گزارش سوال اول:

روش اول: ابتدا تصویر را بارگذاری کرده و در متغیری میریزیم. سپس از آنجا که مانیتور صفحه لپ تاپ بنده 141dpi نمایش میدهد، لذا نسبت های 1200/141 و 600/141 و 300/141 و 150/141 و 75/141 را محاسبه کرده و توسط دستور imresize تصویر اصلی را به این نسبت ها scale میکنیم و سپس آنها را در figure ای کنار هم با title مشخص نمایش میدهیم.

روش دوم: ابتدا تصویر را بارگذاری کرده و در متغیری میریزیم. سپس آن را در figure ای نمایش میدهیم(با عنوان 1200dpi) و آنرا به وسیله تابع print که آرگومانی برای dpi ذخیره سازی تصویر دارد، در فایلی با فرمت png به نام img1200dpi ذخیره میکنیم و سپس آنرا بارگذاری کرده و نمایش میدهیم. برای دیگر dpi ها هم همین کار را تکرار کرده و هر دفعه آرگومان تابع print را متناظر با نیاز سوال، تغییر میدهیم و مابقی کارها، مثل حالت اول است و تفاوتی ندارند.

گزارش سوال دوم:

ابتدا تصویر مورد نظر را بارگذاری کرده و با تابع rgb2gray به تصویری به نوع intensity تبدیل میکنیم.سپس ماتریسی به اندازه 512*512 با درایه های صفر به نام new میسازیم. برای راحتی محاسبات تصویر را به اندازه هایی نزدیک به اندازه واقعی resize میکنیم که اندیس های آن مضرب 512 باشند. سپس ماتریس جدید را به صورت مستطیل هایی به اندازه 10*6 پیمایش کرده و میانگین عددی همه درایه های آن م ;lستطیل را به عنوان نماینده در درایه متناظر آن در ماتریس new جایگذاری میکنیم. با این کار تصویری 512*512 مشابه با تصویر اصلی داریم و با اینکار توانستیم عمل resize را با متد pixel replication پیاده سازی کنیم.

برای پیاده سازی تابع quantize می آییم مکان عددی هر پیکسل تصویر اصلی را در بازه 0 تا 255 میابیم و آنرا به # level of quantization که سوال از ما خواسته، نگاشت میکنیم. یعنی برای مثال عدد 128 در بازه 0 تا 255 میشود عدد 64 در بازه 0 تا 128 . فرمول کلی اینکار میشه:

New#(i,j) = round(double(new(i,j))/256 * #)

که به جای علامت # آن سطح از quantization ای که مورد نیاز است را جایگذاری میکنیم.

سپس موقع نمایش آن تصویر باید به سیستم بفهمانیم قصد داریم تصویر را با فرض اینکه درایه های ماتریس اعدادی بین 0 تا # هستند نمایش دهیم. برای اینکار از دستور زیر استفاده میکنیم:

Imshow(new# , [0 #]);

که اینجا هم به جای # از همان level of quantization ای که نیاز داریم، استفاده میکنیم.

گزارش سوال سوم:

ابتدا تصویر را بارگذاری کرده و طول و عرض آنرا برای استفاده های آتی در متغیر هایی ذخیره میکنیم. سپس subplot ای میسازیم و تصویر اصلی را و تصویری که توسط تابع imnoise به آن نویز اضافه شده را به همراه هیستوگرام هایشان نمایش میدهیم. توجه شود با توجه به آرگومان های تابع imnoise ما نویز گاوسین ای با میانگین 0 و واریانس یک صدم به تصویر اضافه کردیم.

سپس ماتریسی به سایز تصویر اصلی برای میانگین گیری های آینده با درایه های صفر میسازیم. حال دفعه اول 5 مرتبه نویز هایی مثل نویز گفته شده به تصویر اضافه کرده و از آنجایی که تابع imnoise هر دفعه به صورت رندوم نویز اضافه میکند، ما مجموع درایه های تصویر های نویزی را جمع زده و سپس بر تعداد آن ها تقسیم میکنیم(میانگین گیری).

با اینکار قصد داریم نویز گاوسین را کاهش دهیم.حال مثل قبل تصویر نویزی اصلی را به همراه تصویر میانگین گیری شده و هیستوگرام آنها نمایش میدهیم.

در دفعه های بعد به جای 5 بار، 10 بار، 50 بار و 100 بار تصاویر نویزی میسازیم و میانگین میگیریم.

مشاهده میشود که هرچه تعداد این تصاویر در میانگین گیری بیشتر میشود، نویز تصویر نهایی کمتر شده و به تصویر واقعی نزدیک تر میشود(هر با چشم مشخص است و هم از روی هیستوگرام مشاهده میشود).

گزارش سوال چهارم:

ابتدا تصویر مورد نظر را بارگذاری کرده و با تابع rgb2gray به تصویری به نوع intensity تبدیل میکنیم. سپس بوسیله تابع input از کاربر پارامتر shearing را گرفته و بوسیله آن ماتریس تبدیل را میسازیم. سپس بوسیله ماتریس تبدیل، با تابع maketform تابع تبدیل میسازیم.

سپس بوسیله دستور cpselect ابزار control point tool را فراخوانی کرده و با دقت زیاد 4 نقطه کنترلی در تصویر اصلی و تصویر shear شده انتخاب کرده و درون workspace متلب export میکنیم.چون درایه های export شده از نوع double و به فرمتی ناخوانا بودند، ابتدا آنهارا integer کرده و سپس به double تبدیل میکنیم.

حال بوسیله symbolic math toolbox متغیر های c1 ~ c8 را تعریف کرده و 8 معادله مربوطه، تعریف میکنیم. سپس با تابع solve این 8 معادله را حل کرده و پاسخ را در متغیری میریزیم. سپس برای راحتی کار، درایه های آنرا double کرده و به صورت یک بردار ذخیره میکنیم.

حال ماتریسی به اندازه تصویر original میسازیم و طول و عرض تصویر اصلی و تصویر shear شده را هم برای استفاده های آینده، در متغیر هایی ذخیره میکنیم.

حال روی تصویر shear شده به صورت ستونی پیمایش کرده و مختصات هر نقطه را در تصویر اصلی میابیم(به اصطلاح register میکنیم). حال تصویر اصلی را بازسازی کرده و به همراه تصویر shear شده و تفاوت این دو(بوسیله تابع imabsdiff) نمایش میدهیم.

نکته ای که وجود دارد این است که تصویر shear شده در اصل تصویر است و ماتریس ساخته شده نیست، لذا ستون و عرض های آن را باید هنگام register کردن جا به جا کنیم چون تصویر Register به شکل ماتریسی ذخیره میشود و جای ستون و سطر هایش برعکس تصویری هست که به شکل تصویر ساخته شده است.امیدوارم این قسمت آخر را توانسته باشم توضیح دهم و متوجه شوید.

با تشکر از زحمات شما

پویا خانی 99210283