



دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیووتر

گزارش تکلیف دوم درس پردازش تصویر رقمه‌ی

دانشجو:

سید احمد نقوی نوزاد

ش-د: ۹۴۱۳۱۰۶۰

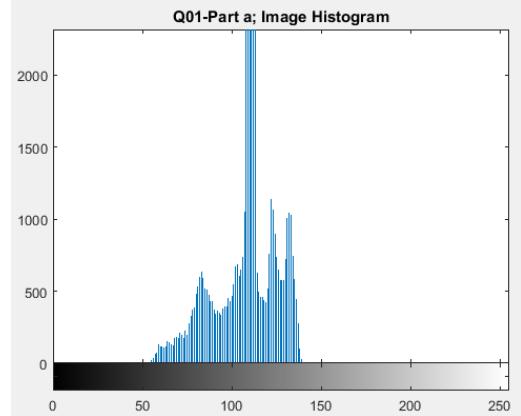
استاد:

دکتر رحمتی

جواب سوال ۱

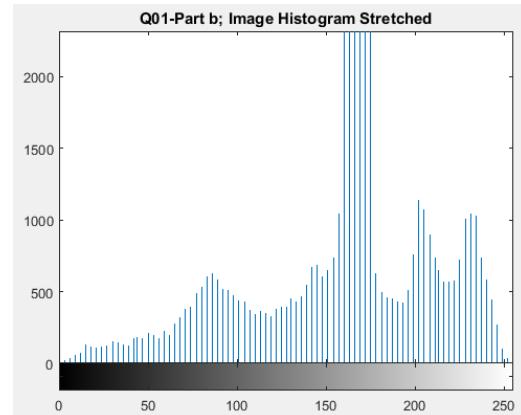
کار با contrast تصاویر

قسمت الف:

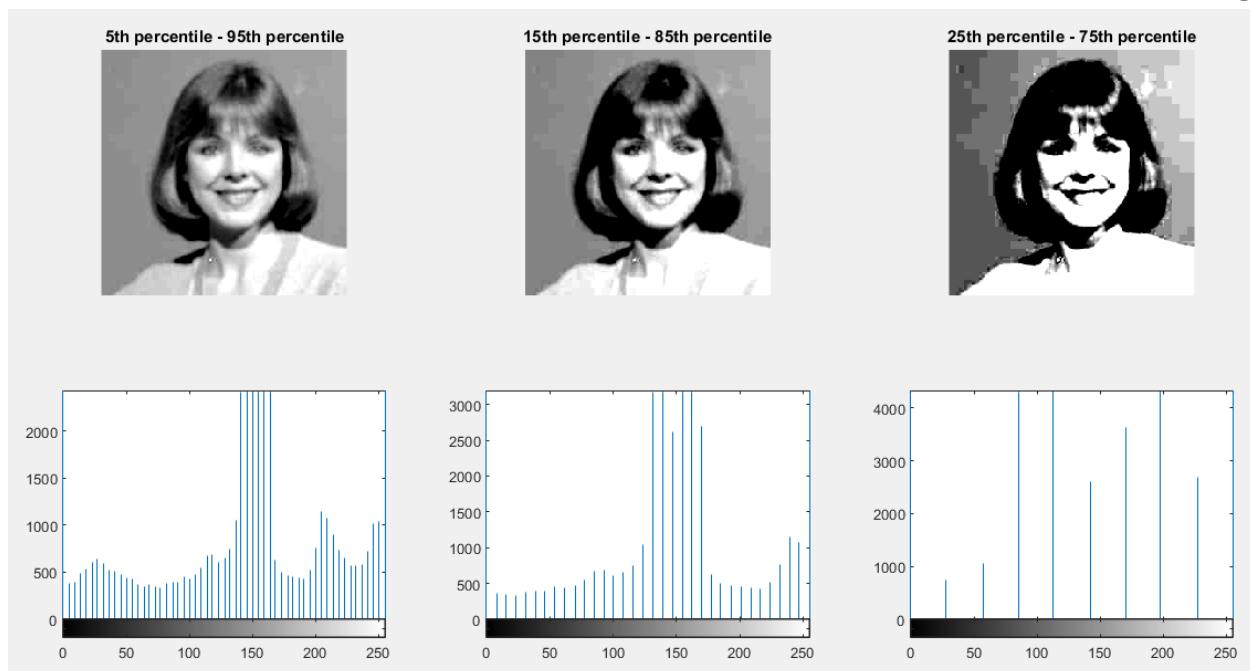


شکل ۱-۱: سمت چپ، تصویر wom.jpg می‌باشد که در سمت راست هیستوگرام آن با ۲۵۶ سطح روشنائی قابل مشاهده است. مشهود است که سطح روشنائی عمدی مقادیر پیکسل‌های تصویر حدوداً در بازه‌ی ۵۰ تا ۱۴۰ قرار داشته و مابقی سطوح روشنائی تقریباً بلااستفاده باقی مانده‌اند و در نتیجه نقاط خیلی تاریک و یا خیلی روشن در تصویر مشاهده نمی‌شود و به سبب آن جزئیات تصویر به سادگی قابل تفکیک نمی‌باشند.

قسمت ب:

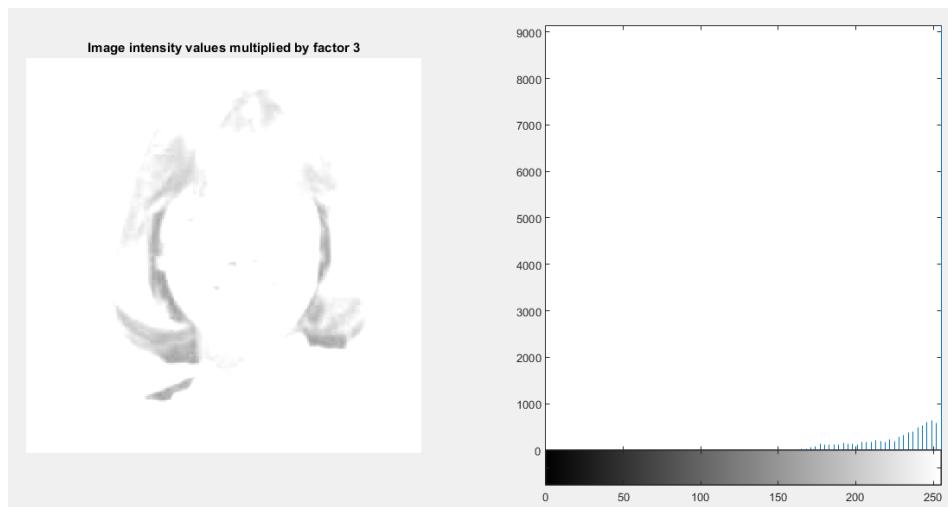


شکل ۱-۲: سمت چپ، تصویر wom.jpg بعد از افزایش بازه‌ی کانتراست می‌باشد که در سمت راست هیستوگرام آن با ۲۵۶ سطح روشنائی قابل مشاهده است. مشهود است که سطح روشنائی مقادیر پیکسل‌های تصویر در بازه‌ی وسیع‌تری پخش شده و به عبارتی تفکیک‌پذیری تصویر نهائی بالا رفته و جزئیات بیشتری از آن قابل برداشت است.

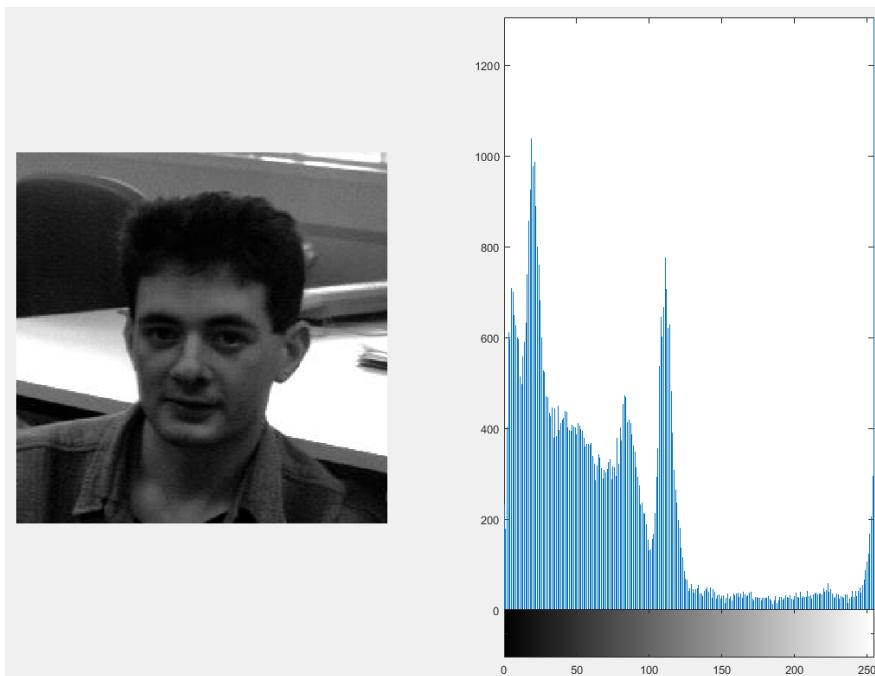


شکل ۱-۳: همانطور که مشاهده می‌شود در زیر هر تصویر حاصله در نظرگرفتن مقادیر مختلف برای پارامترهای c و d، هیستوگرام مربوط به آن نمایش داده شده است. نکته‌ی توجه آن است که با در نظرگرفتن صدک‌های محدودتر برای پارامترهای مذکور، مقادیر با سطوح بالاتر روشنائی در تصویر اولیه به بالاترین سطح یعنی ۲۵۵ و مقادیر با سطوح پائین‌تر روشنائی نیز همگی به پایین‌ترین سطح یعنی ۰ نگاشت می‌شوند؛ و مقادیر میانی این دو صدک در بازه‌ی ۰ تا ۲۵۵ توزیع می‌شوند. به عنوان مثال به ازای صدک‌های ۵ صدم (یعنی مقداری از سطح روشنائی که ۵ درصد پیکسل‌های تصویر سطح روشنائی آن‌ها از این مقدار کمتر یا مساوی است) و ۹۵ صدم (یعنی مقداری از سطح روشنائی که ۹۵ درصد پیکسل‌های تصویر سطح روشنائی آن‌ها از این مقدار کمتر یا مساوی است)، ۵ درصد اولیه از پیکسل‌ها همگی به صفر و ۵ درصد بالاتری همگی به ۲۵۵ نگاشت می‌شوند و پیکسل‌های بین این‌ها در بازه‌ی ۰ تا ۲۵۵ توزیع می‌شوند؛ در نتیجه مقدار زیادی از پیکسل‌های تصویر اطلاعاتشان از دست می‌رود. حال اگر این بازه همچنان محدودتر گردد تصویر به سمتی می‌رود که کاملاً حالت دودوئی داشته و عمدۀ پیکسل‌ها مانند تصویر اول از سمت راست، همگی یا مقدار سطح روشنائی ۰ و یا هم ۲۵۵ دارند و به عبارتی تصویر washedOut شده است و حالت نقاشی آب و رنگ پیدا کرده است.

قسمت د:



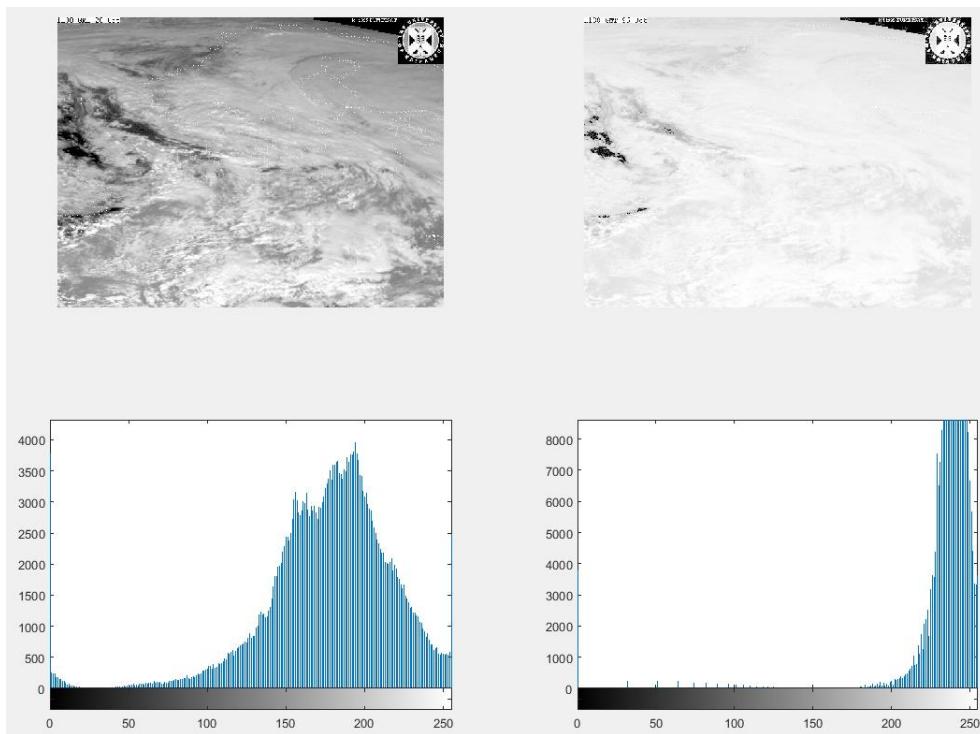
شکل ۱-۴: اگر بخواهیم برای بهبود کانتراست تصویر به جای اصطلاحاً کشش یا همان stretching کانتراست از یک ضریب یکسان (مثلاً سه) برای همه‌ی مقادیر سطح روشنائی پیکسل‌های تصویر استفاده کنیم، قطعاً سطح روشنائی بسیاری از پیکسل‌های تصویر اولیه با سطح روشنائی بالاتر، از مقدار بیشینه برای سطح روشنائی یعنی ۲۵۵ تجاوز کرده و به عبارتی به همان مقدار ۲۵۵ نگاشت می‌شوند و مقادیر پائین‌تر هم ارزش بالاتری پیدا کرده و در کل تصویر نهائی روشن‌تر می‌شود و در نتیجه اطلاعات بسیاری از تصویر از بین می‌رود. به عنوان مثال با استفاده از ضریب ۳ نتایج مطابق با تصویر بالا حاصل می‌گردد.



شکل ۵-۱: همانطور که از تصویر man.jpg و هیستوگرام آن پیداست تعداد پیکسل‌های با سطح روشنائی پایین در تصویر بسیار بیشتر از پیکسل‌های با سطح روشنائی بالا بوده و البته بخشی از پیکسل‌ها نیز دارای سطوح روشنائی می‌باشند.



شکل ۵-۲: عملگر لگاریتمی یک محدوده‌ی باریک از مقادیر سطح روشنائی پایین را به سطح روشنائی بالاتر نگاشت می‌کند و در عوض محدوده‌ی با سطوح روشنائی بالاتر را فشرده‌سازی می‌نماید. در نتیجه‌ی این عمل تصاویری که عمدتاً دارای پیکسل‌های با سطوح روشنائی پایین‌تر می‌باشند به تصاویر روشن‌تری تبدیل می‌شوند و به عبارتی کانتراست آن‌ها بهبود می‌یابد؛ که در اینجا در مورد تصویر man.jpg نیز همینطور می‌باشد.



شکل ۱-۷: همانطور که در مورد قسمت قبل اشاره شد، عملگر لگاریتمی سطوح روشنایی پایین تر را به سطوح بالاتر نگاشت کرده و سطوح بالاتر را نیز فشرده می‌کند که نتیجه‌ی این عمل بر روی تصاویر تیره سبب بهبود کانتراست می‌شود؛ اما برای تصاویر روشن تر مانند تصویر svs.jpg که دارای پیکسل‌های عمدتاً با سطح روشنایی بالا می‌باشد نه تنها سبب بهبود کانتراست تصویر نمی‌شود بلکه سبب از بین رفتن عمدی جزئیات تصویر اصلی نیز می‌گردد.

جواب سوال ۲

برابرسازی هیستوگرام تصویر به صورت دستی



شکل ۱-۸: تصویر حاصله از ماتریس 8×8 با سطوح روشنایی از 0 تا 7 که با تابع `mat2gray()` متلب و البته استفاده از تابع `imresize()` به صورت بالا درآمده است.

حال برای برابرسازی هیستوگرام در ابتدا باید هیستوگرام تصویر اولیه را به دست آوریم و سپس آن را نرمال‌سازی نموده و در نهایت تابع توزیع تجمعی تجربی (Empirical Distribution Function) را برای آن به دست آوریم. مراحل گفته شده به صورت زیر می‌باشند:

مقدار تابع توزیع تجمعی نرمال‌سازی شده ($\sum_{j=0}^k p_r(r_j)$)	مقدار نرمال شده (تقسیم بر تعداد کل پیکسل‌ها برابر ۶۴) ($p_r(r_k) = \frac{n_k}{64}$)	تعداد پیکسل‌های موجود در هر سطح (n _k)	شماره‌ی سطح روشنائی (r _k)
.0781	.0781	5	0
.0781	0	0	1
.0781	0	0	2
.0781	0	0	3
.4531	.3750	24	4
.7344	.2813	18	5
.9063	.1719	11	6
1.0000	.0938	6	7

حال با استفاده از فرمول زیر سطوح روشنائی مقصد را به دست می‌آوریم:

$$s_k = T(r_k) = [(L - 1) * \sum_{j=0}^k p_r(r_j)]$$

که در اینجا L معادل تعداد سطوح روشنائی تصویر اولیه و برابر ۸ می‌باشد. در نهایت داریم:

شماره‌ی سطح روشنائی مبدأ (r _k)	شماره‌ی سطح روشنائی مقصد (s _k)
1	0
1	1
1	2
1	3
3	4
5	5
6	6
7	7

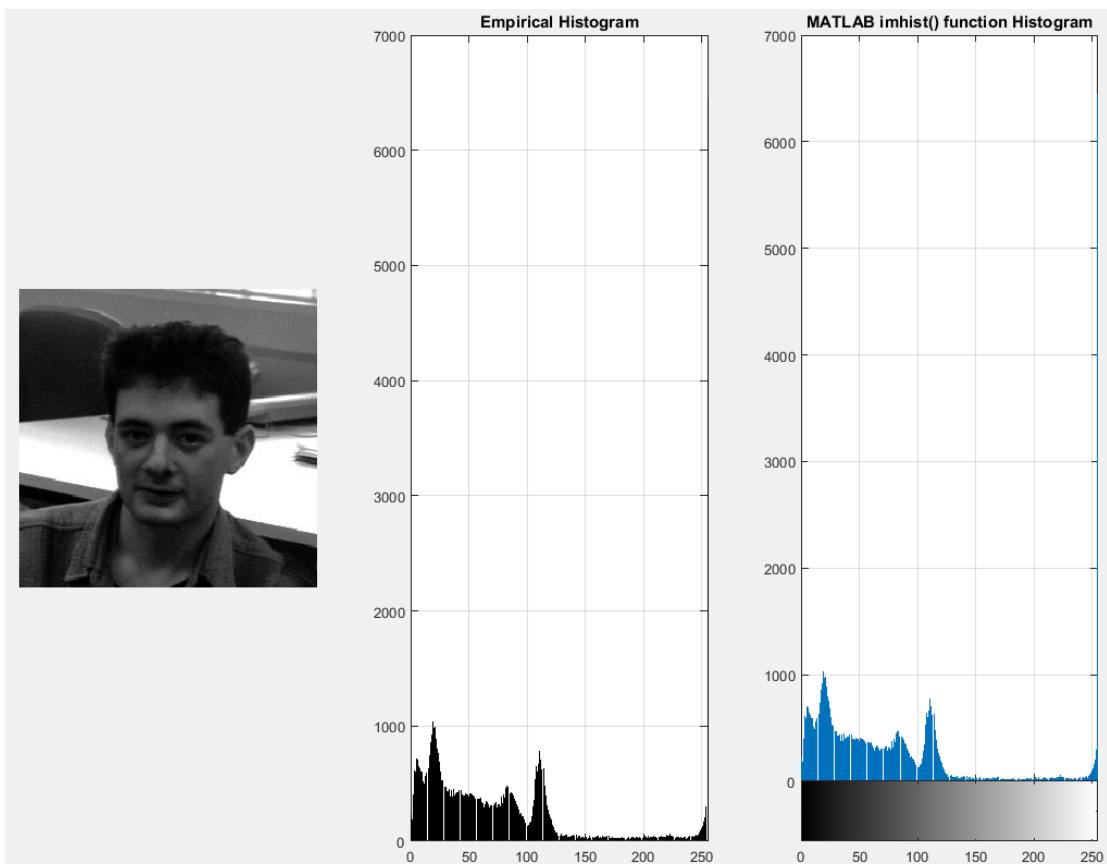
نتایج نهائی برابر سازی هیستوگرام تصویر ارائه شده به صورت زیر می‌باشد:

6	3	3	3	3	3	3	1
6	5	5	5	5	5	3	1
6	5	7	7	6	5	3	1
6	5	7	7	6	5	3	1
6	5	7	7	6	5	3	1
6	5	5	5	5	5	3	3
6	3	3	5	3	3	3	3
6	3	3	5	3	3	3	3

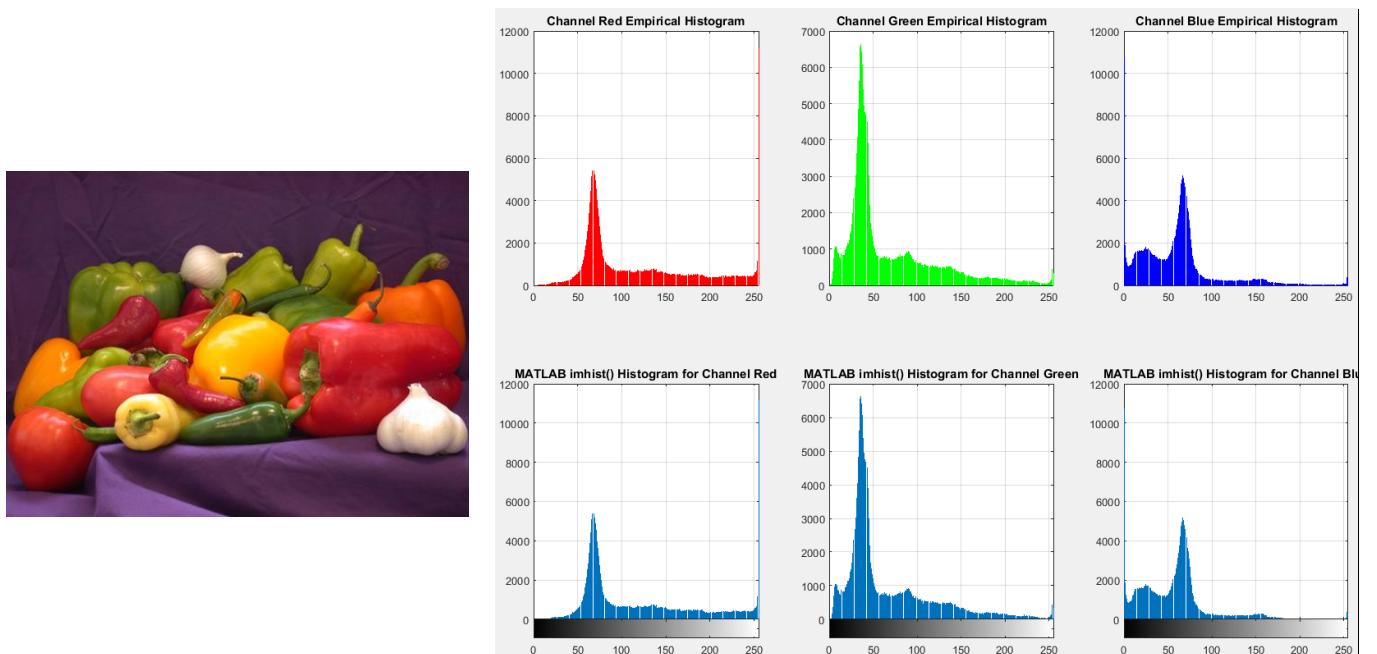


جواب سوال ۳

محاسبه‌ی هیستوگرام یک تصویر بدون استفاده از توابع آماده‌ی متلب



شکل ۱-۳: در سمت چپ تصویر ورودی man.jpg قرار دارد و در سمت راست آن دو نمودار هیستوگرام قابل مشاهده است که یکی هیستوگرام محاسبه شده به صورت تجربی و دیگری محاسبه شده با استفاده ازتابع آماده می باشد و البته با استفاده ازتابع آماده equal() متلب اطمینان حاصل شد که هر دو نمودار هستوگرام با یکدیگر کاملاً برابرند.



شکل ۱-۴: در سمت چپ تصویر ورودی peppers.png قرار دارد که یک تصویر RGB می باشد و در سمت راست آن دو ردیف نمودار هیستوگرام برای هر سه کanal قرمز، سبز و آبی قابل مشاهده است که یک ردیف هیستوگرام محاسبه شده به صورت تجربی و ردیف دیگر محاسبه شده با استفاده ازتابع آماده می باشد و البته با استفاده ازتابع آماده equal() متلب اطمینان حاصل شد که هر دو ردیف نمودار هستوگرام با یکدیگر کاملاً مطابقند.

قسمت ب:

با بررسی تابع `imhist` متلب مشخص می‌شود که این تابع با دریافت تصویر ورودی با تعداد سطوح روشنائی مشخص، تعداد پیکسل‌ها را در هر سطح روشنائی محاسبه نموده و پارامترهای خروجی آن نیز `counts` (معادل تعداد پیکسل‌های موجود در هر سطح روشنائی) و `binLocations` (مکان‌های مشخص برای هر سطح روشنائی) می‌باشد. تعداد سطوح روشنائی تصویر ورودی توسط خود تابع از روی نوع تصویر مشخص می‌گردد؛ به عنوان مثال اگر تصویر ورودی یک تصویر `graylevel` باشد، تعداد ۲۵۶ مکان یا اصطلاحاً `bin` به کار گرفته می‌شود و اگر تصویر ورودی دودوئی باشد، تنها تعداد ۲ `bin` به کار گرفته می‌شود.

قسمت ج:

همانطور که در قسمت الف اشاره گردید، با استفاده از تابع `isequal` مشخص شد که نتایج تجربی و نتایج حاصله از تابع آماده‌ی متلب کاملاً یکسان بودند.

جواب سوال ۴

برابرسازی هیستوگرام یک تصویر بدون استفاده از توابع آماده‌ی متلب

قسمت الف:



شکل ۱-۴: در هر دو تصویر در سمت چپ تصاویر اصلی قرار دارند و در سمت راست آن‌ها هر دو خروجی حاصل از برابرسازی هیستوگرام تصاویر با استفاده از تابع ساختگی و تابع آماده‌ی متلب قرار دارد که همانطور که قابل مشاهده است تفاوت چندانی میان خروجی تابع ساختگی با خروجی تابع آماده‌ی متلب مشاهده نمی‌شود.

قسمت ب:

با بررسی تابع `histeq` می‌شخص می‌گردد که این تابع با دریافت یک تصویر به طور پیش‌فرض تصویر می‌کند که تعداد سطوح روشنائی تصویر ورودی برابر ۶۴ سطح می‌باشد، که اگر تعداد سطوح روشنائی تصویر ورودی از این مقدار بیشتر باشد، هیستوگرام تصویر خروجی مسطح‌تر خواهد بود و علت این امر نیز آن است که محدوده‌ی توزیع پیکسل‌ها بسته‌تر شده است. حال اگر بخواهیم برابرسازی هیستوگرام تصویر ورودی روی تعداد سطوح روشنائی به اندازه‌ی همان تصویر ورودی صورت گیرد، باید تابع `mbوطه` را به صورت `histeq(I,n)` فراخوانی نماییم که `I` همان تصویر ورودی (`Intensity Image`) و `n` نیز مشخصه‌ی تعداد سطوح روشنائی برای هیستوگرام تصویر ورودی می‌باشد که همانطور که بیان شد هر چه از تعداد سطوح روشنائی تصویر ورودی کمتر باشد هیستوگرام خروجی مسطح‌تر شده و بالعکس هرچه از تعداد سطوح روشنائی تصویر ورودی بیشتر باشد به هر کدام از این تعداد سطوح روشنائی به سختی تعداد یکسانی پیکسل تعلق خواهد گرفت.

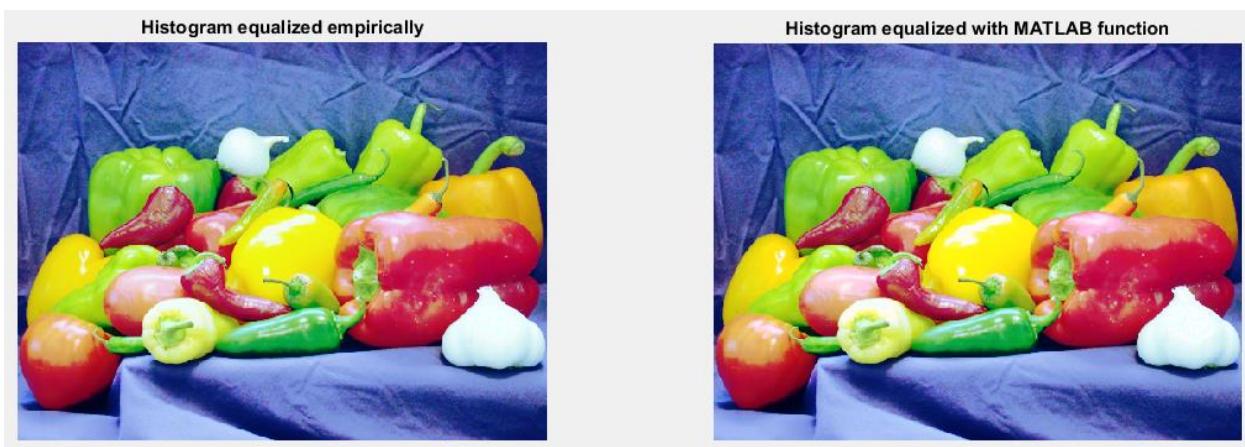
قسمت ج:

در اینجا برای برابری نتایج حاصله ازتابع ساختگی وتابع آماده متلب برای برابرسازی هیستوگرام تصویر، از قطعه کد زیر استفاده می‌کنیم:

```
sum(sum(imgEquMat==imgEquEmp))
```

که مقدار خروجی آن برای هر دو تصویر قسمت الف، مقداری غیر صفر می‌باشد. همانطور می‌توانیم مانند سؤال قبلی از تابع آماده `isequal()` استفاده کنیم که در اینصورت نیز مقدار صفر را دریافت می‌کنیم؛ و این مسئله بیانگر آن است که نتایج تجربی و خروجی متلب با یکدیگر یکسان نبوده و علت آن نیز می‌تواند آن باشد که ما در قطعه کد دستی، در قسمتی که خروجی تجمعی نرمال شده حاصل شده و مقادیر اعشاری دارد، آن را در مقدار $L-1$ برابر تعداد سطوح روشنائی تصویر ورودی می‌باشد) ضرب می‌نماییم که در نتیجه‌ی این عمل باز هم تعدادی عدد اعشاری حاصل می‌گردد و باید به اعداد صحیح گرد شوند که برای این کار از تابع `round()` استفاده می‌کنیم؛ گمان می‌رود که علت نایابری نتایج حاصله این مسئله باشد؟!

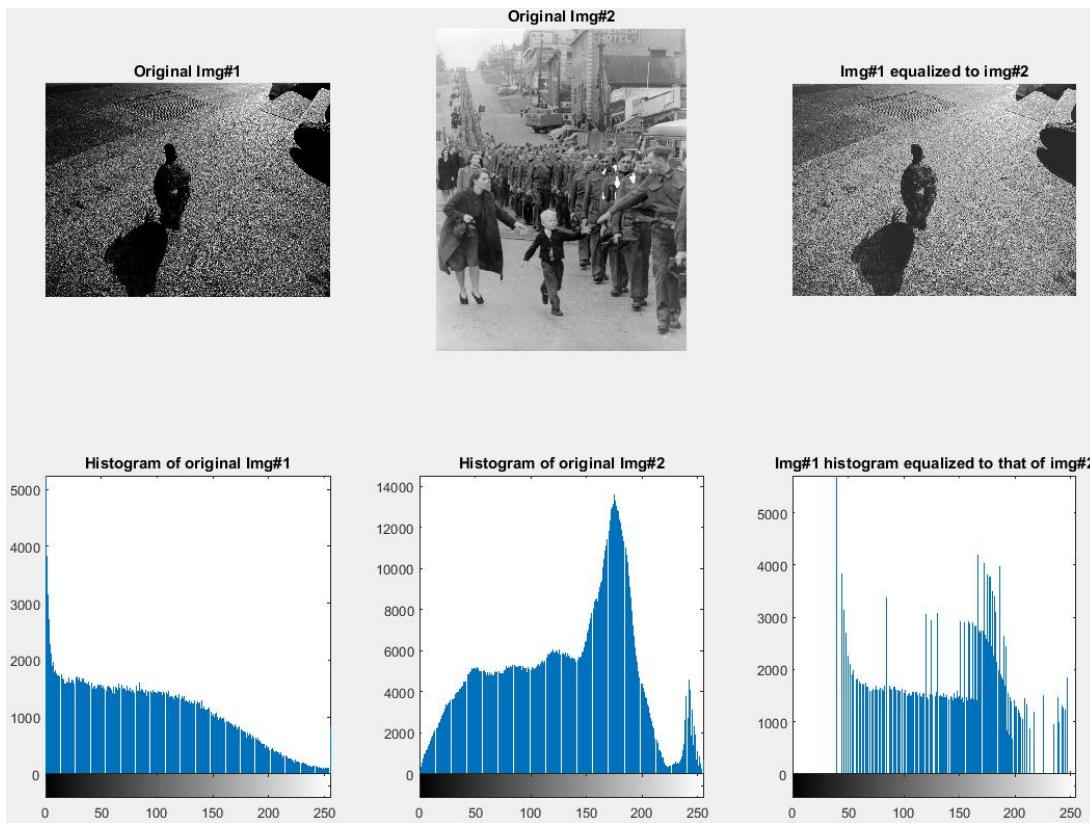
قسمت د:



شکل ۲-۴: همانطور که در قطعه کد پیاده‌سازی شده نیز مشاهده می‌شود، برای تصاویر RGB که ColorType آنها توسط متلب 'truecolor' شناخته می‌شود نه 'grayscale'، باید برای هر سه کanal مقادیر مربوطه را جداگانه در متغیرهای متفاوت قرار داده و در نهایت با هر کدام از آنها برای برابرسازی هیستوگرام مانند یک تصویر grayscale رفتار نمائیم. در نهایت بعد از اعمال برابرسازی هیستوگرام باید هر سه کanal جدید را با هم ترکیب کنیم تا تصویر نهایی با هیستوگرام برابرشده حاصل گردد.

جواب سوال ۵

آشنائی با تطبیق هیستوگرام یک تصویر با هیستوگرام یک تصویر دیگر



شکل ۱-۵: همانطور که مشاهده می‌شود هیستوگرام تصویر `pigeon.jpg` در تلاش برای برابر شدن با هیستوگرام تصویر `soldier.jpg` تا حدی مسطح‌تر شده است، چرا که بیشتر پیکسل‌های تصویر `pigeon.jpg` با توجه به هیستوگرام آن تاریک بوده و تصویر `soldier.jpg` نیز دارای عدالت پیکسل‌های با مقدار روشن‌تر می‌باشد؛ لذا همانطور که در تصویر خروجی مشاهده می‌گردد تعداد پیکسل‌های با سطح روشن‌تر بالا افزایش یافته و در نتیجه تصویر نهایی روشن‌تر شده است.

جواب سوال ۶

آشنائی با حل مسائل فیلترینگ

رونده کانولوشن یک فیلتر h دو بعدی گسسته در یک پنجره f دو بعدی گسسته به این صورت است که باید فیلتر مربوطه را نسبت به مبدأ آن دوران دهیم به طوری که مختصات هر نقطه‌ی $(x, -y)$ با زوج مرتب (x, y) جایگزین گردد. سپس مرکز فیلتر را روی هر کدام از نقاط پنجره‌ی ورودی قرار داده و مقادیر متناظر را در هم ضرب می‌کنیم و به جای مقادیری از فیلتر که بیرون پنجره‌ی f قرار می‌گیرند، مقدار معدل صفر در نظر می‌گیریم. در نهایت مقادیر حاصلضرب‌های جزئی را با هم جمع کرده و در پیکسل مورد نظر در پنجره‌ی f قرار می‌دهیم.

قسمت الف:

در اینجا چون کرنل متقارن است، دوران آن تأثیری ندارد. محاسبات به صورت زیر می‌باشد:

1,0	2,0	1,0				
2,0	4,0	2,0	0	0	0	
1,0	2,0	1,0	1	0	0	
	0	1	2	1	0	
	0	0	3	0	0	
	0	0	0	0	0	

0	0	0	0	0	0	
0	0	1	0	0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

1,0	2,0	1,0				
2,0	4,0	2,0	0	0	0	
1,0	2,0	1,1	0	0		
	0	1	2	1	0	
	0	0	3	0	0	
	0	0	0	0	0	

0	1	0	0	0	0	
0	0	1	0	0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

	1,0	2,0	1,0			
0	2,0	4,0	2,0	0		
0	1,0	2,1	1,0	0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

0	1	2	0	0	0	
0	0	1	0	0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

	1,0	2,0	1,0			
0	0	2,0	4,0	2,0		
0	0	1,1	2,0	1,0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

0	1	2	1	0	0	
0	0	1	0	0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

		1,0	2,0	1,0		
0	0	0	2,0	4,0	2,0	
0	0	1	1,0	2,0	1,0	
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

0	1	2	1	0	0	
0	0	1	0	0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

1,0	2,0	1,0	0	0	0	
2,0	4,0	2,0	1	0	0	
1,0	2,0	1,1	2	1	0	
	0	0	3	0	0	
	0	0	0	0	0	

0	1	2	1	0	0	
1	0	1	0	0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

		1,0	2,0	1,0		
2,0	4,0	2,1	0	0		
1,0	2,1	1,2	1	0		
	0	0	3	0	0	
	0	0	0	0	0	

0	1	2	1	0	0	
1	6	10	6	0		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

0	1,0	2,0	1,0	0	0	
0	0	2,0	4,1	2,0		
0	1,1	2,2	1,1	0		
0	0	3	0	0		
0	0	0	0	0		

0	1	2	1	0	0	
1	6	10	6	1		
0	1	2	1	0		
0	0	3	0	0		
0	0	0	0	0		

			0	0	0	
1,0	2,0	1,0	1	0	0	
2,0	4,0	2,1	2	1	0	
1,0	2,0	1,0	3	0	0	
	0	0	0	0	0	

0	1	2	1	0	0	
1	6	10	6	1		
2	12	20	12	2		
0	0	3	0	0		
0	0	0	0	0		

	0	0	0	0	0	
	0	0	1	0	0	
1,0	2,0	1,1	2	1	0	
2,0	4,0	2,0	3	0	0	
1,0	2,0	1,0	0	0	0	

0	1	2	1	0	0	
1	6	10	6	1		
2	12	20	12	2		
1	0	3	0	0		
0	0	0	0	0		

				0	0	0
0	1	0	1,0	2,0	1,0	
0	1	2	2,1	4,0	2,0	
0	0	3	1,0	2,0	1,0	
0	0	0	0	0	0	

0	1	2	1	0	0	
1	6	10	6	1		
2	12	20	12	2		
0	0	3	0	0		
0	0	0	0	0		

<table

0	0	0	0	0
0	0	1	0	0
1,0	2,1	1,2	1	0
2,0	4,0	2,3	0	0
1,0	2,0	1,0	0	0

0	1	2	1	0
1	6	10	6	1
2	12	20	12	2
1	10	3	0	0
0	0	0	0	0

0	0	0	0	0
0	0	1	0	0
0	1,1	2,2	1,1	0
0	2,0	4,3	2,0	0
0	1,0	2,0	1,0	0

0	1	2	1	0
1	6	10	6	1
2	12	20	12	2
1	10	18	18	0
0	0	0	0	0

0	0	0	0	0
0	0	1	0	0
0	1	1,2	2,1	1,0
0	0	2,3	4,0	2,0
0	0	1,0	2,0	1,0

0	1	2	1	0
1	6	10	6	1
2	12	20	12	2
1	10	18	10	0
0	0	0	0	0

0	0	0	0	0
0	0	1	0	0
0	1	2	1,1	2,0
0	0	3	2,0	4,0
0	0	1,0	2,0	1,0

0	1	2	1	0
1	6	10	6	1
2	12	20	12	2
1	10	18	10	1
0	0	0	0	0

0	0	0	0	0
0	0	1	0	0
0	1	2	1	0
1,0	2,0	1,0	3	0
2,0	4,0	2,0	0	0
1,0	2,0	1,0		

0	1	2	1	0
1	6	10	6	1
2	12	20	12	2
1	10	18	10	1
0	0	0	0	0

0	0	0	0	0
0	0	1	0	0
0	1	2	1	0
1,0	2,0	1,3	0	0
2,0	4,0	2,0	0	0
1,0	2,0	1,0		

0	1	2	1	0
1	6	10	6	1
2	12	20	12	2
1	10	18	10	1
0	3	6	3	0

0	0	0	0	0
0	0	1	0	0
0	1	2	1	0
0	1,0	2,3	1,0	0
0	2,0	4,0	2,0	0
1,0	2,0	1,0		

0	1	2	1	0
1	6	10	6	1
2	12	20	12	2
1	10	18	10	1
0	3	6	0	0

0	0	0	0	0
0	0	1	0	0
0	1	2	1	0
0	0	1,3	2,0	1,0
0	0	2,0	4,0	2,0
1,0	2,0	1,0		

0	1	2	1	0
1	6	10	6	1
2	12	20	12	2
1	10	18	10	1
0	3	6	3	0

Output Image

10	11	9	25	22
8	10	9	26	28
9	99	9	24	25
11	11	12	23	22
10	11	9	22	25

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

=

4.3	6.3	10.0	13.2	11.2
16.3	19.3	24.6	19.6	16.6
16.4	19.7	24.7	19.7	16.4
16.7	20.1	24.4	19.0	15.6
4.7	7.1	9.7	12.5	10.2

Input image

Conv.

Mean mask

Output image

قسمت ج:

برای اعمال فیلتر میانه مرکز یک فیلتر 3×3 را روی هر یک از پیکسل‌های تصویر حرکت داده و پیکسل‌هایی از پنجره f را که با فیلتر همپوشانی دارند را به ترتیب صعودی یا نزولی مرتب کرده و میانه‌ی این عناصر یعنی عنصر پنجم را جایگزین پیکسل مربوطه از پنجره f می‌نماییم. در مورد نواحی لبه نیز از عمل zeroPadding استفاده می‌کنیم، به این معنی که آن قسمت‌هایی از فیلتر که بیرون پنجره قرار می‌گیرد را معادلشان پیکسل با مقدار صفر در نظر می‌گیریم. نتایج به صورت زیر است:

10	11	9	25	22
8	10	9	26	28
9	99	9	24	25
11	11	12	23	22
10	11	9	22	25

Input image

After applying median filter =

0	9	9	9	0
9	9	11	24	24
9	10	12	23	23
10	11	12	22	22
0	10	11	12	0

Output image

برخی از مراحل به صورت زیر خواهد بود:

0	0	0	0	0	0	0	0	0	0
0	10	11	9	25	22	0	0	10	11
0	8	10	9	26	28	0	0	8	10
0	9	99	9	24	25	0	0	9	99
0	11	11	12	23	22	0	0	11	11
0	10	11	9	22	25	0	0	10	11
0	0	0	0	0	0	0	0	0	0

0	11	9	25	22
8	10	9	26	28
9	99	9	24	25
11	11	12	23	22
10	11	9	22	25

0	0	0	0	0	0	0	0	0	0
0	10	11	9	25	22	0	0	8	10
0	8	10	9	26	28	0	0	9	99
0	9	99	9	24	25	0	0	11	11
0	11	11	12	23	22	0	0	10	11
0	10	11	9	22	25	0	0	10	11
0	0	0	0	0	0	0	0	0	0

0	9	9	9	0
9	9	11	24	24
9	10	12	23	23
10	11	12	22	22
0	10	11	12	0

قسمت ۵:

هدف از unsharp masking پررنگ کردن جزئیات ریز در یک تصویر و یا هم تقویت کردن جزئیاتی از تصویر است که به دلایلی از جمله خطای یا هم اثرات طبیعی ناشی از سبک خاصی از تصویربرداری، کدر شده‌اند. در اینجا چون فیلتر مربوطه (فیلتر تیزکننده لaplacian استاندارد با $A=1$) متقارن می‌باشد دوران آن تأثیری در نتیجه‌ی نهائی ندارد. داریم:

10	11	9	25	22
8	10	9	26	28
9	8	9	24	25
11	11	12	23	22
10	11	9	22	25

Input image

*

0	-1	0
-1	5	-1
0	-1	0

=

31	26	0	68	57
11	14	-9	44	67
18	1	-8	37	51
25	13	8	35	37
28	25	0	53	81

Output image

Conv.

Laplacian jernel
with A=1

جواب سوال ۷

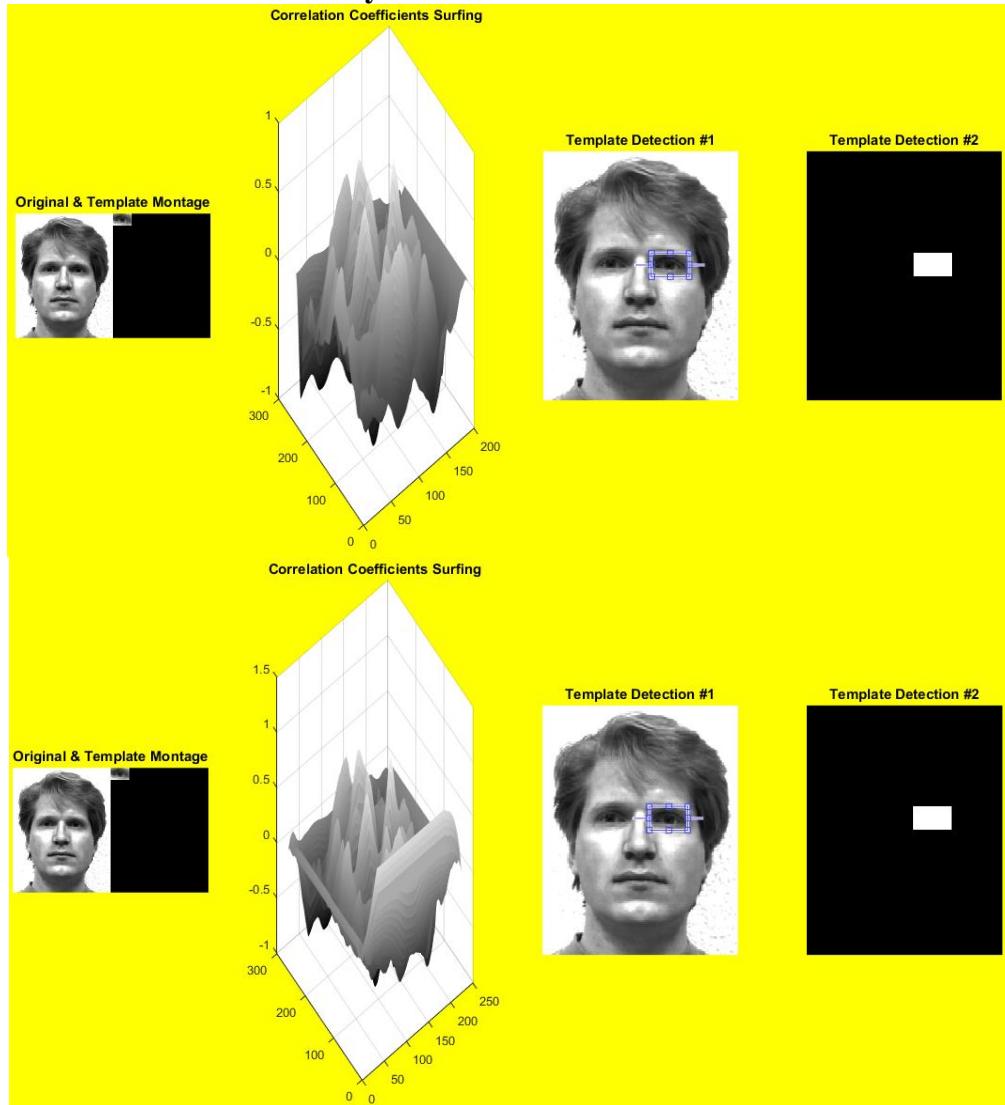
آشنائی با شیوه‌ی تشخیص یک الگو در یک تصویر

قسمت الف:

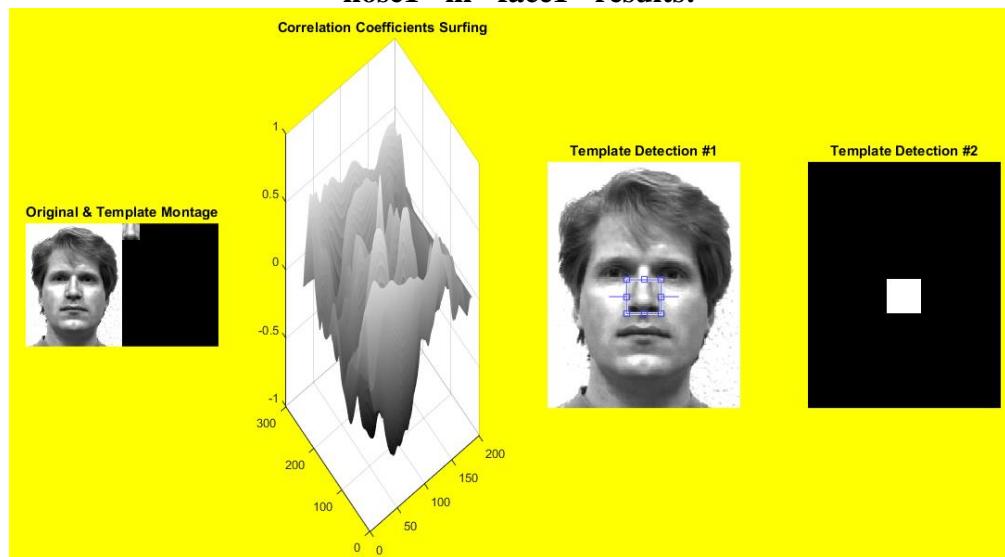
در اینجا برای کشف یک الگو در یک تصویر از یک روش ابداعی برای NCC و نیز ازتابع آماده‌ی normxcross2 استفاده می‌کنیم. در روش ابداعی اگر ابعاد تصویر و الگو به ترتیب $N^m \times N^n$ باشند، ابتدا ماتریس تصویر را به ابعاد جدید $(M+m-1) \times (N+n-1)$ گسترش داده و نواحی خالی را با صفر پر می‌کنیم (که به این عمل اصطلاحا zeroPadding می‌گویند؛ سپس یک ماتریس جدید به نام corrMat با ابعاد همان تصویر اصلی درست می‌کنیم که قرار است در آن خروجی حاصل از تابع آماده‌ی corr2 را ذخیره نمائیم؛ به این ترتیب که در هر نقطه (x,y) از تصویر اصلی یک تکه از تصویر (اصطلاحا batch) را با ابعاد $(x+m-1, y+n-1)$ جدا کرده و خروجی حاصل از فراخوانی تابع corr2 را در نقطه‌ی آغازی قسمت متناظر در ماتریس corrMat ذخیره می‌نماییم. در نهایت قسمتی از ماتریس corrMat که مقدار بیشینه دارد همان نقطه‌ی آغازی قسمت تطابق (detection) می‌باشد که آن را با یک مستطیل (مطابق توضیحات help مطلب در مورد تابع normxcross2) و یا هم الگوی دودوئی نمایش می‌دهیم و همینطور با استفاده از تابع surf مطلب، ماتریس corrMat را رسم می‌نماییم که در آن برابری نقطه‌ی ماسکیم خروجی تابع detection با ابتدای ماتریس surf کاملاً مشهود است. در مورد هر کدام از الگوها ابتدا نتایج حاصله از روش ابداعی و سپس نتایج حاصله از تابع آماده‌ی متلب را قید می‌نماییم:

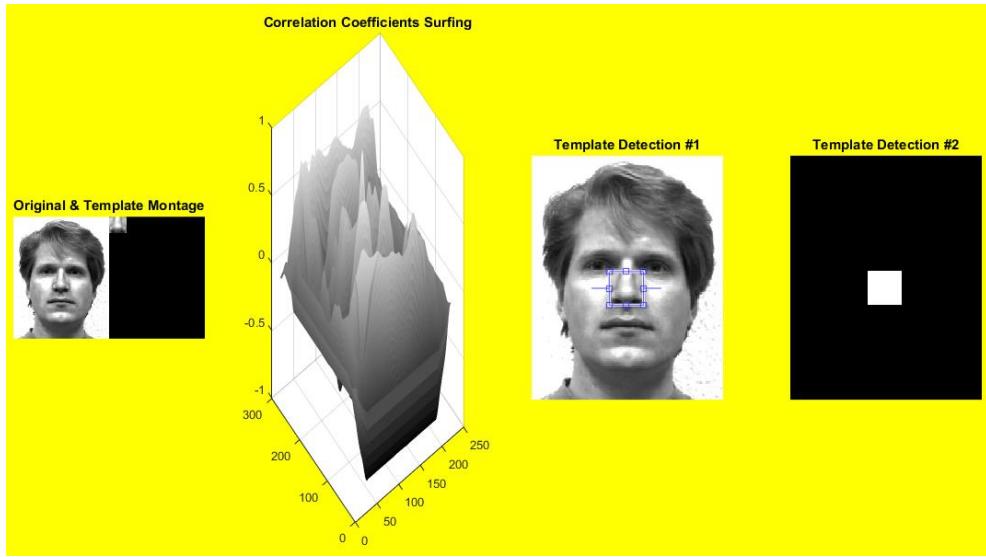
i) کشف الگوهای face1.pgm، nose1.pgm و eye1.pgm در تصویر

"eye1" in "face1" results:

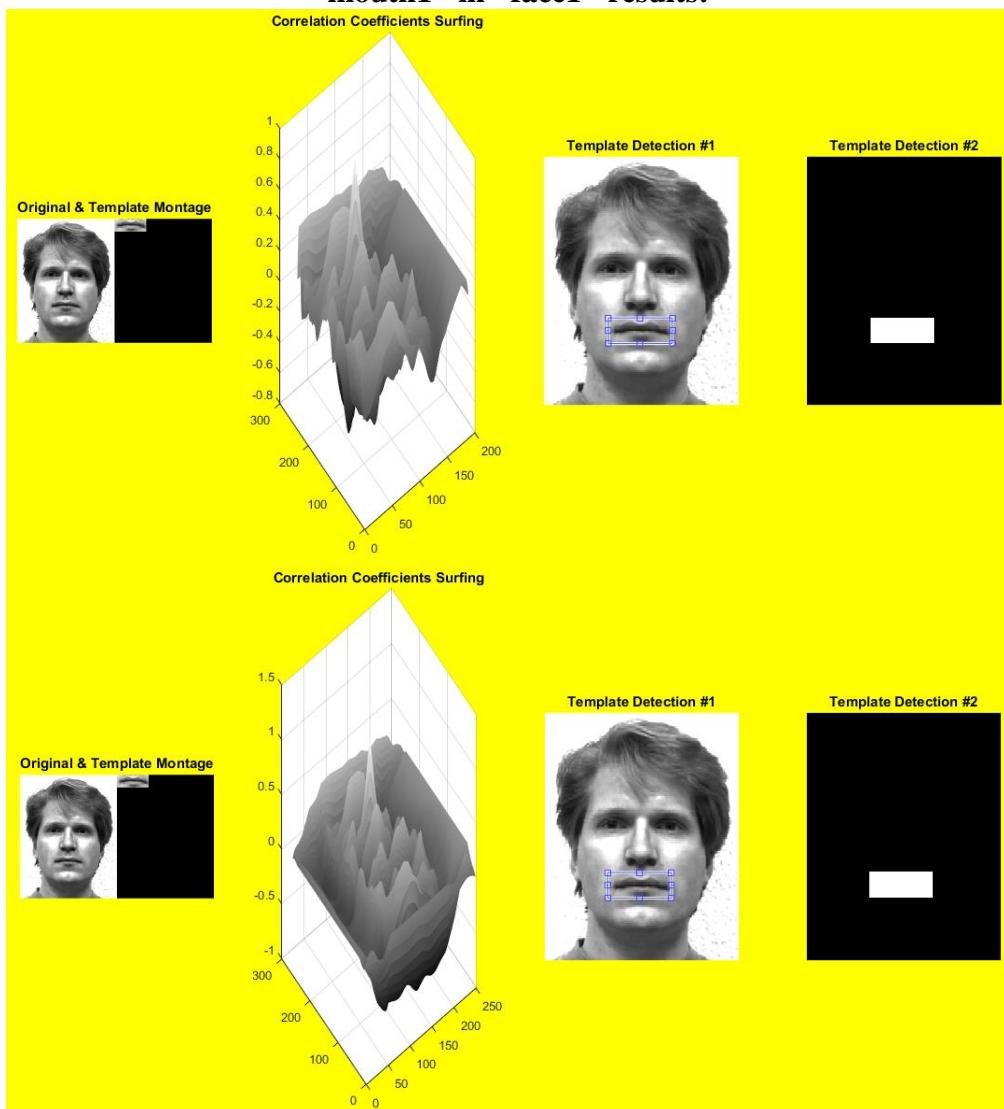


"nose1" in "face1" results:



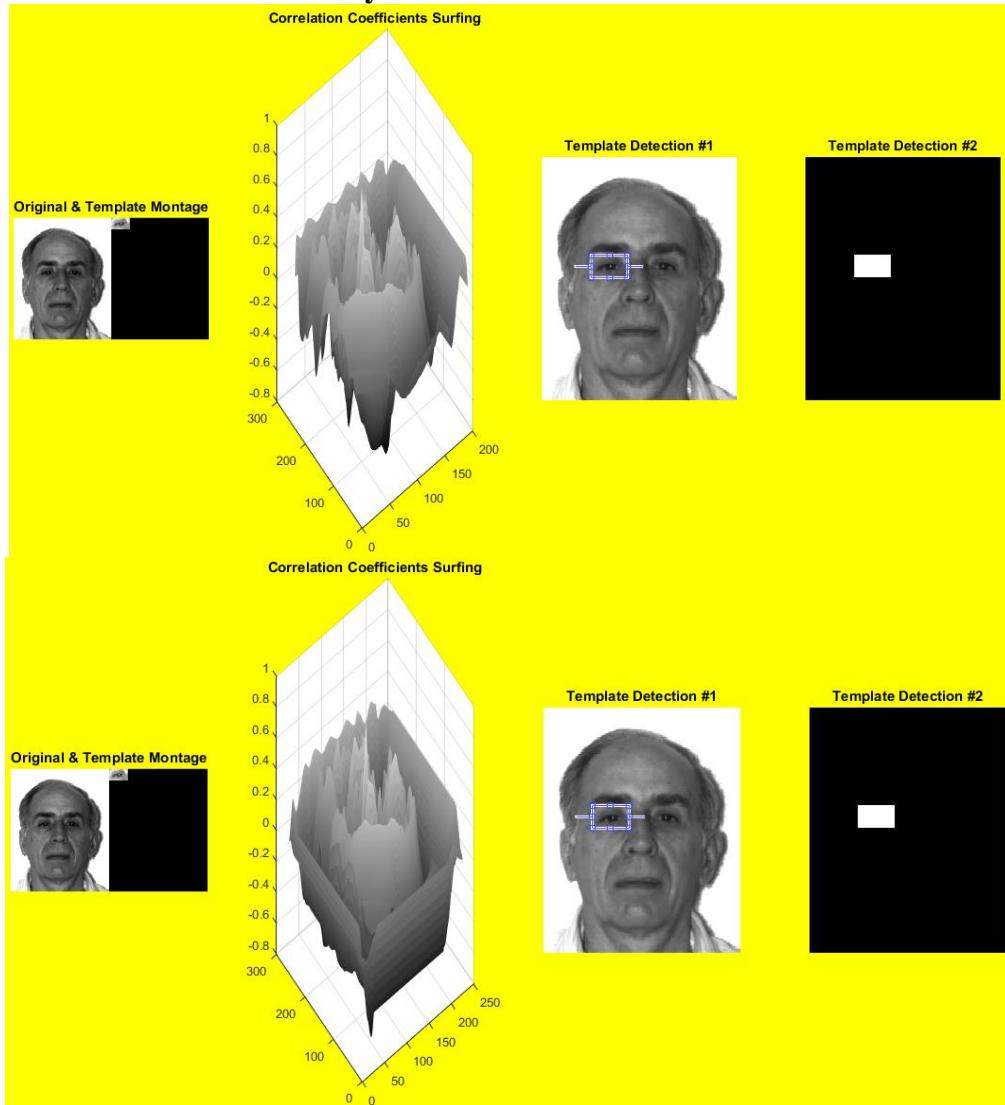


"mouth1" in "face1" results:

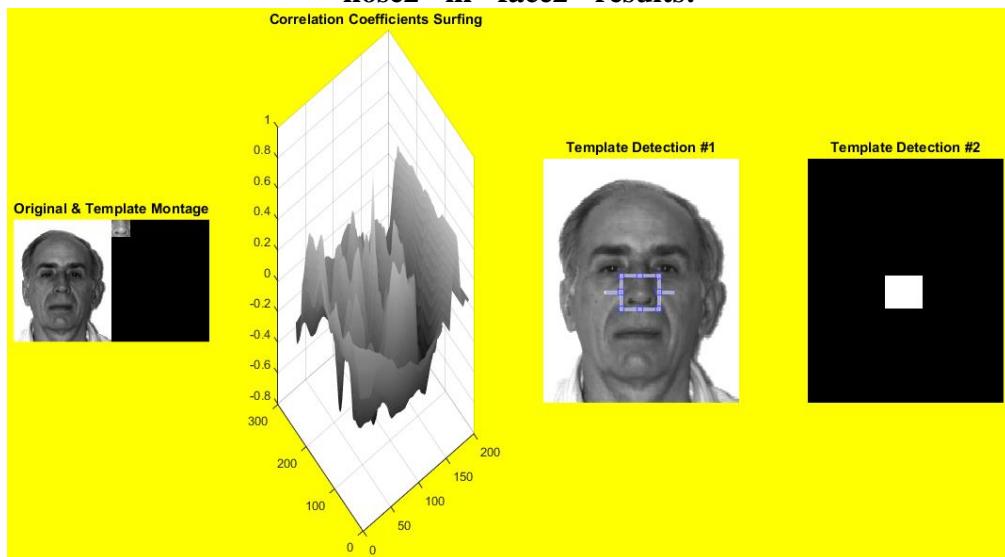


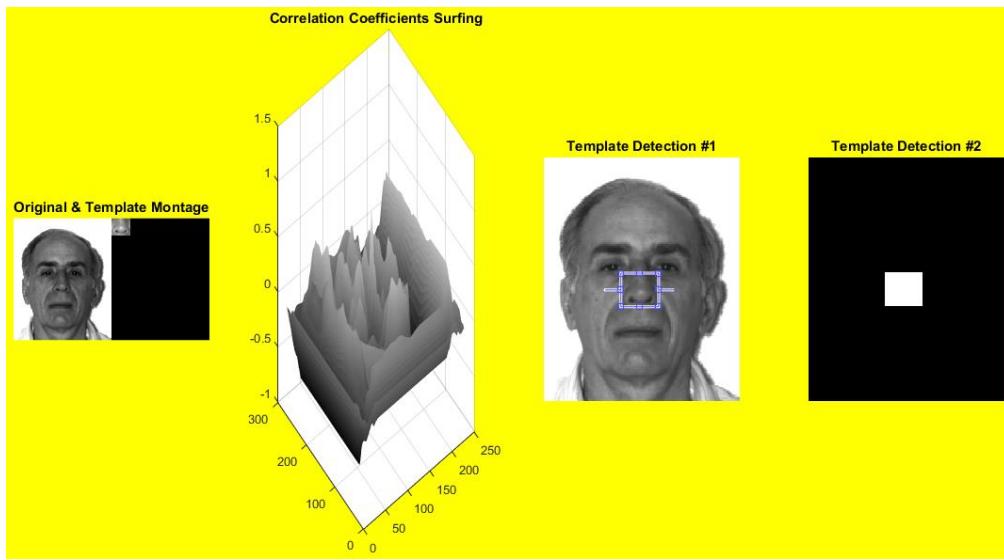
ii) کشف الگوهای .face2.pgm در تصویر mouth2.pgm ، nose2.pgm ، eye2.pgm

"eye2" in "face2" results:

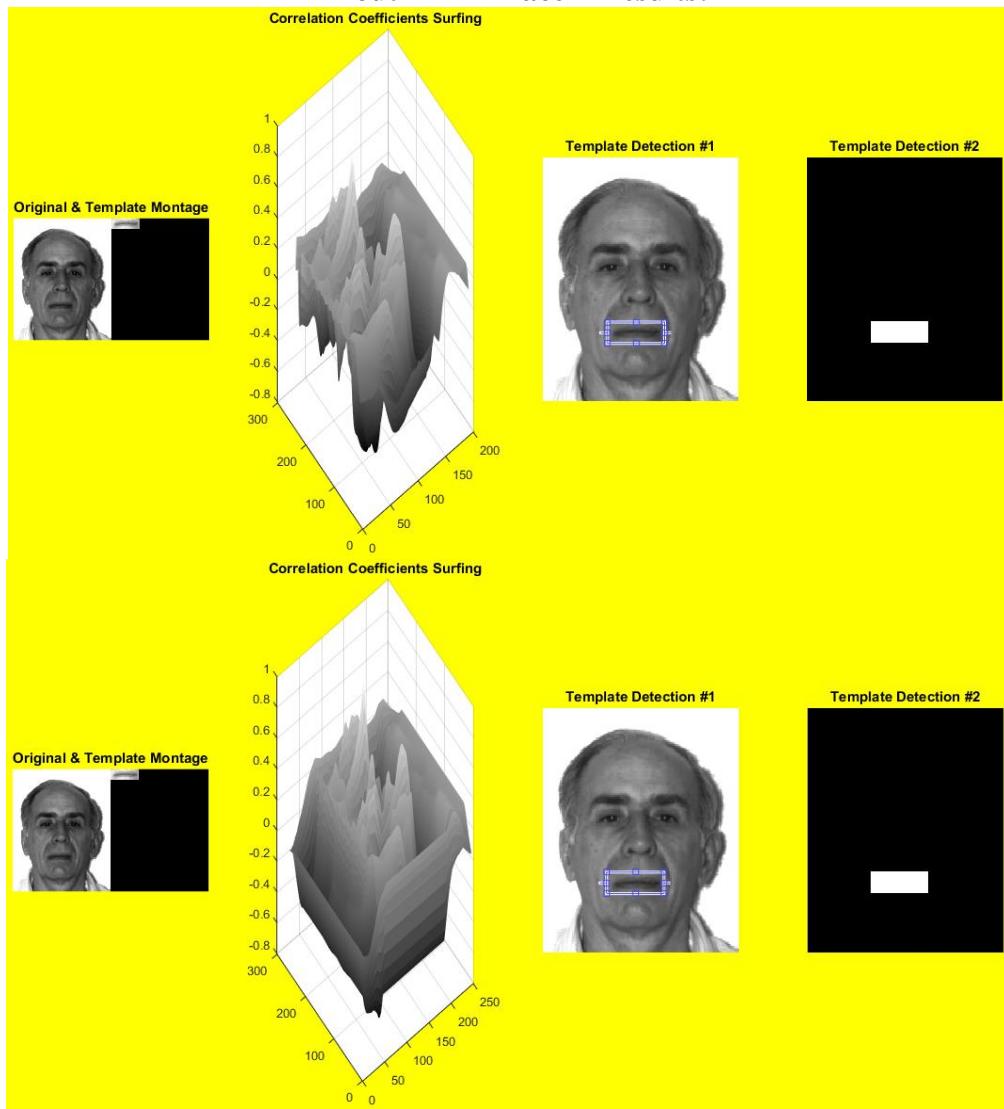


"nose2" in "face2" results:





"mouth2" in "face2" results:

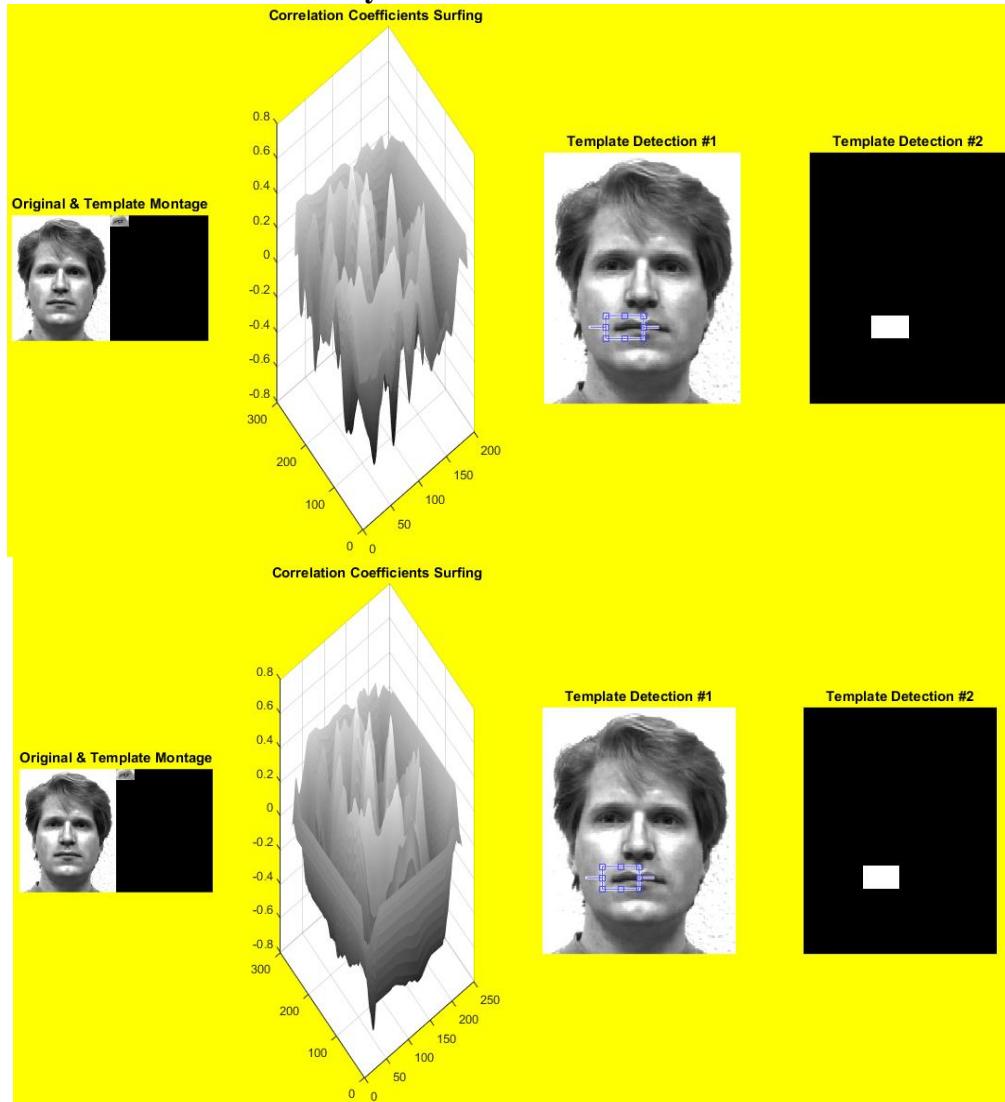


قسمت ب:

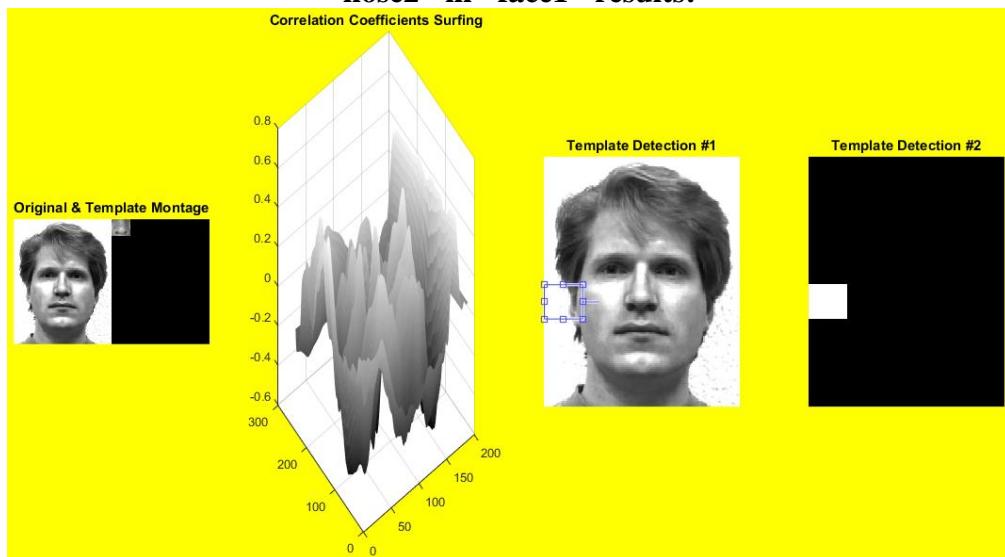
در اینجا نیز مانند قسمت الف برای کشف یک الگو در یک تصویر از یک روش ابداعی برای NCC و نیز از تابع آماده‌ی normxcross2 استفاده می‌کنیم. در مورد هر کدام از الگوها ابتدا نتایج حاصله از روش ابداعی و سپس نتایج حاصله از تابع آماده‌ی متلب را قید می‌نماییم:

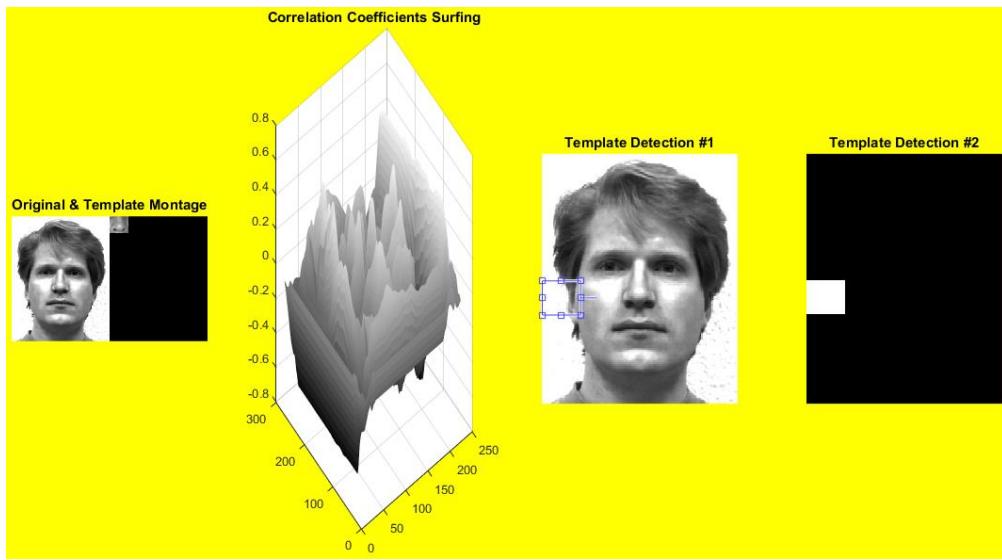
i) کشف الگوهای face1.pgm , mouth2.pgm , nose2.pgm , eye2.pgm در تصویر

"eye2" in "face1" results:

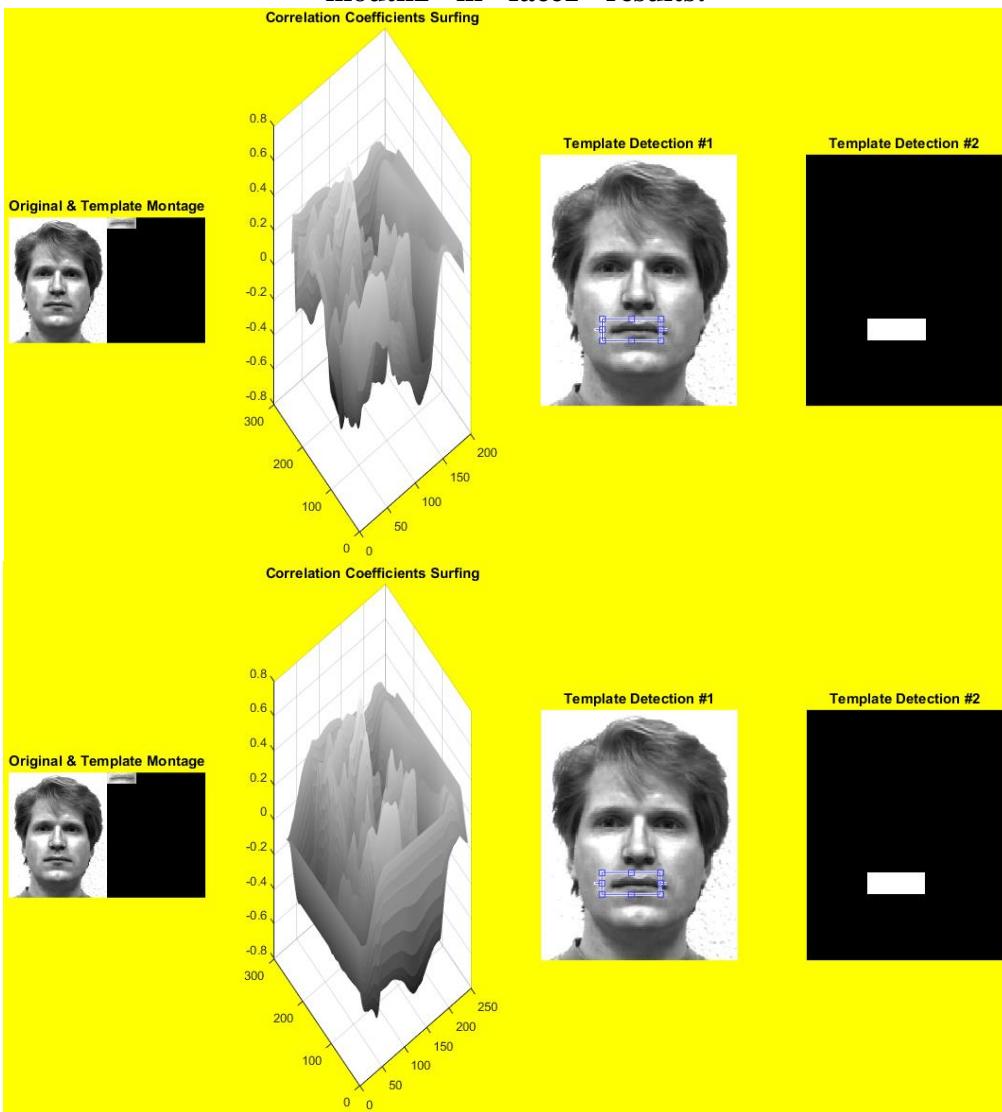


"nose2" in "face1" results:



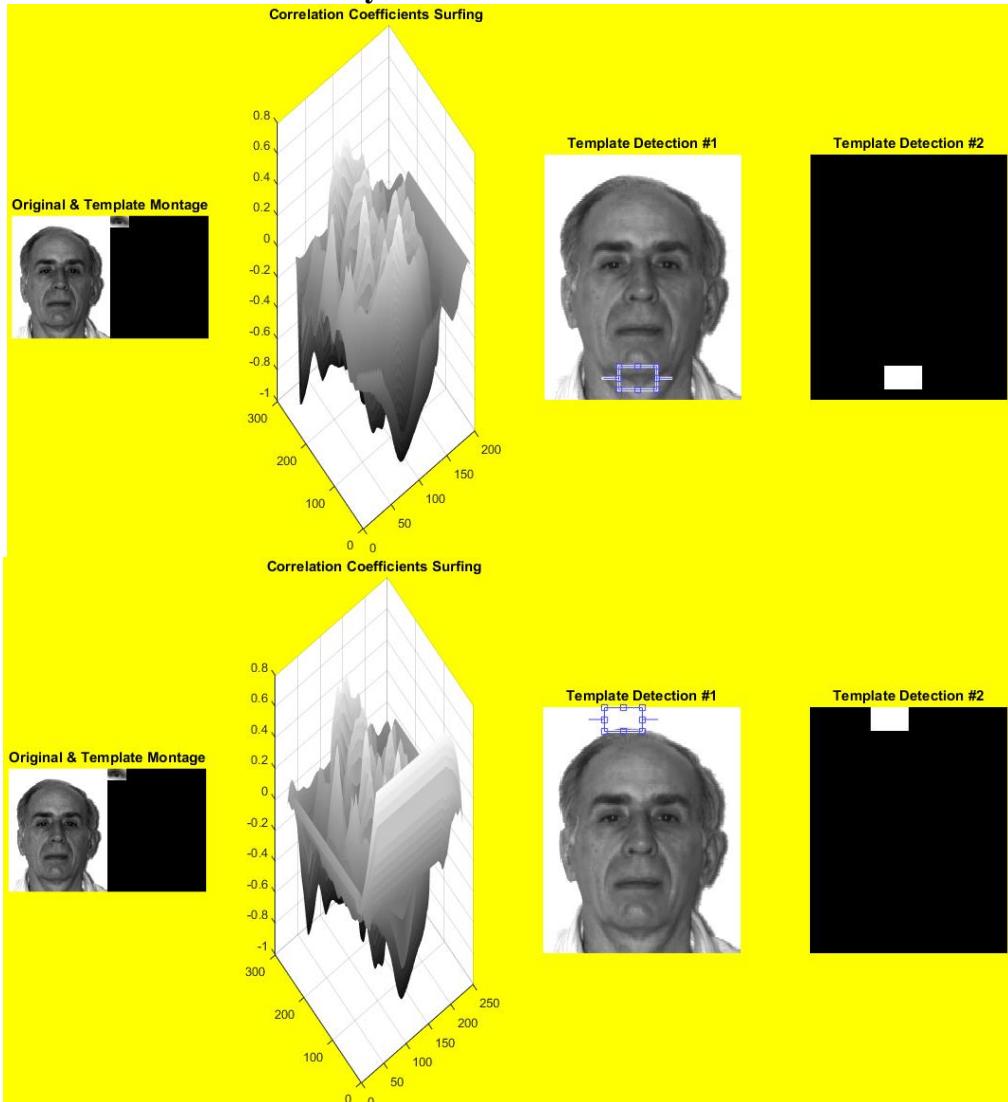


"mouth2" in "face1" results:

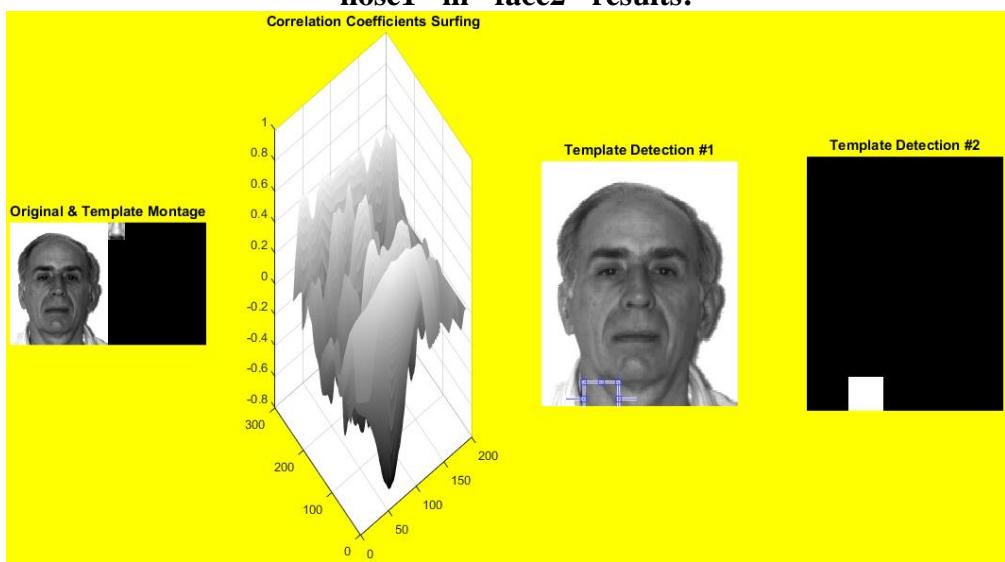


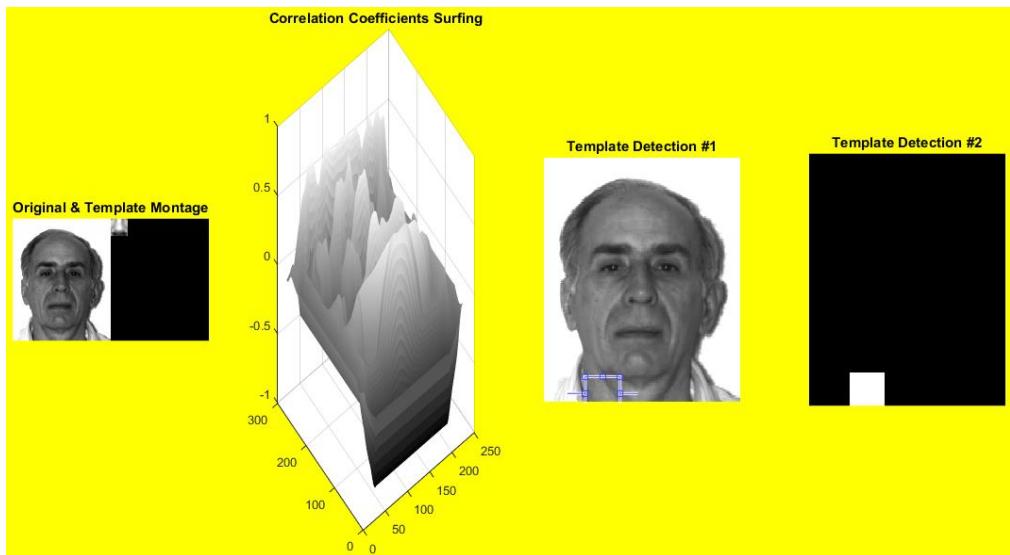
ii) کشف الگوهای `face2.pgm` ، `mouth1.pgm` ، `nose1.pgm` ، `eye1.pgm` در تصویر

"eye1" in "face2" results:

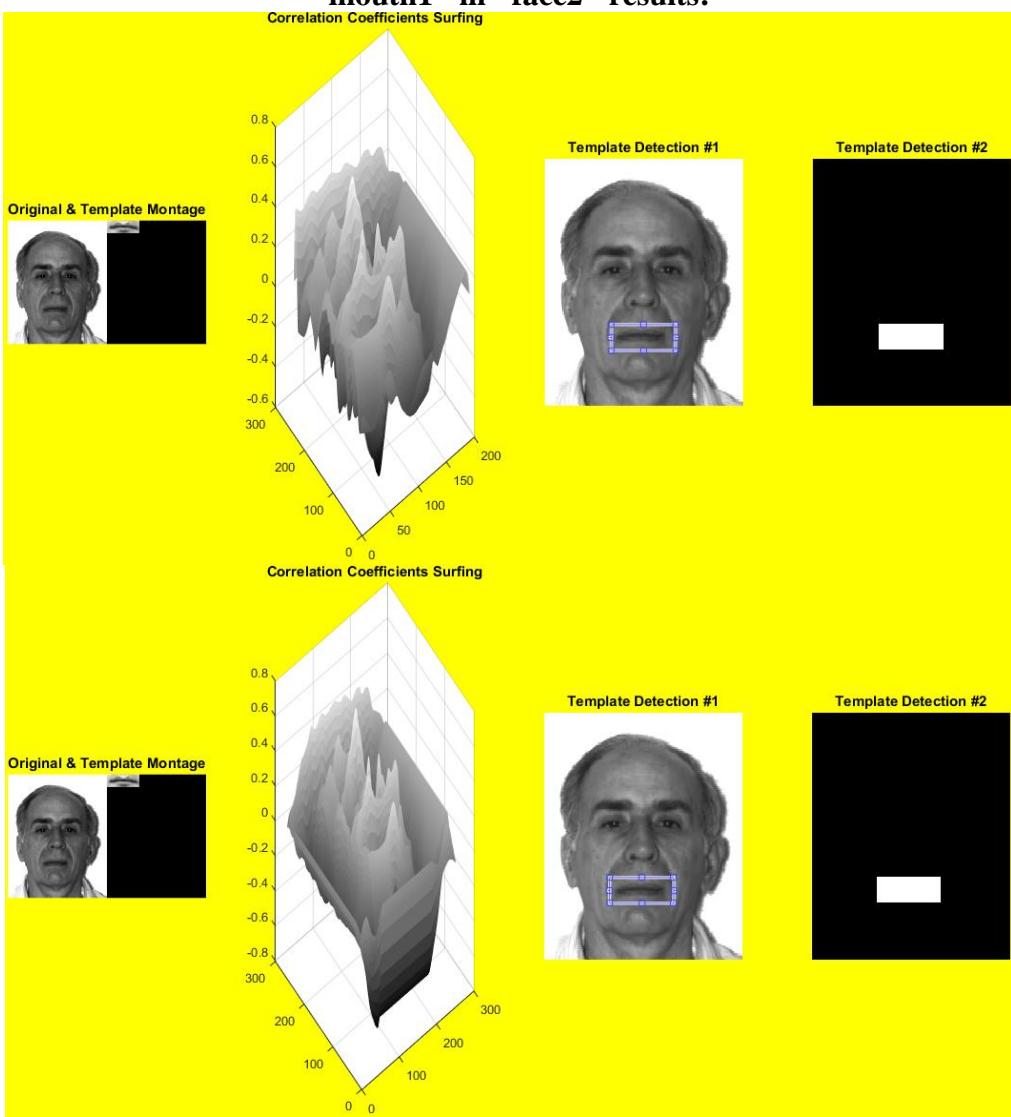


"nose1" in "face2" results:





"mouth1" in "face2" results:



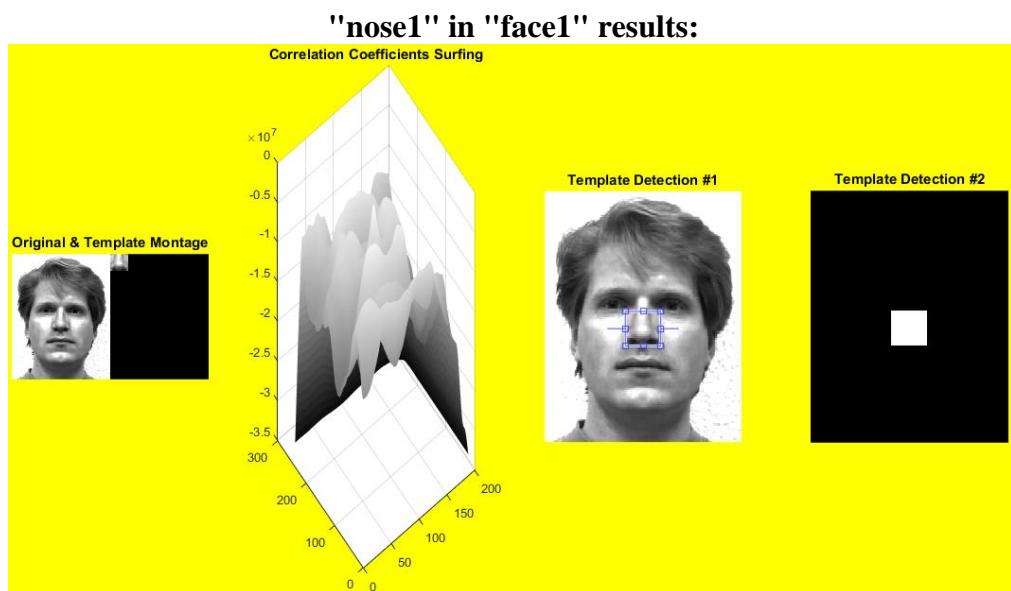
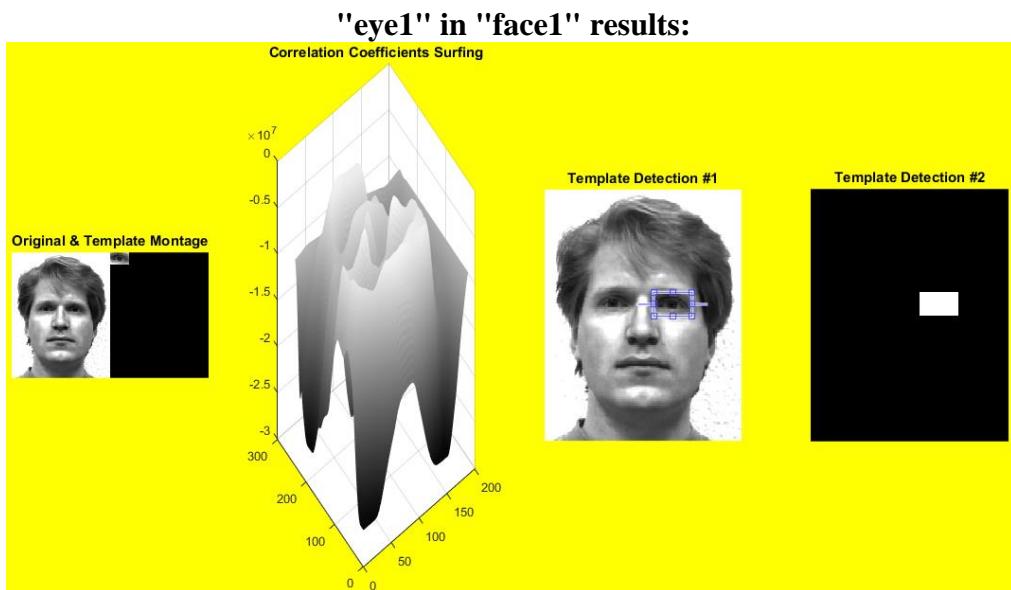
قسمت ج:

در قسمت الف، از آن جا که الگوهای مربوطه در تصاویر اصلی واقعا وجود داشتند، هر دو روش مذکور جوابشان کاملا درست و مطابق یکدیگر بود. اما در قسمت ب، از آن جا که الگوهای مربوطه در تصاویر وجود نداشتند، نتایج به جز حالتی که به دنبال تشخیص الگوی mouth در تصویر بودیم اصلا مطلوب نبوده و در حالت مذکور نیز تقریبا تشخیص صحیحی داشتیم که باز هم قابل اتکاء نمی باشد.

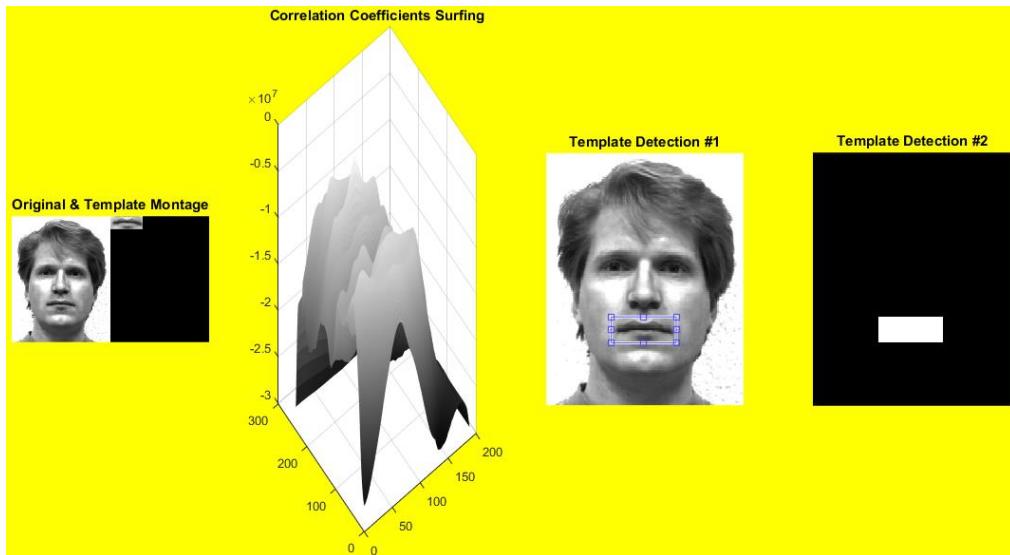
قسمت ۵:

در اینجا نیز برای استفاده از روش SSD(Sum of Squared Differences) مانند روش ابداعی قسمت‌های قبل عمل می‌کنیم؛ با این تفاوت که به جای ماتریس corrMat ماتریس ssdMat خواهیم داشت و اینکه در هر نقطه‌ی آن به جای خروجی (corr2(batch,template)) مقدار خروجی تابع ساختگی (ssdComp(batch,template)) را ذخیره می‌نماییم که درواقع این تابع مقدار SSD را محاسبه می‌نماید و این مقدار برابر مجموع مربعات تفاضلات مقدار پیکسل‌های نظیر به نظیر batch و template می‌باشد. در نهایت ماتریس ssdMat را فرینه نموده و نقطه‌ی بیشینه‌ی آن را می‌یابیم؛ این نقطه مانند قسمت‌های قبل مربوط به NCC، همان نقطه‌ی آغازی قسمت تطابق (detection) می‌باشد که آن را با یک مستطیل (مطابق توضیحات help mtlb راجع به تابع normxcorr2) و یا هم الگوی دودوئی نمایش می‌دهیم و همینطور با استفاده از تابع surf متلب، ماتریس corrMat را رسم می‌نماییم که در آن برابر نقطه‌ی ماقسیمم خروجی تابع surf با ابتدای ماتریس detection کاملاً مشهود است.

i) کشف الگوهای face1.pgm ، mouth1.pgm و nose1.pgm در تصویر eye1.pgm

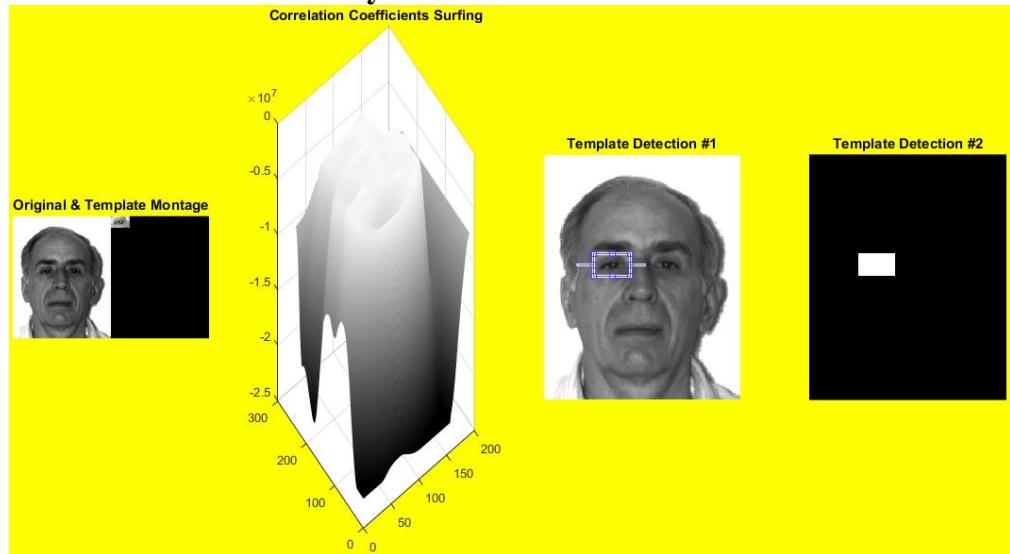


"mouth1" in "face1" results:

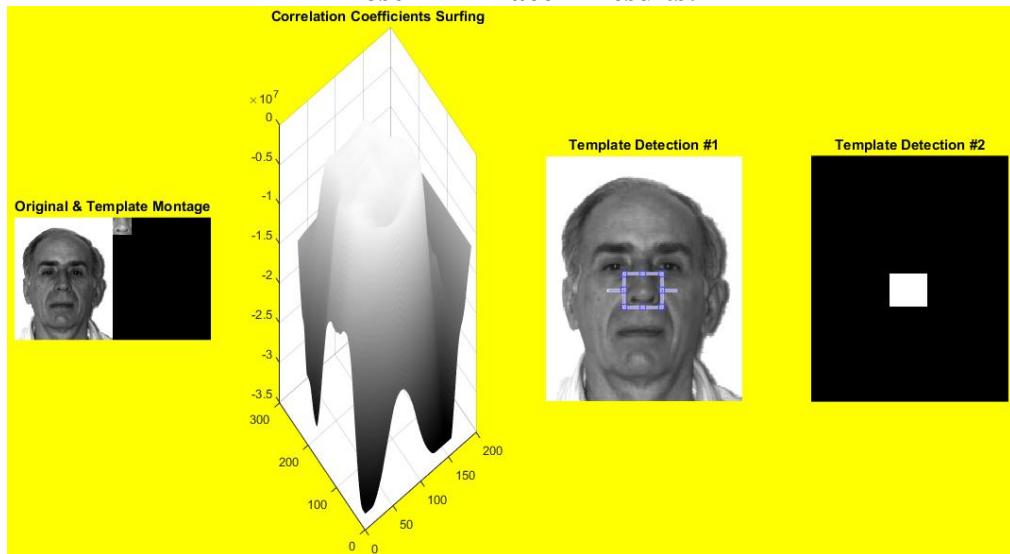


ii) کشف الگوهای mouth2.pgm و nose2.pgm ، eye2.pgm در تصویر face2.pgm

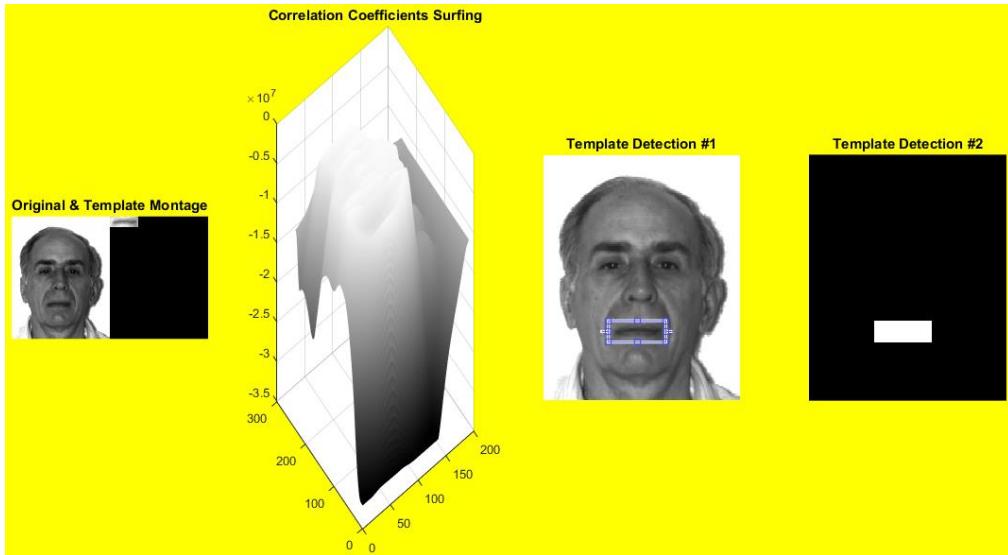
"eye2" in "face2" results:



"nose2" in "face2" results:



"mouth2" in "face2" results:



همانطور که قابل مشاهده است نتایج حاصله از روش SSD نیز مانند روش NCC، برای الگوهایی که در تصویر مورد نظر وجود داشتند کاملا نتایج صحیحی را حاصل نموده است.

قسمت ۵:

همانطور که پیش از این نیز قید شد برای قسمت‌های حاشیه‌ای مقدار صفر در نظر گرفتیم و در نهایت مقادیر Corr2 و SSD را محاسبه نمودیم.

قسمت ۶:

در مورد روش‌های NCC و SSD در این سؤال بخصوص تفاوت خاصی مشاهده نگردید و اتفاقاً نتایج حاصله از هر دو روش کاملا مطلوب بوده و صحت داشتند.

جواب سوال ۱

تفسیر موضوع ColorBalance

میزان‌سازی رنگ در حالت تعیین‌یافته

در تصویربرداری و پردازش تصویر، متوازن‌سازی رنگ به تنظیم سراسری شدت سطح روشنائی رنگ‌های موجود در تصویر (عموماً شامل رنگ‌های قرمز، سبز و آبی به عنوان رنگ‌های اولیه) اطلاق می‌شود و هدف عمدۀ از این عمل یعنی متوازن‌سازی رنگ، تجزیه‌ی صحیح رنگ‌های خاص از جمله رنگ‌های خنثی می‌باشد. لذا به این عمل گاهی متوازن‌سازی خاکستری، متوازن‌سازی خنثی و یا متوازن‌سازی سفید نیز گفته می‌شود. متوازن‌سازی رنگ در عمل ترکیب کلی رنگ‌های تصویر را به منظور بهبود آن متتحول نموده و البته نسخه‌های تعیین‌یافته‌ی متوازن‌سازی رنگ نیز به منظور اصلاح همه‌ی رنگ‌ها به جز موارد خنثی کاربرد دارند. عمدۀ تصاویری که توسط حسگرهای مختلف از نگاتیو گرفته تا دیجیتال ضبط می‌شوند، باید جهت نمایش مجدد و یا بازنگشtar رنگ‌ها از مقادیر فعلی ضبط شده به مقادیر جدید تغییر یابند. شرایط مختلف تصویربرداری مسئله‌ی متوازن‌سازی رنگ را ضروری می‌سازد از جمله اینکه حسگرهای تصویربرداری با حسگرهای موجود در چشم انسان سازگار نمی‌باشند؛ و البته اینکه شرایط مشاهده‌ی پیرامونی در حین تصویربرداری با شرایط مشاهده‌ی مربوط به نمایش مجدد تصویر تفاوت‌های عمدۀ دارند. عملیات توزان رنگ در اکثر برنامه‌های ویرایش تصویر عموماً به طور مستقیم بر روی مقادیر پیکسل‌های هر سه کanal قرمز، سبز و آبی بدون توجه به مدلی که با آن تصویر ضبط شده و یا تکثیر می‌شود، عمل می‌نمایند.

گاهی اوقات تنظیماتی که برای خنثی نگهداشت رنگ‌های خنثی اعمال می‌گردند، متوازن‌سازی سفید نمایده می‌شوند و عبارت متوازن‌سازی رنگ به سایر تنظیمات جهت شبیه‌سازی مطلق رنگ‌های موجود در یک تصویر ضبط شده با رنگ‌های موجود در صحنه‌ی واقعی مربوطه به کار می‌رود؛ و البته این مسئله ضروری می‌نماید که رنگ‌های خنثی شامل خاکستری و یا سفید، در انتشار مجدد نیز خنثی باقی بمانند و به همین دلیل موارد

خاص متوازن‌سازی رنگ از جمله متوزن‌سازی خاکستری و توزان سفید از جمله عناصر مهم و شاید غالب در زمینه‌ی متوازن‌سازی رنگ به حساب می‌آیند. نکته‌ی جالب توجه آن است که عبارت مصطلح بالانس کردن یا متوازن‌سازی رنگ، عمدتاً جهت ارجاع به تصحیح تفاوت‌هایی که در شرایط نورپردازی پیرامونی وجود دارد، به کار می‌رود. با توجه به تفاوت الگوریتم‌های موجود جهت موازن‌سازی رنگ؛ و اینکه این الگوریتم‌ها همیشه به طور آشکار جزئیات متفاوت تصحیح رنگ را تفسیر نمی‌کنند؛ لذا قادر نخواهیم بود تا عمل متوازن‌سازی رنگ را به یک مرحله‌ی خاص در فرآیند تصحیح رنگ اطلاق نمائیم. علاوه بر این تفاوت‌های عده‌ای در اهداف عمل متوازن‌سازی رنگ وجود دارد و به عبارتی هدف از برخی برنامه‌های متوازن‌سازی رنگ، تفسیر دقیق رنگ‌ها می‌باشد در حالی که برخی دیگر این کار را با هدف تولید یک تفسیر مطلوب انجام می‌دهند و البته این تفاوت‌ها سبب می‌گردد تا شناسائی و تمیزدادن عملیات‌های پردازش بالانس کردن رنگ دشوار گردد.

تخمین منبع نور و وفق‌دهی رنگ‌ها

اکثر دوربین‌های دیجیتال ابزار مخصوصی جهت تصحیح رنگ‌ها مبتنی بر نورپردازی محیط دارند، که این امر یا به صورت دستی و با استفاده از پروفایل‌های از پیش تعريف شده میسر می‌شود، و یا هم به صورت خودکار توسط الگوریتم‌های تعیین شده در سخت‌افزار دوربین و یا انجام همگی این تنظیمات توسط کاربر صورت می‌گیرد؛ و البته الگوریتم‌های مزبور با استفاده از روش‌های بالانس‌سازی رنگ تعیین‌یافته (معروف به تنظیم‌کننده‌ی منبع روشنایی و یا تنظیم‌کننده‌ی رنگ‌های روشن) که پیش از این نیز قید شد این امر را میسر می‌سازند. مقالات بسیاری در این زمینه موجود است که چگونه بتوان نورپردازی پیرامونی را با استفاده از داده‌های ضبط شده توسط دوربین دیجیتال تخمین زد و سپس با بهره‌گیری از این اطلاعات داده‌های مربوطه را به شکل مطلوب تبدیل نمود. از جمله‌ی این الگوریتم‌ها Retinex¹، یک شبکه‌ی عصبی مصنوعی و یا هم یک روش بیزین² می‌باشد.

متوازن‌سازی رنگ‌ها و رنگ‌های روشن

میزان‌سازی رنگ در یک تصویر نه تنها رنگ‌های خنثی، بلکه سایر رنگ‌ها را نیز تحت تأثیر قرار می‌دهد و تصویری که رنگ‌های آن بالانس نشده باشند اصطلاحاً داری ته رنگ³ خواهند بود، به این معنی که همه‌ی رنگ‌ها در تصویر مربوطه به یک رنگ نامرتب متمایل شده‌اند؛ و اما ته رنگ به مقدار اندکی از یک رنگ خاص اطلاق می‌شود که ناخواسته و به صورت هموار تمامی و یا تنها بخشی از یک تصویر ضبط شده توسط دوربین تصویربرداری را تحت تأثیر قرار می‌دهد، و البته انواع خاصی از نور حاضر در محیط تصویربرداری می‌توانند سبب شوند تا دوربین‌های دیجیتال و یا فیلم‌برداری مبتلا به پدیده‌ی ته رنگ گردند⁴. میزان‌سازی رنگ همچنین مرتبط با عمل ثبات رنگ⁵ نیز می‌باشد. مسئله‌ی ثبات رنگ در عمل یک نمونه از ثبات و یکپارچگی رنگ‌ها که به صورت ذهنی و ادراکی توسط کاربر قابل درک است، می‌باشد و نیز به خاصیتی از سیستم ادراک رنگ توسط انسان اطلاق می‌شود که موجب این اطمینان می‌گردد که رنگ‌های دریافت شده از اشیاء توسط چشم انسان تحت شرایط متنوع نورپردازی نسبتاً ثابت و بلا تغییر باقی می‌مانند⁶. الگوریتم‌های متنوعی که جهت اعمال ثبات رنگ به کار می‌روند جهت متوازن‌سازی رنگ نیز کاربرد داشته و در نتیجه مسئله‌ی ثبات رنگ به تنظیم و تعدیل رنگ‌های روشن نیز مربوط می‌شود. متوازن‌سازی رنگ به صورت ادراکی مشکل از دو مرحله می‌باشد: اول اینکه منبع نوری که در زیر آن تصویر مربوطه ضبط شده است را شناسائی نمائیم؛ و دوم این که جزئیات تصویر مشکل از سه کanal رنگ مربوطه (RGB) را مقیاس‌بندی⁷ نمائیم و یا در عوض جزئیات را به گونه‌ای متحول نماییم که با منبع نور حاضر در محل همخوانی داشته باشند. ویگیانو⁸ متوجه شد که متوازن‌سازی سفید در نمایشگر RGB خود دوربین دیجیتال نسبت به سایر نمایشگرهای RGB، پس از اخذ تصویر سبب ایجاد عدم ثبات رنگ کمتری چون اعوجاج کمتر رنگ‌ها می‌شود و این مسئله متنج از وجود بیش از ۴۰۰۰ مجموعه‌ی تخمین گر در حسگرهای دوربین می‌باشد. به همین دلیل بهتر است تا عمل متوازن‌سازی رنگ دقیقاً در زمانی که یک تصویر اخذ می‌شود صورت گیرد، تا این که پس از ضبط تصویر در حافظه‌ی دوربین، در یک نمایشگر دیگر این اقدام صورت گیرد.

رباضیات نهفته در متوازن‌سازی رنگ

متوازن‌سازی رنگ گاهی بر روی یک تصویر سه‌جزئی مانند RGB، با استفاده از یک ماتریس 3×3 صورت می‌گیرد، که این شکل دگردیسی تصویر تنها در صورتی کاربرد دارد که تصویر مربوطه توسط دوربین دیجیتال و یا یک فیلتر رنگ⁹ و با استفاده از تنظیمات اشتباه متوازن‌سازی سفید ضبط شده باشد. در اصل هدف ما مقیاس‌بندی همه‌ی روشنایی‌های نسبی در یک تصویر است به گونه‌ای که اشیاء که به نظر می‌رود خنثی هستند، هویا گردند. برای این کار اگر در تصویر ضبط شده، سطحی با روشنایی کانال قرمز R=240 به عنوان یک شیء سفید محسوب گردد، و

¹ Retinex

² Bayesian method

³ Color cast

⁴ https://en.wikipedia.org/wiki/Colour_cast

⁵ Color constancy

⁶ https://en.wikipedia.org/wiki/Color_constancy

⁷ Scaling

⁸ Viggiano

⁹ Color filter

اگر حداکثر مقدار روشنائی در نوع تصویر ضبط شده برابر ۲۵۵ باشد، می‌توانیم تمامی مقادیر قرمز موجود در تصویر را در مقدار $255/240$ ضرب نمائیم؛ و انجام مشابه همین عمل برای مقادیر کanal‌های سبز و آبی، حداقل به صورت تئوری وار منجر به تولید یک تصویر با رنگ‌های بالانس شده می‌گردد. در این نوع تبدیل تصویر ماتریس 3×3 مذکور یک ماتریس قطری به صورت زیر خواهد بود:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255/R_w & 0 & 0 \\ 0 & 255/G_w & 0 \\ 0 & 0 & 255/B_w \end{bmatrix} \begin{bmatrix} \tilde{R} \\ \tilde{G} \\ \tilde{B} \end{bmatrix}$$

به طوری که R و B مقادیر متوازن شده‌ی جزئیات قرمز، سبز و آبی یک پیکسل در تصویر ورودی می‌باشند؛ و \tilde{R} ، \tilde{G} و \tilde{B} مقادیر جزئیات قرمز، سبز و آبی تصویر پیش از متوازن‌سازی رنگ بوده؛ و R_w ، G_w و B_w نیز جزئیات قرمز، سبز و آبی یک پیکسل خاص می‌باشند که معتقدیم متعلق به یک سطح سفید در تصویر ورودی پیش از عملیات متوازن‌سازی رنگ می‌باشد. این روش یک شیوه‌ی ساده‌ی مقیاس‌بندی کanal‌های قرمز، سبز و آبی در تصویر می‌باشد و به همین دلیل است که ابزارهای متوازن‌سازی رنگ در نرم‌افزارهای ویرایش تصویر نظیر Photoshop و GIMP یک ابزار قطره‌چکان^{۱۰} جهت اخذ رنگ از سطح سفید مطلوب کاربر می‌باشند.

در نهایت به ذکر فضاهای بسته می‌کنیم که از لحاظ آماری به بهترین نحو بر روی عمدیه‌های مجموعه‌های تصویر آزمایشی عمل نموده‌اند؛ از جمله‌ی این فضاهای فضاهای 'CMCCAT'، 'Bradford'، 'Sharp' و 'ROMM' می‌باشند.

جواب سوال ۹

آشنائی با کار با فیلترهای مختلف در MATLAB

تصویر ورودی snake.jpg می‌باشد که در زیر قابل مشاهده است:



ردیف

تفسیر عملکرد فیلتر

نتیجه اعمال فیلتر روی تصویر ورودی

A

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

این فیلتر یک فیلتر میانگین‌گیری است که در نهایت باعث ایجاد یک تصویر کدر یا مات می‌شود.



¹⁰ Eyedropper tool

1/18	1/18	1/18
1/18	1/18	1/18
1/18	1/18	1/18

B این فیلتر مطابق با فیلتر A می‌باشد با این تفاوت که پس از

میانگین‌گیری مقدار میانگین نصف می‌شود و این مسئله سبب می‌گردد تا حداقل روشنایی برای هر پیکسل برابر ۱۲۷ باشد.



1/7	1/7	1/7
1/7	1/7	1/7
1/7	1/7	1/7

C این فیلتر نیز مانند فیلتر میانگین عمل می‌کند با این تفاوت که

مخرج کسر نهائی کمتر از تعداد درایه‌های فیلتر یعنی ۹ می‌باشد و این مسئله سبب می‌شود تا مقدار نسبت داده به هر پیکسل روشن‌تر باشد.



-1	-1	-1
-1	9	-1
-1	-1	-1

D این یک فیلتر Standard Laplacian Sharpening از نوع توسعه‌یافته‌ی آن و با در نظر گرفتن همسایگان 8-connected یک پیکسل می‌باشد که باعث می‌شود جزئیات ریز در تصویر خروجی بیشتر نمایان گردد. البته در مورد این تصویر به دلیل وجود فرکانس زیاد سطوح روشنایی که اصطلاحاً نویز نامیده می‌شوند، نتیجه‌های نهائی چندان مطلوب نمی‌باشد و برای رفع آن می‌توان پیش از اعمال این فیلتر از یک فیلتر میانگین گاووسی با انحراف معیار پایین (low sd) استفاده کرد.



-1	-1	-1
-1	18	-1
-1	-1	-1

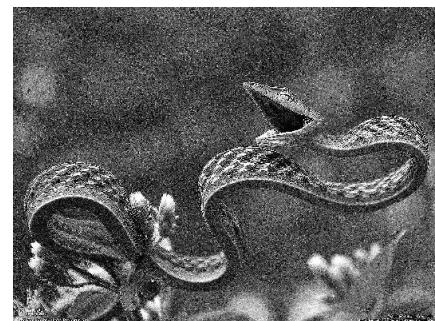
E این فیلتر یک فیلتر high-boost با مقدار $A=10$ است. در اینجا

چون وزن مرکز فیلتر بسیار بالاست در نتیجه‌ی آن مقدار هر پیکسل بسیار زیاد گشته و تصویر نهائی روشن‌تر خواهد بود.



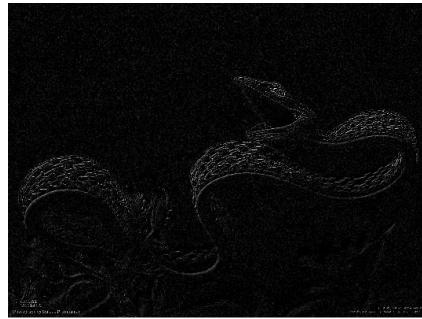
0	-1	0
-1	5	-1
0	-1	0

F این فیلتر نیز یک Standard Laplacian Sharpening Filter می‌باشد اما از نوعی که تنها همسایگان 4-connected یک پیکسل را در نظر می‌گیرد، و مثل حالت D سبب بر جسته‌تر شدن جزئیات ریز در یک تصویر و به عبارتی تیزتر شدن آن می‌شود؛ و در مورد این تصویر خاص با نویز بالا بهتر عمل کرده و به عبارتی در برابر نویز پایدارتر



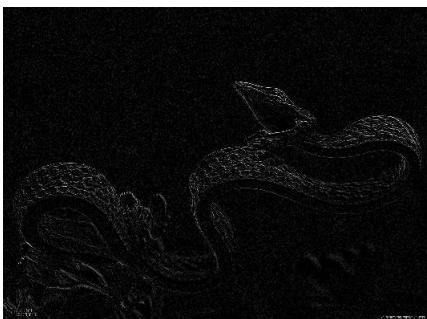
است.

1	0	0
0	0	0
0	0	-1



G این فیلتر یک فیلتر Roberts cross-gradient می‌باشد که بیشتر نسبت به لبه‌های ۴۵ درجه نسبت به pixel grid حساسیت نشان می‌دهد. همانطور که در تصویر نتیجه هم قابل مشاهده است این تغییرات مشهودتر شده‌اند.

0	0	-1
0	0	0
1	0	0



H این فیلتر نیز مانند مورد G یک فیلتر Roberts cross-gradient می‌باشد با این تفاوت که نسبت به لبه‌های ۱۳۵ درجه نسبت به pixel grid واکنش نشان می‌دهد و همانطور که در تصویر نتیجه هم قابل مشاهده است این گونه تغییرات برجسته‌تر شده‌اند.

0	1	0
0	0	0
0	-1	0



I این فیلتر نیز مانند موارد G و H از نوع فیلترهای Roberts cross-gradient می‌باشد با این تفاوت که نسبت به تغییرات افقی حساس‌تر می‌باشد و این مسئله در تصویر خروجی قابل مشاهده است.

0	0	0
-1	0	1
0	0	0



J این فیلتر نیز مانند مورد I می‌باشد تنها با این تفاوت که نسبت به تغییرات عمودی حساس‌تر بوده و مطابق تصویر حاصله آن‌ها را برجسته‌تر می‌نماید.

1	1	1
1	-8	1
1	1	1



K این نیز یک فیلتر لاپلاس است که از آنجائی که یک عملگر مشتق می‌باشد ناپیوستگی‌های سطح خاکستری را در یک تصویر برجسته‌تر نموده و نیز نواحی با تغییرات آرام سطح خاکستری را کمرنگ‌تر می‌نماید. اما همانطور که از تصویر خروجی نیز مشخص است از آن جا که تصویر ورودی دارای تغییرات زیاد سطح روشناهی و به عبارتی نویز است، این

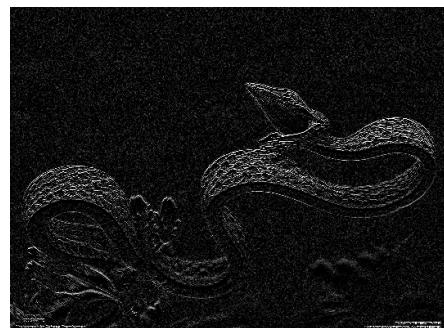
فیلتر سبب تقویت نویزها شده و بهتر است پیش از اعمال این فیلتر، یک

فیلتر میانگین به تصویر اعمال کنیم.

-1	-1	-1
0	0	0
1	1	1

L

این فیلتر نیز مانند فیلتر I سبب برجسته‌تر شدن تغییرات افقی می‌شود با این تفاوت که در برابر نویز ناپایدارتر بوده و مطابق تصویر آن‌ها را نیز آشکار نموده است.



-1	-1	-1
0	1	0
1	1	1

M این فیلتر نیز در واقع از جمع تصویر حاصل از اعمال فیلتر L و تصویر اصلی حاصل شده است و همانطور که قابل مشاهده است صرف نظر از برجسته‌تر شدن تغییرات افقی سایر جزئیات تصویر ورودی نیز نشان داده‌اند.



-1	0	0
0	2	0
0	0	-1

N این فیلتر نیز مانند فیلتر G جهت برجسته‌کردن تغییرات ۴۵ درجه نسبت به کار می‌رود با این تفاوت که به پیکسل اصلی وزن بیشتری اختصاص می‌دهد.



0	0	-1
0	2	0
-1	0	0

O این فیلتر نیز مانند فیلتر H جهت برجسته‌کردن تغییرات ۱۳۵ درجه نسبت به کار می‌رود با این تفاوت که به پیکسل اصلی وزن بیشتری اختصاص می‌دهد.



	1/3	0
0	1/3	0
0	1/3	0

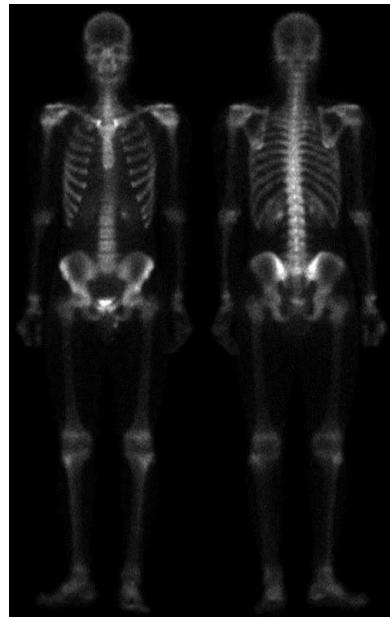
P این فیلتر نیز یک فیلتر میانگین‌گیری تنها در جهت عمودی می‌باشد که همانطور که از تصویر نهائی پیداست، تصویر کمی کدر یا مات شده است.



جواب سوال ۱۰

آشنائی با ترکیب روش‌های بهبود مکانی تصویر (Enhancement Methods)

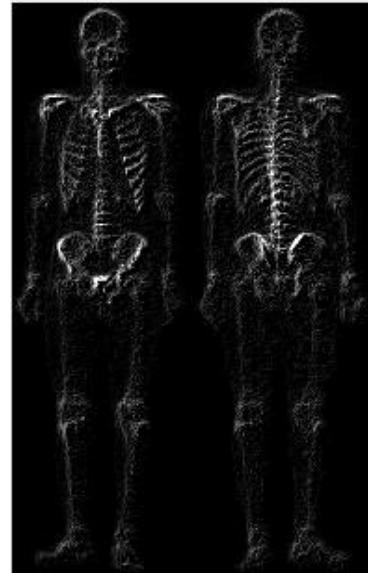
تصویر ورودی skeleton.jpg به قرار زیر می‌باشد:



ترتیب	توضیح	نتیجه
۱	در این مرحله یک فیلتر لاپلاسین به تصویر اعمال می‌کنیم که سبب پرنگ شدن جزئیات موجود در تصویر می‌شود، اما مشکل آن برجسته کردن نویزهای موجود در تصویر می‌باشد که از خروجی سمت راست قابل برداشت است.	

۲

در این مرحله گرادیان تصویر ورودی را با استفاده از عملگرهای Sobel به دست می آوریم که نتیجه‌ی آن همانطور که قابل مشاهده است نسبت به انتقالات برجسته‌ی سطح خاکستری (gray-level ramps and steps) پاسخ قوی‌تری داشته و استثنائاً برخلاف عملگر لاپلاسین نسبت به جزئیات ریز و نویزها پاسخ ضعیف‌تری دارد.



۳

در این مرحله برای کاهش میزان نویز در تصویر حاصل از اعمال عملگر گرادیان، از یک فیلتر میانگین‌گیری 5×5 استفاده می‌کنیم.



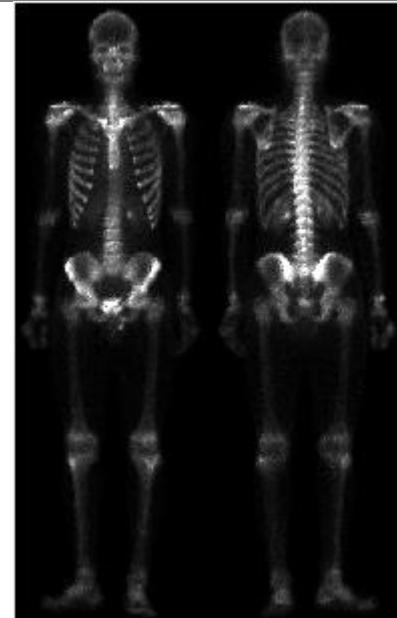
۴

در این مرحله نیز برای جهت حفظ جزئیات ریز در نواحی مطلوب و البته کاهش نویز، از عملیات Gray-level masking استفاده می‌کنیم که از ضرب آرایه‌ای خروجی فیلتر لاپلاسین و گرادیان هموارشده‌ی تصویر به دست می‌آید. به عبارتی گرادیان هموارشده همچون یک ماسک برای خروجی لاپلاسین عمل کرده و در عین اینکه نویزها را حذف می‌نماید، لبه‌های تأثیرگذار را حفظ می‌کند.



۵

در این مرحله نیز تصویر اصلی را به خروجی مرحله‌ی قبلی اضافه می‌کنیم که نتیجه‌ی آن یک تصویر شارپ‌شده در نواحی مطلوب یعنی استخوان‌های فرد می‌باشد.



۶

تصویر مرحله‌ی آخر از لحاظ این‌که جزئیات اصلی را خوب نمایش می‌دهد ارزشمند است، اما دامنه دینامیک هیستوگرام آن کم می‌باشد که برای بهبود این مسئله روش‌های متعددی چون برابرسازی هیستوگرام، تطبیق هیستوگرام و البته قانون توانی (power-law) وجود دارد که ما آخری را انتخاب می‌کنیم و علت آن ضعف دو مورد قبلی (خصوصاً اولی یعنی برابرسازی هیستوگرام) در مورد تصاویر با پیکسل‌های عمدتاً تیره می‌باشد. در اینجا با توجه به فرمول power-law یعنی $V_{out} = A V_{in}^{\gamma}$ ، ما برای A مقدار ۱ و برای γ نیز مقدار $1/2$ را برمی‌گزینیم. همانطور که در تصویر نهائی مشاهده می‌شود علاوه بر جزئیات استخوانی، قامت فرد مورد نظر نیز پس از استفاده از قانون توانی بهتر مشخص شده است.

