Projekt 2

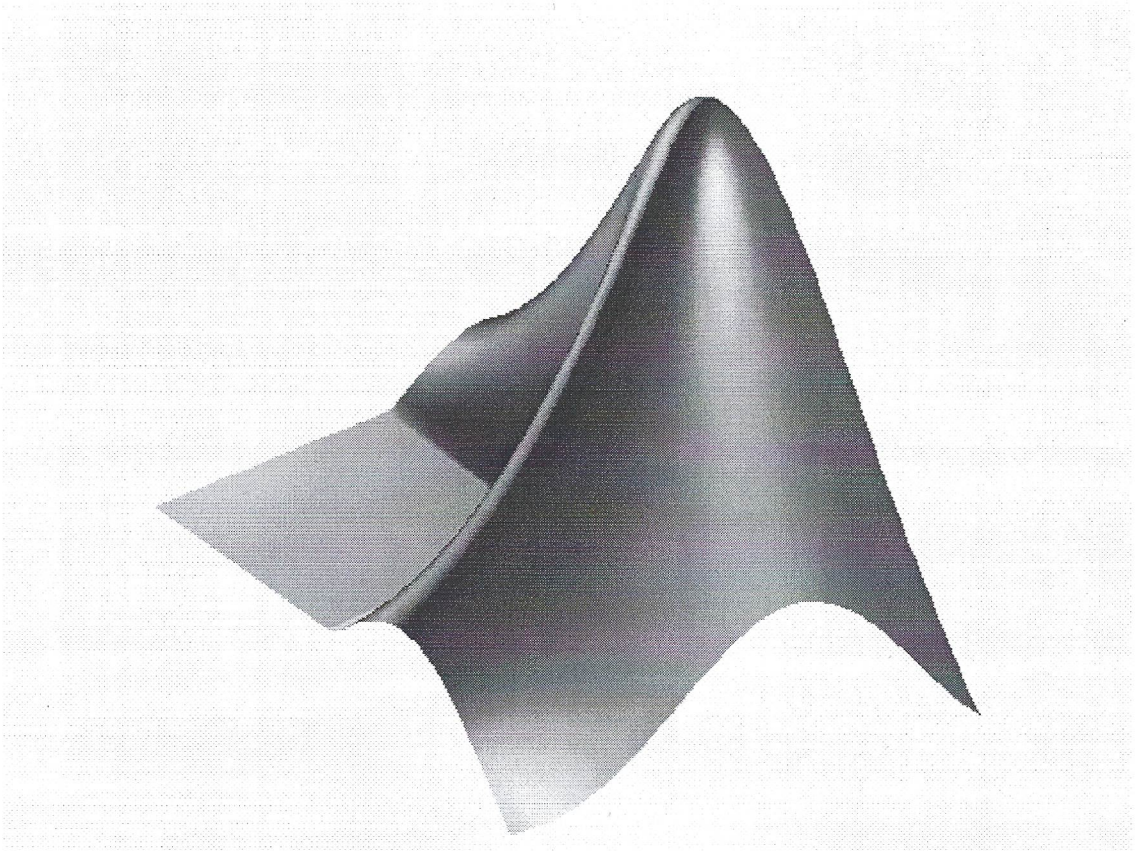
Digital Billed Manipulerings Software (DIMS)

Udført af Gustav Collin Rasmussen s053849

Og Tony B. Rungling s083422

Gruppe 40

Projektet er udført i Bygning 306, databar 30



Indledning

I dette projekt ønsker vi at modificere nogle billeder på otte forskellige måder;

Vi vil for de givne billeder kunne konvertere dem til deres gråtone-versioner,

Negativ-versioner, tilføje billederne warp/fiskeøje-effekt, lægge en ramme omkring,

Spejle billedets ene halvdel over på den anden, lægge fængsels-tremmer over billedet, omdanne det til en superpixeleret udgave, og lægge et skakbræt-mønster over billedet.

Dette gøres med de otte funktioner grayscale, negative, warp, frame, mirror, jail, superpixel og chess.

Beskrivelse af algoritmerne i dims-scriptet til implementering af kommandoerne newX og showGal

Så længe vi ikke modtager kommandoen "sto" for "stop af programmet", vil vi give brugeren mulighed for at indtaste billeder, manipulere dem, gemme dem i et galleri, vælge nye billeder, eller vise galleriet.

Dette gøres med en while-løkke. Inden i denne er vores to algoritmer for newX og showGal.

Hvis brugeren ønsker at se galleriet, taster han "sho", eller hvis han vil skifte billede taster han "new"

Listning af dims.m:

```
function dims()
    % Scriptet DIMS is a driver script for some Digital Image Operations.
    % Hence, it constitutes the interface with the user.

    % Input:      billede, X
    % Output:     depending upon which of the eight imagemanipulating-functions
    %             is chosen, eight different modifications  is given for the
    %             chosen picture

    % Authors:    Gustav Collin Rasmussen og Tony B. Rungling
    % Date:       15/1-2010

    function saveToGal(dim, image)
       IntoGallery = input('Store this image in the gallery (Y/N)?: ','s');

       if strncmpi(IntoGallery,'Y',1)    % IntoGallery starts with Y or y
           name = input('Enter a title for the image: ','s');
           ImagesInGal = ImagesInGal +1;

           gallery(ImagesInGal).dim = dim;
           gallery(ImagesInGal).image = image;
           gallery(ImagesInGal).name = name;
       end
    end
```

```
function loadImage()
    imagefile = input('Enter a file name containing an image: ','s');
    if ~exist(imagefile,'file')
        error('The file is not found! Goodbye!');
    end;

    % If the file is found, the image is read into the array X.
    % We let dimX(3) denote the depth of X
    % (dimX(3) is 1 in case of grayscale),
    % display the image and store it (if wanted) in a gallery
    % of images.
    X = imread(imagefile); dimX = size(X);
    if numel(dimX)==2, dimX = [dimX 1]; end;

    xwindow = figure('Name','The Current Image X');
    image(X), axis image off, colormap gray(256);
end

clear all;     % Clears the workspace
close all;     % Closes all open figure windows

% Initialize gallery
ImagesInGal = 0;
gallery = struct('dim', 0, 'image', 0, 'name', 0);

% Load image from file Obtains a file name
loadImage();
saveToGal(dimX, X);

command = 'a string allowing us to enter the following loop';

% we loop until we receive the command stop possibly shortened or with
% capital letters
while ~strncmpi(command,'sto',3);
    % What does the user want to do?
    command = input('Enter command (newX,showGal,operate,stop): ','s');

    % Hold the user's choice
    newX = false;
    operation = false;
    galleryshown = false;

    if strncmpi(command,'ope',3)
        command = [input(['Enter operation(gra,neg,war,...'
            'fra,mir,jai,sup,che): '],'s') '   '];
        % makes sure that command has 3 chars
        switch lower(command(1:3))
            case {'gra','neg','war','fra','mir','jai','sup','che'}
                operation = true;     % command = operation
            otherwise
                operation = false;    % command = no operation
        end;
        % If the user wants to see the gallery
    elseif strncmpi(command,'sho',3)
        galleryshown = true;
        % If the user wants to exchange his image for another one
    elseif strncmpi(command,'new',3);
```

```
            newX = true;
        end;

% First 3 letters i the command:
% gra, neg, war, fra, mir, jai, sup, che, new, sho or sto.
% Functions are carried out when the user calls them. If needed, a
% parameter is called too.
        if operation
            if strncmpi(command,'gra',3)
                Y = grayscale(X, dimX(3));
            elseif strncmpi(command,'neg',3)
                Y = negative(X);
            elseif strncmpi(command,'war',3)
                % p determines the amount of warp effect, p = 0 gives zero
                % warp.
                p = input('type in a real number p : ');
                Y = warp(X,dimX,p);
            elseif strncmp(command,'fra',3)
                % w gives the width of the frame to put on the image.
                w = input(['type in an integer w so that 0 <= w <= '...
                     num2str(min(dimX(1:2))) ' : ']);
                Y = frame(X,dimX,w,[0 0 255]);
            elseif strncmp(command,'mir',3)
                % lr determines whether to return the mirror-image of the
                % right or the left side of the selected picture
                lr = input('type in an integer lr so that lr = 0 or 1 : ');
                Y = mirror(X,dimX(2),lr);
            elseif strncmp(command,'jai',3)
                % p determines the width of the prison-bars
                p = input('type in a positive real number p : ');
                Y = jail(X,dimX,p);
            elseif strncmp(command,'sup',3)
                % p determines the degree of clarity for the image
                p = input('type in a positive real number p : ');
                Y = superpixel(X,dimX,p);
            elseif strncmp(command,'che',3)
                % p determines the size of the square-fields of the
                % chessboard-pattern to put ontop of the selected image
                p = input('type in a positive real number p : ');
                Y = chess(X,dimX,p);
            end

            % Present the resulting image to the user
            ywindow = figure('Name','The Result of the Operation on X');
            image(Y), axis image off, colormap gray(256)

            % Find dimensions of resulting image
            dimY = size(Y);
            if numel(dimY)==2, dimY = [dimY 1]; end;

            % Save the image to the gallery - if the user wants to
            saveToGal(dimY, Y);

            close(ywindow);
            figure(xwindow);
        elseif galleryshown
            % we loop over the amount of images
            for i = 1 : ImagesInGal
```

```
            subplot(ceil(ImagesInGal / min([ImagesInGal 4])),...
                min([ImagesInGal 4]), i);
            image(gallery(i).image), axis image off, colormap gray(256);
            title(gallery(i).name);
        end
        % the user selects a new image
    elseif newX
        loadFrom = input(['Vil du loade et billede fra en fil,...'
        'skriv da fil, eller fra galleriet, skriv gal:', 's']);
        % if the user selects an image from a file
        if strncmp(loadFrom, 'fil', 3)
            loadImage();
            saveToGal(dimX, X);
            % if the user selects an image from the gallery
        elseif strncmp(loadFrom, 'gal', 3)
            for i = 1 : ImagesInGal
                disp([num2str(i) '. ' gallery(i).name]);
            end

            i = input('Hvilket af ovenstående ønsker du at loade:');

            X = gallery(i).image;
            dimX = gallery(i).dim;
        end
    end
end

    close(xwindow);
end
```

Listning af de otte MATLAB-funktioner:

Listning af grayscale.m:

```
function [ Y ] = grayscale( X, d )
% GRAYSCALE computes the grayscale image corresponding to a given image
%
% Call:    Y   = grayscale( X, d )
% Input Parameters: X is a 2D or 3D array of type uint8 containing an image
%                   d is the 'depth' of X (1 if grayscale, 3 if color)
% Output Parameter: Y is an array of same dimensions and type as X
%                   containing the grayscale image corresponding to X

% Author:  Jørgen B. Sand
% Date:    July 13, 2009

if d >= 3
    Y=uint8(.2989*double(X(:,:,1))+.5870*double(X(:,:,2))+ ...
        .1140*double(X(:,:,3)));
else Y = X;
end;
```

listning af negative.m:

```
function  Y = negative(X)
% NEGATIVE computes the negative of a given image
%
% Call:    Y = negative(X)
% Input Parameter: X is a 2D or 3D array of type uint8 containing an image
% Ouput Parameter: Y is also such an array, containing the negative of X

% Authors:  Gustav Collin Rasmussen og Tony B. Rungling
% Date:     15/1-2010

%*** compute the negative image of X
Y = 255 - X;
```

Listning af warp.m:

```
function Y=warp(X, dimX, p)
% Applies warp effect to the input image
%
% Call: Y=warp(X,p);
% Input Parameters: X is an 2D array of type uint8 containing a grayscale
%                   image.
%                   p is a positive variable which controls the degree of
%                   the effect.

% Ouput Parameter:  Y is an 2D array of type uint8 containg X with an
%                   applied warp effect.

% Authors: Gustav Collin Rasmussen & Tony B. Rungling
% Date:    15/1-2010

% Algorithm by Troels K. Jacobsen

% We extract the dimensions of the image
h = dimX(1); w = dimX(2); d = dimX(3);

% Compute the x-values for the warp
x = ones(1,w);
for i = 2:w
   x(i) = x(i-1) + min([i-1,w+1-i]).^p;
end
x = round((x-1)*(w-1)/(x(end)-1))+1;
x = ((x-1)*(w-1)/(x(end)-1))+1;

% Compute the y-values for the warp
y = ones(1,h);
for i = 2:h
   y(i) = y(i-1)+min([i-1,h+1-i]).^p;
end
y = round((y-1)*(h-1)/(y(end)-1))+1;
y = ((y-1)*(h-1)/(y(end)-1))+1;

% Apply the warp to the (d layers of) X
[XX,YY] = meshgrid(x,y);
```

```
for k=1:d
   Y(:,:,k) = uint8(round(interp2(double(X(:,:,k)),XX,YY)));
end
```

listning af frame.m:

```
function Y=frame(X, dimX, width, colors)
% Applies a frame to the input image.
%
% Call: Y=frame(X,w);
% Input Parameters: X is an 2D array of type uint8 containing an image.
%                   w is the width of the frame in pixels

% Ouput Parameter:  Y is an 2D array of type uint8 containg X with an
%                   applied frame width and the colour colors.

% Authors: Gustav Collin Rasmussen og Tony B. Rungling
% Date:     15/1-2010

% Algorithm

Frame = zeros(dimX(1), dimX(2), dimX(3));

for i = 1 : dimX(3)
    Frame(:,:,i) = colors(i);
end

Frame(width + 1:dimX(1) - width,width + 1:dimX(2) - width,1:3) = 0;
% Creates a matrix Frame
Picture=zeros(dimX(1),dimX(2),dimX(3));
Picture(width + 1:dimX(1) - width,width + 1:dimX(2) - width,1:3) = 1;

if dimX(3) == 1
    % Frame is changed to uint8 and multiplied elementwise
    Y=uint8(Frame(:,:,1)) + uint8(Picture(:,:,1)) .* X;
else
    % Frame is changed to uint8 and multiplied elementwise with X
    Y=uint8(Frame) + uint8(Picture) .* X;
end
```

listning af mirror.m:

```
function Y = mirror(X, w, lr)
% Applies the mirror effect to the input image.

% Call: Y=mirror(X, w, lr);
% Input Parameters: X is an 2D array of type uint8 containing an image.
%                   w is the width
%                   lr is a logical value that specifies which way to go,
%                   left or right.

% Ouput Parameter:  Y is an 2D array of type uint8 containg X with an image
%                   where left and right half of X are replaced with a
%                   mirror image of one of the halves.

% Authors: Gustav Collin Rasmussen & Tony B. Rungling
% Date:    15/1-2010

% Algorithm

Y=X;

if lr==0
    Y(:,1:floor(w/2),:)=X(:,w:-1:w-floor(w/2)+1,:);
else
    Y(:,w:-1:w-floor(w/2)+1,:)=X(:,1:floor(w/2),:);
end

% When lr==0 the right side is mirrored into the left side.
% When lr==1 the left side is mirrored into the right side.
% If the number of columns is odd, the the middle column not changed.
```

Listning af jail.m:

```
function Y = jail(X, dimX, p)
% Applies jail (vertical lines) effect to the input image.

% Call: Y = jail(X, dimX, p);
% Input Parameters: X is an 2D array of type uint8 containing an image.
%                   dimX is the dimensions of the picture
%                   p is a positive variable which controls the distance
%                   between the vertical lines (Bigger p, longer distance).

% Ouput Parameter:  Y is an 2D array of type uint8 containg the original
%                   image with added vertical lines.

% Authors: Gustav Collin Rasmussen & Tony B. Rungling
% Date     15/1-2010
% Algorithm
%[row column]=size(X); %Stores the size of the array X in the variables row
%                      and column.

% Creates the array M.
```

```
M = ones(dimX(1),1) * sin(pi*(1:dimX(2))/p).^2;
% When the image is a black-and-white photo
if dimX(3) == 1
    X = uint8(M.*double(X));
else
    % When the image is a 3-dimensional color-photo, we enter a for-loop
    for i = 1:dimX(3)
       X(:,:,i) = uint8(M.*double(X(:,:,i)));
    end
end

Y = X;
```

Listning af superpixel.m:

```
function  Y = superpixel( X, dimX, p )
% SUPERPIXEL computes the given image for a specified resolution, p.

% Call:    [ Y ] = superpixel(X, dimX, p)

% Input Parameter: X is a 2D or 3D array of type uint8 containing an image,
%                  dimX is the dimensions of the photo
%                  and p is the desired resolution.

% Ouput Parameter: Y is also such an array, containing the new superpixel
%                  version of X

% Authors:  Gustav Collin Rasmussen og Tony B. Rungling
% Date:     15/1-2010

% compute the superpixeled image of X

for i = 1 : dimX(3)
    for x = 1 : (dimX(1) / p)
        for y = 1 : (dimX(2) / p)
            X(1 + (x - 1) * p:x * p,1 + (y - 1) * p:y * p, i) =...
            mean(mean(X(1 + (x - 1) * p:x * p,1 + (y - 1) * p:y * p, i)));
        end
    end
end

Y = X;
```

Listning af chess.m:

```
function  Y = chess(X, dimX, p)
% Y computes the given image with a chessboard-pattern on top of it

% Call:             Y = chess(X, dimX, p)

% Input Parameter: X is a 2D or 3D array of type uint8 containing an image
%                  dimX is the dimensions of the image
%                  p is the size of the edge of the squares of the
%                  chessboard-pattern

% Ouput Parameter: Y is also such an array, containing the picture with the
%                  chessboard-pattern on top.

% Authors:         Gustav Collin Rasmussen og Tony B. Rungling
% Date:            15/1-2010

% compute the image of X with chessboard-pattern.

Negativ = negative(X);

for x = 1 : (dimX(1) / p)
    for y = 1 : (dimX(2) / p)
        if rem((x + y), 2) == 0
            X(1 + (x - 1) * p:x * p,1 + (y - 1) * p:y * p,:) =...
                Negativ(1 + (x - 1) * p:x * p,1 + (y - 1) * p:y * p,:);
        end
    end
end

Y = X;
```

Galleri

Først vises galleriet for farvefotoet butterflies.tif med originalbilledet først, og derefter hver af de otte funktioner anvendt på billedet. Derefter vises det samme for gråtone-billedet cameraman.tif

butterflies.tif:

butterflies original

butterflies grayscale

butterflies negative

butterflies warp

butterflies frame

butterflies mirror

butterflies jail

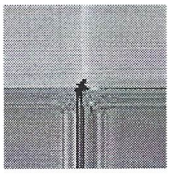
butterflies superpixel

butterflies chess

cameraman.tif:

cameraman original cameraman grayscale cameraman negative cameraman warp

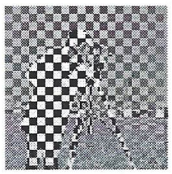


cameraman frame cameraman mirror cameraman jail cameraman superpixel




cameraman chess



Konklusion

I projekt 2 er vi blevet mere fortrolige med brug af arrays, grafik, og subprograms. Vi skrev de otte billedmanipulerings-funktioner først, hvorefter vi afprøvede dem enkeltvis i dims.m ved at udkommentere de andre.
Herefter arbejdede vi på at få galleriet til at fungere, samt muligheden for at skifte billede med newX.