سیگنال و سیستم ها

سهیل حاجیان منش　　پروژه ۲　　۸۱۰۱۰۰۱۱۹

بخش اول) تمرین ۱-۱:

در این قسمت در یک حلقه 32تایی به هریک از کاراکترهایمان یه عدد باینری 32 بیتی نسبت دادیم وMapSet خود را میسازیم که شامل دوسطر 32 ستون است. سطر اول اسامی کاراکترها و سطردوم عدد باینری مربوط به هرکاراکتر.

شکل زیر قسمتی از MapSet را نشان میدهد:

mapSet

2x32 cell

	1	2	3	4	5	6	7	8	9	10	11	12	
1	a	b	c	d	e	f	g	h	i	j	k	l	m
2	00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	0110
3													

تمرین ۲-۱:

ابتدا تک تک حروف پیام را به صورت یک رشته باینری 5 بیتی درآورده و یک رشته باینری کد شده میسازیم.لازم است در انتهای رشته مقدار 11111 را بگذاریم که نشاندهنده ; به معنای انتهای پیام است.

در این مرحله در ابتدا تصویر خروجی را برابر تصویر ورودی اولیه قرار میدهیم،

سپس تعداد سطرها و ستون های تصویر را پیدا کرده و با ضرب آنها تعداد پیکسل های تصویر را مشخص میکنیم.دراینجا اگر طول رشته باینری از تعداد پیکسل های تصویر بیشتر بود باید خطا دهیم.

سپس در دو حلقه تودرتو پیکسل ها را بترتیب ستونی پیمایش میکنیم. هر پیکسل یک عدد دسیمال بین 0 تا 255 است.ابتدا ان را به باینری تبدیل کرده و سپس کم ارزش ترین بیت آن را با رقم باینری رشته رمزگذاری شده جاگذاری میکنیم.تا زمانی که رشته رمز گذاری شده تمام شود این کار را تکرار میکنیم و سپس از تابع خارج میشویم.

تمرین۳-۱:

همانطور که در تصویر مشخص است هیچگونه تفاوتی بیت عکس اصلی و عکس رمزگذاری شده نمیتوان پیدا کرد حداقل با چشم غیرمسلح.

علت آن مشخص است ما در کد به ازای هرپیکسل تنها یک بیت آنرا تغییر داده ایم و آن هم کم ارزش ترین بیت. یعنی درواقع هرپیکسل ما یا یک عدد بزرگتر میشود و یا یک عدد کوچکتر و خب طبیعتا نمیتوان تفاوتی خاصی بین تصویر اصلی و تصویر کدگذاری شده یافت.

تمرین۴-۱:

ابتدا تابع **findInMapSet** را تعریف کرده ام. این تابع **MapSet** و یک ماتریس **1x5** را که شامل ارقام صفر و یک است میگیرد. در ابتدا ماتریس را به یک رشته باینری 5 بیتی تبدیل میکند و بعد کاراکتر متناظر با آن رشته باینری را از **MapSet** پیدا کرده و خروجی میدهد.

در تابع اصلی در دو حلقه تودرتو شروع به پیمایش پیکسل ها میکنیم و سپس به ازای هر پیکسل کم ارزش ترین بیت آنرا در ماتریکس **pocket** ذخیره میکنیم و هر پنج پیکسل یکبار ماتریس **pocket** را به **findInMapSet** میدهیم تا کاراکتر رمز گذاری شده مرتبط با آن 5 پیکسل را پیدا کنیم.در آخر هرموقع

pocket برابر **[1,1,1,1,1]** شود یعنی به انتهای پیام رسیده ایم و پیام دیکود شده را بعنوان خروجی در نظر میگیریم.

```
decodeMsg=decoding(codedPicture,mapSet);
disp(decodeMsg);
```

همانطور که در تصویر

```
>> p3_1_1
signal
>>
```

مشخص است پیام متناظر با تمرین اول که در

تصویر کدگذاری شده بود به درستی دیکود شده است.

تمرین۵-۱:

نوع نویز اضافه شده به تصویر و مقدار آن نقش مهمی دارد. اگر نویز تصادفی و حداقل باشد، ممکن است تاثیر قابل توجهی بر توانایی رمزگشایی پیام نداشته باشد. با این حال، اگر نویز قابل توجه و ساختاری باشد، میتواند استخراج پیام را دشوار کند.

میتوان در حین رمزگذاری روش هایی استفاده کرد که درصورت ایجاد نویز بتوان پیام را همچنان رمز گشایی کرد. بطور مثال میتوان اطلاعاتی اضافی را قبل از افزودن یک پیام به تصویر به پیام اضافه کرد به گونه ای این اطلاعات اضافی را بتوانیم تشخیص بدهیم و اگر هم نویزی وارد تصویر شد این اطلاعات اضافی تغییر بکنند و احتمال تغییر و خراب شدن پیام اصلی را کم بکنند.

همچنین میتوان پیام را بطور تصادفی در پیکسل های مختلف ذخیره کرد البته باید مکانیزمی طراحی کرد که بتوانیم پیکسل های که پیام در آن ذخیره شده است را بترتیب هنگام رمزگشایی پیدا کنیم.

بطور کلی اگر مقدار نویز زیاد باشد هیچکدام از روش های بالا ممکن است جواب ندهند.

تمرین۶-۱:

چند روش برای تشخیص اینکه یک تصویر کد گذاری شده است یا خیر وجود دارد که مواردی از آنها در پایین ذکر کرده ام.البته هیچکدام از این روش ها نمیتوانند تضمین کنند که بطور قطعی جواب درست به ما بدهند.

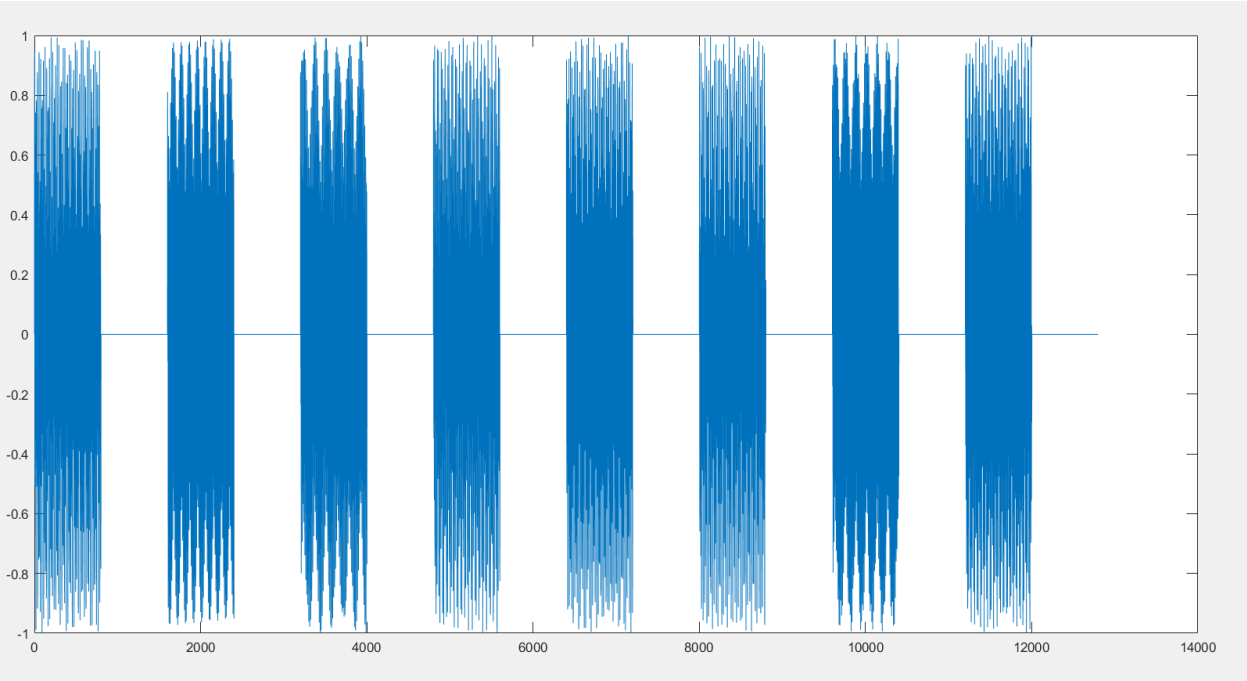
- ***بررسی چشمی***: با بررسی چشمی تصویر شروع کنید. به دنبال هر گونه بی نظمی یا الگویی باشید که طبیعی به نظر نمی رسد. گاهی اوقات، پیام‌های رمزگذاری شده می‌توانند تغییرات ظریفی را ایجاد کنند که ممکن است از نظر بصری آشکار نباشند، اما می‌توانند پس از بررسی دقیق خود را به عنوان یک بی نظمی جزیی نشان دهند و این حس را در ما ایجاد کنند که ممکن است تصویر در آن محدوده خاص رمزگذاری شده باشد.

- ***تجزیه و تحلیل هیستوگرام***: هیستوگرام مقادیر پیکسل در تصویر را تجزیه و تحلیل کنید. یک تصویر معمولی و بدون تغییر معمولاً دارای توزیع نسبتاً یکنواختی از مقادیر پیکسل است. یک تصویر رمزگذاری شده ممکن است الگوهای غیرعادی را در هیستوگرام نشان دهد.

- ***تجزیه و تحلیل نویز***: تصاویر کدگذاری شده ممکن است سطوح نویز کمی بالاتری در کم ارزش ترین بیتشان نسبت به تصاویر معمولی داشته باشند. برای شناسایی این نویز اضافی می توان از تکنیک های آنالیز نویز استفاده کرد.

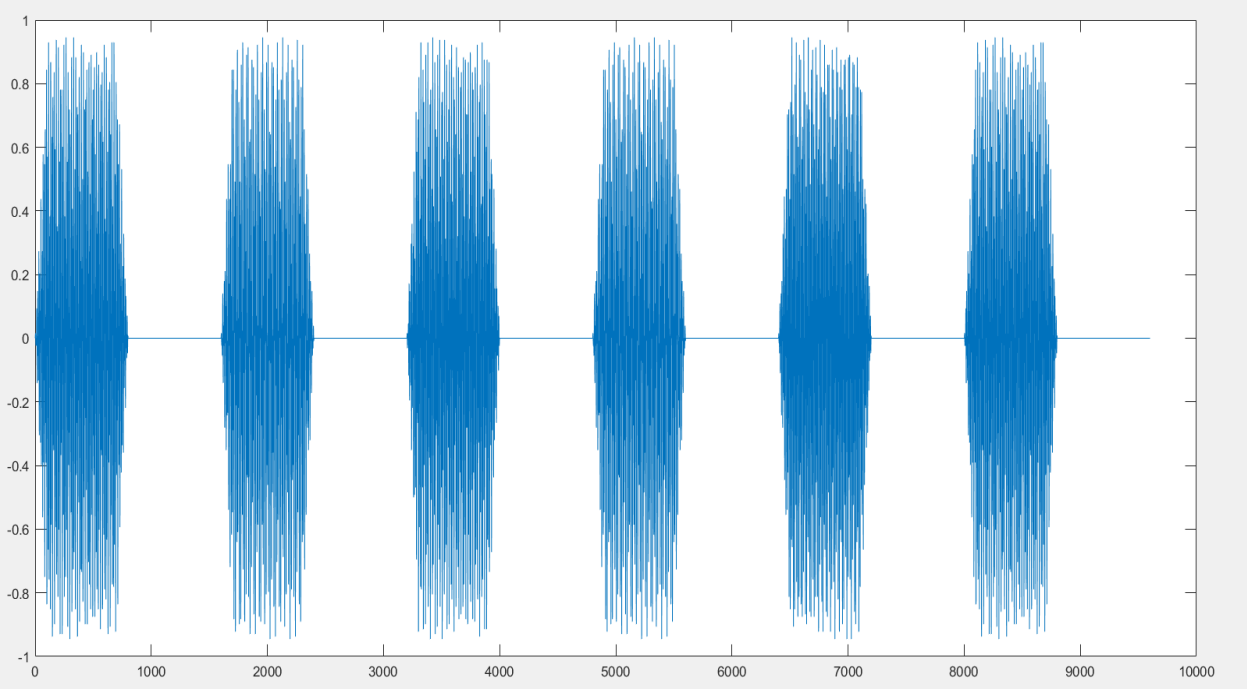
- ***تجزیه و تحلیل آنتروپی***: محاسبه آنتروپی تصویر، به ویژه در مناطق خاص مورد نظر. داده های رمزگذاری شده ممکن است سطوح آنتروپی بالاتری را در نواحی خاص به دلیل تصادفی بودن الگوریتم رمزگذاری نشان دهند. در زمینه تصاویر دیجیتال، آنتروپی به معنای اندازه‌گیری میزان تصادفی بودن یا عدم قطعیت در مقادیر پیکسلی داخل تصویر است. این مفهوم اغلب به عنوان یک معیار آماری برای ارزیابی محتوای اطلاعاتی یا پیچیدگی تصویر استفاده می‌شود

بخش دوم) تمرین۱-۲: در این بخش برای هر کلید یک فرکانس پایین و یک فرکانس بالا تخصیص داده می شود که به فرم سینوسی هستند لذا فرکانس معادل برای هر کلید یک فرکانس سینوسی متشکل از میانگین آنها است. همچنین در انتها فرکانس معادل با استفاده از دستور **zeros** یک تاخیر 0.1 ثانیه ای بوجود می آوریم. سپس برای تولید عدد 43219876 بترتیب سیگنال هر رقم را تولید کرده و به سیگنال ساخته شده تا قبل از آن اضافه میکنیم. در انتها سیگنال مرتبط با عدد داده شده به صورت زیر میباشد:



که با دستور **sound** قابل شنیدن است و طبق خواسته سوال در فایل **y.wav** ذخیره شده است.

تمرین۲-۲: در این بخش یک سیگنال با فرم زیر در ابتدا به ما داده شده است.



شکل بالا از ۶ سیگنال که مربوط به کلید های مختلف است تشکیل شده است. همچنین هر بخش شامل یک بازه صفر است که مربوط به **Toff** میباشد.

ابتدا فایل را با استفاده از دستور **audioread** میخوانیم و در مارتریس **a** ذخیره می کنیم. سپس از سیگنال مورد نظر در بازه های زمانی مطلوب نمونه برداری میکنیم تا سیگنال هر کلید را به صورت جداگانه برای مرحله بعد داشته باشیم. همانطور که گفته شد سیگنال هرکلید شامل بخش غیر صفر وبخش صفر که مرتبط به زمان **off** بودن است میباشد.

سپس برای هر سیگنال بین آن سیگنال و تمامی 12 سیگنال دیگر که مربوط به 12 کلید میباشند **correlation** میگیریم.این کار را با استفاده از تابع **corr2** انجام میدهیم. سپس سیگنالی که بیشترین **correlation** را با سیگنال مربوط داشت به عنوان کلید جواب آن سیگنال درنظر میگیریم و به رشته **result** اضافه میکنیم و در آخر مقدار **result** را چاپ میکنیم.

عدد بدست آمده برای فایل **a.wav** :

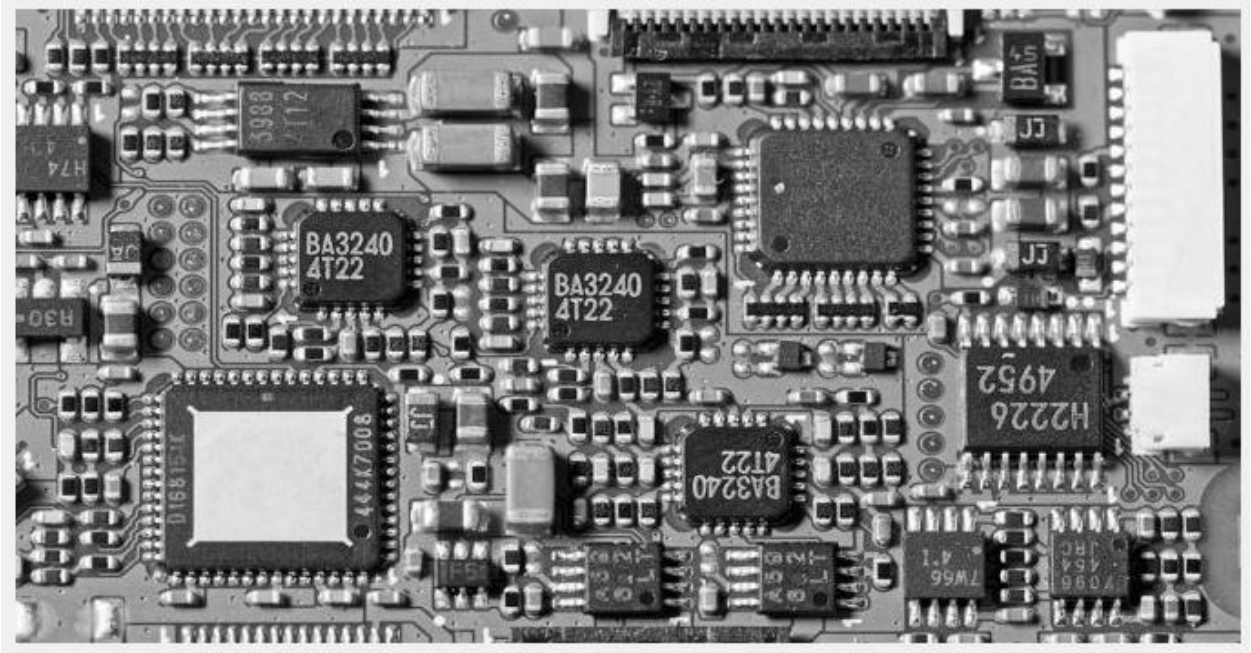
```
>> p3_2_2
810198
>>
```

همچنین اگر فایل **y.wav** را به عنوان ورودی در نظر بگیریم خروجی مقدار صحیح 43218765 خواهد شد که نشان میدهد کد عملکرد صحیحی دارد.

```
>> p3_2_2
43218765
>>
```

بخش سوم) ابتدا با استفاده از دستور imread دو عکس ic و pcb را میخوانیم و به عنوان ورودی به تابع ICreconition می دهیم. سپس ابتدا دو عکس را با استفاده از تابع rgb2gray به رنگ خاکستری تبدیل می کنیم.

خروجی عکس ها تا این مرحله:



حال تابع ضریب همبستگی دو بعدی **normalizeCorr** را با توجه به فرمول داده شده در صورت پروژه تعریف می کنیم الگوریتم عملکرد این تابع به این صورت است که ضریب همبستگی دو ماتریس دو در دو هم سایز را بدست می آورد و خروجی آن یک عدد است(با .* دو ماتریس را درایه به درایه کامل در هم ضرب و با یک بار **sum** گرفتن جمع ستونی و بار دوم **sum** گرفتن مجموع تمام درایه ها را به ما می دهد.)

تابع **corrMatrix** را اینگونه تعریف میکنیم که عکس **pcb** خاکستری و عکس **ic** خاکستری را دریافت کرده و عکس **pcb** را به قطعه ها هم اندازه با **ic**

تقسیم کرده و هر قطعه را با عکس **ic** کورولیشن میگیریم و در یک ماتریس مقادیر کورولیشن ها را ذخیره میکنیم و به عنوان خروجی درنظر میگیریم.

حال یکبار تابع **corrMatrix** را با عکس **ic** و یکبار با دوران یافته آن صدا میکنیم.اکنون از بین این دو ماتریس مختصات قطعاتی که مقدار کولیشن آنها بیشتر از **0.93** است را جدا میکنیم.

در تابع **plotRect** مختصات گوشه چپ بالای نقاط را که در قسمت قبل توضیح داده شد میگیریم و مستطیل ها را به کمک تابع **rectangle** میکشیم.

خروجی نهایی بصورت زیر میباشد که مطابق خواسته سوال است: