گزارش‌کار پروژه سوم سیگنالها و سیستم‌ها

فاطمه کرمی محمدی | ۸۱۰۱۰۰۲۵۶

بخش اول:

۱-۱) با استفاده از کد زیر ابتدا کاراکترها در یک ارایه ریخته شده و سپس در یک حلقه به همراه کد اختصاصی‌شان وارد cell ۲ در ۳۲ می‌شوند.

```
characters = ['a':'z', ' ', '.', ',', '!', '"', ';'];
char_cell = cell(2, 32);
for i = 0 : length(characters) - 1
    char_cell{1, i + 1} = char(characters(i + 1));
    char_cell{2, i + 1} = char(dec2bin(i, 5));
end
```

۱-۲) تابع coding به شکل زیر است:

```
function result_pic = coding(message, picture, mapset)
    len = length(message);
    message = char(message);
    coded_message = zeros(1, len * 5);
    bin = "";
    for i = 1 : len
        lett = message(i);
        for j = 1 : 32
            if char(lett) == char(mapset(1, j))
                bin = char(mapset(2, j));
            end
        end
        k = i - 1;
        for t = 0 : 4
            coded_message(5 * k + t + 1) = str2double(bin(t + 1));
        end
    end
    pic_size = size(picture);
    rows = pic_size(1);
    cols = pic_size(2);
    result_pic = picture;
    if cols * rows < 5 * len
        fprintf("picture size is small for this input.\n");
        return;
    end
    k = 1;
    for r = 1 : rows
        flag = 0;
        for c = 1 : cols
            if k > length(coded_message)
                flag = 1;
                break;
            end
            if coded_message(k) == 0
                result_pic(r, c) = result_pic(r, c) - mod(result_pic(r, c), 2);
            else
                added = 1 - mod(result_pic(r, c), 2);
                result_pic(r, c) = result_pic(r, c) + added;
            end
            k = k + 1;
        end
        if flag == 1
            break;
        end
    end
end
```

این تابع یک پیام، یک عکس خاکستری و mapset ساخته شده در بالا را بعنوان ورودی می‌گیرد و با استفاده از mapset پیام ورودی را به کد باینری تبدیل می‌کند. سپس کد باینری را از سمت چپ به ترتیب در کم اهمیت ترین بیت پیکسل‌های عکس ورودی (با شروع از سطر یک و ستون یک و حرکت سطری) می‌گذارد (زوج یا فرد

بودن مقدار یک پیکسل، مقدار کم اهمیت ترین بیت آن را مشخص می‌کند و در این کد هم از همین خاصیت استفاده شده است). در نهایت با اتمام کد باینری شده عکس نهایی را باز می‌گرداند. همچنین اگر سایز عکس ورودی کوچکتر از مقدار مناسب برای جاسازی پیام باشد این تابع به کاربر خطای مناسب را می‌دهد.

۳-۱) خروجی این تابع به ازای ورودی signal; در مقایسه با عکس اصلی به این شکل خواهد بود:

همانطور که دیده می‌شود تغییراتی که تابع به عکس داده در تصویر مشخص نیست چون در این تغییرات فقط کم اهمیت ترین بیت در تعدادی از پیکسل‌های عکس تغییر کرده است که باعث تغییر بسیار کمی در رنگ آن پیکسل‌ها می‌شود. در نتیجه این تغییرات با چشم قابل دیدن نیست.

۴-۱) تابع decoding عکس کدگذاری شده و mapset را بعنوان ورودی می‌گیرد، سپس از پیکسل‌های ابتدایی عکس ورودی، پیام کدگذاری شده باینری را استخراج می‌کند. وقتی به ; می‌رسد استخراج را تمام می‌کند و با استفاده از mapset پیام را دیکود کرده و در نهایت رشته دیکود شده را خروجی می‌دهد. پیاده سازی این تابع به این شکل است:

```
function result = decoding(picture, mapset)
    result_message = '';
    pic_size = size(picture);
    rows = pic_size(1);
    cols = pic_size(2);
    seq1 = 0;
    for r = 1 : rows
        flag = 0;
        for c = 1 : cols
            here = (r-1) * cols + c;
            if seq1 == 5
                flag = 1;
                break;
            end
            if mod(here, 5) == 1
                seq1 = 0;
            end
            if mod(picture(r, c), 2) == 0
                result_message = [result_message, '0'];
                seq1 = 0;
            else
                result_message = [result_message, '1'];
                seq1 = seq1 + 1;
            end
        end
        if flag == 1
            break;
        end
    end
    len = length(result_message);
    result = '';
    for i = 1 : len/5
        bin = result_message((i-1)*5+1:i*5);
        for j = 1 : 32
            if char(bin) == char(mapset(2, j))
                result = [result, char(mapset(1, j))];
                break;
            end
        end
    end
end
```

خروجی این تابع با ورودی عکس کدگذاری شده توسط رشته ;signal به این شکل خواهد بود:

```
decoded_message =

    'signal;'
```

۵-۱) در صورت اضافه شدن نویز به تصویر پس از رمزگذاری آن دیگر قادر به رمزگشایی تصویر نخواهیم بود چون کدگذاری در تصویر در کم اهمیت ترین بیت هر پیکسل انجام شده است و این بیت با کوچکترین تغییر در مقدار آن پیکسل تغییر می‌کند. اضافه کردن نویز به تصویر هم باعث ایجاد این تغییرات کوچک در مقدار پیکسل‌های تصویر می‌شود پس می‌تواند کاملا رمزگذاری داخل تصویر را خراب کند و آن را تغییر دهد.

۶-۱) در متون انگلیسی هر حرف درصد تکرار مختص خود را دارد. برای مثال درصد تکرار تعدادی از حروف انگلیسی در متون واقعی در شکل زیر آمده است:

Letter	Frequency
E	12.02
T	9.10
A	8.12
O	7.68
I	7.31

برای تشخیص اینکه بخشی از تصویر رمزگذاری شده است یا خیر می‌تواند ابتدا آن بخش را با استفاده از mapset دیکود کرد، سپس درصد تکرار هر حرف در آن را با درصد تکرار آن حرف در متون واقعی انگلیسی مقایسه کرد. هرچقدر این درصد شبیه‌تر باشد احتمال معنی دار بودن و کدگذاری شده بودن آن بخش از تصویر بیشتر است. نکته قابل توجه این است که این روش در متون استخراج شده از عکس با طول بیشتر جواب مناسب‌تری می‌دهد و اگر تعداد حروف استخراج شده از عکس کم باشد احتمال خطای این روش بالاست.

بخش دوم:

۱-۲) این کار با استفاده از کد زیر قابل انجام است:

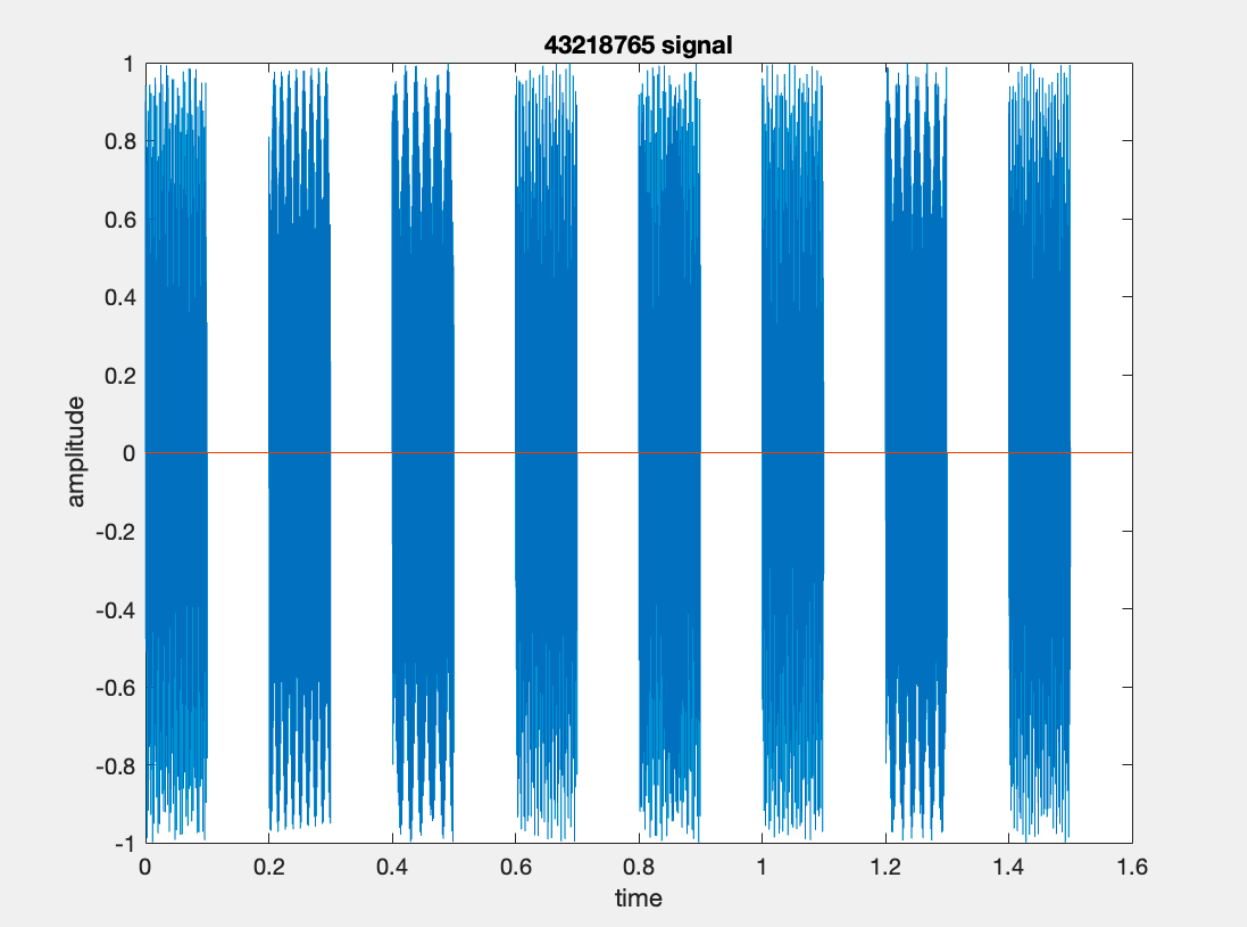
```
fs = 8000;
Ts = 1/fs;
Ton = 0.1;
Toff = 0.1;
t = 0 : Ts : 8*(Ton + Toff);
result = zeros(length(t));
on_len = Ton * fs;
off_len = Toff * fs;
T = on_len + off_len;
result(1       :       on_len + 1) = build_sound(4);
result(T + 1   : 1*T + on_len + 1) = build_sound(3);
result(2*T + 1 : 2*T + on_len + 1) = build_sound(2);
result(3*T + 1 : 3*T + on_len + 1) = build_sound(1);
result(4*T + 1 : 4*T + on_len + 1) = build_sound(8);
result(5*T + 1 : 5*T + on_len + 1) = build_sound(7);
result(6*T + 1 : 6*T + on_len + 1) = build_sound(6);
result(7*T + 1 : 7*T + on_len + 1) = build_sound(5);
result = result(:, 1:2);
sound(result, fs)
file_name = 'y.wav';
audiowrite(file_name, result, fs);
```

که در آن تابع build_sound به این شکل است:

```
function res = build_sound(n)
    fr = [697, 770, 852, 941];
    fc = [1209, 1336, 1477];
    fs = 8000;
    Ts = 1/fs;
    Ton = 0.1;
    T = 0:Ts:Ton;
    x = (n-1)/3;
    r = floor(x) + 1;
    c = mod(n-1, 3) + 1;
    if n == 0
        r = 4;
        c = 2;
    end
    y1 = sin(2 * pi * fr(r) * T);
    y2 = sin(2 * pi * fc(c) * T);
    res = (y1 + y2) / 2;
end
```

این تابع با در اختیار داشتن فرکانس‌های مخصوص هر سطر و هر ستون با گرفتن ورودی n، سیگنال متناظر با کلید n ام را با طول Ton تولید می‌کند.

در کد بالا با استفاده از این تابع سیگنال متناظر با شماره ۴۳۲۱۸۷۶۵ تولید شده و با فاصله‌های Toff در فایل y.wav ریخته می‌شود. نمودار این سیگنال به این شکل است:



۲-۲) در این بخش با استفاده از دستور audioread فایل a.wav خوانده می‌شود. سپس در یک حلقه بازه‌های Ton از آن استخراج شده و correlation آن بازه با سیگنال ناشی از فشردن اعداد ۰ تا ۹ محاسبه می‌شود. عددی که بیشترین correlation را با بازه داشته باشد بعنوان کلیدی در نظر گرفته می‌شود که سیگنال آن بازه را تولید کرده است. این عدد در آرایه number ریخته شده و در نهایت پس از اتمام حلقه عدد شماره‌گیری شده در آرایه خواهد بود. کد این بخش به این صورت است:

```
[a, fs] = audioread('a.wav');

Ts = 1/fs;
Ton = 0.1;
Toff = 0.1;
t = 0 : Ts : length(a)/fs - Ts;

on_len = Ton * fs;
off_len = Toff * fs;
T = on_len + off_len;

number = zeros(1, 6);

for i = 1 : 6
    key = a((i-1)*T+1 : (i-1)*T + on_len + 1);
    match = 0;
    corr_match = 0;
    for j = 0 : 9
        s = build_sound(j);
        if corr2(key', s) > corr_match
            corr_match = corr2(key', s);
            match = j;
        end
    end
    number(i) = match;
end

fprintf("%d", number)
fprintf('\n');
```

برای تست اینکه این کد درست کار می‌کند ابتدا کد روی سیگنال تولید شده در بخش ۲-۱ تست می‌شود. خروجی به این صورت است:

```
43218765
>>
```

در نتیجه این کد درست کار می‌کند. رمز کردن این کد روی فایل a.wav این خروجی را خواهد داد:

```
810198
>>
```

پس سیگنال a.wav منتج از فشردن کلیدهای ۸۱۰۱۹۸ بوده است.

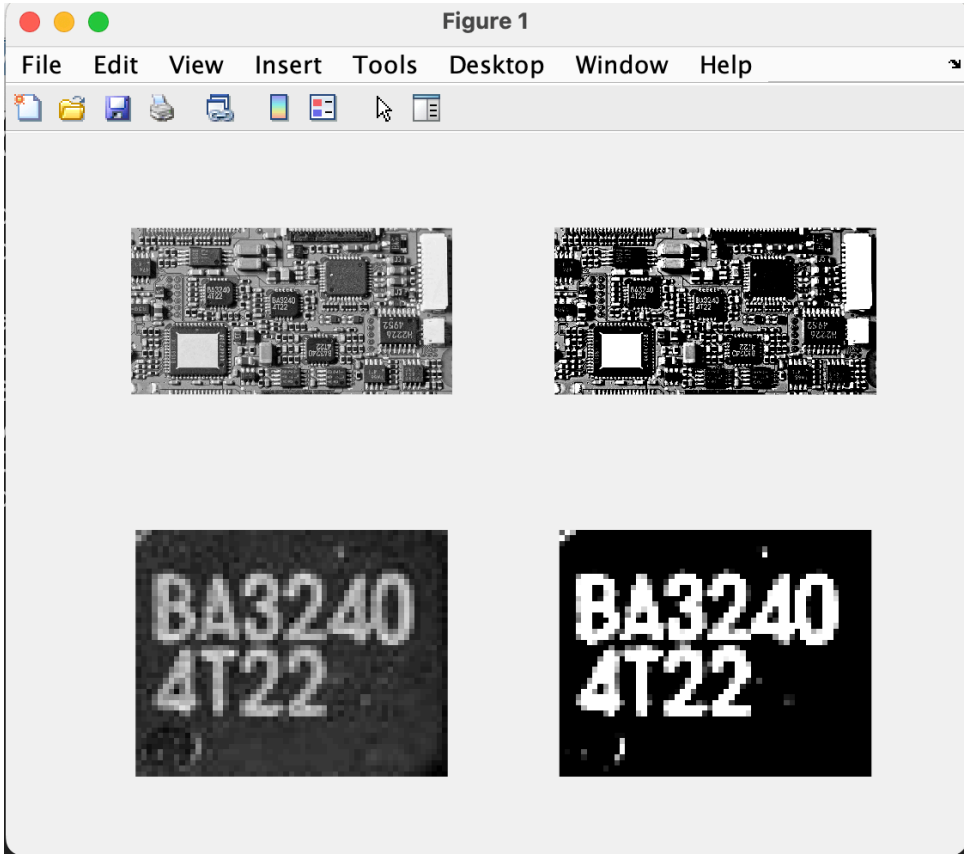
بخش سوم:

در این بخش ابتدا دو تصویر PCB و IC از کاربر گرفته شده با استفاده از تابع rgb2gray خاکستری می‌شوند. سپس به ازای تمام حالت‌هایی که تصویر IC (و دوران یافته آن به اندازه ۱۸۰ درجه) می‌تواند روی تصویر PCB قرار بگیرد، correlation coefficient برای تصویر خاکستری IC و بخشی از تصویر خاکستری PCB که تصویر IC روی آن قرار گرفته محاسبه می‌شود. اگر این correlation از threshhold مشخص شده بیشتر باشد یعنی در آن بخش از تصویر PCB یک IC وجود دارد و توسط تابع rectangle دور آن مستطیل کشیده می‌شود.

کد این بخش به این صورت است:

```
[file, path] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'choose PCB');
s = [path, file];
PCB = imread(s);
[file, path] = uigetfile({'*.jpg;*.bmp;*.png;*.tif'}, 'choose IC');
s = [path, file];
IC = imread(s);

grayPCB = rgb2gray(PCB);
grayIC = rgb2gray(IC);

grayIC_double = double(grayIC);
meanIC = mean(grayIC_double(:));
stdIC = std(grayIC_double(:));
normIC = (grayIC_double - meanIC) / stdIC;

thresh = 0.5;
[ICr, ICc] = size(normIC);
[PCBr, PCBc] = size(grayPCB);
rotIC = imrotate(normIC, 180);
figure;
imshow(PCB);
hold on;
for r = 1 : PCBr-ICr
    for c = 1 : PCBc-ICc
        part = grayPCB(r : r+ICr-1, c : c+ICc-1);
        part_double = double(part);

        meanpart = mean(part_double(:));
        stdpart = std(part_double(:));
        normpart = (part_double - meanpart) / stdpart;

        if (corrcoef(normIC, normpart) > thresh)
            pos = [c, r, ICc, ICr];
            rectangle('Position', pos, 'EdgeColor', 'r', 'LineWidth', 2);
        end
        if (corrcoeff(rotIC, normpart) > thresh)
            pos = [c, r, ICc, ICr];
            rectangle('Position', pos, 'EdgeColor', 'r', 'LineWidth', 2);
        end
    end
end
```

برای دقیق‌تر شدن نتیجه correlation coefficient قبل از پاس دادن دو تصویر به این تابع باید این دو تصویر normalize شوند، یعنی میانگین مقدار پیکسل‌ها از آنها کم شده و تقسیم بر انحراف معیار مقادیر پیکسل‌ها شوند. با این کار میانگین مقدار پیکسل‌ها در عکس ۰ و انحراف معیار آنها ۱ می‌شود. تصویر حاصل از نرمالایز کردن تصویر IC و کل تصویر PCB در شکل زیر نمایش داده شده است:



اشکال سمت چپ خاکستری شده عکس‌های PCB و IC هستند و اشکال سمت راست نرمالیزه شده اشکال سمت چپ هستند.

همچنین تابع correlation coefficient به این صورت عمل می‌کند:

```
function result = corrcoeff(x, y)
    sorat = x .* y;
    sorat = sum(sorat(:));
    sumx2 = x .* x;
    sumx2 = sum(sumx2(:));
    sumy2 = y .* y;
    sumy2 = sum(sumy2(:));
    makhraj = sqrt(sumx2 * sumy2);
    result = sorat / makhraj;
end
```

که معادل محاسبه عبارت زیر برای عکس‌های دو بعدی است:

$$Correlation\ Coeff(x,y) = \frac{\sum_{n=1}^{L} x[n]y[n]}{\sqrt{(\sum_{n=1}^{L} x^2[n]) \times (\sum_{k=1}^{L} y^2[k])}}$$

خروجی نهایی این بخش با threshold = 0.5 به این شکل خواهد بود: