University of Ljubljana, Faculty of Electrical Engineering

Laboratory of Imaging Technologies

Medical Informatics and Diagnostics

LABORATORY PRACTICE

Tomaž Vrtovec

2010

University of Ljubljana, Faculty of Electrical Engineering

Laboratory of Imaging Technologies

Medical Informatics and Diagnostics

LABORATORY PRACTICE

Tomaž Vrtovec

2010

Contents

Lab Work 0: Introduction to Matlab 1

Lab Work 1: Signal-to-noise ratio 4

Lab Work 2: Image synthesis and masking 7

Lab Work 3: Image visualization 11

Lab Work 4: Image interpolation and decimation 14

Lab Work 5: Image grayscale transformations 18

Lab Work 6: Image geometrical transformations 21

Lab Work 7: Image spatial filtering 26

Lab Work 8: Image geometrical registration (1) 30

Lab Work 9: Image geometrical registration (2) 35

Lab Work 10: Image segmentation 39

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 0: Introduction to Matlab

Navodila

This introductory lab work should serve to acquire basic skills for loading, displaying and saving images in the Matlab environment.

1. Use the function `imread()` to load the bitmap image slika1.bmp as a matrix. Use the function `imshow()` to display the image on screen. Use the function `imwrite()` to save the image into JPEG format with quality 10 (instead of the default 75).

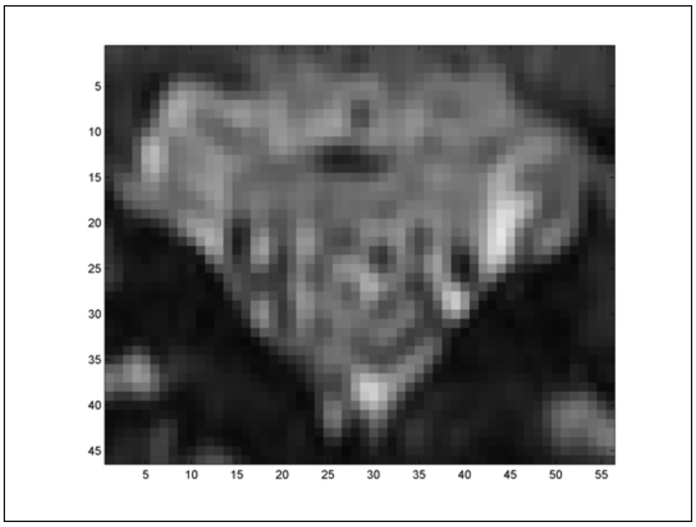
2. The file slika2-08bit.raw contains a two-dimensional (2D) image in raw data format (RAW). The image of size $X \times Y = 482 \times 384$ picture elements is stored with 8 bits per picture element. Load the image as a matrix by using the function `fread()` in combination with functions `fopen()` and `fclose()`.

3. Display the image on screen by using the function `image()`. Adjust the display to the format by using the function `colormap()`.

4. The file slika3-16bit.raw contains a 2D image in raw data format (RAW), however, the image of size $X \times Y = 478 \times 387$ picture elements is stored with 16 bits per picture element. Load the image as a matrix and display it on screen.

5. Save the 16-bit image as slika3-08bit.raw in 8-bit raw data format (RAW) by using the function `fwrite()` in combination with functions `fopen()` and `fclose()`.

6. Extract the area between picture elements 210 and 265 in x direction and between picture elements 180 and 225 in y direction from the image. Display the extracted area on screen and verify the extraction procedure by comparing the results to the image below.



7. Write the function for loading an arbitrary RAW image:

   ```
   function oImage = loadImage(iPath, iDim, iType),
   ```

   where `iPath` is the path to the image (directory and filename), `iDim` is the vector of image size (in picture elements), and `iType` is the image format (number of bits). The function returns the loaded image `oImage` as a matrix.

8. Write the function for displaying an arbitrary RAW image on screen:

   ```
   function [hFigure, hAxes] = displayImage(iImage, iTitle),
   ```

   where `iImage` is the image to be displayed and `iTitle` is the title of the display window that contains the coordinate system of the displayed image. The function returns the handle `hFigure` to the display window and the handle `hAxes` to the coordinate system.

9. Write the function for saving an arbitrary RAW image:

   ```
   function saveImage(iImage, iPath, iType),
   ```

   where `iImage` is the image to be saved, `iPath` is the path to the image (directory and filename), and `iType` is the image format (number of bits).

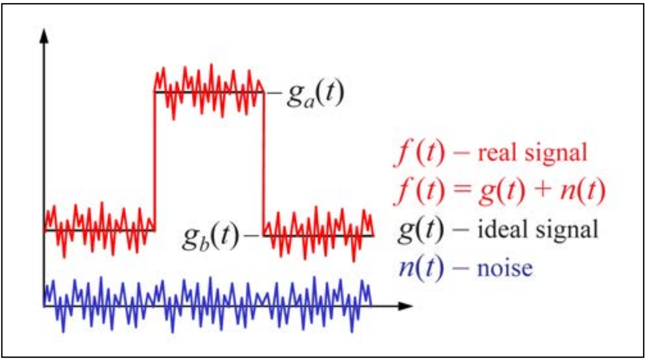
10. Test the performance of the functions `loadImage()`, `displayImage()` and `saveImage()` by using the image slika4-08bit.raw of size $X \times Y = 83 \times 100$ picture elements in 8-bit format.

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

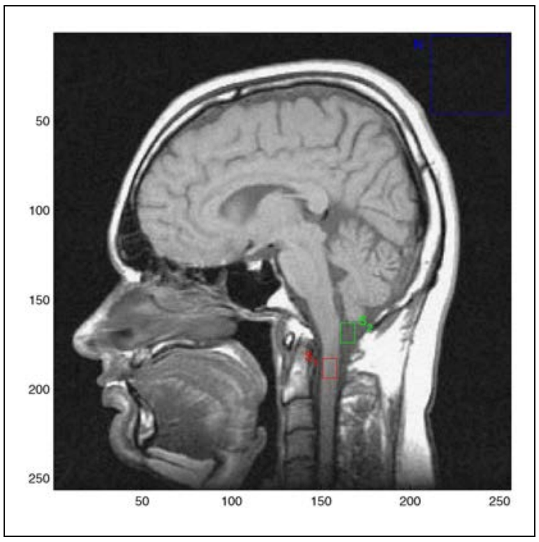
Lab Work 1: Signal-to-noise ratio

Navodila

The signal-to-noise ratio (SNR) is important for the evaluation of the strength (power) of the signal relative to the noise present in the signal. SNR is a measure of reliability or ability to detect the presence or changes in physical and biochemical properties that are measured by the observed signal. There are three basic definitions for the SNR, i.e. the amplitude (SNR_A), differential (SNR_D) and power (SNR_M) definition. Reporting the SNR is, however, not standardized but depends on the type of signal and noise, and on the area of application.



The image head-256x256-08bit.raw of size $X \times Y = 256 \times 256$ picture elements is stored with 8 in raw data format (RAW). The image represents a two-dimensional (2D) magnetic resonance (MR) lateral cross-section of the human head.



1. Load the image in Matlab as a 2D matrix and display it on screen, where you take into account the correct scale of grayscale values. Use the functions `fread()`, `image()` and `colormap()`.

2. The image histogram is a graphical tool to visualize the frequency distribution of the grayscale values of the picture elements in the image. The histogram abscissa values represent the dynamic range of the grayscale values. For the chosen grayscale value on the abscissa, the corresponding histogram ordinate value represents the number of picture elements in the image with the chosen grayscale value. Write the function for computing and displaying the histogram of an image:

   ```
   function oHist = displayHistogram(iImage, iTitle),
   ```

   where `iImage` is the image for which the histogram is computed, and `iTitle` is the title of the histogram figure. The function returns the image histogram `oHist` in the form of a single row vector. The length of the row vector is equal to the dynamic range of the grayscale values. Compute the image histogram without using Matlab's advanced functions, whereas for visualization use the function `bar()`.

3. The image can be divided into regions that belong to different anatomical structures or areas:

   - The spinal cord represents the region of the first signal S_1. It is present, for example, within area $(x_1, y_1, x_2, y_2) = (151, 183, 158, 193)$.
   - The cerebrospinal fluid is the liquid that surrounds the brain and the spinal cord, and it represents the region of the second signal S_2. It is present, for example, within area $(x_1, y_1, x_2, y_2) = (161, 163, 168, 173)$.
   - The image background does not belong to any anatomical structure, but is in the ideal case a region without a response and therefore represents the noise N. It is present, for example, within area $(x_1, y_1, x_2, y_2) = (212, 2, 254, 45)$.

   Determine the given image regions of singal and noise, display their images and the corresponding histograms.

4. Compute the amplitude signal-to-noise ratio SNR_A for the given image regions:

$$\mathrm{SNR}_A = \left.\frac{\mu_a}{\sigma_n}\right|_{\mu_b=0},$$

   where μ_a is the mean amplitude of the upper signal level, μ_b is the mean amplitude of the lower signal level, and σ_n is the standard deviation of the noise. In this case take into account that the mean amplitude of the lower signal level is equal to zero ($\mu_b = 0$).

5. Compute the differential signal-to-noise ratio SNR_D for the given image regions, which can be computed according to two different equations:

$$\mathrm{SNR}_D = \frac{\mu_a - \mu_b}{\sigma_n} \qquad \text{in} \qquad \mathrm{SNR}_D = \frac{|\mu_a - \mu_b|}{\sqrt{\sigma_a^2 + \sigma_b^2}},$$

   where σ_a and σ_b are the standard deviations of the upper and lower signal level, respectively (in this case the signal S_1 and the signal S_2).

6. Compute the power signal-to-noise ratio SNR_M for the given image regions:

$$\mathrm{SNR}_M = \frac{|\mu_a - \mu_b|^2}{\sigma_a^2 + \sigma_b^2}.$$

7. Write the function that applys additive Gaussian noise to the image:

```
function [oImage, oNoise] = addNoise(iImage, iStd),
```

where `iImage` is the image to which the Gaussian noise is applied, and `iStd` is the standard deviation of the applied noise (mean amplitude of the applied noise is $\mu = 0$) that is modelled by the function `randn()`. The function returns the noisy image `oImage` and the matrix of the applied noise `oNoise` (the matrix has the same dimensions as the image). Observe the images, the corresponding histograms and the signal-to-noise ratios for images with applied Gaussian noise of varying standard deviation.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Enclose the histogram image of the given 2D image of the head.

2. Enclose the image regions of signals S_1, S_2 and N, and the corresponding histogram images.

3. Compute the amplitude (SNR_A), differential (SNR_D) and power (SNR_M) signal-to-noise ratios for the given image regions of signals and noise. Report the values also in decibels. What is the meaning of these values, if you consider the given computation regions?

4. Which signal-to-noise ratio, SNR_A or SNR_D, is more likely to be computed if we want to observe only the spinal cord, represented by the signal S_1? Explain your answer.

5. What is the recommended size of the region used to evaluate the noise N? Explain your answer.

6. Enclose the image to which noise of standard deviation $\sigma = 25$ was applied and the corresponding histogram image.

7. Enclose the image of the noise of standard deviation $\sigma = 25$ and the corresponding histogram image. What do you have to take into account when displaying the image of the noise and when computing the corresponding histogram? Explain your answer.

8. How does applying noise to the image affects the shape of the corresponding histogram and the differential signal-to-noise ration SNR_D? Explain your answer.

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 2: Image synthesis and masking

Navodila

Image synthesis is, in general, the creation of new images and merging of existing images into new images. The synthesis of two images can be, for example, performed by the logical OR operation, which is defined as the summation of grayscale values at the corresponding picture elements of images (non-logical multiple-bit values) of equal size.

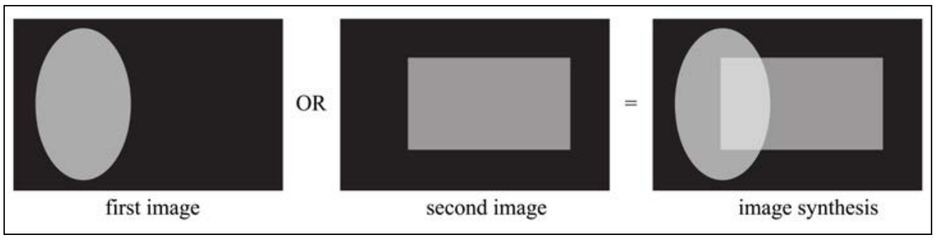
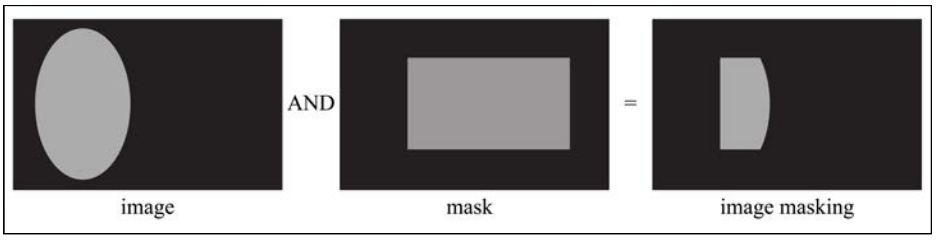
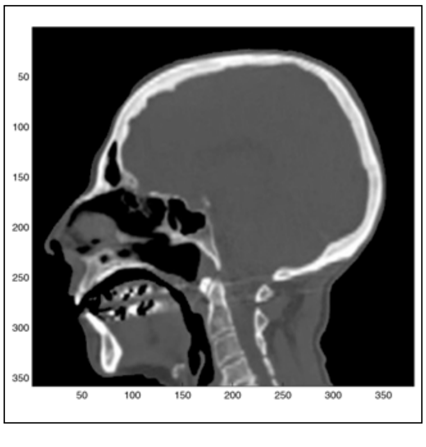
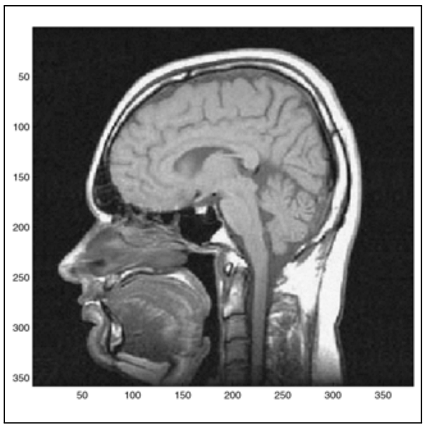


Image masking is a technique for extracting image areas that are defined by the corresponding mask. The mask represents the locations of those picture elements that are retained in the resulting image. The masking of an image with a mask can be, for example, performed by the logical AND operation, which is defined as the multiplication of grayscale values at the corresponding picture elements of the image (non-logical multiple-bit values) and mask (logical bit values) of equal size. The generation of masks can be also considered as image synthesis, as the mask values are stored on chosen locations in an empty image.



The images head-CT-380x358-08bit.raw in head-MR-380x358-08bit.raw of size $X \times Y = 380 \times 358$ picture elements are stored with 8 bits in raw data format (RAW). The first image represents a computed tomography (CT), while the second image represents a magnetic resonance (MR) lateral cross-section of the human head.

1. Write the function for the synthesis of a mask in the shape of a rectangle:

   ```
   function oMask = maskRectangle(iMask, iX1, iY1, iX2, iY2, iValue),
   ```

   where `iMask` is the initial input mask, `iX1` and `iY1` are the coordinates of the upper left rectangle corner, `iX2` and `iY2` are the coordinates of the lower right rectangle corner, and `iValue` is the mask value. The function returns the mask `oMask` of the same size as the input mask. Display the image of the mask on screen by using rectangle parameters $(x_1, y_1) = (165, 150)$ and $(x_2, y_2) = (225, 355)$, and setting the mask value adequately for the display.

2. Perform masking of the given CT image with the mask in the shape of a rectangle. Use the rectangle parameters from the previous task and set the mask value adequately for masking. Display the masked image on screen.

3. Write the function for the synthesis of a mask in the shape of an ellipse:

   ```
   function oMask = maskEllipse(iMask, iX, iY, iA, iB, iValue),
   ```

   where `iMask` is the initial input mask, `iX` and `iY` are the coordinates of the ellipse center, `iA` and `iB` are the ellipse half-axes, and `iValue` is the mask value. The function returns the mask `oMask` of the same size as the input mask. Use the function `round()` to round the coordinates to integer values. Display the image of the mask on screen by using ellipse parameters $(x, y) = (190, 130)$, $a = 120$ and $b = 100$, and setting the mask value adequately for the display.

4. Perform masking of the given MR image with the mask in the shape of an ellipse. Use the ellipse parameters from the previous task and set the mask value adequately for masking. Display the masked image on screen.

5. Merge the mask in the shape of a rectangle and the mask in the shape of an ellipse into a new mask. Use the rectangle parameters from task 1 and ellipse parameters from task 3, and set the mask value adequately for masking. Display the image of the obtained mask on screen.

6. Write the function for the synthesis of a mask in the shape of a chessboard:

   ```
   function oMask = maskChessboard(iMask, iW, iH, iValue1, iValue2),
   ```

   where `iMask` is the initial input mask, `iW` and `iH` are the width and height of the chessboard field, and `iValue1` and `iValue2` are the mask values at forward-diagonal and back-diagonal chessboard fields, respectively. The function returns the mask `oMask` of the same size as the input mask. Use the function `round()` to round the coordinates to integer values. Take into account that the number of chessboard fields does not always match the image size. Display the image of the mask on screen by using chessboard parameters $w = 60$ and $h = 40$, and setting the mask values adequately for the display.

7. Perform masking of the given CT image with the mask in the shape of a chessboard by using only the forward-diagonal fields. Perform masking of the given MR image with the mask in the shape of a chessboard by using only the back-diagonal fields. Use the chessboard parameters from the previous task and set the mask values adequately for masking. Display the masked images on screen.

8. Merge the CT image, masked with the chessboard mask of forward-diagonal fields, and the MR image, masked with the chessboard mask of back-diagonal fields, into a new image. Display the obtained image on screen.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Enclose the image obtained by masking the given CT image with a mask in the shape of a rectangle by using rectangle parameters $(x_1, y_1) = (165, 150)$ and $(x_2, y_2) = (225, 355)$. Enclose the image obtained by masking the given MR image with a mask in the shape of an ellipse by using ellipse parameters $(x, y) = (190, 130)$, $a = 120$ and $b = 100$.

2. Enclose the image of the mask obtained by merging the mask in the shape of a rectangle and the mask in the shape of an ellipse. Use the rectangle and ellipse parameters from the previous question. Enclose the image obtained by masking the given CT image with the merged mask.

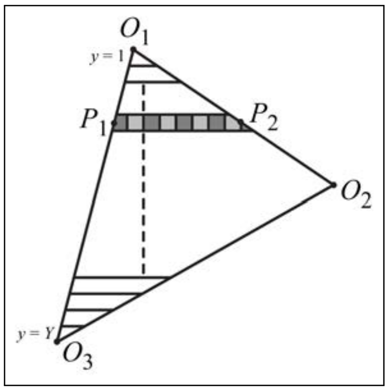
3. Enclose the image obtained by merging the given CT image, masked with a chessboard of forward-diagonal fields, and the given MR image, masked with a chessboard of back-diagonal fields. Use chessboard parameters $w = 60$ and $h = 40$ for both masks.

4. The synthesis of the masked CT and the masked MR image in the form of a complementary chessboard can be used to display the differences between the two images. Are there any other ways to show the difference between the two images and what has to be taking into account?

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

The general shape of a mask with straight sides is a polygon with an arbitrary number of corners, which can be represented by a group of adjacent triangles. The triangle therefore represents the elementary construct of an arbitrary mask with straight sides. Because of the discrete representation of images (picture elements), the most established technique for generating triangles is the "scan-line" algorithm. The algorithm is based on searching the intersections of each triangle row y with the triangle sides, and setting the mask value to all the picture elements found between both intersections in the x direction.



Write the function for the synthesis of a mask in the shape of a triangle:

```
function oMask = maskTriangle(iMask, iX, iY, iValue),
```

where `iMask` is the initial input mask, `iX` $= [x_1, x_2, x_3]$ and `iY` $= [y_1, y_2, y_3]$ are the coordinates of the triangle corners, and `iValue` is the mask value. The function returns the mask `oMask` of the same size as the input mask. Use the function `round()` to round the coordinates to integer values. Display the image of the mask on screen by using triangle parameters $(x_1, y_1) = (230, 20)$, $(x_2, y_2) = (50, 250)$ and $(x_3, y_3) = (300, 340)$, and setting the mask values adequately for the display.

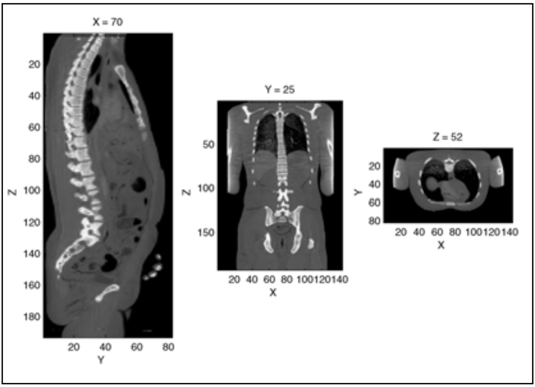
University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 3: Image visualization

Navodila

The objects that are observed in images (e.g. anatomical structures) are in general three-dimensional and therefore have to observed and analyzed in three-dimensional (3D) images, represented as functions $f(x, y, z)$ of the spatial coordinates x, y and z. The cross-sections and projections enable image visualization in less-dimensional spaces. The main difference between cross-sections and projections is that for cross-sections, only the domain of the projection is defined, whereas for the projections, the values do not represent just the original grayscale values, but are obtained from an arbitrary function of all picture elements defined on the cross-section domain. Cross-sections are therefore a specific sub-group of projections.

The image vh-143x082x193-08bit.raw of size $X{\times}Y{\times}Z = 143{\times}82{\times}193$ picture elements is stored with 8 bits in raw data format (RAW). The image was acquired by computed tomography (CT) and represents the middle part of the human body in 3D. The image vh-071x041x096-08bit.raw of size $X{\times}Y{\times}Z = 71{\times}41{\times}96$ represents the exactly same structures as the first image but is of smaller size, which can be of great help when developing the algorithms, as data loading and processing is much faster.



1. Write the function for loading a 3D image:

   ```
   function oImage = loadImage3D(iPath, iDim, iType, iMode),
   ```

   where `iPath` is the path to the image (directory and filename), `iDim` is the vector of image size (in picture elements), `iType` is the image format (number of bits), and `iMode` is the loading mode as follows:

   - `iMode` = 1: The 3D matrix is obtained by sequential loading of two-dimensional (2D) matrices that are stacked into a 3D matrix,
   - `iMode` = 2: The 3D matrix is obtained by a single loading of a one-dimensional (1D) vector that is reshaped into a 3D matrix by using the functions `permute()` and `reshape()`.

   The function returns the loaded image `oImage` as a 3D matrix.

2. On the basis of the 3D image we can extract basic two-dimensional (2D) planar cross-sections. For the 3D image $f(x, y, z)$, the function $f_{x=x_c}(y, z) = f(x_c, y, z)$ defines the lateral (sagittal) cross-section, the function $f_{y=y_c}(x, z) = f(x, y_c, z)$ defines the frontal (coronal) cross-section, and the function $f_{z=z_c}(x, y) = f(x, y, z_c)$ defines the transverse (axial) cross-section. Write the function for extracting an arbitrary basic planar cross-section:

```
function oSection = getCrossSection(iImage, iPlane, iNum),
```

where `iImage` is the 3D image from which the 2D cross-section is extracted, `iPlane` is the cross-sectional plane (lateral, frontal or transverse), and `iNum` is the number of the cross-section. The function returns the basic planar cross-section `oSection`. Display on screen the image of the obtained basic planar cross-section.

3. Extract every basic planar cross-sections between two chosen cross-sections and display them on screen in a time sequence (video). For pausing the display use the function `pause()`. Display on screen the video of all lateral basic planar cross-sections between number $x_1 = 40$ and number $x_2 = 100$.

4. On the basis of the 3D image we can extract different orthogonal projections. For the 3D image $f(x, y, z)$, the function $f_{\max}(y, z) = \max_{k=1...X} (f(k, y, z))$ defines the lateral maximum intensity projection (MIP), the function $f_{\max}(x, z) = \max_{k=1...Y} (f(x, k, z))$ defines the frontal MIP, and the function $f_{\max}(x, y) = \max_{k=1...Z} (f(x, y, k))$ defines the transverse MIP. Similarly we can define also other projections:

   - average intensity projection (AvgIP),
   - minimal intensity projection (MinIP),
   - median intensity projection (MedIP),
   - standard deviation intensity projection (StdIP),
   - variance intensity projection (VarIP).

   Write the function for extracting an arbitrary orthogonal projection:

   ```
   function oProjection = getOrthogonalProjection(iImage, iPlane, iFunc),
   ```

   where `iImage` is the 3D image from which the 2D projection is extracted, `iPlane` is the projection plane (lateral, frontal or transverse), and `iFunc` is the projection type (MIP, AvgIP, MinIP, MedIP, StdIP, VarIP). The function returns the orthogonal projection `oProjection`. Display on screen the image of the obtained orthogonal projection.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Enclose the images of the following basic 2D planar cross-sections of the given 3D image:

   - lateral cross-section for the picture element $x_c = 76$,
   - frontal cross-section for the picture element $y_c = 31$,
   - transverse cross-section for the picture element $z_c = 102$.

2. Enclose the images of the lateral, frontal and transverse orthogonal projections for the:
    - maximum intenisty projection (MIP),
    - average intenisty projection (AvgIP).

3. Which of the orthogonal projections among MIP, AvgIP, MinIP, MedIP, StdIP in VarIP is reasonable to extract for the given example of CT image visualization? Explain your answer.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Surface rendering is a group of techniques for visualizing the surface of a given 3D object or structure. One of the most established surface rendering technique is the iso-surface. First, the 3D image is segmented into regions by advanced segmentation techniques or by simple thresholding of grayscale values. Second, the iso-surface function is applied to connect all points in space with the same iso-value (threshold value). Finally, the surface is represented as a set of surface points (vertices) and a set of surface triangles (faces), which connect the surface points. By using the function `surfaceRendering()` that implements the iso-surface technique, try to render the surface in the given 3D image `iImage` for different thresholds of grayscale values `iThreshold`:

```
>>  % surface rendering by iso-surface
>>  function pHandle = surfaceRendering(iImage, iThreshold)
>>      % compute the iso-surface
>>      oPatch = isosurface(iImage, iThreshold);
>>      % display the iso-surface
>>      pHandle = patch(oPatch, 'FaceColor', 'red', 'EdgeColor', 'none');
>>      % set the coordinate axes
>>      daspect([1 1 1]); axis tight;
>>      set(gca, 'YDir', 'reverse', 'ZDir', 'reverse');
>>      % set the lighting
>>      isonormals(iImage, pHandle); camlight; lighting gouraud;
>>      % set the view
>>      view(3);
```

1. Which threshold gives satisfying results?
2. Can you think of a simple algorithm to define the optimal threshold?
3. How can be the quality of the given surface rendering improved?

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 4: Image interpolation and decimation

Navodila

Image interpolation is a procedure that increases the number of picture elements of an image and enables the determination of grayscale values at arbitrary locations between the sampling points. **Image decimation** is a procedure that decreases the number of picture elements of an image by sampling the image with a lower frequency.

Test the techniques for image interpolation and decimation on image head-256x256-08bit.raw of size $X \times Y = 256 \times 256$ picture elements and stored with 8 bits in raw image format (RAW). The image represents a two-dimensional (2D) magnetic resonance (MR) lateral cross-section of the human head.

1. The zero-order interpolation in 2D (nearest neighbor interpolation) determines the grayscale value in point (x, y) as the grayscale value of its nearest point on the discrete grid of sampling points. Write the function for the zero-order interpolation (oversampling) of a 2D image:

```
function oImage = interpolate0Image2D(iImage, iCoor, iSize, oSize),
```

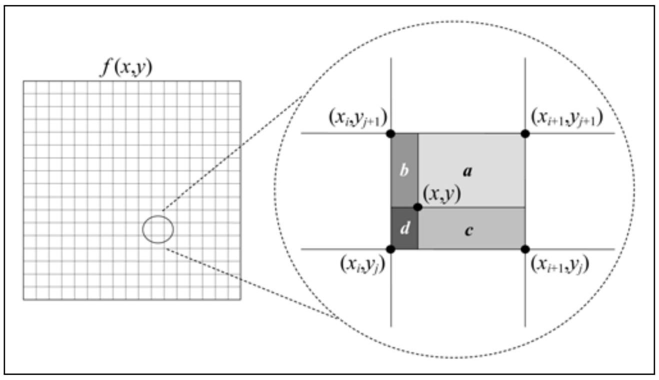
   where `iImage` is the 3D image that is interpolated, `iCoor` $= [x, y]$ are the coordinates of the upper left corner of the interpolation region, `iSize` $= [x, y]$ is the size of the interpolation region, and `oSize` $= [x, y]$ is the size of the output image. The function returns the interpolated image `oImage` (region) as a matrix. Use the function `round()` to define the nearest point on the discrete grid of sampling points.

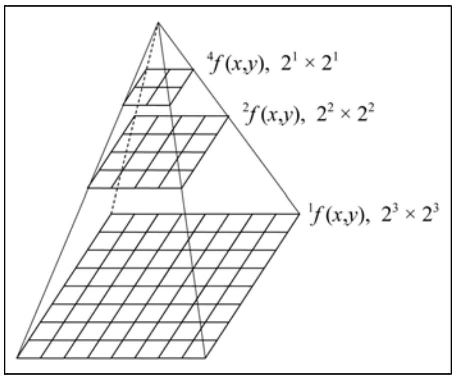
2. The first-order interpolation in 2D (bilinear interpolation) determines the grayscale value in point (x, y) as a weighted sum of grayscale values of its four neighboring points on the discrete grid of sampling points:

$$f(x,y) = af(x_i, y_j) + bf(x_{i+1}, y_j) + cf(x_i, y_{j+1}) + d(x_{i+1}, y_{j+1}),$$

   where $f(x, y)$ is the grayscale value at point (x, y). The weight of a chosen neighboring sampling point is proportional to the area of the rectangle defined by the point (x, y), for which the grayscale value is computed, and the point that is diagonally opposite to the point (x, y), for example:

$$a = \frac{(x_{i+1} - x)(y_{j+1} - y)}{(x_{i+1} - x_i)(y_{j+1} - y_j)}.$$





${}^4f(x,y),\ 2^1\times 2^1$
${}^2f(x,y),\ 2^2\times 2^2$
${}^1f(x,y),\ 2^3\times 2^3$

Write the function for the first-order interpolation (oversampling) of a 2D image:

```
function oImage = interpolate1Image2D(iImage, iCoor, iSize, oSize),
```

where the input and output parameters are the same as for the zero-order interpolation function. Use the function `floor()` to define the nearest low point on the discrete grid of points.

3. Image decimation is defined as the sampling of the image with a lower sampling frequency. By considering the Nyquist theorem, the input image bandwidth has to be reduced before sampling and can be achieved by using a low-bandwidth digital filter. The filtered and by 2 decimated image on level $2n$ is obtained by the convolution of the image from the previous level n with the kernel $c(i,j)$ of the digital filter of size $M\times M$:

$$ {}^{2n}f(x,y) = \sum_{i=-M}^{i=M}\sum_{j=-M}^{j=M} c(i,j)\cdot {}^{n}f(2x-i,2y-j)\,. $$

The size of the decimated image is therefore reduced according to the decimation level. Multi-level decimation can be represented in the shape of a pyramid as a sequence of decimations by 2. The kernel $c(i,j)$ of the digital filter that satisfies the conditions is in 2D equal to:

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

$M = 1$

$\frac{1}{400}$	$\frac{1}{80}$	$\frac{1}{50}$	$\frac{1}{80}$	$\frac{1}{400}$
$\frac{1}{80}$	$\frac{1}{16}$	$\frac{1}{10}$	$\frac{1}{16}$	$\frac{1}{80}$
$\frac{1}{50}$	$\frac{1}{10}$	$\frac{4}{25}$	$\frac{1}{10}$	$\frac{1}{50}$
$\frac{1}{80}$	$\frac{1}{16}$	$\frac{1}{10}$	$\frac{1}{16}$	$\frac{1}{80}$
$\frac{1}{400}$	$\frac{1}{80}$	$\frac{1}{50}$	$\frac{1}{80}$	$\frac{1}{400}$

$M = 2$

Write the function for the decimation (subsampling) of a 2D image:

```
function oImage = decimateImage2D(iImage, iKernel, iLevel),
```

where `iImage` is the 3D image that is decimated, `iKernel` $= c(i,j)$ is the kernel of the digital filter of size $M \times M$, and `iLevel` is the integer decimation level. The function returns the decimated image `oImage` as a matrix. Decimate the given image with the kernel $c(i,j)$ of the digital filter of size $M = 1$ and $M = 2$ using an arbitrary decimation level.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Use the following parameter values for interpolating the given 2D image: `iCoor` $= [160, 95]$, `iSize` $= [45, 60]$ and `oSize` $= [200, 200]$. Enclose the image of the interpolation region in the original image and the interpolated image, obtained by zero-order interpolation and by first-order interpolation.

2. What are the advantages and what are the disadvantages of the zero-order interpolation? What are the advantages and what are the disadvantages of the first-order interpolation? What can we achieve by interpolations of higher orders, for example with the second-order interpolation (bicubic interpolation)?

3. List a few examples where efficient image interpolation is needed.

4. Use the following parameter values for decimating the given 2D image: `iKernel` $\rightarrow M = 1$ and `iLevel` $= 3$. Enclose the image obtained by such decimation.

5. What are the advantages and what are the disadvantages of decimation? By what factor is the image size reduced when decimated? By what factor is the amount of data reduced?

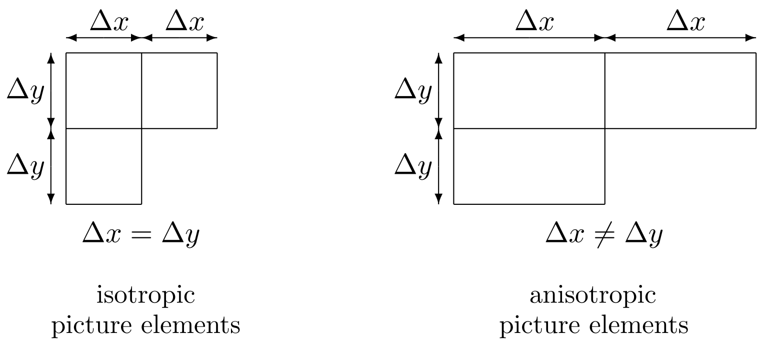
6. What conditions must be fulfilled for the kernel coefficients $c(i,j)$ of the digital filter used for image decimation? If you wanted to decimate the image simply by sampling every second picture element without reducing the image bandwidth, what would in such case be the shape and the values of the kernel $c(i,j)$ of the digital filter?

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

1. The first-order interpolation in 3D (trilinear interpolation) determines the grayscale value in point (x, y, z) as a weighted sum of grayscale values of its eight neighboring points on the discrete grid of sampling points. The weight of a chosen neighboring sampling point is proportional to the volume of the block defined by the point (x, y, z), for which the grayscale value is computed, and the point that is diagonally opposite to the point (x, y, z). Write the function `interpolate1Image3D()` for the first-order interpolation of a 3D image (e.g. the image from Lab Work 3).

2. In all of the presented problems, the picture elements were assumed to be isotropic, which means that the dimensions of the picture element were equal (uniform) for all coordinate directions. However, the picture elements are in general anisotropic, which means that their dimensions are not equal for all coordinate directions. Because the picture elements are the result of an arbitrary transform from the real world to the image space, the dimensions of picture elements are measured in units $\frac{\text{mm}}{\text{pixel}}$ ("milimeters per picture element"). Generalize the first-order interpolations in 2D and 3D by taking into account arbitrary dimensions of picture elements.



3. Write the kernel $c(i, j)$ of the digital filter of size $M = 1$ for the decimation in 3D. Test the kernel of the digital filter by decimating a 3D image (e.g. the image from Lab Work 3).

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 5: Image grayscale transformations

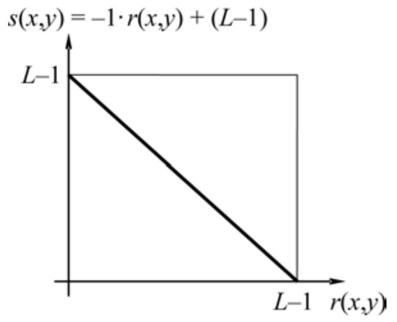
Navodila

The intensity transformation $\mathcal{T} : \mathbb{R} \rightarrow \mathbb{R}$ in general transforms the grayscale values of picture elements from the dynamic range of the original image $[0 \ldots L_r - 1]$ to the dynamic range of the transformed image $[0 \ldots L_s - 1]$.

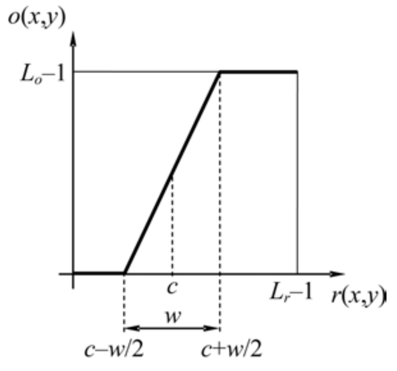
- **Linear transformation:** The grayscale value at each picture element of the reference image $r(x, y)$ is multiplied by the constant value a (changing image contrast) and added the constant value b (changing image brightness):

  $$s(x, y) = a \cdot r(x, y) + b \, .$$

  The resulting image $s(x, y)$ contains linearly transformed grayscale values.



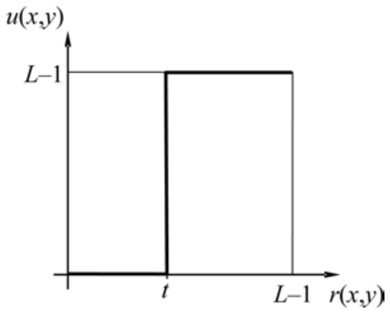
- **Linear windowing:** An arbitrary window with center at c and width of w is defined in the dynamic range of the reference image:
  - all grayscale values in reference image $r(x, y)$ that are smaller than the lower window edge $c - \frac{w}{2}$ are transformed to value 0,
  - all grayscale values in reference image $r(x, y)$ that are greater than the upper window edge $c + \frac{w}{2}$ are transformed to value $L_o - 1$,
  - all grayscale values in reference image $r(x, y)$ that are within the window are transformed with the linear transformation.



$$o(x, y) = \begin{cases} 0 & \text{pri} \quad r(x, y) < c - \frac{w}{2} \, , \\ \frac{L_o - 1}{w}\Big(r(x, y) - \big(c - \frac{w}{2}\big)\Big) & \text{pri} \quad c - \frac{w}{2} \leq r(x, y) \leq c + \frac{w}{2} \, , \\ L_o - 1 & \text{pri} \quad r(x, y) > c + \frac{w}{2} \, . \end{cases}$$

The resulting image $o(x, y)$ contains grayscale values that are within the window linearly transformed to fit the whole dynamic range $[0 \ldots L_o - 1]$.

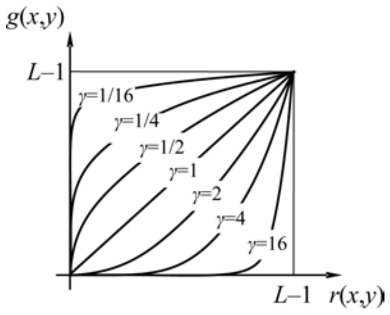
- **Thresholding:** The grayscale value at each picture element of the reference image $r(x,y)$ that is smaller or equal to the threshold t is transformed to value 0. If it is greater than the threshold, it is transformed to the largest possible grayscale value $L-1$:



$$u(x,y) = \begin{cases} 0 & \text{pri} \quad r(x,y) \le t\,, \\ L-1 & \text{pri} \quad r(x,y) > t\,. \end{cases}$$

The resulting image $u(x,y)$ is the thresholded image.

- **Gamma transformation:** This non-linear continuous transformation is, when the dynamic range of the reference image is equal to the dynamic range of the transformed image $L_r = L_o = L$, defined as:



$$g(x,y) = (L-1)^{1-\gamma} r^{\gamma}(x,y)\,.$$

The resulting image $g(x,y)$ is the gamma transformed image. The gamma transformation is useful when a continuous and non-linear increase in the contrast of brightest areas is needed, but at the expense of a decrease in contrast of darkest areas ($\gamma > 1$), or vice-versa ($\gamma < 1$).

The image spine-250x200x097-16bit.raw is a three-dimensional (3D) image of the spine acquired by the computed tomography (CT) imaging technique. The image of size $X \times Y \times Z = 250 \times 200 \times 97$ picture elements, where the size of the picture element is equal to $\Delta X \times \Delta Y \times \Delta Z = 0.6 \times 0.6 \times 4.0\,\text{mm}$, is stored with 16 bits in raw data format (RAW).

1. Load the image as a 3D matrix `oImage`. Take into account that the grayscale values are stored with 16 bits as signed integer values (data format `'int16'`). Extract arbitrary 2D lateral (x = const.), frontal (y = const.) and transversal (z = const.) basic planar cross-sections from the image.

2. Write the function for an arbitrary linear grayscale transformation:

   ```
   function sImage = scaleImage(iImage, iSlope, iIntersection),
   ```

   where `iImage` is the image to be transformed, and `iSlope` and `iIntersection` are the parameters of the linear transformation (parameters of the line a and b). The function returns the transformed image `sImage`. Test the function on image `oImage` using parameter values of `iSlope` $= 1$ and `iIntersection` $= -1024$.

3. Write the function for an arbitrary linear grayscale windowing:

   ```
   function wImage = windowImage(iImage, iCenter, iWidth),
   ```

where `iImage` is the image to be transformed, and `iCenter` and `iWidth` are the parameters of the linear windowing (window center c and window width w). The function returns the transformed image `wImage`. Test the function on image `sImage` using parameter values of `iCenter` $= 400$ and `iWidth` $= 2000$. Take into account that the dynamic range of an 8-bit screen (monitor) is equal to $L_o = 2^8$.

4. Write the function for an arbitrary grayscale thresholding:

   ```
   function tImage = thresholdImage(iImage, iThreshold),
   ```

   where `iImage` is the image to be transformed and `iThreshold` is the parameter of the thresholding (threshold t). The function returns the transformed image `tImage`. Test the function on image `wImage` using parameter value of `iThreshold` $= 127$.

5. Write the function for an arbitrary gamma grayscale transformation:

   ```
   function gImage = gammaImage(iImage, iGamma),
   ```

   where `iImage` is the image to be transformed and `iGamma` is the parameter of gamma transformation (gamma value γ). The function returns the transformed image `gImage`. Test the function on image `wImage` using parameter value of `iGamma` $= 2$.

6. Display arbitrary lateral ($x =$ const.), frontal ($y =$ const.) and transversal ($z =$ const.) basic planar cross-sections of 3D images `oImage`, `sImage`, `wImage`, `tImage` and `gImage`. Adjust the width and height of each displayed cross-section to the image size in milimeters by using the function `image(gridX, gridY, iImage)`, where vectors `gridX` and `gridY` contain the location (in milimeters) of each picture element along x and y coordinate axis.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Enclose the lateral, frontal and transversal cross-sections at $x = 125$, $y = 100$ and $z = 30$, respectively, for the 3D images `oImage`, `sImage` ($a = 1$, $b = -1024$), `wImage` ($c = 400$, $w = 2000$), `tImage` ($t = 127$) iand `gImage` ($\gamma = 2$). The size of the displayed images must be adjusted to the size of the picture element.

2. Write down the dynamic range $[0 \ldots L-1]$ for images `oImage`, `sImage`, `wImage`, `tImage` and `gImage`. What happens when picture elements that are outside the dynamic range of the 8-bit screen (monitor) are displayed? Explain your answer.

3. Are the transformed images useful only for visualization or also for storage? Explain your answer.

4. Enclose the histograms of images `oImage`, `sImage`, `wImage`, `tImage` and `gImage`. What happens with the histograms after greyscale transformations? Explain your answer.

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 6: Image geometrical transformations

Navodila

Geometrical transformations $\mathcal{T}$: $\mathbb{R}^2 \to \mathbb{R}^2$ or $\mathbb{R}^3 \to \mathbb{R}^3$ enable the transformation of picture elements (x, y) in a 2D image, or picture elements (x, y, z) in a 3D image, to new locations (u, v), or (u, v, w), by preserving the grayscale values of the picture elements:

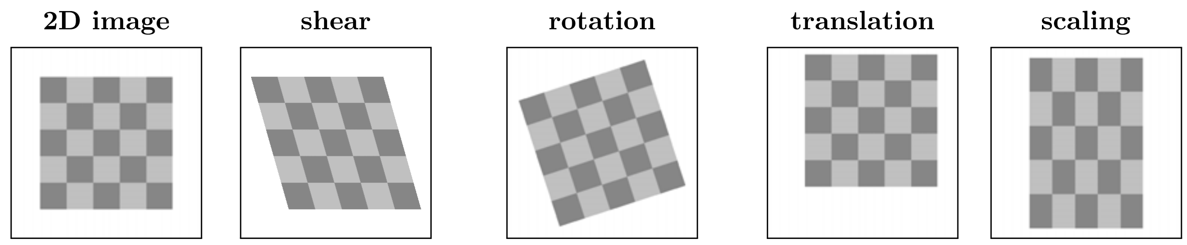
$$(u, v) = \mathcal{T}(x, y) \qquad \text{or} \qquad (u, v, w) = \mathcal{T}(x, y, z)\,.$$

The most general linear geometrical transformation is the **affine transformation**, which consists of arbitray scaling, rotation, translation and shear. The affine transformation can be defined as a product of the coordinates vector with the transformation matrix $\mathbf{T}$. In 2D, this matrix is completely defined by six parameters:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

The parameters t_x and t_y represent translation in x and y direction, respectively, and the parameters a_{ij} represent scaling, rotation and shear. The advantage of the matrix representation is in the ability to compose the affine transformation as a series of individual elementary transformations, i.e. scaling ($\mathbf{T}_{\text{scal}}$), translation ($\mathbf{T}_{\text{trans}}$), rotation ($\mathbf{T}_{\text{rot}}$) and shear ($\mathbf{T}_{\text{shear}}$):

$$\mathbf{T}_{\text{affine}} = \mathbf{T}_{\text{shear}} \mathbf{T}_{\text{rot}} \mathbf{T}_{\text{trans}} \mathbf{T}_{\text{scal}}\,.$$



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & g_{xy} & 0 \\ g_{yx} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

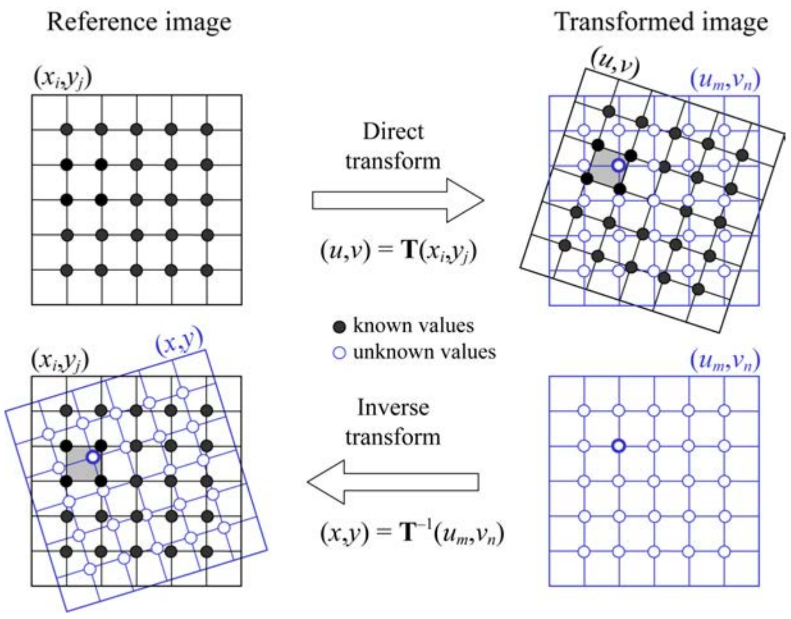
After transformation, the picture elements of the reference image at locations (x_i, y_j) are transformed to new locations (u, v):

$$(u, v) = \mathbf{T}(x_i, y_j)\,.$$

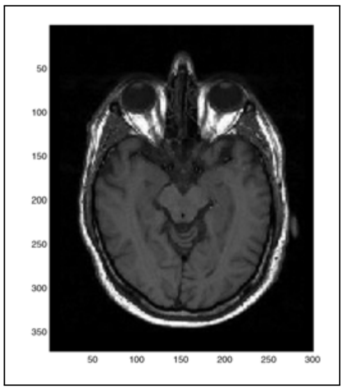
In general, the new locations do not coincide with the discrete grid of sampling points in the transformed image (u_m, v_n). Interpolation from neighboring transformed points is therefore required to determine the grayscale values at points (u_m, v_n). Such direct transformation approach can complicate the interpolation process, because the neighboring transformed points can be arbitrary distributed in space. Their spatial distribution depends on the type and parameters of the transformation. As a result, the use of the inverse transformation is recommended:

$$(x, y) = \mathbf{T}^{-1}(u_m, v_n)\,,$$

which defines for every discrete sampling point of the transformed image (u_m, v_n) its corresponding location (x, y) in the reference image, i.e. the location from which was the discrete sampling point (u_m, v_n) transformed and that is in general located between four sampling points of the discrete grid (x_i, y_j) of the reference image. The grayscale value at the sampling point (u_m, v_n) can be, in this case, computed by interpolating the grayscale values at the corresponding point (x, y) in the reference image, as the neighboring points are ordered in the discrete grid and can be therefore easily determined. In most cases, the bilinear interpolation is used for 2D images because it is relatively simple and fast.



The two-dimensional (2D) image head-150x372-08bit.raw, acquired by the magnetic resonance (MR) imaging technique, represents an axial cross-section of the human head. The image is stored with 8 bits in raw data format (RAW), is of size $X \times Y = 150 \times 372$ picture elements, and the size of the picture element is equal to $\Delta X \times \Delta Y = 2.0 \times 1.0$ mm.



1. Write the function that computes the matrix of affine transformation in 2D for the given input parameters:

```
function oMatrix = getAffineMatrix2D(iShear, iRot, iTrans, iScal),
```

   where iShear $= [g_{xy}, g_{yx}]$ is the shear vector, iRot $= \varphi$ is the angle of rotation, iTrans $= [t_x, t_y]$ is the vector of translation, and iScal $= [k_x, k_y]$ is the vector of scaling factors. The function returns the 2D affine transformation matrix oMatrix.

2. Write the function that computes the grid of picture element locations for the given input image:

   ```
   function [oGridX, oGridY] = getGrid2D(iDim, iSize),
   ```

   where `iDim` $= [X, Y]$ is the size of the reference image in picture elements, and `iSize` $= [\Delta X, \Delta Y]$ is the size of the picture element in milimeters. The function returns the vectors `oGridX` and `oGridY` that contain, respectively, the x and y picture element locations in the reference image in milimeters:

   $$\texttt{oGridX} = [\Delta x, 2\Delta x, \ldots, X\Delta X] \qquad \text{in} \qquad \texttt{oGridY} = [\Delta y, 2\Delta y, \ldots, Y\Delta Y].$$

3. Write the function for the affine transformation of points in 2D:

   ```
   function [oCoorX, oCoorY] = affineTransform2D(iGridX, iGridY, iMatrix),
   ```

   where `iGridX` and `iGridY` are the coordinate grids of picture elements in milimeters, and `iMatrix` is the inverse affine transformation matrix $\mathbf{T}^{-1}$. The function returns the matrices `oCoorX` and `oCoorY` that contain, respectively, the x and y coordinates of the transformed picture elements in the reference image in milimeters. For example, if `oCoorX`$(u, v) = x_c \in \mathbb{R}$ and `oCoorY`$(u, v) = y_c \in \mathbb{R}$, then the value at location (x_c, y_c) in the reference image is transformed to the discrete location (u, v).

4. The file bilinearInterpol.m contains the function for the bilinear interpolation of grayscale values according to the given coordinates of the transformed points:

   ```
   function oImage = bilinearInterpol(iImage, iSize, iCoorX, iCoorY, iBgr),
   ```

   where `iImage` is the reference 2D image, `iSize` $= [\Delta X, \Delta Y]$ is the size of the picture element in milimeters, `iCoorX` and `iCoorY` are the matrices of trasnformed point coordinates, and `iBgr` is the grayscale value of the picture elements that are not located inside the reference image and therefore represent the background (usually, `iBgr` $= 0$ is used, but to verify the transaformation use `iBgr` $= 127$). The function returns the interpolated image `oImage`. Use the given function for the interpolation of grayscale values according to the given transformed point coordinates.

5. Adjust the above function so that you take into account also the arbitrary origin of the coordinate system `iCenter` $= [c_x, c_y]$.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. The rotational and translational transformation can be joined into the rigid transformation, where usually rotation is performed before translation. Write down the general matrix of the rigid transformation $\mathbf{T}_{\text{rig}}$ in 2D.

2. What are the properties of the rigid transformation (parallelity of lines, preservation of angles, preservation of distances)? Hint: Try different rigid transformations on the test image test-190x108-08bit.raw (size 190×108 picture elements, size of picture element 1.0×1.0 milimeters, stored with 8 bits in raw data format).

3. The rigid transformation can be joined with scaling factor k; $k = k_x = k_y$ to obtain the similarity transformation. Write down the general matrix of the similarity transformation $\mathbf{T}_{\text{sim}}$ in 2D.

4. What are the properties of the similarity transformation ? Hint: Try different similarity transformations on the test image.

5. Enclose the transformed images when appying shear of `iShear` $= [0.1, 0.5]$. The origin of the coordinate system is first located at $(c_x, c_y) = (0, 0)$ and then in the center of the image. The background value should be set to `iBgr` $= 127$.

6. Enclose the transformed images when applying rotation of `iRotation` $= -30°$. The origin of the coordinate system is first located at $(c_x, c_y) = (0, 0)$ and then in the center of the image. The background value should be set to `iBgr` $= 127$.

7. Enclose the transformed images when applying translation of `iTranslation` $=$ $[+20, -30]$ mm. The origin of the coordinate system is first located at $(c_x, c_y) = (0, 0)$ and then in the center of the image. The background value should be set to `iBgr` $= 127$.

8. Enclose the transformed images when applying scaling of `iScaling` $= [0.7, 1.4]$. The origin of the coordinate system is first located at $(c_x, c_y) = (0, 0)$ and then in the center of the image. The background value should be set to `iBgr` $= 127$.

9. Enclose the transformed images when applying rigid transformation of `iRotation` $= -30°$, `iTranslation` $= [+20, -30]$ mm and `iScaling` $= [0.7, 1.4]$. The origin of the coordinate system is first located at $(c_x, c_y) = (0, 0)$ and then in the center of the image. The background value should be set to `iBgr` $= 127$.

10. Enclose the transformed images when applying affine transformation of `iShear` $= [0.1, 0.5]$, `iRotation` $= -30°$, `iTranslation` $= [+20, -30]$ mm and `iScaling` $= [0.7, 1.4]$. The origin of the coordinate system is first located at $(c_x, c_y) = (0, 0)$ and then in the center of the image. The background value should be set to `iBgr` $= 127$.

11. What are the properties of the affine transformation (parallelity of lines, preservation of angles, preservation of distances)? Hint: Try different affine transformations on the test image.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

The affine transformation was defined as a specific series of elementary transformations, i.e. the series shear-rotation-translation-scaling:

$$\mathbf{T}_{\text{affine}} = \mathbf{T}_{\text{shear}}\mathbf{T}_{\text{rot}}\mathbf{T}_{\text{trans}}\mathbf{T}_{\text{scal}}\,.$$

What happens when we choose another order in the series? Explain your answer.

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 7: Image spatial filtering

Navodila

Spatial filtering is defined as a local transformation of image grayscale values, performed by the operator $\mathcal{T}$ in image domain:

$$g(x,y) = \mathcal{T}\big(f(x,y)\big)\,,$$

where $f(x,y)$ is the input image, $g(x,y)$ is the filtered image, and $\mathcal{T}$ is the local operator (filter kernel), which is defined on the orthogonal discrete domain of size $M \times N$, with M and N being integer odd numbers; $M = 2m+1$ and $N = 2n+1$. The filtered image $g(x,y)$ is obtained by moving the operator and computing its response in each discrete image point. When the filter coefficients $w(i,j)$ are symmetrical according to its center, the filtering operation is equal to the convolution of the image with the filter:

$$g(x,y) = \sum_{i=-m}^{m}\sum_{j=-n}^{n} w(i,j)\cdot f(x-i,y-j)\,,$$

The problem of computing the convolution at the image edges is usually solved by enlarging the discrete image domain on each side of the image for m and n in x and y direction, respectively. The grayscale values are then extrapolated over the image edges, which ensures their continuous transition.

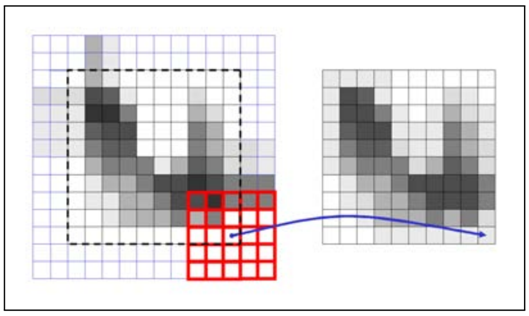


Image blurring is simply performed by computing the arithmetical average of all grayscale values within the filter domain. Besides arithmetical averaging, we often use weighted averaging, which emphasizes the center of the filter, or Gaussian filtering, which defines filter coefficients according to the Gaussian distribution:

$$w(i,j) = \frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{i^2+j^2}{2\sigma^2}}\,.$$

Arithmetical averaging

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

Weighted averaging

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Gaussian filtering

0,01	0,08	0,01
0,08	0,64	0,08
0,01	0,08	0,01

$(\sigma = 0,5)$

Image sharpening is analogous to spatial differentiation of grayscale values. The most simple operator for the second derivative is the Laplace operator. Image sharpening is performed by subtracting the weighted image of the second derivative from the input image $f(x, y)$:

$$g(x, y) = f(x, y) - c \left[\nabla^2 f(x, y)\right] ,$$

where the constant c defines the level of sharpening. Often, unsharp masking is used. The input image $f(x, y)$ is subtracted from the blurred image $F(x, y)$, resulting in the mask $m(x, y)$, which is then added to the input image according to the level of sharpening c:

$$m\big(f(x, y)\big) = f(x, y) - F\big(f(x, y)\big) \qquad \Longrightarrow \qquad g(x, y) = f(x, y) + c \left[m\big(f(x, y)\big)\right] .$$

The images can be also sharpened by using first derivatives or gradients. The vector image of the gradient $g(x, y)$ in each image point (x, y) points into the direction of the largest change of the function $f(x, y)$. The gradient amplitude can be computed in each image point as:

$$G(x, y) = \mathrm{amp}\left[\nabla f(x, y)\right] = \sqrt{g_x^2(x, y) + g_y^2(x, y)} \, .$$

The gradient components $g_x(x, y)$ and $g_y(x, y)$ that represent image partial derivatives in x and y direction, respectively, are usually computed by using the Sobel operator.

Laplace operator

1	1	1
1	-8	1
1	1	1

Sobel operator for x direction

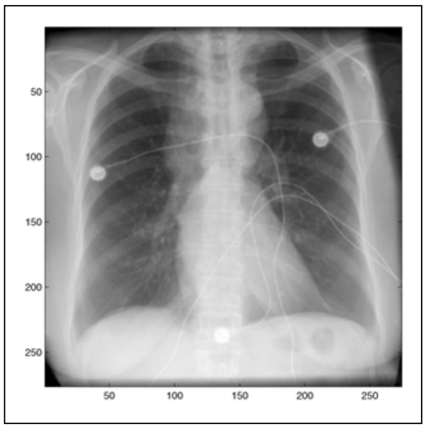
-1	0	1
-2	0	2
-1	0	1

Sobel operator for y direction

-1	-2	-1
0	0	0
1	2	1

Statistical filtering arranges the grayscale values within the filter domain according to their size, from the largest to the smallest. The most known statistical filter is the median filter. The median is the central value in the ordered series of elements, therefore the response of the median filter of size $N \times N$ is equal to $\frac{N(N+1)}{2}$-th largest grayscale value. Besides the median, other functions can be used, such as the maximal or minimal value.

The two-dimensional (2D) radiographic image chest-274x276-08bit.raw represents a frontal X-ray projection of the chest. The image of size $X \times Y = 274 \times 276$ is stored with 8 bits in raw data format (RAW).



1. Write the function for spatial filtering of an image in 2D:

   ```
   function oImage = kernelFiltering2D(iImage, iKernel),
   ```

   where `iImage` is the input image, and `iKernel` is the filter kernel of size $N \times N$. The function returns the matrix of filtered values `oImage` of the same size as the input image. In general, the dynamic range of the filtered values is not equal to the dynamic range of the input image, moreover, the filtered values are not always integers. When displaying the results on screen, we therefore modify the dynamic range of the filtered values to 8 bits: `oImage = uint8(oImage)`.

2. Apply different filter kernels to perform image blurring (arithmetical averaging, weighted averaging, Gaussian filtering). Display the results on screen.

3. Sharpen the image by using the method based on the Laplace operator and unsharp masking. Set the level of sharpening to $c = 2$. Display the results on screen.

4. Determine the gradient images in x and y direction by using the Sobel operator. Compute also the amplitude gradient image. Display the results on screen.

5. Write the function for statistical filtering of an image in 2D:

   ```
   function oImage = statisticalFiltering2D(iImage, iLength, iFunc),
   ```

   where `iImage` is the input image, `iLength` is the length N of the square filter $N \times N$, and `iFunc` is the name of the Matlab statistical function, e.g. `'median'`. Perform the filtering by:

   ```
   oValue = feval(iFunc, iVector),
   ```

   where `iVector` is the vector of grayscale values that correspond to the given position of the filter on image, and `oValue` is the filtered value at the current position of the filter. The function returns the filtered image `oImage`.

6. Filter the image with the median filter, the maximal value filter and the minimal value filter. Display the results on screen.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Enclose the images that were blurred with arithmetical averaging, weighted averaging and Gaussian filtering. For what is, in general, useful image blurring?

2. Write down the filter kernel of size $N \times N = 5 \times 5$ for image blurring with arithmetical averaging, with weighted averaging, and with Gaussian filtering ($\sigma = 2$).

3. Enclose the images, sharpened by the Laplace operator and unsharp masking. Enclose also the image of the response to the Laplace operator and the unsharp mask image. For what is, in general, useful image sharpening? What is the disadvantage of sharpening, visible also in the obtained images?

4. Enclose the images of the Sobel gradients in x and y direction as well as the amplitude image. For what are, in general, useful image gradients?

5. Enclose the images, obtained by statistical filtering with the median filter, the maximal value filter and the minimal value filter. For what is, in general, useful statistical image filtering? What is the consequence of using each of the filters? What is the main difference between median filtering and image blurring?

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

1. Write the function for the determination of the 3D Gaussian filter kernel:

```
function oKernel = getGaussianKernel2D(iLength, iStd),
```

   where `iLength` is the arbitrary filter size N, and `iStd` is the standard deviation of the Gaussian distribution. The function returns the Gaussian filter kernel `oKernel`. Take into account that the filter coefficients are symmetrical and that their sum must be equal to 1.

2. The filtering of 2D images is performed by filter kernels in the form of a 2D matrix. Similarly is the filtering of 3D images performed by filter kernels in the form of a 3D matrix. For example, to determine the gradient in x direction, the Sobel operator is equal to:

`SobelX(:,:,1) =`

$$= \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

`SobelX(:,:,2) =`

$$= \begin{array}{|c|c|c|}\hline -2 & 0 & 2 \\ \hline -4 & 0 & 4 \\ \hline -2 & 0 & 2 \\ \hline \end{array}$$

`SobelX(:,:,3) =`

$$= \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

   What is the form of the corresponding Sobel operators that determine the gradients in y and z direction?

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 8: Image geometrical registration (1)

Navodila

The geometrical registration of two images can be defined as the techniques that searches for the optimal geometrical transformation $\mathcal{T}$, which will transform the images so that the same structures (or objects) will be located at the same position in both images. One of the most popular techniques is the registration of corresponding control points in the reference and input image. The term "corresponding" means that the pairs of control points represent the same anatomical structures (or objects) in both images. From the set of control point pairs, we can determine the transformation $\mathcal{T}$, so that the transformed control points in the reference image $\mathcal{T}(x_k, y_k)$ are aligned as good as possible with the control points in the input image (u_k, v_k), or vice versa:

$$\mathcal{T}(x_k, y_k) \leftrightarrow (u_k, v_k) \qquad \text{or} \qquad (x_k, y_k) \leftrightarrow \mathcal{T}^{-1}(u_k, v_k)\,.$$

The affine registration in 2D is defined by the affine geometrical transformation in 2D:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

The **interpolative affine registration** transforms the image so that the corresponding control point pairs are perfectly aligned. The six parameters of the affine transformation can be uniquely defined from three pairs ($K = 3$) of noncolinear points $(x_k, y_k) \leftrightarrow (u_k, v_k)$. The transformation between these three point pairs can be represented by the transformatin matrix $\mathbf{T}$:

$$\mathbf{T} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1}.$$

The **approximative affine registration** transforms the image so that the corresponding control point pairs are approximately aligned. If more than three pairs ($K > 3$) of control points $(x_k, y_k) \leftrightarrow (u_k, v_k)$ exist, the obtain system of equations is over-defined (more equations than unknowns). The control point pairs can therefore not be perfectly (interpolatively) aligned, however, they can be approximately aligned by searching for the minimal mean square distance R^2 between all corresponding control point pairs $(x_k, y_k) \leftrightarrow (u_k, v_k)$ after registration:

$$R^2 = \frac{1}{K} \sum_{k=1}^{K} \big(\mathcal{T}(x_k, y_k) - (u_k, v_k)\big)^2$$

$$R^2 = \frac{1}{K} \sum_{k=1}^{K} \big((a_{11}x_k + a_{12}y_k + t_x - u_k)^2 + (a_{21}x_k + a_{22}y_k + t_y - v_k)^2\big)\,.$$

This procedure is also known as the least-squares fitting. The optimal values of six unknown parameters are obtained by setting to zero the derivatives of the mean square distance R^2, which results in six equations for the six unknown parameters:

$$\begin{bmatrix} \overline{xx} & \overline{xy} & \overline{x} & 0 & 0 & 0 \\ \overline{xy} & \overline{yy} & \overline{y} & 0 & 0 & 0 \\ \overline{x} & \overline{y} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{xx} & \overline{xy} & \overline{x} \\ 0 & 0 & 0 & \overline{xy} & \overline{yy} & \overline{y} \\ 0 & 0 & 0 & \overline{x} & \overline{y} & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ t_x \\ a_{21} \\ a_{22} \\ t_y \end{bmatrix} = \begin{bmatrix} \overline{ux} \\ \overline{uy} \\ \overline{u} \\ \overline{vx} \\ \overline{vy} \\ \overline{v} \end{bmatrix}$$

The overline elements represent the mean values, e.g.:

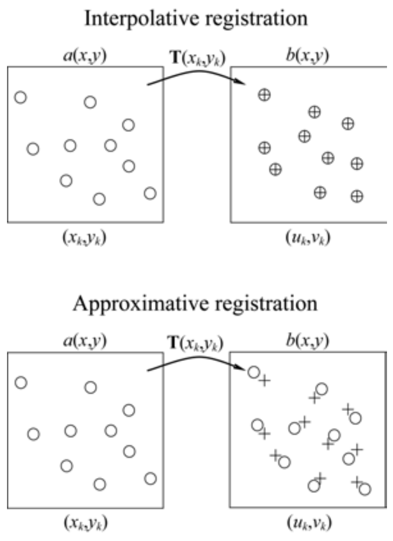
$$\overline{ux} = \frac{1}{K}\sum_{k=1}^{K} u_k x_k \quad \text{and} \quad \overline{x} = \frac{1}{K}\sum_{k=1}^{K} x_k .$$

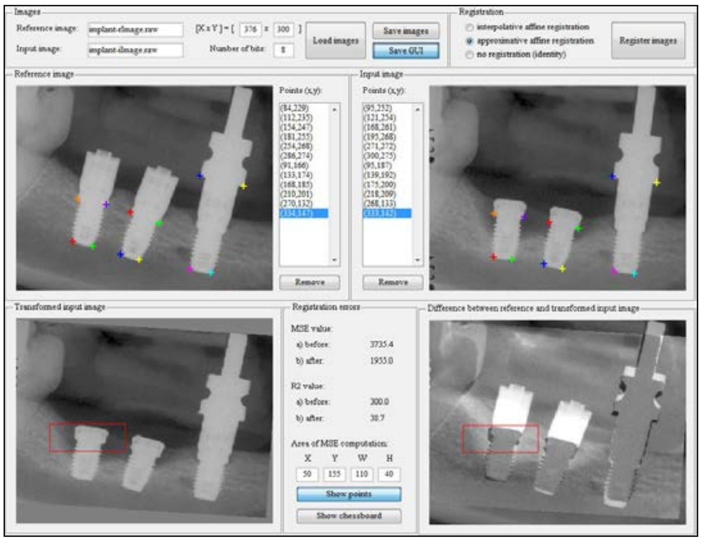
The system of the equations above can be represented in matrix form. The vector t of unknown parameters of the approximative affine registration $\mathcal{T}$ is then computed by inverting the matrix P_{xy}:

$$P_{xy} \cdot t = p_{uv} \quad \rightarrow \quad t = P_{xy}^{-1} \cdot p_{uv} \, .$$

The **similarity measure** is an arbitrary scalar function, defined for all corresponding picture elements of the reference image $a(x, y)$ and the transformed input image $b(x, y)$, and reaching its optimal value when the registration of images is optimal. A classical similarity measure is the mean square error (MSE) of the grayscale values:

$$\mathrm{MSE}(a, b) = \frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(a(x_i, y_j) - b(x_i, y_j)\right)^2 \, .$$




Images
Reference image: implant-rImage.raw
Input image: implant-iImage.raw
[X x Y] = [376 x 300]
Number of bits: 8
Load images
Save images
Save GUI
Registration
interpolative affine registration [✓ approximative affine registration] no registration (identity)
Register images

Reference image
Points (x,y):
(84,229)
(112,235)
(154,247)
(181,255)
(254,268)
(286,274)
(91,166)
(133,174)
(168,185)
(210,201)
(270,132)
(334,147)
Remove

Input image
Points (x,y):
(95,252)
(121,254)
(168,261)
(195,268)
(271,272)
(300,275)
(95,187)
(139,192)
(175,200)
(218,209)
(268,133)
(333,142)
Remove

Transformed input image

Registration errors
MSE value:
a) before: 3735.4
b) after: 1955.0
R2 value:
a) before: 300.0
b) after: 38.7
Area of MSE computation:
X: 50 Y: 155 W: 110 H: 40
Show points
Show chessboard

Difference between reference and transformed input image

Extract the files from the archive ZIP file `lab08.zip` to a selected folder and run the graphical user interface (GUI) `lab08.m` in Matlab. By using the GUI, you can load the reference and input image (the image size and filenames are preset), select the control points by simple clikcing on the selected image, call the functions for affine image registration, and display the means square distance between control points and the mean square error of grayscale values.

Follow the instructions and modify the existing functions for the affine registration (interpolative and approximative) and for the computation of registration errors (R^2 and MSE):

1. Write the registration procedures into the existing function affineImageRegistration2D.m:

```
>>  % affine image registration in 2D
>>  function [oImage, oLand, oMatrix] = affineImageRegistration2D(rLand,...
>>        iLand, iImage, iType)
>>      % initialize transformation by identity
>>      oImage = iImage;
>>      oLand = iLand;
>>      oMatrix = eye(3);
>>      % interpolative affine registration
>>      if (strcmp(iType, 'interpolative affine registration'))
>>            % !!!  WRITE YOUR CODE HERE !!!
>>      % approximative affine registration
>>      elseif (strcmp(iType, 'approximative affine registration'))
>>            % !!!  WRITE YOUR CODE HERE !!!
>>      end
>>      % affine transformation of the image in 2D according to oMatrix
>>      % !!!  WRITE YOUR CODE HERE !!!
>>      % affine transformation of the points in 2D according to oMatrix
>>      % !!!  WRITE YOUR CODE HERE !!!
```

The input and output parameters are defined as:

`rLand`	The points $[x_1, y_1;\ x_2, y_2;\ \ldots;\ x_k, y_k]$ in the reference image that with points `iLand` form the corresponding control point pairs.
`iLand`	The points $[u_1, v_1;\ u_2, v_2;\ \ldots;\ u_k, v_k]$ in the input image that with points `rLand` form the corresponding control point pairs.
`iImage`	The input image.
`iType`	The type of the affine transformation: interpolative (`'interpolative affine registration'`) or approximative (`'approximative affine registration'`).
`oImage`	The transformed input image.
`oLand`	The transformed points $[x'_1, y'_1;\ x'_2, y'_2;\ \ldots;\ x'_k, y'_k]$ in the input image.
`oMatrix`	The matrix of the geometrical transformation $\mathbf{T}$.

To transform the input image, you can use the following function:

```
function oImage = affineImageTransform2D(iImage, iMatrix),
```

where `iImage` is the input image, and `iMatrix` is the matrix of the geometrical transformation $\mathbf{T}$. The function returns the transformed input image `oImage`.

2. Write the procedures for the computation of the mean square distance and mean square error of the registration into the existing function computeError.m:

```
>>  % error R2 and MSE before and after registration
>>  function [R2, MSE] = computeError(rLand, iLand, oLand, ...
>>        rImage, iImage, oImage, iArea)

>>      % initial value for R2 and MSE
>>      MSE = zeros(1,2);
>>      R2 = zeros(1,2);

>>      % !!!  WRITE YOUR CODE HERE !!!
```

The input and output parameters are defined as:

`rLand`	The points $[x_1, y_1;\ x_2, y_2;\ \ldots;\ x_k, y_k]$ in the reference image that with points `iLand` form the corresponding control point pairs.
`iLand`	The points $[u_1, v_1;\ u_2, v_2;\ \ldots;\ u_k, v_k]$ in the input image that with points `rLand` form the corresponding control point pairs.
`oLand`	The transformed points $[x'_1, y'_1;\ x'_2, y'_2;\ \ldots;\ x'_k, y'_k]$ in the input image.
`rImage`	The reference image.
`iImage`	The input image.
`oImage`	The transformed input image.
`iArea`	The area of MSE computation in the form $[x, y, w, h]$, where (x, y) are the coordinates of the upper-left area corner, w is the area width, and h is the area height.
`R2`	The vector of the mean square distance between the control point pairs in the vector form $[r_1^2, r_2^2]$, where r_1^2 is the distance before registration, and r_2^2 is the distance after registration.
`MSE`	The vector of the mean square error of grayscale values in the vector form $[m_1, m_2]$, where m_1 is the error before registration, and m_2 is the error after registration.

Test the interpolative and approximative affine registration by clicking the corresponding point pairs and pressing the button Register images. In the case the functions affineImageRegistration2D.m or computeError.m are modified, you do not have to restart the GUI and define the control point pairs from the start, but only perform the registration.

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Enclose the image of the GUI for a successful approximative affine registration, based on 12 control point pairs. (The GUI is saved to picture picture_6_user_interface.jpg by pressing the button Save GUI.)
2. Determine 6 control point pairs, perform the approximative affine registration, and write down the mean square distance between R^2 the control point pairs and the mean square error MSE of the grayscale values before and after registration. Set the area of MSE computation to `iArea` $= [50, 155, 110, 40]$. Repeat the procedure 10 times, however, try to always define the control points as similarly as possible. Compute the mean values and standard deviations for the obtained R^2 and MSE. What are the conclusions of such experiments? Explain you answer.
3. Enclose the registered image, the image of differences and the chessboard image for a successful interpolative affine registration. (The images are saved to picture_3_registered.bmp, picture_4_difference.bmp and picture_5_chessboard.bmp by pressing the button Save images.)
4. Which points did you choose for control point pairs? What is important when choosing these points? Explain your answer.
5. What are the disadvantages of the similarity measure MSE? Explain your answer.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

Write a function with the coordinates of an arbitrary number of points in the reference image `rLand`, the reference image `rImage`, and the input image `iImage` as the function input parameters. The function output parameters are the coordinates of the corresponding points in the input image`iLand`, which are determined so that the mean square error of all grayscale values between images `rImage` and `iImage` is minimal.

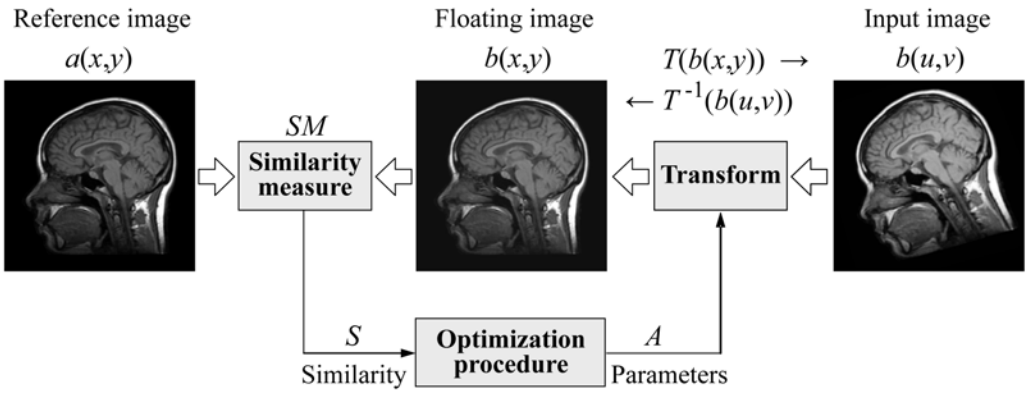
University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 9: Image geometrical registration (2)

Navodila

The geometrical registration with the **similarity optimization** is based on the maximization of the similarity between the grayscale values of all corresponding picture elements in two images. The transformation, similarity measure and optimization procedure have to be chosen so that an automated iterative similarity optimization is performed. In each step of the iteration, the optimization procedure proposes the parameter values of the matrix $\mathbf{T}$, which defines the geometrical transformation. According to the proposed parameters, the input image $b(u,v)$ is transformed into the floating image $b(x,y)$, which is then compared to the reference image $a(x,y)$ by computing the similarity measure (SM) that defines the similarity S:

$$\mathrm{S} = \mathrm{SM}\big(a(x,y) \leftrightarrow b(x,y)\big) = \mathrm{SM}\big(a(x,y) \leftrightarrow \mathbf{T}^{-1}\big(b(u,v)\big)\big)\,.$$



The similarity measure is an arbitrary scalar function, defined between the corresponding picture elements in the reference image $a(x,y)$ and the floating image $b(x,y)$, and reaches the optimal value at the optimal image registration:

- Mean square error (MSE):

$$\mathrm{MSE}(a,b) = \frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(a(x_i,y_j) - b(x_i,y_j)\right)^2\,.$$

- Mean absolute error (MAE):

$$\mathrm{MAE}(a,b) = \frac{1}{IJ}\sum_{i=1}^{I}\sum_{j=1}^{J}\left|a(x_i,y_j) - b(x_i,y_j)\right|\,.$$

- Correlation coefficient (CC):

$$\mathrm{CC}(a,b) = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(a(x_i,y_j)-\overline{a}\right)\left(b(x_i,y_j)-\overline{b}\right)}{\sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(a(x_i,y_j)-\overline{a}\right)^2\sum_{i=1}^{I}\sum_{j=1}^{J}\left(b(x_i,y_j)-\overline{b}\right)^2}}\,.$$

- Mutual information (MI):

$$\mathrm{MI}(a,b) = H(a) + H(b) - H(a,b)\,,$$

where $H(a)$ is the entropy of the reference image $a(x,y)$, $H(b)$ is the entropy of the floating image $b(x,y)$, and $H(a,b)$ is their joint entropy:

$$H(a) = -\sum_{s_a=0}^{L-1} p_a(s_a)\log\big(p_a(s_a)\big)$$

$$H(b) = -\sum_{s_b=0}^{L-1} p_b(s_b)\log\big(p_b(s_b)\big)$$

$$H(a,b) = -\sum_{s_a=0}^{L-1}\sum_{s_b=0}^{L-1} p_{ab}(s_a,s_b)\log\big(p_{ab}(s_a,s_b)\big)\,.$$

The probability distributions $p_a(s_a)$ and $p_b(s_b)$, as well as the joint distribution $p_{ab}(s_a,s_b)$, are estimated by normalizing the corresponding histograms $h_a(s_a)$, $h_b(s_b)$ and $h_{ab}(s_a,s_b)$:

$$p_a(s_a) = \frac{h_a(s_a)}{IJ}\,, \qquad p_b(s_b) = \frac{h_b(s_b)}{IJ} \qquad \text{and} \qquad p_{ab}(s_a,s_b) = \frac{h_{ab}(s_a,s_b)}{IJ}\,.$$

The variables s_a and s_b denote the discrete grayscale values of image $a(x,y)$ and image $b(x,y)$.

The images test-A-040x060-08bit.raw (reference test image A), test-B1-040x060-08bit.raw (first input test image B1) and test-B2-040x060-08bit.raw (second input test image B2) of size $X\times Y = 40\times 60$ picture elements are stored with 8 bits in raw data format (RAW).

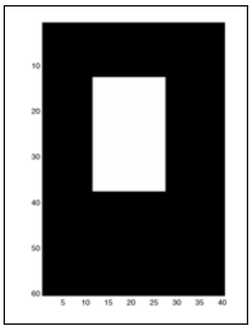


image A

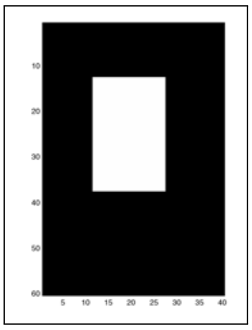


image B1

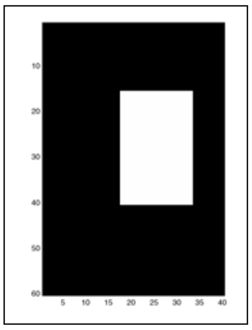


image B2

1. Write the algorithm for the registration of the reference image (A) and the input image (B1 or B2), based on an arbitrary similarity measure SM. The geometrical transformation is defined only by the translation in x direction (parameter t_x) and the translation in y direction (parameter t_y). Instead of the optimization procedure, use the exhaustive search procedure, defined by $t_x = -10\ldots+10$ and $t_y = -10\ldots+10$.

2. Compute the values of each similarity measure (MSE, MAE, CC and MI) in each point of the parameter space of the transformation and display them as a surface in the parameter space of the transformation (t_x,t_y). Use the function `surf(Tx, Ty, SM)`, where the input parameters are equal to:

`Tx`	The matrix of translations t_x in x direction: $\big[t_{x1}, t_{x2}, \ldots, t_{xn};\;\; t_{x1}, t_{x2}, \ldots, t_{xn};\;\; \ldots;\;\; t_{x1}, t_{x2}, \ldots, t_{xn}\big]$.
`Ty`	The matrix of translations t_y in y direction:: $\big[t_{y1}, t_{y1}, \ldots, t_{y1};\;\; t_{y2}, t_{y2}, \ldots, t_{y2};\;\; \ldots;\;\; t_{ym}, t_{ym}, \ldots, t_{ym}\big]$.
`MP`	The matrix of the similarity measure at locations (t_{xi}, t_{yj}): $\big[\mathrm{MP}_{11}, \mathrm{MP}_{12}, \ldots, \mathrm{MP}_{1n};\;\; \mathrm{MP}_{21}, \mathrm{MP}_{22}, \ldots, \mathrm{MP}_{2n};\;\; \ldots;\;\; \mathrm{MP}_{m1}, \mathrm{MP}_{m2}, \ldots, \mathrm{MP}_{mn}\big]$.

Use the grayscale color map (function `colormap(·)`) and view from above (function `view(0,90)`).

3. Determine the optimal (maximal or minimal) value of each similarity measure SM and the translation parameters t_x^{opt} and t_y^{opt}, which give the optimal (maximal or minimal) value of the observed similarity measure SM.

4. Display the difference between the reference image (A) and the floating image (transformed B1 or B2) at translation parameters t_x^{opt} and t_y^{opt} for each similarity measure (MSE, MAE, CC and MI). Adjust the display for showing the image difference (add the middle of the dynamic range of the image to the image difference, i.e. value 127).

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

Two-dimensional (2D) images of the head, acquired by the magnetic resonance (MR) imaging, are given as the T$_2$-weighted image head-T2-083x100-08bit.raw (reference image A), T$_1$-weighted image head-T1-083x100-08bit.raw (first input image B1) and SD-weighted image head-SD-083x100-08bit.raw (second input image B2). The images of size $X \times Y = 83 \times 100$ picture elements are stored with 8 bits in raw data format (RAW).

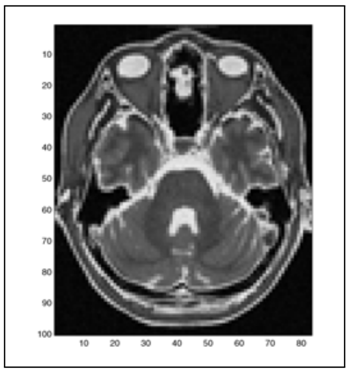


image A - MR T$_2$

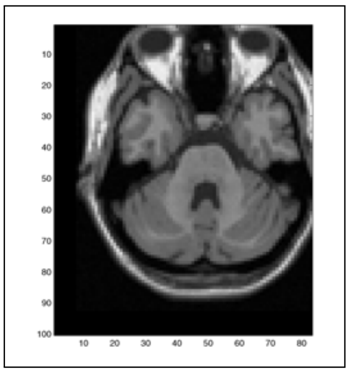


image B1 - MR T$_1$

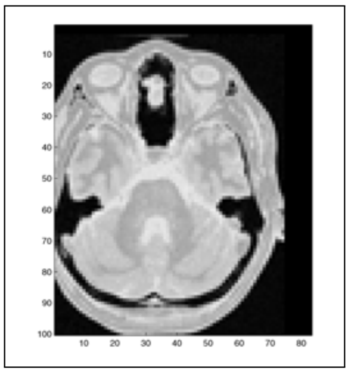


image B2 - MR SD

1. Display each similarity measure (MSE, MAE, CC and MI) as a surface in the parameter space of the transformation, defined by $t_x = -15 \ldots +15$ and $t_y = -15 \ldots +15$.

2. Determine the optimal value of each similarity measure and the corresponding translation parameters t_x^{opt} and t_y^{opt}, at which the observed similarity measure (MSE, MAE, CC and MI) reaches its optimum.

3. Display the difference between the reference image (A) and the floating image (transformed B1 and B2) at translation parameters t_x^{opt} and t_y^{opt} for each similarity measure (MSE, MAE, CC and MI).

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

In practice, the exhaustive procedure is not useful as it is time consuming, which is especially important when a larger number of transformation parameters is present in matrix $\mathbf{T}$. Write the registration algorithm that defines the optimal translation parameters t_x^{opt} and t_y^{opt} by an optimization procedure. Use the function `fminsearch(·)` to search for the minimum of the scalar function of multiple variables.

University of Ljubljana, Faculty of Electrical Engineering
Medical Informatics and Diagnostics (MID)

Lab Work 10: Image segmentation

Navodila

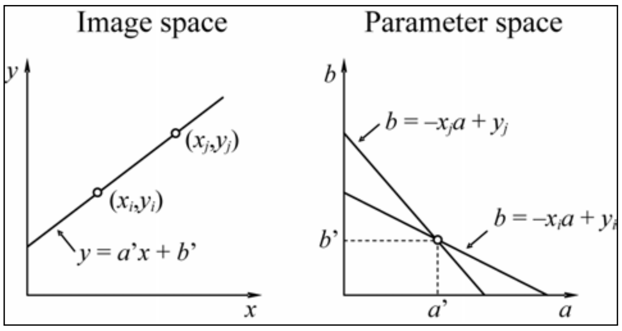


Image segmentation refers to a collection of techniques that divide the image into basic areas or objects. One of the techniques for the determination of image objects is the line search. An efficient technique for line search in images is based on the **Hough transform**. For a binary image of edges $f(x, y)$, we can draw an arbitrary number of lines through each edge point (x_i, y_i):

$$y_i = ax_i + b\,.$$

The same equation can be written in the parameter space (a, b), where the edge point coordinates x_i and y_i are the parameters of the line:

$$b = -x_i a + y_i\,.$$

For a different edge point (x_j, y_j), we obtain in the parameter space (a, b) the following line:

$$b = -x_j a + y_j\,,$$

which intersects, in point (a', b'), the line that belongs to point (x_i, y_i). The parameters (a', b') define the line in the image space (x, y), which passes through points (x_i, y_i) and (x_j, y_j).
For each pair of edge points in the image space (x, y), the parameters of the line that passes through both points can be defined. In the parameter space (a, b), the points (a', b') can be drawn. The most frequent points in the parameter space (a, b) and their corresponding lines in the image space (x, y) represent the line that passes through most edge points. However, vertical lines represent a problem, as for such lines the parameter a and therefore the size of the parameter space (a, b) limits to infinity. This problem can be avoided by representing the lines with polar coordinates (r, φ):

$$x \cos\varphi + y \sin\varphi = r\,,$$

For a vertical line ($\varphi = 0°$), r defines the intersection of the line with the x axis. For a horizontal line ($\varphi = 90°$), r defines the intersection of the line with the y axis. In the parameter space (r, φ), each sine curve:

$$x_i \cos\varphi + y_i \sin\varphi = r$$

represents a set of lines that, in the image space (x, y), pass through the point (x_i, y_i). The intersection of two sine curves (r', φ') defines the parameters of the line that passes through the points (x_i, y_i) and (x_j, y_j) in the image space:

$$x \cos\varphi' + y \sin\varphi' = r'.$$

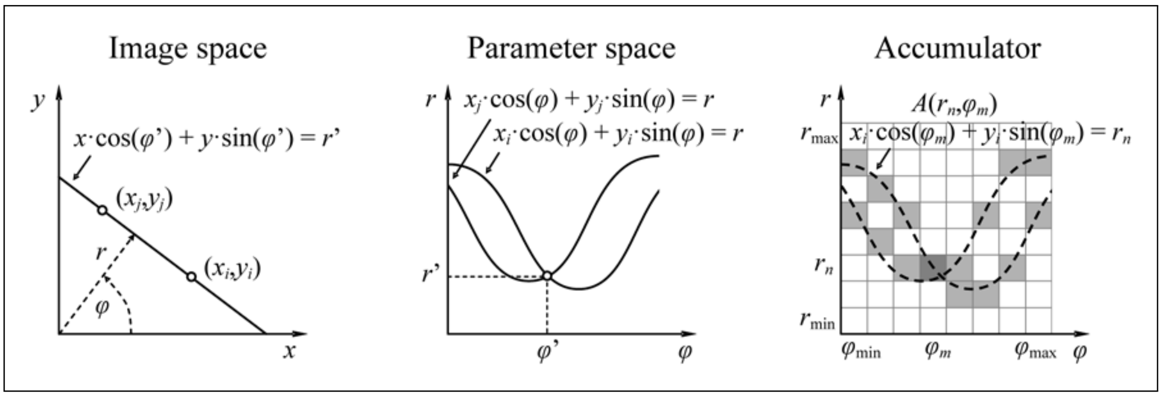


Image space
Parameter space
Accumulator
y
x
x·cos(φ') + y·sin(φ') = r'
(x_j,y_j)
(x_i,y_i)
r
φ
r
x_j·cos(φ) + y_j·sin(φ) = r
x_i·cos(φ) + y_i·sin(φ) = r
r'
φ'
φ
r
A(r_n,φ_m)
r_max
x_i·cos(φ_m) + y_i·sin(φ_m) = r_n
r_n
r_min
φ_min
φ_m
φ_max
φ

One of the advantages of the Hough transform is the possibility to represent the parameter space (r, φ) in a discrete form, therefore obtaining accumulator cells. The range of parameters r and φ can be divided among the accumulator cells:

$$(r_{\min}, r_{\max}) = (0, r_{\text{diag}}) \qquad \text{and} \qquad (\varphi_{\min}, \varphi_{\max}) = (-90^\circ, +180^\circ)\,.$$

The value of each accumulator cell $A(r_n, \varphi_m)$ is first set to 0. For each edge point (x_i, y_i) and for each discrete value φ_m on interval $(\varphi_{\min}, \varphi_{\max})$, the corresponding value of the parameter r is computed. The value of the parameter r is then rounded to the closest discrete value r_n, and the value of the accumulator cell at (r_n, φ_m) is increased by 1:

$$A(r_n, \varphi_m) \quad \leftarrow \quad A(r_n, \varphi_m) + 1\,.$$

After the procedure is finished, the value of each accumulator cell $A(r_n, \varphi_m)$ represents the number of edge points that, in the image space, lie on the line:

$$x \cos \varphi_m + y \sin \varphi_m = r_n\,.$$

The lines that represent the image edge points are defined by searching for those accumulator cells $A(r_n, \varphi_m)$, which values are large enough and represent local accumulator maxima.

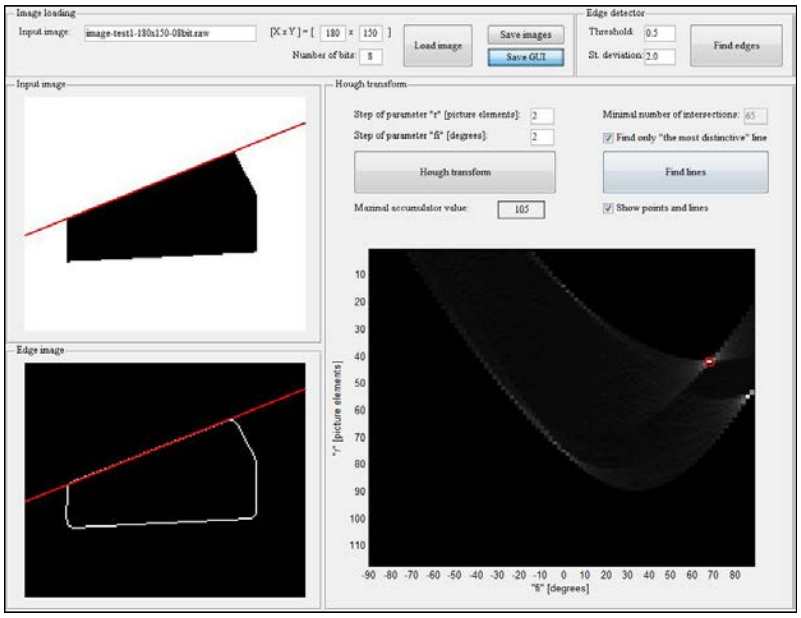


Image loading
Input image: image-test1-180x150-08bit.raw
[X x Y] = [180 x 150]
Number of bits: 8
Edge detector
Threshold: 0.5
St. deviation: 2.0
Hough transform
Step of parameter "r" [picture elements]: 2
Step of parameter "fi" [degrees]: 2
Minimal number of intersections: 65
[✓] Find only "the most distinctive" line
Maximal accumulator value: 105
[✓] Show points and lines

Extract the files from the archive ZIP file `lab10.zip` to a selected folder and run the graphical user interface (GUI) `lab10.m` in Matlab. By using the GUI, you can load the input image, determine the binary image of the edges, display the accumulator of the Hough transform, search for the local maxima of the accumulator, and display the corresponding lines.

Follow the instructions and modify the existing function houghTransform2D2P.m for the Hough transform in two dimensions (2D) for lines in the two-parameter space (r, φ):

```
>>  % Hough transform in 2D for 2 parameters
>>  function [oAcc, rangeR, rangeF] = houghTransform2D2P(iImage, stepR, stepF)

>>      % set blind output values
>>      rangeR = 1;
>>      rangeF = 1;
>>      oAcc = 0;
```

The input and output parameters are defined as:

iImage	The binary image of the edges, where the values different from zero represent edges in the original image: $f(x_i, y_i) \neq 0 \quad \rightarrow \quad (x_i, y_i)$ is an edge point.
stepR	The step of parameter r: $\Delta r = r_{n+1} - r_n$.
stepF	The step of parameter φ: $\Delta\varphi = \varphi_{m+1} - \varphi_m$.
oAcc	The accumulator of the Hough transform: $A(r_n, \varphi_m)$.
rangeR	The vector of parameter values r: $[r_1, r_2, \ldots, r_n, \ldots, r_{\max}]$.
rangeF	The vector of parameter values φ: $[\varphi_1, \varphi_2, \ldots, \varphi_m, \ldots, \varphi_{\max}]$.

Test the algorithm on test images image-test1-180x150.raw and image-test2-180x150.raw of size $X \times Y = 180 \times 150$ picture elements. Test also on image image-neck-170x170.raw of size $X \times Y = 170 \times 170$ picture elements, representing a sagittal magnetic resonance (MR) cross-section of the human neck. (When using the Canny edge detector, you have to appropriately set the threshold and standard deviation of the detector, so that the edge points will represent all interesting structures in the image.)

Vprašanja

Poročilo vaje vsebuje lastnoročno (brez uporabe računalnika) zapisane odgovore na naslednja vprašanja, medtem ko so zahtevane slike natisnjene ter priložene poročilu.

1. Enclose the images of a successful search for the most distinctive line ($n = 1$) in the second test image (image-test2-180x150.raw). What can be concluded about the Hough transform from the results of this experiment? Explain your answer.
2. Enclose the images of a successful search for the most distinctive lines ($n > 1$) in image image-neck-170x170.raw. Set the smallest number of intersections to the maximal value that still allows the display of lines that pass along the anterior and posterior edges of the spinal canal or spinal cord. Write down the chosen parameters of the edge detector (threshold, standard deviation) and the chosen number of intersections in the accumulator.

3. What does the accuracy of line search by Hough transform depend on? Explain your answer.
4. What are the properties of the Hough transform? Explain your answer.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati poročilu. Služijo naj v razmislek ter kot vzpodbuda za boljše razumevanje vsebine.

For the image `image-circles-160x160-08bit.raw` of size $X \times Y = 160 \times 160$ picture elements try to define the first K most distinctive circles by using the Hough transform. (Note: The problem is completely separated from the given GUI, which is adapted for the given line search problem.)