

**GWFR**

**GEOMLIB (β)**

**MARINE ENGINEERING GEOPHYSICAL  
DATA PROCESSING TOOLBOX**

Ivan V. Dmitriev  
04.11.2019

**Contents**

1	Image-data with world-file and XYZ-data description .....	4
1.1	XYZ gridded data.....	4
1.2	World-file name .....	4
1.3	World-file main contents.....	4
1.4	World-file additional contents.....	5
1.5	Raster-image type, matrix-image type and XYZ-data type.....	6
1.6	RGB-image type.....	6
1.7	Palette features .....	7
1.8	gWfr's Structures description.....	7
2	Functions description .....	8
2.1	Read raster-image with world-file.....	8
2.2	Read Geotiff-image file.....	8
2.3	Write raster-image with world-file.....	9
2.4	Draw image with world-file's data.....	9
2.5	Draw image with world-file's data.....	10
2.6	Convert XYZ-grid-data to matrix-image .....	11
2.7	Convert image to XYZ-grid-data .....	13
2.8	Convert matrix-image to raster-image .....	14
2.9	Convert raster-image to matrix-image .....	15
2.10	Cut rectangular zone from image .....	16
2.11	Adds/cut borders to image.....	17
2.12	Resize image .....	18
2.13	Combine colors/palettes for raster-image1 and raster-image2.....	19
2.14	Merge Image1 and Image2 using world-data.....	21
3	Example.....	23
	Citation.....	25

## ***Figures list***

<b><i>Figure 1</i></b> World file components.....	5
<b><i>Figure 2</i></b> Data-types and functions for conversion .....	6
<b><i>Figure 3</i></b> gWfrDraw example results .....	10
<b><i>Figure 4</i></b> gWfrXyzDraw example results.....	11
<b><i>Figure 5</i></b> gWfrXyz2Mat example1 results (the absetnt points in border was drawn yellow) .....	12
<b><i>Figure 6</i></b> gWfrXyz2Mat example2 results (the absetnt points in border was drawn yellow) .....	12
<b><i>Figure 7</i></b> gWfrXyz2Mat LX and LY handle forming .....	13
<b><i>Figure 8</i></b> gWfrIm2Xyz example results .....	14
<b><i>Figure 9</i></b> gWfrMat2Im example results .....	15
<b><i>Figure 10</i></b> gWfrCut example results .....	16
<b><i>Figure 11</i></b> gWfrAdd example results .....	18
<b><i>Figure 12</i></b> gWfrResize example results .....	19
<b><i>Figure 13</i></b> gWfrCombineColors example results .....	21
<b><i>Figure 14</i></b> gWfrMergeData example results .....	22
<b><i>Figure 15</i></b> gPts2DFilt script's steps .....	24

## 1 Image-data with world-file and XYZ-data description

The functions set were created for manipulations with SSS mosaic and pts-files (XYZ-gridded-data). There are:

- resize, cut and merge SSS-mosaic;
- cut SSS-mosaic image file with world-file in several parts;
- convert pts-file (MBES grid saved as XYZ) to matrix, apply processing and convert matrix to pts-file back (see *Example*).

gWfr lets to do elementary manipulations with raster-image file with world-file (read, write, merge, merge palettes, etc) and convert XYZ-gridded-data to raster-image file with world-file creation.

### 1.1 XYZ gridded data

This data is gridded in “parallelogram” web-cell (prefer rectangular web-cell with perpendicular grid-lines) and format to three columns: X, Y and Z. The grid web can be rotate relative axis (*Figure 1*).

### 1.2 World-file name

The world-file name must be same with image name, but extension must be correct: there are three letters; first letters is first letters of image’s extension; second letters is last letters of image’s extension; third letter is “w”. For example: \*.tif and \*.tfw; \*.tiff and \*.tfw; \*.png and \*.pgw.

### 1.3 World-file main contents

The world file is an ASCII text file, which contains the following lines:

Line1\_A: pixel size in the x-direction in map units;

Line2\_D: rotation parameter about y-axis;

Line3\_B: rotation parameter about x-axis;

Line4\_E: pixel size in the y-direction in map units, almost always negative;

Line5\_C: x-coordinate of center of upper left pixel;

Line6\_F: y-coordinate of center of upper left pixel.

**The D and B parameters are not angular rotations and that the A and E parameters do not correspond to the pixel size!** Better descriptions of the A, D, B and E parameters are (*Figure 1*):

Line 1: A: x-component of the pixel width (x-scale);

Line 2: D: y-component of the pixel width (y-skew);

Line 3: B: x-component of the pixel height (x-skew);

Line 4: E: y-component of the pixel height (y-scale), typically negative.

([https://en.wikipedia.org/wiki/World\\_file](https://en.wikipedia.org/wiki/World_file))

The “rotation” is “around” (C,F) point’s coordinates.

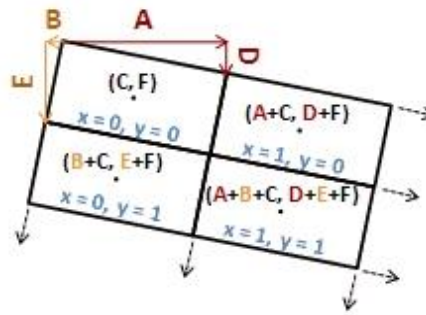


Figure 1 World file components

The Coordinates conversion:

Forward conversion:  $[x';y']=[A \ B \ C; \ D \ E \ F]*[x;y;1]$  or  $x'=Ax+By+C; y'=Dx+Ey+F;$

Reverse conversion:  $x=(Ex'-By'+BF-EC)/(AE-DB); y'=(-Dx'+Ay'+DC-AF)/(AE-DB);$

where,

$x,y$  – column and row raster's number for pixel;

$x';y'$  – pixels coordinates in axis XY (see below)

^+Y

|

----> +X

The left\_up\_angle of image (Line5\_C, Line6\_F) is corresponded to column-1 and raw-1 of raster/matrix.

#### 1.4 World-file additional contents

The world-file can include four additional values: [key aS bS BgVal]. These values are used to describe physical values in raster.

Line7\_key: 9999999999;

Line8\_aS: multiple for image's color value;

Line9\_bS: shift for image's color value;

Line10\_BgVal: the code of "absent" color (it is means – no data for raster's pixel with BgVal code).

The  $DataOriginalValue=aS*ColorValue+bS;$  so, we can convert coded pixels color to its Original Value (for example in meters for bathymetry).

File's example, with additional lines:

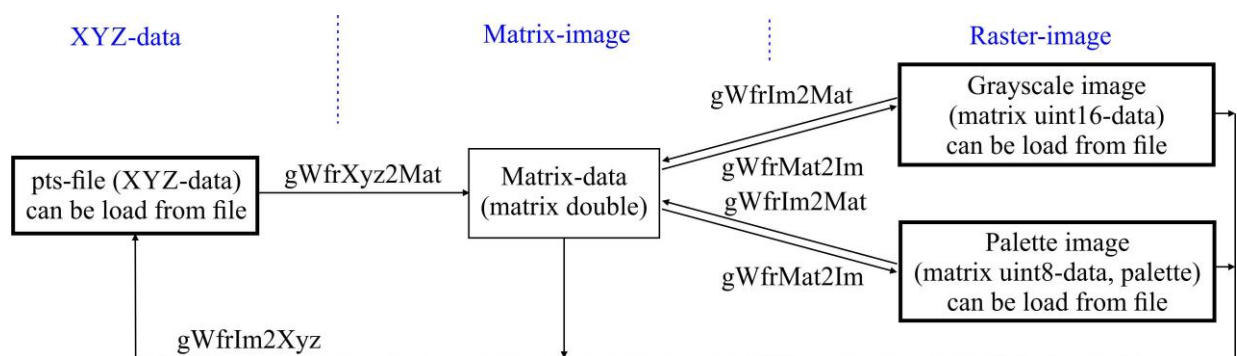
Line from file	Description
0.10001159	x-component of the pixel width (x-scale)
0	y-component of the pixel width (y-skew)
0	x-component of the pixel height (x-skew)
-0.1	y-component of the pixel height (y-scale), typically negative
684083.88367	x-coordinate of center of upper left pixel
5887188.48824	y-coordinate of center of upper left pixel
9999999999	the key "below is additional content"
-0.01	the multiple for color value
-38	the shift for color value
255	the code of "absent" color (no data)

## 1.5 Raster-image type, matrix-image type and XYZ-data type

The follow raster-image types are used:

- Palette image. 2D-matrix with values from 0 to 255 (usually, values type is uint8), refer to Palette. Palette contained three columns R-G-B, with double numbers from 0 to 1; for some functions means, that Palette is gray (for example gWfrCombineColors). The “absent” color code (values from 0 to 255) set in Head.BgVal field (see description below). When XYZ convert to Palette image (by gWfrMat2Im function), the Head.BgVal=255.
- Grayscale image. 2D-matrix with values from 0 to 65536 (usually, values type is uint16); there is no palette, the image’s uint16 numbers are code grey colors. The “absent” color code (values from 0 to 65536) set in Head.BgVal field (see description below). When XYZ convert to Grayscale image (by gWfrMat2Im function), the Head.BgVal=65535.
- Matrix-image. 2D-matrix (values type is double), includes DataOriginalValues. When XYZ convert to matrix (by gWfrXyz2Mat function), the Head.BgVal=nan. Head.K is empty.
- XYZ-data. There are three rows, which contained X, Y and Z values (type double).

The Data-types and functions for conversion are showed in *Figure 2*.



*Figure 2* Data-types and functions for conversion

## 1.6 RGB-image type

The functions gWfrRead, gWfrGeotiffRead, gWfrWrite, gWfrDraw can work with R-G-B-image and R-G-B-image with alpha-layer. But, today I have no idea – how the reference to “absolute scale” can be realized with using “multiple for image’s color value” and “shift for image’s color value”. We need table “RGB-color - drawing parameter absolute value” to do it.

Today the MatLab’s functions can used to do it “hardly”:

rgb2gray – convert RGB image or colormap to grayscale;

rgb2ind – convert RGB image to indexed image;

cmunique – eliminate duplicate colors in colormap; convert grayscale or truecolor image to indexed image.

## 1.7 Palette features

The Palette or Grayscale images, was created by gWfrMat2Im function, have a “linear” colors change (constant step between colors levels). The RGB-palette for some functions means gray (used only first column “Red”).

The DataOriginalValues are calculated, using a) 2D-matrix values for Grayscale image and b) 2D-matrix values and Palette values for Palette image:  $\text{DataOriginalValue} = aS * \text{Color} + bS$ .

## 1.8 gWfr’s Structures description

The Paletter image or Grayscale image is load to 2D-matrix (uint8 or uint16), named as Data.

The world-file data load to Head structure, which includes:

Head.Color – colormap for raster-image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value";

Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code).

When used gWfrGeotiffRead, three additional fields can create:

Head.ProjName – the values from geotiff’s GeoAsciiParamsTag;

Head.ProjInfo – the values from geotiff’s GeoDoubleParamsTag;

Head.GeoKeyInfo – the values from geotiff’s GeoKeyDirectoryTag,

These fields are not used in gWfr-functions and created as information only.

## 2 Functions description

### 2.1 Read raster-image with world-file

**function [Head,Data]=gWfrRead(fname)**

Read raster-image file with world-file. Example: \*.tif and \*.tfw.

Parameters:

fname – image’s file name with extension (the world-file extension created as first and last letters of the image's extension and ending with a 'w');

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;  
DataOriginalValue=a\*Color+b;

Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Data – raster-image data matrix.

Function Example:

```
>> [Data,Head]=gWfrRead('d:\03_Block-3_SSS_Mosaic\Block-3_2.tif');
```

### 2.2 Read Geotiff-image file

**function [Head,Data,inf]=gWfrGeotiffRead(fname)**

Read Geotiff-image file and copy world-file values from tiff’s geo-tags.

To extract all geotiff information use: geotiffinfo, geotiffread, geotiffwrite form Mapping Toolbox. See tiff tags description in <https://www.awaresystems.be/imaging/tiff/tifftags.html>;  
<https://www.loc.gov/preservation/digital/formats/fdd/fdd000279.shtml>

Parameters:

fname – Geotif-image file name with geo-tags;

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;  
DataOriginalValue=a\*Color+b;

Head.BgVal=[] – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Head.ProjName – GeoAsciiParamsTag values;

Head.ProjInfo – GeoDoubleParamsTag values;

Head.GeoKeyInfo – GeoKeyDirectoryTag values;

Data – raster-image data matrix.

inf – image information ("imread" function’s output).



Function Example:

```
>> [Head,Data,~]=gWfrGeotiffRead('d:\03_Block-3_SSS_Mosaic\Block-3_2.tif');
```

### 2.3 Write raster-image with world-file

**function gWfrWrite(fname,Head,Data)**

Write raster-image file with world-file.

Parameters:

fname – image’s file name with extension (the world-file extension created as first and last letters of the image's extension and ending with a 'w');

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

$DataOriginalValue=a*Color+b$ ;

Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Data – raster-image data matrix.

Function Example:

```
>> gWfrWrite('d:\03_Block-3_SSS_Mosaic\Block-3_2.tif',Head,Data);
```

### 2.4 Draw image with world-file’s data

**function gWfrDraw(Head,Data,figNum,key)**

Draw raster-image or matrix-image in world-data axis.

Parameters:

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

$DataOriginalValue=a*Color+b$ ;

Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Data – raster-image or matrix-image data matrix or matrix.

figNum – figure number;

key – drawing method: 1) is image with pixels numbers in axis; 2) is image with axis constructed from Head.Wf (angles must be zero); 3) plot3 with axis constructed from Head.Wf; 4) is surf with axis constructed from Head.Wf.

Function Example:

```
>> [Data,Head.Color]=imread('d:\Ge0MLib_logo3.png'); Head.Wf=[1 0 0 -1 100 50]; Head.K=[1 0];  
Head.BgVal=0;
```

```

>> gWfrDraw(Head,Data,10,1); % Figure 3, a
>> gWfrDraw(Head,Data,10,2); % Figure 3, b
>> gWfrDraw(Head,Data,10,2); % Figure 3, c
>> gWfrDraw(Head,Data,10,4);colormap('winter'); % Figure 3, d

```

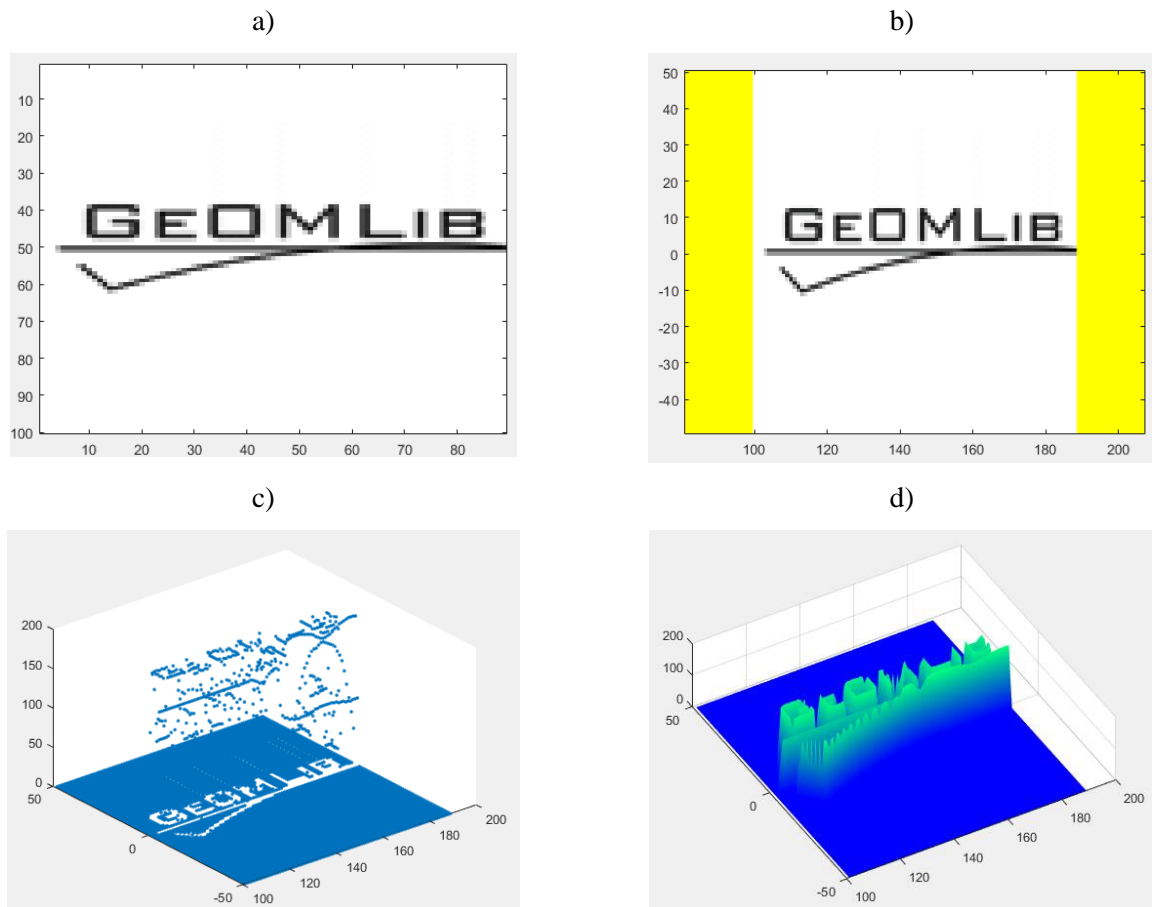


Figure 3 gWfrDraw example results

## 2.5 Draw image with world-file's data

**function gWfrXyzDraw(XYZ,figNum,key)**

Draw XYZ-points.

Parameters:

XYZ – X,Y,Z-data in 3-rows-vector;

figNum – figure number;

key – drawing method: 1) draw plot(X,Y) in the plane like to postmap, axis equal; 2) draw plot3(X,Y,Z) in 3D, axis not equal;

the "Data cursor" instrument can used to draw X,Y,Z values for point; when drawing, the values X,Y,Z copy to clipboard.

Example:

```

>> XYZ=dlmread('e:\example1.pts');gWfrXyzDraw(XYZ,10,1); % Figure 4

```

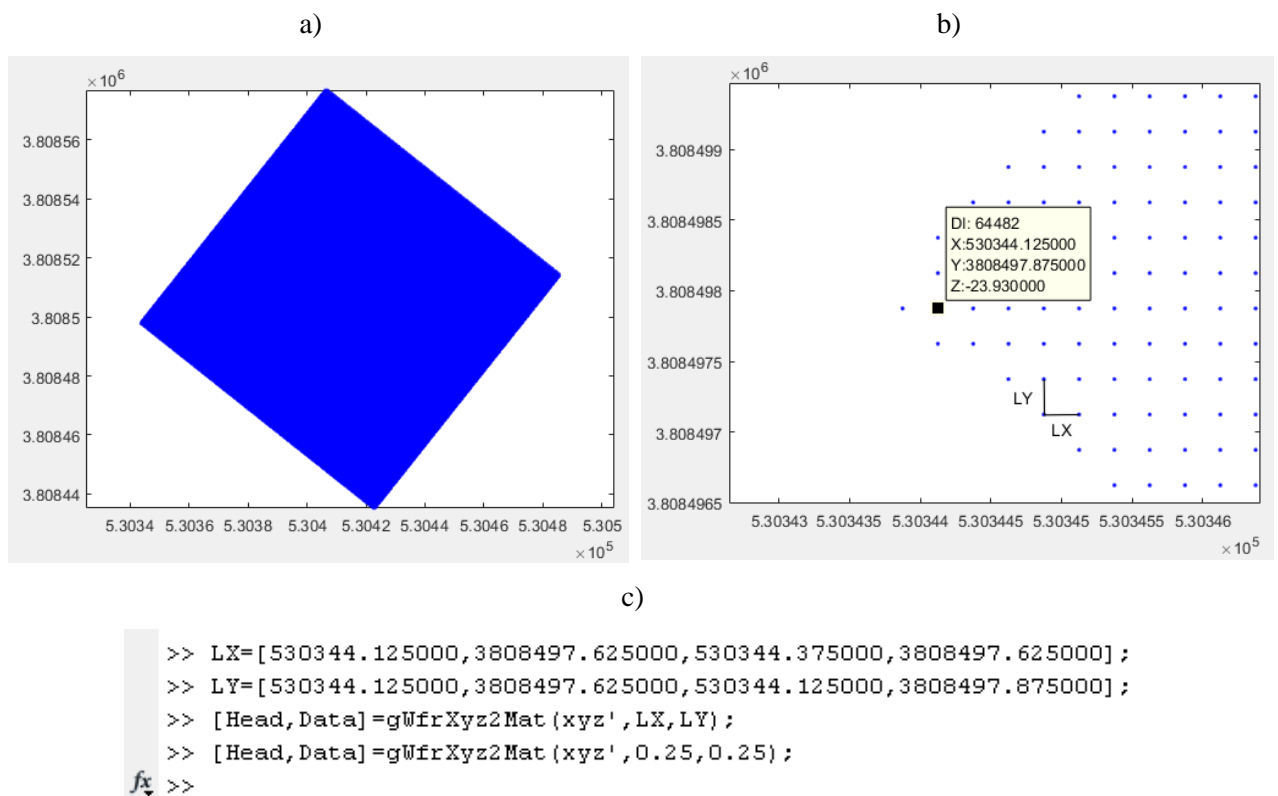


Figure 4 gWfrXyzDraw example results

- a) post-map for all data;
- b) zoomed part; the "Data cursor" instrument using example;
- c) LX and LY values formed using "Data cursor" and clipboard (input for gWfrXyz2Mat function).

## 2.6 Convert XYZ-grid-data to matrix-image

**function [Head,Data]=gWfrXyz2Mat(XYZ,LX,LY)**

Convert XYZ-data to matrix-image and Head-structure. Z values write to matrix, not existing values set to nan. SkewX and skewY must be zero. **Will be changed in future.**

Parameters:

XYZ – rows with X,Y-coordinates and Z-data.

LX=const or LX=[lx\_x1 lx\_y1 lx\_x2 lx\_y2] is horizontal segment along X axis; see [Figure 7](#) and [Figure 4](#);

LY=const or LY=[ly\_x1 ly\_y1 ly\_x2 ly\_y2] is vertical segment along Y axis; see [Figure 7](#) and [Figure 4](#);  
if grid-web lines parallel to X and Y axis, than can use LX=web-step-along-X, LY=web-step-along-Y.

Head – header structure, which includes:

Head.Color – colormap, empty for matrix image;

Head.Wf – world-file values: [scaleX 0 0 scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

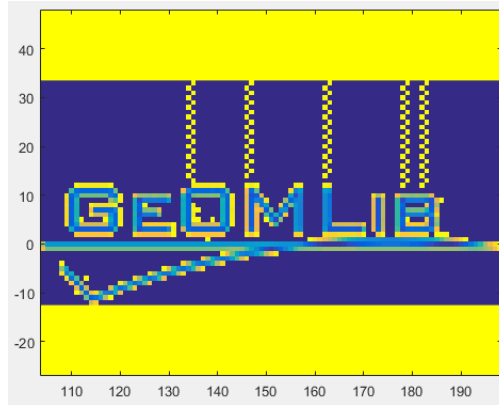
Z=DataOriginalValue=a\*Color+b;

Head.BgVal=nan – the code of "absent" color (it is means – no data for raster's pixel with BgVal code);

Data – matrix-image (type double);

### Function Example1:

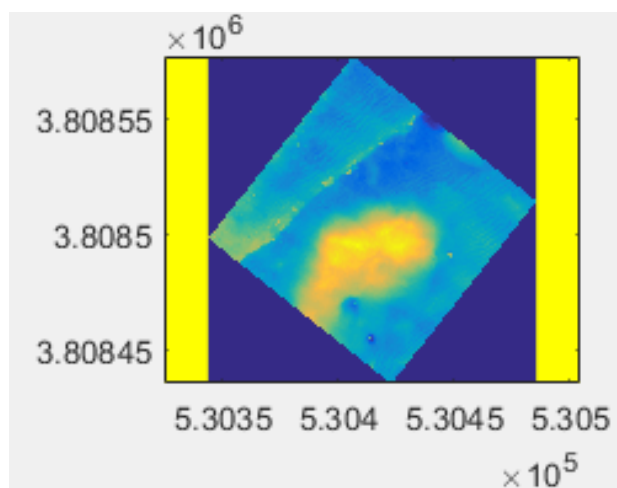
```
>> [Data,Head.Color]=imread('d:\Ge0MLib_logo3.png'); Head.Wf=[1 0 0 -1 100 50]; Head.K=[1 0];  
Head.BgVal=0;  
>> XYZ=gWfrIm2Xyz(Head,Data,0,[],1);  
>> [Head1,Data1]=gWfrXyz2Mat(XYZ,[0 5 1 5],[5 0 5 1]);  
>> gWfrDraw(Head1,Data1,10,2); % Figure 5
```



*Figure 5* gWfrXyz2Mat example1 results (the absetnt points in border was drawn yellow)

### Function Example2:

```
>> xyz=dlmread('e:\example1.pts'); gWfrXyzDraw(xyz,10,1);  
>> LX=[530344.125000,3808497.625000,530344.375000,3808497.625000]; % use gWfrXyzDraw  
>> LY=[530344.125000,3808497.625000,530344.125000,3808497.875000]; % use gWfrXyzDraw  
>> [Head,Data]=gWfrXyz2Mat(xyz,LX,LY);  
>> [Head,Data]=gWfrXyz2Mat(xyz,0.25,0.25); % the grid-lines are parallel to X&Y axis, see Figure 4  
>> gWfrDraw(Head,Data,10,2); % Figure 6
```



*Figure 6* gWfrXyz2Mat example2 results (the absetnt points in border was drawn yellow)

X	Y	Z	
530422.625	3808435.375	-24.138	
530422.875	3808435.375	-24.121	
530422.375	3808435.625	-24.154	LY
530422.625	3808435.625	-24.131	
530422.875	3808435.625	-24.120	
530423.125	3808435.625	-24.117	
530422.125	3808435.875	-24.157	LX
530422.375	3808435.875	-24.138	
530422.625	3808435.875	-24.127	
530422.875	3808435.875	-24.124	
530423.125	3808435.875	-24.127	
530423.375	3808435.875	-24.141	
530421.625	3808436.125	-24.163	
530421.875	3808436.125	-24.160	
530422.125	3808436.125	-24.141	
530422.375	3808436.125	-24.133	
530422.625	3808436.125	-24.124	

LX=[530422.125, 3808436.125, 530422.375, 3808436.125];  
 LY=[530422.875, 3808435.375, 530422.875, 3808435.625];

Figure 7 gWfrXyz2Mat LX and LY handle forming

## 2.7 Convert image to XYZ-grid-data

**function XYZ= gWfrIm2Xyz(Head,Data,BgVal,K,SortKey)**

Convert raster-image or matrix-image and Head-structure to XYZ-data. The value of color (first palette's column for palette image) is used for Z creation.

Parameters:

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

$$Z = \text{DataOriginalValue} = a * \text{Color} + b;$$

Head.BgVal – the code of “absent” color (it is means – no data for raster's pixel with BgVal code);

Data – raster-image or matrix-image data matrix;

K – forced multiple (a) and shift (b) for "Data Original Value" calculation from Color; if K==[], then used Head.K;

BgVal – forced code of “absent” color; if BgVal==[], then used Head.BgVal; it can set to NaN (for matrix with type double); excepted in XYZ;

SortKey – if 1 than sort along Y (values along X change quick), else sort along X (values along Y change quick);

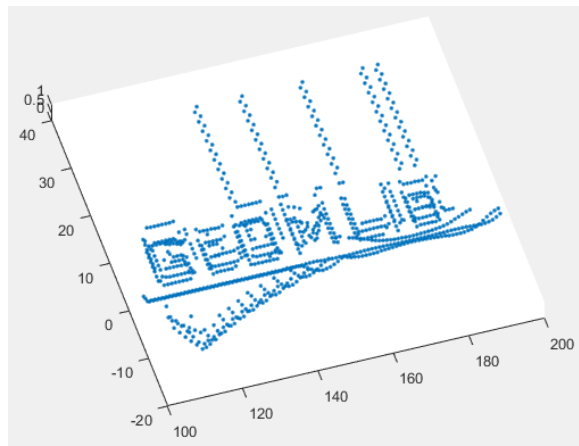
XYZ – rows (type double) with X,Y-coordinates and Z-data without BgVal value; if Head.K is not empty, then  $Z = a * \text{Head.Color} + b$ .

Function Example:

```
>> [Data,Head.Color]=imread('d:\Ge0MLib_logo3.png'); Head.Wf=[1 0 0 -1 100 50]; Head.K=[1 0];  
Head.BgVal=0;
```

```
>> XYZ=gWfrIm2Xyz(Head,Data,0,[],1);
```

```
>> figure(2);plot3(XYZ(1,:),XYZ(2,:),XYZ(3,:),'');axis xy; % Figure 8
```



*Figure 8* gWfrIm2Xyz example results

## 2.8 Convert matrix-image to raster-image

**function [Head,Data]= gWfrMat2Im(Head,Data,MinMax,BgVal,imType)**

Convert matrix-image to raster-image: palette (254 levels + “absent” color=255, grey palette) or grayscale (65534 levels + “absent” color=65535).

Parameters:

Head – input header structure, which includes:

Head.Color=[] – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

DataOriginalValue=a\*Color+b;

Head.BgVal=nan – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Data – input matrix-image (type double);

MinMax – minimal and maximal values link to level-0 and level-254/65534; if empty, than min and max will calculate from Data;

BgVal – code of “absent” color for palette (palette image only);

imType – raster-image type: 'uint8' or 'palette'- palette, 'uint16' or 'grayscale' - grayscale without palette;

Head – output header structure, which includes:

Head.Color – linear palette for palette-image or empty for grayscale-image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value"; DataOriginalValue=a\*Color+b;

Data – output raster-image data matrix. Function Example 1:

```
>> [Head1,Data1]=gWfrMat2Im(Head,Data,[],1,'palette');
```

Function Example 2:

```
>> [Data,Head.Color]=imread('d:\Ge0MLib_logo3.png'); Head.Wf=[1 0 0 -1 100 50]; Head.K=[1 0];
```

```
Head.BgVal=0;
```

```
>> XYZ=gWfrIm2Xyz(Head,Data,0,[],1);
```

```

>> [Head1,Data1]=gWfrXyz2Mat(XYZ,[0 5 1 5],[5 0 5 1]); % input "background" to Data1 as nan
>> [Head3,Data3]=gWfrMat2Im(Head1,Data1,[],1,'palette'); % set "background" color level-255 [1 1 1];
>> gWfrDraw(Head,Data,10,2); % Figure 9, a
>> gWfrDraw(Head3,Data3,11,2); % Figure 9, b

```

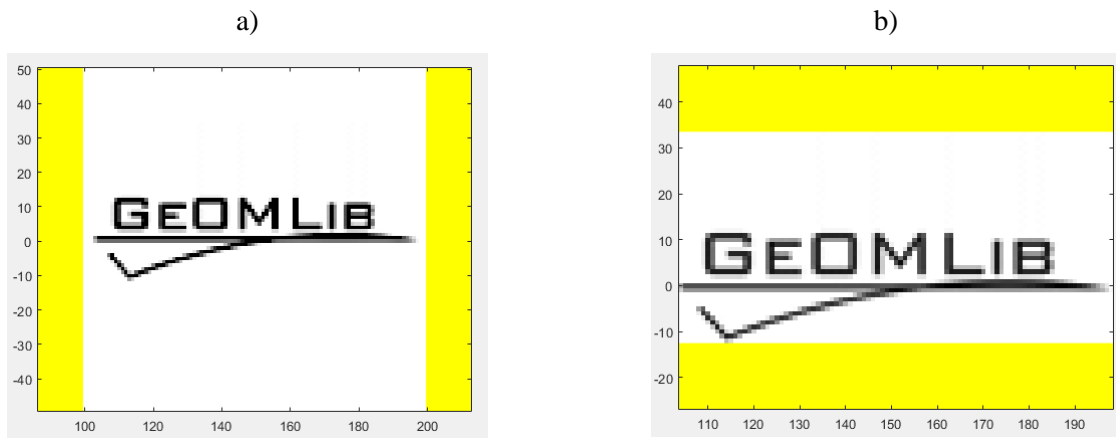


Figure 9 gWfrMat2Im example results

## 2.9 Convert raster-image to matrix-image

**function [Head,Data]=gWfrIm2Mat(Head,Data,BgVal,K)**

Convert raster-image (palette or grayscale) to matrix-image (type double).

Parameters:

Head – input header structure, which includes:

Head.Color – palette for palette-image or empty for grayscale-image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value"; DataOriginalValue=a\*Color+b;

Data – input raster-image data matrix.

K – forced multiple (a) and shift (b) for "Data Original Value" calculation from Color; if K==[], then used

Head.K;

BgVal – forced code of "absent" color; if BgVal==[], then used Head.BgVal;

Head – output header structure, which includes:

Head.Color=[] – colormap, empty for matrix image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

DataOriginalValue=a\*Color+b;

Head.BgVal=nan – the code of "absent" color (it is means – no data for raster's pixel with BgVal code);

Data – output matrix-image (type double);

Function Example:

```

>> [Head1,Data1]=gWfrIm2Mat(Head,Data,[],[]);

```

## 2.10 Cut rectangular zone from image

**function [Head,Data]=gWfrCut(Head,Data,LimS)**

Cut rectangle from raster-image or matrix-image and correct world-data; cutting zone input in pixels.

Parameters:

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

$Z=DataOriginalValue=a*Color+b$ ;

Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Data – raster-image or matrix-image data matrix;

LimS – limits for cut zone [minX maxX minY maxY] in pixels.

Math:

Line1\_A: pixel size in the x-direction in map units;

Line2\_D: rotation parameter about y-axis;

Line3\_B: rotation parameter about x-axis;

Line4\_E: pixel size in the y-direction in map units, almost always NEGATIVE;

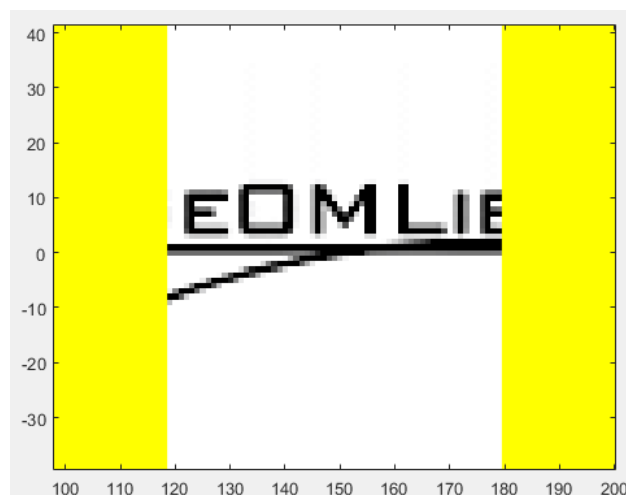
Line5\_C: x-coordinate of center of upper left pixel;

Line6\_F: y-coordinate of center of upper left pixel.

$x'=Ax+By+C$ ;  $y'=Dx+Ey+F$ .

Function Example:

```
>> [Data,Head.Color]=imread('d:\Ge0MLib_logo3.png'); Head.Wf=[1 0 0 -1 100 50]; Head.K=[1 0];  
Head.BgVal=0;  
>> [Head1,Data1]=gWfrCut(Head,Data,[20 80 10 90]);  
>> gWfrDraw(Head1,Data1,10,2); % Figure 10
```



*Figure 10* gWfrCut example results



## 2.11 Adds/cut borders to image

**function [Head0,Data0]=gWfrAdd(Head,Data,AddLim,BgVal)**

Adds/cut borders for raster-image or matrix-image and correct world-data; borders value set in pixels.

Parameters:

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

$$Z=\text{DataOriginalValue}=a*\text{Color}+b;$$

Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Data – raster-image or matrix-image data matrix;

AddLim – adds for image border in pixels [addX\_left addX\_right addY\_up addY\_down] in pixels; positive values are added, negative - are cut;

BgVal – forced code of “absent” color; if BgVal==[], then used Head.BgVal; it can set to NaN (for matrix-image with type double);

Math:

Line1\_A: pixel size in the x-direction in map units;

Line2\_D: rotation (skew) parameter about y-axis;

Line3\_B: rotation (skew) parameter about x-axis;

Line4\_E: pixel size in the y-direction in map units, almost always NEGATIVE;

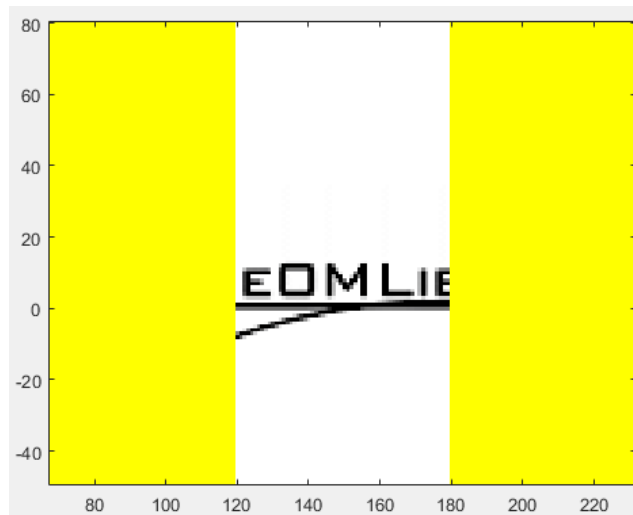
Line5\_C: x-coordinate of center of upper left pixel;

Line6\_F: y-coordinate of center of upper left pixel.

$$x'=Ax+By+C; y'=Dx+Ey+F.$$

Function Example:

```
>> [Data,Head.Color]=imread('d:\Ge0MLib_logo3.png'); Head.Wf=[1 0 0 -1 100 50]; Head.K=[1 0];  
    Head.BgVal=0;  
>> [Head1,Data1]= gWfrAdd(Head,Data,[-20 -20 30 0],0);  
>> gWfrDraw(Head1,Data1,10,2); % Figure 11
```



*Figure 11* gWfrAdd example results

## 2.12 Resize image

**function [Head,Data]=gWfrResize(Head,Data,divXY,rFl)**

Resize raster-image or matrix-image with world-data correction.

Parameters:

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

$DataOriginalValue = a * Color + b$ ;

Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Data – raster-image or matrix-image data matrix.

divXY – divider [divX divY] for [scaleX skewY skewX scaleY];

rFl – resize method name: 1) repeat or delete values - 'Wfr'; 2) use "imresize" function from Image Processing Toolbox with methods 'nearest', 'bilinear', 'bicubic', 'box', 'triangle', 'cubic', 'lanczos2', 'lanczos3'.

Function Example 1:

```
>> [Head1,Data1]=gWfrResize(Head,Data,[2 2], 'Wfr');
```

Function Example 2:

```
>> [Head2,Data2]=gWfrResize(Head,Data,Head0.Wf([1 4])./Head.Wf([1 4]),'Wfr');
```

Function Example 3:

```
>> [Data,Head.Color]=imread('d:\Ge0MLib_logo3.png'); Head.Wf=[1 0 0 -1 100 50]; Head.K=[1 0];
Head.BgVal=0;
```

```
>> [Head3,Data3]=gWfrResize(Head,Data,[0.5 0.5], 'Wfr');
```

```
>> gWfrDraw(Head3,Data3,10,2); % Figure 12, a
```

```
>> [Head3,Data3]=gWfrResize(Head,Data,[2 2], 'Wfr');
```

```

>> gWfrDraw(Head3,Data3,11,2); % Figure 12, b
>> [Head3,Data3]=gWfrResize(Head,Data,[2 2],'bicubic');
>> gWfrDraw(Head3,Data3,11,2); % Figure 12, c
>> [Head3,Data3]=gWfrResize(Head,Data,[2 2],'box');
>> gWfrDraw(Head3,Data3,11,2); % Figure 12, d

```

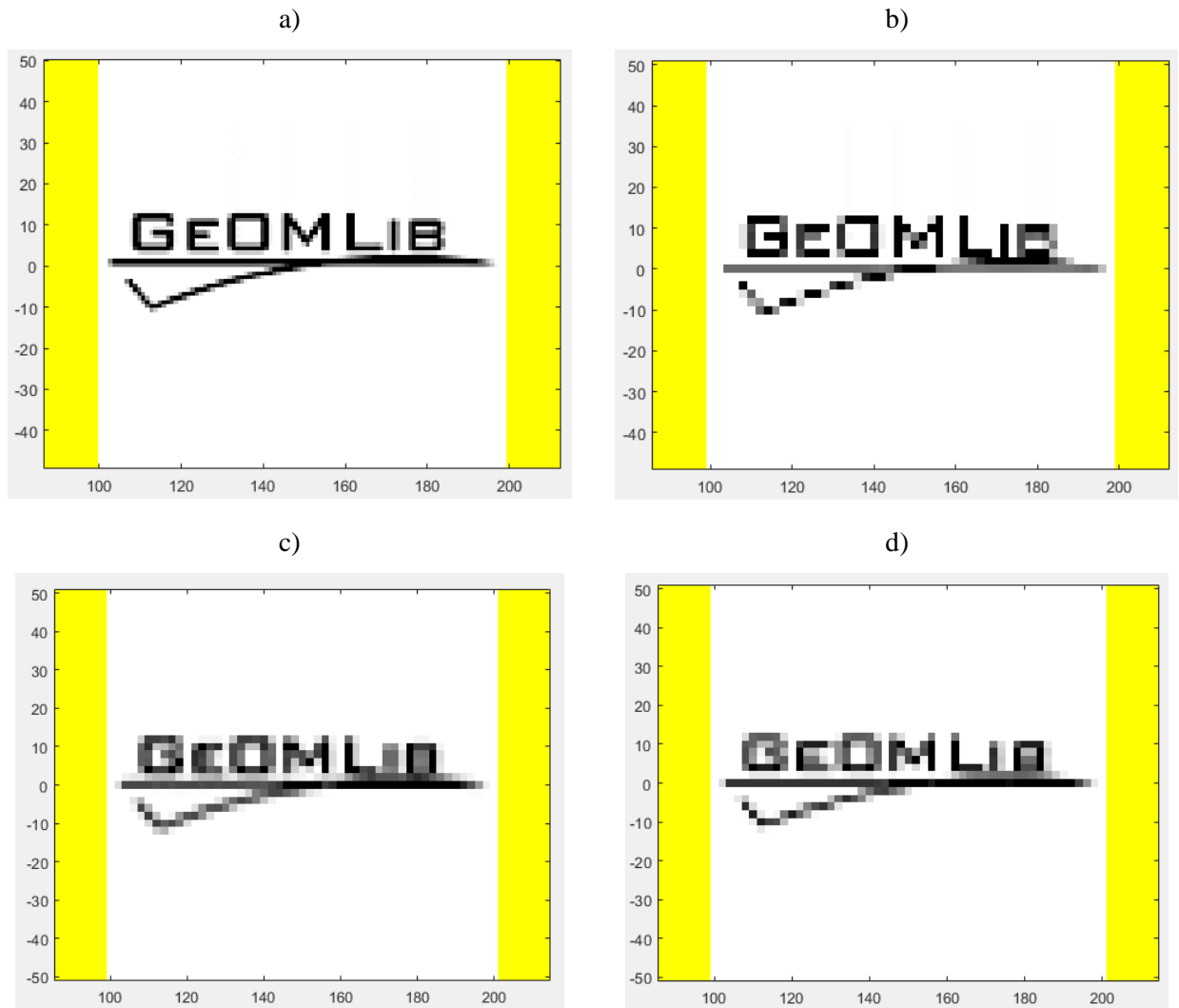


Figure 12 gWfrResize example results

### 2.13 Combine colors/palettes for raster-image1 and raster-image2

```

function [Head1n,Data1n,Head2n,Data2n]=gWfrCombineColors(Head1,Data1,Head2,Data2,
MinMax,imType)

```

Combine colors/palettes for raster-image1 and raster-image2 using Head.K field (renew colors/raletter are linear).

Parameters:

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX skewY skewX scaleY left\_up\_angle\_X left\_up\_angle\_Y];  
 Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;  
 $Z = \text{DataOriginalValue} = a * \text{Color} + b$ ;  
 Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);  
 Data – raster-image or matrix-image data matrix;  
 Head1,Data1,Head2,Data2 – input Data and Header;  
 Head1n,Data1n,Head2n,Data2n – output Data and Header; Head1n.K==Head2n.K; the Head.BgVal will  
 be set 255 for palette-image and 65535 for grayscale-image; BgVal value for palette is [0 0 0];  
 imType – forced output raster-image type: 'uint8' or 'palette'- palette, 'uint16' or 'grayscale' - grayscale  
 without pallete;  
 MinMax – minimal and maximal values link to level-0 and level-254/65534; if empty, than min and max  
 will calculate from Data.

Algorithm: 1) convert raster-image to matrix-image; 2) find [min max] for matrix-images; 3)  
 convert matrix-image to raster-image.

Function Example 1:

```
>> [Data1,Head1.Color]=imread('d:\Ge0MLib_logo3.png'); Head1.Wf=[1 0 0 -1 100 50]; Head1.K=[1
    0]; Head1.BgVal=0;
>> Data2=Data1;Head2=Head1; Head2.K=[4 0];
>> [Head1n,Data1n,Head2n,Data2n]=gWfrCombineColors(Head1,Data1,Head2,Data2,[],[]);
>> image(Data1n);colormap(Head1n.Color); % Figure 13, a
>> image(Data2n);colormap(Head2n.Color); % Figure 13, b
>> Head1n
Head1n =
    Wf: [1 0 0 -1 100 50]
    Color: [256x3 double]
    K: [3.9843137254902 0]
    BgVal: 255
>> Head2n
Head2n =
    Wf: [1 0 0 -1 100 50]
    Color: [256x3 double]
    K: [3.9843137254902 0]
    BgVal: 255
```

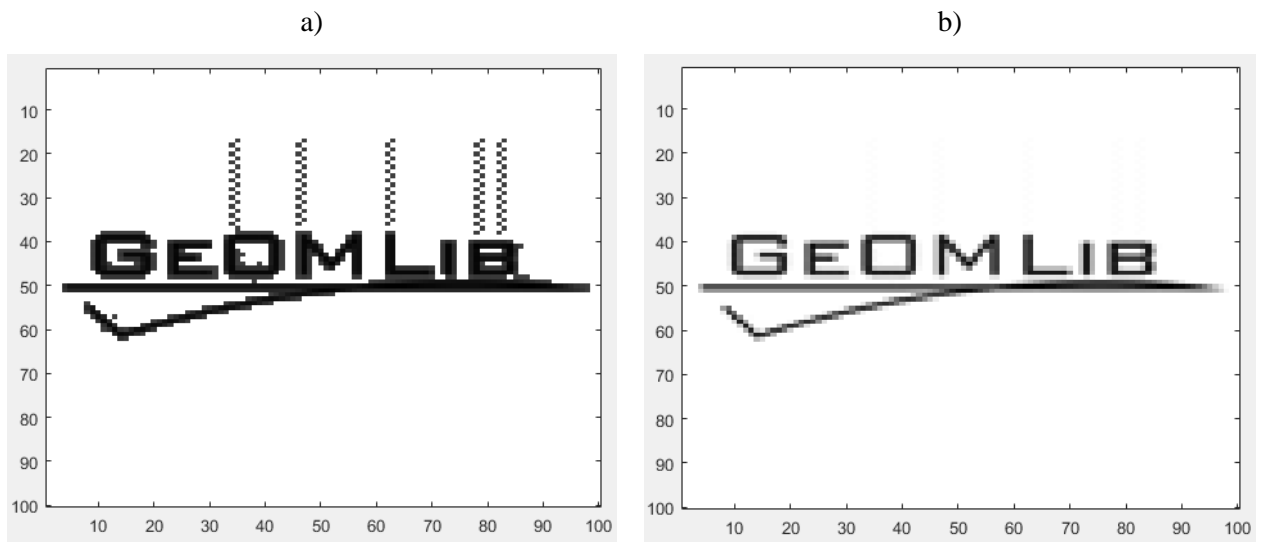


Figure 13 gWfrCombineColors example results

## 2.14 Merge Image1 and Image2 using world-data

**function [Head3,Data3]= gWfrMergeData(Head1,Data1,BgVal1,Head2,Data2,BgVal2,IntKey)**

Merge Image1 and Image2 using world-data. The features: a) Image2 is cover up Image1; b) world file's Line2\_D (skewX) and Line3\_B (skewY) must be zero.

Parameters:

Head – header structure, which includes:

Head.Color – colormap for palette image;

Head.Wf – world-file values: [scaleX 0 0 scaleY left\_up\_angle\_X left\_up\_angle\_Y];

Head.K=[a b] – multiple (a) and shift (b) for "Data Original Value" calculation from Color;

$DataOriginalValue = a * Color + b$ ;

Head.BgVal – the code of “absent” color (it is means – no data for raster’s pixel with BgVal code);

Data – raster-image or matrix image;

Head1,Data1 – input underlying images;

Head2,Data2 – input overlying images;

BgVal1, BgVal2 – forced code of “absent” color for Image1 and Image2; the OutputImage.BgVal=BgVal1; if empty, then Head.BgVal values are used;

IntKey – key for colormap correction; if 0 than OutputImage.Color palette and OutputImage.K will be copied from Head1.Color.

Head3,Data3 – output merged image.

Additional features.

0) SkewX and skewY are zero. **Will be changed in future.**

1) If IntKey==1:

-- try to Combine Colors using Head1.K and Head2.K values;

-- else try to find overlap area for Image1 and Image2 and calculate Head.K.

2) Means that palette for palette-image is gray - for calculations used Head.Color(:,1) column only.

3) Means that Head1.Wf(1)==Head2.Wf(1), Head1.Wf(4)==Head2.Wf(4); the  
OutputImage.Wf(1)=Head1.Wf(1), OutputImage.Wf(4)=Head1.Wf(4).

Function Example 1:

```
>> [Head3,Data3]=gWfrMergeData(Head1,Data1,[],Head2,Data2,[],0);
```

Function Example 2:

```
>> [Data1,Head1.Color]=imread('d:\Ge0MLib_logo3.png'); Head1.Wf=[1 0 0 -1 100 50]; Head1.K=[1  
0]; Head1.BgVal=0;
```

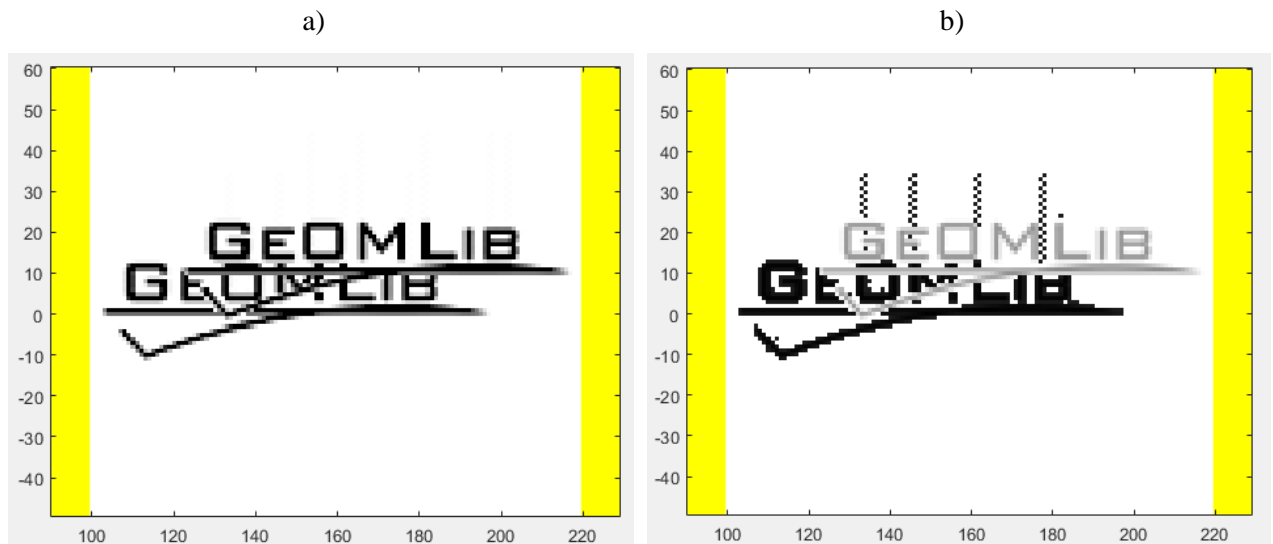
```
>> Data2=Data1;Head2=Head1; Head2.K=[4 5]; Head2.Wf=[1 0 0 -1 120 60];
```

```
>> [Head3,Data3]=gWfrMergeData(Head1,Data1,[],Head2,Data2,[],0);
```

```
>> gWfrDraw(Head3,Data3,10,2); % Figure 14, a
```

```
>> [Head3,Data3]=gWfrMergeData(Head1,Data1,[],Head2,Data2,[],1);
```

```
>> gWfrDraw(Head3,Data3,10,2); % Figure 14, b
```



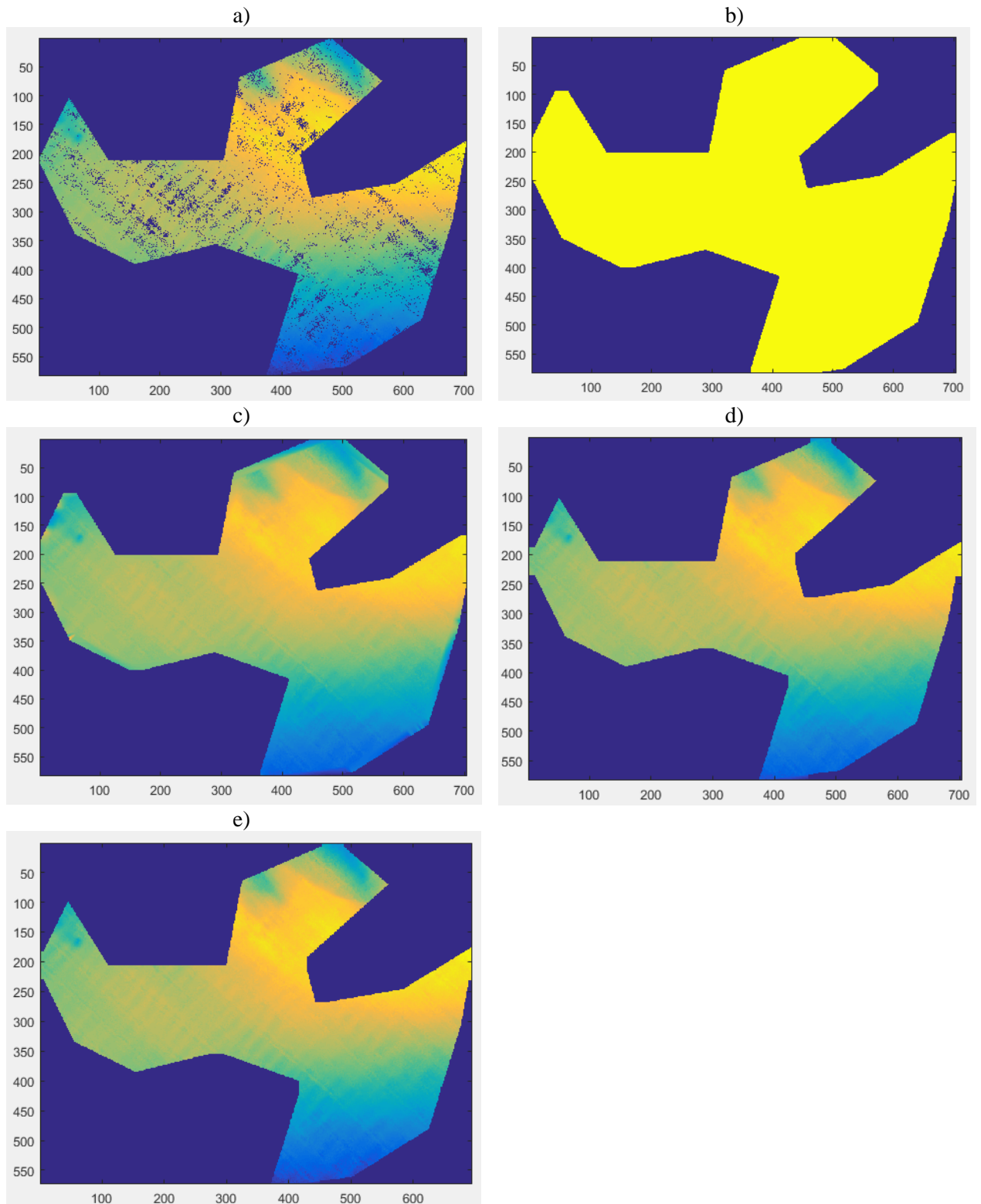
*Figure 14* gWfrMergeData example results

### 3 Example

```
%script gPts2DFilt
%2D filtration for pts-file (XYZ bathy data); warning! the skewX and skewY are zero for pts-file.
%Coordinate axis:
%^+Y (N)
%|
%|
%----> +X (E)

%====Read pts-file and convert it to matrix-image=====
XYZ=dlmread('d:\Reglam\For_ID.txt');
dE=[####47.875 ####026.875 ####48.125 ####026.875]; %dE=[lx_x1 lx_y1 lx_x2 lx_y2] is horizontal segment along X axis;
dN=[####48.125 ####026.875 ####48.125 ####027.125]; %dN=[ly_x1 ly_y1 ly_x2 ly_y2] is vertical segment along Y axis;
[Head,Data]=gWfrXyz2Mat(XYZ',dE,dN); %Data is "matrix-image" contane grid
gWfrDraw(Head,Data,1,1); % Figure 15, a
%====Find holes and prepare mask=====
B=uint8(~isnan(Data)); %convert to black&white image;
B=bwmorph(B,'close'); %use Image Processing Toolbox -- fill small "holes";
B=bwmorph(B,'dilate',10); %use Image Processing Toolbox -- add 10 pixels to black-area borders
figure(2);imagesc(B); % Figure 15, b
%====Prepare triangulation web=====
dx=dE(3)-dE(1);dy=dN(4)-dN(2); %use [stepX 0 0 stepY left_up_angle_X left_up_angle_Y]
Bo=scatteredInterpolant(round((XYZ(:,1)-min(XYZ(:,1)))/dx)+1,round((max(XYZ(:,2))-XYZ(:,2))/dy)+1,XYZ(:,3),'natural','linear');
%====Interpolate data for holes=====
L=isnan(Data);
[L2,L1]=find((B==1)&L);L3=(B==1)&L;
Data2=Data;Data2(L3)=Bo(L1,L2);
gWfrDraw(Head,Data2,3,1); % Figure 15, c
%====2D filtration=====
F=zeros(21,21);F(11,11)=1;%create emulation of filter
Data3=gData2DFilt(Data2,F,1); %make filtration with normalization
gWfrDraw(Head,Data3,4,1); % Figure 15, d
[Head2,Data4]=gWfrAdd(Head,Data3,[-5 -5 -5 -5],nan);%cut 5-pixels borber
gWfrDraw(Head2,Data4,5,1); % Figure 15, e
%====Write new pts-file=====
XYZ2=gWfrIm2Xyz(Head2,Data4,nan,[1 0],1);
dlmwrite('d:\Reglam\For_ID2.txt',XYZ2','precision','%.3f');

%mail@ge0mlib.com 19/06/2018
```



*Figure 15* gPts2DFilt script's steps

- a) original pts-file converted to matrix-image (grid);
- b) mask for holes interpolation;
- c) interpolated matrix-image;
- d) filtered matrix-image, using 2D filter;
- e) matrix-image with 5-pixels cut.



## Citation

- 1) <https://support.esri.com/en/technical-article/000002860>
- 2) <http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/world-files-for-raster-datasets.htm>
- 3) [https://en.wikipedia.org/wiki/World\\_file](https://en.wikipedia.org/wiki/World_file)
- 4) <https://www.awaresystems.be/imaging/tiff/tifftags.html>;
- 5) <https://www.loc.gov/preservation/digital/formats/fdd/fdd000279.shtml>