**UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL**
**ESCOLA DE ENGENHARIA**
**DEPARTAMENTO DE ENGENHARIA ELÉTRICA**

RÔMULO BERGESCH RAFFIN

# MÉTODO PARA DETECÇÃO DE CORRUPÇÃO DE IMAGENS DTI EM TEMPO REAL

Porto Alegre
2016

Rômulo Bergesch Raffin

# Método para Detecção de Corrupção de Imagens DTI em Tempo Real

Trabalho apresentado para a conclusão de graduação em Engenharia Elétrica na Universidade Federal do Rio Grande do Sul.

Orientadora: Léia Bernardi Bagesteiro

Porto Alegre

2016

Rômulo Bergesch Raffin
Método para Detecção de Corrupção de Imagens DTI em Tempo Real/ Rômulo Bergesch Raffin. – Porto Alegre, 2016-
59 p. : il. (algumas color.) ; 30 cm.

Orientadora: Léia Bernardi Bagesteiro

Trabalho de Conclusão de Curso – Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Departamento de Engenharia Elétrica, 2016.

1. Ressonância Magnética. 2. Imagem por Tensor de Difusão. 3. Detecção de Corrupção. I. Léia Bernardi Bagesteiro. II. Universidade Federal do Rio Grande do Sul. III. Escola de Engenharia. IV. Título

CDU 02:141:005.7

Rômulo Bergesch Raffin

# Método para Detecção de Corrupção de Imagens DTI em Tempo Real

Trabalho apresentado para a conclusão de graduação em Engenharia Elétrica na Universidade Federal do Rio Grande do Sul.

Aprovado em: Porto Alegre, 14 de dezembro de 2016:

---

**Léia Bernardi Bagesteiro**
Orientadora

---

**Altamiro Amadeu Susin - UFRGS**
Membro da Banca

---

**Alexandre Rosa Franco - PUCRS**
Membro da Banca

---

**Dario F. G. de Azevedo - PUCRS**
Membro da Banca

Porto Alegre
2016

*Este trabalho é dedicado àqueles*
*que vem tentando fazer do Brasil*
*um país melhor.*

# Agradecimentos

Em primeiro lugar, agradeço aos meus familiares. Sobretudo aos meus pais, Ney e Sandra, por me estimularem a ser curioso e por me ensinarem que os bens mais preciosos que alguém pode ter são saúde e educação. Agradeço ao Alexandre Franco por estar sempre disponível nos momentos de dúvidas. Ao Instituto do Cérebro do Rio Grande do Sul por disponibilizar os exames DTI para este trabalho. À minha orientadora, Léia Bagesteiro, por ser sempre atenciosa e fazer críticas construtivas.

*"Se vi mais longe*
*foi por estar de pé*
*sobre o ombro de gigantes."*
*(Isaac Newton)*

# Resumo

A fim de possibilitar a detecção de corrupção de exames de Imagem por Tensor de Difusão (DTI) em tempo real, desenvolveu-se um método que consiste de ajustar os espectros dos *slices* das imagens a gaussianas bidimensionais e utilizar os dados resultantes desse processo para classificação via *Support-Vector Machine* (SVM). Foram utilizados 6652 *slices* de cinco sujeitos diferentes. Em 1000 iterações nas quais dividiu-se o total de imagens em 90% para treino e 10% para teste, obteve-se uma taxa de acerto média de 96,60% com desvio padrão de 0,67%. Para implementação em tempo real é recomendado o processamento das imagens intercaladamente utilizando dois núcleos de processamento em paralelo.

**Palavras-chaves**: Ressonância Magnética. DTI. Detecção de Corrupção. *Machine Learning.*

# Abstract

In order to enable real-time corruption detection of Diffusion Tensor Imaging (DTI) exams, a method consisting of fitting the images' slices' spectra and using the data resulting from this process for classification via Support-Vector Machine (SVM) was developed. A set of 6652 slices from five different subjects was used. After 1000 iterations in which the images were divided in two groups (90% for training and 10% for testing), the mean success rate was 96.60% with a standard deviation of 0.67%. For real-time implementation, parallel processing of interweaved slices using two processing cores is recommended.

**Key-words**: Magnetic Resonance Imaging. DTI. Corruption Detection. Machine Learning.

# Lista de ilustrações

# Sumário

# Introdução

O sujeito que vai ser submetido a um exame médico por imagem tem que seguir um procedimento como: ir para o hospital, aguardar na sala de espera, trocar sua roupa por um jaleco, guardar seus pertences em um armário, ser dirigido à sala de exame, escutar as orientações do procedimento, deitar-se na maca, posicionar sua cabeça na bobina de aquisição, ter seus membros presos para evitar movimento, esperar a máquina funcionar com bastante barulho para, finalmente, poder vestir novamente suas roupas e ir embora.

Algumas vezes, acontece de a pessoa já ter voltado para sua casa ou trabalho e receber uma ligação dizendo que houve problema na aquisição das imagens e que deve retornar ao hospital para coletá-las novamente. Ela precisa reorganizar sua agenda para repetir todo o procedimento de exame já descrito. A nova coleta de imagens gera prejuízo tanto para o paciente – transporte e ausência no trabalho, por exemplo – quanto para o hospital, que precisa reagendar o sujeito no normalmente apertado cronograma da máquina sem ser paga pelo novo exame. Visando diminuir esse tipo de ocorrências, pensou-se na possibilidade de utilizar *software* para detectar automaticamente corrupções em imagens de ressonância magnética por tensores de difusão durante a própria aquisição.

Como não foi encontrada uma aplicação conforme desejada, decidiu-se pelo desenvolvimento de um método para tal de maneira que pudesse ser implementado independentemente de modelo e fabricante de aparelho. Optou-se pela utilização de *softwares* livres para que o sistema possa ser disponibilizado com a maior abrangência possível.

## Objetivo

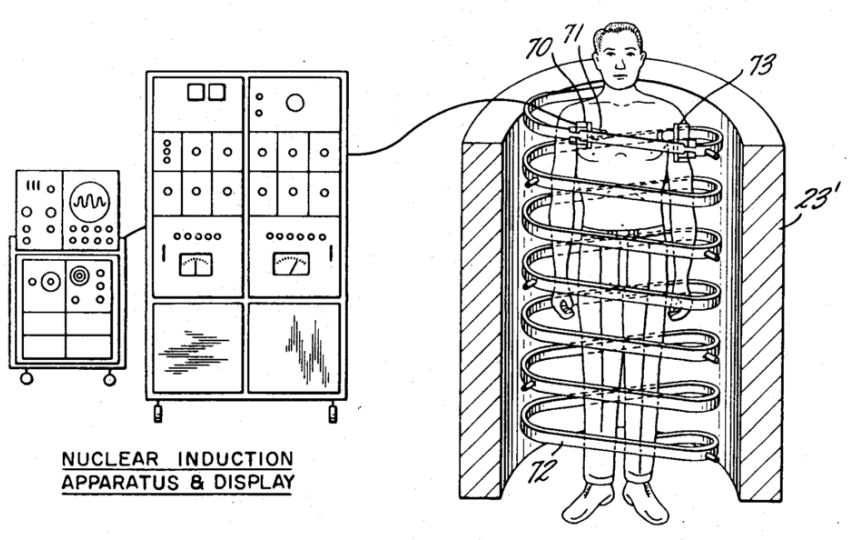
Desenvolver um método que possibilite detectar imagens DTI corrompidas durante o processo de aquisição do exame. Esse método deve utilizar *software* livre e que possibilite instalação simples independentemente do fabricante do *scanner* de ressonância magnética.

# 1 Fundamentação Teórica

## 1.1 Imagem por Ressonância Magnética (MRI)

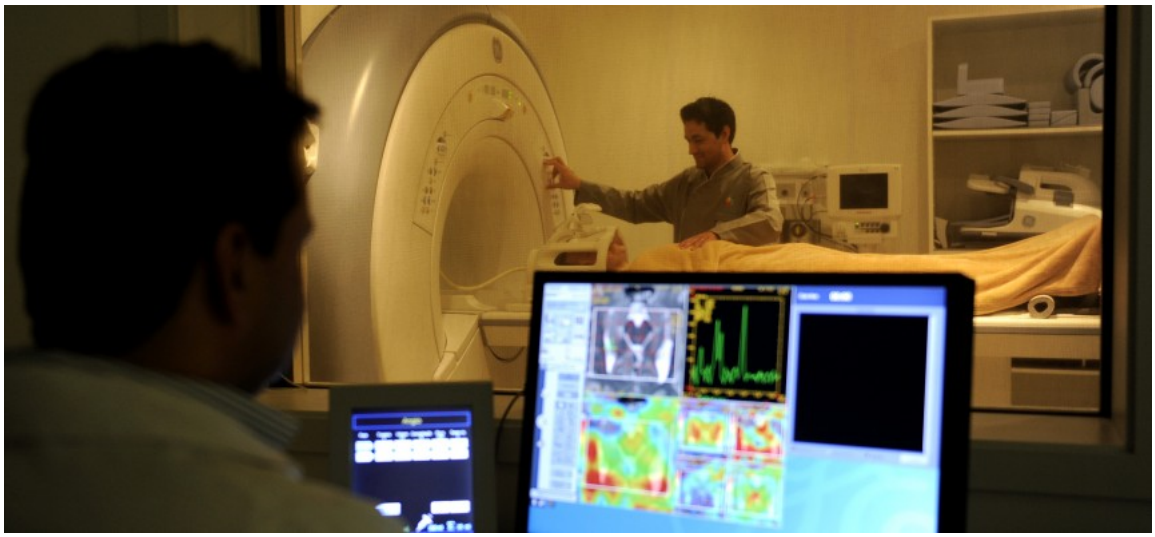
Uma sequência de estudos no campo da Física – que incluem o spin, o momento magnético, o Efeito Zeeman e a Frequência de Larmor – culminaram na técnica para medir momentos magnéticos nucleares proposta por Isidor Isaac Rabi, em 1937, chamada de Ressonância Magnética Nuclear (RMN) (SILVA, 2004). Em 1959, J. R. Singer fez a conexão entre RMN e os estudos de Linus Pauling sobre as características magnéticas da hemoglobina (PAULING; CORYELL, 1936) e sugeriu que a ressonância magnética poderia ser uma maneira não-invasiva de medir fluxo sanguíneo in vivo. A primeira aplicação de fato da técnica veio com Raymond Damadian, que descobriu em 1971 que alguns tumores em ratos poderiam ser distinguidos de tecido normal com o uso de ressonância magnética e fez, em 1972, um pedido de patente para um "aparato e método para detectar câncer em tecido" (DAMADIAN, 1972), cujo equipamento mostrado na Figura 1 tem bastante semelhança com *scanners* contemporâneos, como o que pode ser visto na Figura 2. No entanto, são geralmente creditados como pais da Imagem por Ressonância Magnética Paul Lauterbur e Peter Mansfield, que, independentemente, desenvolveram métodos para distinguir sinais de RMN oriundos de diferentes fontes e a excitação seletiva de regiões, características fundamentais das tomografias, e, por isso, receberam em conjunto um Prêmio Nobel (MCROBBIE et al., 2006).

Figura 1: Aparato para detectar câncer em tecido




Fonte: Damadian (1972)

Figura 2: Equipamento contemporâneo de MRI



Fonte: Instituto do Cérebro do Rio Grande do Sul (InsCer)

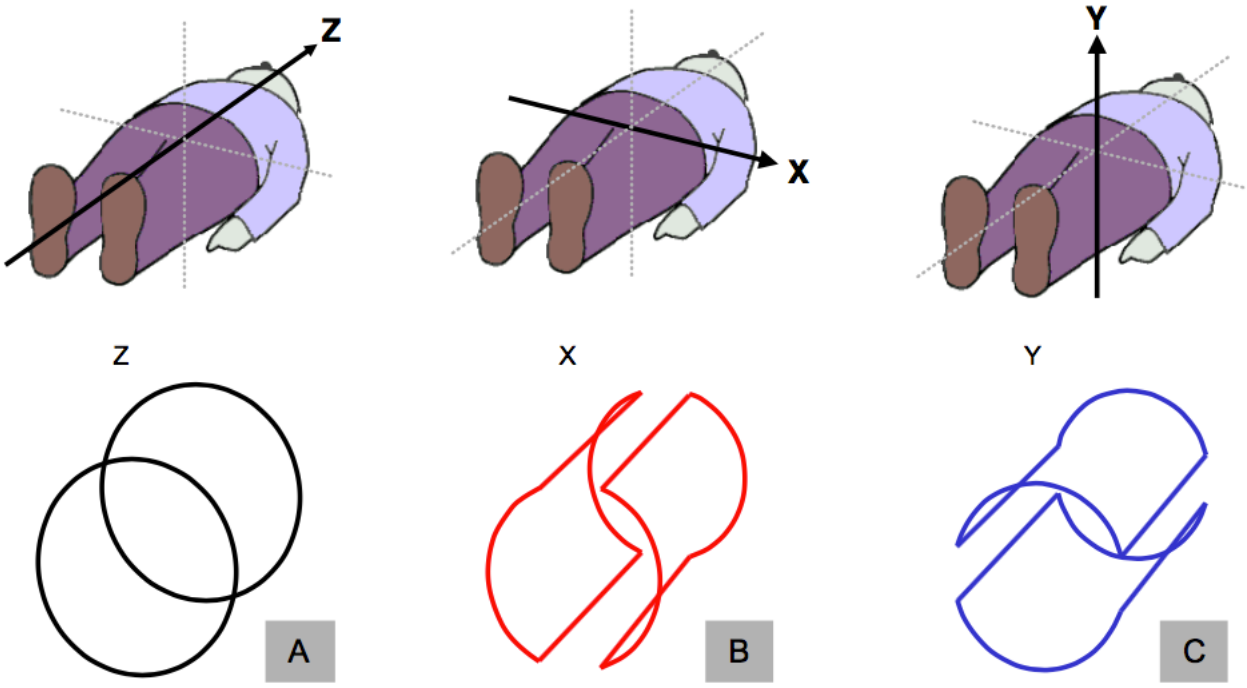
As estruturas na natureza tendem ao seu estado de menor energia potencial. Se as estruturas em questão são núcleos atômicos sujeitos a um campo magnético bastante intenso, a tendência é de que seus spins se alinhem com ele. No entanto, esse alinhamento não é perfeito, fazendo com os spins fiquem girando ao redor do eixo do campo principal, o que é chamado de precessão. As partículas precedem à Frequência de Larmor, que é dada por

$$f = \gamma \cdot B, \tag{1.1}$$

com $\gamma = 42MHzT^{-1}$ para prótons e B sendo o campo magnético a que estão sujeitas. Essa é a base para os *scanners* de ressonância magnética (BROWN et al., 2014).

São quatro os campos magnéticos necessários em MRI. O campo magnético principal (chamado de $B_0$) tem seu eixo alinhado com o do cilindro do equipamento e do sujeito a ser analisado, como pode ser visto na Figura 3(A), e é praticamente uniforme. Sua intensidade fica na ordem das unidades de Tesla, sendo que a maior parte dos aparelhos atuais é de 1,5 ou 3,0T. Para efeito de comparação, o campo magnético na superfície terrestre varia entre 30 e 60 $\mu T$, cerca de 100.000 vezes mais fraco. Há também um campo magnético gradiente nessa direção, gerado por uma Bobina de Maxwell (cujo formato está na parte inferior da Figura 3(A)). Nas Figura 3(B) e Figura 3(C), estão sendo mostrados outros dois eixos (ortogonais a $B_0$) nos quais outros dois campos magnéticos gradientes (que variam linearmente no espaço) agem. Para que isso seja possível na geometria cilíndrica do *scanner* de MRI, são utilizadas Bobinas de Golay, que estão ilustradas na Figura 3(B) e Figura 3(C) abaixo dos eixos em que atuam. Os campos gradientes têm intensidade entre 30 e 45 mT/m.

Figura 3: Campos magnéticos e formatos de bobinas




Fonte: Blink (2010)

Com essa disposição de campos, a Equação de Larmor pode ser reescrita como:

$$f(x, y, z) = \gamma \cdot B(x, y, z) \tag{1.2}$$

Dessa forma, pode-se ver que prótons localizados em diferentes pontos do cilindro vão ter frequências de precessão diferentes.

Partículas subatômicas só absorvem energia em frequências específicas quantizadas. Sabendo, então, a distribuição das frequências de precessão dos prótons de interesse, pode-se excitá-los em grupos por meio de radiofrequência. Geralmente, faz-se uma divisão em fatias (em inglês, *slices*) ao longo do eixo z para serem excitadas uma a uma. Bobinas de excitação são responsáveis por emitir radiação eletromagnética num espectro correspondente às frequências definidas pelos campos gradientes nas direções x e y. A absorção de energia pelos prótons faz com que seus momentos magnéticos sejam defletidos para um estado de maior energia e, depois, relaxam e retornam ao estado inicial. Essa dinâmica dos momentos magnéticos dos prótons induz em bobinas receptoras uma corrente elétrica.

A corrente elétrica induzida nas bobinas é um sinal composto de diversas frequências, devido aos gradientes em cima das partículas excitadas. Diz-se que imagens formadas por esses sinais estão no "espaço $k$". Portanto, utiliza-se ferramentas de análise espectral (Fourier) para que esse sinal seja transformado em uma imagem que mostra distribuição espacial da matéria no plano excitado, o tipo de imagem com que se está mais acostu-

mado. Cada fatia é subdivida em blocos chamados de *voxels*, que são como *pixels*, mas em três dimensões. A concatenação das sucessivas fatias forma um volume.

O significado do que se vê depende de uma combinação de fatores que inclui intensidade de campo magnético, ordem de aplicação, duração temporal, entre outros, que é chamada de sequência de pulsos. Pode-se, então, descrever uma imagem de ressonância magnética como:

$$v(x, y, z) = I(p) \tag{1.3}$$

que significa que cada voxel no volume é definido por suas coordenadas espaciais e seu valor (ou intensidade) é dependente da sequência de pulsos utilizada.

## 1.2 Imagem por Tensores de Difusão (DTI)

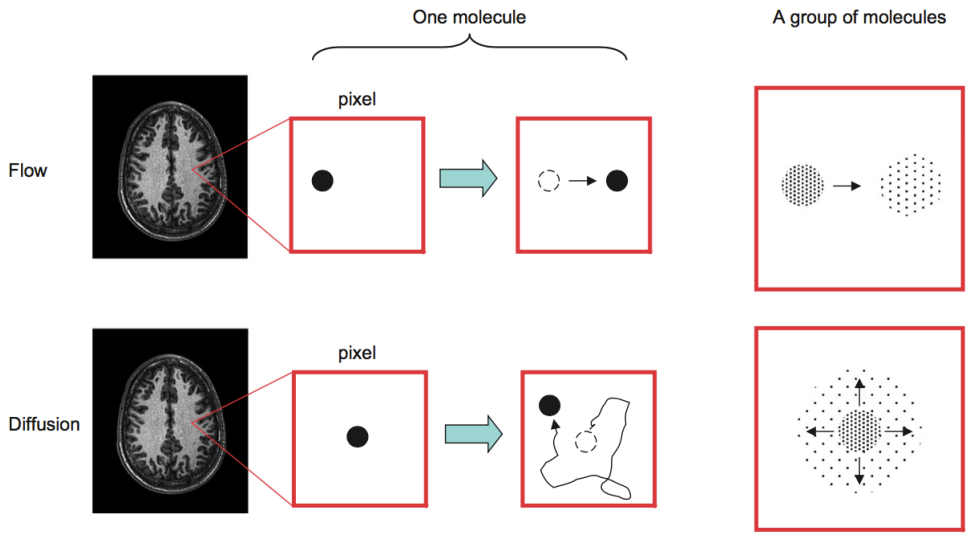
Uma família de sequências de pulsos faz com que o sinal da ressonância magnética seja proporcional à difusão de moléculas de água. Ou seja, a imagem gerada é ponderada por difusão (*Diffusion Weighted Imaging* - DWI). Tomando um ponto de vista nas dimensões de um voxel (na ordem de $1mm^3$), pode-se diferenciar "difusão"de "fluxo"da seguinte maneira:

- O **fluxo** de um grupo de moléculas de água é o movimento ordenado das moléculas em uma certa direção. O centro de massa do grupo também se movimenta nessa direção. Ilustração na parte superior da Figura 4.
- A **difusão** de um grupo de moléculas de água consiste de um movimento desordenado em todas as direções possíveis. O centro de massa do grupo permanece imóvel (na média). Chama-se também de movimento aleatório das partículas, ou Browniano. Ilustração na parte inferior da Figura 4.

Se as moléculas de água têm seu movimento restrito por barreiras físicas, obviamente, a difusão não ocorre em todas as direções. No cérebro, esse fenômeno é observado principalmente na substância branca, onde a água se difunde ao longo dos tratos axonais. Essa é a premissa básica da Imagem por Tensores de Difusão (DTI, do inglês *Diffusion Tensor Imaging*)(MORI; TOURNIER, 2014).

As sequências de pulso que ponderam por difusão o fazem somente numa direção. Portanto, os voxels de um volume de DWI indicam o quão intensa foi a difusão das moléculas de água em uma direção estabelecida pela combinação de gradientes. A técnica do DTI envolve adquirir uma sucessão de DWIs cada uma sensibilizada para uma direção diferente. Dessa forma, sabendo o quão restrita é a difusão da água em cada direção, é possível calcular um tensor de difusão em cada *voxel*, o que permite estimar a estrutura cerebral.

Figura 4: Fluxo versus difusão




Fonte: Mori e Tournier (2014)

O tensor de difusão tem o seguinte formato:

$$\overline{D} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix}, \tag{1.4}$$

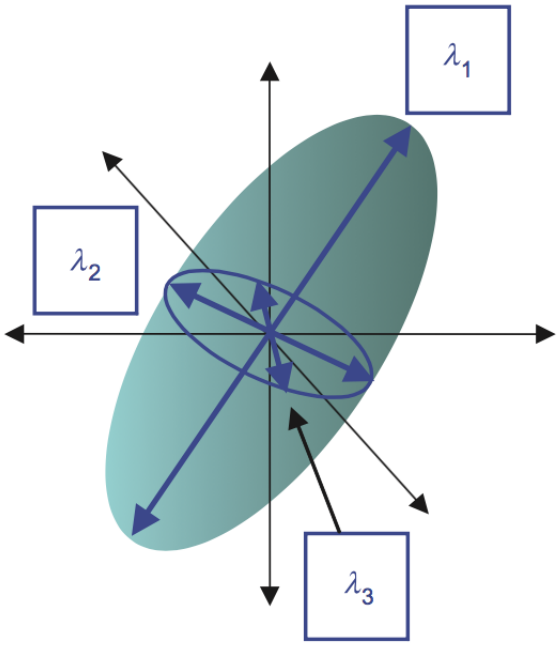
onde os subscritos indicam as direções de difusão. Nota-se que é um tensor simétrico ($D_{xy} = D_{yx}$, por exemplo), o que faz com que se tenha somente 6 parâmetros independentes no tensor. Fica mais interessante representar o tensor por meio dos seus autovalores e autovetores (cujos índices estão em ordem decrescente): $\lambda_1, \lambda_2, \lambda_3, \boldsymbol{v}_1, \boldsymbol{v}_2$ *e* $\boldsymbol{v}_3$. Considerando que os autovetores indicam as direções principais de difusão e os autovalores indicam a intensidade da difusão nessas direções, pode-se interpretar esse conjunto de valores como sendo um elipsoide de difusão, conforme ilustrado na Figura 5.

Uma aplicação dos tensores de difusão é a tractografia, que é a renderização gráfica da estimativa de organização de substância branca cerebral. Essa técnica possibilita uma abordagem *in vivo* da anatomia, que pode ser utilizada para identificar mudanças causadas por tumores, por exemplo, como ilustra a Figura 6.

Os tensores de difusão também são utilizados por meio de métricas derivadas como a anisotropia fracionada (FA, do inglês *Fractional Anisotropy*), que é a indicação do quão direcional é a difusividade intravoxel. A FA é muitas vezes utilizada para estudos entre grupos. Pode-se calcular a FA por meio da seguinte equação:

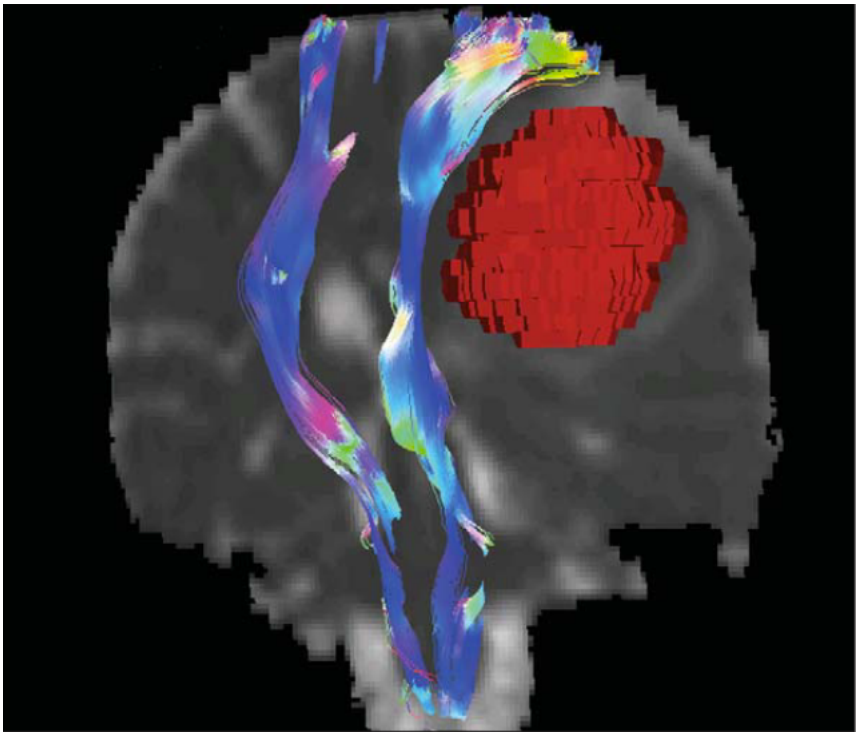
$$FA = \frac{\sqrt{(\lambda_1 - \lambda_2)^2 + (\lambda_2 - \lambda_3)^2 + (\lambda_3 - \lambda_1)^2}}{\sqrt{2}\sqrt{\lambda_1^2 + \lambda_2^2 + \lambda_3^2}} \tag{1.5}$$

Figura 5: Elipsoide de difusão




Fonte: Mori e Tournier (2014)

Figura 6: Distorção das vias motoras causada por um tumor



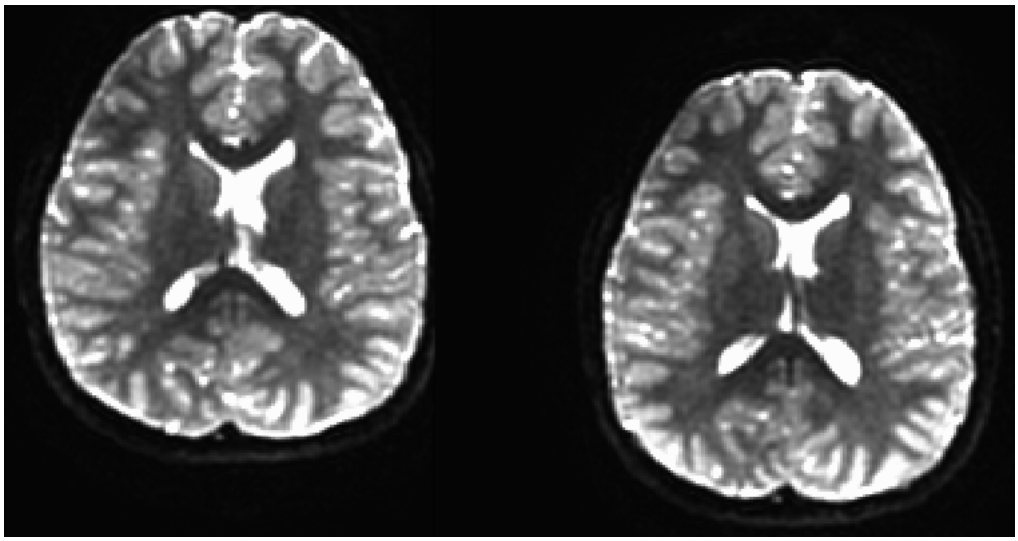
Fonte: Ciccarelli, Catani e Johansen-Berg (2008)

## 1.3 Artefatos em DTI

### 1.3.1 Erro de corregistro

Ocorre quando o sujeito move a cabeça distâncias maiores que as dimensões dos voxels. Isso faz com que *slices* contíguos apresentem um desalinhamento entre si. Na Figura 7 fica evidente o erro de corregistro, nesse caso, causado por translação. Esse tipo de artefato é comum a diversas modalidades de MRI. É possível corrigir o corregistro por meio de métodos de translação e rotação de corpo rígido.

Figura 7: Exemplo de erro de corregistro



Fonte: O Autor (2016)

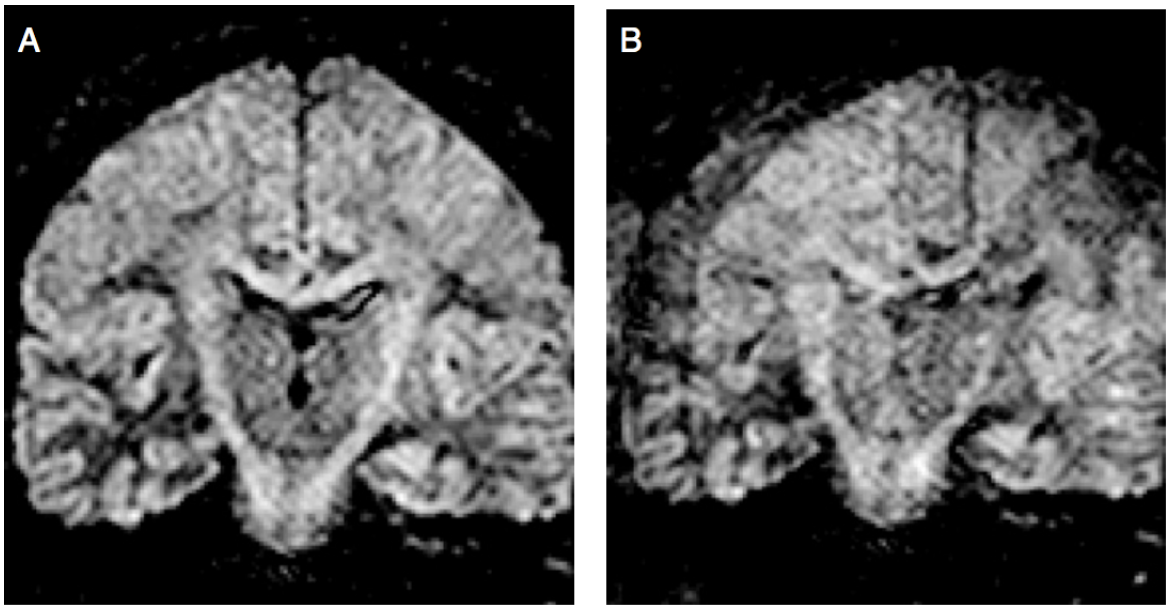
### 1.3.2 *Ghosting*

Também é causado por movimentação da cabeça, que, junto dos fortes campos magnéticos particulares do DTI, causa erro na fase da imagem no espaço $k$, gerando artefato na imagem do cérebro como o que está ilustrado na Figura 8. No caso do *ghosting*, movimentações menores que o tamanho de voxel já causa problemas. Um método que visa minimizar esse tipo de artefato é o imagem eco planar em único disparo (SS-EPI, do inglês *Single-Shot Echo-Planar Imaging*), que adquire imagens mais rapidamente ao custo de resolução. Mesmo com SS-EPI, o *ghosting* ainda ocorre em alguns casos e não é corrigível, fazendo com que as imagens afetadas sejam descartadas.

### 1.3.3 Causados por correntes parasitas

Além da ponderação por difusão, os gradientes utilizados causam também correntes elétricas parasitas nas outras partes do aparelho (ou até mesmo no sujeito sendo

Figura 8: Exemplo de *ghosting*



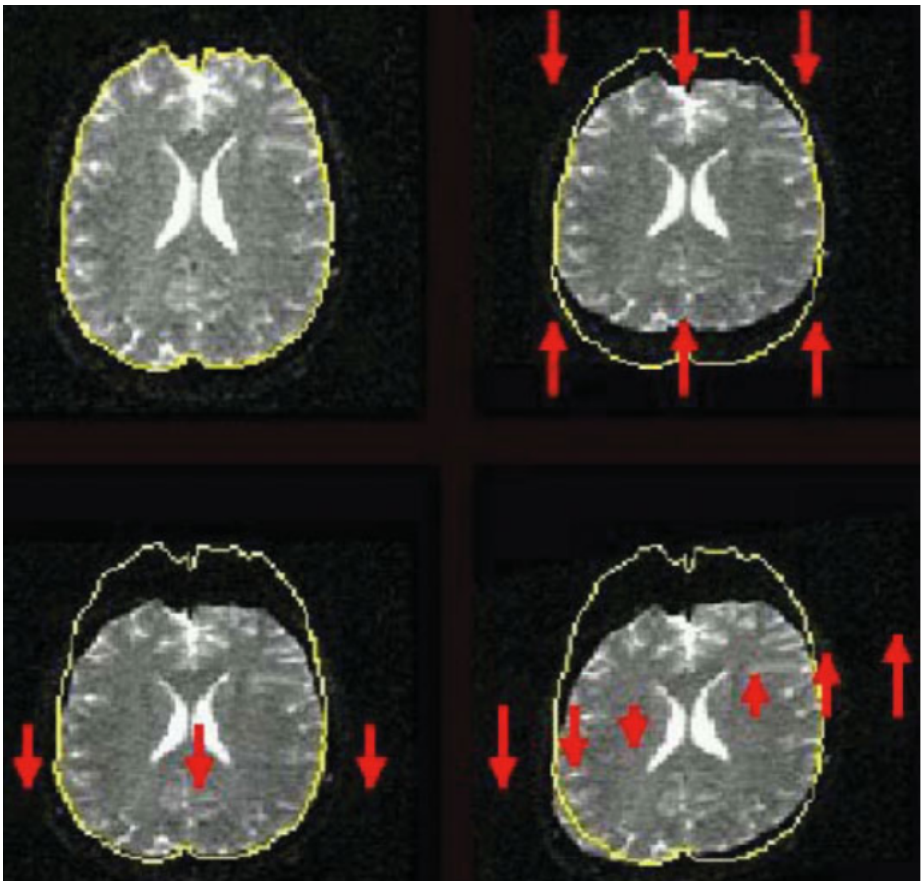

Fonte: Mori e Tournier (2014)

examinado), conforme a Lei de Faraday. Essas correntes parasitas geram também campos magnéticos gradientes, que interagem com o sistema. Os artefatos causados por correntes parasitas geralmente se manifestam como deformações geométricas nas neuroimagens (BIHAN et al., 2006). A Figura 9 mostra compressão, achatamento e cisalhamento como exemplos dessas deformações. Existem diversos recursos para minimizar as correntes parasitas, como pré-ênfase dos gradientes e blindagem de bobinas. Além disso, as deformações podem ser corrigidas por meio de transformação afim utilizando uma imagem adquirida sem gradiente como referência.

### 1.3.4 Causados por não homogeneidade do campo magnético

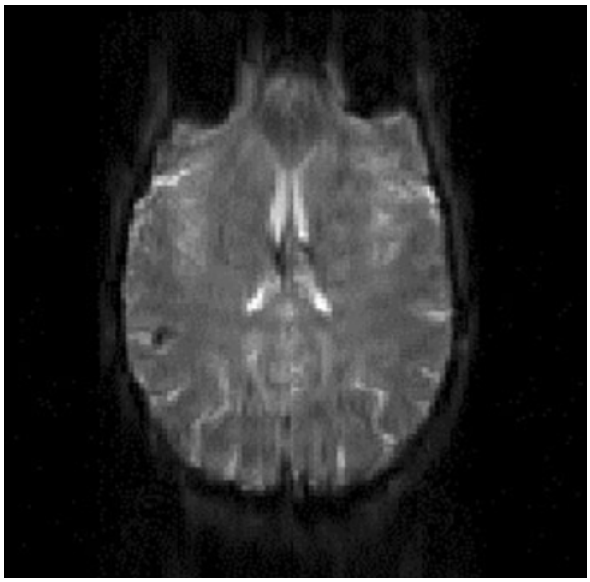
Por mais homogêneo que seja o campo magnético no interior da máquina, a cabeça humana não se magnetiza uniformemente, devido à sua construção. Dessa forma, a distribuição de frequências de ressonância não fica uniforme, fazendo com que apareçam deslocamentos e deformações na imagem final. A Figura 10 mostra uma imagem que foi distorcida pela não homogeneidade causada pelas diferenças de susceptibilidades magnéticas do tecido cerebral, osso e seios da face (cheios de ar). Pode-se mapear o campo magnético com uma sequência de pulsos específica e fazer uma correção baseada na distribuição do campo. Também é possível corrigir a imagem adquirindo-a duas vezes com direções concorrentes ou antiparalelas de gradientes.

Figura 9: Exemplos de distorções causadas por correntes parasitas



Fonte: Bihan et al. (2006)

Figura 10: Deformação causadas por não homogeneidade do campo

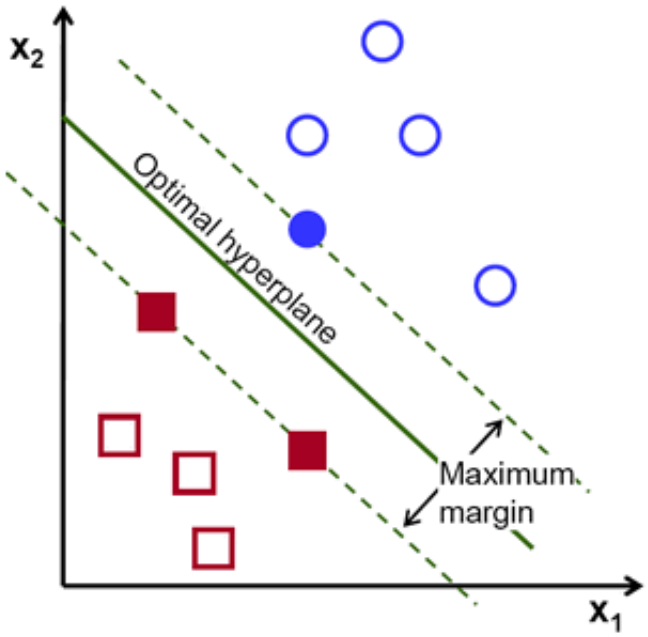


Fonte: Bihan et al. (2006)

## 1.4 Support-Vector Machines (SVM)

O paradigma de *Support-Vector Machines* (SVM) é uma ferramenta da área de aprendizado de máquinas utilizada para diferenciação entre grupos. Uma série de amostras classificadas entre os grupos de interesse chamada de conjunto de treinamento é utilizada para encontrar o hiperplano que melhor separe os grupos no hiperespaço das suas características (SHALEV-SHWARTZ; BEN-DAVID, 2014). A Figura 11 ilustra esse conceito de uma maneira mais intuitiva em um espaço bidimensional. Nesse caso, as amostras estão espalhadas em um plano e a SVM busca a reta que melhor separe os grupos (CORTES; VAPNIK, 1995). Após definido o hiperplano de separação, pode-se utilizá-lo para classificar novas amostras como pertencente a um dos grupos.

Figura 11: Separação de grupos via SVM

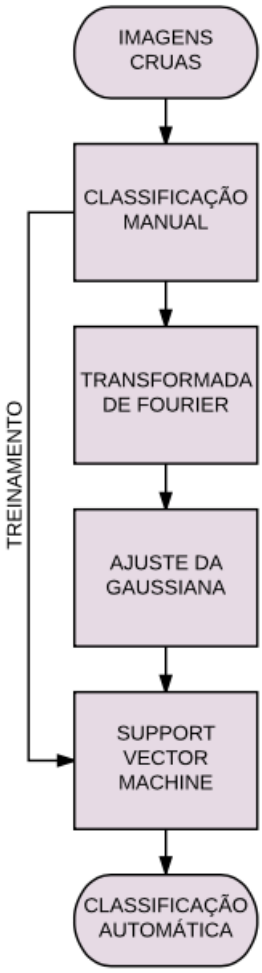



Fonte: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm

# 2 Metodologia

ESCREVER O QUÊ?

Figura 12: Fluxograma do método




Fonte: O Autor (2016)

## 2.1 Aquisição dos dados

As imagens foram adquiridas utilizando um *scanner* General Electric (Milwaukee, WI, EUA) Signa HDxT 3,0T no Instituto do Cérebro do Rio Grande do Sul (InsCer). O protocolo DTI utilizado consiste de uma imagem T2 sem sensibilização para difusão (imagem B0) e 33 imagens ponderadas por difusão ($b = 750s/mm^2$) em matrizes 256x256x50 com tamanho de voxel 1,0x1,0x2,4$mm^3$. A duração do exame é de 7min30s.

## 2.2 Sujeitos examinados

Foram utilizadas imagens de cinco sujeitos identificados na Tabela 1 escaneados para um estudo guarda-chuva mediante aprovação do Comitê de Ética da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) sob registro CAAE 30895614.5.0000.5336. A escolha dos sujeitos foi feita visando formar um conjunto com imagens normais e com corrupção em diferentes intensidades para que a classificação fosse o mais abrangente quanto possível. O número de sujeitos ficou limitado a 5 devido à restrição de tempo para execução do trabalho.

Tabela 1: Dados dos sujeitos examinados

| Sujeito | Sexo | Idade |
|---|---|---|
| S1 | Feminino | 8 anos |
| S2 | Masculino | 9 anos |
| S3 | Feminino | 9 anos |
| S4 | Masculino | 9 anos |
| S5 | Feminino | 8 anos |

Fonte: O Autor (2016)

## 2.3 Análise das imagens

Os exames dos cinco sujeitos totalizam

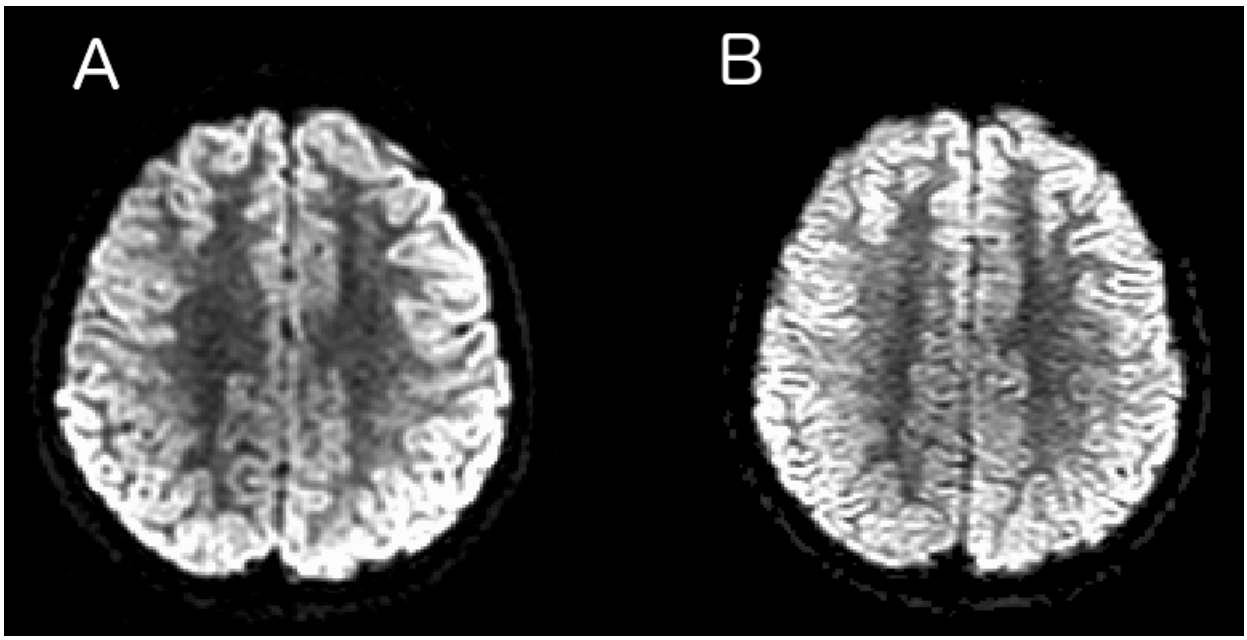
$$5\ sujeitos \cdot (1\ B0 + 33\ direções) \cdot 50\ (slices/direção) = 8500\ slices \tag{2.1}$$

que foram manualmente classificados como “normais”, “corrompidos” ou “outros”. A terceira categoria engloba *slices* que estão nos extremos inferiores e superiores dos volumes contendo pouca informação de interesse para DTI, como topo do crânio e tronco cerebral, por exemplo. Por esse motivo, não foi considerada na análise. Dos 8500 *slices*, 6652 contendo imagens normais e corrompidas foram, então, analisados. O grande número de imagens descartadas da análise se dá pelos sujeitos examinados serem crianças e, portanto, terem cérebros menores, ocupando menos o espaço escaneado.

Na Figura 13 dois slices adjacentes de um exame DTI foram selecionados. Em (A) uma imagem normal e em (B) uma imagem corrompida. Pode-se notar que em (B) há uma espécie de ondulação sobreposta à figura de cérebro. Isso sugere que a corrupção da imagem está ligada a componentes de frequência relativamente alta. Portanto, partiu-se para uma análise espectral das imagens.

Nesta etapa, utilizou-se a ferramente 3dFFT do conjunto de *softwares* AFNI (*Analysis of Functional NeuroImages*, disponível em `https://afni.nimh.nih.gov`) para

Figura 13: *Slices* adjacentes em um exame DTI




Fonte: O Autor (2016)

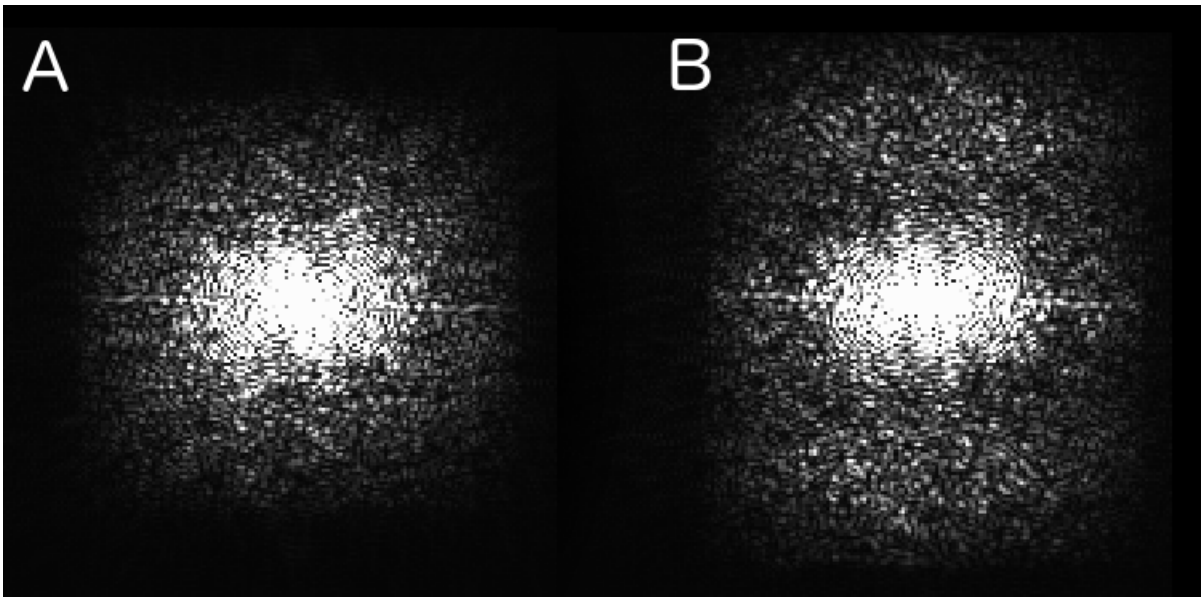
computar a Transformada de Fourier das imagens DTI *slice* a *slice*. A Figura 14 mostra os espectros dos *slices* utilizados como exemplo na Figura 13. Nota-se o espalhamento no espectro B, da imagem corrompida. Houve deslocamento de energia da região central da imagem (baixas frequências) para as regiões mais externas (altas frequências) ao longo do eixo vertical. Mais imagens estão disponíveis no Anexo A.

## 2.4 Processamento das imagens

A linguagem de programação Python (série 2.7.x, disponível em `www.python.org`) foi utilizada para a implementação do pipeline de processamento das imagens. Para trabalhar com as neuroimagens, é necessária uma biblioteca para leitura de imagens no formato NIfTI-1 (DFWG, 2004). Foi utilizada a biblioteca NiBabel (disponível em `www.nipy.org/nibabel`), que possibilita a leitura e escrita de diversos formatos de neuroimagens utilizando Python. As imagens são, então, transformadas em *arrays* por meio de uma biblioteca para computação científica chamada NumPy (disponível em `www.numpy.org`).

As imagens transformadas em *arrays* são processadas *slice* a *slice* de acordo com o plano em que foram adquiridas (no caso deste trabalho, plano axial). Primeiramente, é feita a Transformada Rápida de Fourier (FFT, do inglês *Fast Fourier Transform*) de cada slice utilizando o pacote de FFTs do conjunto de bibliotecas SciPy (disponível em `www.scipy.org`) e normaliza-se cada espectro de maneira que a soma de todos os *bins*

Figura 14: Espectros dos *slices* da Figura 13




Fonte: O Autor (2016)

seja 100. Na sequência, cada espectro é ajustado utilizando a o pacote de ajuste de curvas do SciPy a uma gaussiana bidimensional elíptica no formato

$$f(x,y) = A \cdot exp(-(a \cdot (x-x_0)^2 - 2b \cdot (x-x_0)(y-y_0) + c \cdot (y-y_0)^2)), \tag{2.2}$$

em que

$$a = \frac{\cos^2\theta}{2\sigma_x^2} + \frac{\sin^2\theta}{2\sigma_y^2}, \quad b = -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2}, \quad c = \frac{\sin^2\theta}{2\sigma_x^2} + \frac{\cos^2\theta}{2\sigma_y^2}, \tag{2.3}$$

e é calculado o valor quadrático médio (RMS, do inglês *Root Mean Square*) do resíduo entre o espectro e a gaussiana gerada com os parâmetros calculados no ajuste. O código para o processamento das imagens está no Anexo B.

## 2.5 Classificação das imagens

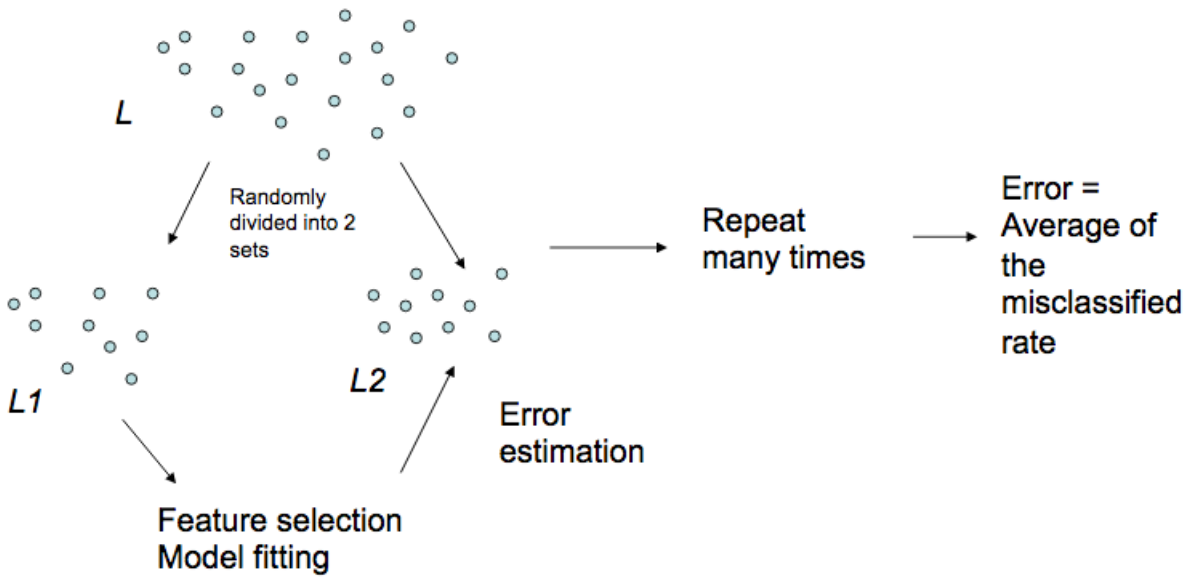
O processamento das imagens resulta em cinco parâmetros para cada *slice*: A, a, b, c e o resíduo do ajuste. Eles são as características utilizadas para classificar as imagens entre normais e corrompidas. Isso é feito por meio de uma *Support-Vector Machine* (SVM) implementada utilizando a biblioteca scikit-learn (disponível em `scikit-learn.org`) para *machine learning* em Python. O código para a classificação está no Anexo C.

# 3 Resultados e Discussões

## 3.1 Taxa de acertos

Para avaliar a classificação das imagens, foi feita uma validação cruzada de Monte Carlo. Mil vezes os 6652 *slices* foram separados 90% para treino e 10% para teste da SVM aleatoriamente. Em cada iteração, as classificações dadas pela SVM foram comparadas às manuais. O processo está ilustrado na Figura 15. A taxa de acerto média foi de 96,60% com desvio padrão de 0,67%.

Figura 15: Validação cruzada de Monte Carlo




Fonte: Man (2009)

É importante ressaltar essa não é a taxa de acertos em relação a corrupção ou não de um exame completo, e sim *slice* a *slice*. Para dizer se o exame precisa ser refeito, é necessário estabelecer um limite para o erro introduzido devido à corrupção das imagens. Cada direção tem uma contribuição de 1/N (N sendo o número de direções de gradientes) no tensor de difusão. Se é estabelecido um erro máximo de 5%, no caso de 33 direções de gradiente, dois slices de mesma posição corrompidos já são suficientes para que o exame seja considerado corrompido (se estiverem em posições diferentes, não estão contribuindo para o cálculo dos mesmos tensores e, portanto, não acumulando os erros).

As imagens utilizadas neste estudo foram adquiridas com $b = 750s/mm^2$, que é um valor baixo de ponderação por difusão, mas comum em protocolos DTI. Para b até

cerca de $1500 s/mm^2$, não há uma variação muito grande nas características das imagens ponderadas por difusão. Acima disso, é necessário reavaliar a eficácia do método, pois os padrões de artefatos passam a ser um pouco diferentes.

## 3.2 Tempo de processamento

Os dados foram processados em um único núcleo de processador Intel Core i7 2,8GHz 1ª geração tendo tempo de processamento de cada sujeito conforme a Tabela 2. Para que seja feito em "tempo real", o tempo de processamento deve ser semelhante ou inferior à duração do exame. Pode-se ver pela Tabela 2 que o processamento levou, em média, 2,1min mais que o tempo de exame, que é de 7,5min.

Tabela 2: Tempo de processamento dos exames

| Sujeito | Tempo de processamento |
|---|---|
| S1 | 8,2 min |
| S2 | 10,1 min |
| S3 | 9,4 min |
| S4 | 9,0 min |
| S5 | 11,3 min |
| Média: 9,6min | Desvio padrão: 1,0min |

Fonte: O Autor (2016)

Na prática, se o DTI está sendo feito junto de outros exames na mesma sessão, essa diferença média de 2,1min pode ser contornada colocando o DTI antes um outro exame, durante o qual possa ser terminado o processamento de forma imperceptível para o sujeito escaneado. Por outro lado, se o DTI é feito isoladamente, o intervalo entre final de exame e resultado do processamento pode ficar desconfortável para a pessoa dentro da máquina.

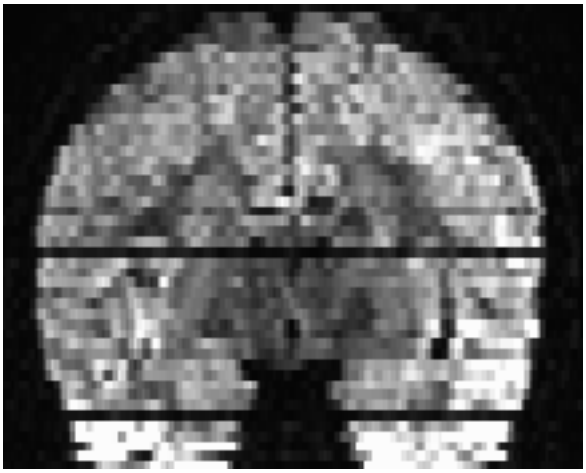
Como o método proposto analisa os exames *slice* a *slice*, é possível paralelizá-lo com dois núcleos de processamento computando imagens intercaladas. Dessa forma, o tempo médio para se ter o resultado cai para aproximadamente a metade, cerca de 4,8min. Esse tempo já é mais que suficiente para que o método funcione em "tempo real".

## 3.3 Trabalhos Semelhantes

Existe um trabalho sobre detecção de corrupção em DTI em tempo real feito por Li et al. (2013). Eles propõem uma métrica chamada de cISID (do inglês, *corrected Inter-Slice Intensity Discontinuity*), que analisa os exames volume a volume em um perspectiva

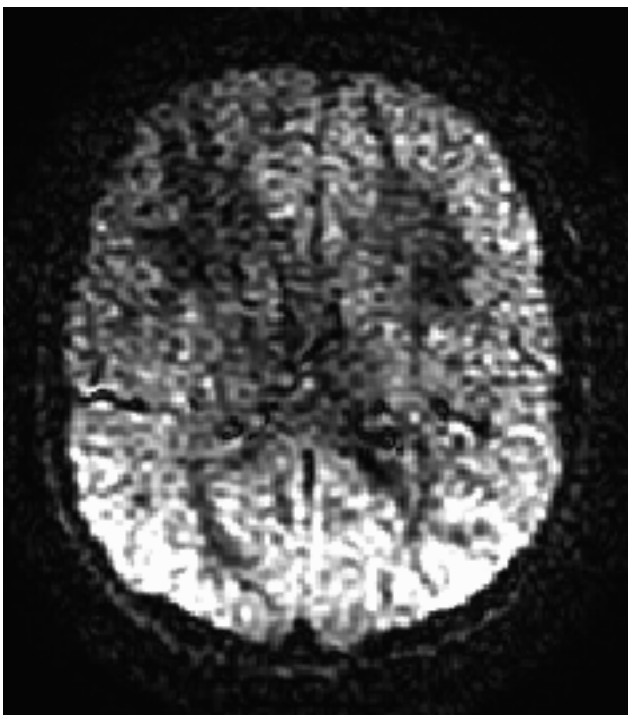
ortogonal à aquisição (se os *slices* foram adquiridos axialmente, utilizam a vista sagital ou a coronal). Dessa forma, buscam identificar descontinuidades causadas por *slices* corrompidos. Na Figura 16 pode-se ver um corte coronal cuja descontinuidade central é causada pela imagem da Figura 17, que nitidamente está corrompida.

Figura 16: Corte coronal de um volume com descontinuidades



Fonte: O Autor (2016)

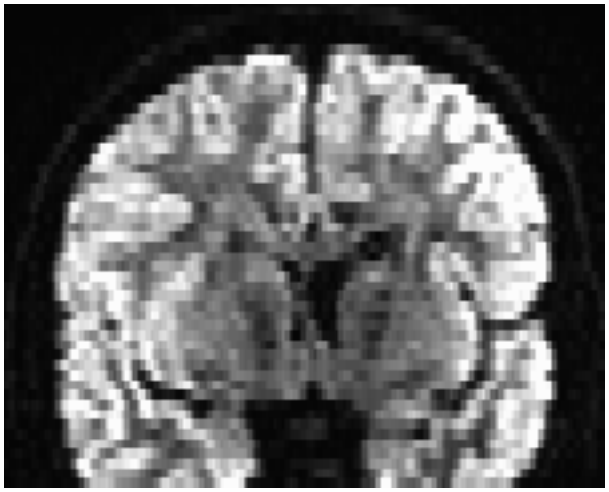
Figura 17: *Slice* causador da descontinuidade central na Figura 16



Fonte: O Autor (2016)

Entretanto, é possível que um *slice* esteja corrompido e não cause descontinuidade perceptível para identificá-lo como corrompido. Um exemplo está na Figura 18, que mostra um corte coronal sem descontinuidades evidentes. No entanto, a imagem da Figura 19 pertence a esse volume e está corrompida. Mesmo sendo menos evidente a presença de artefatos nessa imagem, o método via ajuste de gaussiana acertou na sua classificação como corrompida. Pode-se notar o espalhamento no espectro dela, conforme mostra a Figura 20.

Figura 18: Corte coronal de um volume sem descontinuidades



Fonte: O Autor (2016)

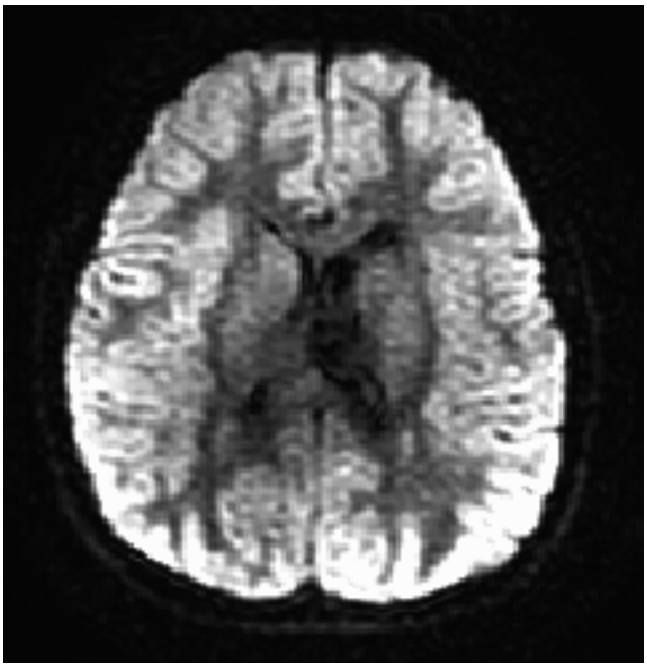
Por também levar em conta descontinuidades entres *slices* adjacentes, o cISID fica suscetível a falhas quando ocorrem artefatos que afetam um grupo de imagens de maneira semelhante. No processamento *slice* a *slice*, não há problema quando isso ocorre, pois as avaliações são feitas independentemente. Por outro lado, o cISID é computacionalmente mais simples, o que implica em um menor tempo de processamento. Mas como o cISID só pode ser computado depois que um volume inteiro é adquirido, a maior velocidade de processamento acaba não sendo relevante.

## 3.4 Trabalhos Futuros

Devem ser feitas mais avaliações do método. Um número maior de sujeitos e cérebros adultos devem ser testados, bem como imagens adquiridas em máquinas diferentes e de diferentes fabricantes. Além disso, seria interessante verificar se o método se mantém eficaz para valores altos de b, como os utilizados em protocolos HARDI.

Para que o processamento e a classificação sejam utilizados no ambiente de exame, é necessário estabelecer comunicação entre o computador no qual é feito o processamento e o aparelho de ressonância magnética. Além disso, deve-se proporcionar uma interface

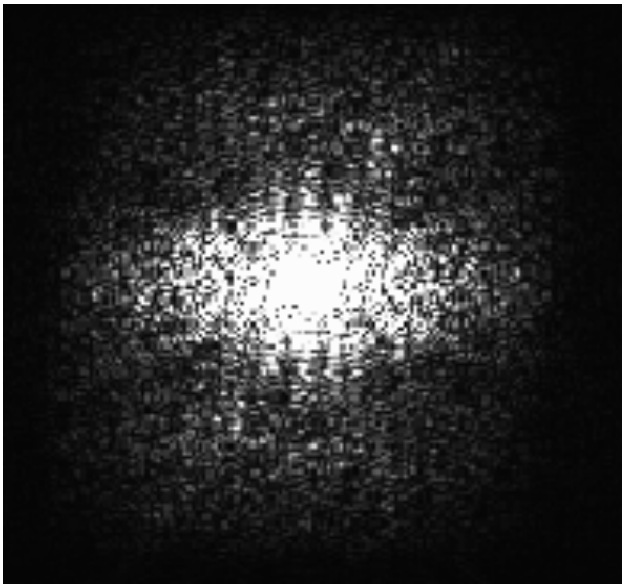
Figura 19: *Slice* corrompido pertencente ao volume da Figura 18



Fonte: O Autor (2016)

gráfica para o operador da máquina, na qual possam ser ajustados os parâmetros para indicação de necessidade de reaquisição do exame, como foi feito por Much (2016).

Figura 20: Espectro do *slice* da Figura 19



Fonte: O Autor (2016)

# Conclusão

O método proposto teve uma taxa de acerto média de 96,60% com desvio padrão de 0,67% numa validação cruzada de 1000 iterações com 6652 *slices* adquiridos com $b = 750s/mm^2$. Para utilização em tempo real, recomenda-se o uso de dois núcleos de processamento computando imagens intercaladas. Para valores de b médios e altos, é necessário reavaliá-lo com as devidas imagens.

# Referências


BIHAN, D. L. et al. Artifacts and pitfalls in diffusion MRI. *Journal of Magnetic Resonance Imaging*, v. 24, n. 3, p. 478–488, 2006. ISSN 10531807. Citado 2 vezes nas páginas 28 e 29.

BLINK, E. *Basic MRI Physics*. 1st. ed. [S.l.: s.n.], 2010. 76 p. Citado na página 23.

BROWN, R. W. et al. *Magnetic Resonance Imaging: Physical Principles and Sequence Design*. 2nd. ed. [S.l.]: Wiley-Blackwell, 2014. 1008 p. Citado na página 22.

CICCARELLI, O.; CATANI, M.; JOHANSEN-BERG, H. Diffusion-based tractography in neurological disorders: concepts, applications, and future developments. *Lancet Neurology*, 2008. Citado na página 26.

CORTES, C.; VAPNIK, V. Support-Vector Networks. *Machine Learning*, 1995. ISSN 15730565. Citado na página 30.

DAMADIAN, R. V. Apparatus and method for detecting cancer in tissue. *US Patent 3,789,832*, 1972. Citado na página 21.

DFWG, D. F. W. G. The Nifti-1 Data Format. p. 1–4, 2004. Citado na página 33.

LI, Y. et al. Image corruption detection in diffusion tensor imaging for post-processing and real-time monitoring. *PloS one*, v. 8, n. 10, jan 2013. Citado na página 36.

MCROBBIE, D. W. et al. *MRI From Pictures To Protons*. 2nd. ed. [S.l.]: Cambridge University Press, 2006. ISBN 9780511349447. Citado na página 21.

MORI, S.; TOURNIER, J.-D. *Introduction to Diffustion Tensor Imaging: And Higher Models*. 2nd. ed. [S.l.]: Elsevier, 2014. Citado 4 vezes nas páginas 24, 25, 26 e 28.

MUCH, M. D. *Sistema de detecção de movimento em tempo real para exames de ressonância magnética funcional*. Dissertação (Mestrado) — PUCRS, 2016. Citado na página 39.

PAULING, L.; CORYELL, C. D. The Magnetic Properties and Structure of Hemoglobin, Oxyhemoglobin and Carbonmonoxyhemoglobin. *Proceedings of the National Academy of Sciences of the United States of America*, v. 22, 1936. Citado na página 21.

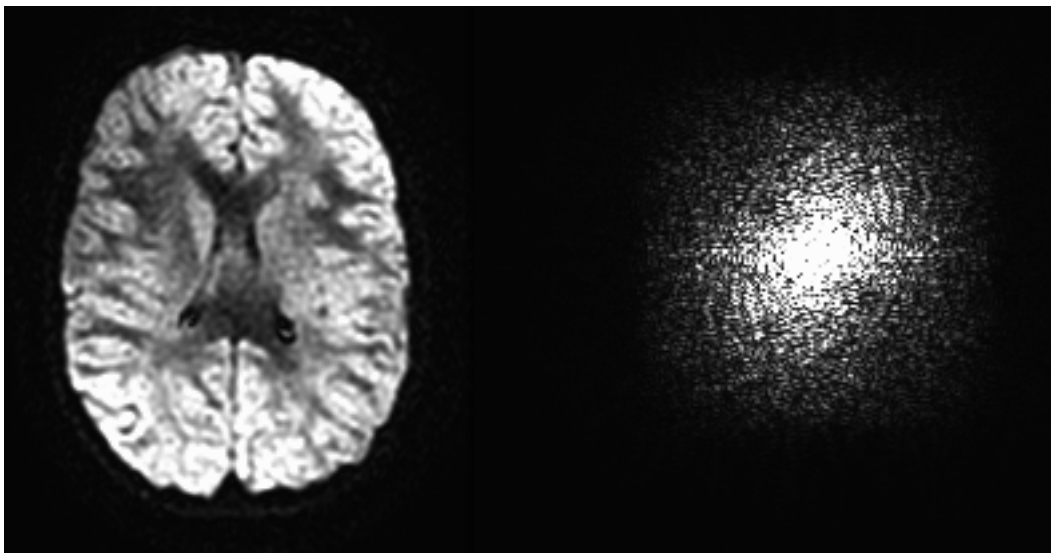
SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. [S.l.]: Cambridge University Press, 2014. Citado na página 30.

SILVA, A. L. B. Bathista e. *Elementos Históricos de Ressonância Magnética Nuclear*. São Carlos - SP: Instituto de Física de São Carlos - Universidade de São Paulo, 2004. 50 p. Citado na página 21.

# Anexos

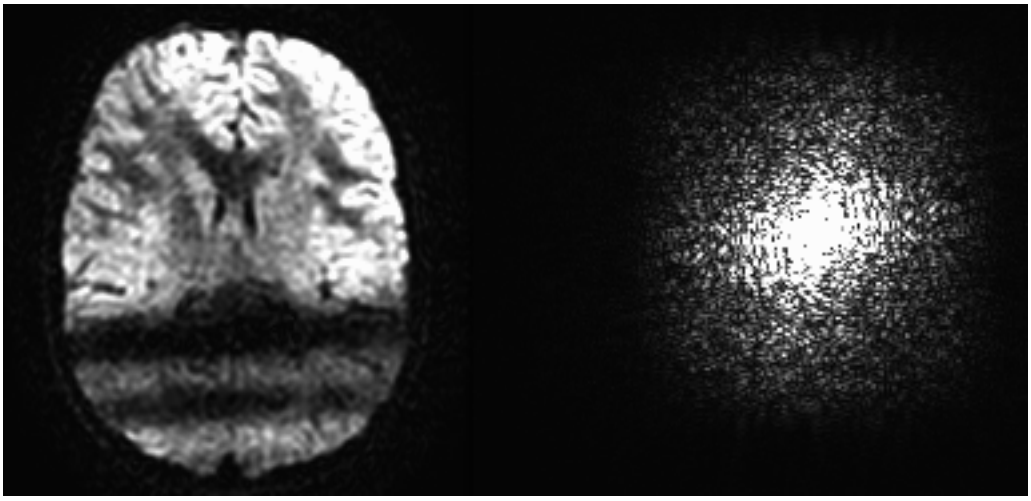
# ANEXO A – Imagens e espectros

Figura 21: Imagem normal e seu espectro



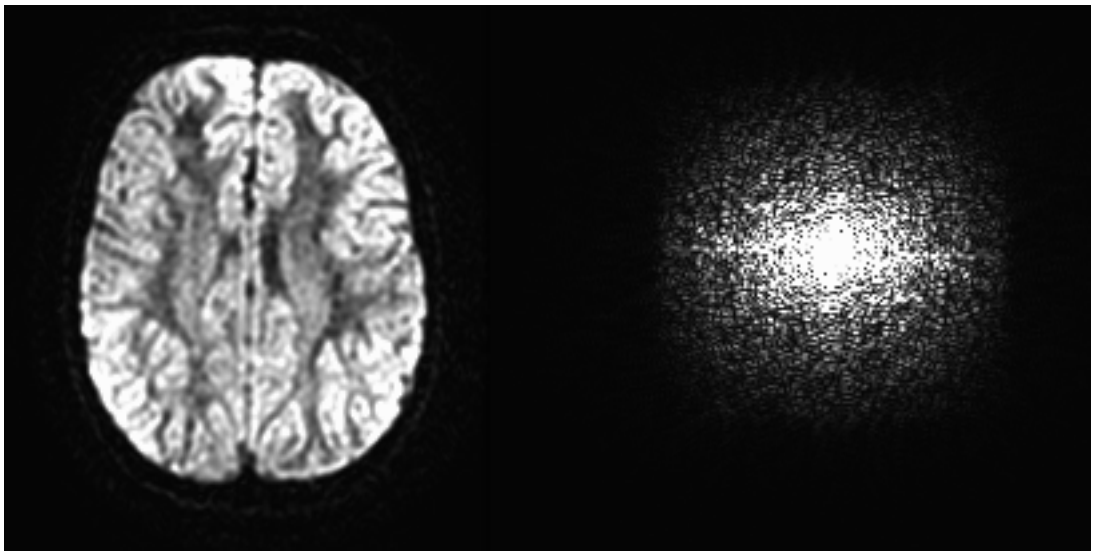
Fonte: O Autor (2016)

Figura 22: Imagem corrompida e seu espectro



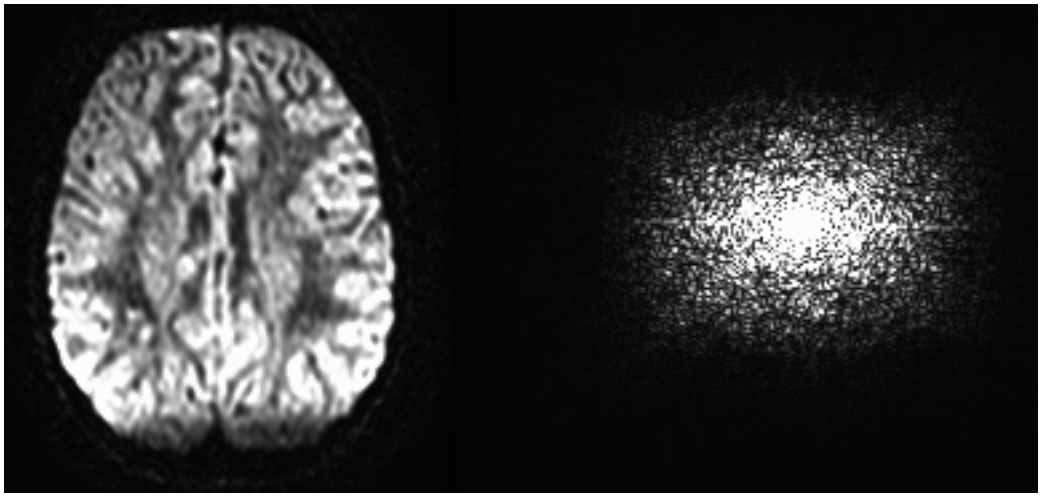
Fonte: O Autor (2016)

Figura 23: Imagem normal e seu espectro



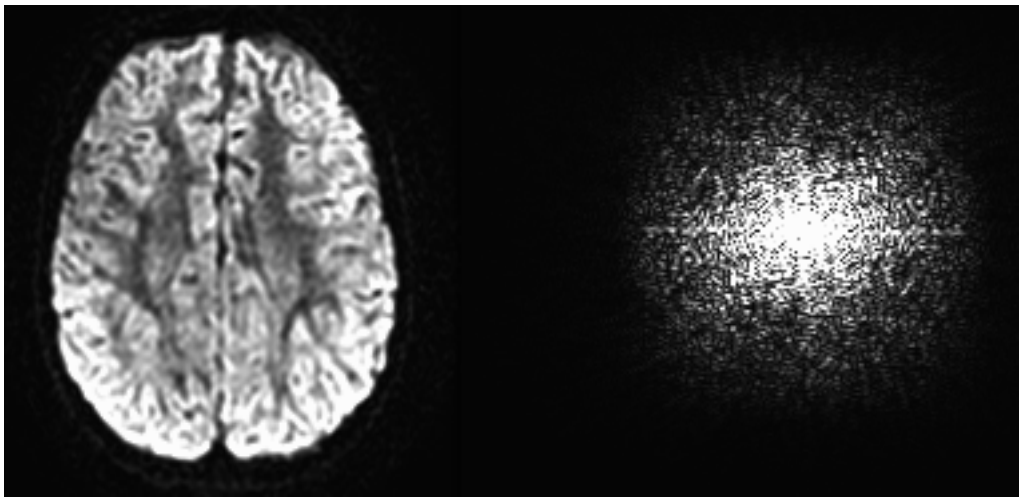
Fonte: O Autor (2016)

Figura 24: Imagem corrompida e seu espectro



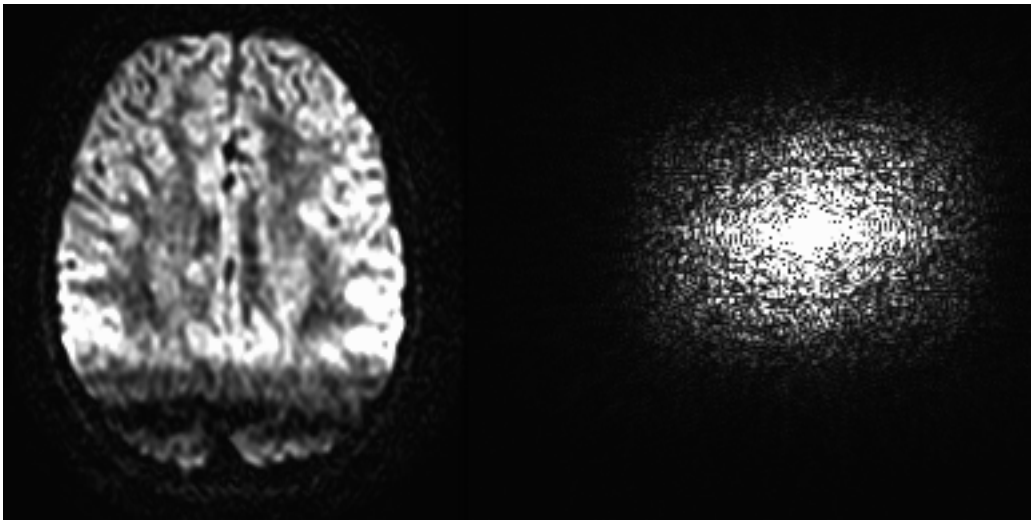
Fonte: O Autor (2016)

Figura 25: Imagem normal e seu espectro



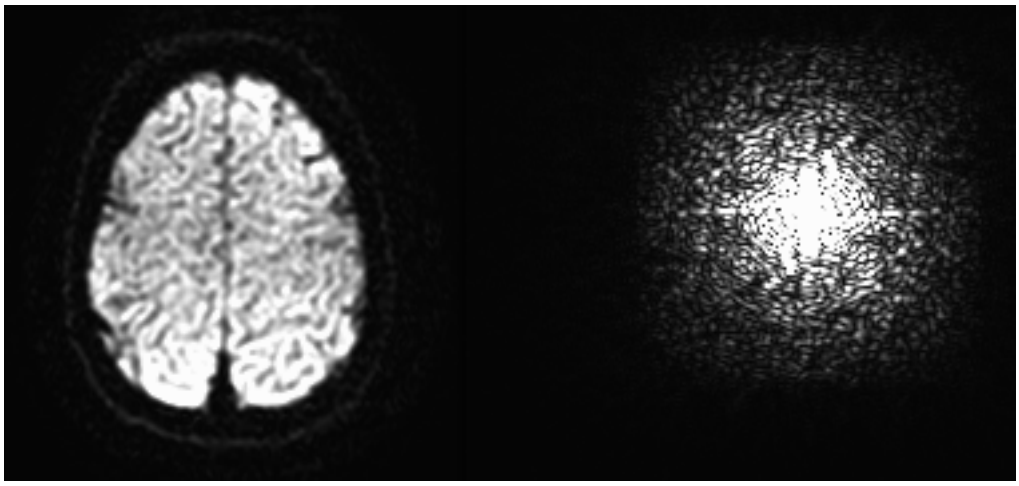
Fonte: O Autor (2016)

Figura 26: Imagem corrompida e seu espectro



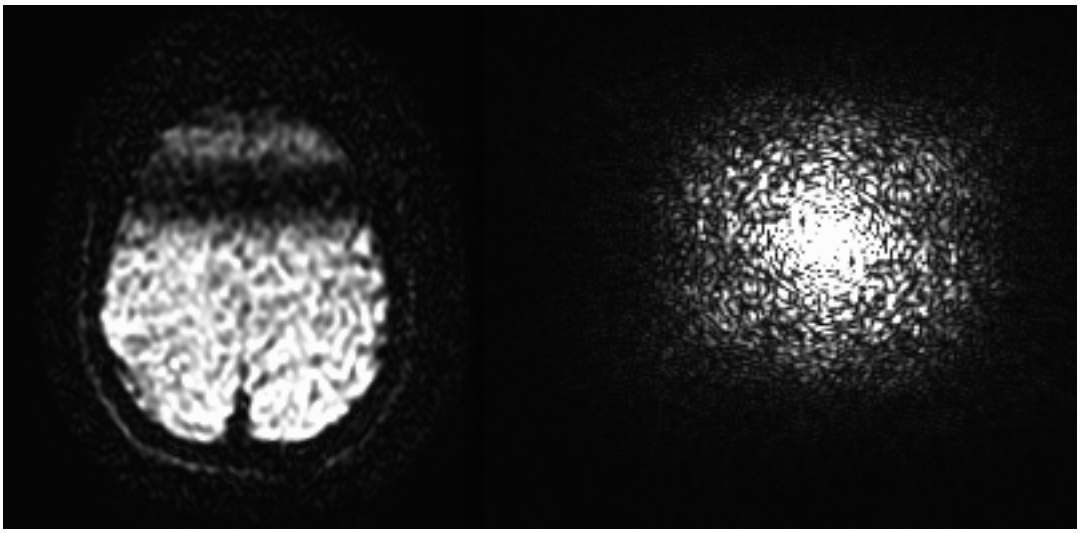
Fonte: O Autor (2016)

Figura 27: Imagem normal e seu espectro



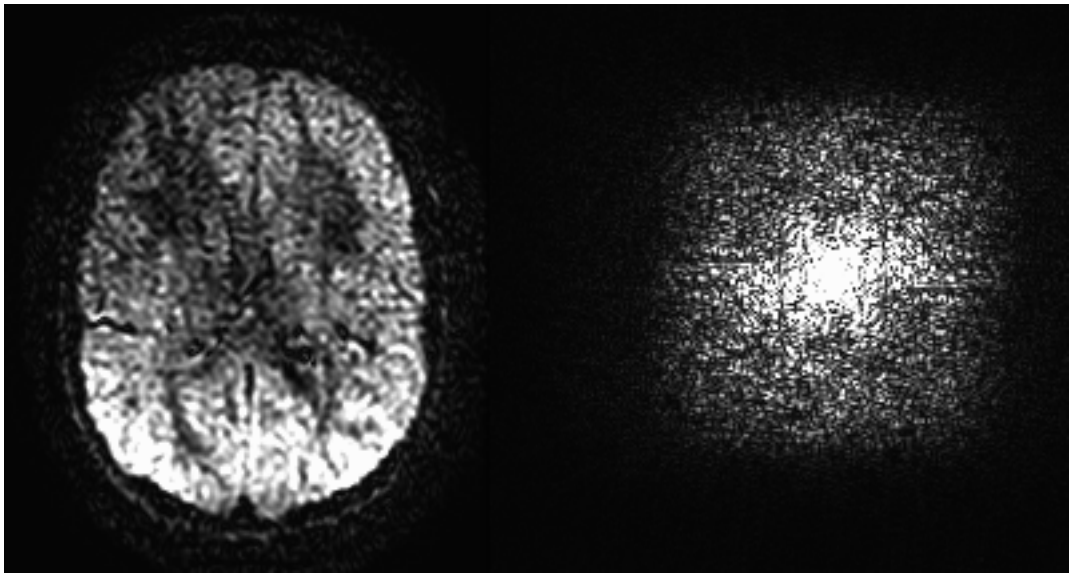
Fonte: O Autor (2016)

Figura 28: Imagem corrompida e seu espectro



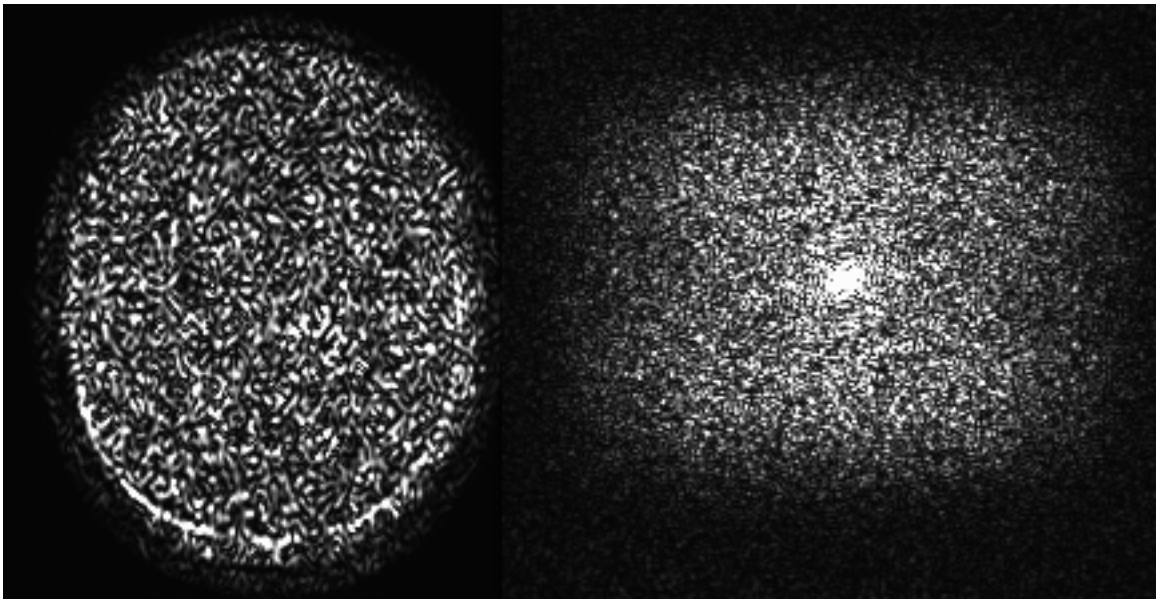
Fonte: O Autor (2016)

Figura 29: Imagem corrompida e seu espectro



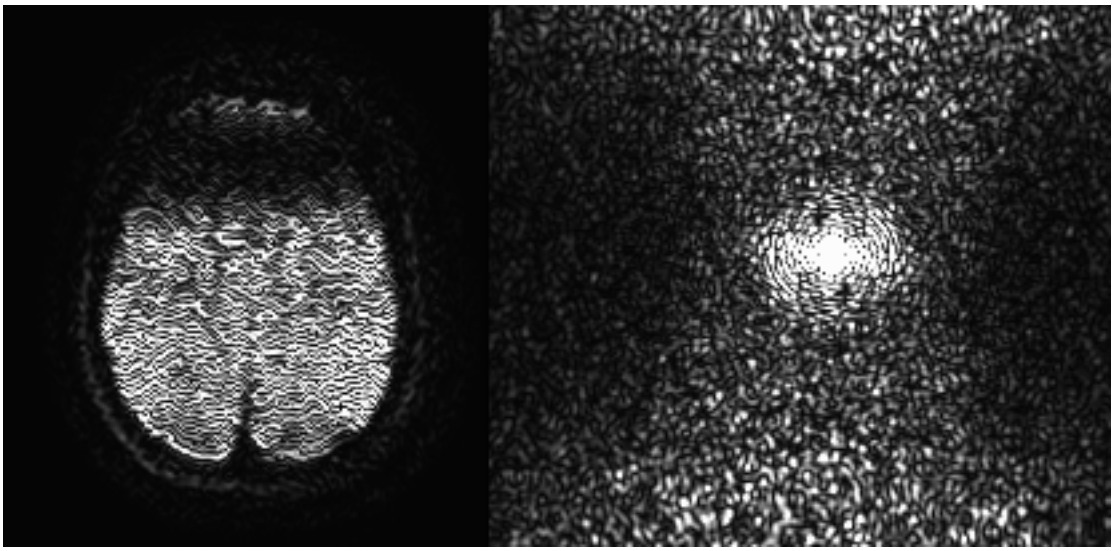
Fonte: O Autor (2016)

Figura 30: Imagem corrompida e seu espectro



Fonte: O Autor (2016)

Figura 31: Imagem corrompida e seu espectro



Fonte: O Autor (2016)

# ANEXO B – Código para o processamento

```
"""
AJUSTE DE GAUSSIANA BIDIMENSIONAL ELIPTICA A ESPECTRO DE NEUROIMAGEM
"""

# Importando bibliotecas necessarias
import os
import numpy as np
from scipy import optimize as opt
import nibabel as nib
from scipy import fftpack
import pickle
import time


# Definindo a gaussiana bidimensional eliptica
def gauss2d(xy, amp, x0, y0, a, b, c):
    x, y = xy
    expoente = a * (x - x0)**2
    expoente -= 2 * b * (x - x0)**2 * (y - y0)**2
    expoente += c * (y - y0)**2
    return amp * np.exp(-expoente)

# Iniciando timer para tempo total
start_total = time.time()

# Definindo diretorio com os exames a serem analisados
data_dir = os.path.expanduser('~') + '/' + 'RT-DTI' + '/' + 'EXAMES' + '/'
# Definindo diretorio para guardar os dados do processamento
out_dir = os.path.expanduser('~') + '/' + 'RT-DTI' + '/' + 'results' + '/'

# Listando sujeitos a serem analisados
subjects = ['S1','S2','S3','S4','S5']

# Criando lista para guardar erros
erros = []

# Iniciando laco para repetir processamento para cada sujeito
for subj in subjects:

    # Timer de inicio do sujeito
```

```
41      start_subj = time.time()
42      # Mensagem de inicio do sujeito
43      print('\n\nIniciando processamento do sujeito %s\n' % subj)
44
45      # Criando lista para guardar dados do sujeito
46      dados = []
47
48      # Definindo caminho para a imagem do sujeito
49      exam_file = data_dir + subj + '/' + subj + '.nii.gz'
50      # Carregando o exame no ambiente Python
51      exam = nib.load(exam_file)
52      # Carregando os dados do exame
53      exam_data = exam.get_data()
54      # Extraindo informacoes das dimensoes da imagem
55      sizeX, sizeY, numSlices, numDir = exam_data.shape
56
57      # Criando array para descrever as coordenadas como sequencia de pontos
58      x = []
59      y = []
60      for i in xrange(sizeX):
61          for j in xrange(sizeY):
62              x = np.append(x,i)
63              y = np.append(y,j)
64      xy = np.array([x,y])
65
66      # Iniciando laco para repetir processamento para cada direcao
67      for direc in xrange(numDir):
68          # Trabalhando slice a slice na direcao
69          for sl in xrange(numSlices):
70              # Separando slice de interesse do volume
71              slice = exam_data[:,:,sl,direc]
72              # Fazendo a fft do slice
73              slice_fft = fftpack.fft2(slice)
74              # Colocando as frequencias mais baixas no centro
75              slice_fft = fftpack.fftshift(slice_fft)
76              # Calculando o espectro de potencia
77              ps = np.abs(slice_fft)**2
78              # Normalizando o espectro para 100
79              ps = ps / (ps.sum()/100)
80
81              # Transformando espectro em sequencia de pontos
82              ps_1D = np.ravel(ps)
83
84              # Definindo uma estimativa inicial para o ajuste
85              # Parametros [amp,x0,y0,a,b,c]
86              guess = [1,128,128,1,1,1]
87
```

```
88              # Testando se o ajuste converge
89              try:
90                  # Fazendo o ajuste do espectro a gaussiana
91                  fit_params, uncert_cov = opt.curve_fit(gauss2d,xy,ps_1D,p0=
    guess)
92
93                  # Calculando a gaussiana dada pelo ajuste
94                  ps_fit = gauss2d(xy,*fit_params)
95                  # Calculando o RMS do residuo do ajuste
96                  resid = np.sqrt(np.mean((ps_1D - ps_fit)**2))
97
98                  # Guardando os dados do slice no vetor
99                  dados_slice = [subj,direc,sl,fit_params,uncert_cov,resid]
100                 dados.append(dados_slice)
101
102             # Caso haja erro
103             except RuntimeError:
104                 # Guardando slice na lista de erros
105                 erros.append([subj,direc,sl])
106                 # Informando na tela que houve erro
107                 print('Parametros otimos nao encontrados!!')
108                 print('Seguindo adiante!')
109                 # Guardando os dados do slice
110                 dados_temp = [subj,direc,sl,'erro','erro','erro']
111                 dados.append(dados_temp)
112
113             # Informando na tela final do processamento do slice
114             print("Finalizado slice %d de %d" % (sl+1, numSlices))
115         # Informando na tela final do processamento da direcao
116         print("Finalizada direcao %d de %d\n" % (direc+1, numDir))
117
118     # Guardando dados do sujeito como stream de bytes
119     pickle.dump(dados,open(out_dir + "dados_%s.p" % subj,"wb"))
120
121     # Terminando cronometria do processamento do sujeito
122     end_subj = time.time()
123     # Informando na tela o tempo total do sujeito
124     print("Tempo do sujeito: %.1f min\n" % ((end_subj - start_subj)/60))
125
126 # Terminando a cronometria da sessao de processamento
127 end_total = time.time()
128 # Informando o tempo total da sessao na tela
129 print('Tempo total: %.1f min\n'% ((end_total - start_total)/60))
130 # Informando os slices em que o processamento falhou, caso haja
131 if erros:
132     print('Erros em:')
133     print(erros)
```

```
134     print('\n')
```

rt–dti_proc.py

# ANEXO C – Código para a classificação

```
"""
CLASSIFICACAO DE NEUROIMAGEM VIA SUPPORT-VECTOR MACHINE
"""

# Importando bibliotecas necessarias
import pickle
from sklearn import svm
import matplotlib.pyplot as plt
import numpy as np
import os

# Definindo sujeitos a serem classificados
subjects = ['S1','S2','S3','S4','S5']

# Criando arranjos para guardar os dados
# Features: amp, a, b, c, residuo
features = np.zeros((1700*len(subjects),5))
grupos = np.zeros(1700*len(subjects))
slices = np.zeros((1700*len(subjects),3))

# Criando contador para preencher os dados
i = 0

# Definindo o diretorio com os dados
data_dir = os.path.expanduser('~') + '/' + 'RT-DTI' + '/' + 'results' + '/'

# Carregando dados do processamento de cada sujeito
for subj in subjects:
    # Deserializando dados
    s_data = pickle.load(open('%sdata_%s.p' % (data_dir,subj),'rb'))
    for direc in xrange(34):
        for slice in xrange(50):
            # amplitude
            features[i][0]=s_data[direc][slice][3][0]
            # a
            features[i][1]=s_data[direc][slice][3][3]
            # b
            features[i][2]=s_data[direc][slice][3][4]
            # c
            features[i][3]=s_data[direc][slice][3][5]
```

```
41              # residuo
42              features[i][4]=s_data[direc][slice][5]
43              # classificacao manual
44              # 1 para normal e 0 para corrompido
45              grupos[i]=s_data[direc][slice][6]
46              # Indicacao da imagem
47              # Nome, direcao e slice
48              slices[i][0] = s_data[direc][slice][0][1]
49              slices[i][1] = s_data[direc][slice][1]
50              slices[i][2] = s_data[direc][slice][2]
51              # incrementando o contador
52              i = i + 1
53
54 # Removendo slices nao utilizados (flag 2)
55 features = features[grupos != 2,:]
56 slices = slices[grupos != 2,:]
57 grupos = grupos[grupos != 2]
58 # Definindo o numero de imagens
59 n = len(grupos)
60 # Criando array para estatistica
61 stat = np.array([])
62
63 # Realizando experimentos aleatorios
64 for run in xrange(1000):
65     # Escrevendo na tela o numero da iteracao
66     print("\n\n Iteracao %i" % run)
67
68     # Gerando ordem aleatoria para as imagens
69     ordem = np.random.permutation(n)
70
71     # Colocando imagens em ordem aleatoria
72     f = features[ordem]
73     # Colocando as classificacoes manuais na mesma ordem
74     g = grupos[ordem]
75
76     # Dividindo imagens em conjuntos de treino (90%) e de teste (10%)
77     f_treino = f[:0.9 * n]
78     g_treino = g[:0.9 * n]
79     f_teste = f[0.9 * n:]
80     g_teste = g[0.9 * n:]
81
82     # Definindo o kernel da svm
83     kernel = 'linear'
84
85     # Criando o classificador
86     classificador = svm.SVC(kernel=kernel)
87     # Treinando o classificador
```

```
88      classificador.fit(f_treino,g_treino)
89      # Comparando a saida do classificador com a classificacao manual do
        grupo de teste
90      taxa_acertos = classificador.score(f_teste,g_teste)
91      # Guardando taxa de acertos da iteracao
92      stat = np.append(stat,taxa_acertos)
93      # Mostrando na tela
94      print(taxa_acertos)
95
96 # Calculando e mostrando na tela media e desvio padrao das taxas de acertos
97 print("\n Media: %f" % np.average(stat))
98 print("\n Desvio: %f" % np.std(stat))
```

rt–dti_svm.py