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

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 10th module added
Eliza Kowalczyk	28-Nov-2017	Description of 9th 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

2. Assumptions

about the app - the aim, what you can do here etc.

3. Structure

dependences (tree), modules‘ descriptions

Module	Input	Output	Befo
1	**k**-space signals	**x**-space fully reconstructed data	
2			
3			
4			
5	reconstructed, normalized image	Rician noise-free image	re
6			
8	image	image without non-brain tissues	
9	320x240 image	640x480 image	de
10	1)segmentated image 2)defined plane, image	1)3D model (e.g. vtkPolyData) 2)cross-section image	1)segmenta
11			

4. User guide

4.1. Requirements

what user need to use this app - e.g. windows version etc

4.2. Instruction

instructions for user - GUI screens etc

5. Detailed description

5.1. Module 1

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.(**??**) 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 $\mathbf{x} = (x, y_i)$ 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:

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

where λ is a regularization parameter ($\lambda > 0$) and $\mathbf{D}$ is a prior image known as a regularization image. Selection of the parameter λ and $\mathbf{D}$ can be performed using different procedures. In this module $\mathbf{A}$ 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

-detailed description of module, algorithm etc, NO CODES, only theoretical!

5.3. Module 3

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 to estimate the spatial dependent pattern of the variance of noise in SENSE reconstructed images. For this to work some additional information must be known beforehand, such as the sensitivity maps of each receiver coil. In the background of a SENSE MR image, where the SNR is zero, the Rician PDF (Probability density function) simplifies to a (non-stationary) Rayleigh distribution, whose second order moment is defined as:

$$E\left\{M^2\left(x\right)\right\} = 2 \cdot \sigma_R^2 \cdot \left(x\right) \tag{5.8}$$

Since σ_R^2 is *x*-dependent, $E\left\{M^2\left(x\right)\right\}$ will also show a different value for each *x* position.

Let us assume that each coil in the *x* -space is initially corrupted with uncorrelated Gaussian noise with the same variance σ_n^2 and there is a correlation between coils ρ so that matrix σ becomes:

$$\Sigma = \sigma_n^2 \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{pmatrix} = \sigma_n^2 \left(I + \rho\left[1 - I\right]\right) \tag{5.9}$$

with *I* the $L \times L$ identity matrix and *1* a $L \times L$ matrix of 1's. For each *x* value, we define the global map

$$G_{w_i} = W_i^* \left(I - \rho\left[1 - I\right]\right) W_i, \quad i = 1, \cdots, r \tag{5.10}$$

Global map $G_w(x)$ can be easily inferred from G_{w_i} values. Note that $G_w(x)$ is strongly related fo the g-factor, so the first equation becomes

$$E\left\{M^2\left(x\right)\right\} = 2 \cdot \sigma_n^2 G_w\left(x\right) \tag{5.11}$$

and

$$\sigma_n^2 = \frac{E\left\{M^2\left(x\right)\right\}}{2G_w\left(x\right)} \tag{5.12}$$

By using this regularization we can assure a single σ_n^2 value for all the points in the image. We can now define a noise estimator based on the local sample estimation of the second order moment:

$$\left\langle M^2(x)\right\rangle_x = \frac{1}{|\eta(x)|} \sum_{p \in \eta(x)} M^2(p) \tag{5.13}$$

with $\eta(x)$ a neighborhood centered in x. $\left\langle M^2(x)\right\rangle_x$ is know to follow a Gamma distribution whose mode is $\sigma_n^2(|\eta(x)|-1)/|\eta(x)|$. Then

$$mode\left\{\frac{\left\langle M_L^2\right\rangle_x}{G_w(x)}\right\}=2\sigma_n^2\frac{|\eta(x)|-1}{|\eta(x)|}\approx 2\sigma_n^2 \tag{5.14}$$

when $|\eta(x)|\gg 1$. The estimator is the defined as

$$\widehat{\sigma_n^2}=\frac{1}{2}mode\left\{\frac{\left\langle M_L^2(x)\right\rangle_x}{G_w(x)}\right\} \tag{5.15}$$

and consequently the noise in each pixel is estimated as

$$\widehat{\sigma_R^2}=\frac{1}{2}mode\left\{\frac{\left\langle M_L^2(x)\right\rangle_x}{G_w(x)}\right\}G_w(x) \tag{5.16}$$

This estimator is only vaild over the background pixels. However, no segmentation of these pixels is needed: the use of the mode allows us to work with the whole image. On the other hand, to carry out the estimation, the sensitivity map of each coil and the correlation between coils must be known beforhand. These parameters are needed for the SENSE encoding, and thus, they can be easily obtained.

5.4. Module 4

-detailed description of module, algorithm etc, NO CODES, only theoretical!

5.5. Module 5

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:

$$\begin{aligned} NLM(Y(p)) &= \sum_{\forall q\in Y} w(p,q)Y(q) \\ 0\le w(p,q)\le 1, &\sum_{\forall q\in Y} w(p,q)=1 \end{aligned} \tag{5.17}$$

It can be seen that value of *p* is calculated as weighted average of pixels in the image (*q*), having fulfilled restrictions from **??**. 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 **??** shows how to determine this similarity.

$$w(p,q)=\frac{1}{Z(p)}e^{\frac{d(p,q)}{h^2}} \tag{5.18}$$

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*.

$$Z(p) = \sum_{\forall q} e^{\frac{d(p,q)}{h^2}} \tag{5.19}$$

$$d(p,q) = G_p ||Y(N_p) - Y(N_q)||^2_{R_{sim}} \tag{5.20}$$

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 **??**, 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 **??**,

– assessing the unbiased value of each point accordingly to the equation **??**.

$$UNLM(Y) = \sqrt{NLM(Y)^2 - 2\sigma^2} \tag{5.21}$$

$$\sigma = \sqrt{\frac{\mu}{2}} \tag{5.22}$$

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:

$$Y_i(p) = \sqrt{\sum_{j \in \Theta_i^N} \sum_{q \in N_p} w_i^j(p,q) M_j^2(q) - 2\sigma^2} \tag{5.23}$$

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.

$$d_i^j(p,q) = (M_i(N_p) - M_j(N_q))^T G_p (M_i(N_p) - M_j(N_q)) \tag{5.24}$$

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

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:

$$S\left(b, \boldsymbol{g}\right) = S_0 exp\left(-b\boldsymbol{g}^{\boldsymbol{T}}\boldsymbol{D}\boldsymbol{g}\right) \tag{5.25}$$

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:

$$\boldsymbol{D} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \tag{5.26}$$

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

$$ln\left(\frac{S\left(b, \boldsymbol{g}\right)}{S_0}\right) = -b\boldsymbol{g}^{\boldsymbol{T}}\boldsymbol{D}\boldsymbol{g} \tag{5.27}$$

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.

$$MD = \frac{\lambda_1 + \lambda_2 + \lambda_3}{3} \tag{5.28}$$

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

$$RA = \sqrt{\frac{(\lambda_1 - MD)^2 + (\lambda_2 - MD)^2 + (\lambda_3 - MD)^2}{3\,MD}} \tag{5.29}$$

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

$$FA = \sqrt{\frac{3}{2}}\sqrt{\frac{(\lambda_1 - MD)^2 + (\lambda_2 - MD)^2 + (\lambda_3 - MD)^2}{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \tag{5.30}$$

4. VR (Volume Ratio)

$$VR = \frac{\lambda_1\lambda_2\lambda_3}{MD^3} \tag{5.31}$$

DTI pre-processing pipeline[1]

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**]

[1] Scientific articles enumerated by Mr Pięciak mention these pre-processing steps as necessary for proper diffusion tensor estimation.

5.7. Module 8

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

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: $y_p = \frac{1}{N}\sum_{i=1}^{N} x_i + n$, 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 and n is some additive noise.

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.

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.

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).

5.9. Module 10

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.

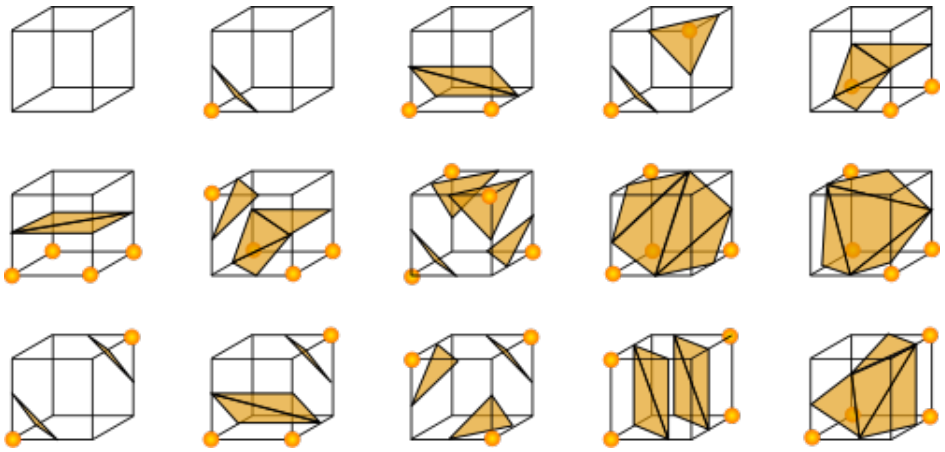

Figure 5.1. Triangulation for the 15 patterns.

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

5.10. Module 11

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.

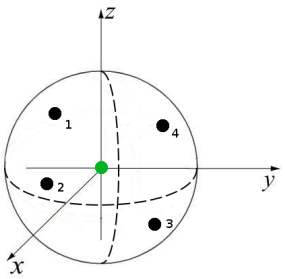

Figure 5.2. Visualization of pixels taken to interpolate

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.

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

-code

6.3. Module 2

-code

6.4. Module 3

-code

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

6.5. Module 4

-code

6.6. Module 5

-code

6.7. Module 6

-code

6.8. Module 8

-code

6.9. Module 9

-code

6.10. Module 10

-code

6.11. Module 11

-code

7. Tests

7.1. Module 1

-module 1 tests

7.2. Module 2

7.3. Module 3

7.4. Module 4

7.5. Module 5

7.6. Module 6

7.7. Module 8

7.8. Module 9

7.9. Module 10

7.10. Module 11

7.11. 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	9th module developer
Karolina Gajewska	10th module developer
Michał Kotarba	11th module developer

List of Figures