Fv, int *Fm, int *H, int *P, int *V, int
*M){
    for (int a = 0; a < 4; a++) {
        for (int b = 0; b < 4; b++) {
            int Htmp = 0, Ptmp = 0, Vtmp = 0, Mtmp = 0 ;
            for (int i = 0; i < 16; i++) {
                for (int j = 0; j < 16; j++) {
                    Htmp += Fh[(16 * b + j) + (16 * a + i)] ;
                    Ptmp += Fp[(16 * b + j) + (16 * a + i)] ;
                    Vtmp += Fv[(16 * b + j) + (16 * a + i)] ;
                    Mtmp += Fm[(16 * b + j) + (16 * a + i)] ;
                }
            }
            H[a + 4 * b] = Htmp ; P[a + 4 * b] = Ptmp ;
            V[a + 4 * b] = Vtmp ; M[a + 4 * b] = Mtmp ;
        }
    }
}

```

- Hàm main bao gồm đọc đường dẫn tới thư mục chứa folder các hình ảnh cần nhận diện và ảnh cần so sánh, chuyển đổi sang Gray để đưa qua kernel, lưu giữ giá trị các cờ cạnh hướng và bản đồ tính chất từng ảnh.
- Dễ dàng và liên tục kiểm tra

```

int main() {
    char directory[100];
    printf("Nhập đường dẫn đến thư mục chứa 10 ảnh BMP: ");
    scanf("%s", directory);

    char filenames[NUM_IMAGES][150];
}

```

```

struct dirent *de;
DIR *dr = opendir(directory);
int imageCount = 0;

if (dr == NULL) {
    printf("Không thể mở thư mục: %s\n", directory);
    return 1;
}

while ((de = readdir(dr)) != NULL && imageCount < NUM_IMAGES) {
    if (strstr(de->d_name, ".bmp") != NULL) {
        snprintf(filenamees[imageCount], sizeof(filenamees[imageCount]), "%s/%s",
directory, de->d_name);
        imageCount++;
    }
}

closedir(dr);
printf("Số lượng ảnh BMP tìm thấy: %d\n", imageCount);

int H[NUM_IMAGES][16], P[NUM_IMAGES][16], V[NUM_IMAGES][16], M[NUM_IMAGES][16];
int *Fh[NUM_IMAGES], *Fp[NUM_IMAGES], *Fv[NUM_IMAGES], *Fm[NUM_IMAGES];
unsigned char *rgbImage[NUM_IMAGES], *grayImage[NUM_IMAGES];
int widths[NUM_IMAGES], heights[NUM_IMAGES];

for (int i = 0; i < imageCount; i++) {
    rgbImage[i] = readBMPimg(filenamees[i], &widths[i], &heights[i]);
    grayImage[i] = convertToGray(rgbImage[i], widths[i], heights[i]);
    Fh[i] = (int *)calloc(widths[i] * heights[i], sizeof(int));
    Fp[i] = (int *)calloc(widths[i] * heights[i], sizeof(int));
    Fv[i] = (int *)calloc(widths[i] * heights[i], sizeof(int));
    Fm[i] = (int *)calloc(widths[i] * heights[i], sizeof(int));

    Kernel(grayImage[i], widths[i], heights[i], Fh[i], Fp[i], Fv[i], Fm[i]);
    APEDfunction(Fh[i], Fp[i], Fv[i], Fm[i], H[i], P[i], V[i], M[i]);
    free(grayImage[i]);
    free(rgbImage[i]);
}

printf("Nhấn phím bất kỳ để bắt đầu so sánh ảnh test (nhấn 'Esc' để thoát):\n");

while (1) {
    if (_kbhit()) {
        char ch = _getch();
        if (ch == 27) {
            break;
        }
    }
}

```

```
char inputFilename[150];
printf("Nhập tên file ảnh cần so sánh (bao gồm đường dẫn): ");
scanf("%s", inputFilename);

int testWidth, testHeight;
unsigned char *testImg = readBMPimg(inputFilename, &testWidth,
&testHeight);
unsigned char *testgrayImg = convertToGray(testImg, testWidth,
testHeight);

int *inputFh = (int *)calloc(testWidth * testHeight, sizeof(int));
int *inputFp = (int *)calloc(testWidth * testHeight, sizeof(int));
int *inputFv = (int *)calloc(testWidth * testHeight, sizeof(int));
int *inputFm = (int *)calloc(testWidth * testHeight, sizeof(int));

Kernel(testgrayImg, testWidth, testHeight, inputFh, inputFp, inputFv,
inputFm);

int testH[16], testP[16], testV[16], testM[16];
APEDfunction(inputFh, inputFp, inputFv, inputFm, testH, testP, testV,
testM);

int matchIndex = -1;
int mindiff = 1e4;
for (int i = 0; i < imageCount; i++) {
    int currentDiff = 0; // Khởi tạo độ khác biệt cho ảnh thứ i
    for (int j = 0; j < 16; j++) {
        currentDiff += abs(testH[j] - H[i][j]);
        currentDiff += abs(testP[j] - P[i][j]);
        currentDiff += abs(testV[j] - V[i][j]);
        currentDiff += abs(testM[j] - M[i][j]);
    }
    if (currentDiff < mindiff) {
        mindiff = currentDiff;
        matchIndex = i;
    }
}

if (matchIndex != -1) {
    printf("Ảnh được so sánh là gần nhất với: %s\n",
filenames[matchIndex]);
} else {
    printf("Không tìm thấy ảnh phù hợp.\n");
}

free(testgrayImg);
free(testImg);
```



```
        free(inputFh);
        free(inputFp);
        free(inputFv);
        free(inputFm);
    }
}
for (int i = 0; i < imageCount; i++) {
    free(Fh[i]);
    free(Fp[i]);
    free(Fv[i]);
    free(Fm[i]);
}
return 0;
}
```

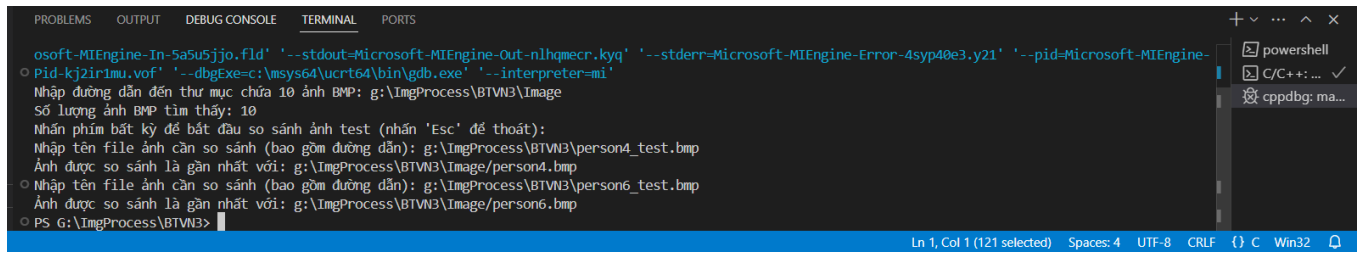
Trong đó, hàm này thực hiện khởi tạo biến vị trí ảnh trùng (matchIndex) và giá trị mindiff để so sánh. Ở đây so sánh bằng cách lấy giá trị các bản đồ tính chất cạnh hướng của ảnh cần so sánh trừ cho giá trị tương ứng từng ảnh, tìm giá trị khác biệt nhỏ nhất và trả về vị trí có giá trị đó.

```
int matchIndex = -1;
int mindiff = 1e4;
for (int i = 0; i < imageCount; i++) {
    int currentDiff = 0; // Khởi tạo độ khác biệt cho ảnh thứ i
    for (int j = 0; j < 16; j++) {
        currentDiff += abs(testH[j] - H[i][j]);
        currentDiff += abs(testP[j] - P[i][j]);
        currentDiff += abs(testV[j] - V[i][j]);
        currentDiff += abs(testM[j] - M[i][j]);
    }
    if (currentDiff < mindiff) {
        mindiff = currentDiff;
        matchIndex = i;
    }
}

if (matchIndex != -1) {
    printf("Ảnh được so sánh là gần nhất với: %s\n", filenames[matchIndex]);
} else {
    printf("Không tìm thấy ảnh phù hợp.\n");
}
```

## 2. KẾT QUẢ SO SÁNH

## Xử lý ảnh và video



```
o soft-MIEngine-In-5aSu5jjo.fld' '--stdout=Microsoft-MIEngine-Out-nlhqmecr.kyq' '--stderr=Microsoft-MIEngine-Error-4syp40e3.y21' '--pid=Microsoft-MIEngine-
pid-kj2ir1mu.vof' '--dbgExe=c:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Nhập đường dẫn đến thư mục chứa 10 ảnh BMP: g:\ImgProcess\BTVN3\Image
Số lượng ảnh BMP tìm thấy: 10
Nhấn phím bất kỳ để bắt đầu so sánh ảnh test (nhấn 'Esc' để thoát):
Nhập tên file ảnh cần so sánh (bao gồm đường dẫn): g:\ImgProcess\BTVN3\person4_test.bmp
Ảnh được so sánh là gần nhất với: g:\ImgProcess\BTVN3\Image\person4.bmp
Nhập tên file ảnh cần so sánh (bao gồm đường dẫn): g:\ImgProcess\BTVN3\person6_test.bmp
Ảnh được so sánh là gần nhất với: g:\ImgProcess\BTVN3\Image\person6.bmp
PS G:\ImgProcess\BTVN3>
```

Link folder chứa source code và các tài nguyên hình ảnh:

[https://studenthcmusedu-my.sharepoint.com/:f:/g/personal/20200331\\_student\\_hcmus\\_edu\\_vn/EmoVGZGcL85Ip-eWBnLyGlgB5xJSSN-N7r\\_4BV1rNha2xg?e=o08zqr](https://studenthcmusedu-my.sharepoint.com/:f:/g/personal/20200331_student_hcmus_edu_vn/EmoVGZGcL85Ip-eWBnLyGlgB5xJSSN-N7r_4BV1rNha2xg?e=o08zqr)