ارشیا عطایی نایینی ۸۱۰۱۰۰۲۵۲

گزارش تمرین کامپیوتری سوم

بخش اول)

۱-۱) MapSet به صورت زیر است:

a	b	c	d	e	f	g	h	i	j	k	l	m
00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100

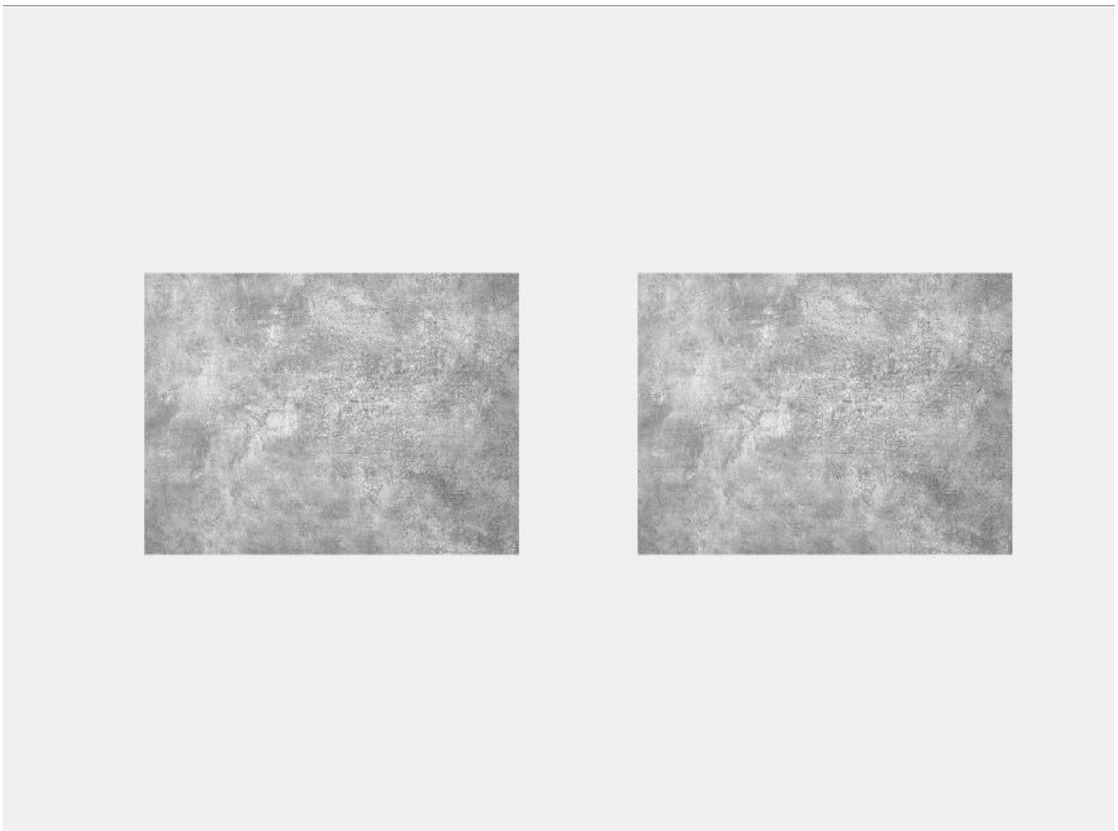
14	15	16	17	18	19	20	21	22	23	24	25	26
n	o	p	q	r	s	t	u	v	w	x	y	z
01101	01110	01111	10000	10001	10010	10011	10100	10101	10110	10111	11000	11001

27	28	29	30	31	32
	.	,	!	"	;
11010	11011	11100	11101	11110	11111

۱-۲) به این‌صورت عمل می‌کنیم که ابتدا پیام را به رشته باینری مورد نظر تبدیل می‌کنیم و از پیکسل بالا راست شروع کرده و بیت کم ارزش آن‌را برابر بیت مورد نظر رشته باینری می‌گذاریم.

```
function [pic] = coding(code, graypic, Mapset)
    [n, m] = size(graypic);
    pnt=0;
    pic = zeros(n, m);
    [aa, code_len] = size(code);
    pic = graypic;
    if (n * m < code_len * 5)
        fprintf("The size doesn't fit\n");
        return
    end
    for i=1:code_len
        koj = 0;
        for j=1:32
            if (Mapset{1, j} == code(i))
                koj = j;
                break;
            end
        end
        code_bin = Mapset{2, koj};
        l = (i - 1) * 5;
        for j=1:5
            r = floor(l / m) + 1;
            c = mod(l, m) + 1;
            if (code_bin(j) == '1')
                pic(r, c) = bitor(1, pic(r, c));
            end
            if (code_bin(j) == '0')
                pic(r, c) = bitand(254, pic(r, c));
            end
            l = l + 1;
        end
    end
end
```

۱-۳) تصویر در صفحه بعد



۱-۴) مطابق بخشای قبلی این بار برعکس عمل کرده و انقدر کم ارزش ترین بیت‌ها را ۵ تا ۵ تا می‌خوانیم تا به ۱۱۱۱۱ برسیم و خاتمه بدهیم.

```
function [res] = coding(code_pic, Mapset)
    [n, m] = size(code_pic);
    res = '';
    for i=0:5:n * m - 1
        a = '';
        for j=0:4
            r = floor((i + j) / m) + 1;
            c = mod((i + j), m) + 1;
            if (mod(code_pic(r,c), 2) == 1)
                a = [a, '1'];
            end
            if (mod(code_pic(r, c), 2) == 0
                a = [a, '0'];
            end
        end
        for j=1:32
            if (Mapset{2, j} == a)
                res = [res, Mapset{1, j}];
                break;
            end
        end
        if (a == '11111')
            break;
        end
    end
    fprintf("code: %s\n", res);
end
```

خروجی این کد برای عکس رمزنگاری شده:

```
code: signal;
```

۱-۵) خیر زیرا با برای ما کم ارزش ترین بیت اهمیت دارد با کوچکترین نویز ممکن است کم ارزش ترین بیت به هر پیکسل به سادگی عوض شود و رمز به طور کامل تغییر کند.

۱-۶) ایده‌ای که می‌توان استفاده کرد این است که ابتدا اگر قرار باشد پیامی ارسال شود باید طول زیادی داشته باشد، سپس کاراکترهای رشته باینری حاصل از قرار دادن کم ارزش‌ترین بیت هر پیکسل را نگاه کرده و با الگوریتم‌های حدس و تطابق آن با متن می‌سنجیم که آیا این می‌تواند یک متن باشد یا خیر. به عنوان مثال چند درصد آن حروف صدادار هستند یا چند درصد حروف پر کاربرد الفبای انگلیسی.

بخش دوم)

۲-۱) برای این بخش همان‌طور که گفته شد یک سینال می‌سازیم که به ازای Ton ثانیه فرکانس آن عدد را دارا باشد و بعد از آن به ازای Toff ثانیه صفر باشد.

```
clear;
fr = [697 770 852 941];
fc = [1209 1336 1477];
fs = 8000;
Ts = 1/fs;
Ton = 0.1;
Toff = 0.1;
number = '43218765';
[aa, number_len] = size(number);
t = 0:Ts:Ton - Ts;
r = [4 1 1 1 2 2 2 3 3 3];
c = [2 1 2 3 1 2 3 1 2 3];
t_len = length(t);
res = zeros(1, t_len * (number_len * 2 - 1));
for i=1:number_len
    l = (i - 1) * t_len * 2;
    l = l + 1;
    rr = l + t_len - 1;
    n = double(number(i)) - double('0');
    y1 = sin(2 * pi * fr(r(n + 1)) * t);
    y2 = sin(2 * pi * fc(c(n + 1)) * t);
    y = (y1 + y2) / 2;
    res(l:rr) = y;
end
sound(res, fs);
```

۲-۲) دقیقا عکس قبلی را انجام می‌دهیم و با correlation گیری میفهمیم کدام صدا بیشترین تطابق را دارد.

```
[a, fs] = audioread('a.wav');
a = a';
a_len = length(a);
res = '';
for i=1:t_len * 2:a_len
    x = a(i:i + t_len - 1);
    mx = zeros(1, 10);
    for j=0:9
        rr = r(j + 1);
        cc = c(j + 1);
        y1 = sin(2 * pi * fr(rr) * t);
        y2 = sin(2 * pi * fc(cc) * t);
        y = (y1 + y2) / 2;
        mx(j + 1) = corr2(x', y');
    end
    [b, koj] = max(mx);
    res = [res, char(double('0') + koj - 1)];
end
fprintf("the phonenumber is: %s\n", res);
```

```
the phonenumber is: 810198
```

بخش سوم)

استراتژی کلی به اینصورت است که ابتدا عکس را خاکستری کرده و سپس دو تصویر را نرمالایز کرده(پس از جداسازی به اندازه IC) و با حرکت دادن تصویر IC بر روی تصویر مدار و correlation گیری، نقاطی که دارای correlation بیشتر از threshold شدن را به عنوان IC شناسایی کرده و آن‌ها را رسم می‌کنیم:

```
function [] = ICrecognition(picture, IC)
    old_pic = picture;
    picture = rgb2gray(picture);
    old_picg = picture;
    picture = double(picture);
    mean_val = mean(picture(:));
    std_val = std(picture(:));
    picture = (picture - mean_val) / std_val;
    imshow(picture);
    picture = old_picg;
    IC = rgb2gray(IC);
    hold on
    figure
    imshow(IC);
    IC = double(IC);
    mean_val = mean(IC(:));
    std_val = std(IC(:));
    IC = (IC - mean_val) / std_val;
    imshow(IC);
    [IC_n, IC_m] = size(IC);
    [pic_n, pic_m] = size(picture);
```

در این بخش ابتدا دو عکس را خاکستری کرده و IC را نرمالایز می‌کند.

```
    threshold = 0.5;
    IC_rot = imrotate(IC, 180);
    figure
    imshow(old_pic);
    hold on;
    for i=1:pic_n - IC_n + 1
        for j = 1:pic_m - IC_m + 1
            y = picture(i:i + IC_n - 1, j:j + IC_m - 1);
            if (mycorrcoef(IC, y) > threshold || mycorrcoef(IC_rot, y) > threshold)
                rectangle('Position', [j, i, IC_m, IC_n], 'EdgeColor', 'b', 'LineWidth', 2);
            end
        end
    end
    hold on;
end
```

در این بخش عکس IC را برروی عکس مدار حرکت می‌دهیم و correlation ceoff را انجام می‌دهیم و بیشترین شباهت‌ها را پیدا می‌کنیم و با تابع rectangle آن‌ها را رسم می‌کنیم.

```
function [res] = mycorrcoef(x, y)
    y = double(y);
    mean_val = mean(y(:));
    std_val = std(y(:));
    y = (y - mean_val) / std_val;
    summ = x .* y;
    sumx = x .* x;
    sumy = y .* y;
    summ = sum(summ(:));
    sumx = sum(sumx(:));
    sumy = sum(sumy(:));
    res = summ / sqrt(sumx * sumy);
end
```

در انتها تابع mycorrcoef که حاصل correlation را حساب می‌کند(در ابتدا تصویر را نرمالایز می‌کند)