ENGGEN 131

MATLAB PROJECT

```
function [frames] = GenerateFrameList(F0,SS,n)
%GenerateFrameList generates a list of frames we
are interested in which
%can be used by other functions. In particular it
will be useful for
%determining which frames to extract from a movie
file.
% Inputs: F0 = Frames-naut, which is the starting
frame number
%         SS = Step size
%         n = The number of frames to generate
% Outputs: frames = a 1 x 'n' 1D array, where n is
the desired number of frames (n).
%                   The first element of the array
will be the starting frame
%                   number and each subsequent
element will have a frame
%                   value that is the step size
greater than the last.
% Author: Sooyong Kim

%The first value will always be F0
frames(1)=F0;

%Make a for loop to add the subsequent elements
into the array
for i=1:(n-1)
    frames(i+1)=frames(i)+SS;
end
end
```

```
function [ImageFiles] =
GenerateImageList(ImageDirectory,FileExtension)
%The GenerateImageList function retrieves a list of the names
of all the
%images with a particular file extension that arecontained in
a specified
%directory
% Inputs: ImageDirectory = A string containing the name of the
directory the
%                          images are contained in
%         FileExtension = A string containing the file
extension of the
%                         images to fetch
% Output: ImageFiles = A 1xn 1D cell array containing n
strings where each
%                      element is the filename of an image from
the specified
%                      directory that has a particular file
extension
% Author: Sooyong Kim

%Obtain the structure array using the dir function
FileList = dir (ImageDirectory);

%Extract the file names from the FileList structure array
FileNames = {FileList.name};

%Now we want to compare each element in the cell whether they
end in the
%file extension we want. If they do match, then this loop adds
that element
%into the allocated ImageFiles cell array.

%Create a counting index to store the matching file names into
our
%ImageFile cell array
i=1;
for j=1:length(FileNames)
    if endsWith(FileNames{j}, FileExtension)==1
        ImageFiles{i} = FileNames{j};
        i=i+1;
    end
end

end
```

```
function [DistanceSQ] =
PixelDistance(ThreeDPoint1,ThreeDPoint2)
%The PixelDistance function calculates the square
of the distance between
%two points in colour space.
% Inputs: ThreeDPoint1=An array containing three
elements representing a
%                       point in 3D colour space
%         ThreeDPoint2=An array containing three
elements representing a
%                       second point in 3D colour
space
% Outputs: DistanceSQ=The square of the distance
between the two points in
%                     3D colour space.
% Author: Sooyong Kim

%If values are uint8 integers, double them.
ThreeDPoint1=double(ThreeDPoint1);
ThreeDPoint2=double(ThreeDPoint2);

%Element by Element operations:
DistanceSQ=sum((ThreeDPoint1-ThreeDPoint2).^2);

end
```

```
function [MedRed,MedGreen,MedBlue] =
MedianPixel(RGB)
```

```
%The MedianPixel function calculates the median
RGB values from a list of pixels.
% Input: RGB=A 1xnx3 3D array of RGB values
representing a list of pixels
% Output: MedRed=The median red value, which will
be the median of all the
%                R values from the list of pixels
%         MedGreen=The median green value, which
will be the median of all
%                  the G values from the list of
pixels
%         MedBlue=The median blue value, which
will be the median of all
%                 the B values from the list of
pixels
% Author: Sooyong Kim

%Calculate the median for each row in the RGB
array, and round it
Medians = round(median(RGB,2));

%Assign the values to its respective outputs
MedRed=Medians(1);
MedGreen=Medians(2);
MedBlue=Medians(3);

end
```

```
function [Medians] = ModifiedMedianPixel(RGB)
%The ModifiedMedianPixel function calculates the
median RGB values from a list of pixels.
% Input: RGB=A nxnx3 3D array of RGB values
representing a list of pixels
% Output: Medians = An array of the medians in
each row of the RGB values
% Author: Sooyong Kim

%Calculate the median for each row in the RGB
array, and round it
Medians = round(median(RGB,2));
end
```

```
function [DistRed,DistGreen,DistBlue] =
MostDistantPixel(RGB)
%The MostDistantPixel function calculates the
pixel from a list that is
%most distant from the median RGB values of that
list. The distance metric
%to be used is that described in the PixelDistance
function
% Inputs: RGB = A 1xnx3 3D array of RGB values
representing a list of pixels
% Outputs: DistRed = The red value of the most
distant pixel
%          DistGreen = The green value of the most
distant pixel
%          Dist Blue = The blue value of the most
distant pixel
% Author: Sooyong Kim

%Call the median of each colour using the
MedianPixel function, and store
%them in an array
[MedRed,MedGreen,MedBlue] = MedianPixel(RGB);
Medians = [MedRed,MedGreen,MedBlue];

[rows,cols,col] = size (RGB);

%Obtain the RGB points of the given RGB array
using a for loop and
%store them in a cell array

for i=1:cols
    for j=1:3
        Point(j)=RGB(1,i,j);
    end
    Points{i}=Point;

end

%This calculates the distance between each point
and the median, and stores
%them into the Distanes array.
for i=1:cols
```

```
    [DistanceSQ] =
PixelDistance(Medians,Points{i});
    Distances(i) = DistanceSQ;
end

MaxDistance=max(Distances);

%Locate  the position of the maximum distance in
the array, where 'k' is
%the position of the MaxValue in the array
k=find(Distances==MaxDistance);

%The most distant point is the 'k' element in the
array
MostDistantPoint = Points{k};

%Assign them into their respective outputs
DistRed = MostDistantPoint(1);
DistGreen = MostDistantPoint(2);
DistBlue = MostDistantPoint(3);
```

```
function [pictures] =
ReadImages(ImageDirectory,FileArray)
```

```
%The ReadImages function reads a specified list of
images given the
%filenames and the directory the files are located
in.
% Input: ImageDirectory = A string containing the
name of the directory the
%                         images are contained in
%        FileArray = A 1xn 1D cell array
containing n strings where each
%                    element is a filename of an
image to read
% Output: pictures = 1xn 1D cell array containing
n images, where each
%                    element is an RGB image.
% Author: Sooyong Kim

%Locate and change the directory, while saving the
original working space
oldFolder = cd(ImageDirectory);
cd;

%Allocate an array to store the images, where the
image is an RGB image
pictures={};

%Create a for loop to process the images in the
FileArray, and store them
%in the created pictures array
for i=1:length(FileArray)
    pictures{i}=imread(FileArray{i});
end

%Return to the original working space
cd (oldFolder);
cd;
end
```

```
function [RemovedImage] = RemoveAction(images)
%The RemoveAction function creates an image that has the
action removed by
%applying a median filter
% Inputs: images = A 1xn 1D cell array containing n images,
where each
%                 element is an RGB image
% Outputs: RemovedImage = An RGB image that was obtained by
taking the median
%                         RGB values of the stack of
corresponding pixels from
%                         the source images.
% Author: Sooyong Kim

%We must process each pixel individually and find the median

%Obtain the dimensions of the image using one of the images as
a sample
[rows,cols,col]=size(images{1});

%Allocate arrays for the action image (rows,cols,col) in uint8
integers;
%and for the RGB pixels to be used to obtain the median pixel.
RGB = zeros(rows,length(images),col);
RemovedImage = zeros(rows,cols,col, 'uint8');

%This for loop processes each pixel of the images in the same
position and
%stores them in a nxnx3 RGB array, then calls the
ModifiedMedianPixel function to
%calculate the median of the pixels, then stores the outputs
into the
%RemovedImage array.
for i=1:cols
    for j=1:rows
        for k=1:col
            for l=1:length(images)
                RGB(j,l,k)=images{l}(j,i,k);
            end
        end
    end
    [Medians] = ModifiedMedianPixel(RGB);
    RemovedImage((1:rows),i,(1:col)) = Medians;
end

end
```

```
function [ActionImage] = ActionShot(images)
%The ActionShot function creates an action shot image by
finding the pixels
%from a stack of images that are most distant from the
median RGB values
% Input: image = A 1xn 1D cell array containing n images,
where each element
%                is an RGB image
% Output: ActionImage = An action shot in the form of an
RGB image
% Author: Sooyong Kim

%Obtain the dimensions of the image using one of the
images as a sample
[rows,cols,col]=size(images{1});

%Allocate arrays for the action image (rows,cols,col) in
uint8 integers; and
%for the RGB pixels to be used to obtain the most distant
pixel.
RGB = zeros(1,length(images),3);
ActionImage = zeros(rows,cols,col, 'uint8');

%This for loop processes each pixel of the images in the
same position and
%stores them in a 1xnx3 RGB array, then calls the
MostDistantPixel function
%to calculate the most distant pixel, then stores the
outputs into the ActionImage array.
for i=1:rows
    for j=1:cols
        for k=1:col
            for l=1:length(images)
                RGB(1,l,k)=images{l}(i,j,k);
            end
        end
        [DistRed,DistGreen,DistBlue] =
MostDistantPixel(RGB);
        ActionImage(i,j,1) = DistRed;
        ActionImage(i,j,2) = DistGreen;
        ActionImage(i,j,3) = DistBlue;
    end
end

end
```