دانشگاه صنعتی شریف

دانشکده مهندسی برق

مسعود ناطقی ۹۶۱۰۲۵۶۷

تمرین درس تجزیه‌های تانسوری

دکتر حاجی‌پور

۱.

```
function [G, U1, U2, U3] = hooi(T, R1, R2, R3)
[I, J, K] = size(T);
% hosvd initialization
[U1, U2, U3] = hosvd(T, R1, R2, R3);
numItter = 10;
for i = 1:numItter
    i %#ok<NOPRT>
    Z1 = kruskal_tucker(T, eye(I), U2', U3');
    [U, ~, ~] = svd(unfold(Z1, 1));
    U1 = U(:, 1:R1);

    Z2 = kruskal_tucker(T, U1', eye(J), U3');
    [U, ~, ~] = svd(unfold(Z2, 2));
    U2 = U(:, 1:R2);

    Z3 = kruskal_tucker(T, U1', U2', eye(K));
    [U, ~, ~] = svd(unfold(Z3, 3));
    U3 = U(:, 1:R3);
end
G = kruskal_tucker(T, U1', U2', U3');
```

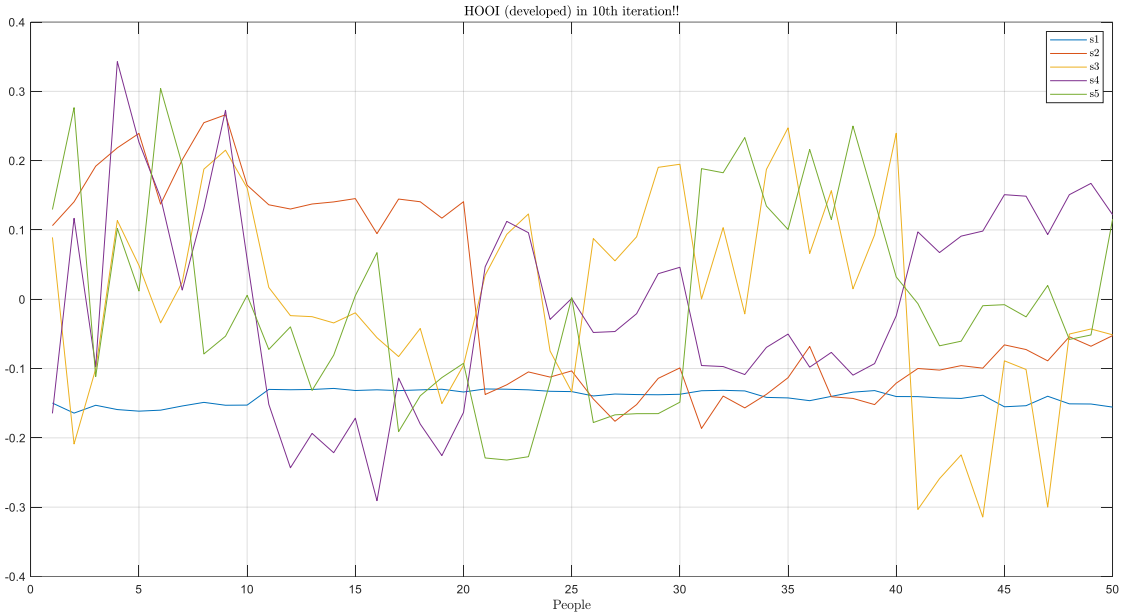
در ادامه برای بررسی صحت کارکرد الگوریتم از یک مثال استفاده کردیم.

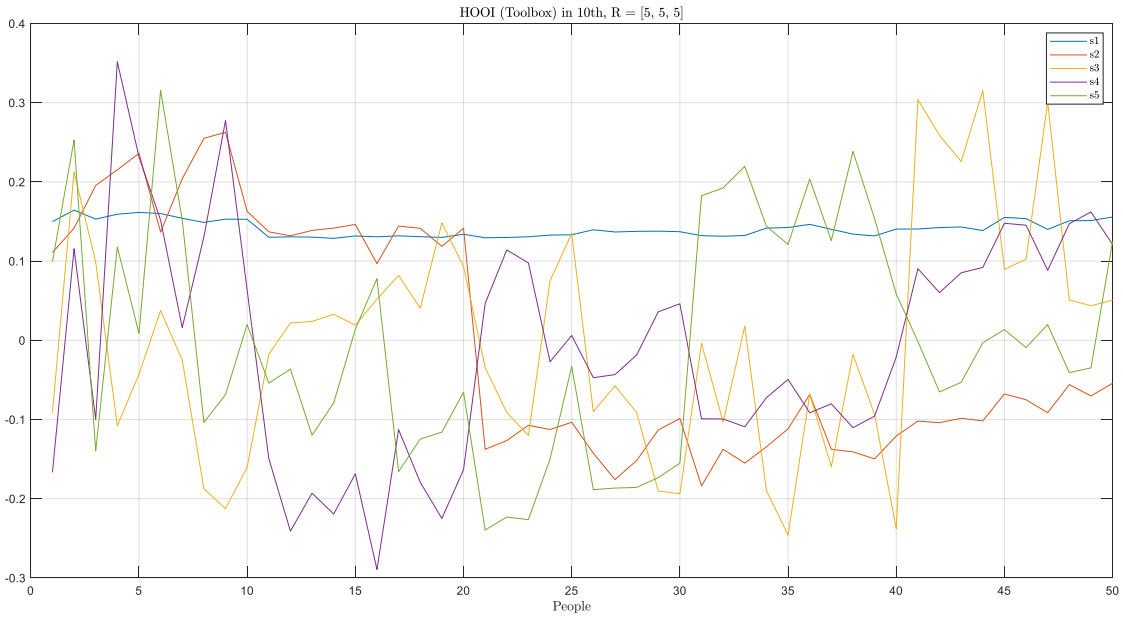
```
X(:,:,1) =

    0.3044    0.4145    0.0278
    0.5889    0.3807    0.2933
    0.6126    0.7163    0.0840
    0.4007    0.8773    0.1230


X(:,:,2) =

    0.7537    0.6280    0.1976
    0.4775    0.8674    0.0450
    0.9580    0.9319    0.0624
    0.0535    0.7594    0.0065
```

```
X_hat(:,:,1) =

    0.3279    0.4271    0.0593
    0.6364    0.3718    0.1508
    0.5749    0.7084    0.1070
    0.4270    0.8737    0.0523


X_hat(:,:,2) =

    0.7659    0.5951    0.1699
    0.4725    0.8623    0.0660
    0.9195    0.9709    0.1839
    0.0673    0.7498   -0.0397
```

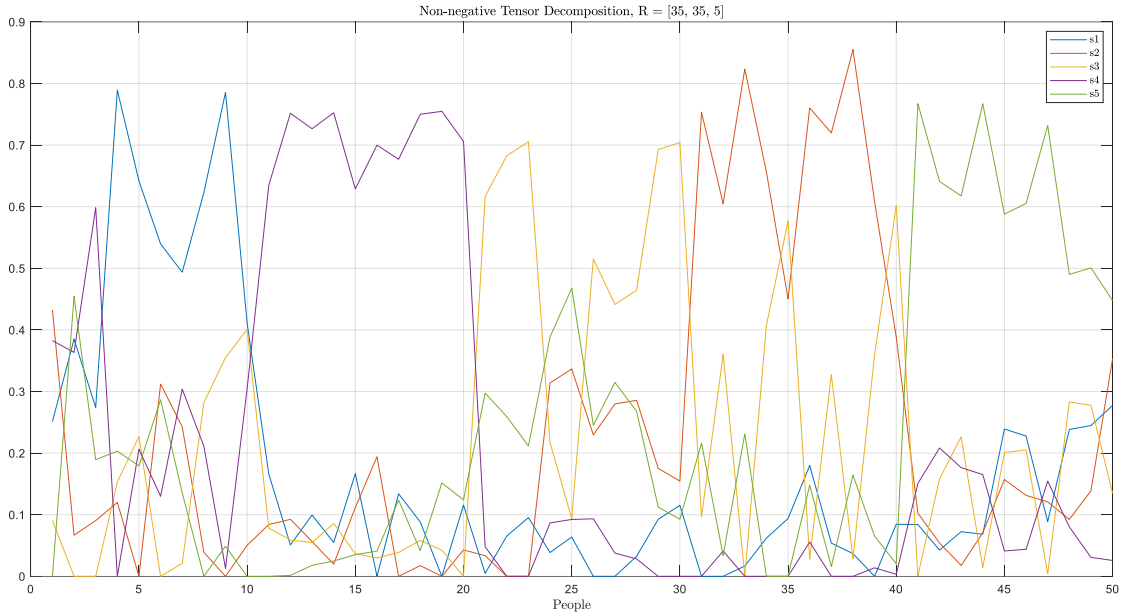
هم چنین مقدار خطا برابر $\left|X - \hat{X}\right|_F = 0.2339$ شد.

۲. این کار را به وسیله سه تابع انجام دادیم. تابع hooi.m که خودمان آن را شبیه‌سازی کردیم. تابع tucker_als.m از تولباکس tensor_toolbox-v3.2.1 و تابع ntd.m[1] که همان Non-negative Tensor Decoposition را انجام می‌دهد.





[1] https://www.caam.rice.edu/~optimization/bcu/ntd/index.html



با مقایسه نمودارها می‌توان فهمید که الگوریتم‌های HOOI به هیچ عنوان نمی‌توانند عملکرد خوبی داشته باشند. هم الگوریتم شبیه‌سازی‌شده و هم الگوریتم تولباکس عملکرد بسیار بدی دارند و نمی‌توانند روی این مجموعه داده خوشه‌بندی را انجام دهند. در عین حال عملکرد الگوریتم Non-negative Tensor Decoposition بسیار مطلوب است. در محور x هر ۱۰تا عکس مربوط به یک نفر است و ملاحظه می‌شود که ستون‌های ماتریس فاکتور $A^{(3)}$ تنها برای یکی از افراد مقدار بزرگی دارند و برای مابقی افراد مقدار آن‌ها به نسبت کوچک‌تر است. به عبارت دیگر منحنی اول برای شخص اول مقدار بزرگی‌دارد و برای مابقی افراد مقدار کوچکی دارد و به همین ترتیب برای بقیه منحنی‌ها. بنابراین خوشه‌بندی با استفاده از الگوریتم ntd کاملا موفقیت آمیز بود. زیرا تصویر یک آرایه با مقادیر مثبت است و بنابراین قید نامنفی بودن این الگوریتم در خوشه‌بندی موثر بسیار کمک می‌کند.

۳. الف)

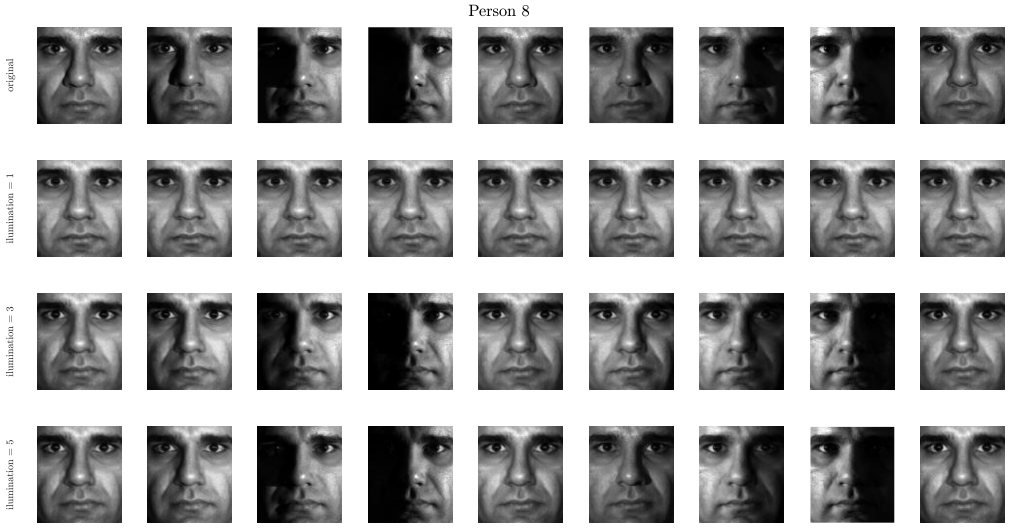

Person 1
original
illumination = 1
illumination = 3
illumination = 5
Person 2
original
illumination = 1
illumination = 3
illumination = 5
Person 3
original
illumination = 1
illumination = 3
illumination = 5

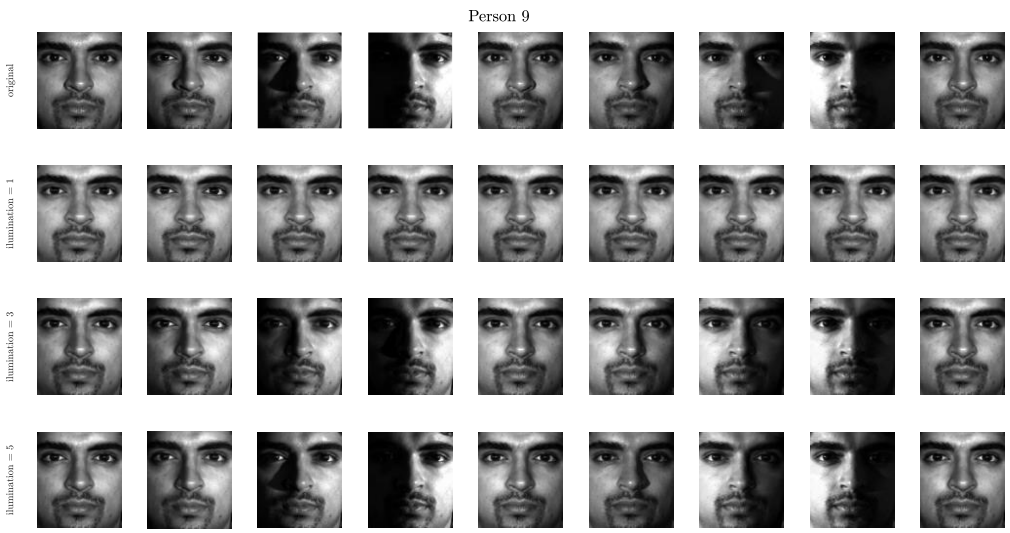
Person 4

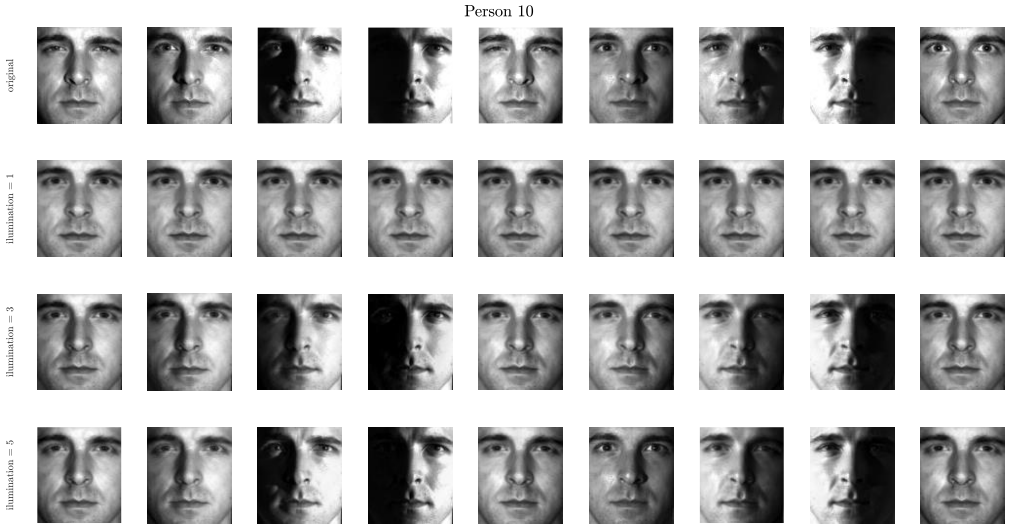
Person 5

Person 6

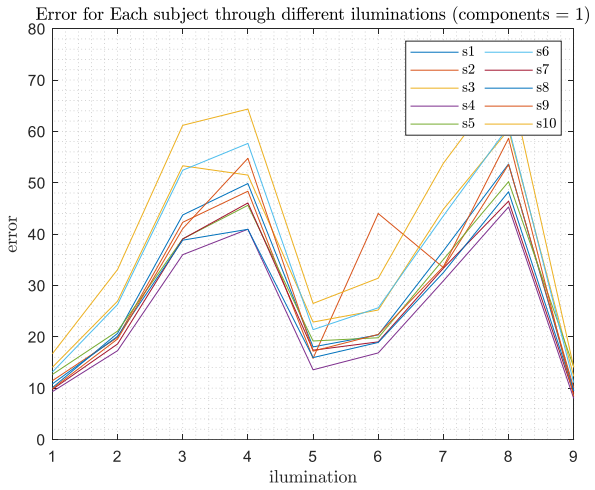

Person 7
original
illumination = 1
illumination = 3
illumination = 5

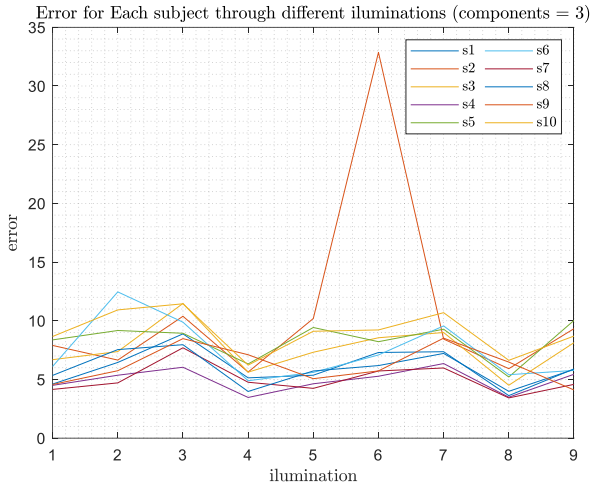

Person 8
original
illumination = 1
illumination = 3
illumination = 5

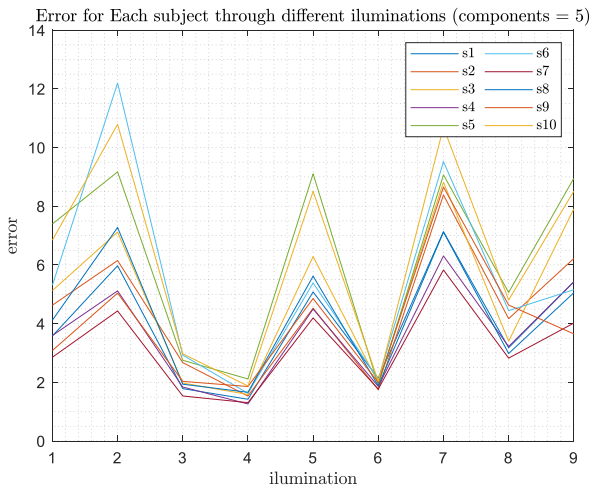

Person 9
original
illumination = 1
illumination = 3
illumination = 5

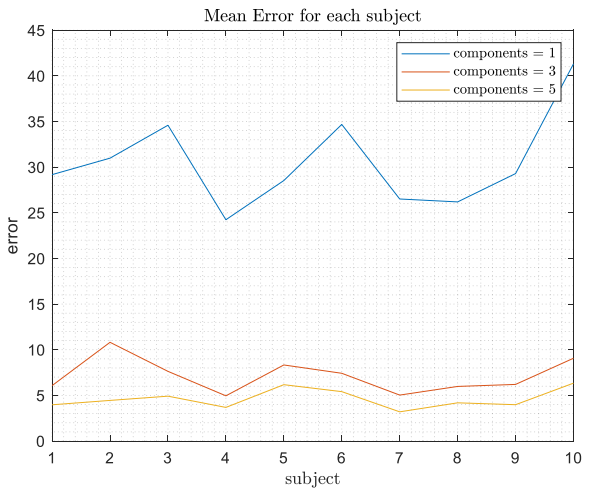


با مقایسه تصاویر مشاهده می‌کنیم هر چه از تعداد مولفه‌های کمتری استفاده کنیم، سایه‌ای که روی عکس‌ها می‌افتد بییشتر مهو می‌شود. با استفاده از تعداد مولفه‌های بیشتر تصاویر دقیق‌تر بازسازی می‌شوند اما ممکن است برای کاربردهایی مثلا شناسایی فرد در حالت‌های مختلف نوری از تعداد مولفه‌های کمتری استفاده کنیم تا تصاویر حالات نوری مختلف شبیه یک تصویر واحد بشوند. این موضوع از نمودارهای زیر نیز قابل برداشت است. هر قدر از تعداد مولفه‌های بیشتری استفاده کنیم خطا کمتر می‌شود.









(ب


Person 1
original
components = 10
components = 30
components = 50


Person 2
original
components = 10
components = 30
components = 50


Person 3
original
components = 10
components = 30
components = 50

Person 4

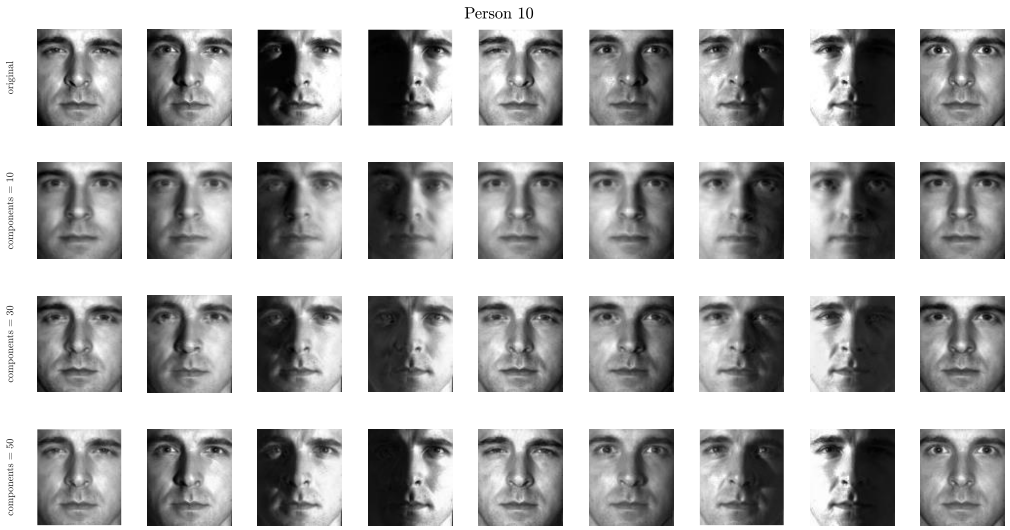
Person 5

Person 6

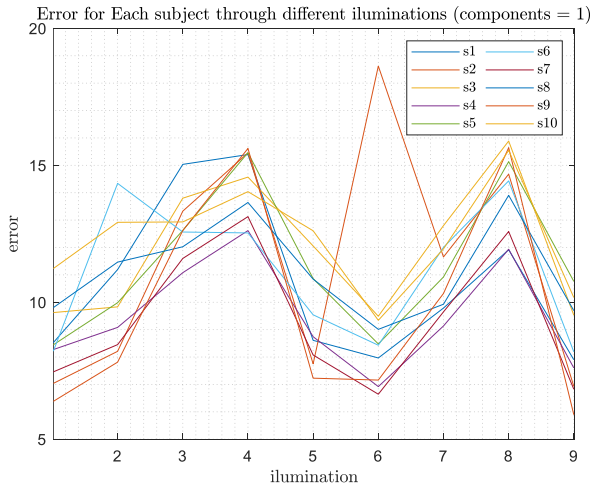
Person 7

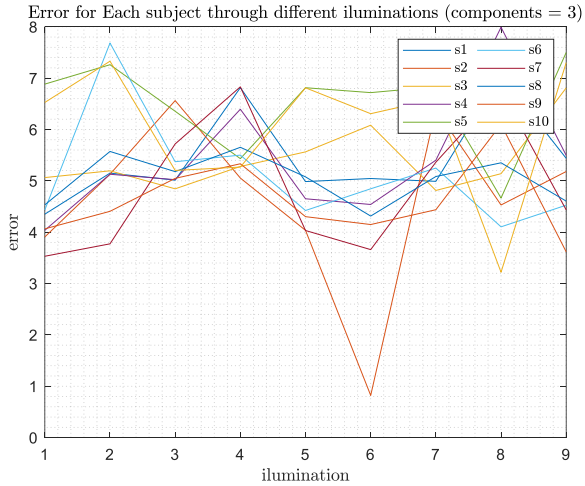
Person 8

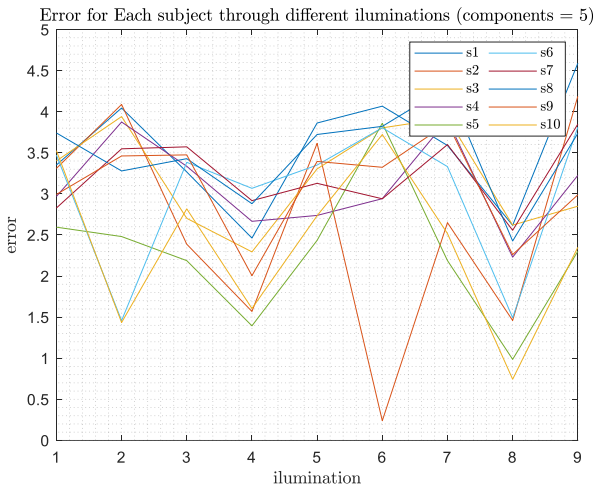
Person 9

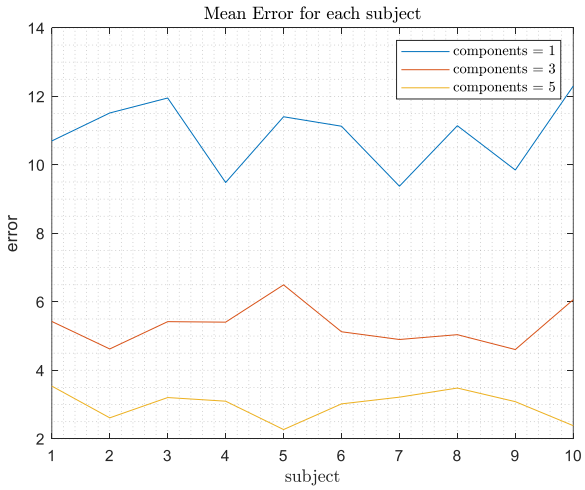


در این روش نیز مشابه قبل هر چه از تعداد مولفه‌های بیشتری استفاده کنیم، جزئیات بیشتری را می‌توانیم نمایش دهیم و خطای بازسازی کمتر می‌شود. این روش بر خلاق قبل برای حذف سایه‌ها از همه‌جای تصویر استفاده می‌کند و برای حذف سایه به اندازه روش تجزیه tucker موفق نیست. این مطلب از نمودارهای زیر مشخص است.









ج) با مقایسه تصاویر حاصل از دو روش متوجه می‌شویم که روش Tucker در حذف سایه‌ها از تصاویر، بسیار بهتر عمل می‌کند. اما در عین حال تجزیه Tucker خطای بیشتری در بازسازی تصاویر دارد. روش SVD سعی می‌کند اطلاعات تصویر را از همه جا کم کند و بنابراین تصویر حاصل از این روش محوتر است و جزئیات کمتری دارد. بنابراین علی‌رغم اینکه تجزیه tucker خزای بزرگ‌تری دارد اما در حذف سایه‌ها بهتر عمل می‌کند. هم‌چنین کیفیت تصویر با این روش بهتر است و خروجی به تصویر اصلی نزدیک‌تر است. این موضوع را در نمودار زیر نشان دادیم.