Introduction to Computational and Biological Vision

A Generic image diffusion system

Final Project
by

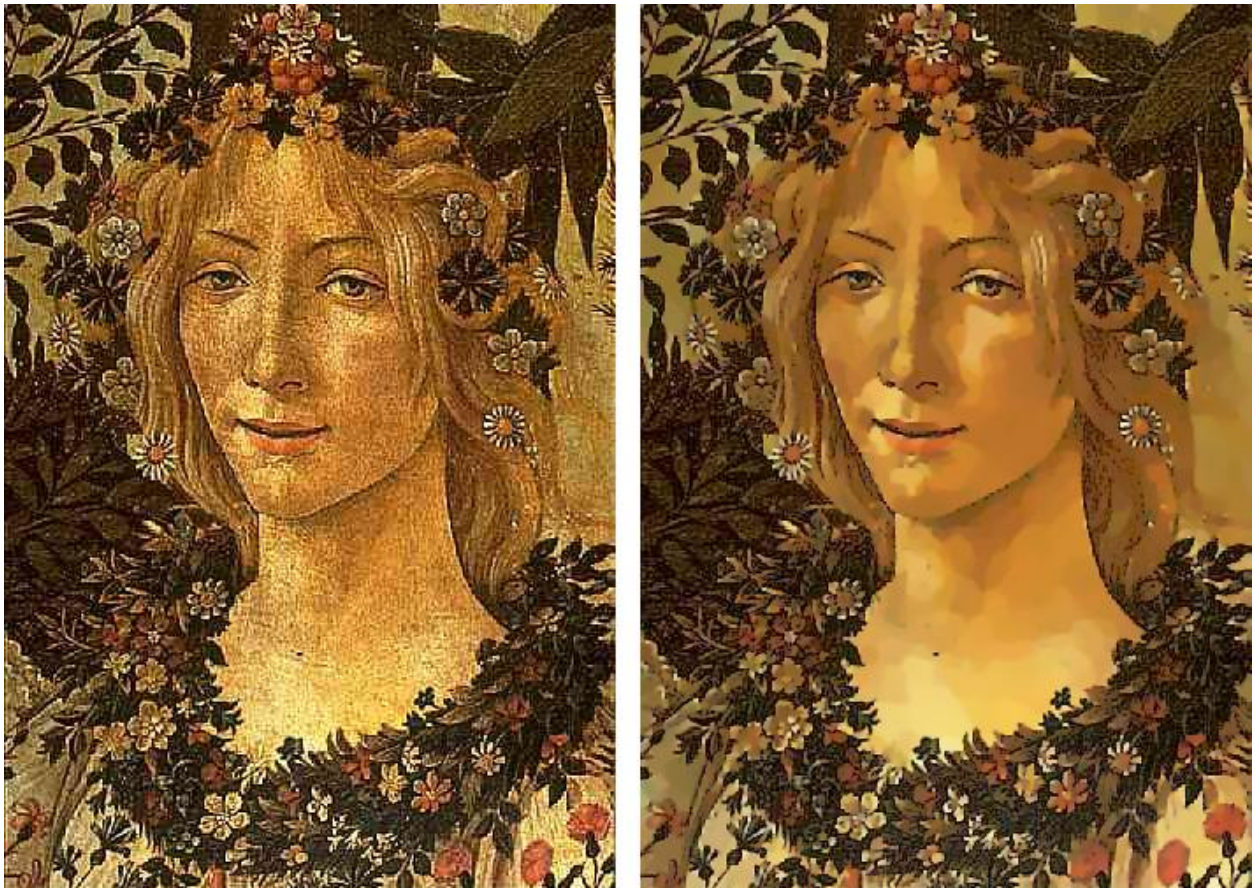
Alexander Shapiro
I.D. 310907589

Project's goals

Developing interactive system that will allow a plug&play type of mechanisms for image diffusion with built in input and output features for convenient production of results. Implementation of several classical diffusion techniques. The system has to be written in Matlab. "Plug&play type of mechanisms" means, no techniques for image processing will be hardcoded, but instead modules system will be used, thus allowing users to add additional image processing functions to the program.

Introduction

What is diffusion?
Physics definition: diffusion is the net action of matter (particles or molecules), heat, momentum, or light whose end is to minimize a concentration gradient.
In the image processing the different diffusion techniques are used for smoothing and edge detection.
This is an example of what can be achieved by applying diffusion to image:



Course of action

1. Constructing general framework for the system – basic image manipulations.
2. Adding the ability to process images.
3. Adding the ability to work with different modules.
4. Implementing two diffusion modules – Gaussian diffusion and Anisotropic Diffusion.

Implementation

1. First the **graphical user interface** for the system was constructed. It contains the image window, buttons for loading and saving images and buttons for enabling Zoom mode and Pan mode. The "Reset" button was added to return enlarged and/or panned image to its original size.

Load Image – opens standard file-open dialog and allows the user to choose image file of following formats: BMP, JPG, PNG, GIF. After image is chosen, the program loads it to the image area. The image is enlarged/reduces to fit the image area of the program. The program is capable of loading and showing color images, but no processing of such images is supported. The only images that are fully supported are grayscale 8-bit images.

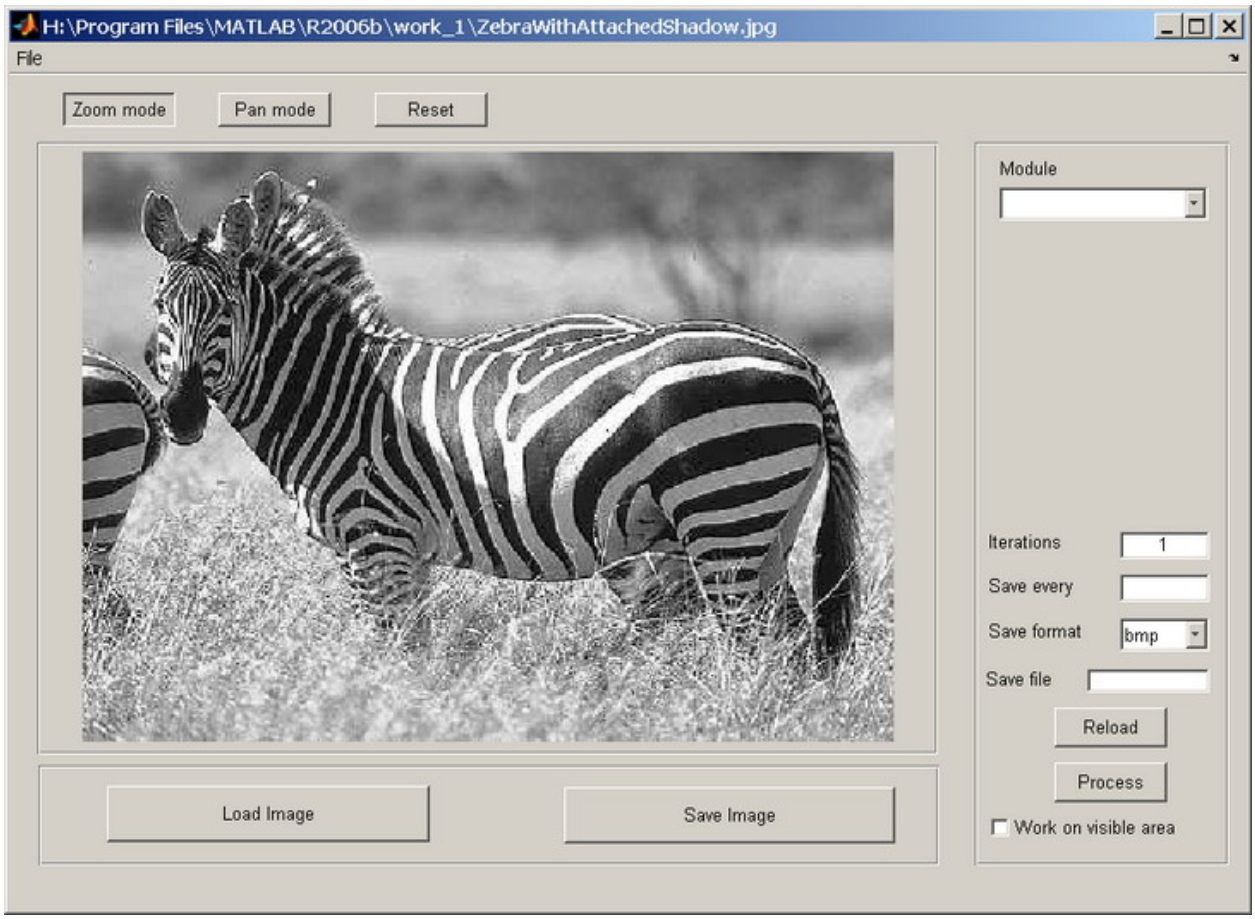
Save Image – opens standard file-save dialog and allows the user to save the image currently displayed in following formats: BMP, JPG, PNG, GIF and EPS (PostScript).
Note: when saving in EPS format, only the visible area of the image (if image was enlarged) is saved. This is due to some limitation of Matlab approach of printing to file.

Zoom mode – when button is pressed, it enables Zoom mode. In this mode, the mouse pointer changes to little circle with the plus sign inside.
Pointing the mouse on desired area and pressing the left mouse button, zooms in chosen region. To zoom the image out, we need to press right mouse button or alternatively press left mouse button while holding down the Alt key.
Another way to zoom in or out is to use mouse wheel. When left mouse button is pressed, the user can drag the mouse – rectangular area is selected and when the mouse button is released, this area is enlarged
To turn the zoom mode off, just press the Zoom button again. Enabling Zoom mode will disable Pan mode if it was previously enabled.
Note: the right mouse button and mouse wheel are only operational if the version of Matlab used is 7.3.0(2006b) or greater.

Pan mode – when button is pressed, it enables Pan mode. In this mode, the mouse pointer changes to little hand. Pressing the left mouse button and dragging the image will change the image area that is currently visible according to the drag direction. The Pan mode is useful when the image is enlarged and user wants to see another region of the image. Enabling Pan mode will disable Zoom mode if it was previously enabled.

Reset – when the image is enlarged and/or panned, pressing the Reset button will change the image size/position to fit the image window.
Note: pressing the Reset button will not change the currently enabled mode (normal/zoom/pan).

2. Adding the **modules panel**



The modules panel is added to the right side.

<u>This panel consists of several parts:</u>

Modules combo-box – list of diffusion modules found by program on the startup. The program searches for modules files in program's startup directory and loads every module it finds to the list. The empty space under the modules box is reserved for different module parameters and text fields for entering these parameters are added dynamically on module selection.

Iterations – the number in this text field defines how many times should the diffusion function be applied to the image. By default this number is equal to 1.

Save every – this parameter defines whether to save the intermediate results if the number of iterations is greater then 1. If the number in this text field is N, then every N iterations the result of the diffusion will be saved to a file. If N is greater

then the number in the Iterations field – this parameter is ignored and no intermediate results are saved.

Save format – this is combo-box that allows the user to choose the format of intermediate image saves. The formats are BMP, JPG, PNG, GIF.

Save file – this text field defines the prefix of the file names of intermediate results. The names of these files are constructed as follows: [prefix]_diffused_by_module_[name_of_diffusion_module]_[iteration_number]. By default, if this field is left empty, the prefix is "image".

Reload – pressing this button discards any changes maid to the image. The image is reloaded and no changes are saved.

Process – applies the selected diffusion module to the image, according to entered parameters.

Work on visible area checkbox – if checked, the diffusion is applied only to the visible part of the image. To process only part of the image, select the desired part with zoom rectangle and check this box.

3. Modules

The meaning of being generic is to allow the user to add new diffusion modules to the program in the simplest way. The system allows to convert any diffusion function in the form of Matlab's m-file to a module, if it accepts image matrix as input, takes no more then 5 additional parameters and produces the output matrix with the same size. To create a module we first need to add prefix "mod_" to the function's file name and to the function itself, also there are no spaces allowed in the m-file name. Use underscore symbol instead of space. On startup the program seeks all the files with this prefix and adds it to the available modules list. But this is not enough to convert the function to a module. We need to add 8 lines of specific comments to the beginning of the function and change (if needed) function's signature. We change the signature as follows:

Function has to accept the image matrix as first parameter and 5 additional numerical parameters. It doesn't have to use all or any of these parameters, bat they have to be defined.

Here is an example:

function out = mod_diffusion_name(in_matrix,real_param_1,real_param_2,fake1,fake2,fake3)

Here we have diffusion function with name "mod_diffusion_name", 2 real parameters, that the function uses – "real_param_1" and "real_param_2", and 3 additional parameters that are not used by the function – "fake1","fake2","fake3".
When we use specific modules, we not just need to enter the additional parameters for the module, but also we need to know the meaning of each parameter. So we have to find the way to define the parameters names in the module. Here we use 8 special lines of comments:

```
%-------------------------------------------
% params_number=3
% param_name1=First_parameter_name default=1
% param_name2=Second_parameter_name default=5.2
% param_name3=Third_parameter_name default=10
% param_name4=null default=0
% param_name5=null default=0
%-------------------------------------------
```

The first line is a separation line – it should not contain spaces.
On the second line we define the number of parameters that the function uses:

```
% params_number=[number_of_parameters]
```

Third through seventh lines define the names of the parameters and default values that are used if the user doesn't enters them:

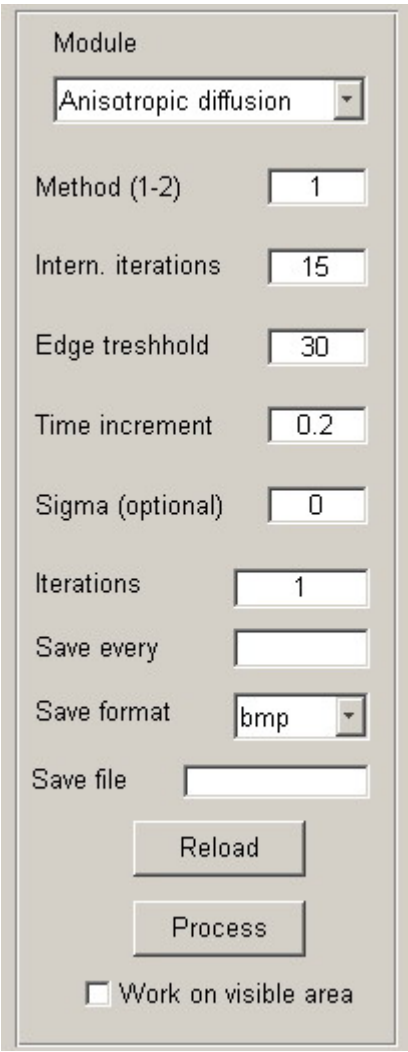
```
% param_name1=[name_of_the_parameter] default=[default_value]
```

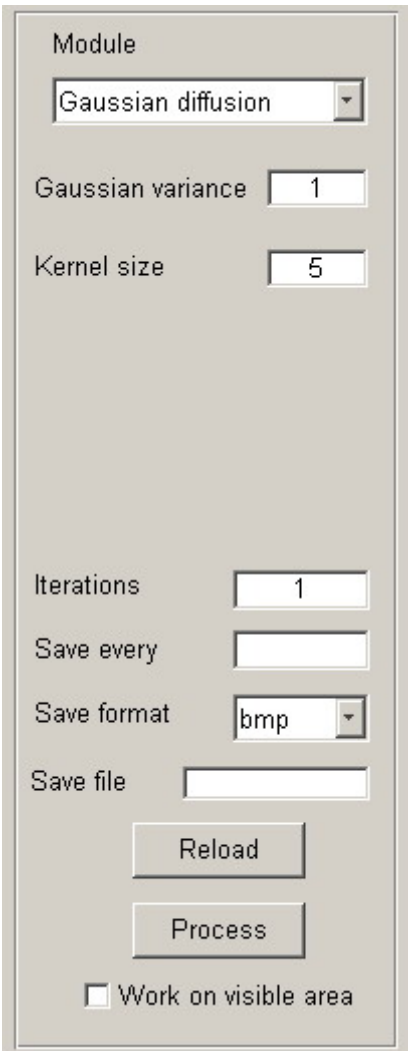
The name of the parameter should not contain spaces – use underscore instead.
The default value should be integer or floating point number.
If the parameter is not used – its name should be set to "null" and its default value should be set to "0".
The last line is exactly the same as the first line – just a separator.

```
%-------------------------------------------
% params_number=5
% param_name1=Method_(1-2) default=1
% param_name2=Intern._iterations default=15
% param_name3=Edge_treshhold default=30
% param_name4=Time_increment default=0.2
% param_name5=Sigma_(optional) default=0
%-------------------------------------------
function Jd=mod_Anisotropic_diffusion(J,method,N,K,dt,sigma2)
```

```
%-------------------------------------------
% params_number=2
% param_name1=Gaussian_variance default=1
% param_name2=Kernel_size default=5
% param_name3=null default=0
% param_name4=null default=0
% param_name5=null default=0
%-------------------------------------------
function out = mod_Gaussian_diffusion(in,sigma,kernel_size,fake1,fake2,fake3)
```

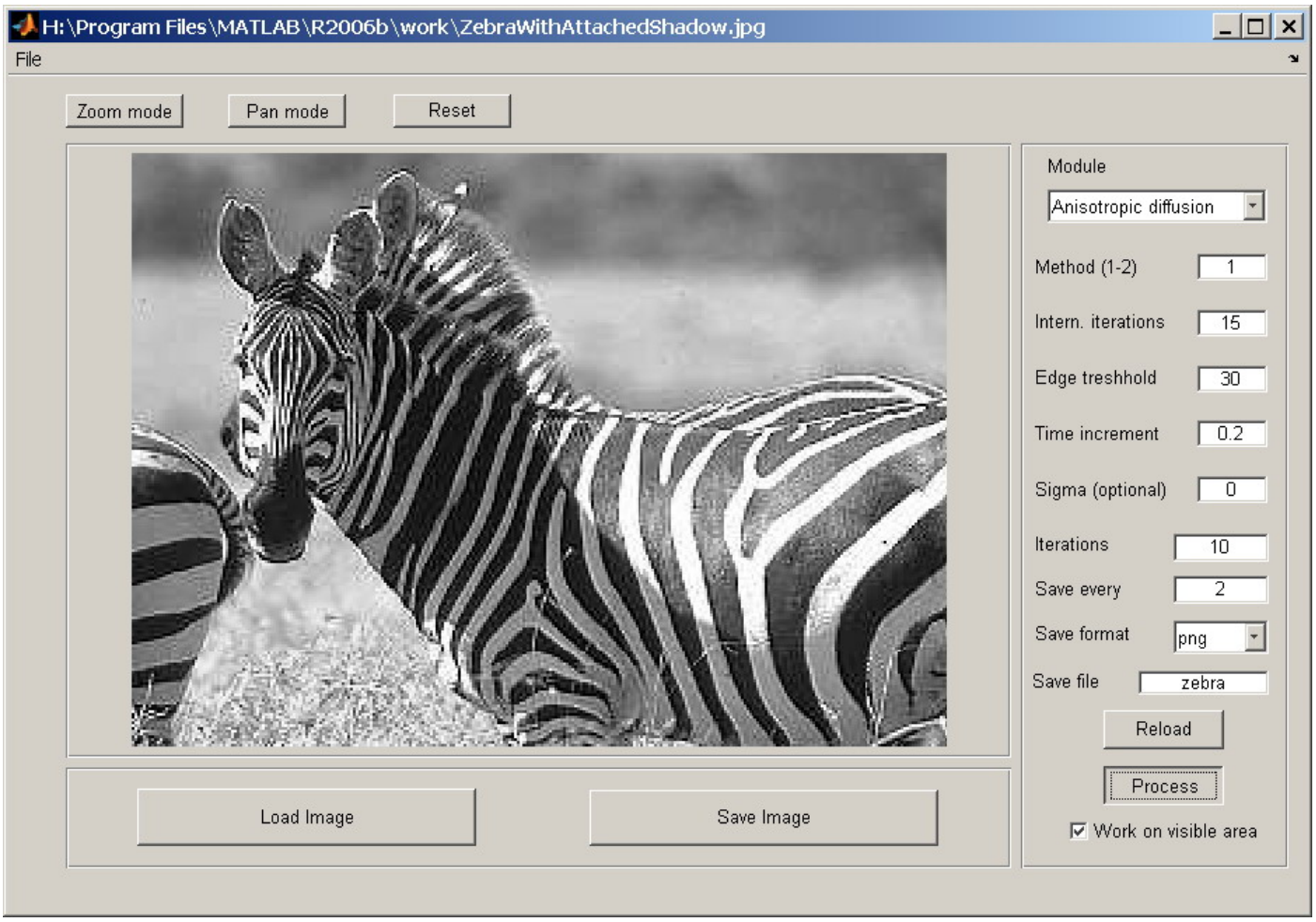
When the module name is selected from the available modules list, the system parses the module's file header and extracts needed information:





We can see here that all the names that were entered with underscore are converted to normal names with spaces.

And finally the whole program looks like this:



4. **<u>Diffusion modules</u>**

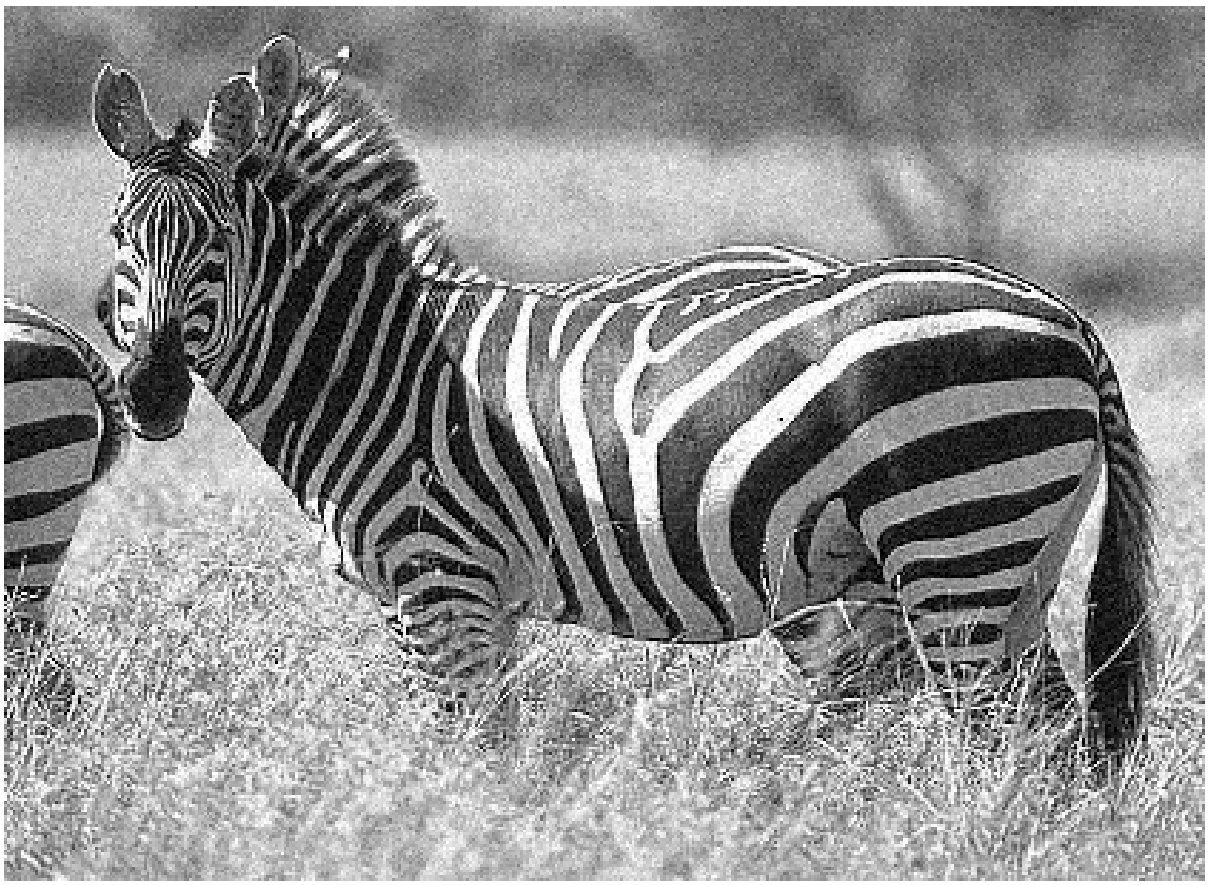
Currently, there are 2 diffusion modules implemented and fully operational.

The first one is ***Gaussian diffusion*** – used for smoothing the image. The algorithm is very simple. It uses convolution between image matrix and square kernel matrix.
The size of the kernel and the values of each element is calculated according to parameters entered by user in the module's panel. The user defines the size of the kernel and the value of Gaussian variance (sigma parameter).
Each element of the kernel is calculated by following expression:

$$a(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

This diffusion method is very simple and fast, but it doesn't preserve edges of objects in an image.

The sample image with additive noise:



And the same image after applying Gaussian diffusion:

As we can see – the noise is gone, but the image looks blurred.
Of cause we can get a better result, if we adjust the kernel size and the variance parameters.

The second diffusion method implemented in the module is ***Anisotropic diffusion***.

The algorithm is based on the article "Scale-Space and Edge Detection Using Anisotropic Diffusion" written by Pietro Perona and Jitendra Malik in 1990. Perona and Malik formulate the anisotropic diffusion filter as a diffusion process that encourages intraregion smoothing while inhibiting interregion smoothing. They proposed two methods of anisotropic diffusion. Both methods are implemented in the module and user can choose each one of them.

When the module is applied to the sample image, we get following results:

We can see here that only the background is blurred – the edges of zebra's lines are enhanced.

Conclusion

The goal of the project is achieved – we have fully functional framework with graphical user interface and plug&play modules for image manipulation and processing. The system architecture allows very simple addition of new diffusion modules or conversion of existing Matlab's functions to the module's format. Also two diffusion modules are implemented – Gaussian diffusion and Anisotropic diffusion (two methods of Perona and Malik).

The system was tested on Windows platform (WinXP SP2 English) and requires Matlab version 7.3.0.267(R2006b), August 03, 2006 to operate correctly.

Date: 22/02/2007

References:

- Built-in Matlab help
- Various sources about Matlab programming on the Internet
- Gaussian Smoothing - http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
- Nonlinear Anisotropic Diffusion Filtering - http://www.cs.sfu.ca/~stella/papers/blairthesis/main/node22.html
- PDE-based Image Filtering - http://visl.technion.ac.il/~gilboa/PDE-based_image_filtering.html
- P. Perona, J. Malik "Scale-space and edge detection using anisotropic diffusion", PAMI 12(7), pp. 629-639, 1990