CS4220 Assignment 1 Due: 2/5/13 (Tuesday) at 11pm

You must work either on your own or with one partner. You may discuss background issues and general solution strategies with others, but the solutions you submit must be the work of just you (and your partner). If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group. Each problem is worth 5 points. One point may be deducted for poor style.

Topics: Elementary matrix operations in MATLAB. Some connections between geometry and linear algebra. The singular value decomposition.

1 Elliptical Point Clouds

In this problem you are to implement a function that generates a cloud of points that sit inside a given tilted ellipse. First, we cover the background math. We define the ellipse $\mathcal{E}(a,b)$ as the set of points (x,y) that satisfy $(x/a)^2 + (y/b)^2 = 1$. We assume that $a \geq b$ and thus a is the major semiaxis and b is the minor semiaxis. A point (u,v) is inside or on $\mathcal{E}(a,b)$ if

$$\sqrt{(u+f)^2+v^2} \; + \; \sqrt{(u-f)^2+v^2} \; \leq \; a/2 \tag{1}$$

where $f = \sqrt{a^2-b^2}$.

Rotation plus translation in the plane is a 2-by-2 matrix-vector product followed by a vector add:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \tag{2}$$

The point (u,v) is rotated about the origin counterclockwise by ϕ degrees. The x and y components are then adjusted by the xy-translation parameters x_0 and y_0.

Let $\mathcal{E}(a,b,\phi,x_0,y_0)$ be the ellipse obtained by rotating and translating each point on $\mathcal{E}(a,b)$ with rotation angle ϕ (degrees) and xy-translation parameters x_0 and y_0. Note that if (u,v) is inside $\mathcal{E}(a,b)$ then $(\tilde{u},\tilde{v})$ is inside $\mathcal{E}(a,b,\phi,x_0,y_0)$ where $\tilde{u}$ and $\tilde{v}$ are specified by (2). You are now set to complete the following function so that it performs as specified:

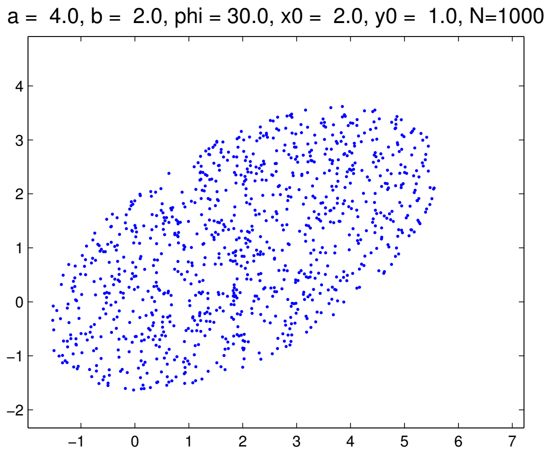
```
  function [x,y] = Cloud(a,b,phi,x0,y0,N)
% a,b,phi,x0,y0 are scalars with 0<b<=a and -180<=phi<=180.
% N is a positive integer with N>=100.
% x and y are column N-vectors that specify a cloud of uniformly distributed
% random points that are inside or on E(a,b,phi,x0,y0).
```

Hints. We can generate m uniformly distributed points inside the rectangle with vertices (a,b), $(-a,b)$, $(-a,-b)$, and $(a,-b)$ as follows:

```
x = a*(-1+rand(m,1));
y = b*(-1+rand(m,1));
```

Since the area of the rectangle is $4ab$ and the area of $\mathcal{E}(a,b)$ is πab, we expect that approximately $m(\pi/4)$ points will be inside $\mathcal{E}(a,b)$. (Why?) If we set $m =$ `ceil`$(4N/\pi)$, then for large m we can expect that approximately N of the points will be inside $\mathcal{E}(a,b)$. To allow for statistical variation, set $m =$ `ceil`$(5N/\pi)$. With extremely high probability, there will be at least N "inside" points. Use equation (1) and the MATLAB `find` function to determine the indices associated with the inside points. By selecting any N of them you have then produced a size-N $\mathcal{E}(a,b)$ cloud. To complete the implementation of `Cloud`, simply apply (2) to each of the points inside $\mathcal{E}(a,b)$. You will then have the required size-N cloud inside $\mathcal{E}(a,b,\phi,x_0,y_0)$.

Your solution should be completely vectorized for full credit. (It is possible to have a loop-free implementation.) Finally, remember that `phi` is in degrees so you will have to convert to radians before using `cos` and `sin`. A test script `ShowCloud` is available on the course website. Sample output:



Submit your implementation of `Cloud` to CMS.

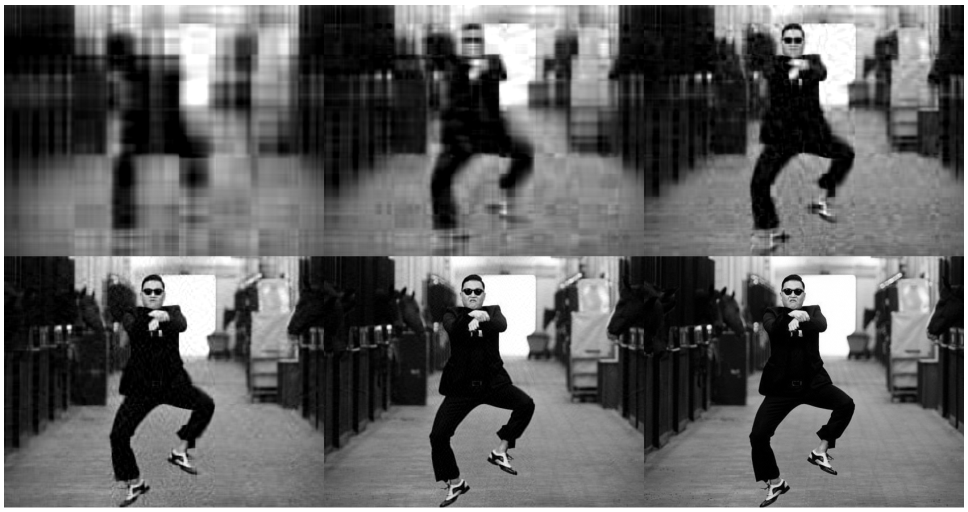
2 Gangnam-Style SVD

Download the script `ShowGangnam` and the file `Gangnam.jpg` from the course website. The script shows how to read a jpeg file into the MATLAB workspace and perform computations that relate to it. Study the script and the insights provided by the comments.

Having the SVD of the Gangnam matrix enables us to generate low-rank approximations of the image. Recall that if $A = U\Sigma V^T$ is the SVD of A and $r \le \text{rank}(A)$, then

$$A_r \;=\; U(:,1{:}r)\Sigma(1{:}r,1{:}r)V(:,1{:}r)^T \;=\; \sum_{k=1}^{r} \sigma_k U(:,k)V(:,k)^T$$

is the closest rank-r matrix to A. Write a script `GangnamSVD` that displays the following image:



This is a 2-by-3 gluing of six images. Let A be the Gangnam matrix as defined in `ShowGangnam`. In the top row of the solution image we have the images based on A_5, A_{10} and A_{20} while the bottom row images are based on A_{40}, A_{80}, and A_{160}. Submit `GangnamSVD` to CMS. (Note. Ignore error messages that indicate that an image is too large to display.)

3 A Point on an Ellipse

An important consequence of the SVD of $A \in \mathbb{R}^{n\times n}$ is that

$$\sigma_n \;\leq\; \|\, Ax \,\| \;\leq\; \sigma_1 \tag{3}$$

where σ_n and σ_1 are the smallest and largest singular values and $x \in \mathbb{R}^n$ has unit 2-norm. To see why this is the case, let us assume that $n = 2$ and that A has the SVD

$$A \;=\; U\Sigma V^T \;=\; [u_1 \mid u_2] \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} [v_1 \mid v_2]^T \;=\; \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T.$$

Here, u_1 and u_2 are the first and second columns of U and v_1 and v_2 are the first and second columns of V. Because these two matrices are orthogonal, we have $u_i^T u_j = \delta_{ij}$ and $v_i^T v_j = \delta_{ij}$. Note that if $c^2 + s^2 = 1$ and

$$x = cv_1 + sv_2, \tag{4}$$

then

$$\|\, x \,\|^2 \;=\; x^T x \;=\; (cv_1 + sv_2)^T(cv_1 + sv_2) \;=\; c^2 v_1^T v_1 + csv_1^T v_2 + csv_2^T v_1 + s^2 v_2^T v_2 \;=\; c^2 + s_2 \;=\; 1$$

so x is a unit vector. Let us see what A does to this vector:

$$\begin{aligned} Ax \;&=\; U\Sigma V^T(cv_1 + sv_2) \;=\; U\Sigma(cV^T v_1 + sV^T v_2) \;=\; U\Sigma\left(c\begin{bmatrix} 1 \\ 0 \end{bmatrix} + s\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) \;=\; U\Sigma\begin{bmatrix} c \\ s \end{bmatrix} \\ &=\; U\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}\begin{bmatrix} c \\ s \end{bmatrix} \;=\; U\begin{bmatrix} c\sigma_1 \\ s\sigma_2 \end{bmatrix} \;=\; [u_1 \mid u_2]\begin{bmatrix} c\sigma_1 \\ s\sigma_2 \end{bmatrix} \;=\; c\sigma_1 u_1 + s\sigma_2 u_2. \end{aligned}$$

It follows that

$$\|\, Ax \,\|_2^2 \;=\; c^2\sigma_1^2 + s^2\sigma_2^2. \tag{5}$$

To make this as large as possible set $c = 1$ and $s = 0$ for then $\|\, Ax \,\|_2 = \sigma_1$. This corresponds to setting $x = v_1$ in (4). To make (5) as small as possible, set $c = 0$ and $s = 1$ in (4) for then $\|\, Ax \,\|_2 = \sigma_2$. This corresponds to setting $x = v_2$. Thus we have shown that (3) holds in the $n = 2$ case.

As a way of making sure that you understand these matrix-vector manipulations associated with the SVD, implement the following function so that it performs as specified:

```
  function x = EllipsePoint(A,lambda)
% A is a 2x2 matrix and lambda satisfies 0<=lambda<=1.
% x is a 2-vector with unit 2-norm length with the property that
%          norm(A*x) = lambda*s1 + (1-lambda)*s2
% where s1 and s2 are the largest and smallest singular values of A.
```

Thus, `EllipsePoint` finds a unit vector so that $\|\, Ax \,\|_2$ takes on a prescribed value defined by λ. If $\lambda = 0$, then x will minimize $\|\, Ax \,\|_2$. If $\lambda = 1$, then x will maximize $\|\, Ax \,\|_2$. Another way to think about what `EllipsePoint` does is to recall that

$$\{\, y \mid y = Ax,\; \|\, x \,\|_2 = 1 \,\}$$

is an ellipse. As you walk around the ellipse, your distance to the origin is never smaller than σ_2 and never larger than σ_1. At least once in your journey your distance to the origin must equal $\lambda\sigma_1 + (1-\lambda)\sigma_2$, a value that is in between the two singular values.

Submit your implementation of `EllipsePoint` to CMS.