دانشکده مهندسی کامپیوتر

دانشگاه اصفهان

# مستندات پروژه جبرخطی

میترا عمرانی ۹۹۳۶۱۳۰۴۷

خرداد 1402

در این پروژه عکس‌هایی را متناسب با دیتاست انتخابی google script images به برنامه می‌دهیم.
در ادامه به آن guassian نویز اضافه می‌کنیم و سپس عملیات denoising را بر روی ان انجام می‌دهیم.
مستندات این پروژه شامل مثال‌ها و توابع پیاده‌سازی و دلیل انجام این کار است.

**ابتدا به بررسی دلیل این نوع عملکرد می‌پردازیم:**
اعمال نویز گاوسی روی یک تصویر و سپس حذف نویز می تواند کیفیت کلی تصویر را با کاهش تاثیر نویز بهبود بخشد. در اینجا به این دلیل است که این فرآیند معمولاً مورد استفاده قرار می گیرد:

۱. شبیه سازی شرایط دنیای واقعی:

نویز گاوسی نوعی نویز تصادفی است که به طور طبیعی در بسیاری از سیستم های تصویربرداری رخ می دهد. با اضافه کردن نویز گاوسی به یک تصویر، می‌توانیم نویزهایی را که ممکن است در سناریوهای دنیای واقعی وجود داشته باشد، مانند نویز حسگر دوربین یا نویز انتقال، شبیه‌سازی کنیم.

۲. بهبود جزئیات تصویر:

نویز اضافه شده به تصویر می تواند تغییرات با فرکانس بالا را ایجاد کند که ممکن است جزئیات یا بافت های خاصی را در تصویر بهبود بخشد. این می تواند تصویر را واضح تر یا از نظر بصری جذاب تر نشان دهد.

۳. حذف نویز برای کیفیت بهتر:

پس از افزودن نویز گاوسی، مرحله بعدی حذف نویز تصویر است. الگوریتم های حذف نویز برای کاهش تاثیر نویز و در عین حال حفظ جزئیات مهم تصویر طراحی شده اند. این الگوریتم ها تصویر را تجزیه و تحلیل می کنند و اجزای نویز را حذف می کنند و در نتیجه تصویری تمیزتر و از نظر بصری دلپذیرتر ایجاد می کنند.

که در این پروژه به ویژگی سوم توجه داشته‌ایم.

قسمت اصلی برنامه (mian) :
در این قسمت از توابع opencv برای باز کردن تصاویر و resize کردن آن‌ها استفاده شده است.
در قسمت بعد تابع add_gaussian_noise(img, 0,50) بر روی تصویر صدا زده می‌شود تا به آن نویز از نوع گوسی اضافه کند.
سپس تابع denoise(noisy_image,5) را بر روی تصویری که در مرحله‌ی قبل به آن نواز اضافه کردیم صدا می‌زنیم.

توابع به پنج دسته تقسیم می‌شوند.

## ۱. توابعی که نویز تولید می‌کنند :

### ۱. box_muller_transform(mu, sigma):

این تابع تبدیل Box-Muller را پیاده‌سازی می‌کند، که روشی برای تولید جفت‌های استاندارد مستقل اعداد تصادفی معمولی توزیع شده است. دو پارامتر mu (میانگین) و سیگما (انحراف استاندارد) نیاز دارد. در داخل تابع، دو عدد تصادفی u1 و u2 را با استفاده از تابع random.random() تولید می کند. سپس، معادلات تبدیل Box-Muller را برای محاسبه z0 و z1، که اعداد تصادفی معمولی توزیع شده استاندارد هستند، اعمال می‌کند. در نهایت، z0 و z1 را به ترتیب با سیگما و mu تغییر داده و تغییر می دهد و مقادیر تبدیل شده را برمی گرداند.

$$Z_0 = R\cos(\Theta) = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

$$Z_1 = R\sin(\Theta) = \sqrt{-2\ln U_1}\sin(2\pi U_2).$$

۲. add_gaussian_noise(image, mu=0, sigma=1):

این تابع نویز گاوسی را به تصویر اضافه می کند. یک تصویر را به عنوان ورودی و دو پارامتر اختیاری، mu (میانگین) و سیگما (انحراف استاندارد) می گیرد که به ترتیب 0 و 1 پیش فرض هستند. این تابع ابتدا تصویر را به کانال های قرمز، سبز و آبی (r، g، b) جدا می کند. سپس، روی هر پیکسل در تصویر تکرار می شود و با استفاده از تابع box_muller_transform، نویز گاوسی را به کانال مربوطه اضافه می کند. پس از اضافه کردن نویز، تابع مقادیر پیکسل را در محدوده معتبر 0 تا 255 برمی‌گرداند. در نهایت، تصویر اصلاح‌شده را با نویز گاوسی اضافه شده برمی‌گرداند.

## ۲. توابعی که که عملیات دینویز کردن را انجام می‌دهد.

۱. denoise(image,threshold) :

کد ارائه شده حذف نویز را بر روی یک تصویر با استفاده از یک آستانه مشخص انجام می دهد. تصویر را به کانال های رنگی خود (آبی، سبز و قرمز) تقسیم می‌کند و سپس فرآیند حذف نویز را برای هر کانال به طور جداگانه اعمال می‌کند. فرآیند حذف نویز شامل اعمال یک آستانه برای هر پیکسل در کانال است. اگر مقدار پیکسل زیر آستانه باشد، روی 0 (سیاه) و در غیر این صورت روی 255 (سفید) تنظیم می‌شود.

این به حذف نویز از تصویر و افزایش وضوح آن کمک می کند. در نهایت، کانال‌ها دوباره با هم ادغام می شوند تا تصویر دینویز شده را تشکیل دهند. مقدار آستانه را نیز می‌توان به عبارتی همان تعداد بردارهای منفرد دانست، در توضیحات توابع بعدی به این موضوع پرداخته خواهد شد.

۲. denoise_channel(channel, threshold):

U, s, Vt =calculate_SVD1(channel) :

این خط تابعی را برای محاسبه تجزیه ارزش واحد (SVD) کانال ورودی فراخوانی می‌کند. SVD یک تکنیک فاکتورسازی ماتریسی است که یک ماتریس را به سه ماتریس جداگانه تجزیه می‌کند: U، s ، Vt ماتریس‌های متعامد هستند و s یک ماتریس مورب حاوی مقادیر منفرد است.

s[s < آستانه] = 0:
این خط یک آستانه را برای مقادیر (s) منفرد به دست آمده از SVD اعمال می کند. هر مقدار منفرد زیر آستانه مشخص شده روی 0 تنظیم می شود. این مرحله آستانه به حذف نویز از کانال کمک می‌کند.
خواهیم دید با افزایش این مقدار که به منزله‌ی افزایش تعداد بردارهای منفرد است دینویزینگ بهتر انجام می‌شود، هرچند از حدی به بعدی تصویر ممکن است وضوح خود را از دست بدهد پس باید مقدار و تعداد مناسب این را بررسی کرد.
این مورد در قسمت نتیجه در عکس‌ها بیان شده است.

denoised_channel = U.dot(np.diag(s)).dot(Vt):
این خط با ضرب ماتریس های U، s (تبدیل به یک ماتریس مورب با استفاده از np.diag) و Vt، کانال حذف شده را بازسازی می کند. این عملیات ضرب به طور موثر مقادیر منفرد حذف شده را با ماتریس‌های متعامد اصلی ترکیب می‌کند.

۳. calculate_SVD1(A):
مرحله ۱: ماتریس کوواریانس C را با گرفتن حاصل ضرب نقطه ای A با جابجایی آن A.T محاسبه می‌کنیم.

مرحله ۲: تجزیه مقدار ویژه را بر روی ماتریس کوواریانس C انجام می‌دهیم. تابع power_iteration برای محاسبه مقادیر ویژه و بردارهای ویژه C فراخوانی می شود. همچنین می توانید از روش های دیگری مانند qr_eigen برای انجام تجزیه استفاده کنیم که هردو در کد پیاده سازی شده‌اند.
نکته قابل توجه این است که بسته به نوع انتخابي باید مقدار آستانه را نیز تنظم کنیم تا بهترین نتیجه را بگیریم.

مرحله ۳: مقادیر ویژه و بردارهای ویژه را بر اساس مقادیر ویژه به ترتیب نزولی مرتب می‌کنیم. تابع np.argsort برای بدست آوردن شاخص های مرتب

شده استفاده می‌شود و سپس مقادیر ویژه و بردارهای ویژه بر این اساس بازآرایی می شوند.

مرحله ۴: مقادیر مفرد را با گرفتن قدر مطلق مقادیر ویژه و سپس جذر آن محاسبه کنید.
مرحله ۵: بردارهای منفرد چپ U را با ضرب A در بردارهای ویژه مرتب شده و تقسیم بر مقادیر منفرد محاسبه کنید.

مرحله ۶: بردارهای مفرد سمت راست VT را با جابجایی بردارهای ویژه مرتب شده عادی کنید.

در نهایت، تابع بردارهای منفرد چپ U، مقادیر منفرد و بردارهای منفرد راست نرمال شده VT را برمی گرداند.

## ۳. توابعی که برای محاسبه‌ی مقادیر ویژه استفاده می‌شوند.

۱. qr_eig(A, max_iter=50):
یک ماتریس A را به عنوان ورودی و یک پارامتر اختیاری max_iter می گیرد که حداکثر تعداد تکرارها را برای الگوریتم QR مشخص می کند.
الگوریتم QR یک روش تکراری است که به طور مکرر از فرآیند گرم اشمیت برای تجزیه ماتریس A به یک ماتریس متعامد Q و یک ماتریس مثلثی بالا R استفاده می کند. این فرآیند حداکثر بار انجام می شود.
پس از تکرارها، ماتریس A به ضرب R و Q به روز می شود که ماتریسی مشابه با A اصلی است اما با تخمین های ارزش ویژه بهبود یافته است.
عناصر مورب ماتریس به روز شده A به عنوان مقادیر ویژه استخراج می شوند.
برای هر مقدار ویژه، کد با کم کردن مقدار ویژه ضربدر ماتریس هویت از A، یک ماتریس A_eig می سازد.

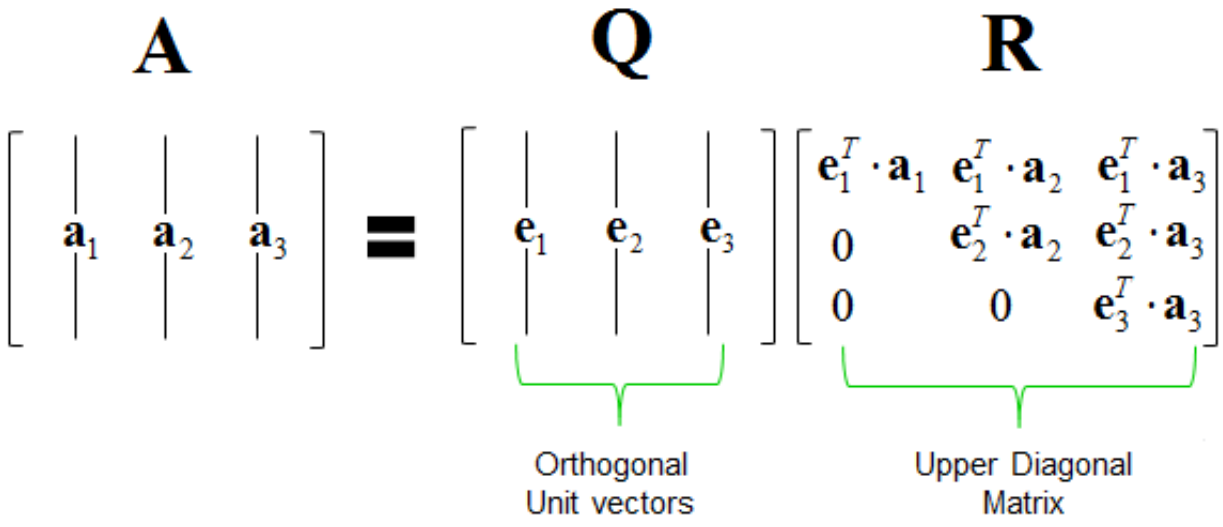
۲. power_method(A, max_iter=50):

برای یافتن بردار ویژه مربوط به هر مقدار ویژه فراخوانی می شود. تابع power_method احتمالاً روش تکرار توان را برای یافتن بردار ویژه غالب یک ماتریس پیادهسازی میکند.
مقادیر ویژه و بردارهای ویژه به صورت چند تایی بازگردانده می شوند و بردارهای ویژه به صورت ستون هایی در یک آرایه مرتب شده اند.

۳. gram_schmidt(A):

فرآیند گرام اشمیت برای متعامد کردن یک ماتریس A و تجزیه آن به یک ماتریس متعامد Q و یک ماتریس مثلثی بالایی R پیاده سازی میکنیم.

$$
\begin{aligned}
\mathbf{u}_1 &= \mathbf{a}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{a}_2 - \operatorname{proj}_{\mathbf{u}_1} \mathbf{a}_2, & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{a}_3 - \operatorname{proj}_{\mathbf{u}_1} \mathbf{a}_3 - \operatorname{proj}_{\mathbf{u}_2} \mathbf{a}_3, & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
&\vdots & &\vdots \\
\mathbf{u}_k &= \mathbf{a}_k - \sum_{j=1}^{k-1} \operatorname{proj}_{\mathbf{u}_j} \mathbf{a}_k, & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}
\end{aligned}
$$



## ۴. توابع کاربردی بصورت کلی:

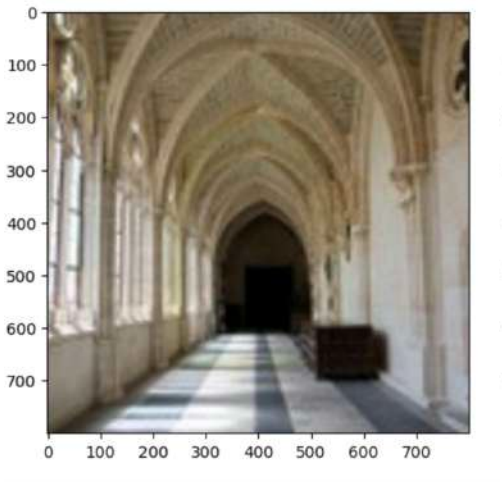
۱. norm(vector):

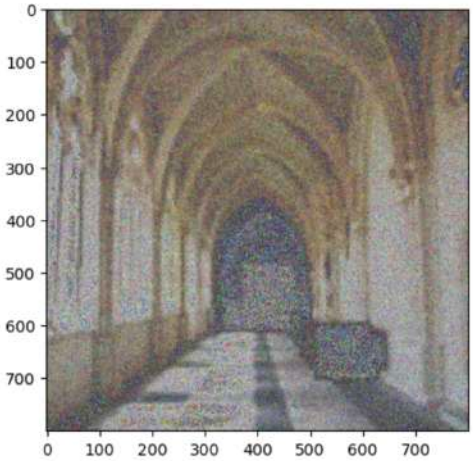
همان محاسبه‌ی نرم اقلیدسی، یا به اصطلاح نرم دوم است.

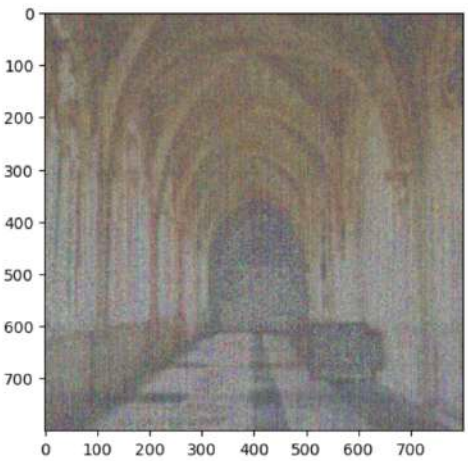
$$u = \begin{bmatrix} u_1 \\ u_2 \\ \cdots \\ u_n \end{bmatrix}$$

$$||u||_2 = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2}$$

قسمت نتایج و تصاویر:



درصورت استفاده از توابع دست پیاده سازی شده برای قسمت مقادیر ویژه چون در svd یک پارامتر طبق عنوان iteration داریم و برای نتیجه‌ی بهینه باید عدد آن بالا مانند 1000 و سطح آستانه 100 باشد.
در این حالت ممکن است پردازش طولانی شود اما نتایج بالا را به ما می‌دهد.
درصورتی که در این تابع از محاسبه آماده‌ی مقادیر ویژه استفاده کند عدد آستانه اگر برابر 40 باشد تصاویر با کیفیت بهتر ارائه خواهند شد.

بطور کلی با افزایش سطح آستانه به عبارتی تعداد برداها و مقادیر منفرد دینویزینگ عکس بهتر می‌شود اما از حدی به بعد عکس دیگر قابل تشخیص نیست و همه قسمت‌های ان باهم ترکیب شده و فقط قسمت‌های جزئي و خاصی از آن مشخص می‌شود.