تمرین کامپیوتری 3 سیگنال ها و سیستم ها

مرضیه موسوی 810101526

مهراد لیویان 810101501

بخش یک:

-1

ابتدا تابع Build_mapset را برای ساختن mapset درست کردیم:

الفبای مورد نظرمان را تعریف کردیم و با استفاده از دستور dec2bin ورژن باینری آن ها را گذاشتیم:

```
function MapSet=Build_mapset()
    MapSet=cell(2,32);
    alphabet = 'abcdefghijklmnopqrstuvwxyz .,!";';
    for i = 1:32
        MapSet{1,i} = alphabet(i);
        MapSet{2,i} = dec2bin(i-1, 5);
    end
```

چرا1-i ؟چون می خواهیم اولین حرف الفبا به صورت00000 باشد

-2

```
function codedImage = coding(message, image, mapset)
    % Check if the message can fit into the image
    if length(message) > numel(image)
        error('Error in Encoding: Message is too large to encode in the given image.');
    end

    % Convert each character in the message to its binary representation using the mapset
    messageBinary = cell(1, length(message));
    for i = 1:length(message)
        currentChar = message(i);
        % Find the binary representation for the current character in the mapset
        charIndex = strcmp(currentChar, mapset(1, :));
        if ~any(charIndex)
            error('Error: Character not found in mapset.');
        end
        messageBinary{i} = mapset{2, charIndex};  % Store the binary representation
    end

    % Concatenate the binary representation of the entire message
    codedBinaryMessage = strcat(messageBinary{:});

    % Make a copy of the image to embed the message
    codedImage = image;

    % Embed each bit of the binary message into the least significant bit of the image pixels
    for i = 1:length(codedBinaryMessage)
        % Get the current pixel value and convert to binary
        pixelValue = codedImage(i);
        pixelBinary = dec2bin(pixelValue, 8);  % Ensure 8-bit representation for consistency

        % Replace the least significant bit with the current message bit
        pixelBinary(end) = codedBinaryMessage(i);

        % Convert binary string back to decimal and store in the coded image
        codedImage(i) = bin2dec(pixelBinary);
    end
end
```

پروسه ی رمزنگاری مانند پروسه ی گفته شده در کلاس است.ابتدا همه ی کاراکتر های در مسیجمان را با استفاده از مپ ستمان به کاراکتر های باینری تبدیل می کنیم و آن ها را concat می کنیم.حال از نقطه ای در عکس شروع می کنیم.و یک پوینتر هم بر سر استرینگ رمزنگاری شده مان می گذاریم.به ترتیب هر بیت آخر پیکسل را اگر بیت متناظر در استرینگ 1 بود 1 و در غیر این

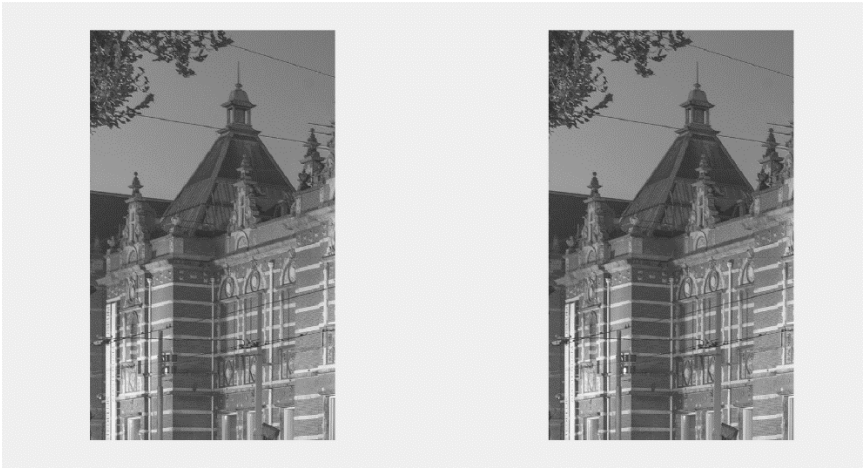
صورت 0 می گذاریم.بخشی هم مربوط به error handling است که در صورت بزرگتر بودن سایز عکس یا نبود کاراکتر در میست ارور دهد.

-3

اسکریپتی که برای تست نوشتیم به صورت زیر است:

```
mapset =Build_mapset()
input = imread("Amsterdam.jpg");
input = rgb2gray(input);
codedImage = coding ('signal;' , input , mapset);
subplot(1,2,1);
imshow(input);
subplot(1,2,2);
imshow(codedImage);
```

ابتدا میست را می سازیم.تصویر را سیاه و سفید می کنیم.سمی کولن آخر سیگنال برای اتمام پیام است.حال پیام و عکسی که می خواهیم رمزنگاری روی آن انجام شود و مپ ست را به تابع coding که در بخش قبل نوشتیم پاس می دهیم.سپس تصویر کد شده و تصویر اصلی را نمایش می دهیم.



همانطور که در تصویر نیز مشخص است تصویر تفاوت چندانی نکرده است.دلیل آن این است که فقط lsb پیکسل ها تغییر کرده که در یک عدد 8 بیتی نسبت به مقدار عدد بسیار کوچک است.

-4

اسکریپت متلب به صورت زیر است:

```
function decodedMessage = decoding(picture, mapSet, threshold)
    % Initialize the binary message string
    binMessage = '';

    % Extract the LSB from each pixel in the picture
    for i = 1:numel(picture)
        pixelBinary = dec2bin(picture(i), 8);  % Ensure an 8-bit binary representation
        lsb = pixelBinary(end);  % Get the least significant bit
        binMessage = [binMessage, lsb];  % Append LSB to the binary message
    end

    % Initialize the decoded message
    decodedMessage = '';

    % Decode the binary message in chunks
    while length(binMessage) >= threshold
        % Extract the first 5 bits for character decoding
        binChar = binMessage(1:5);
        binMessage = binMessage(6:end);  % Remove the processed bits

        % Find the corresponding character in the mapSet
        wordIndex = find(strcmp(binChar, mapSet(2, :)), 1);  % Return the first match
        if isempty(wordIndex)
            error('Error in Decoding: Binary segment not found in mapSet.');
        end

        char = mapSet{1, wordIndex};

        % Check for termination character
        if char == ';'
            decodedMessage = [decodedMessage, char];
            break;
        end

        % Append decoded character to the message
        decodedMessage = [decodedMessage, char];
    end

    % Display the decoded message
    disp(decodedMessage);
end
```

ابتدا یکبار روی تک تک پیکسل های عکس پیمایش می کنیم و lsb آن ها را ذخیره می کنیم.تا یک سایز مشخصی 5 تا 5تا بیت استخراج می کنیم.حال 5 بیت استخراج شده را در مپست جست و جو می کنیم.بخش error handling مطمین می شود که حتما عدد در مپست ما باشد.

حال کاراکتر یافت شده را به مسیج دیکود شده اضافه می کنیم تا زمانی که semicolon را ببینیم.

اسکریپت تست به صورت زیر است:

```
mapset=Build_mapset();
input=imread('Amsterdam.jpg');
coded=coding('signal;', input, mapset);
decoding(coded,mapset,35);
```

نتیجه ی تست به صورت زیر است که درست است:

```
>> test_decoding
signal;
```

-5

اگر کلماتی که بهم نزدیک هستند (edit distance کمی دارند) و با تغییرات کوچکی بهم تبدیل می شوند را داشته باشیم اگر به خاطر نویز مثلا به جای apple عبارت aple مخابره شد چون کلمات نزدیک را داریم می توانیم جایگزینی را بعد از رمزنگاری انجام دهیم و پیام درست را بدست آوریم.

بخش دو:

تابعی به نام icrecognition نوشتیم و آن را در اسکریپت متلب صدا زدیم:

```
IC = imread('IC.png');
PCB = imread('PCB.jpg');
threshold = 0.7;

ICRecognition(IC, PCB, threshold);
```

حال به بررسی بخش به بخش کد icrecognition می پردازیم:

```
function ICRecognition(templateImage, mainImage, threshold)
    % ICrecognition: Locates and highlights templateImage matches in mainImage
    % using normalized cross-correlation.
    %
    % Inputs:
    %   - templateImage: Template image (IC) to find in the main image
    %   - mainImage: Main image (PCB) for searching
    %   - threshold: Correlation threshold to define significant matches

    % Convert images to grayscale if needed
    templateImage = ensureGrayscale(templateImage);
    grayMainImage = ensureGrayscale(mainImage);

    % Calculate cross-correlation for template and 180-degree rotated template
    [correlationMatrix, correlationMatrixRotated] = computeCorrelations(templateImage, grayMainImage);

    % Display main image and overlay matched regions
    figure;
    imshow(mainImage);
    title('Detected Matching Regions');
    hold on;

    % Highlight matches for original and rotated templates
    highlightMatches(correlationMatrix, templateImage, threshold, 'b');
    highlightMatches(correlationMatrixRotated, templateImage, threshold, 'g');

    hold off;
end
```

ابتدا در تابع ensure grayscale چک می کنیم که تصویر سیاه سفید است یا خیر

```
function imgGray = ensureGrayscale(img)
    % Convert image to grayscale if it has 3 channels
    if size(img, 3) == 3
        imgGray = rgb2gray(img);
    else
        imgGray = img;
    end
end
```

در صورتی که عکس دارای سه بعد یا کانال بود یعنی سیاه سفید نیست و باید سیاه سفید شود و برای سیاه سفید کردن از تابع rgb2gray استفاده کردیم.

در بخش بعدی correlation عکس و تمپلیت و تمپلیت 180 درجه برعکس شده (طبق صورت پروژه یا خود ic هست یا ic که 190 درجع دوران کرده)را محاسبه کردیم:

```
function correlationMap = normalizedCorrelation(PCB, IC)
    % Convert IC image to double for precision in calculations
    IC = double(IC);

    % Get dimensions of IC and PCB images
    [IC_height, IC_width] = size(IC);
    [PCB_height, PCB_width] = size(PCB);

    % Initialize an empty matrix to store correlation values
    correlationMap = zeros(PCB_height - IC_height + 1, PCB_width - IC_width + 1);

    % Iterate over possible positions in the PCB image where IC can fit
    for row = 1:(PCB_height - IC_height + 1)
        for col = 1:(PCB_width - IC_width + 1)
            % Extract a sub-image from PCB matching the IC's size
            subImage = double(PCB(row:row + IC_height - 1, col:col + IC_width - 1));

            % Calculate the normalized cross-correlation
            numerator = sum(sum(IC .* subImage));
            denominator = sqrt(sum(sum(IC .* IC)) * sum(sum(subImage .* subImage)));
            correlationMap(row, col) = numerator / denominator;
        end
    end
end
```

ابتدا عکس ic را به دابل تبدیل می کنیم تا دقت اندازه گیری بالا رود.مانند یک فیلتر ic را روی pcb میندازیم و به اندازه ی آن آرایه از 0 می گیریم که مقادیر correlation را ذخیره کند.مستطیل های با سایز ic در عکس پیدا می کنیم.با استفاده از فرمول داده شده در صورت پروژه کورلیشن را حساب کرده و ذخیره می کنیم.

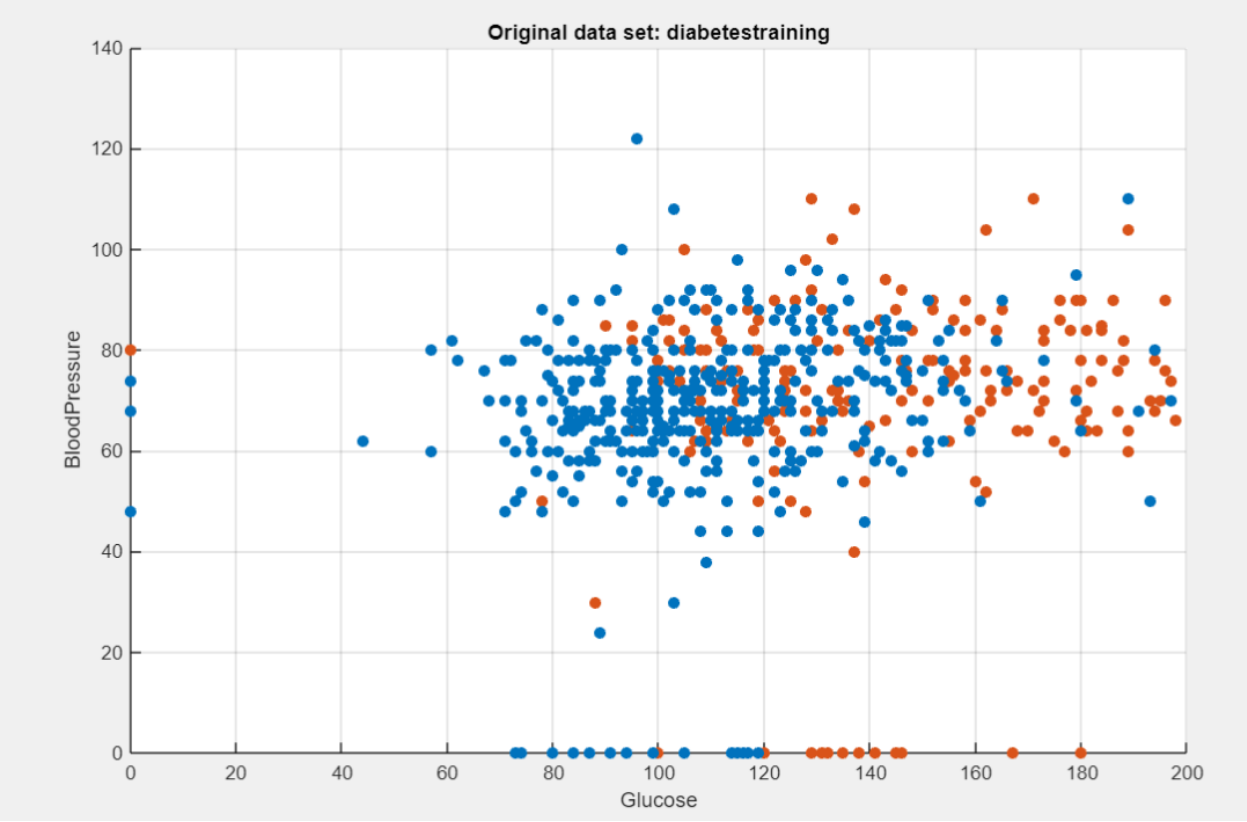
بخش بعدی مربوط به تابع کشیدن مستطیل ها می باشد.

```
function highlightMatches(corrMatrix, template, threshold, color)
    % Identify local maxima above threshold and overlay rectangles
    localMaxima = imregionalmax(corrMatrix) & (corrMatrix > threshold);
    [row, col] = find(localMaxima);

    % Adjust for padding and overlay rectangles
    rowAdj = row - size(template, 1) + 1;
    colAdj = col - size(template, 2) + 1;
    for i = 1:length(rowAdj)
        rectPos = [colAdj(i), rowAdj(i), size(template, 2), size(template, 1)];
        rectangle('Position', rectPos, 'EdgeColor', color, 'LineWidth', 2);
    end
end
```

کورلیشن و تمپلیت آستانه را می گیریم.هر لوکال ماکسیممی که پیدا کردیم با استفاده از طول و عرض تمپلیت طول و عرض مستطیل ic را می گیریم.در اینجا در بخش کورلیشن دنبال ماکسیمم های محلی می گردیم.همانطور که در گذشته به دنبال ماکسیمم در کورلیشن می گشتیم.همچنین یک آستانه تعیین می کنیم که اگر نویز رندومی به عنوان مستطیل انتخاب شد و ماکسیمم بود آن دورش مستطیل کشیده نشود.

بخش سه:



-1

دقت بدست آمده ی مدل با 6 فیچر گفته شده به صورت زیر است:

Model 2: SVM
Status: Trained

Training Results

Accuracy (Validation)	77.3%
Total cost (Validation)	136
Prediction speed	~3000 obs/sec
Training time	9.7317 sec
Model size (Compact)	~25 kB

- **Model Hyperparameters**
- **Feature Selection: 6/6 individual features selected**
- **PCA: Disabled**
- **Misclassification Costs: Default**
- **Optimizer: Not applicable**

-2

اکیورسی به ازای استفاده از هر یک از فیچر ها به صورت زیر است:

2 SVM	Accuracy (Validation): **77.3%**	
Last change: Linear SVM	6/6 features	
3 SVM	Accuracy (Validation): 74.3%	
Last change: Linear SVM	1/6 features	
4 SVM	Accuracy (Validation): 65.3%	
Last change: Changed 2 features	1/6 features	
5 SVM	Accuracy (Validation): 65.3%	
Last change: Changed 2 features	1/6 features	
6 SVM	Accuracy (Validation): 65.3%	
Last change: Changed 2 features	1/6 features	
7 SVM	Accuracy (Validation): 65.5%	
Last change: Linear SVM	1/6 features	
8 SVM	Accuracy (Validation): 65.3%	
Last change: Changed 2 features	1/6 features	

	Features
1	Glucose
2	BloodPressure
3	SkinThickness
4	Insulin
5	BMI
6	Age

همانطور که مشخص است فیچری که بیشترین تاثیر را روی دیابت داشتن دارد میزان گلوکز خون است و بقیه فیچر ها میزان تاثیرشان باهم برابر است.

-3

اسکریپت متلب به صورت زیر است:

```
predictedLabels = trainedModel.predictFcn(diabetestraining);
trueLabels = diabetestraining.label;
accuracy = mean(predictedLabels == trueLabels) * 100;
fprintf('Training Phase Accuracy: %.2f%%\n', accuracy);
```

دقت به دست آمده برای این مدل :

```
>> q3_3
Training Phase Accuracy: 77.50%
```

-4

اسکریپت متلب به صورت زیر است:

```
predictedLabels = trainedModel.predictFcn(diabetesvalidation);
trueLabels = diabetesvalidation.label;
accuracy = mean(predictedLabels == trueLabels) * 100;
fprintf('Training Phase Accuracy: %.2f%%\n', accuracy);
```

میزان دقت به دست آمده :

```
Training Phase Accuracy: 78.00%
>> predictedLabels = trainedModel.predictFcn(trainingDiabetesData);
```