به نام خدا

سید محمد حسین مظهری 810101520 پروژه 3 درس سیگنال و سیستم

پارسا دقیق 810101419

بخش اول :

تمرین 1-1)

اسکریپت ایجاد کردن mapset :

```
% 1- Creating the mapset
Nch = 32 ;
mapset = cell(2,Nch) ;
Alphabet = 'abcdefghijklmnopqrstuvwxyz .,!";' ;
for i = 1:Nch
    mapset{1,i} = Alphabet(i) ;
    mapset{2,i} = dec2bin(i-1,5) ;
end
```

ابتدا یک سلول 2x32 می سازیم و کاراکتر ها را در ردیف اول و مقدار باینری متناظر را در ردیف دوم قرار میدهیم.

تمرین 1-2)

استراتژی انتخاب بلوک، کنتراست رنگی است . ابتدا کل عکس را به بلوک های 5x5 تقسیم می کنیم . سپس کوچک ترین بیت های هر بلوک را صفر کرده و کنتراست را حساب میکنیم و بلوک ها را به ترتیب کنتراست بیشتر در یک آرایه ذخیره می کنیم .و نقطه ی بالا چپ بلوک ها را نیز به همان ترتیب در ماتریسی دیگر ذخیره می کنیم. (کوچک ترین بیت را برای این صفر میکنیم که کنتراست به درستی محاسبه شود . در حقیقت ما در این تابع میخواهیم این بیت را تغییر دهیم و پس از این تغییر ، کنتراست بلوک ها نیز تغییر می کند . در تابع decoding میخواهیم ترتیب بلوک ها همانند قبل باشد تا پیام به درستی خوانده شود ، اما اگر موقع محاسبه

کنتراست این بیت نیز در نظر گرفته شود ، ممکن است ترتیب بلوک ها هنگام decoding مانند encoding نباشد .

در ابتدا اندازه ی پیام را بررسی می کنیم تا قابل گنجاندن در عکس باشد . سپس پیام را به رشته ی باینری معادل تبدیل می کنیم (با استفاده از mapset)

سپس تصویر را بلوک بندی کرده و مقدار کنتراست آنها را حساب میکنیم و بر اساس آن مرتب می کنیم .

برای قرار دادن پیام در بلوک ها ، از اولین بلوک(که بیشترین کنتراست را دارد) شروع می کنیم .

هنگام نوشتن مقدار باینری در بلوک ، از بالا سمت چپ بلوک تا انتهای بلوک ، مقدار بیت آخر هر پیکسل را تغییر می دهیم . بعد از اتمام شدن کار یک بلوک ، آن را در تصویر قبلی در همان محل قرار میدهیم تا تصویر جدید آرام آرام ایجاد شود .

اسکریپت مربوط به تابع encode :

```
function codedpic = coding(message, initialpic, mapset)

    % checking size of the message
    if length(message)*5 < (size(initialpic, 1)-rem(size(initialpic, 1), 5))*(size(initialpic, 2)-rem(size(initialpic, 2), 5))
        % Turning message into binary
        message_len = length(message);
        binarymessage = ''; % Initialize binary message string
        for i = 1:message_len
            ch = message(i);
            found = false;
            for j = 1:32
                if ch == mapset{1, j}
                    binarymessage = [binarymessage, mapset{2, j}]; % Append binary representation of character to binary message
                    found = true;
                    break;
                end
            end
            if ~found
                fprintf('Character "%c" not found in mapset. Skipping...\n', ch);
            end
        end
        binarymessage_len = length(binarymessage);

```

```
24          % Finding blocks with most contrast values
25          blocks = {};
26          contrasts = [];
27          indices = [];
28          for i = 1:5:size(initialpic, 1)-rem(size(initialpic, 1), 5)
29              for j = 1:5:size(initialpic, 2)-rem(size(initialpic, 2), 5)
30                  block = initialpic(i:i+4, j:j+4);
31                  blockWithoutLSB = bitand(block, 254); % Clear LSB
32                  contrast = max(blockWithoutLSB(:)) - min(blockWithoutLSB(:));
33                  blocks{end+1} = block;
34                  contrasts(end+1) = contrast;
35                  indices(end+1, :) = [i, j];
36              end
37          end
38          [sortedContrasts, sortIdx] = sort(contrasts, 'descend');
39          sortedBlocks = blocks(sortIdx);
40          sortedIndices = indices(sortIdx, :);
41
```

```
42          % Writing message in the blocks
43          i = 1;
44          for l = 1:length(sortedBlocks)
45              block_to_change = sortedBlocks{l};
46              i_start = sortedIndices(l, 1);
47              j_start = sortedIndices(l, 2);
48              for m = 1:size(block_to_change, 1)
49                  for n = 1:size(block_to_change, 2)
50                      val = block_to_change(m, n);
51                      if i <= binarymessage_len
52                          val_bin = dec2bin(val, 8);
53                          val_bin(end) = binarymessage(i);
54                          block_to_change(m, n) = bin2dec(val_bin); % Convert back to decimal and store in the block
55                          i = i + 1;
56                      else
57                          break;
58                      end
59                  end
60                  if i > binarymessage_len
61                      break;
62                  end
63              end
64              initialpic(i_start:i_start+4, j_start:j_start+4) = block_to_change;
65          end
66          codedpic = initialpic;
67      else
68          disp('message is too large!') ;
69          codedpic = 1 ;
70      end
71  end
```

تمرین 1-3)

تصویر encode شده و تصویر اصلی:

Original PIC

Coded PIC

تفاوتی میان تصویر اصلی و تصویر دارای پیام مشاهده نمیشود . علت این مسئله آن است که ما بخش هایی با بیشترین کنتراست را انتخاب کردیم و همچنین کم ارزش ترین بیت آن ها را تغییر دادیم

تمرین 1-4)

در اینجا نیز همانند تابع encoding تصویر را بر اساس بیشترین کنتراست بلوک بندی می کنیم و در یک آرایه قرار میدهیم (کنتراست را بدون توجه به بیت آخر محاسبه می کنیم)

حال پیام داخل عکس را باید بخوانیم . از بلوک با بیشترین کنتراست شروع کرده و از بالا سمت چپ بلوک تا انتهای بلوک پیش می رویم تا زمانی که به ; برسیم .پس از آن باید بیت ها را به کاراکتر ها تبدیل کنیم . برای این کار 5 بیت 5 بیت جدا میکنیم و عدد به دست آمده را به decimal تبدیل می کنیم . کاراکترهای متناظر را پیدا میکنیم و کنار هم قرار میدهیم تا پیام رمزگذاری شده ایجاد گردد. در انتها این پیام را چاپ می کنیم.

اسکریپت مربوط به decoding :

```
function DcodedMessage = decoding(codedpic, Alphabet, blocksize)

    % Finding blocks with most contrast value
    blocks = {};
    contrasts = [];
    for i = 1:blocksize:size(codedpic, 1)-rem(size(codedpic, 1), blocksize)
        for j = 1:blocksize:size(codedpic, 2)-rem(size(codedpic, 2), blocksize)
            block = codedpic(i:i+blocksize-1, j:j+blocksize-1);
            blockWithoutLSB = bitand(block, 254); % Clear LSB
            contrast = max(blockWithoutLSB(:)) - min(blockWithoutLSB(:));
            blocks{end+1} = block;
            contrasts(end+1) = contrast;
        end
    end
    [sortedContrasts, sortIdx] = sort(contrasts, 'descend');
    sortedBlocks = blocks(sortIdx);

    % Decoding the message
    DcodedMessageBin = '';
    for k = 1:length(sortedBlocks)
        block_to_look = sortedBlocks{k};
        for m = 1:size(block_to_look, 1)
            for n = 1:size(block_to_look, 2)
                val = block_to_look(m, n);
                val_bin = dec2bin(bitand(val, 1), 1); % Extract LSB
                DcodedMessageBin = [DcodedMessageBin val_bin];
                if length(DcodedMessageBin) >= 5 && strcmp(Alphabet(bin2dec(DcodedMessageBin(1:5))+1), ';')
                    break;
                end
            end
            if length(DcodedMessageBin) >= 5 && strcmp(Alphabet(bin2dec(DcodedMessageBin(1:5))+1), ';')
                break;
            end
        end
        if length(DcodedMessageBin) >= 5 && strcmp(Alphabet(bin2dec(DcodedMessageBin(1:5))+1), ';')
            break;
        end
    end

    % Translate binary representation to characters
    DcodedMessage = '';
    for i = 1:5:length(DcodedMessageBin)
        chunk = DcodedMessageBin(i:min(i+4, length(DcodedMessageBin))); % Ensure chunk size is at most 5 bits
        index = bin2dec(chunk) + 1;
        if index <= length(Alphabet)
            character = Alphabet(index);
            if strcmp(character, ';')
                break;
            else
                DcodedMessage = [DcodedMessage character];
            end
        end
    end
    disp(DcodedMessage) ;
end
```

اسکریپت main جهت تست :

```
close all;
clc;
clear;

% 1- Creating the mapset
Nch = 32 ;
mapset = cell(2,Nch) ;
Alphabet = 'abcdefghijklmnopqrstuvwxyz .,!";' ;
for i = 1:Nch
    mapset{1,i} = Alphabet(i) ;
    mapset{2,i} = dec2bin(i-1,5) ;
end

% 2- Coding function
X = imread('Amsterdam.jpg') ;
initialpic = rgb2gray(X) ;

message = 'signal;' ;
codedpic = coding(message,initialpic,mapset) ;

% 3- plotting
figure
plot1 = subplot(1,2,1) ;
imshow(initialpic)
title('Original PIC')
plot2 = subplot(1,2,2) ;
imshow(codedpic)
title('Coded PIC')
linkaxes([plot1 plot2])
```

```
% 4- Decoding function
blocks_size = 5 ;
DcodedMessage = decoding(codedpic,Alphabet,blocks_size) ;
```

نتیجه تست :

```
signal
>>
```

تمرین 1-5)

در ادامه روشی را برای رمزگشایی پیامی که ممکن است دچار نویز شده باشد را ارائه میدهیم .

از روش Hamming code استفاده می کنیم .مثلا اگر ما تصویری 4x4 داشته باشیم ، با 5 بیت میتوان نویز ها را تشخیص داد و اصلاح کرد .

دو بیت را به ستون ها اختصاص می دهیم . یک بیت برای ستون 1و3 و بیت دیگر برای ستون 2و4 . این دو بیت بیانگر زوج یا فرد بودن تعداد یک ها در ستون های مربوط به خودشان است .

دو بیت دیگر را به ردیف ها اختصاص می دهیم . یک بیت برای ردیف 1و3 و بیت دیگر برای ردیف 2و4 . این دو بیت بیانگر زوج یا فرد بودن تعداد یک ها در ردیف های مربوط به خودشان است .

هنگام مخابره ی پیام این بیت ها را مشخص میکنیم که اگر در مسیر انتقال ، پیام دچار نویز شد از روی این بیت ها بتوانیم نویز را تشخیص دهیم .

حال اگر روی این بیت ها نویز داشته باشیم چه ؟

بیت پنجم بیانگر زوج یا فرد بودن تعداد یک ها در کل تصویر است .

اگر یکی از آن 4 بیت دچار نویز شود ، با این بیت پنجم میتوان آن را تشخیص داد و اصلاح کرد.

بیت های حامل پیام data bit و بیت هایی که نشان دهنده ی زوج یا فرد بودن تعداد یک ها بودند، parity bit نام دارند.

(این ایده تقریبا در درس مدار منطقی بیان شد توضیحات مطرح شده بر اساس آن است .)

بخش دوم :

در این بخش می خواهیم از ضریب همبستگی نرمالایز شده استفاده کنیم . پس برای محاسبه ی correlation به جای استفاده از تابع corr2 از تابع normxcorr2 استفاده می کنیم.

برای پیدا کردن IC های مد نظر در PCB ، باید تصویر IC را در کل تصویر PCB حرکت داد و مقدار correlation را در تمامی مناطق صفحه حساب کرد . از آنجایی که هر دو تصویر rgb هستند ، باید برای هر سه رنگ این محاسبه را انجام دهیم و میانگین این اعداد را به عنوان correlation نهایی اعلام می کنیم .

با دستور find و قرار دادن threshold به اندازه 0.6 مناطق مورد نظر را پیدا کرده و با دستور rectangle آنها را مشخص می کنیم .یک بار تمام این کار ها را نیز برای IC ای که 180 درجه چرخیده است نیز انجام می دهیم .(IC اصلی را با دستور imrotate می چرخانیم و سپس مجددا correlation گیری می کنیم.)

اسکریپت تابع این بخش:

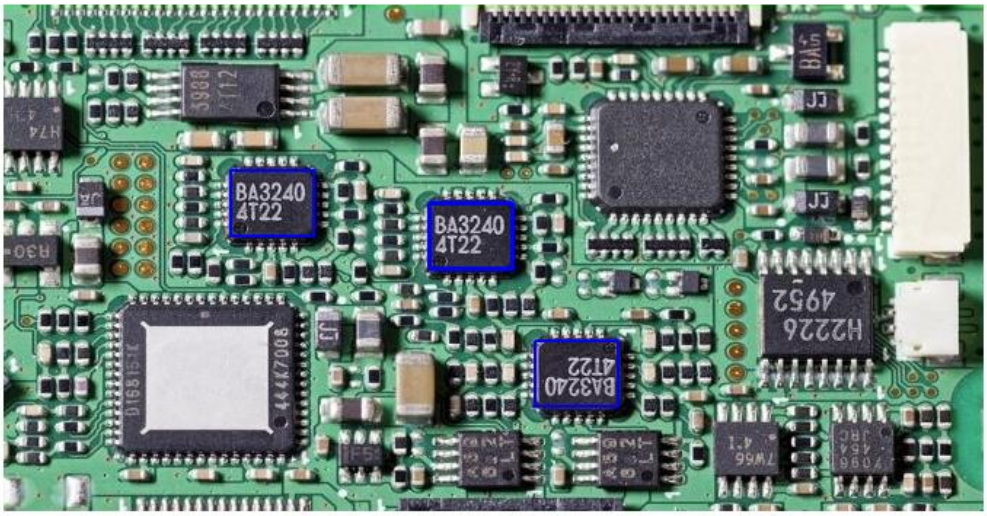
```
function marked_image = ICrecognition (IC_image,PCB_image)
    %calculating correlation for three colors
    corr1 = normxcorr2(IC_image(:,:,1),PCB_image(:,:,1)) ;
    corr2 = normxcorr2(IC_image(:,:,2),PCB_image(:,:,2)) ;
    corr3 = normxcorr2(IC_image(:,:,3),PCB_image(:,:,3)) ;
    corr = (corr1+corr2+corr3)/3 ;
    corrIdx = find(abs(corr(:))>0.6) ;
    figure
    imshow("PCB.jpg") ;
    hold on ;
    % showing the IC by a blue rectangle
    for i=1:length(corrIdx)
        [peakY,peakX] = ind2sub(size(corr),corrIdx(i)) ;
        corr_offset = [peakX-size(IC_image,2),peakY-size(IC_image,1)] ;
        bbox = [corr_offset(1),corr_offset(2),size(IC_image,2),size(IC_image,1)] ;
        rectangle('Position',bbox,'EdgeColor','b') ;
        hold on ;
    end
    %calculating correlation for rotated image
    rotated_ic = imrotate(IC_image,180) ;
    corr1 = normxcorr2(rotated_ic(:,:,1),PCB_image(:,:,1)) ;
    corr2 = normxcorr2(rotated_ic(:,:,2),PCB_image(:,:,2)) ;
    corr3 = normxcorr2(rotated_ic(:,:,3),PCB_image(:,:,3)) ;
    corr = (corr1+corr2+corr3)/3 ;
    corrIdx = find(abs(corr(:))>0.6) ;
    %showing the rotated IC by a blue rectangle
    for i=1:length(corrIdx)
        [peakY,peakX] = ind2sub(size(corr),corrIdx(i)) ;
        corr_offset = [peakX-size(IC_image,2),peakY-size(IC_image,1)] ;
        bbox = [corr_offset(1),corr_offset(2),size(IC_image,2),size(IC_image,1)] ;
        rectangle('Position',bbox,'EdgeColor','b') ;
        hold on ;
```

اسکریپت تست این بخش:

```
close all;
clc;
clear;

ic = imread("IC.png") ;
pcb = imread("PCB.jpg") ;
ICrecognition(ic,pcb) ;
```

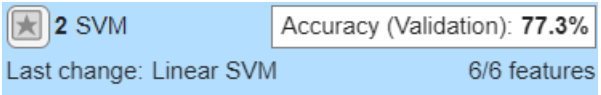
خروجی اسکریپت:



بخش سوم :

تمرین 3-1)

دقت به دست آمده با استفاده از تمام فیچر ها :



تمرین 3-2)

Glucose: 74%

Blood Pressure : 65.3 %

Skin Thickness : 65.3 %

Insulun: 65.3 %

BMI : 64.3 %

Age : 65.3 %

گلوکز ارتباط بیشتری به دیابتی بودن یک نفر دارد .

تمرین 3-3)

لیبل 77.5 ٪ داده ها را درست تشخیص می دهد.

```
>> main3

similarity =

   77.5000
```

اسکریپت این بخش:

```
dia1 = trainedModel.predictFcn(diabetestraining) ;
dia2 = table2array(diabetestraining) ;
dia2 = dia2(:,7) ;
s = dia1==dia2 ;
similarity = (sum(s)/numel(s))*100
```

تمرین 3-4)

لیبل 78 ٪ داده ها را درست تشخیص می دهد.

```
>> main3_4

similarity =

    78
```

اسکریپت این بخش:

```
dia1 = trainedModel.predictFcn(diabetesvalidation) ;
dia2 = table2array(diabetesvalidation) ;
dia2 = dia2(:,7) ;
s = dia1==dia2 ;
similarity = (sum(s)/numel(s))*100
```