VisualRank Using Image Similarity

1. Using the theory above, rank all 1400 MPEG7 shape images using VisualRank.

a. Load the sim.mat data file into MATLAB. This file contains the similarity matrix S for the entire image dataset, as well as a cell array of corresponding filenames.

```
load('sim.mat'); % retrive the variables from the binary file 'sim.mat' into
similarity matrix S and cell array of corresponding filenames
disp(S) % display similarity matrix S
```

```
  Columns 1 through 8
 1.0000    0.9585    0.8079    0.7192    0.9924    0.8633    0.7679    0.8492
 0.9585    1.0000    0.7748    0.6916    0.9530    0.8842    0.7857    0.8392
 0.8079    0.7748    1.0000    0.8702    0.8082    0.7393    0.6907    0.7165
 0.7192    0.6916    0.8702    1.0000    0.7198    0.6704    0.6312    0.6448
 0.9924    0.9530    0.8082    0.7198    1.0000    0.8587    0.7664    0.8437
 0.8633    0.8842    0.7393    0.6704    0.8587    1.0000    0.8895    0.8407
 0.7679    0.7857    0.6907    0.6312    0.7664    0.8895    1.0000    0.8495
 0.8492    0.8392    0.7165    0.6448    0.8437    0.8407    0.8495    1.0000
 0.6913    0.6577    0.5592    0.4778    0.6863    0.6485    0.6284    0.7175
 0.5417    0.5257    0.7032    0.6266    0.5441    0.5676    0.6018    0.5473
```

```
disp(filenames); % display corresponding filenames to the images in matrix S
```

b. Start by setting A=S, then remove elements from A to leave visual hyperlinks between only those images that are positively correlated.

```
A = S; % define adjacency matrix A
A(A<0) = 0; % remove negative values from the matrix A
```

```
A = 1400×1400
    1.0000    0.9585    0.8079    0.7192    0.9924    0.8633    0.7679 ···
    0.9585    1.0000    0.7748    0.6916    0.9530    0.8842    0.7857
    0.8079    0.7748    1.0000    0.8702    0.8082    0.7393    0.6907
    0.7192    0.6916    0.8702    1.0000    0.7198    0.6704    0.6312
    0.9924    0.9530    0.8082    0.7198    1.0000    0.8587    0.7664
    0.8633    0.8842    0.7393    0.6704    0.8587    1.0000    0.8895
    0.7679    0.7857    0.6907    0.6312    0.7664    0.8895    1.0000
    0.8492    0.8392    0.7165    0.6448    0.8437    0.8407    0.8495
    0.6913    0.6577    0.5592    0.4778    0.6863    0.6485    0.6284
    0.5417    0.5257    0.7032    0.6266    0.5441    0.5676    0.6018
      ⋮
```

c. Finally, remove the elements from A that correspond to loop edges in the visual similarity graph.

The diagonal elements of an adjacency matrix are typically zero, but a nonzero diagonal element indicates a self-loop, or a node that is connected to itself by an edge. Calling *diag* twice returns a diagonal matrix composed of the diagonal elements of the original matrix A.

```
A = A - diag(diag(A)); % remove the loop edges by subtracting diagonal
elements of the original matrix A from matrix A
```

```
A = 1400×1400
         0    0.9585    0.8079    0.7192    0.9924    0.8633    0.7679 ⋯
    0.9585         0    0.7748    0.6916    0.9530    0.8842    0.7857
    0.8079    0.7748         0    0.8702    0.8082    0.7393    0.6907
    0.7192    0.6916    0.8702         0    0.7198    0.6704    0.6312
    0.9924    0.9530    0.8082    0.7198         0    0.8587    0.7664
    0.8633    0.8842    0.7393    0.6704    0.8587         0    0.8895
    0.7679    0.7857    0.6907    0.6312    0.7664    0.8895         0
    0.8492    0.8392    0.7165    0.6448    0.8437    0.8407    0.8495
    0.6913    0.6577    0.5592    0.4778    0.6863    0.6485    0.6284
    0.5417    0.5257    0.7032    0.6266    0.5441    0.5676    0.6018
      ⋮
```

d. Use A to create the hyperlink matrix H.

Since each row of the adjacency matrix corresponds to one image, we can form the required matrix of link-following likelihoods by normalising each row so that they sum to 1.

```
H = normalize(A, 2, 'norm', 1); % construct hyperlink matrix H by normalising
each row so that they sum to 1
```

```
H = 1400×1400
         0    0.0121    0.0102    0.0091    0.0125    0.0109    0.0097 ⋯
    0.0123         0    0.0100    0.0089    0.0123    0.0114    0.0101
    0.0081    0.0078         0    0.0087    0.0081    0.0074    0.0069
    0.0055    0.0053    0.0067         0    0.0055    0.0052    0.0049
    0.0126    0.0121    0.0103    0.0091         0    0.0109    0.0097
    0.0067    0.0069    0.0058    0.0052    0.0067         0    0.0069
    0.0050    0.0051    0.0045    0.0041    0.0050    0.0058         0
    0.0066    0.0065    0.0055    0.0050    0.0065    0.0065    0.0066
    0.0103    0.0098    0.0083    0.0071    0.0102    0.0096    0.0093
    0.0052    0.0051    0.0068    0.0061    0.0053    0.0055    0.0058
      ⋮
```

e. Form the random jump matrix J.

Adjacency matrix describes the connections between the nodes in the graph by the location of nonzero values. If node i and node j are connected, then A(i,j) or A(j,i) is nonzero; otherwise, A(i,j) and A(j,i) are zero.

H = ones(1400) is the adjacency matrix of a graph with 1400 nodes where each node is connected to all the others.

```
J = ones(size(H))/length(H); % create the jump matrix J
```

```
J = 1400×1400
```

```
  10^-3 ×
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143  ⋯
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143    0.7143
      ⋮
```

f. Given a damping factor of d = 0.85, create the modified visual hyperlink matrix Htilde.

It has become standard to use d=0.85, in which case there is a d = 0.85 =85% chance the user will select random images as usual and a $(1-d)$ = 0.15 =15% chance the user will get bored and jump to look for another images.

```
d = 0.85; % damping factor
Htilde = d*H + (1-d)*J; % modified visual hyperlink matrix Htilde
```

```
  Htilde = 1400×1400
    0.0001    0.0104    0.0088    0.0078    0.0108    0.0094    0.0083  ⋯
    0.0106    0.0001    0.0086    0.0077    0.0105    0.0098    0.0087
    0.0070    0.0067    0.0001    0.0075    0.0070    0.0064    0.0060
    0.0048    0.0046    0.0058    0.0001    0.0048    0.0045    0.0042
    0.0108    0.0104    0.0088    0.0079    0.0001    0.0094    0.0084
    0.0058    0.0060    0.0050    0.0045    0.0058    0.0001    0.0060
    0.0044    0.0045    0.0039    0.0036    0.0044    0.0051    0.0001
    0.0057    0.0056    0.0048    0.0043    0.0057    0.0056    0.0057
    0.0088    0.0084    0.0072    0.0061    0.0088    0.0083    0.0080
    0.0046    0.0044    0.0059    0.0053    0.0046    0.0048    0.0051
      ⋮
```

g. Find the VisualRank vector r using an initial vector creating the initial vector.

```
r = zeros(1,length(A)); % intialize ranking vector
r(1,1) = 1; % add 1 as a place to start
```

```
  r = 1×1400
       1     0     0     0     0     0     0     0     0     0     0     0
```

```
nmax = 50 % iterate 50 times
for n=1:nmax
        r = r * Htilde
end
```

```
  ...
  r = 1×1400
    0.2560    0.2540    0.2856    0.3318    0.2554    0.3342    0.3711  ⋯
```

h. Given this VisualRank vector, which image is ranked highest? Display this image.

```
[val,rank] = sort(r,'descend'); % rank and sort in descending order
```

```
val =
    0.0011    0.0011    0.0011    0.0011    0.0011    0.0011    0.0011
rank =
         713         714         719         701         704         705
```

```
highest_ranked_image = rank(1); % determine highest ranked image
```

```
highest_ranked_image = 713
```

```
filenames_h = filenames(highest_ranked_image,1); % extract filenames for the
corresponding highest ranked image
```

```
filenames_h = 1×1 cell array
    {'device9-20.png'}
```

```
C = convertCharsToStrings(filenames_h); % converts characters to strings
Figure; % plot the highest ranked image
img_1 = imread(sprintf("./MPEG7/%s", C));
imshow(img_1);
title(sprintf("The highest ranked image is %s", C));
```

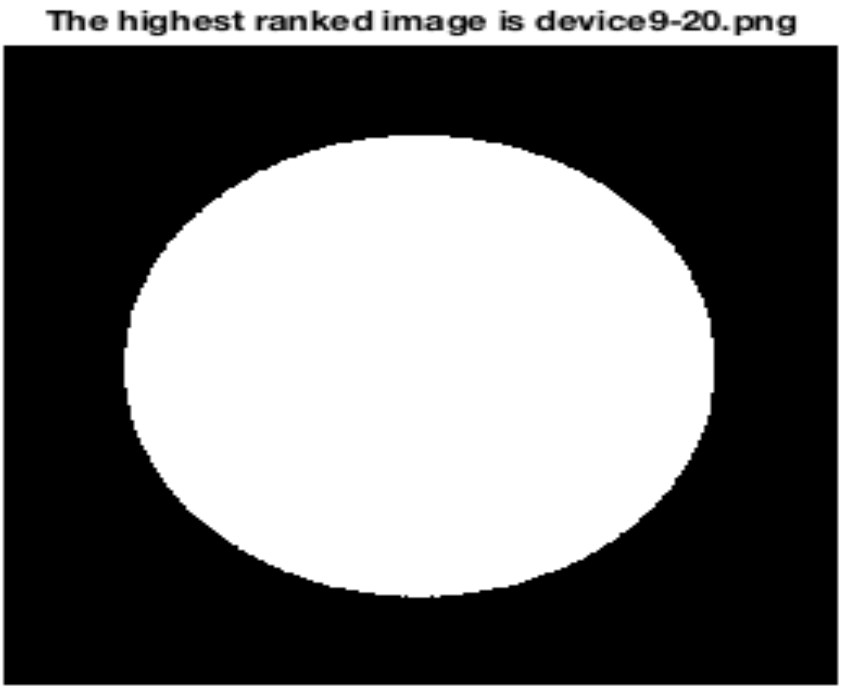

Fig 1: The highest ranked image according to Visualrank vecotor.

VisualRank ranks images in a group by how representative they are of the group as a whole. This is a difficult task when the group of images is large and varied, as it is in this case. The only common feature among all 1400 images is that the images tend to depict some white object toward the centre of the frame, as can be seen by the average of all the images:

Thus, if you have answered everything above correctly, you should have found a #1 image that looks very vaguely like this.

In practice, VisualRank is much more usefully applied to smaller groups of similar or related images, as is the focus of the following tasks.

2. Rank the 20 heart-shaped images (indices 81 through 100) using VisualRank.

a. Make a smaller 20x20 adjacency matrix by indexing the necessary rows andcolumns of the full adjacency matrix from above.

```
A = S; % deine adjacency matrix A
As = A(81:100,81:100); % create a smaller adjacency matrix As
```

```
As =
1.0000    0.8476    0.9673    0.8830    0.9497    0.9948    0.9257    0.8508
0.9506    0.8991    0.9327    0.8640    0.7887    0.9667    0.9471    0.7971
0.9597    0.9386    0.8701    0.8347    0.8476    1.0000    0.8164    0.7820
0.8006    0.8474    0.8417    0.7744    0.8729    0.7496    0.7991    0.7471
0.8273    0.8362    0.8188    0.6859    0.8140    0.8189    0.7970    0.6982
0.9673    0.8164    1.0000    0.8730    0.9175    0.9629    0.8926    0.8161
0.9182    0.9293    0.9002    0.8419    0.7567    0.9351    0.9161    0.7805
0.9273    0.9065    0.8386    0.8264
```

```
As(As<0) = 0; % remove negative values from the matrix A
As = As - diag(diag(As)); % remove the loop edges by subtracting diagonal
elements of the original matrix A from matrix A
```

b. Form the corresponding visual hyperlink matrix and find the VisualRank vector for all 20 heart images.

```
Hs = normalize(As, 2, 'norm', 1); % construct the corresponding hyperlink
matrix Hs
Js = ones(size(Hs))/length(Hs); % create the jump matrix Js
d = 0.85; % damping factor
Htildes = (d*Hs) + (1 - d)*Js; % create the modified hyperlink matrix Htildes
rs = zeros(1,20); % initialise ranking vector
rs(1,1)=1; % add 1 as a place to start
nmax = 5; % iterate 5 times
for n=1:nmax
        rs = rs * Htildes;
end
```

```
rs =
0.0533    0.0480    0.0521    0.0489    0.0514    0.0532    0.0505    0.0473
0.0515    0.0493    0.0511    0.0488    0.0453    0.0525    0.0516    0.0456
0.0522    0.0514    0.0489    0.0470
```

```
[~,rank] = sort(rs,'descend'); % rank and sort in descending order values
of VisualRank vector rs
```

```
val =
0.0533    0.0532    0.0525    0.0522    0.0521    0.0516    0.0515    0.0514
0.0514    0.0511    0.0505    0.0493    0.0489    0.0489    0.0488    0.0480
0.0473    0.0470    0.0456    0.0453
rank =
     1     6    14    17     3    15     9    18     5    11     7    10
4    19    12     2     8    20    16    13
```

```
filenames_s = filenames(81:100); % extract filenames for the corresponding
images
```

c. Given this ranking, which heart-shaped image is most representative of the group of heart-shaped images? Display this image.

```
most = rank(1); % most representative heart-shaped image according to
VisualRank
```

```
most = 1
```

```
disp('The most representative heart shaped image is: '), filenames_s(most,1);
% display this image
```

```
The most representative heart shaped image is:
ans = 1×1 cell array
    {'Heart-1.png'}
```

```
C1 = convertCharsToStrings(filenames_s); % convert characters to strings
figure; % plot the most representative image
img_2 = imread(sprintf("./MPEG7/%s", C1(most,1)));
```

```
imshow(img_2);
title(sprintf("The highest ranked image is %s", C1(most,1)));
```

Fig 2: The most representative heart-shaped image according to VisualRank vector.

d. Similarly, which heart-shaped image is the least heart-shaped (according toVisualRank)? Also display this image.

```
least = rank(20); % least heart-shaped image according to VisualRank;
```

```
least = 13
```

```
disp('The least heart shaped image is: '), filenames_s(least,1); % display
this image
```

```
The least heart shaped image is:
ans = 1×1 cell array
    {'Heart-20.png'}
```

```
figure; % plot the least heart-shaped image
img_3 = imread(sprintf("./MPEG7/%s", C1(least,1)));
imshow(img_3);
```

```
title(sprintf("The least heart shaped image is %s", C1(least,1)));
```

Fig 3: The least representative heart-shaped image according to VisualRank vector.

3. Pretend you are a search engine that has been queried for an image search of thefollowing pentagon-looking shape that is present in the dataset as 'device6-18.png'(at index 650):

MATLAB has a nearest function that finds the nearest nodes to a particular node on a graph. We will use this function to search for the images similar (near) to the image above and then refine these search results using VisualRank.

a. We first need to create a graph from the similarity adjacency matrix A created in Task 1. However, we cannot use this matrix directly, as it contains edge weights that are larger when images are more similar, corresponding to a further distance. We would instead like images that are similar to each other to be nearer in the graph. This can be achieved simply forming a new adjacency matrix for which the elements are the reciprocal of those in A. Do this, and, using the digraph function, form the graph G corresponding to this new adjacency matrix.

```
A = S; % define adjacency matrix A
A(A<0) = 0; % remove negative values from the matrix A
A = A - diag(diag(A)); % remove the loop edges by subtracting diagonal
elements of the original matrix A from matrix A
Arec = 1./A; % find reciprocal of A
```

```
Arec =

Inf       1.0433    1.2378    1.3904    1.0077    1.1583    1.3022    1.1775
1.4465    1.8461    1.6230    1.0451    1.5654    1.0155    1.0192    1.0270
1.0164    1.4507    1.5541    1.0367    6.9077    7.6259    4.8946    6.3065
5.5613    8.1230    9.9362   10.2551    5.9820    7.9313    1.0433
```

```
G = digraph(Arec, filenames); % Generate  a new adjacency matrix for which
the elements are  reciprocal of those in A by using digraph
```

```
G =
  digraph with properties:
Edges: [1960000×2 table]
Nodes: [1400×1 table]
```

```
G.Edges; % view the edge list of the graph
```

```
ans = 1960000×2 table
                  EndNodes                  Weight
_____________________________________    _______
{'Bone-1.png'}    {'Bone-1.png' }          Inf
{'Bone-1.png'}    {'Bone-10.png'}       1.0433
{'Bone-1.png'}    {'Bone-11.png'}       1.2378
{'Bone-1.png'}    {'Bone-12.png'}       1.3904
{'Bone-1.png'}    {'Bone-13.png'}       1.0077
{'Bone-1.png'}    {'Bone-14.png'}       1.1583
{'Bone-1.png'}    {'Bone-15.png'}       1.3022
{'Bone-1.png'}    {'Bone-16.png'}       1.1775
{'Bone-1.png'}    {'Bone-17.png'}       1.4465
{'Bone-1.png'}    {'Bone-18.png'}       1.8461
```

```
G = rmedge(G, 1:numnodes(G), 1:numnodes(G)); % remove all self-loops from the graph
```

```
G =
  digraph with properties:
Edges: [1958600×2 table]
Nodes: [1400×1 table]
```

b. Using MATLAB's nearest function, find the 10 images nearest to 'device6-18.png' in the graph G.

Index number of the 'device6-18.png' image is 650 in the filenames

```
[NODEISD, DIST]  = nearest(G,650,10); % determine which nodes are in a radius of 10 from node 650
```

```
NODEISD =
   645    654    657    647    653    649    644    643    655    656
DIST =
    1.0447    1.2318    1.2765    1.2847    1.2869    1.3032    1.3215
1.3622    1.3656    1.3747
```

```
Ten_nearest = NODEISD((1:10),1); % determine the ten nearest nodes
```

```
Ten_nearest =
   645    654    657    647    653    649    644    643    655    656
```

```
filenames_t = filenames(Ten_nearest); % extract filenames for the corresponding images
```

```
disp ('The 10 nearest images are: '),filenames_t; % display the filenames of the corresponding images
```

```
The 10 nearest images are:
```

```
filenames_t = 10×1 cell array
{'device6-13.png'}     {'device6-3.png' }     {'device6-6.png' }
{'device6-15.png'}     {'device6-20.png'}     {'device6-17.png'}
{'device6-12.png'}     {'device6-11.png'}     {'device6-4.png' }
{'device6-5.png' }
```

```
% assigning values to x and y axes to form a bar plot
x = categorical({'device6-13','device6-3', 'device6-6', 'device6-15',
'device6-20', 'device6-17', 'device6-12', 'device6-11', 'device6-4', 'device6-
5'});
x = reordercats(x,{'device6-13','device6-3', 'device6-6', 'device6-15',
'device6-20', 'device6-17', 'device6-12', 'device6-11', 'device6-4', 'device6-
5'});
y = Ten_nearest;
```

```
y =
   645    654    657    647    653    649    644    643    655    656
```

```
figure; % bar plot the ten nearest images
b = bar(x,y, 'm', 'EdgeColor', 'c', 'LineWidth', 1);
title('Ten Nearest Images');
```

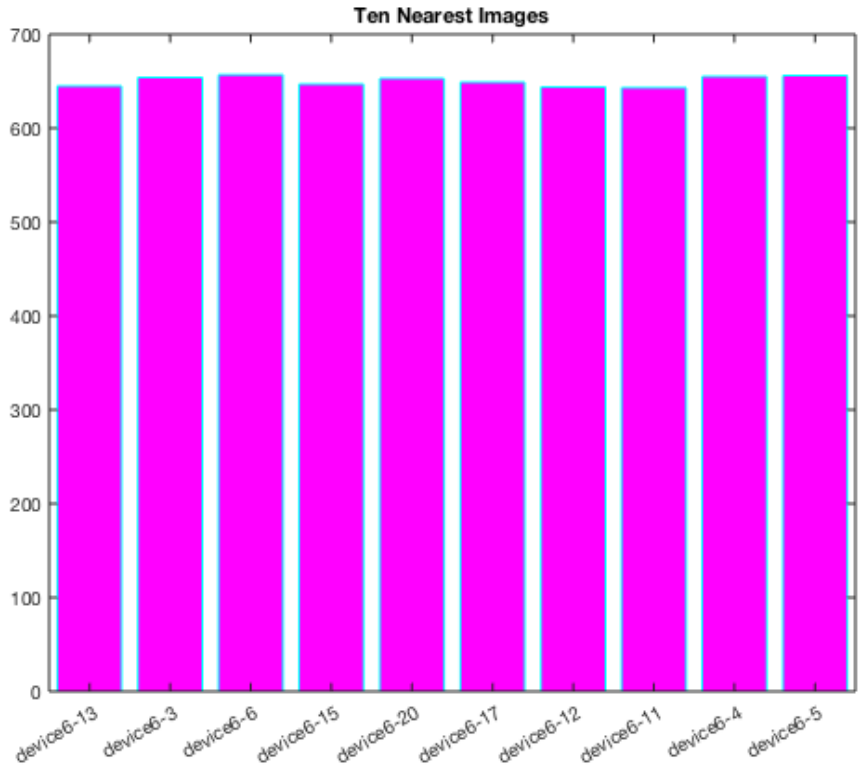

Fig 4: Bar plot of ten nearest images according to the nearest function in MatLab.

```
C3 = convertCharsToStrings(filenames_t);
figure; % plot the ten nearest images according to nearest function
for i = 1:10
subplot(5,2,i);
```

```
im_4 = imread(sprintf("./MPEG7/%s", C3(i, 1)));
imshow(im_4);
title(sprintf("Rank %d:  %s",i, C3(i,1)));
end
```

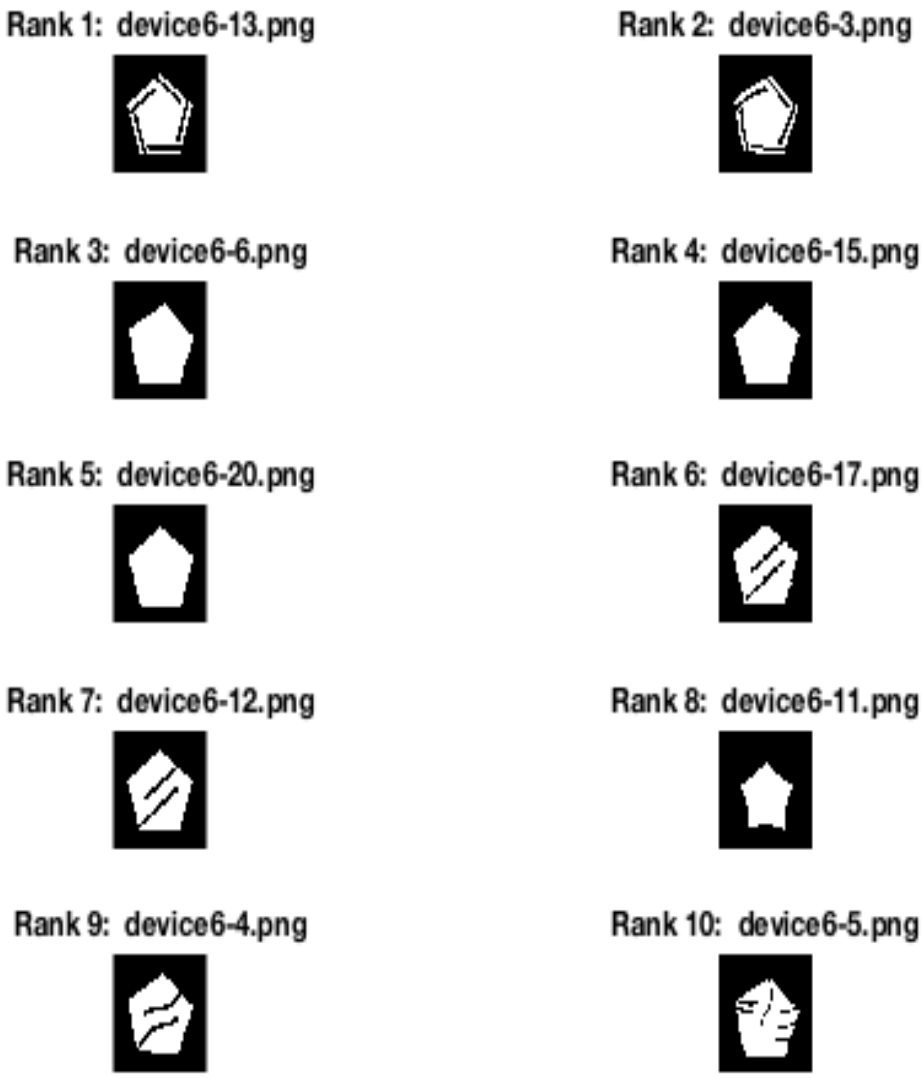

Fig 5: Rank of ten nearest images to image device6-18.png determined with the nearest function in MatLab.

c. Finally, rank these 10 nearest images using VisualRank and display them on a figure in their ranked order. This is the final search result.

```
A = S; % define matrix A
At = A([643 644 645 647 649 653 654 655 656 657],[643 644 645 647 649
653 654 655 656 657]); % create a small adjacency matrix for those 10
images
```

```
 At =
 1.0000    0.6203    0.7309    0.7280    0.6187    0.7274    0.7223
 0.6768    0.7279     0.6203   1.0000    0.7697    0.8299    0.9710
 0.7257    0.8128    0.7607    0.8297     0.7309    0.7697   1.0000
 0.7566    0.7756    0.8149    0.7348    0.7255    0.7821    0.7280
 0.7752    1.0000    0.8314    0.9859    0.7625    0.8223    0.8852
 0.6187    0.9710    0.7566    0.8314    1.0000    0.8299    0.7225
 0.7625    0.8307    0.7274    0.8287    0.7756    0.9859    0.8299
 0.7626    0.8221    0.8874    0.9545
```

```
At(At<0) = 0; % remove negative values from the matrix At
At = At - diag(diag(At)); % remove the loop edges by subtracting diagonal
elements of the original matrix At from matrix At
Ht = normalize(At, 2, 'norm', 1); % construct the corresponding hyperlink
matrix Ht
Jt = ones(size (Ht))/length(Ht); % construct the jump matrix Jt
d = 0.85 % dumping factor
Htildet = d*Ht + (1-d)*Jt; % construct modified hyperlink matrix Htildet
rt = zeros(1,10); % initialise ranking vector
rt(1,1) = 1; % add 1 as a place to start
nmax = 5; % iterate 5 times
for n=1:nmax
        rt = rt * Htildet;
end
```

```
 rt =
     0.0888    0.1006    0.0972    0.1058    0.1004    0.1058    0.0962
 0.0987    0.1002    0.1063
```

```
 [val,rank] = sort(rt,'descend'); %  rank and sort in descending order values
of VisualRank vector rt
```

```
 val =
     0.1063    0.1058    0.1058    0.1006    0.1004    0.1002    0.0987
 0.0972    0.0962    0.0888
 rank =
     10     6     4     2     5     9     8     3     7     1
```

```
filanames_t = filenames([643 644 645 647 649 653 654 655 656 657],1); %
extract the filenames for the corresponding images
Top_ten = filanames_t (rank,1); % ranked filenames according to the ranked
images
```

```
 Top_ten = 10×1 cell array
```

```
{'device6-6.png' }     {'device6-20.png'}     {'device6-15.png'}
{'device6-12.png'}     {'device6-17.png'}     {'device6-5.png' }
{'device6-4.png' }     {'device6-13.png'}     {'device6-3.png' }
{'device6-11.png'}
```

```
C4 = convertCharsToStrings(Top_ten); % convert character to string
figure % plot the ranked 10 images according to VisualRank function
for i = 1:10
   subplot(5,2,i);
   Img_5 = imread(sprintf("./MPEG7/%s", C4(i,1)));
   imshow(Img_5);
   title(sprintf("Rank %d: %s",i ,C4(i,1)));
end
```

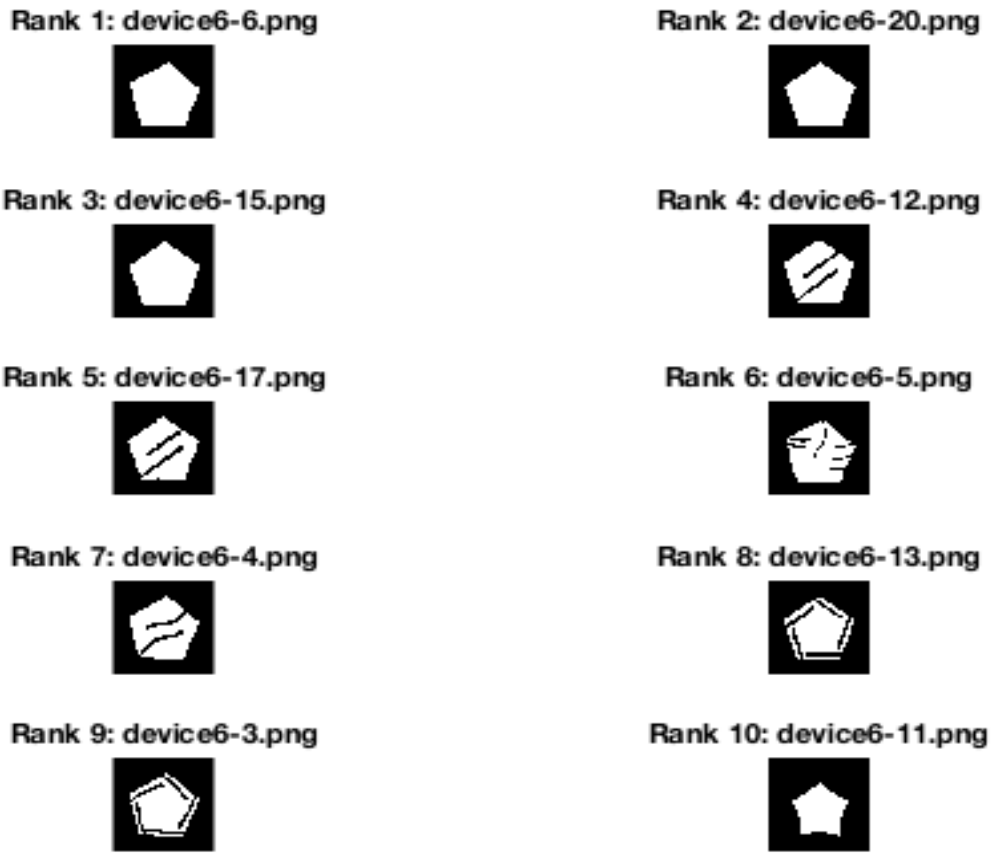

Fig 6: Visual presentation of ten ranked images according by VisualRank vector.

```
 % assigning values to x and y axes to form a bar plot
 x = categorical({'device6-6','device6-20', 'device6-12', 'device6-15',
'device6-17', 'device6-5', 'device6-4', 'device6-13', 'device6-3', 'device6-
11'});
 x = reordercats(x,{'device6-6','device6-20', 'device6-12', 'device6-15',
'device6-17', 'device6-5', 'device6-4', 'device6-13', 'device6-3', 'device6-
11'}); % preserve the order of of categories
 y = val(1:10) % assign magnitude of rank for each element of vector rt to y-
 axis
 figure % bar plot the ten images according to VisualRank
 b = bar(x,y, 'm', 'EdgeColor', 'c', 'LineWidth', 1);
 title('VisualRank of Ten Images');
```

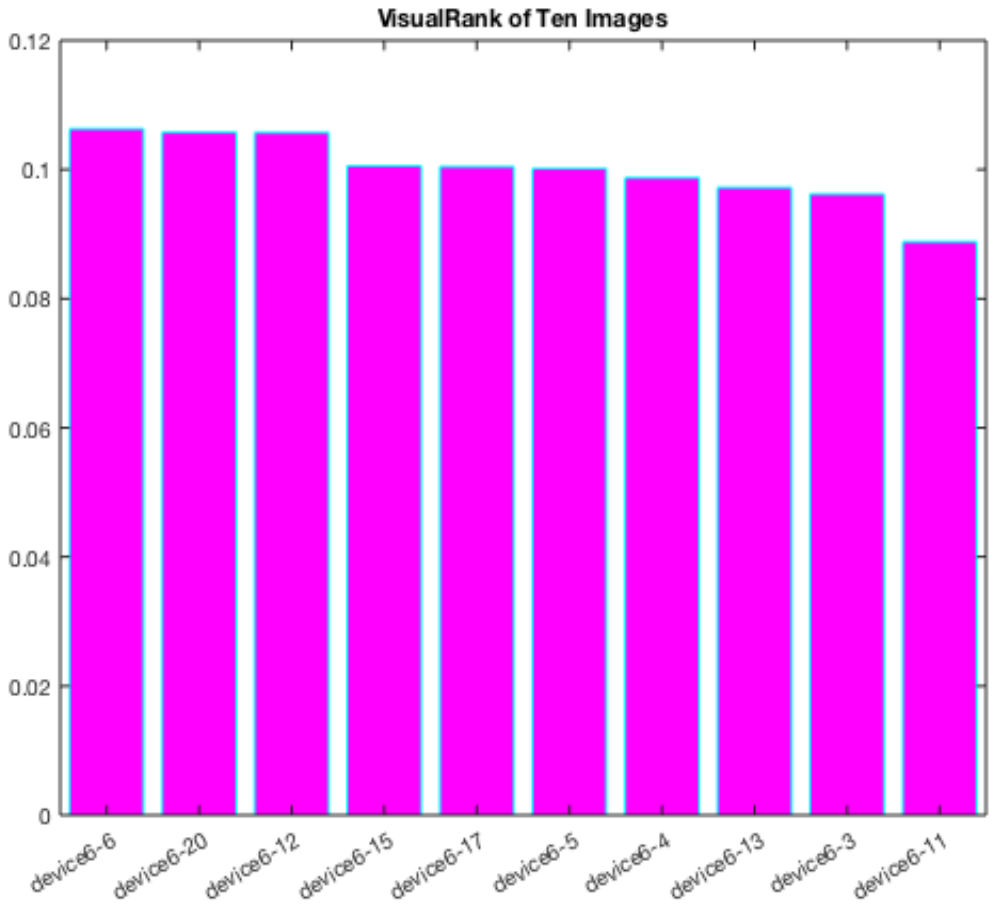

Fig 7: Bar plot shows the magnitude of the elements of VisualRank vector corresponding to the ten images in descending order.

Optional: VisualRank function

The VisualRank function is written in separate file visualrank.m. Task 2 and 3c can be performed through this function.

2. Rank the 20 heart-shaped images (indices 81 through 100) using VisualRank.
a. Make a smaller 20x20 adjacency matrix by indexing the necessary rows andcolumns of the full adjacency matrix from above.
b. Form the corresponding visual hyperlink matrix and find the VisualRank vector for all 20 heart images.

```
load sim.mat; % load the matrix into MatLab environment
A = S; % define adjacency matrix A
As = A(81:100,81:100); % create a smaller adjacency matrix As
Heart = visualrank(As); % apply visualrank function to all of the 20 heart
images
filenames_s = filenames(81:100); % extract filenames for the corresponding
images
 [val2,rank2] = sort(Heart,'descend'); % sort and rank elements of r vector in
descending order
```

c. Given this ranking, which heart-shaped image is most representative of the group of heart-shaped images? Display this image.

```
most = rank2(1); % most representative heart-shaped image according to
VisualRank function
disp('The most representative heart shaped image is: '), filenames_s(most,1);
% display this image
C1 = convertCharsToStrings(filenames_s); % convert characters to strings
figure; % plot the most representative image
img_6 = imread(sprintf("./MPEG7/%s", C1(most,1)));
imshow(img_6);
title(sprintf("The highest ranked image is %s", C1(most,1)));
```

d. Similarly, which heart-shaped image is the least heart-shaped (according toVisualRank)? Also display this image.

```
least = rank2(20); % least heart-shaped image according to VisualRank
disp('The least heart shaped image is: '), filenames_s(least,1); % display this
image
figure; % plot the least heart-shaped image
img_7 = imread(sprintf("./MPEG7/%s", C1(least,1)));
imshow(img_3);
title(sprintf("The least heart shaped image is %s", C1(least,1)));
```

3 c. Finally, rank these 10 nearest images using VisualRank and display them on a figure in their ranked order. This is the final search result.

```
load sim.mat; %load the matrix into MatLab environment
A = S; % define adjacency matrix A
At = A([643 644 645 647 649 653 654 655 656 657],[643 644 645 647 649 653 654
655 656 657]); % create a small adjacency matrix for those 10 images
Device = visualrank(At); % apply visualrank function to all 10  images
filanames_t = filenames([643 644 645 647 649 653 654 655 656 657],1); % extract
the filenames for the corresponding images
[val3,rank3] = sort(Device,'descend'); % sort and rank elements of r vector in
descending order
Ten_img = filanames_t (rank3,1); % ranked filenames according to the ranked
images
C6 = convertCharsToStrings(Ten_img); % convert character to string


Figure; % plot the ranked 10 images according to VisualRank function
for i = 1:10
   subplot(5,2,i);
   Img_8 = imread(sprintf("./MPEG7/%s", C6(i,1)));
   imshow(Img_8);
   title(sprintf("Rank %d: %s",i ,C6(i,1)));
end
```

```
% assigning values to x and y axes

x = categorical({'device6-6','device6-20', 'device6-12', 'device6-15', 'device6-
17', 'device6-5', 'device6-4', 'device6-13', 'device6-3', 'device6-11'});

x = reordercats(x,{'device6-6','device6-20', 'device6-12', 'device6-15',
'device6-17', 'device6-5', 'device6-4', 'device6-13', 'device6-3', 'device6-
11'}); % preserve the order of of categories
y = val3(1:10); % assign magnitude of rank for each element of vector rt to y-
axis

figure; % bar plot the ten images according to VisualRank function
b = bar(x,y, 'm', 'EdgeColor', 'c', 'LineWidth', 1);
title('VisualRank of Ten Images');
```