سوال ۱: با توجه به مطالب کلاس، الگوریتم HOOI برای تجزیه Tucker یک تانسور با رتبه دلخواه را پیاده‌سازی می‌کنیم.

```
function [G, U] = HOOI(T, R)
    number_of_itterations = 100;
    U = cell(1, length(R));
    for i = 1:length(R)
        Tn = tenmat(T, i);
        [Un, ~, ~] = svd(Tn.data);
        U{i} = Un(:, 1:R(i));
    end
    for i = 1:number_of_itterations
        for j = 1:length(R)
            temp_U = U;
            for k = 1:length(R)
                temp_U{k} = pinv(temp_U{k});
            end
            temp_U(j) = [];
            temp_R = 1:length(R);
            temp_R(j) = [];
            Z = ttm(T, temp_U, temp_R);
            Zn = tenmat(Z, j);
            [Un, ~, ~] = svd(Zn.data);
            U{j} = Un(:, 1:R(j));
        end
    end
    temp_U = U;
    for k = 1:length(R)
        temp_U{k} = pinv(temp_U{k});
    end
    G = ttm(T, temp_U, 1:length(R));
end
```

برای بررسی درستی الگوریتم پیاده‌سازی شده، عملکرد آن را به ازای یک تانسور دلخواه با ابعاد $I = [5, 3, 2]$، ارزیابی می‌کنیم. برای ایجاد تانسور مذکور، یک تانسور هسته تصادفی با ابعاد $R = [3, 2, 2]$ را در ماتریس‌های عامل متعامد یکه تصادفی با ابعاد مناسب ضرب می‌کنیم. ورودی به صورت زیر بدست می‌آید:

```
T is a tensor of size 5 x 3 x 2
    T(:,:,1) =
        0.1051   -0.2093    0.2068
        0.0401   -0.0550    0.0802
        0.0265   -0.0300    0.0535
       -0.0338    0.0819   -0.0656
        0.0324   -0.0492    0.0646
    T(:,:,2) =
        0.7359   -0.7946    1.4856
        0.1734    0.0395    0.3627
        0.1729    0.0038    0.3597
        0.0419    0.0169    0.0880
        0.1910   -0.0725    0.3930
```

حال با استفاده از الگوریتم پیاده‌سازی شده، تانسور هسته و ماتریس‌های عامل تانسور ورودی را بدست می‌آوریم. خروجی به صورت زیر بدست می‌آید:

```
G is a tensor of size 3 x 2 x 2
    G(:,:,1) =
        1.9823   -0.0035
       -0.0018   -0.2886
        0.0026   -0.0132
    G(:,:,2) =
        0.0036    0.1024
        0.0225    0.0245
       -0.1201   -0.0457

U1 =

   -0.9382    0.3294   -0.0176 U2 =
   -0.1856   -0.6671    0.2469                      U3 =
   -0.1877   -0.5433   -0.0527   -0.4069    0.1509
   -0.0320   -0.1657   -0.9661    0.3945    0.9186    0.1622    0.9868
   -0.2215   -0.3521    0.0517   -0.8239    0.3653    0.9868   -0.1622
```

سپس بر اساس خروجی‌های الگوریتم، تانسور ورودی را بازسازی می‌کنیم. خروجی به صورت زیر بدست می‌آید:

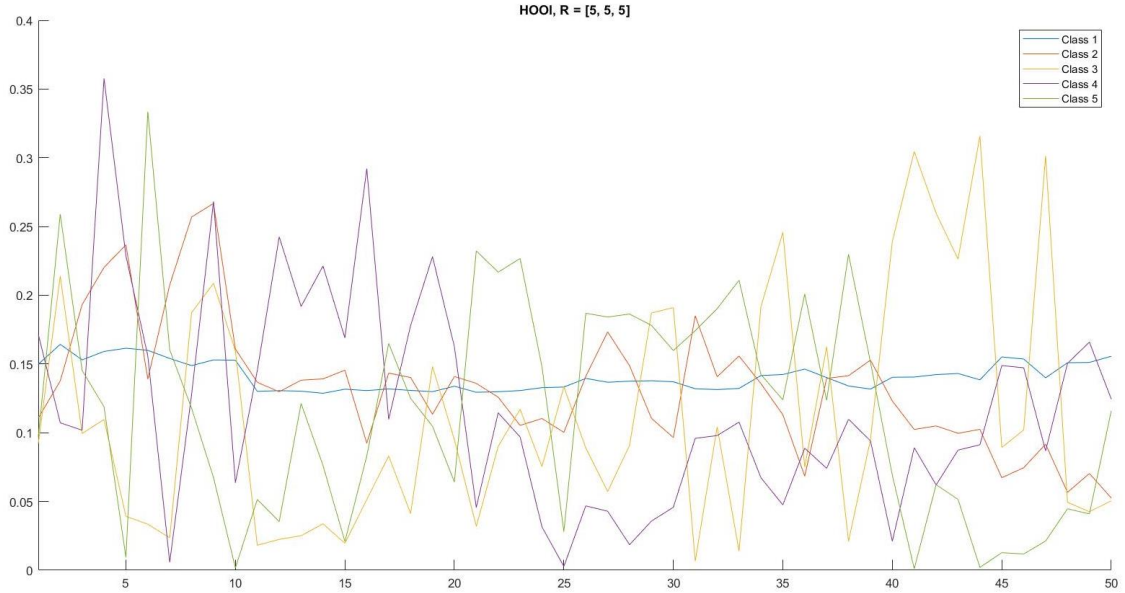
```
recovered_T is a tensor of size 5 x 3 x 2
    recovered_T(:,:,1) =
        0.1051   -0.2093    0.2068
        0.0401   -0.0550    0.0802
        0.0265   -0.0300    0.0535
       -0.0338    0.0819   -0.0656
        0.0324   -0.0492    0.0646
    recovered_T(:,:,2) =
        0.7359   -0.7946    1.4856
        0.1734    0.0395    0.3627
        0.1729    0.0038    0.3597
        0.0419    0.0169    0.0880
        0.1910   -0.0725    0.3930
```

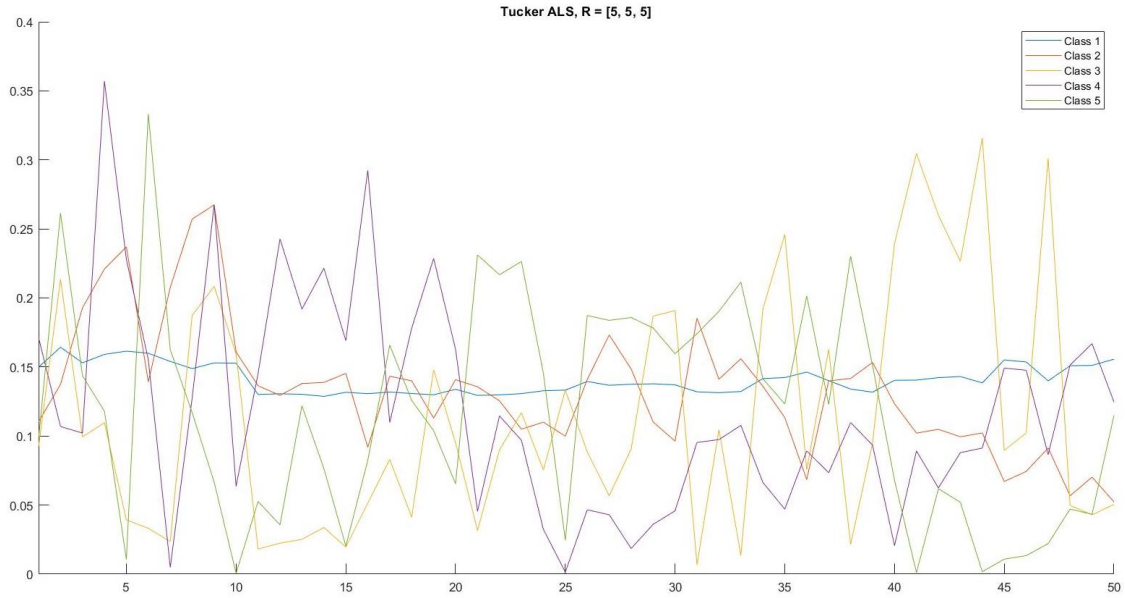
$$\|\mathrm{T} - \mathrm{recovered\_T}\|_{\mathrm{F}} = 1.1212 * 10^{-15}$$

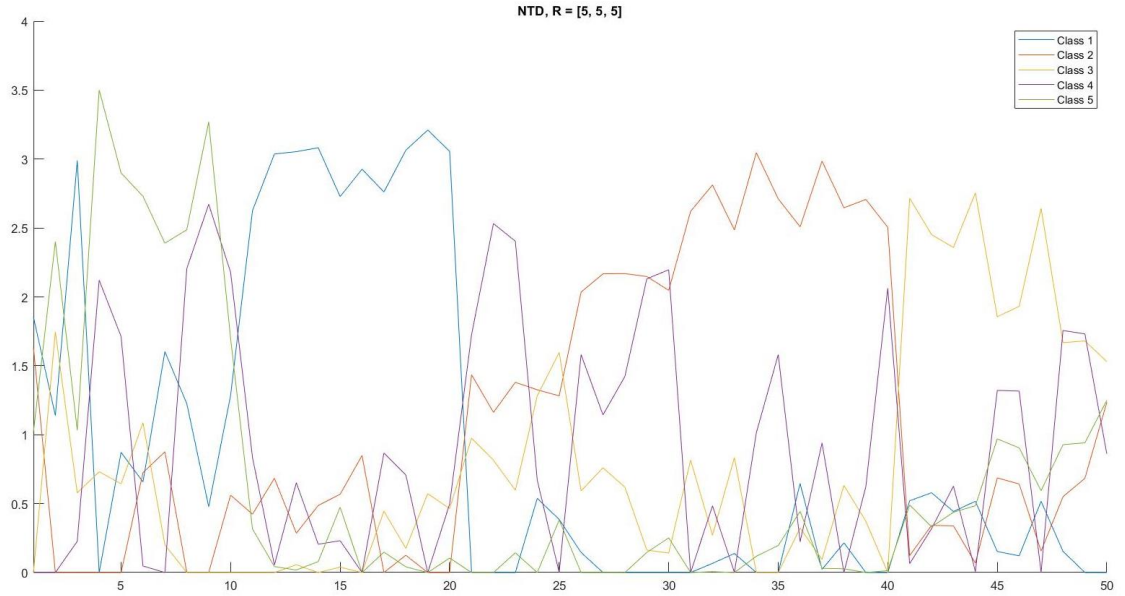
به وضوح الگوریتم پیاده‌سازی شده، عملکرد مناسبی داشته است و تجزیه Tucker تانسور مورد نظر را به درستی بدست آورده است.

سوال ۲ موارد خواسته شده را به ترتیب به ازای دو حالت $R = [5, 5, 5]$ و $R = [30, 25, 5]$ انجام می‌دهیم.

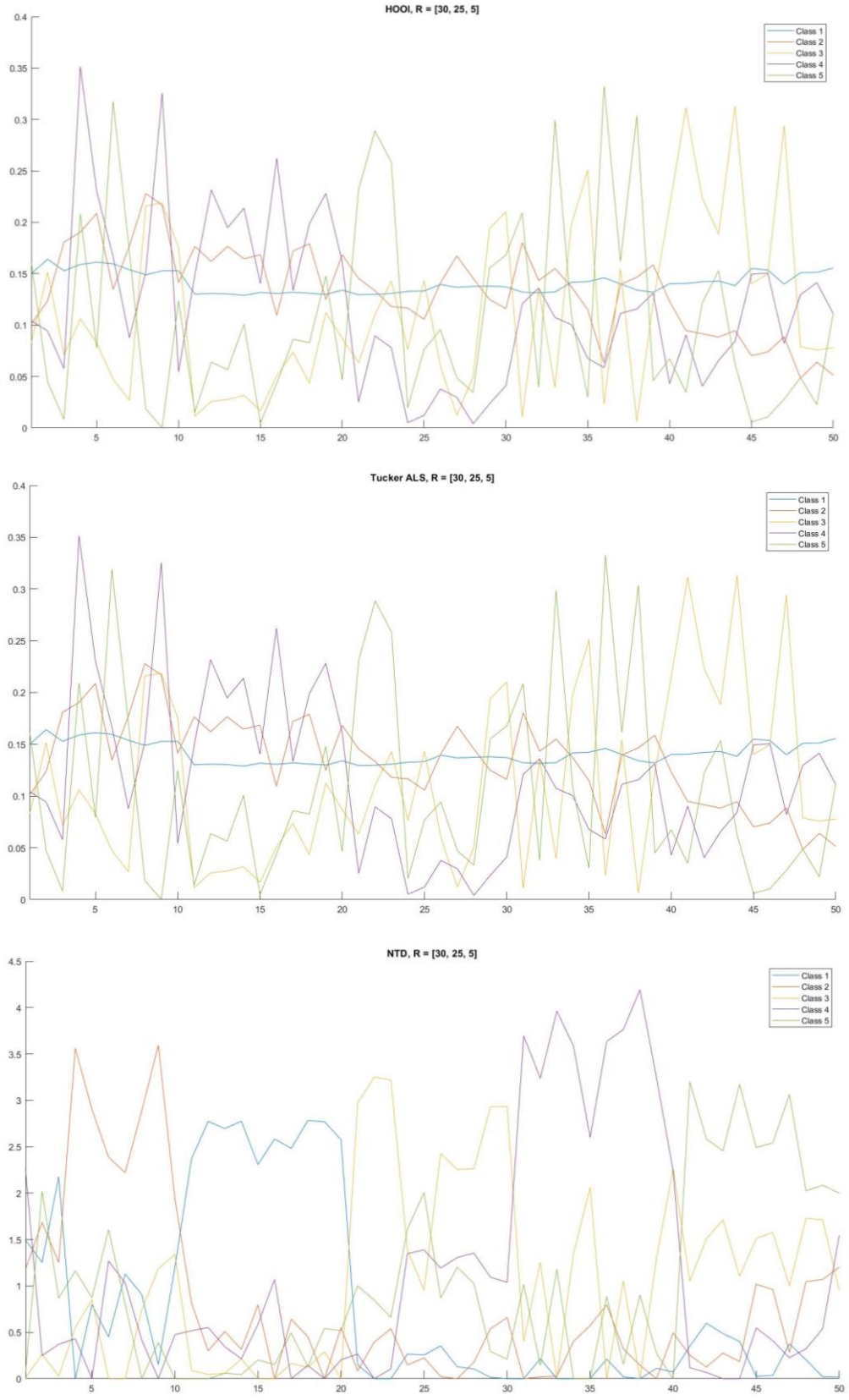
برای حالت $R = [5, 5, 5]$ خروجی به صورت زیر بدست می‌آید:





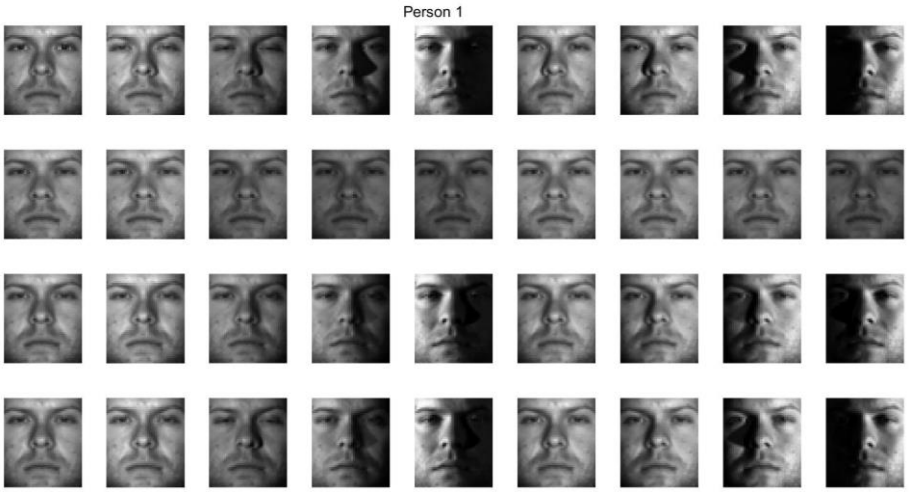


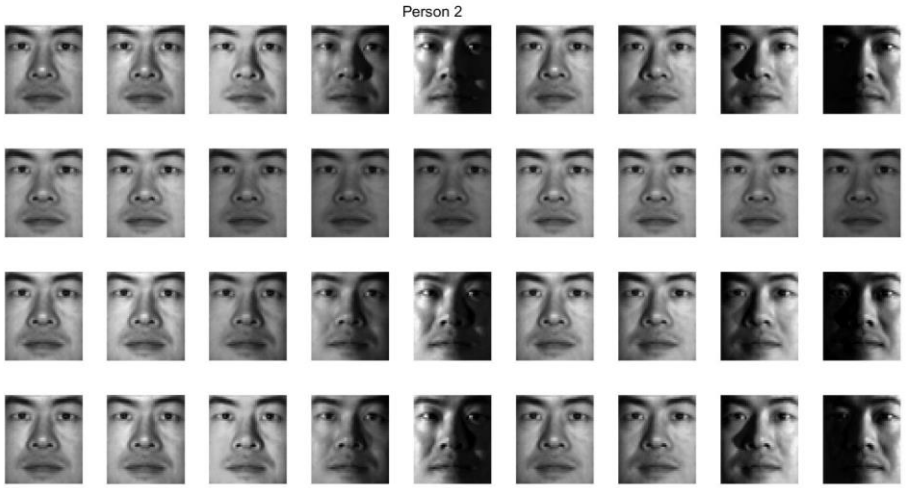
برای حالت $R = [30, 25, 5]$ خروجی به صورت زیر بدست می‌آید:

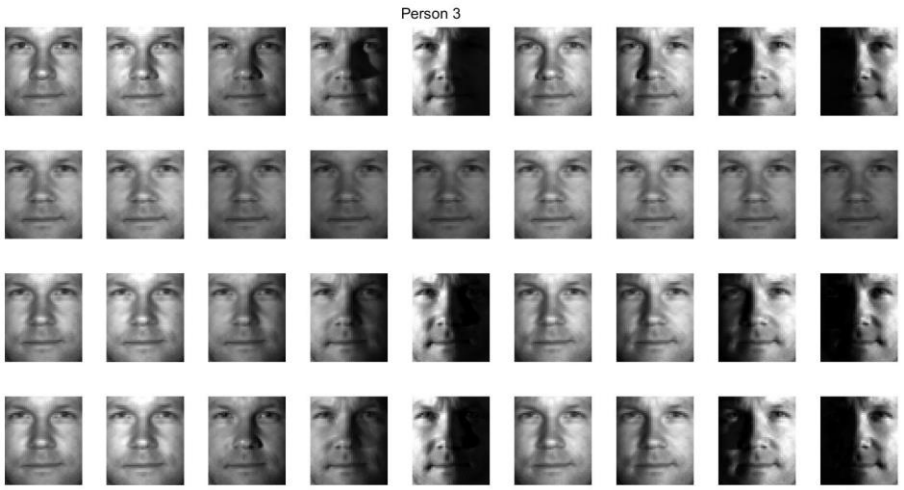


به وضوح مشخص است که عملکرد الگوریتم تجزیه Non-negative Tucker بهتر از عملکرد الگوریتم‌های تجزیه Tucker است. در حالی که در الگوریتم‌های تجزیه Tucker، یک یا حتی دو مولفه از مولفه‌های تشخیص داده شده، در همه تصاویر تقریبا به میزان یکسان حضور دارند و به نوعی معرف میانگین هستند، هر یک از مولفه‌های خروجی الگوریتم تجزیه Non-negative Tucker، تقریبا در تصاویر مربوط به یک شخص، مقدار بزرگی دارند و در بقیه تصاویر با ضرایب کوچکی شرکت کرده اند. همچنین لازم به ذکر است که دقت این الگوریتم با افزایش بعد تانسور هسته در مدهای دیگر، افزایش یافته است.

سوال ۳: الف) موارد خواسته شده را به ترتیب انجام می‌دهیم. خروجی به صورت زیر بدست می‌آید (در هر تصویر، سطر اول مربوط به تصاویر اصلی و سطرهای بعدی مربوط به تصاویر بازسازی شده به ازای مقادیر ۱، ۳ و ۵ برای بعد فضای illumination هستند):





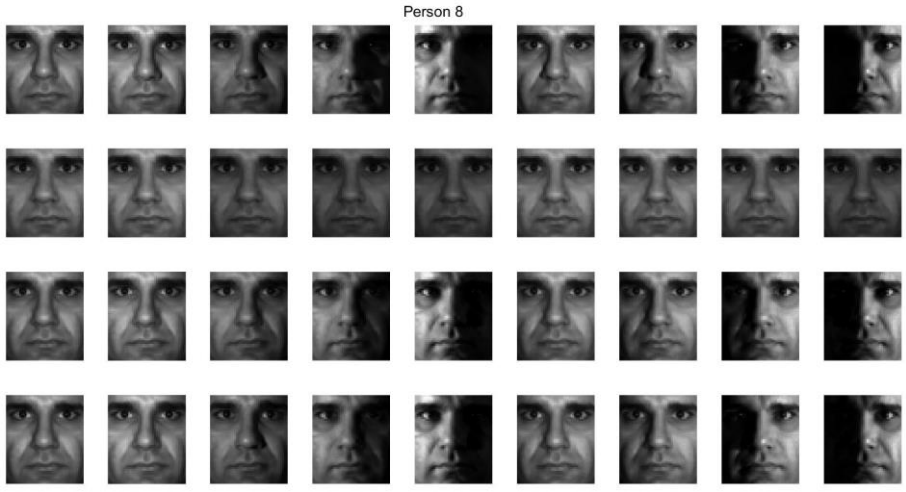


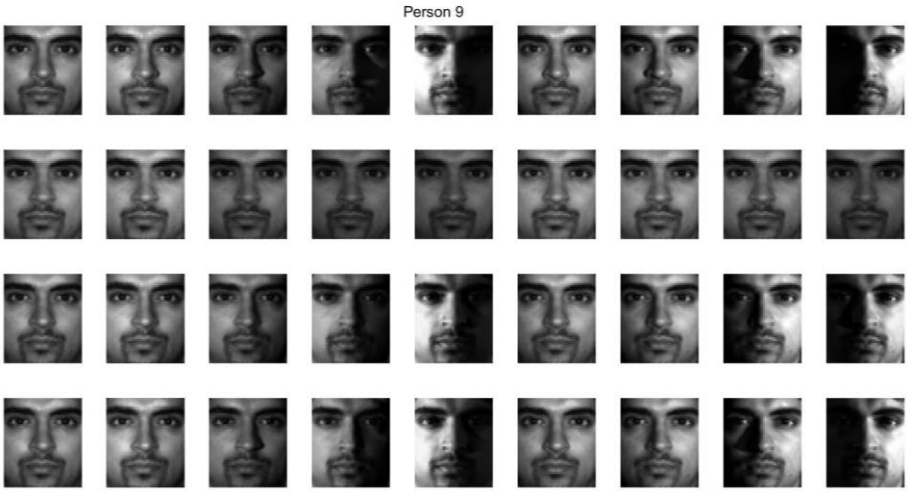
Person 4

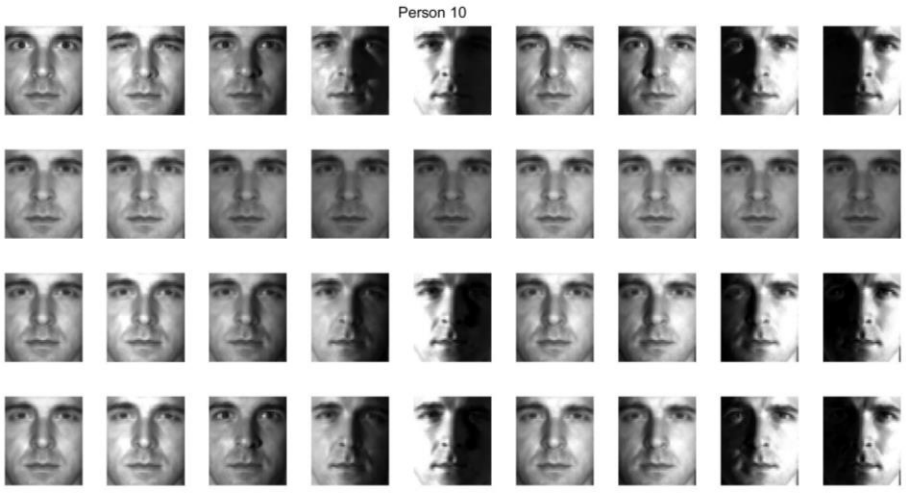
Person 5

Person 6

Person 7

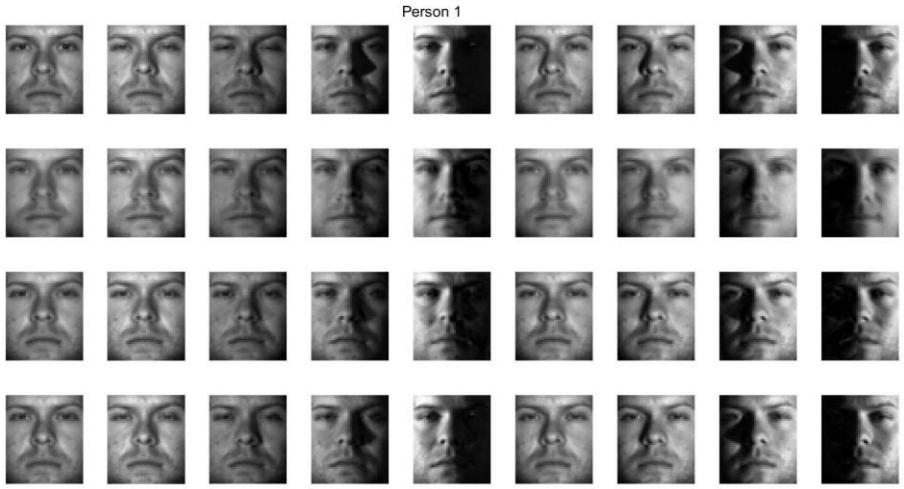


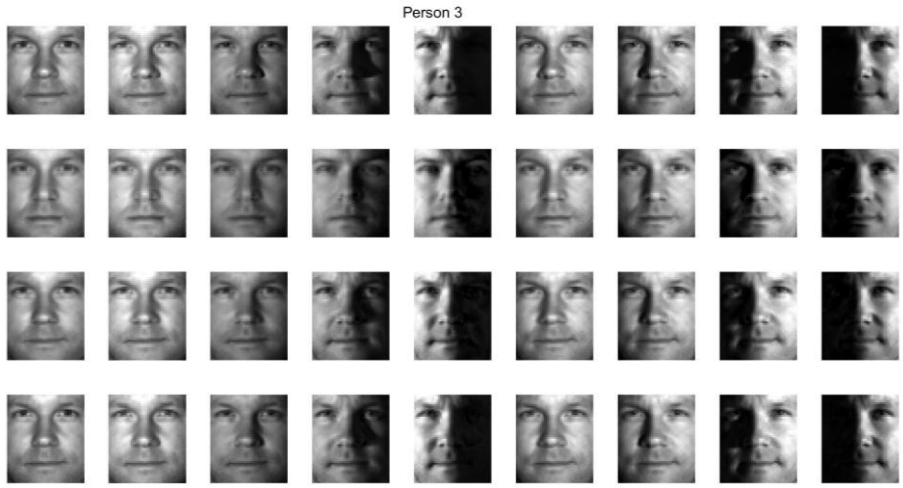


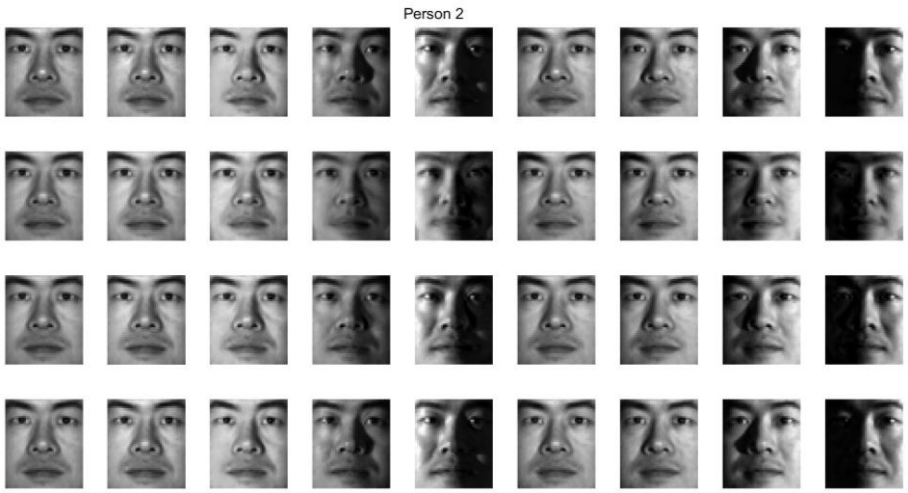


با توجه به تصاویر بدست آمده، هر چه بعد فضای illumination کمتر باشد، میزان حذف اطلاعات مربوط به جهت تابش نور بیشتر است و در نتیجه خروجی تمرکز بیشتری روی بدست آوردن صورت فرد به ازای تابش نور از جهت رو به رو خواهد داشت. با افزایش بعد فضای illumination می‌توان میزان تاثیر اطلاعات مربوط به جهت تابش نور در بازسازی تصاویر را کنترل نمود.

ب) موارد خواسته شده را به ترتیب انجام می‌دهیم. خروجی به صورت زیر بدست می‌آید (در هر تصویر، سطر اول مربوط به تصاویر اصلی و سطرهای بعدی مربوط به تصاویر بازسازی شده به ازای مقادیر ۱۰، ۳۰ و ۵۰ برای رتبه تجزیه SVD هستند):





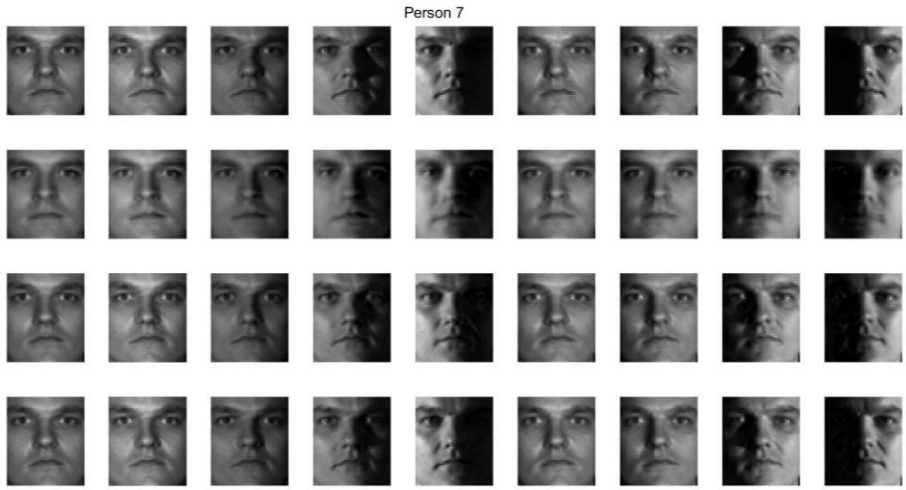


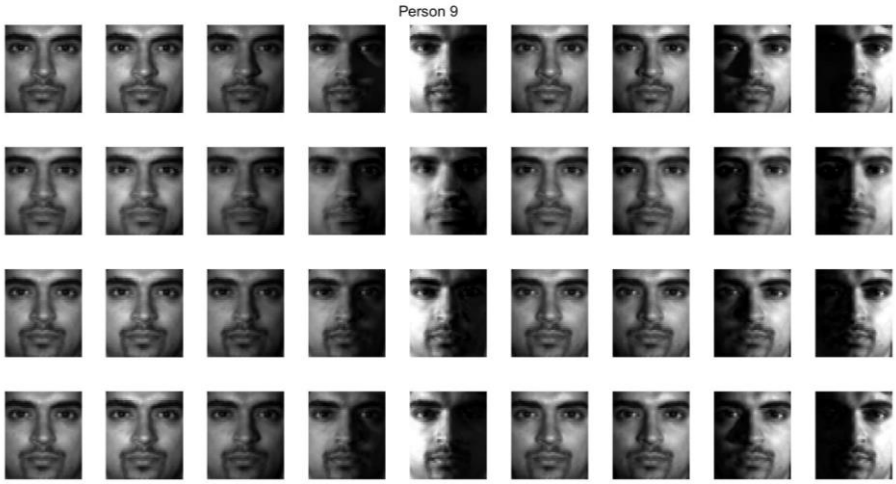
Person 4

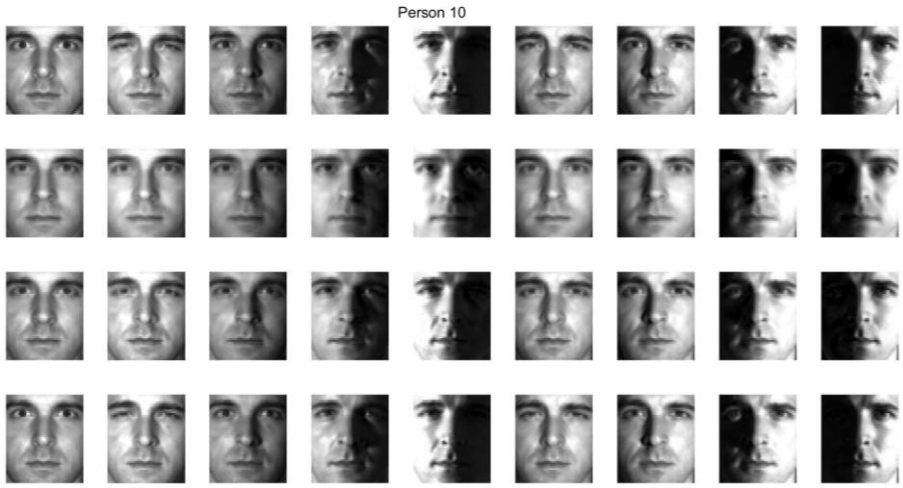
Person 5

Person 6

Person 7







هر چه رتبه تجزیه SVD کمتر باشد، دقت بازسازی تصاویر کمتر است و جزئیات بیشتری حذف می‌شوند. شایان ذکر است حذف جزئیات، در همه پیکسل‌های تصاویر، تقریبا به طور یکسان اتفاق می‌افتد و در نتیجه این تجزیه تاثیری در حذف اطلاعات مربوط به جهت تابش نور ندارد. با افزایش رتبه تجزیه SVD می‌توان میزان حذف جزئیات در بازسازی تصاویر را کنترل نمود.

ج) با توجه به اثبات‌های انجام شده، تجزیه SVD رتبه-k، بهترین تخمین رتبه-k است و در نتیجه تصاویر بازسازی شده از نظر اندازه، اختلاف کمی با تصاویر اصلی دارند؛ اما در این تجزیه حذف جزئیات در تمام سطح تصویر، تقریبا به طور یکسان صورت می‌گیرد و کنترلی روی نوع اطلاعات حذف شونده وجود ندارد. در حالی که در تجزیه Tucker، با ساخت تانسور مناسب، می‌توان اطلاعات مربوط به فرد، جهت تابش نور و جزئیات را از یکدیگر جدا کرد و هر یک را به طور مستقل کاهش داد. در نتیجه، با وجود این که بازسازی انجام شده توسط تجزیه Tucker نسبت به بازسازی انجام شده توسط تجزیه SVD، خطای بیشتری دارد، ممکن است حاوی اطلاعات مفیدتری باشد. برای مثال در این تمرین با کاهش بعد فضای illumination توانستیم سایه‌های به وجود آمده در اثر جهت تابش نور را تقریبا به طور کامل حذف کنیم.