AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING, AUTOMATICS,
COMPUTER SCIENCE AND BIOMEDICAL ENGINEERING

DEPARTMENT OF AUTOMATICS AND BIOMEDICAL ENGINEERING

DADM project documentation - draft

Authors: *SIMENS Healthkare*
Degree programme: *Biomedical engineering*
Supervisor: *PhD. Tomasz Pięciak*

Kraków, 2017

Contents

1. List of changes 7
2. Assumptions 9
3. Structure 11
4. User guide 13
  4.1. Requirements 13
  4.2. Instruction 13
5. Detailed description 15
  5.1. Module 1. MRI reconstruction 15
  5.2. Module 2. Intensity inhomogeneity correction 17
  5.3. Module 3. Non-stationary noise estimation 18
  5.4. Module 4. Non-stationary noise filtering 1 19
  5.5. Module 5. . Non-stationary noise filtering 2 20
  5.6. Module 6. Diffusion tensor imaging 22
  5.7. Module 8. Skull stripping 24
  5.8. Module 9. Segmentation 24
  5.9. Module 10. Upsampling 26
  5.10. Module 11. Brain 3D 29
  5.11. Module 12. Oblique imaging 30
6. Implementation 33
  6.1. Tools 33
  6.2. Module 1. MRI reconstruction 33
  6.3. Module 2. Intensity inhomogenity correction 33
  6.4. Module 3. Non-stationary noise estimation 33
  6.5. Module 4. Non-stationary noise filtering 1 34
  6.6. Module 5. Non-stationary noise filtering 2 34
  6.7. Module 6. Diffusion tensor imaging 34
  6.8. Module 8. Skull stripping 34
  6.9. Module 9. Segmentation 34
  6.10. Module 10. Upsampling 34
  6.11. Module 11. Brain 3D 36

6.12. Module 12. Oblique imaging ... 36

7. Tests ... 37

7.1. Module 1. MRI reconstruction ... 37

7.2. Module 2. Intensity inhomogenity correction ... 37

7.3. Module 3. Non-stationary noise estimation ... 37

7.4. Module 4. Non-stationary noise filtering 1 ... 37

7.5. Module 5. Non-stationary noise filtering 2 ... 37

7.6. Module 6. Diffusion tensor imaging ... 37

7.7. Module 8. Skull stripping ... 37

7.8. Module 9. Segmentation ... 37

7.9. Module 10. Upsampling ... 37

7.10. Module 11. Brain 3D ... 41

7.11. Module 12. Oblique imaging ... 41

7.12. Application ... 41

8. Authors ... 43

1. List of changes

Name	Date	Details
Sylwia Mól	19-Nov-2017	Document created
Sylwia Mól	20-Nov-2017	Structure changed
Sylwia Mól	21-Nov-2017	Chapter "Authors" added, in-out table added
Malwina Molendowska	26-Nov-2017	Description of 1st module added
Eliza Kowalczyk	27-Nov-2017	Description of 9th module added
Karolina Gajewska	27-Nov-2017	Description of 11th module added
Eliza Kowalczyk	28-Nov-2017	Description of 10th module changed
Alicja Martinek	28-Nov-2017	Description of 5th module changed
Mateusz Pabian	29-Nov-2017	Description of 6th module changed
Jacek Fidos	29-Nov-2017	Tools description added
Anna Grzywa	29-Nov-2017	Description of 8th module added
Magdalena Rychlik	29-Nov-2017	Description of 4th module
Michał Kotarba	29-Nov-2017	Description of 12th module added
Magdalena Kucharska	29-Nov-2017	Description of 9th module added
Klaudia Gugulska	30-Nov-2017	Description of 2nd module added
Sylwia Mól	2-Dec-2017	Titles added, numeration changed
Jacek Fidos	3-Dec-2017	Files separation
Eliza Kowalczyk	9-Dec-2017	Enhancement of description of 10th module
Michał Kotarba	9-Dec-2017	Changed in-out for my module
Kacper Turek	9-Dec-2017	Description of 3rd module changed and in-out for my module added
Anna Grzywa	9-Dec-2017	In-out for 8th module added
Sylwia Mól	10-Dec-2017	"Structure" & "Assumptions" chapters changed
Klaudia Gugulska	10-Dec-2017	Update of description of 2nd module
Jacek Fidos	4-Jan-2018	Short SciPy description
Eliza Kowalczyk	14-Jan-2018	Upsampling documentation update
Sylwia Mól	17-Jan-2018	Structure and assumptions changed
Eliza Kowalczyk	17-Jan-2018	Upsampling documentation update

2. Assumptions

The aim of this project is to create the system for MRI data pre-processing and post-processing using Python, SciPy, Cython and Qt5 (detailed description of tools available in chapter *Implementation*, section *Tools*). The project is divided into 11 modules (more: chapter *Structure*).

The application is called SieMRI (the abbreviation for Siemens MRI). It allows to recontruct MRI images, correct intensity inhomogenities, estimate non-stationary noise and remove it using one of two methonds, show diffusion tensors, strip the skull, show specific parts of human brain, upsample the image or visualize brain in 3D or using specific plane. MRI image reconstruction starts automatically when user choose the file with data. Other functionalities are non-obligatory - the user decide if specific functionality should be used.

3. Structure

As mentioned in chapter *Assumptions* the project is divided into 11 modules. Each module includes one system‘s functionality, as follows:

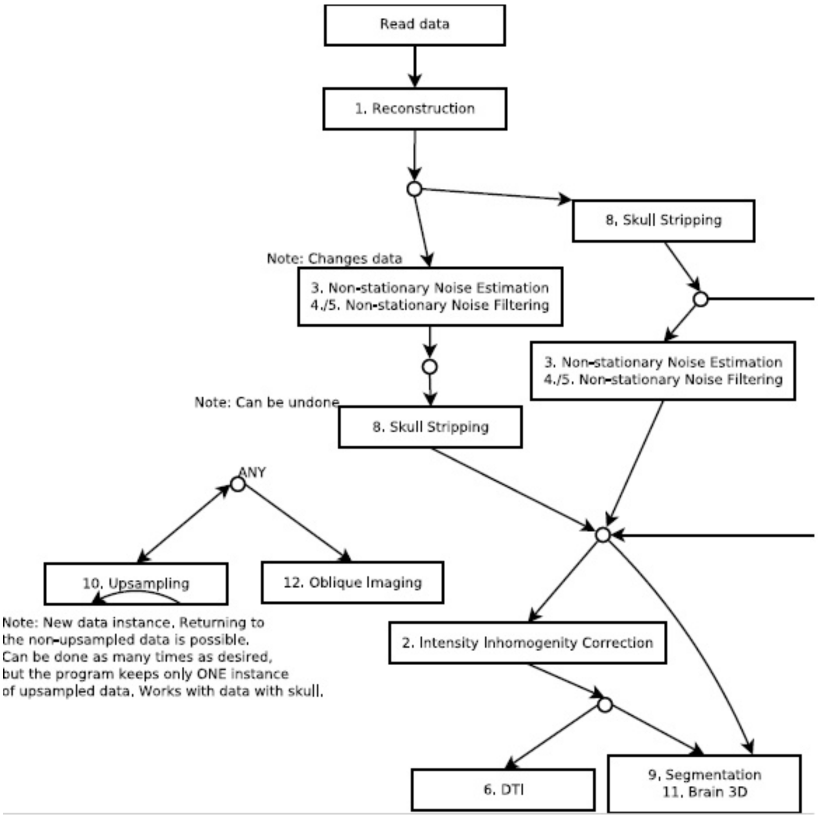
- Module 1.: MRI images reconstruction using Cartesian SENSE algorithm.
- Module 2.: Intensity inhomogenity correction on MRI reconstructed images.
- Module 3.: Non-stationary noise estimation (a homomorphic approach)
- Module 4.:Non-stationary noise filtering #1 with LMMSE filter.
- Module 5.: Non-stationary noise filtering #2 using unbiased non-local means filter.
- Module 6.:Diffusion tensor imaging (WLS, NLS algorithms)
- Module 8.: Skull stripping
- Module 9.: MRI segmentation of the human brain
- Module 10.: Upsampling based on non-local means approach
- Module 11.: Brain 3D visualization using marching cubes method
- Module 12.: Oblique imaging

Module 7. is excluded due to limited human resources and time.

The input and output signals for each module are listed in the table.

Module	Input	Output	Before that module
1	**k**-space signals	**x**-space fully reconstructed data	———
2	reconstruccted image	image with intensity correction	image reconstruction
3	reconstructed image	estimated noise map	reconstruction
4	reconstructed, normalized image	Rician noise-free image	intensity correction
5	reconstructed, normalized image	Rician noise-free image	reconstruction
6	reconstructed, normalized image; gradient-sequence vectors	estimated diffusion tensor 3D 6-channel image	modules 1-5, 8*
8	reconstructed, normalized, filtered image	image without non-brain tissues	modules 1- 4 or 5
9	image without non-brain tissues	segmentated image	
10	320x240 image	640x480 image	denoised data
11	1)segmentated image 2)defined plane, image	1)3D model (e.g. vtkPolyData) 2)cross-section image	1)segmentation 2) denoised data
12	many images from one examination	one image from non-standard perspective	denoised data

Dependencies between modules are shown on the picture. 11th module is divided into two parts, because every part of module requires other pre-procesing activities.


Read data
1. Reconstruction
8. Skull Stripping
3. Non-stationary Noise Estimation
4./5. Non-stationary Noise Filtering
3. Non-stationary Noise Estimation
4./5. Non-stationary Noise Filtering
8. Skull Stripping
ANY
10. Upsampling
12. Oblique Imaging
2. Intensity Inhomogenity Correction
6. DTI
9. Segmentation
11. Brain 3D

Note: Changes data

Note: Can be undone

Note: New data instance. Returning to
the non-upsampled data is possible.
Can be done as many times as desired,
but the program keeps only ONE instance
of upsampled data. Works with data with skull.

Figure 3.1. Dependencies between modules

4. User guide

4.1. Requirements

what user need to use this app - e.g. windows version etc
Currently conda is sufficient.

4.2. Instruction

instructions for user - GUI screens etc

5. Detailed description

5.1. Module 1. MRI reconstruction

The aim of this module is to formulate mathematical algorithm, which enables proper data reconstruction for images obtained with parallel MRI scans. The reconstruction is performed with use of Sensitivity Encoding (SENSE) algorithm in least squares (LS) solution context and Tikhonov regularization method.

Generally, parallel MRI acquisitions are targeted to diminish time needed for data sampling. The usage of multiple coils enabled simultaneous acquisition of signals. A further step, which is acquiring partial data from **k**-space, leads to craved time savings, meanwhile maintaining full spatial resolution as well as contrast at the same time. However, the approach of omitting lines in acquisition step results in data aliasing, i.e. folded images that need further data processing.

To clearly mark out how data is processed in this module, we list following reconstruction steps: i) the application of 2D Fourier Transform transform (2D FFT) to **k**-space data (acquired raw signals) from multiple coils. The result is a set of **x**-space images with folded pixels, ii) the sensitivity maps estimation of coils profiles (the information is needed to properly unfold subsampled data) and iii) the proper unfolding data process with usage of SENSE reconstruction algorithm and its alterations.

The most crucial step in processing is estimation of sensitivity coil profiles as a successful image reconstruction with use of pMRI algorithms highly depends on accurate sensitivity coil assessment. As sensitivity information varies from scan to scan it is impossible to obtain absolute maps. To obtain reliable knowledge, reference scans have to be conducted each time an examination is performed. These low-resolution information helps to estimate coil profiles with use of the many methods i.e. dividing each component coil image by a 'sum of squares' image.

It basic formulation, SENSE algorithm is applied to Cartesian MRI data undersampled uniformly by a factor r (i.e. $r = 2$ means that every other line in **k**-space is skipped). After Fourier transformation, each pixel in **x**-space image received in l-th coil can be seen as weighted sum of r pixels from full FOV, each multiplied by corresponding localized values of maps. The distance between those 'aliased' points in the full FOV is always equal to the desired FOVy value divided by subsampling rate. Obviously, depending on subsampling rate the number of folded pixels changes. Basically, the signal in one pixel at a certain location (x, y) received from l-th component coil image D_l^S with chosen subsampling rate r can be written as:

$$D_l^S(x,y) = S_l(x,y_1)D^R(x,y_1) + S_l(x,y_2)D^R(x,y_2) + ... + S_l(x,y_r)D^R(x,y_r), \tag{5.1}$$

where index l counts from 1 to L (number of coils) and index i counts from 1 to r. Eq.(5.1) can be rewritten as:

$$D_l^S(x,y) = \sum_{i=1}^{r} S_l(x,y_i)D^R(x,y_i) \quad \text{for} \quad l = 1,...,L. \tag{5.2}$$

Including all L coils the above equation can be rewritten in a matrix form:

$$\mathbf{D}^S(\mathbf{x}) = \mathbf{S}(\mathbf{x})\mathbf{D}^R(\mathbf{x}), \tag{5.3}$$

The vector $\mathbf{D}^S(\mathbf{x})$ denotes the aliased coil image values at a specific location **x** = (x, y_i) and has a length of L, $\mathbf{S}(\mathbf{x})$ is a LxR matrix and represents the sensitivities values for each coil at the r superimposed positions and $\mathbf{D}^R(\mathbf{x})$ lists the r pixels from full FOV image to be reconstructed. The closed-form solution of the problem is as follows:

$$\widehat{\mathbf{D}^R(\mathbf{x})} = (\mathbf{S}^H(\mathbf{x})\mathbf{S}(\mathbf{x}))^{-1}\mathbf{S}^H(\mathbf{x})\mathbf{D}^S(\mathbf{x}), \tag{5.4}$$

where $\widehat{\mathbf{D}^R(\mathbf{x})} = [\widehat{D^R(x, y_1)}, ..., \widehat{D^R(x, y_r)}]^T$ and $\mathbf{S}^H(\mathbf{x})$ is the conjugate transpose of the $\mathbf{S}(\mathbf{x})$ matrix. The final reconstruction image is defined as:

$$M(\mathbf{x}) = \left|\widehat{\mathbf{D}^R(\mathbf{x})}\right|. \tag{5.5}$$

The 'unfolding' process can be performed as long as inversion of $\mathbf{S}(\mathbf{x})$ matrix is possible. Therefore, we cannot set the value of subsampling rate exceeding the number of coils L. To restore full FOV data, SENSE algorithm has to be recalled for each pixel in aliased **x**–space image.

A regularization approach is defined as an inversion method that introduces additional information in order to stabilize the solution. This method is beneficial as it roughly matches the desired solution and is less sensitive to perturbations of the data. Tikhonov regularization is a common approach to obtain an inexact solution to a linear system of equations. In particular, the Tikhonov regularized estimate reads as follows:

$$\widehat{\mathbf{D}^R_{reg}} = \underset{\mathbf{D}^R}{\text{argmin}} \left\{ \left\|\mathbf{D}^S - \mathbf{S}\mathbf{D}^R\right\|^2 + \lambda^2 \left\|\mathbf{A}(\mathbf{D}^R - \mathbf{D})\right\|^2 \right\}, \tag{5.6}$$

where λ is a regularization parameter ($\lambda > 0$) and **D** is a prior image known as a regularization image. Selection of the parameter λ and **D** can be performed using different procedures. In this module **A** is assumed to be an identity matrix. The first term provides fidelity to the data and the second introduces prior knowledge (e.x. median filtered initial guess of LS SENSE) about the expected behaviour of $\mathbf{D}^R$. The Tikhonov regularization problem is given by:

$$\widehat{\mathbf{D}^R_{reg}} = \mathbf{D} + (\mathbf{S}^H\mathbf{S} + \lambda\mathbf{A}^H\mathbf{A})^{-1}\mathbf{S}^H(\mathbf{D}^S - \mathbf{S}\mathbf{D}). \tag{5.7}$$

A reasonable value for λ can be picked using many technique, i.e. the L-curve criterion or generalized cross-validation.

Module input: Synthetic MR images are brain MRI slices coming from BrainWeb are normalized to [0-255] (all with intensity non-uniformity INU=0). Only T1- and T2-weighted data is used. The dataset is free of noise and the background areas are set to zero. The slice thickness equals 1 mm. These images are used then to simulate synthetic noisy accelerated parallel Cartesian SENSE MRI data according to following steps (the data simulation is performed with use of eight receiver coils ($L = 8$)): i) simulated sensitivity maps (divided into the ratio 3:1 for real and imaginary parts, respectively) are added to fully-sampled **x**-space data, ii) correlated complex Gaussian noise with different values of standard deviations is added to each coil image, iii) 2D FFT and data subsampling with chosen reduction factor r is performed and iv) 2D iFFT is applied to recover data in **x**-space. Then, data reconstruction process is conducted.

Module output: The output is full resolution reconstructed data performed with two different algorithms: SENSE (LSE) and Tikhonov regularization.

5.2. Module 2. Intensity inhomogeneity correction

The Intensity inhomogeneity of the same tissue varies with the location of the tissue within the image. In other words it refers to the slow, nonanatomic intensity variations of the same tissue over the image domain. It can be due to imaging instrumentation (such as radio-frequency nonuniformity, static field inhomogeneity, etc.) or the patient movement. This artifact is particularly severe in MR images captured by surface coils. Although intensity inhomogeneity is usually hardly noticeable to a human observer, many medical image analysis methods, such as segmentation and registration, are highly sensitive to the spurious variations of image intensities.

The aim of this module is estimation of intensity inhomogeneity field in MR image using surface fitting method. The method fits a parametric surface to a set of image features that contain information on intensity inhomogeneity. The resulting surface, which is usually polynomial or spline based, represents the multiplicative inhomogeneity field that is used to correct the input image.

The steps of this approach are:

1. Extract a background image from the corrupted MRI image, for example, by smoothing the image with a Gaussian filter of a large bandwidth (about 2/3 the size of the MRI image) to filter out all the image details that correspond to highfrequency components.
2. Select few data points from the background image and save their coordinates and graylevel values into a matrix $D = (xi, yi, gi), i = 1, 2, ...n$. It is recommended not to select points from the regions where there is no MRI signal since this regions has no bias field signal.
3. Select a parametric equation for the fitted surface . It is better to fit simple surfaces such as low order polynomial surfaces since they are very smooth and their parameters are very easy to estimate.
4. Estimate the parameters of the surface that best fits the data in matrix D by the method of nonlinear least-squares.
5. Use the fitted equation to generate an image of the bias field signal.
6. Divide the corrupted MRI image by the estimated bias field image in step 5.

Even though different surfaces can reasonably fit the data very well and it is not possible to tell which surface is most likely represents the actual bias field signal, however, in practice the bias signal estimated by fitting a smooth 2-dimensional polynomial surface to a background image can be used effectively to restore the corrupted MRI image.

Fitting of the surface can be done by means of the Levenberg-Marquardt algorithm for nonlinear least squares fitting of a function $f(x, y; a_1, ..., a_m)$ of known form to n data points $(x_1, y_1, g_1), ..., (xn, yn, gn)$. For example, a polynomial surface of degree three can be fitted which has the following equation: $f(x, y; a) = a_1x^3 + a_2y^3 + a_3$, where $a = a_1, a_2, a_3$ is the parameter vector that define the surface. If we substitute the data points in the nonlinear function we get an overdetermined set of equations, i.e.,

$$\left\{ \begin{matrix} g_1 = f(x_1, y_1; a_1, a_2, ..., a_m) \\ . \\ . \\ g_n = f(x_n, y_n; a_1, a_2, ..., a_m) \end{matrix} \right\} \tag{5.8}$$

These equations can be solved to obtain the unknown parameter vector $(a_1, a_2, ..., a_m)$ by minimizing the sum of the squares of the differences between the data and the fitted function

$$Q(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{n}(g_i - f(x_i, y_i; a_1, a_2, ..., a_m))^2 \tag{5.9}$$

Let $r_i(\mathbf{a}) = (g_i - f(x_i, y_i; a_1, a_2, ..., a_m)$, which is the residual vector of point i, then equation(??) can be written as:

$$Q(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{n}(r_1(\mathbf{a}))^2 \tag{5.10}$$

According to the Levenberg-Marquardt algorithm, Eq(??) can be solved iteratively to find the values of the parameters vector (**a**) starting from an initial estimate of the parameter vector $(\mathbf{a}_0)$ using:

$$\mathbf{a}_{i+1} = \mathbf{a}_i - (H + \lambda diag[H])^{-1} \bigtriangledown Q(\mathbf{a}_i) \tag{5.11}$$

where H and $\bigtriangledown$Q($\mathbf{a}_1$) are Hessian matrix and the gradient of Eq. ?? both evaluated at ai, diag is the diagonal elements of the Hessian matrix. At each iteration, the algorithm tests the value of the residual error and adjusts λ accordingly.

List of References
[**2a1**], [**2a2**]

5.3. Module 3. Non-stationary noise estimation

Magnetic Resonance Imaging (MRI) is known to be affected by several sources of quality deterioration, due to limitations in the hardware, scanning times, movement of patients, or even the motion of molecules in the scanning subject. Among them, noise is one source of degradation that affects acquisitions. The presence of noise over the acquired MR signal is a problem that affects not only the visual quality of the images, but also may interfere with further processing techniques such as registration or tensor estimation in Diffusion Tensor MRI.

The aim of this module is estimation of non-stationary noise maps based on MRI image. The module contains a homomorphic method for the non-stationary noise of Gauss and Rice aswell. For the high SNR (signal to noise ratio) algoritm for Gaussian case should be used and for the lower value algorithm for Rician case. In the first case image is corrupted with Gaussian noise which has mean equal to zero spatially-dependent variance $\sigma^2(x)$:

$$I(x) = A(x) + N(x; 0, \sigma^2(x)) = A(x) + \sigma(x) \cdot N(x; 0, 1) \tag{5.12}$$

The goal is to estimate $\sigma(x)$ from the final image $I(x)$.

The variance of the noise $\sigma^2(x)$ slowly differs across the image. Therefore the first step is to remove mean of the image in order to avoid contribution of $A(x)$:

$$I_n(x) = I(x) - E\{I(x)\} = \sigma(x) \cdot N(x; 0, 1) \tag{5.13}$$

where $E\{I(x)\}$ is expected value in each point in the image. It is estimated as local mean using 3×3 window.

The next step is to separate signals $\sigma(x)$ and $N(x)$ by applying the logarithm, but to do so the absolute value of $I_n(x)$ is needed:

$$log\,|I_n(x)| = log\sigma(x) + log\,|N(x)| \tag{5.14}$$

where $log\,|N(x)|$ has its energy distributed in all frequencies and $log\sigma(x)$ is a low frequeny signal.

$log\sigma(x)$ is the interesting part, so to get rid of $log\,|N(x)|$ low pas filter has to be used:

$$LPF\,\{log\,|I_n(x)|\} \approx log\sigma(x) + \delta_N \tag{5.15}$$

where δ_N is a lowa pass resideu of $log\,|N(x)|$ that must be removed from the estimation. Throughout various calculations the final outcome of the LPF was delineated as:

$$LPF\,\{log\,|I_n(x)|\} \approx log\sigma(x) - log\sqrt{2} - \frac{\gamma}{2} \tag{5.16}$$

with γ being Euler-Mascheroni constant.

Taking the exponential of every part of foregoing formula estimator of $\sigma(x)$ can be defined as:

$$\widehat{\sigma(x)} = \sqrt{2}e^{LPF\{log|I(x)-E\{I(x)\}|\}+\gamma/2} \tag{5.17}$$

The Rician case is similar to the described abocv Gaussian case, but a bit more complicated. The signal $A(x)$ is corrupted with complex Gaussian noise with zero mean and spatially-dependent variance $\sigma^2(x)$, which has module following a nonstationary Rician distribution:

$$I(x; A(x), \sigma(x)) = \left|A(x) + N_r(x; 0, \sigma^2(x)) + j \cdot N_i(x; 0, \sigma^2(x))\right| \tag{5.18}$$

To simplify, the dependence of $I(x)$ with $A(x)$ and $\sigma(x)$ are removed.

First couple of steps in Rician case are congenial to the steps taken in Gaussian case. The local mean calulated using 3×3 window is substracted from the image so it can be centered. After that logarithm of the absolute value of centered image is filtered by low passing filter, giving as a result:

$$log\,|I_n(x)| = log\sigma(x) + log\,|G(s_0(x))|\,LPF\,\{log\,|I_n(x)|\} \approx log\sigma(x) + \delta_R \tag{5.19}$$

where σ_R is a low pass residue of $log\,|G(s_0(x))|$. Again after various operation previouse formula presents as:

$$LPF\,\{log\,|I_n(x)|\} \approx log\sigma(x) - log\sqrt{2} - \frac{\gamma}{2} + \varphi(s_0(x)) \tag{5.20}$$

with $\varphi(s_0(x))$ being a Rician/Gaussian correction function. Finally the estimator for $\sigma(x)$ is defined as:

$$\widehat{\sigma(x)} = \sqrt{2}e^{LPF\{log|I(x)-E\{I(x)\}|\}}e^{\gamma/2-\varphi(s_0(x))} \tag{5.21}$$

5.4. Module 4. Non-stationary noise filtering 1

The aim of this module is to remove Rician noise from MR images. If both real and imaginary parts of signal are corrupted with zero-mean uncorrelated Gaussian noise with equal variance, the envelope of magnitude signal will follow a Rician distribution. Many processes allow to remove noise, here the method is used to denoise MR images is the linear minimum square error estimator (LMMSE). The main purpose of LMMSE is to find a closed-form estimator for a signal that follows a Rician distributions. It is more efficient than optimization-based solutions. The estimator uses information of the sample distribution of local statistics of the image such as the local mean, the local variance and the local mean square value. In this method, the true value for each noisy pixel is estimated by a set of pixels selected from a local neighborhood.

The LMMSE estimator for a 2-D signal with Rician distribution is defined:

$$\widehat{A^2_{ij}} = E\{A^2_{ij}\} + C_{A^2_{ij}M^2_{ij}} C^{-1}_{M^2_{ij}M^2_{ij}} (M^2_{ij} - E\{M^2_{ij}\}) \tag{5.22}$$

where A_{ij} is the unknown intensity value in pixel ij, M_ij the observation vector, $C_{A^2_{ij}M^2_{ij}}$ the cross-covarience vector and $C_{M^2_{ij}M^2_{ij}}$ the covarience matrix. If the estimator is simplified to be pointwise, vectors and matrics become scalar values. Then the assuming local ergodicity with a square nighbourhood around the pixel ij, the finally equation for LMMSE is defined:

$$\widehat{A^2_{ij}} = \langle M_ij^2\rangle - 2\sigma_n^2 + K_{ij}(M^2_{ij} - \langle M^2_{ij}\rangle) \tag{5.23}$$

with K_{ij}

$$K^2_{ij} = 1 - \frac{4\sigma_n^2(\langle M_ij^2\rangle - 2\sigma_n^2)}{\langle M_ij^4\rangle - \langle M_ij^2\rangle^2} \tag{5.24}$$

The LMMSE estimator is related to the quality of the estimate of the noise variance σ_n^2. For the noise estimation the mode of the sample mean is used:

$$\widehat{\sigma_n} = \sqrt{\frac{2}{\pi}} mode(\widehat{\mu_1}_{ij}) \tag{5.25}$$

where $\widehat{\mu_1}_{ij}$

$$\widehat{\mu_1}_{ij} = \frac{1}{|\eta_{ij}} \sum_{p:\ \eta_{ij}} I_p \tag{5.26}$$

The use of the LMMSE method should makes the filtering process computationally far more efficient and easier to implement. Also the use of local statistics should decrease estimator dependent of parameters such as the size of window.

Module input: Reconstructed, normalized and corrected data.

Module output: Image without Rician noise.

5.5. Module 5. . Non-stationary noise filtering 2

Magnetic Resonance images are endangered of being corrupted by noise and artifacts. Since they are used as a basis for medical diagnosis their quality has to be at highest possible level. Noise can be dealt with by changing the parameters of images acquisition, however it increases the scanning time, which is undesirable in medical imaging. To overcome this obstacle, post-processing methods like filtering are employed for denoising. In domain of MRI denoising many filters may be used, though here emphasis is put on Unbiased Non-Local Means (UNLM) filter, which is an extension of NLM filter. In order to understand Unbiased version of this algorithm, the basic one has to be presented.

Having image *Y*, the NLM algorithm calculates the new value of point *p* accordingly to the equation:

$$\begin{gathered} NLM(Y(p)) = \sum_{\forall q \in Y} w(p,q)Y(q) \\ 0 \leq w(p,q) \leq 1, \sum_{\forall q \in Y} w(p,q) = 1 \end{gathered} \tag{5.27}$$

It can be seen that value of *p* is calculated as weighted average of pixels in the image (*q*), having fulfilled restrictions from 5.27. To determine before mentioned average the similarity between squared neighbourhoods widows centered around pixels *p* and *q* are calculated. The size of the window can determined by the user, defined by parameter R_{sim}. Equation 5.28 shows how to determine this similarity.

$$w(p,q) = \frac{1}{Z(p)} e^{\frac{d(p,q)}{h^2}} \tag{5.28}$$

Z(p) is the normalizing constant which also uses exponential decay parameter *h* and the weighted Euclidean distance measure for pixels in each neighbourhood, called *d*.

$$Z(p) = \sum_{\forall q} e^{\frac{d(p,q)}{h^2}} \tag{5.29}$$

$$d(p,q) = G_p ||Y(N_p) - Y(N_q)||^2_{R_{sim}} \tag{5.30}$$

In above equation G_p stands for a Gaussian weighting function that has a 0 mean and standard deviation usually equal to 1.

Once NLM filter is fully explained, unbiased extension of it can be examined. It builds on the properties of MRI signal. According to [**5a1**] the magnitude signal of MRI follows a Rician distribution. Furthermore, for low intensity regions the Rician distribution approaches to a Rayleigh one, whilst for high intensity it shifts towards Gaussian. It was investigated that this bias can be handled by filtering the squared MRI image, since it is not longer signal-dependent [**5a1**]. As a consequence the bias, which equals $2\sigma^2$ [**5a3**] can be deleted with ease. The blueprint for UNLM can be summarized in:

- noise estimation - which can be done by calculating standard deviation of background in the image, following formula 5.32, where μ is the mean value of background of squared magnitude image. To distinguish background and the body on the MRI scan the Otsu thresholding method [**5a4**] can be successfully used,
- calculating NLM values for each point of image as in 5.27,
- assessing the unbiased value of each point accordingly to the equation 5.31.

$$UNLM(Y) = \sqrt{NLM(Y)^2 - 2\sigma^2} \tag{5.31}$$

$$\sigma = \sqrt{\frac{\mu}{2}} \tag{5.32}$$

UNML implementation for diffusion weighted data becomes a bit less trivial task. Noise estimation is done in different fashion. From distribution of local averages of voxels in the background, the mode has to be calculated and then corrected with a factor of $\sqrt{\frac{2}{\pi}}$ [**5a2**]. Based on assumption that gradients in similar directions present related behaviours, UNLM for DWI can be formulated as:

$$Y_i(p) = \sqrt{\sum_{j \in \Theta_i^N} \sum_{q \in N_p} w_i^j(p,q) M_j^2(q) - 2\sigma^2} \tag{5.33}$$

Where weights are calculated as for structural data and *M* is a vector containing gray values. Another difference can be found in way the distance between voxels is calculated. It is, the distance in domain of gray levels in the image.

$$d_i^j(p,q) = (M_i(N_p) - M_j(N_q))^T G_p (M_i(N_p) - M_j(N_q)) \tag{5.34}$$

UNLM in this case not only looks for in the same gradient image *i*, but also searches images *j* near the mainly investigated direction i. Working in that manner allows the objective of using joint information (including correlation between channels) to be met [**5a2**].

It is worth mentioning that UNLM filter's performance is highly dependent on parameter values. The optimal values of them were examined in [**5a1**] and same values are adapted in presented implementation. Properly-tuned filter can significantly increase SNR of the scans while preserving body structures.

Module input: Previously reconstructed, normalized and corrected data.

Module output: Image with deleted Rician noise by unbiased non-local means filter.

5.6. Module 6. Diffusion tensor imaging

The aim of this module is to estimate brain tissue diffusion tensor from Diffusion Weighted Images (DWI). DWI are obtained using a different pulse sequence than anatomical MRI images and as such differ in their information content. Concretely, Diffusion Tensor Imaging (DTI) gives an insight into tissue microstructure, probing it using gradient-pulse excited water molecules. This enables indirect measurements of structural orientation and the degree of anisotropy as water molecules diffuse differs between tissues.

In DTI-MRI the measured signal is defined as:

$$S\left(b, \boldsymbol{g}\right) = S_0 exp\left(-b\boldsymbol{g}^{\boldsymbol{T}}\boldsymbol{D}\boldsymbol{g}\right) \tag{5.35}$$

where $S\left(b, \boldsymbol{g}\right)$ is the measured signal, S_0 is the reference signal without diffusion gradient attenuation, b is the diffusion weight scalar, $\boldsymbol{g}$ is the diffusion encoding gradient vector of unit length and $\boldsymbol{D}$ is the diffusion tensor.

The diffusion tensor $\boldsymbol{D}$ is symmetric and describes molecular mobility along each direction:

$$\boldsymbol{D} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \tag{5.36}$$

Furthermore, the Eq. (5.35) can be rewritten as:

$$ln\left(\frac{S\left(b, \boldsymbol{g}\right)}{S_0}\right) = -b\boldsymbol{g}^{\boldsymbol{T}}\boldsymbol{D}\boldsymbol{g} \tag{5.37}$$

Estimating $\boldsymbol{D}$ from the above equation can be done using Weighted Least Squares (WLS) or Nonlinear Least Squares (NLS) algorithms. In order to correctly estimate the diffusion tensor it is necessary to acquire at least seven DWI - one for each direction of diffusion and one in the absence of diffusion gradient.

There exists a couple of strategies of visualizing this high-dimmensional output array containing the estimated tensor in each voxel of the analyzed slice. One of the approaches is to $\boldsymbol{D}$ and project the main eigenvector direction into color space, where customarily x = red, y = green and b = blue. This 2D visualization also allows to perform a sanity check by viewing the main direction of diffusion of white matter, corresponding to fiber longitudal axis, in corpus callosum.

Diagnoalized estimated diffusion tensor can be used to extract additional information about tissue microstructure. This module calculates the following biomarker images:

1. MD (Mean Diffusivity) is the mean of tensor eigenvalues. MD is an inverse measure of membrane density and is very similar for both gray (GM) and white matter (WM) and is higher for corticospinal fluid (CSF).

MD is sensitive to cellularity, edema and necrosis.

$$MD = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} \tag{5.38}$$

2. RA (Relative Anisotropy) exhibits high degree of contrast between anisotropic (WM) and isotropic tissues.

$$RA = \sqrt{\frac{(\lambda_1 - MD)^2 + (\lambda_2 - MD)^2 + (\lambda_3 - MD)^2}{3\,MD}} \tag{5.39}$$

3. FA (Fractional Anisotropy) is a summary measure of microstructural integrity. It is highly sensitive to microstructural changes without considering the type of change.

$$FA = \sqrt{\frac{3}{2}}\sqrt{\frac{(\lambda_1 - MD)^2 + (\lambda_2 - MD)^2 + (\lambda_3 - MD)^2}{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \tag{5.40}$$

4. VR (Volume Ratio)

$$VR = \frac{\lambda_1 \lambda_2 \lambda_3}{MD^3} \tag{5.41}$$

DTI pre-processing pipeline

In order to improve diffusion tensor estimation it is imperative to remove artifacts. In addition to standard MRI pre-processing, one needs to correct for artifacts arising from using of diffusion-gradient pulse sequences and longer acquisition time. While hardware manufacturers try to proactively diminish some of these effects, software processing is still mandatory.

1. Eddy currents and subject motion removal.
   This step attempts to realign (register) all images obtained during diffusion-gradient sequences to one T_2-weighted reference image by finding an affine transform for all diffusion-weighted images.

2. Magnetic susceptibility (local B_0 inhomogeneity) correction.
   This step attempts to map the real B_0 field map using pairs of gradient-weighted images obtained from gradient sequences along the same direction but opposite polarity. The estimated field map can be used to reconstruct images without distortions.

3. Skull stripping (Module 8).

Module I/O

- **Input** - DiffusionData object instance containing 3D ***slice*** (image pixel intensity varying in time with gradient sequence or lack thereof), ***bvalue*** vector corresponding to used diffusion-gradient sequence intensity for each image, ***bvector*** array containing diffusion-gradient sequence unit vector corresponding to ***bvalue*** and ***other*** with general information obtained from data format (most importantly: voxel physical dimensions and time between subsequent echo pulses).

- **Output** - same as input DiffusionData object instance with a new field ***tensor*** containing estimated diffusion tensor data. Biomarker images (MD, RA, FA, VR) can be calculated from ***tensor*** using implemented methods and then displayed on a 2x2 grid.

List of References

[**6_dti_1**], [**6_dti_2**], [**6_dti_3**], [**6_dti_4**], [**6_dti_5**], [**6_dti_6**], [**6_dti_7**], [**6_dti_8**], [**6_dti_9**], [**6_dti_10**]

5.7. Module 8. Skull stripping

The aim of this module is to remove pieces of skull from MRI Image using at pleasure chosen algorithm from the literature. Skull stripping is performed with use of hybrid approach that combines watershed algorithms and deformable surface models.

Preliminary processing to isolate the brain form extra-cranial or non-brain tissues such as e.g. the eye sockets, skin from MRI head scans is commonly referred as skull stripping. Skull stripping methods which are available in the literature are broadly classified into five categories: mathematical morphology-based metods, intensity-based methods, deformable surface-based methods, atlas-based methods, and hybrid methods. Each skull stripping method has their own merits and limitations.

In this module is proposed a hybrid approach to robustly and automatically segment brain from non-brain tissues in T1-weighted MR image. The skull stripping consists of series of sequential steps:

1. Some relevant parameters are estimated from the input image (the coordinates of the brain COG, the average brain radius BR, an upper bound for the intensity of the cerebrospinal fluid CFS – white matter parameters).
2. Watershed algorithm is performed on the intensity image, with global minimum initialized within the cerebral white matter.
3. Deformable surface procedure to recover parts of cortex that may have been erroneously removed in watershed algorithm, using smoothness constraints on the shape of the skull and atlas information.

Watershed algorithms are based on image intensities. Typically, they attempt to locate the local maxima/minima of the norm of the image intensity gradient to segment the image into different connected components. The algorithm proceeds in two steps:

- Watershed transform - the sorting all of voxels (of the gray level inverted image) according to their intensity.
- Post-watershed correction - assessment the validity of the watershed segmentation and retrospectively correct it, because the result image is often inaccurate and nonsmooth, with extracerebral tissues and CSF frequently remaining.

Deformable surface algorithm used the watershed segmentation outputs a segmented volume with most of non-brain tissues removed. A deformable balloon-like template employes this brain volume. An initial template deformation is first completed using global parameters regarding the brain/non-brain border to roughly match the boundary of the brain. Next, the correctness of resulting surface is verified by an atlas-based analysis, and if important structures have been removed, it is modified.

5.8. Module 9. Segmentation

Brain MRI segmentation is an essential task in many clinical applications because it influences the entire analysis. This is because different processing steps rely on accurate segmentation of anatomical regions. For example, MRI segmentation is commonly used for measuring and visualizing different brain structures, for delineating lesions, for analysing brain development, and for image-guided interventions and surgical planning.

MRI Processing To prepare brain MRI for segmentation, it is necessary to perform several preprocessing steps. (Fig. 5.1).

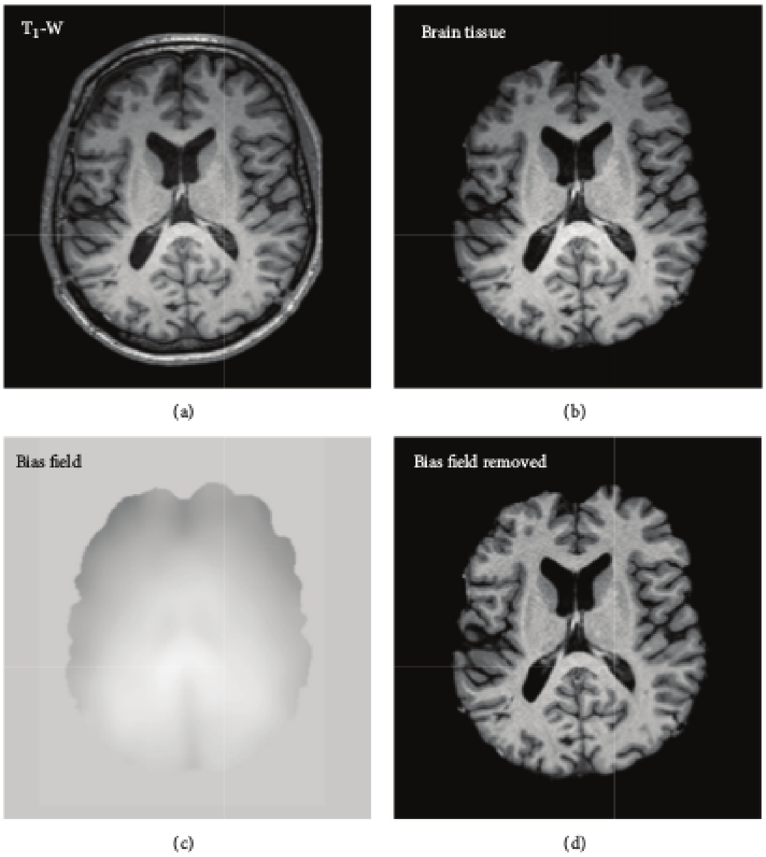


Figure 5.1. Triangulation for the 15 patterns.

The most important steps are: MRI bias field correction, image registration and brain extraction.

Basic An image for segmentation can be defined in 2D space (pixels) or in 3D space (voxels). Each image element is specified by its intensity value and coordinates for pixels (i,j) and for voxels (i,j,k). Intensity values are typically represented by a gray value 0, . . . , 255.

The goal of image segmentation is to divide an image into a set of semantically meaningful, homogeneous, and nonoverlapping regions of similar attributes such as intensity, depth, color, or texture. Thesegmentation result is either an image of labels identifying each homogeneous region or a set of contours which describe the region boundaries.

Fundamental components of structural brain MRI analysis include the classification of MRI data into specific tissue types and the identification and description of specific anatomical structures. The problems of segmentation and classification are interlinked because segmentation implies a classification, while a classifier implicitly segments an image. In the case of brain MRI, image elements are typically classified into three main tissue types: white matter (WM), gray matter (GM), and cerebrospinal fluid (CSF) (Fig. **??**).

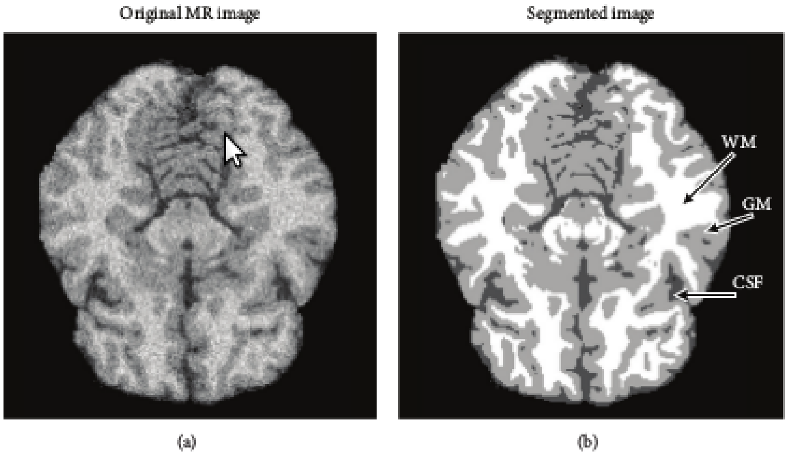


Figure 5.2. Triangulation for the 15 patterns.

One of the most important features for brain MRI segmentation is the intensity of brain tissue. However, intensity-based segmentation algorithms will lead to wrong results when intensity value are corrupted by MRI artifacts.

MRI Segmentation Methods There is no single method that can be suitable for all images, nor are all methods equally good for a particular type of image. For example, some of the methods use only the gray level histogram, while some integrate spatial image information to be robust for noisy environments. Some methods use probabilistic or fuzzy set theoretic approaches, while some additionally integrate prior knowledge (specific image formation model, e.g., MRI brain atlas) to further improve segmentation performance. The segmentation methods, with application to brain MRI, may be grouped as follows:

- manual segmentation;
- intensity-based methods (including thresholding, region growing, classification, and clustering);
- atlas-based methods;
- surface-based methods (including active contours and surfaces, and multiphase active contours);
- hybrid segmentation methods.

List of References
[**09a1**]

5.9. Module 10. Upsampling

Image spatial resolution is limited by several factors. There are for example time of acquisition, organ motions, signal to noise ratio or medical equipment which do not meet the specialized requirements. Specialists usually make relatively small number of 2-D slices. It takes less time but it also results in large slice thickness and spacing between slices. Another example are voxel-wise analysis. The method requires all image volumes of one subject to be analysed in one space. If they are not acquired from the same location (because of the organ motions) the interpolation must be involved.

Interpolation is a method of constructing new points based on the existing ones. In other words finding a value of a new point in High Resolution (HR) image based on points in Low Resolution (LR) pictures.

In classical interpolation techniques pixels in LR data y can be related to the corresponding x values of HR data:

$$y_p = \frac{1}{N} \sum_{i=1}^{N} x_i + n, \text{ where}$$

y_p is the pixel of LR image at location p,
x_i is each one of the N High Resolution pixels contained within this LR pixel,
n is some additive noise.

In classical interpolation techniques the High Resolution pixels x are calculated as weighted average of existing LR pixels y.

$$x_i = \frac{1}{M} \sum_{i=1}^{M} w_{ij} * y_j, \text{ where}$$

w are weighted calculated as Euclidean distance between M existing surrounding LR pixels and the coordinates of new ones.

The biggest problem is that to find High Resolution data from the Low Resolution values. Unfortunately, there is an infinite number of values that meet that condition. Interpolation methods can be divided into three basic techniques.

The first group are the most common ones, like linear or spline-based interpolation. These techniques assume that it is possible to count the value of a new point by determination some kind of generic function. The main disadvantage is that they are correct only for images of homogeneous regions. As is well known, brain consists of grey substance, white substance and cerebral spinal fluid, so above-mentioned methods are not appropriate for MRI images interpolation. The second one is Super Resolution technique. It is commonly used to increase image resolution on functional MRI (fMRI) and Diffusion Tensor Imaging (DTI). The method is based on acquisition of multiple Low Resolution (LR) images of the same object. It is time consuming and not adequate for clinical applications.

The last but not least method is non-local patch-based technique which is based on self-similarity of a single image. It is possible to improve resolution by extracting information from a single image instead of acquiring several pictures.

First of all, there is a constraint which says that the original LR image has to be equal to the downsampled version X of the reconstructed image. It is named subsampling consistency constraint which is written as:

$$y_p - \frac{1}{N} \sum i = 1^N X_i = 0$$

To make the above equation real, the noise has to be removed. The presence of noise has to be minimalized on the LR image by applying a filter, for example MNLM3D filter for 3D MR images based on Non-Local Means filter. The filter removes the noise effectively without affecting the image structure.

The aim of this module is to increase MRI image resolution by upsampling. The input data is a single 320x240 image. After the processing it is going to be twice as big. To be more specific 640x480 pixels. The process can be named as double upsampling.

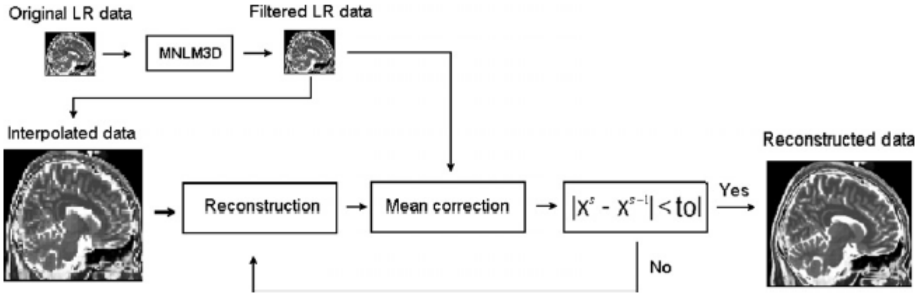

Original LR data
MNLM3D
Filtered LR data
Interpolated data
Reconstruction
Mean correction
$|x^s - x^{s-1}| < tol$
Yes
No
Reconstructed data

Figure 5.3. Diagram of the patch-based non-local algorithm [**9art1**]

.

In a nutshell, the first step of the algorithm is to divide every pixel of LR image into more pixels. Then the patch is designed. It is a rectangle which will be the area of interest. The pixels that are inside the area are taken into account during calculation the values of new points. After that an appropriate estimator is used to correctly calculate the values of new pixels. Every pixel has to be classified into one of three groups (grey substance, white substance or cerebral spinal fluid).

In order to show how the algorithm is designed the following pictured were generated using exemplary data. At the beginning there is an image 256x256 pixels.

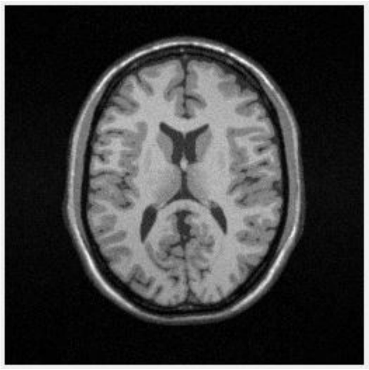


Figure 5.4. DICOM Low Resolution image 256x256

.

The first step is to add new pixels to increase the resolution M times horizontally and N times vertically. The pixels are equal to zero. After this process an appropriate estimator has to be chosen to make a decision about the values of new points.

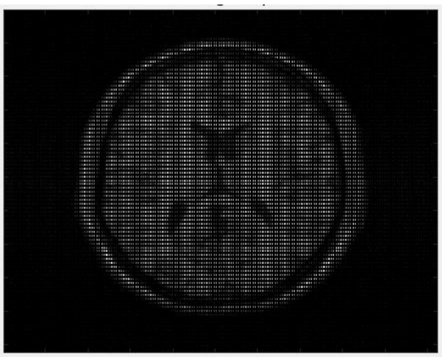


Figure 5.5. DICOM image with new points with N=2 and M=2

.

Patch-based non-local regularization is used to reconstruct new voxels. The voxel under study is reconstruced using all similar voxels which are placed inside the patch. Contrary to classical techniques which use local neighbourhood of the reconstructed voxel, the proposed method uses the only relevant information, the context of the voxel.

Another step in the algorithm diagram (Fig. 5.3) is mean correction. It has to be applied to ensure that the downsampled reconstructed HR X image is equal to the original y LR one. Thus the local mean value of X fits with the value of the y by adding the corresponding offset.

$$X = X + NN * (D(X) - y), \text{ where}$$

D is an operator that downsamples HR into original LR,
NN is a Nearest Neighbour operator that interpolates LR image to HR.

The Regularization and Correction steps are repeated until no important difference between two consecutive reconstructions. Tolerance is determined by Mean Absolute Difference .

List of References
[**9art1**] [**9art2**]

5.10. Module 11. Brain 3D

To prepare tree dimension visualization of the cerebral cortex is used algorithm of marching cubes.

The input data is multiple 2D slices of MR image. The marching cubes algorithm create a polygonal representation of constant density surfaces from a 3D array of data. To select the cerebral cortex is used output data from segmentation made in module 8. The space of the image is divided into a regular grid of cubes. In each iteration one cube is considered. At each vertex of cube is determined how the surface intersects this cube. The density value and compared with the limit value - surface constant. If the data value is bigger than suface constant, one is assinged to a cube's vertex. There are 256 combinations of cube orientation relative to the surface, but we can distinguish 15 basic patterns, that repeat as symmetrical reflections, produces all possibilities (Fig. **??**). If all values are less than the constant value, then the cube does not form any polygon. Otherwise, the edges of the polygon are defined (by linear interpolation) at the edges that intersect the surface. Using central differences, a unit normal at each cube vertex is calculated and then normal to each trangle vertex is interpolated. The output of the algorithm is the triangle vertices and vertex normals.

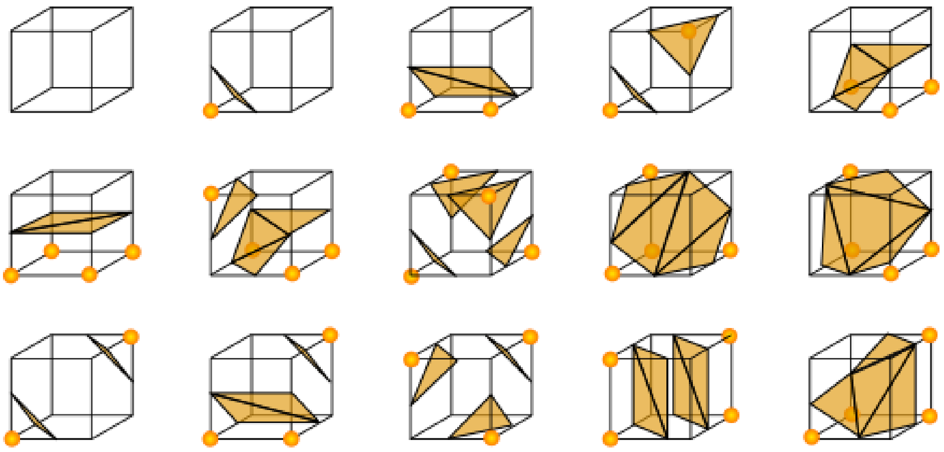


Figure 5.6. Triangulation for the 15 patterns.

To visualization the model, obtained by marching cubes, the VTK library is used, which enables building the three-dimension model.

The second part of this module includes visualization of the brain's cross-section on arbitrarily defined plane.

There are three primary imaging planes that are performed in medical imaging:

- axial plane, which is any plane that divides the body into superior and inferior parts, roughly perpendicular to spine.
- sagittal plane, which is any imaginary plane parallel to median plane.
- coronal plane, which is any vertical plane that divides the body into anterior and posterior sections.

The MRI produces two-dimensional images that consist of slices of brain's and is usually performed in axial plane. To receive images in the remaining planes, linear interpolation is used

5.11. Module 12. Oblique imaging

The literature to this module is as useful as nipples on men. Everything is about inventing how to realize point 1. of the list.

Oblique Imaging is a technique to create non-perspective projections from 3D or multiple 2D images.

In order to create oblique image it is essential to:

- choose two angles under which the plane will be inclined,
- create a matrix of points that this plane consists of,
- from existing points pick those, which will be used in the image,
- interpolate points that are not existing.

Type of interpolation can vary, but in this project interpolation based on mean will be used. To interpolate one pixel mean of all pixels around him with given proximity is taken.

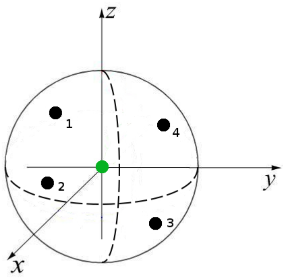


Figure 5.7. Visualization of pixels taken to interpolate

6. Implementation

6.1. Tools

Python - The main programming language of this project is Python. The utilized language version is **Python 3.5.4** (last "bugfix" release of Python 3.5).

VTK - The Visualization Toolkit (VTK) is an open-source software system for 3D computer graphics, modeling, image processing, volume rendering, scientific visualization, and information visualization. The project uses version **7.0**, as it is compatible with python 3.5 (compatibility assured by menpo project[1]).

Cython - Cython is an optimising static compiler for both the Python programming language and the extended Cython programming language. It allows to represent the most computionally complex operations with more efficient code. This project uses version **0.26.1**.

PyQT - PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python. This project uses its version **5.6** to create Graphical User Interface (GUI) for the application.

SciPy - SciPy is a collection of open source packages, which serves to import .mat MATLAB data into NumPy arrays and perform computations on them.

Conda - Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. This project uses **Anaconda3** distribution to provide standarized enviroment with all of the required packages for our developers.

6.2. Module 1. MRI reconstruction

-code

6.3. Module 2. Intensity inhomogenity correction

-code

6.4. Module 3. Non-stationary noise estimation

-code

---

[1] http://www.menpo.org/

6.5. Module 4. Non-stationary noise filtering 1

-code

6.6. Module 5. Non-stationary noise filtering 2

-code

6.7. Module 6. Diffusion tensor imaging

-code

6.8. Module 8. Skull stripping

-code

6.9. Module 9. Segmentation

-code

6.10. Module 10. Upsampling

The implementation began with making a decision about the input data. The input image has to be 2 dimensional without a noise. The uploaded image will be considered as a 2 dimensional array of double values. Other parameters that will be needed are vertical and horizontal extensions. It has to be a total number. The size of the input array will be multiplied by the number. For example, the image 256x256 pixels with extensions equal to 3 will be 768x768 pixels as an output. The function needs also the window size. In one iteration, the function will take into consideration only pixels witch are inside the window. The loop will stop only when the whole image will be interpolated. To see the results the *plotting* parameter has to be a boolean *True* value.
The below picture shows the block diagram of the algorithm (Fig. 6.1) with comments added.

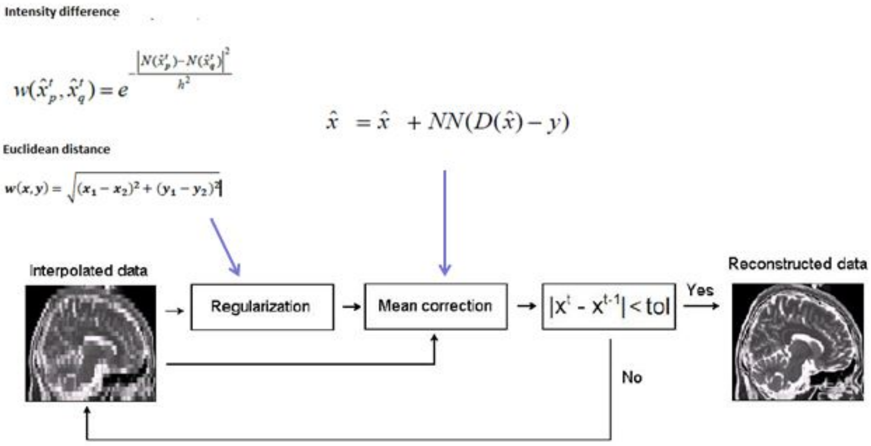

Intensity difference
$w(\hat{x}'_p,\hat{x}'_q)=e^{-\frac{|N(\hat{x}'_p)-N(\hat{x}'_q)|^2}{h^2}}$
Euclidean distance
$w(x,y)=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$
$\hat{x}=\hat{x}+NN(D(\hat{x})-y)$
Interpolated data
Regularization
Mean correction
$|x^t - x^{t-1}| < tol$
Yes
No
Reconstructed data

Figure 6.1. Algorithm block diagram with comments

.

The following steps of the implementation will be discussed:

1. **Initial interpolation**
   To form the input image into the output with chosen size the initial interpolation is needed. As showed in [**9art1**] the spline interpolation is used. In short, spline interpolation is an interpolation where interpolant is a special kind of piecewise polynomial called spline. It is recommended, because of the fact that the method makes small error even with low degree polynomials for the spline. Taking into consideration the edges the extrapolation was applied. To be more specific, the last row *n* is equal to the previous one *n-1*. The same with the first row, the first column and the last column of the input. To show the results, the extensions are set as 2. The input data is 256x256 pixels. The undermentioned picture (Fig. 6.2) shows the result of the initial interpolation. The scale is included so it is easy to see that the size is twice as big. Now, the zeros (Fig. 5.5) have nonzero values, so the next steps can be made.

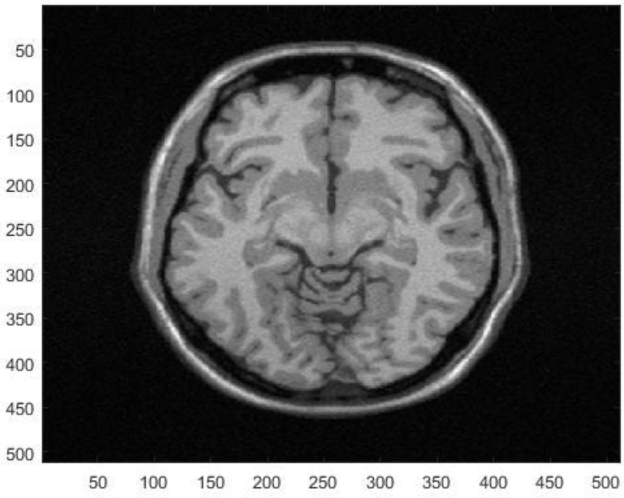

50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500

Figure 6.2. The image after initial interpolation 512x512 pixels

.

2. **Regularization**

   The main functionality is the regularization step. The function makes the square window which contains some pixels. In one iteration of a loop, only the pixels which are inside the window are considered. Next, the window moves and takes another pixels. It is repeated till the end of the image. The main goal is to determine of the weight which will be used at the end.

   The first weight is calculated as the difference between the pixels intensities.

   $$w_1(x,y) = e^{\frac{|N(x)-N(y)|^2}{h^2}}, \text{ where}$$

   $N(x)$, $N(y)$ are window and image intensity values,

   h is a level, which is equal to the half of the standard deviation of the input image.

   The next part of the whole weight is an Euclidean distance weight. It is simply calculated as the Euclidean distance between every pixel in the window and every pixel in the image. It is based on the below equation.

   $$w_2(x,y) = \frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}}{h^2}, \text{ where}$$

   h is a level, which is equal to the half of the standard deviation of the input image.

   The total weight is calculated as $w_{total} = w_1 * w_2$. The output image is determined as the result of the sum of the multiplication of the weight and the window, divided by the sum of the weight.

3. **Mean correction**

   The mean correction step is a place in the function where the correctness of the equation $X = X + NN * (D(X) - y)$. It was abovementioned, that the downsampled HR has to be equal to the input data.

4. **Tolerance checking** At the beginning it was established that only the 2D images will be taken into consideration and the upsampling function will be applied only once, so the step is excluded. There were experiments to upsample the data several times, but the result were not satisfied.

6.11. Module 11. Brain 3D

-code

6.12. Module 12. Oblique imaging

-code

7. Tests

7.1. Module 1. MRI reconstruction

-tests for module 1

7.2. Module 2. Intensity inhomogenity correction

7.3. Module 3. Non-stationary noise estimation

7.4. Module 4. Non-stationary noise filtering 1

7.5. Module 5. Non-stationary noise filtering 2

7.6. Module 6. Diffusion tensor imaging

7.7. Module 8. Skull stripping

7.8. Module 9. Segmentation

7.9. Module 10. Upsampling

Results

To visualize the upsampling functionality several output with different parameters are showed (Fig. 7.1, 7.2).

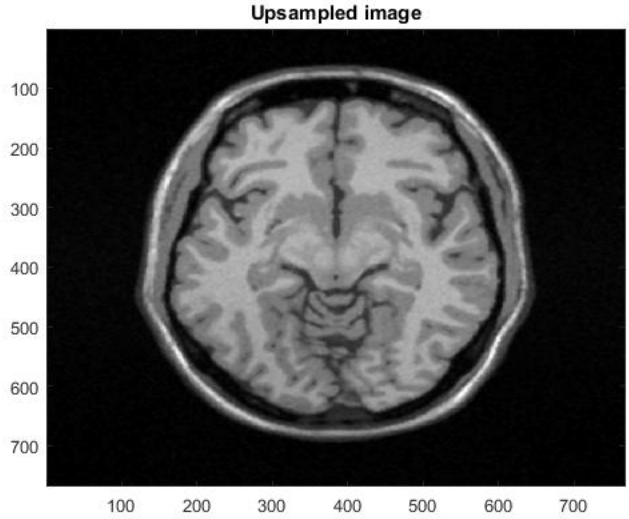

Upsampled image
100
200
300
400
500
600
700
100
200
300
400
500
600
700

Figure 7.1. The upsampled image with extensions equal to 3 and window equal to 2

.

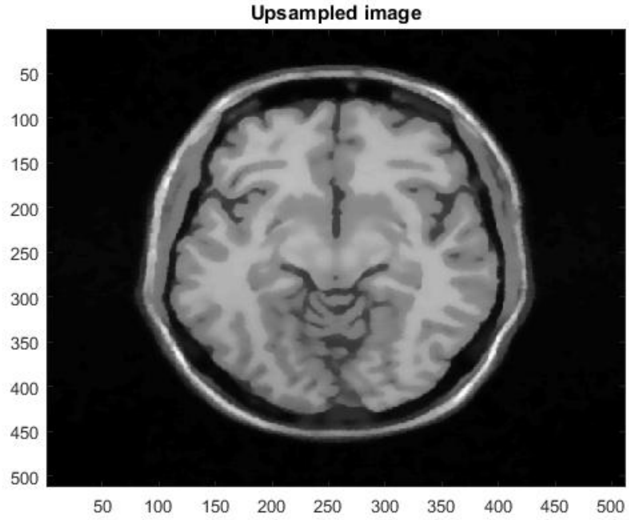

Upsampled image
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500

Figure 7.2. The upsampled image with extensions equal to 2 and window equal to 5

.

To better see the comparison (Fig.7.3, 7.4)

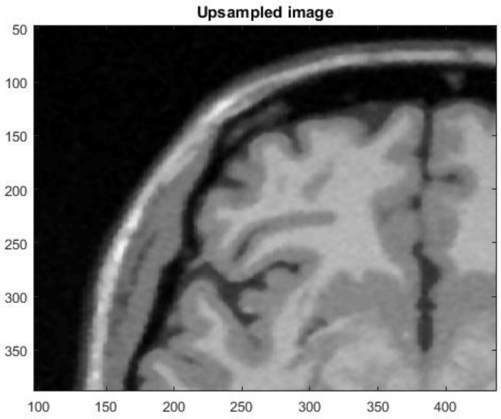

Upsampled image
50
100
150
200
250
300
350
100
150
200
250
300
350
400

Figure 7.3. The upsampled image with extensions equal to 3 and window equal to 2

.

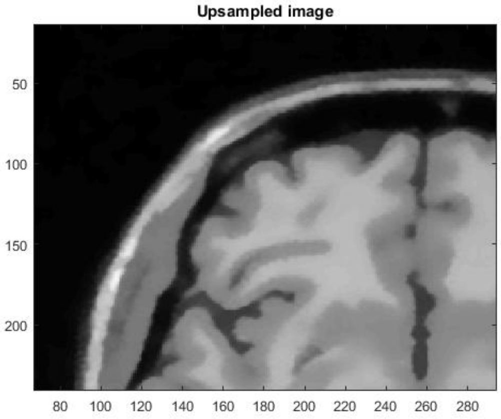

Upsampled image
50
100
150
200
80
100
120
140
160
180
200
220
240
260
280

Figure 7.4. The upsampled image with extensions equal to 2 and window equal to 5

.

Comparison with the other methods

To confirm the better functionality of the proposed method the results will be compared. The other are Linear, Nearest Neighbour, Cubic and Spline interpolation methods (Fig 7.5, 7.6, 7.7, 7.8). To better see the comparison (Fig. 7.9).

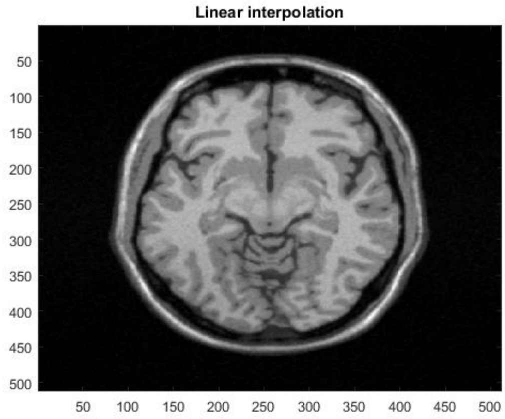

Linear interpolation
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500

Figure 7.5. Linear interpolation

.

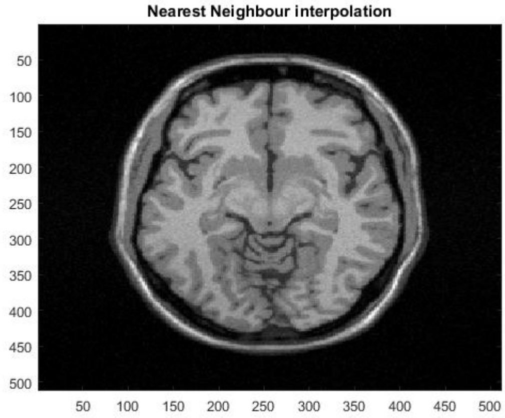

Nearest Neighbour interpolation
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500

Figure 7.6. Nearest neighbour interpolation

.

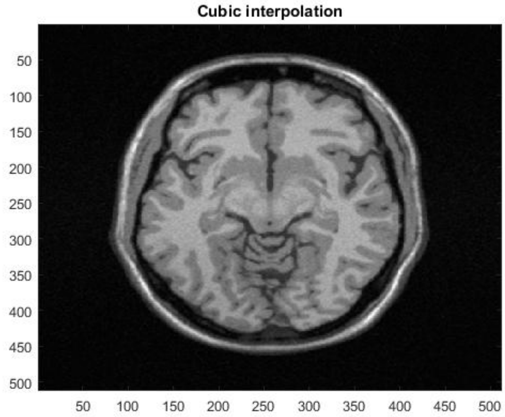

Cubic interpolation
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500

Figure 7.7. Cubic interpolation

.

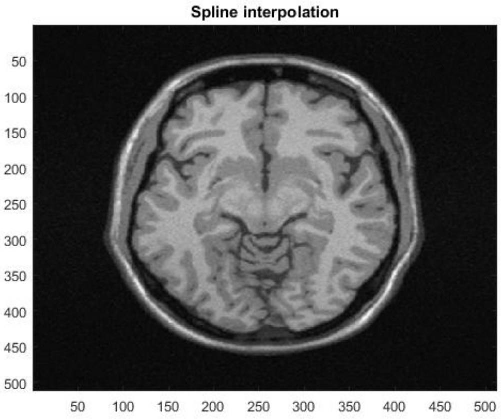

Spline interpolation
50
100
150
200
250
300
350
400
450
500
50 100 150 200 250 300 350 400 450 500

Figure 7.8. Spline interpolation

.

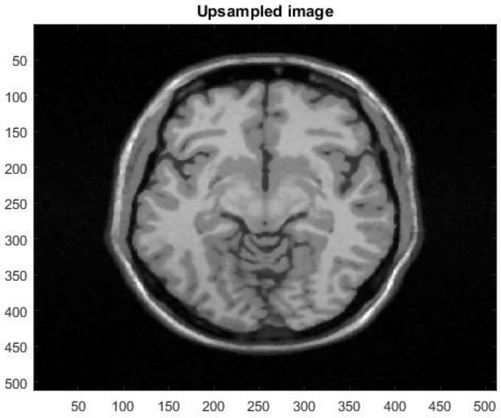

Upsampled image
50
100
150
200
250
300
350
400
450
500
50 100 150 200 250 300 350 400 450 500

Figure 7.9. Proposed method

.

Conclusions

In conclusion, implementation of a new interpolation method met with success. Unquestionably, the proposed method gives better results than other popular interpolation methods.

7.10. Module 11. Brain 3D

7.11. Module 12. Oblique imaging

7.12. Application

-whole app tests

8. Authors

Authors of this project are students of Biomedical Engineering, AGH UST, Krakow, Poland.

Name	Role
Sylwia Mól	Project Manager
Jacek Fidos	Software architect
Maciej Gryczan	GUI engineer
Adrian Stopiak	Vizualization engineer
Malwina Molendowska	1st module developer
Klaudia Gugulska	2nd module developer
Kacper Turek	3rd module developer
Magdalena Rychlik	4th module developer
Alicja Martinek	5th module developer
Mateusz Pabian	6th module developer
Anna Grzywa	8th module developer
Magdalena Kucharska	9th module developer
Eliza Kowalczyk	10th module developer
Karolina Gajewska	11th module developer
Michał Kotarba	12th module developer

List of Figures

3.1 Dependencies between modules . . . 12

5.1 Triangulation for the 15 patterns. . . . 25
5.2 Triangulation for the 15 patterns. . . . 26
5.3 Diagram of the patch-based non-local algorithm [**9art1**] . . . 28
5.4 DICOM Low Resolution image 256x256 . . . 28
5.5 DICOM image with new points with N=2 and M=2 . . . 29
5.6 Triangulation for the 15 patterns. . . . 30
5.7 Visualization of pixels taken to interpolate . . . 31

6.1 Algorithm block diagram with comments . . . 35
6.2 The image after initial interpolation 512x512 pixels . . . 35

7.1 The upsampled image with extensions equal to 3 and window equal to 2 . . . 38
7.2 The upsampled image with extensions equal to 2 and window equal to 5 . . . 38
7.3 The upsampled image with extensions equal to 3 and window equal to 2 . . . 39
7.4 The upsampled image with extensions equal to 2 and window equal to 5 . . . 39
7.5 Linear interpolation . . . 40
7.6 Nearest neighbour interpolation . . . 40
7.7 Cubic interpolation . . . 40
7.8 Spline interpolation . . . 41
7.9 Proposed method . . . 41