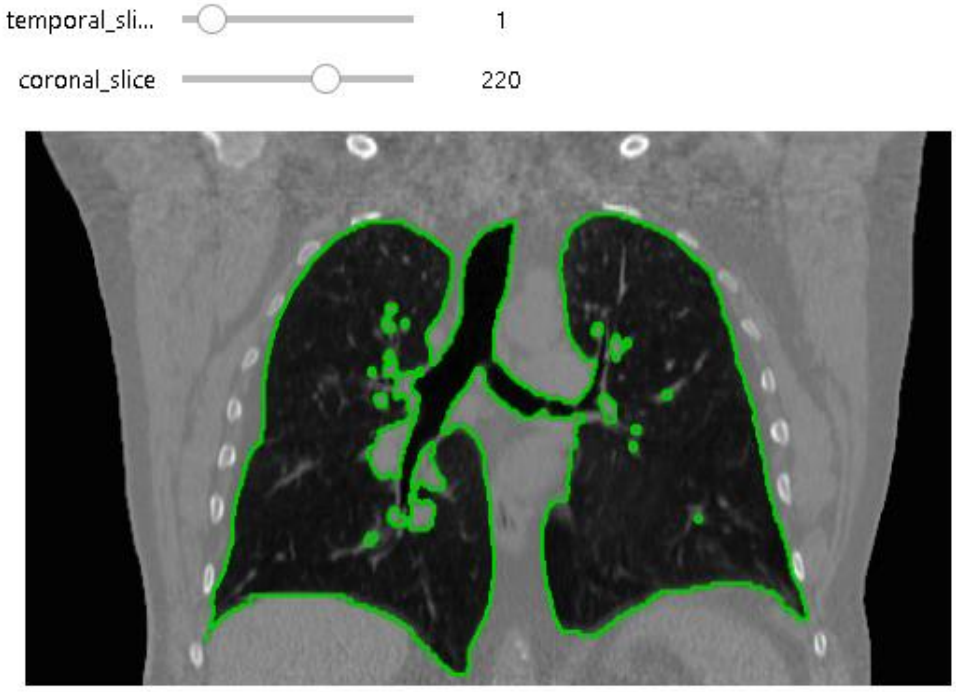
Nome: Marcio Campos N° USP: 11871625
Disciplina: Processamento e análise de imagens médicas
Relatório: Registro de imagens

Questões:

1) Registration_Demons.ipynb - Descrição do Processo de Registro Não-Rígido com o Algoritmo Demons

O objetivo principal do método apresentado no arquivo é alinhar de forma precisa volumes de tomografia computadorizada (CT) 4D do modelo POPI (Point-validated Pixel-based Breathing Thorax Model), que consiste em conjuntos de imagens temporais, máscaras segmentadas e pontos de referência correspondentes, as etapas presentes no arquivo consiste em:

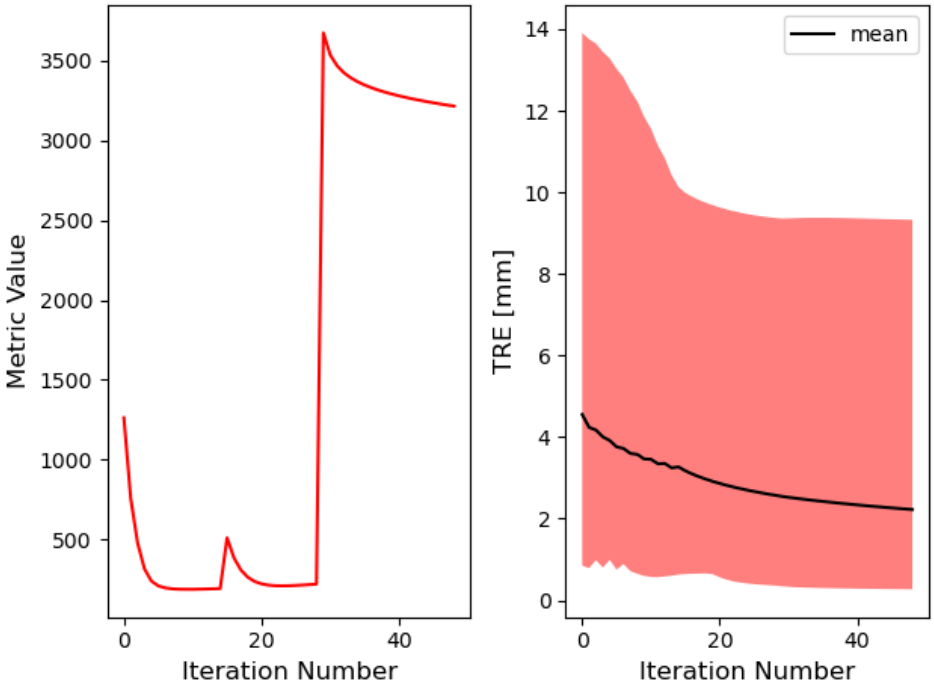
O script utiliza a função interact da biblioteca *ipywidgets* para criar uma interface interativa que permite visualizar as fatias coronais das imagens com sobreposição das máscaras. A interface permite selecionar dinamicamente as fatias temporais e coronais das imagens fixa e móvel, ajustando a visualização conforme necessário para melhor compreensão dos dados. Isso facilita a análise qualitativa do alinhamento entre as imagens e a identificação de regiões de interesse, como os pulmões.

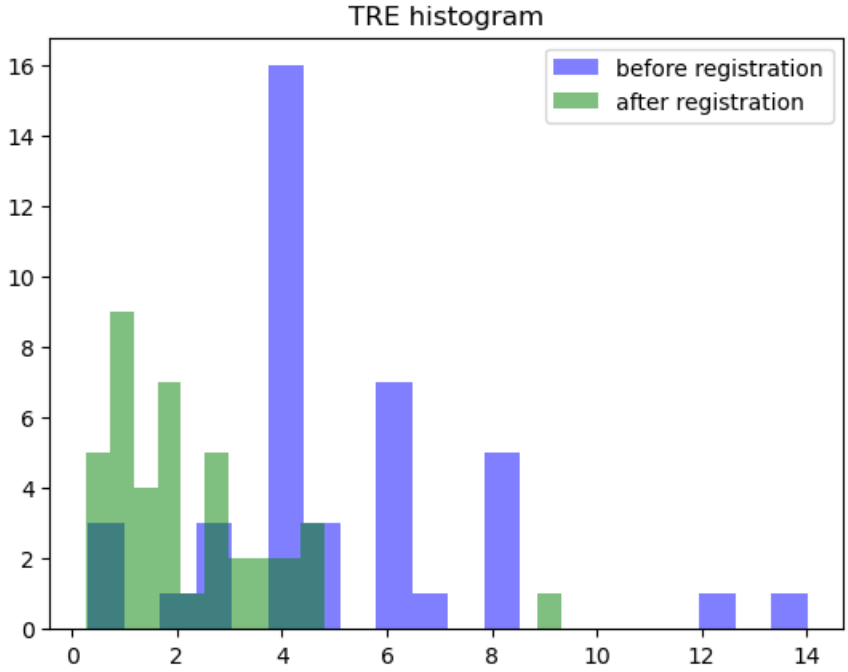


O script então define a função demons_registration, que encapsula a configuração e execução do método de registro Demons. Dentro dessa função, um objeto ImageRegistrationMethod é criado e configurado para utilizar a métrica Demons com um limite de intensidade de 10 HU, apropriada para imagens médicas onde intensidades similares indicam similaridade estrutural. A transformação inicial é definida como uma transformação de campo de deslocamento (DisplacementFieldTransform), iniciada com uma transformação identidade. A função configura um framework multi-resolução com fatores de encolhimento e sigmas de suavização progressiva, utilizando interpolação linear. O otimizador escolhido é o de

gradiente descendente, configurado com uma taxa de aprendizado de 1.0, 20 iterações e tolerâncias de convergência específicas, permitindo uma otimização refinada. Se pontos de referência correspondentes forem fornecidos, a função adiciona comandos de callback para monitorar a métrica de similaridade e os erros de registro durante as iterações.

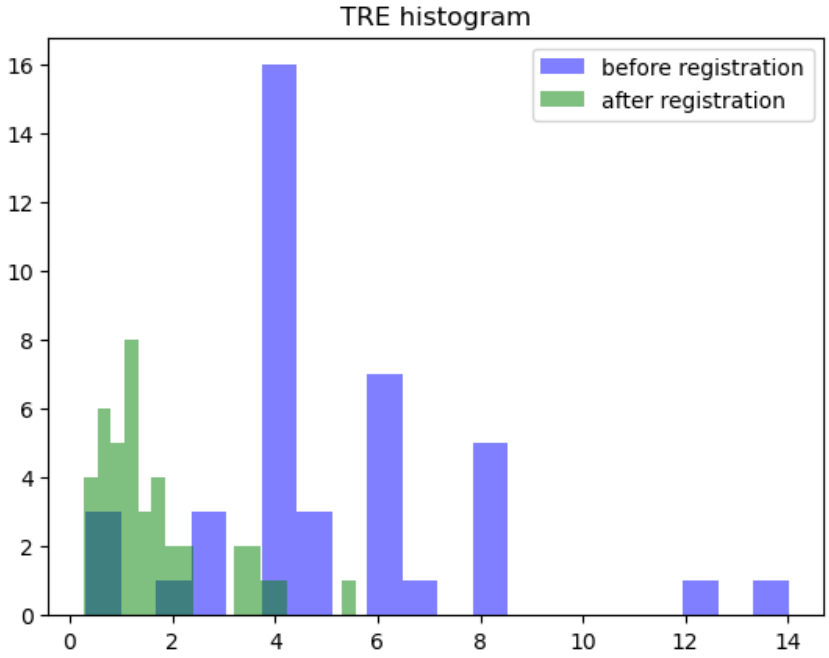
Após definir a função de registro Demons, o script seleciona índices específicos para as imagens fixa e móvel (por exemplo, índices 0 e 7). Executa o registro intra-modal utilizando a função demons_registration, aplicando pontos de referência correspondentes para melhorar a precisão do alinhamento. Os erros de registro são então calculados antes e após o registro utilizando a função registration_errors, que compara os pontos fixos e móveis transformados, fornecendo estatísticas como erro médio, desvio padrão e erro máximo. Esses erros são visualizados através de histogramas que comparam os erros antes e após o registro, oferecendo uma representação gráfica da eficácia do processo de registro.





Em seguida, o script define funções auxiliares smooth_and_resample e multiscale_demons para realizar um registro Demons em múltiplas escalas. A função smooth_and_resample aplica um filtro gaussiano para suavizar a imagem e depois reamostra a imagem com um fator de encolhimento especificado, ajustando o espaçamento dos voxels para facilitar o processamento em diferentes resoluções. A função multiscale_demons utiliza essas funções para criar uma pirâmide de imagens em diferentes resoluções e executa o registro Demons em cada nível da pirâmide, refinando a transformação de deslocamento a cada passo. Isso permite uma abordagem mais eficiente e precisa, especialmente para imagens de alta resolução ou estruturas complexas.

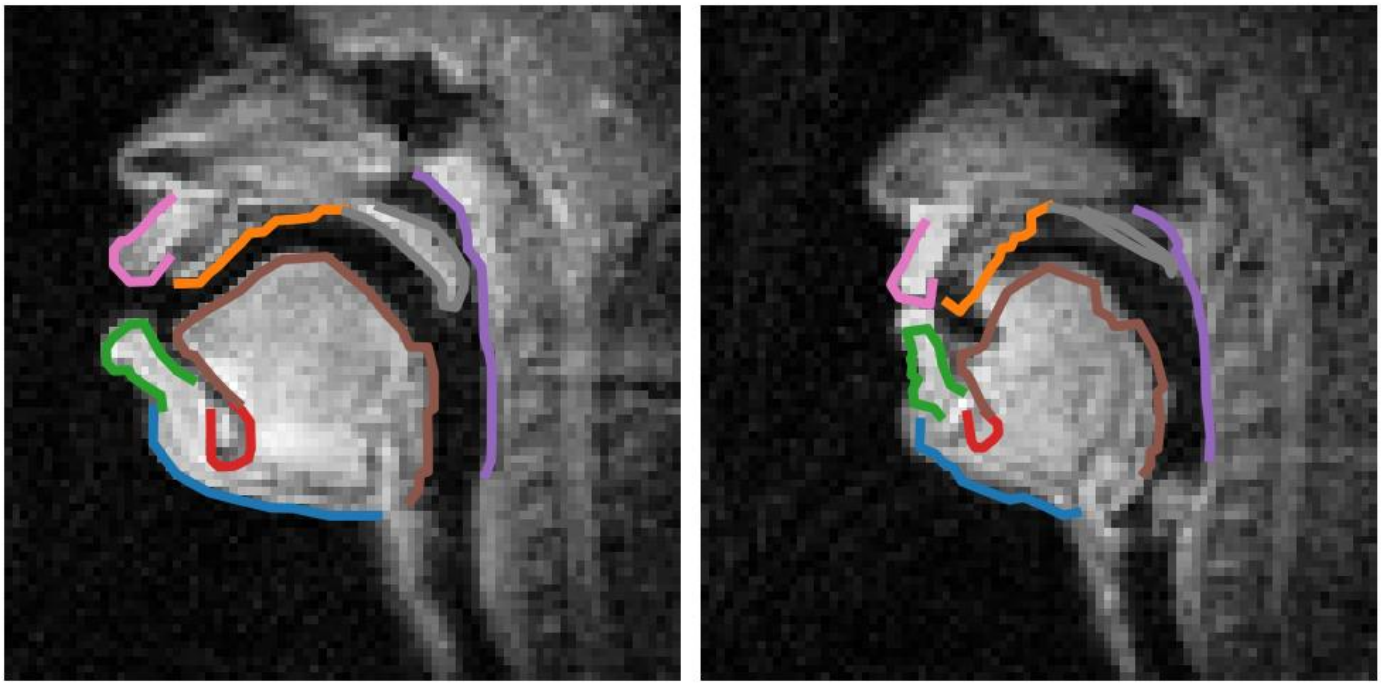
Posteriormente, o script configura e executa um filtro Demons simétrico (FastSymmetricForcesDemonsRegistrationFilter) com 20 iterações e regularização definida para um campo de deslocamento viscoso e elástico. O filtro é configurado para suavizar o campo de deslocamento durante as atualizações e ajustar as deslocamentos totais com base em desvios padrão específicos. Após configurar o filtro, ele é adicionado ao processo de registro multi-resolução definido anteriormente, permitindo que o registro seja realizado de forma eficiente através das diferentes escalas da pirâmide de imagens. O registro é executado entre as imagens fixas e móveis selecionadas, e os erros de TRE são novamente calculados e comparados antes e após o registro, fornecendo uma avaliação quantitativa da melhoria no alinhamento das imagens.



Além disso, o script inclui a leitura e visualização de imagens segmentadas e novas imagens com contornos sobrepostos. Utilizando a função multi_image_display2D, ele exibe as imagens segmentadas e novas imagens em uma interface gráfica 2D, onde os contornos das segmentações são plotados sobre as imagens para facilitar a visualização das áreas segmentadas. Em seguida, realiza um registro difeomórfico Demons (DiffeomorphicDemonsRegistrationFilter) entre a imagem segmentada e a nova imagem, aplicando uma transformação inicial centralizada utilizando uma transformação rígida (Euler2DTransform). Após o registro

difeomórfico, os contornos transformados são plotados sobre as novas imagens, permitindo uma visualização direta do impacto do registro nas segmentações.

Finalmente, o script avalia a sobreposição das segmentações antes e após o registro utilizando métricas como o Coeficiente de Dice e a Distância de Hausdorff. Utilizando o LabelOverlapMeasuresImageFilter, calcula o Coeficiente de Dice, que mede a similaridade entre as segmentações dos pulmões na imagem fixa e na imagem móvel antes e após o registro. Além disso, a Distância de Hausdorff é calculada usando o HausdorffDistanceImageFilter, fornecendo uma medida da máxima distância entre as fronteiras das segmentações, antes e após o registro. Essas métricas quantitativas fornecem uma avaliação objetiva da melhoria no alinhamento e na sobreposição das segmentações resultantes do processo de registro.



2) Registration_Errors.ipynb

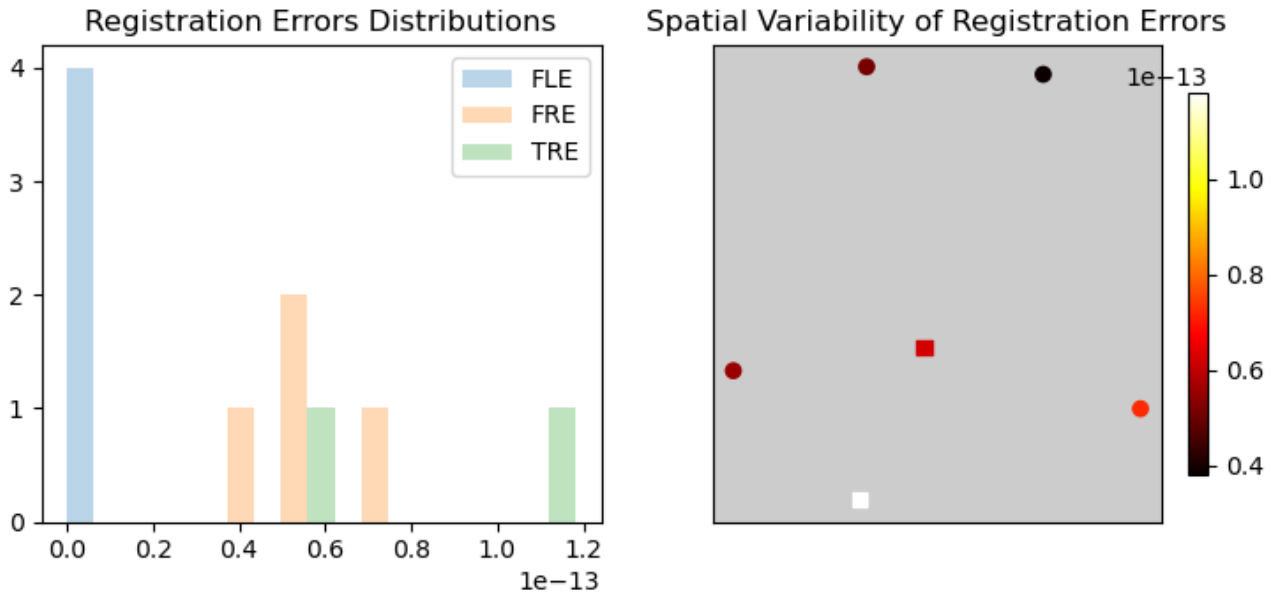
Inicialmente, o script define conjuntos ideais de pontos fiduciais e alvos para as imagens fixa e móvel. Esses conjuntos representam correspondências entre as duas imagens. As listas ideal_fixed_fiducials e ideal_fixed_targets contêm as coordenadas dos pontos na imagem fixa, enquanto ideal_moving_fiducials e ideal_moving_targets correspondem aos pontos na imagem móvel. Em seguida, essas listas são copiadas para garantir que as versões originais permaneçam inalteradas durante as manipulações subsequentes.

Para preparar os dados para registro, os pontos fiduciais são achatados em listas (fixed_fiducials_flat e moving_fiducials_flat), conforme exigido pela função LandmarkBasedTransformInitializer. Esta função inicializa uma transformação rígida (Euler2DTransform) baseada na correspondência entre os pontos fiduciais das imagens fixa e móvel. A transformação resultante alinha geometricamente as duas imagens com base nos pontos fornecidos.

O script então calcula os erros de registro de marcos (FRE) e de alvos (TRE) utilizando a transformação inicial. Os erros são avaliados antes de qualquer registro adicional ser aplicado, fornecendo uma linha de base para comparação. A função

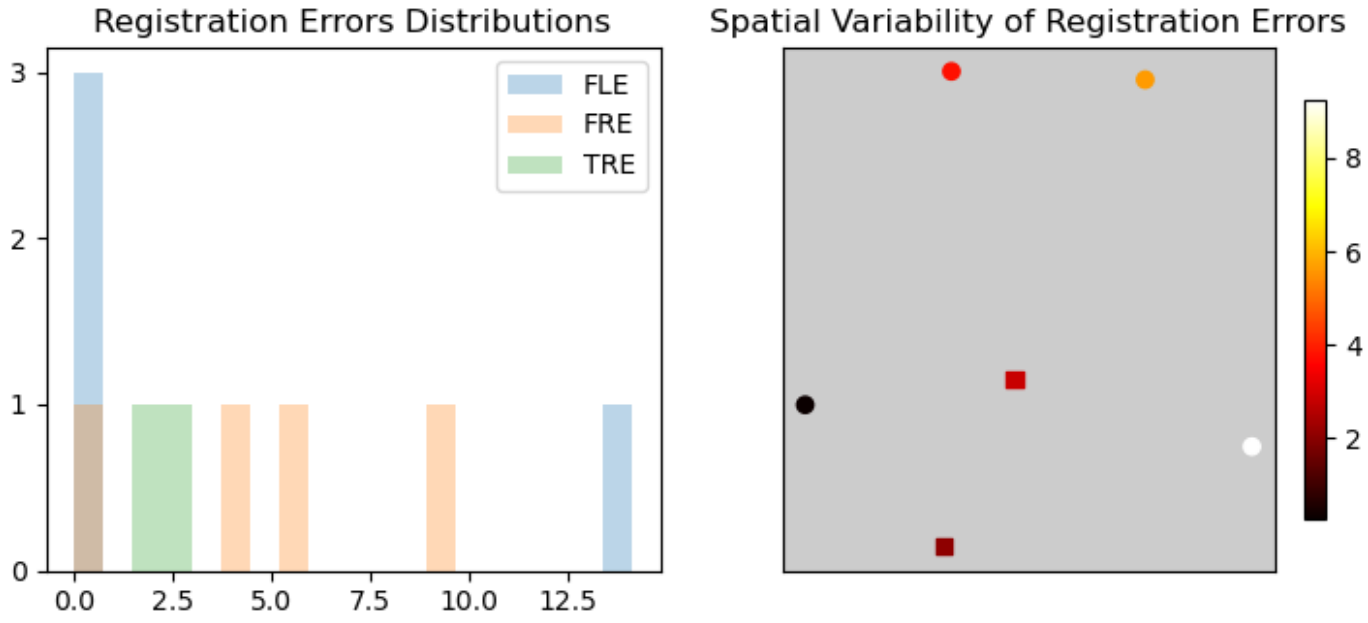
display_errors é utilizada para visualizar esses erros, apresentando-os de forma gráfica com o título "Ideal Input", que reflete a condição ideal.

Ideal Input

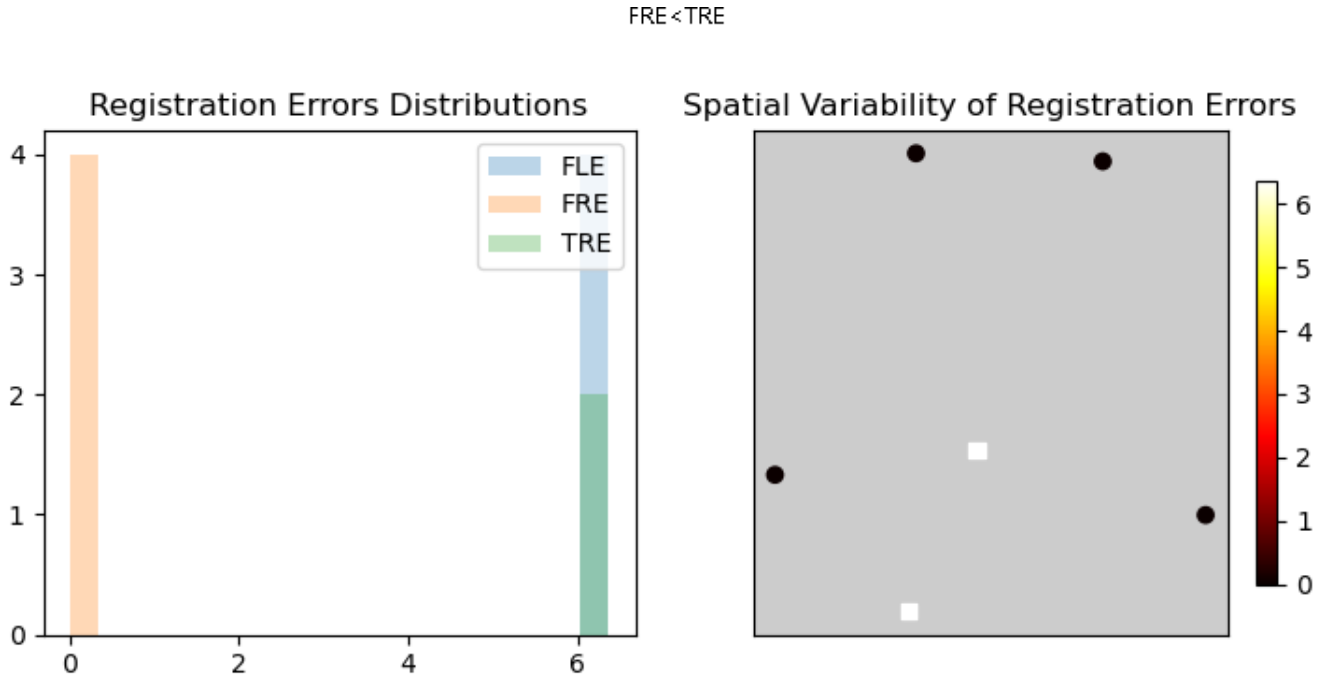


Em seguida, o script introduz um outlier nos pontos fiduciais da imagem móvel. Especificamente, o quarto ponto fiducial na lista moving_fiducials é alterado para uma posição anômala ([88.07960498849866, 22.34621428425981]). Este outlier representa uma discrepância significativa em relação aos pontos originais, simulando uma situação em que um dos pontos de referência está incorretamente posicionado ou é afetado por ruído. Após a introdução do outlier, a transformação rígida é recalculada com os pontos fiduciais modificados, e os erros de FRE e TRE são novamente calculados e exibidos. O título "Single Outlier" indica que apenas um ponto fiducial foi afetado, permitindo observar como um único outlier pode impactar a precisão do registro.

Single Outlier

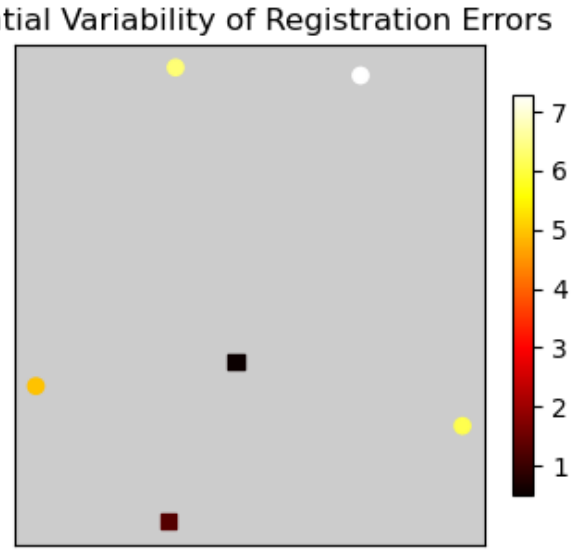


Posteriormente, o script avalia o impacto de um viés sistemático aplicado a todos os pontos fiduciais da imagem móvel. Um vetor de viés [4.5, 4.5] é adicionado a cada coordenada dos pontos fiduciais, deslocando-os uniformemente na direção definida pelo vetor. Este viés simula uma situação onde todas as medições ou posicionamentos dos pontos estão consistentemente deslocados, possivelmente devido a erros de calibração ou outros fatores sistemáticos. Após a aplicação do viés, a transformação rígida é novamente inicializada e os erros de FRE e TRE são calculados. A visualização é rotulada como "FRE<TRE" para indicar que, embora o FRE possa permanecer baixo devido à correspondência direta dos fiduciais deslocados, o TRE pode aumentar devido ao deslocamento sistemático dos alvos.



Além disso, o script explora uma condição onde metade dos pontos fiduciais recebem um viés positivo e a outra metade um viés negativo, criando um padrão de deslocamento oposto para diferentes pontos. Esta situação representa um cenário mais complexo onde os erros não são uniformes e afetam diferentes regiões da imagem de maneiras distintas. Após ajustar os pontos fiduciais de acordo com esse padrão, a transformação rígida é recalculada e os erros de FRE e TRE são avaliados novamente. A visualização é intitulada "FRE>TRE", sugerindo que, nesta condição, tanto o FRE quanto o TRE podem ser significativamente afetados, refletindo a complexidade adicional introduzida pelos deslocamentos opostos.

FRE > TRE



Em etapas posteriores do script, são definidos novos conjuntos de pontos fiduciais e alvos, seguidos pela utilização de uma interface de manipulação de pontos (PairedPointDataManipulation). Este componente permite interagir manualmente com os pontos de referência, ajustando-os conforme necessário para refinar a transformação inicial. Ao selecionar subconjuntos de pontos fiduciais e alvos, é possível explorar como diferentes combinações de pontos influenciam a precisão do registro.

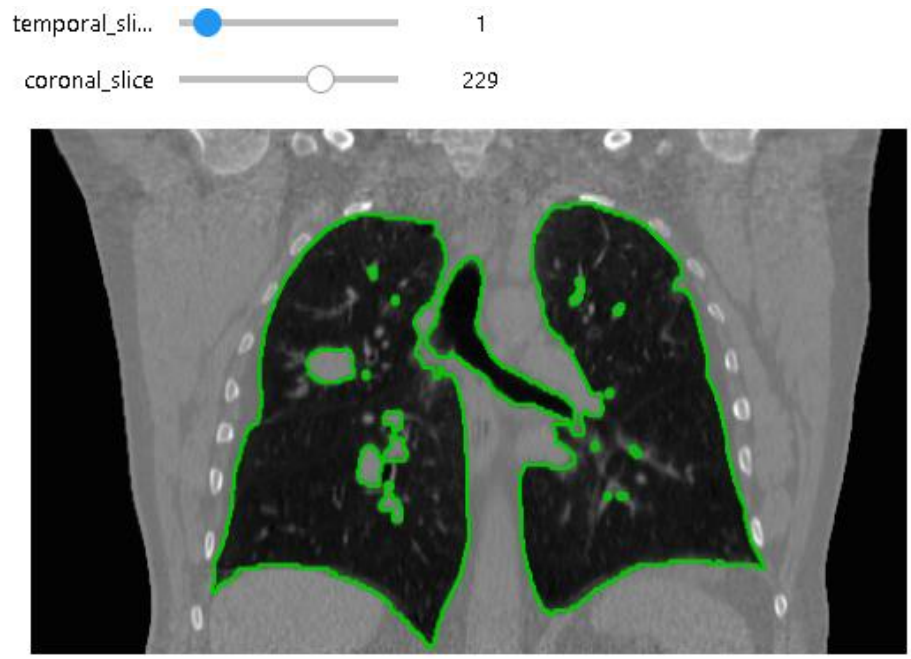
3) Registration_FFD.ipynb

O objetivo principal deste script é alinhar imagens provenientes de diferentes modalidades, como tomografia computadorizada (CT) e ressonância magnética (MR), focando especificamente na segmentação dos pulmões. O script integra várias etapas, incluindo a leitura e preparação das imagens, aplicação de máscaras, aquisição de pontos de referência, execução de registros intra-modais utilizando transformações BSpline, e avaliação da precisão do registro através de métricas como o Coeficiente de Dice e a Distância de Hausdorff. A seguir, descrevo detalhadamente cada parte do script e suas funcionalidades.

Inicialmente, o script cria listas para armazenar imagens, máscaras e pontos de referência. Utilizando um loop que percorre os índices de 0 a 9, ele lê arquivos de imagem, máscaras e pontos de referência correspondentes a cada caso. As máscaras são carregadas sem conversão de formato, preservando suas propriedades originais. Os pontos de referência são lidos utilizando uma função personalizada read_POPI_points, que provavelmente interpreta arquivos de pontos específicos para cada imagem. Essa etapa prepara os dados necessários para o registro, garantindo que cada imagem móvel tenha sua respectiva máscara e conjunto de pontos fiduciais associados.

Em seguida, o script cria uma interface interativa que permite visualizar as fatias coronais das imagens com sobreposição das máscaras. Essa visualização facilita a análise qualitativa do alinhamento entre as imagens e a identificação de regiões de

interesse, como os pulmões. A interface permite selecionar dinamicamente as fatias temporais e coronais das imagens fixa e móvel, ajustando a visualização conforme necessário para melhor compreensão dos dados.

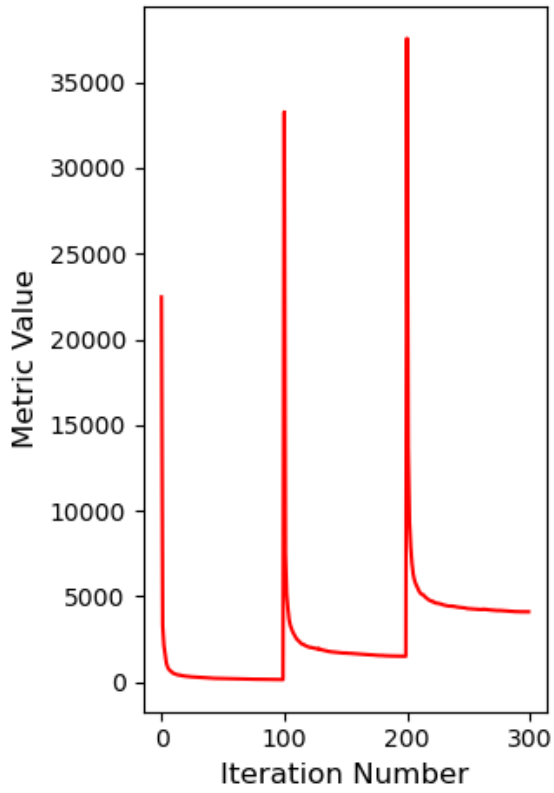


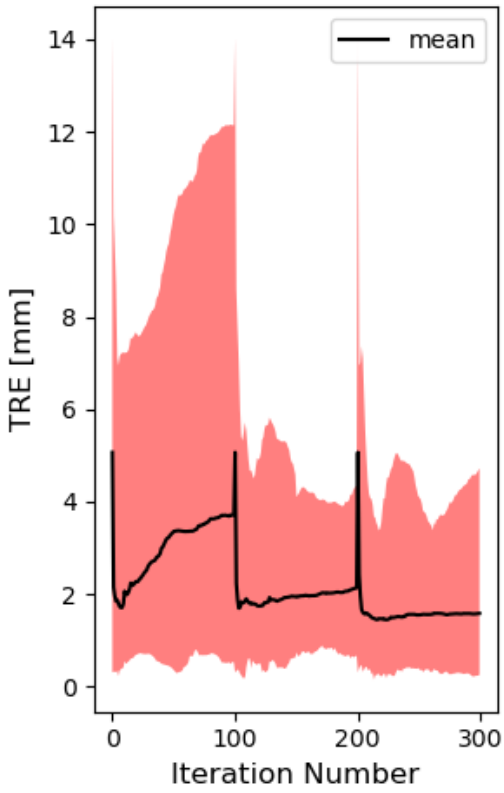
O script então aplica um filtro de estatísticas de forma (LabelShapeStatisticsImageFilter) às máscaras carregadas, calculando o volume dos pulmões em cada imagem. Esse cálculo é realizado multiplicando o tamanho físico dos labels (segmentações) por um fator de conversão para obter o volume em litros. Essa métrica fornece uma avaliação quantitativa da extensão das estruturas segmentadas, permitindo comparações entre diferentes imagens e registros. O resultado foi dado como:

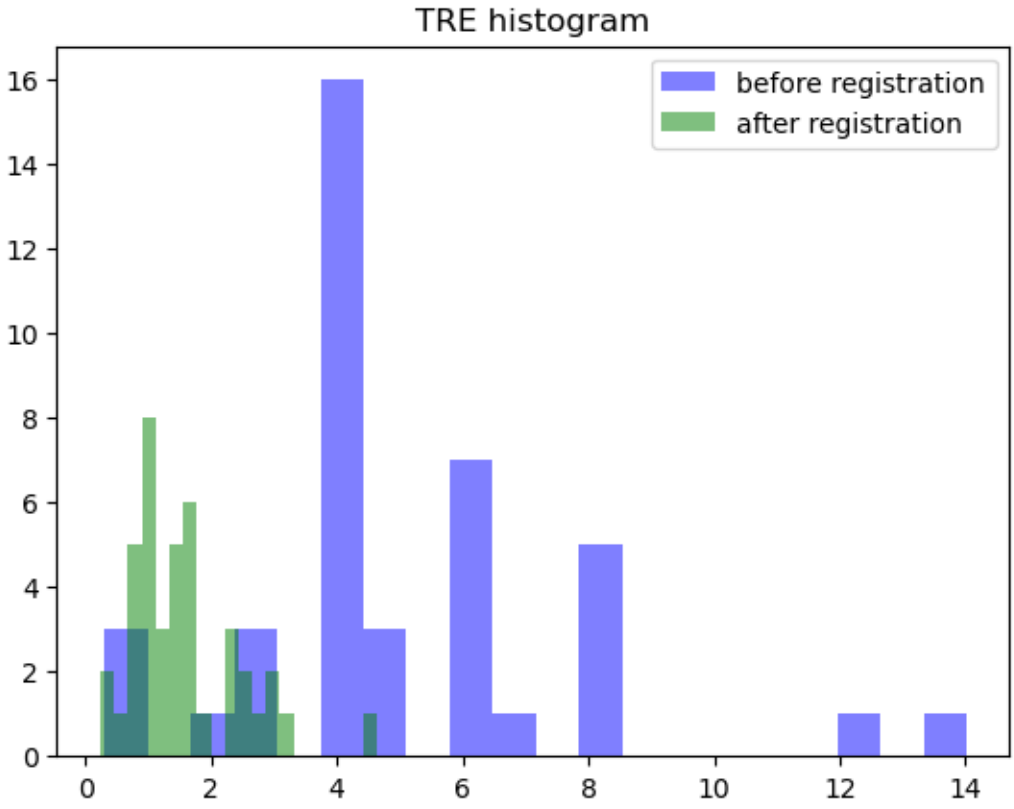
Lung volume in image 0 is 5.455734929689075 liters.
Lung volume in image 1 is 5.527017905172941 liters.
Lung volume in image 2 is 5.554014379499289 liters.
Lung volume in image 3 is 5.451784830895104 liters.
Lung volume in image 4 is 5.301415956621658 liters.
Lung volume in image 5 is 5.191841246039885 liters.
Lung volume in image 6 is 5.09130732168864 liters.
Lung volume in image 7 is 5.11128669632256 liters.
Lung volume in image 8 is 5.195802788867059 liters.
Lung volume in image 9 is 5.335378032491021 liters.

A seguir, são definidas duas funções de registro intra-modal utilizando transformações BSpline (bspline_intra_modal_registration e bspline_intra_modal_registration2). A primeira função inicializa uma transformação BSpline com espaçamento físico de 50mm entre os pontos de controle, ajustando o tamanho da malha com base no tamanho físico da imagem fixa. Ela configura o método de registro para usar a métrica de Média dos Quadrados (Mean Squares) e uma estratégia de amostragem aleatória de 1% dos pixels, opcionalmente utilizando

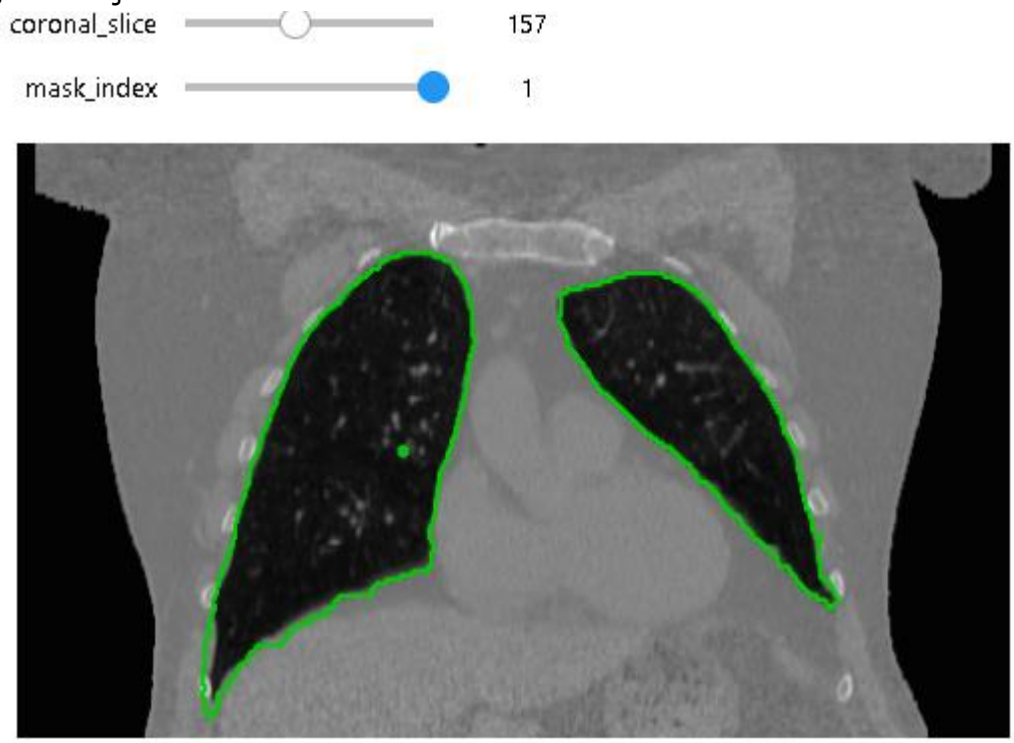
uma máscara fixa para restringir a amostragem a regiões de interesse, como os pulmões. A função também configura um framework multi-resolução com fatores de encolhimento e sigmas de suavização progressiva, utilizando interpolação linear e um otimizador LBFGS com tolerâncias de convergência específicas. Se pontos de referência correspondentes forem fornecidos, a função adiciona comandos de callback para monitorar a métrica de similaridade e os erros de registro durante as iterações.





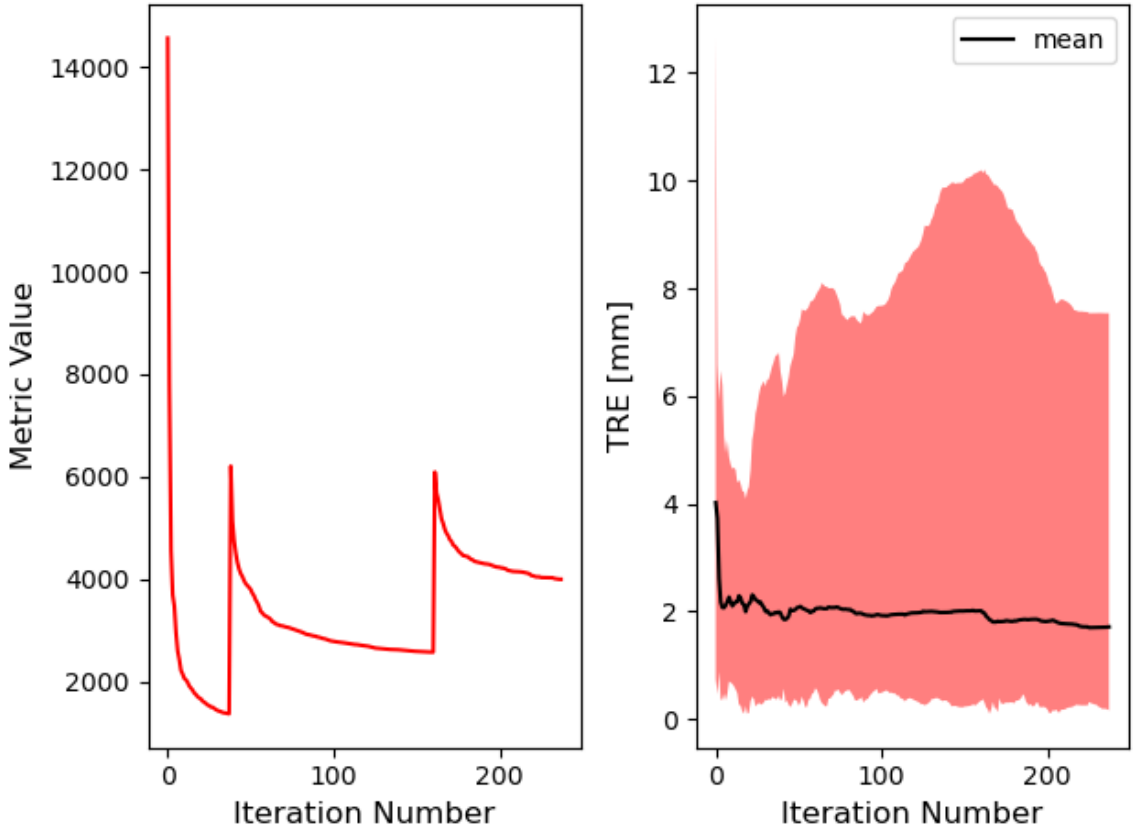


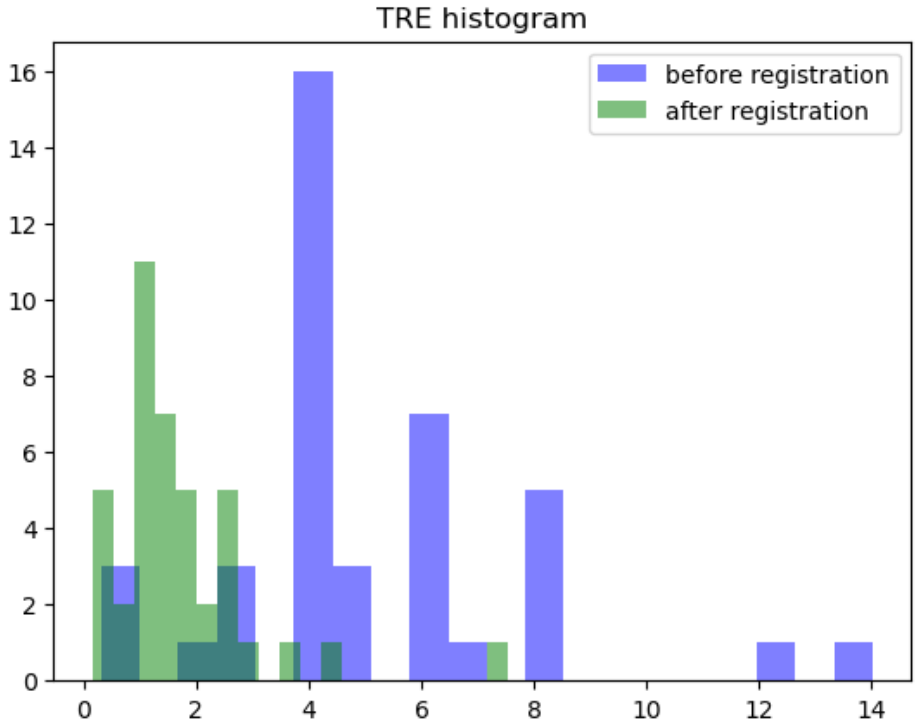
Além disso, o script realiza a transferência das segmentações dos pulmões através da transformação estimada. Utilizando a função Resample do *SimpleITK*, ele aplica a transformação BSpline à máscara da imagem móvel, utilizando interpolação de vizinho mais próximo para preservar a integridade das labels segmentadas. As segmentações antes e depois do registro são então exibidas interativamente através de outra interface gráfica, permitindo a visualização do impacto do registro nas segmentações.



A segunda função de registro (bspline_intra_modal_registration2) aprimora a primeira ao ajustar o tamanho inicial da malha BSpline, reduzindo-o para um quarto do tamanho original e refinando-o através de múltiplas resoluções. Ela utiliza o otimizador LBFGS2, que é mais adequado para adaptações na resolução da malha durante a otimização. Além disso, a função inclui callbacks para monitorar a métrica e os erros de registro, similar à primeira função.

Após definir as funções de registro, o script seleciona índices específicos para as imagens fixa e móvel (por exemplo, índices 0 e 7). Ele executa o registro intra-modal utilizando a primeira função de registro BSpline, aplicando uma máscara específica dos pulmões e utilizando pontos de referência correspondentes. Os erros de registro são então calculados antes e após o registro, permitindo a avaliação da melhoria no alinhamento das imagens. Esses erros são visualizados através de histogramas que comparam os erros antes e depois do registro, fornecendo uma representação gráfica da eficácia do processo de registro.



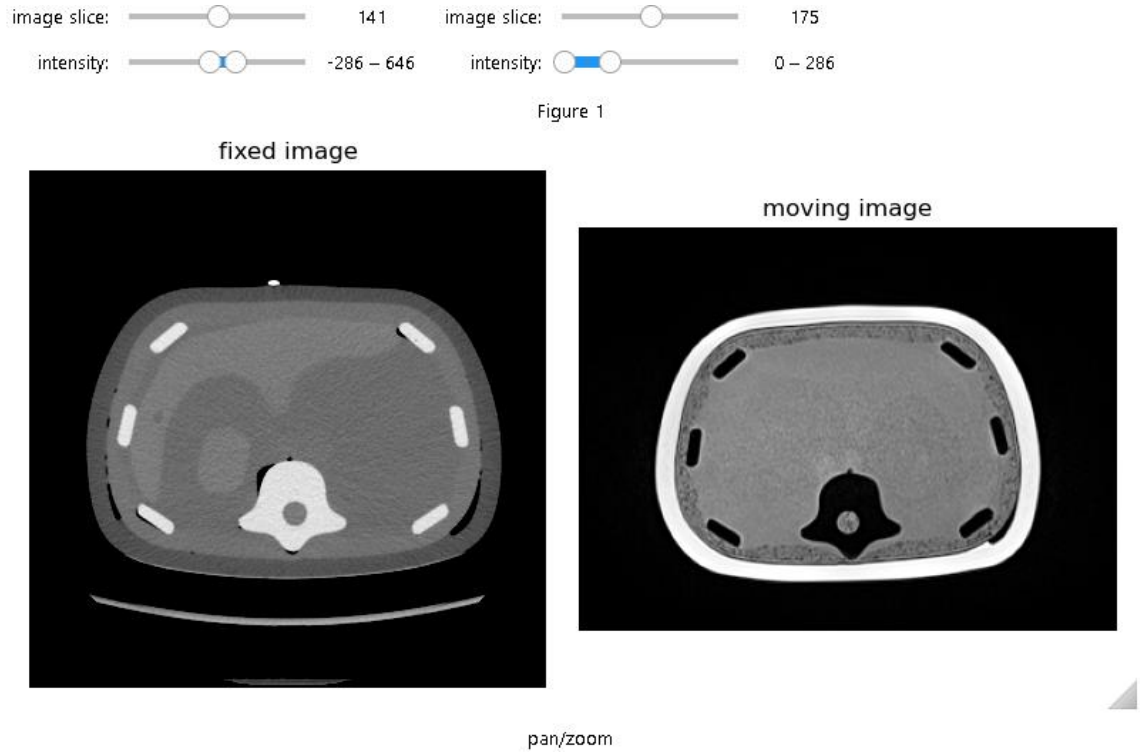


4) Registration_Initialization.ipynb

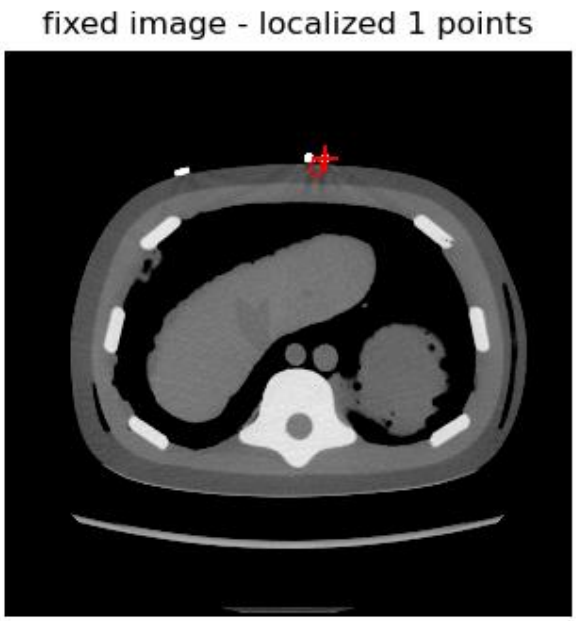
O script começa importando as bibliotecas necessárias, incluindo o *SimpleITK*, *numpy*, *os*, e componentes de interface gráfica como *ipywidgets* e *gui*. Também executa scripts auxiliares para configurar o ambiente de teste e atualizar caminhos para scripts de download de dados. A função multires_registration é definida para encapsular a configuração e execução de um registro multi-resolução, onde parâmetros como a métrica de informação mútua, amostragem aleatória, interpolador linear,

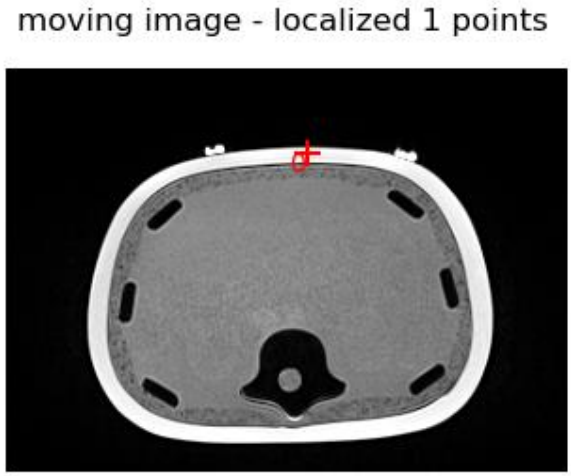
otimizador de gradiente descendente com taxa de aprendizado de 1.0 e 100 iterações são estabelecidos. Além disso, são definidos fatores de encolhimento e sigmas de suavização para diferentes níveis de resolução, permitindo uma abordagem progressiva que melhora a eficiência e a precisão do registro.

Em seguida, o script localiza o diretório de dados e lê as séries de imagens fixa (por exemplo, CT) e móvel (por exemplo, MR T1) utilizando o *ImageSeriesReader* do *SimpleITK*. Para facilitar a visualização, aplica-se o ajuste de janela/nível nas imagens e exibe-as utilizando uma interface gráfica personalizada (gui.MultiImageDisplay). A transformação inicial é estimada usando o CenteredTransformInitializer com uma transformação Euler 3D, centralizando a transformação no alinhamento geométrico das imagens.

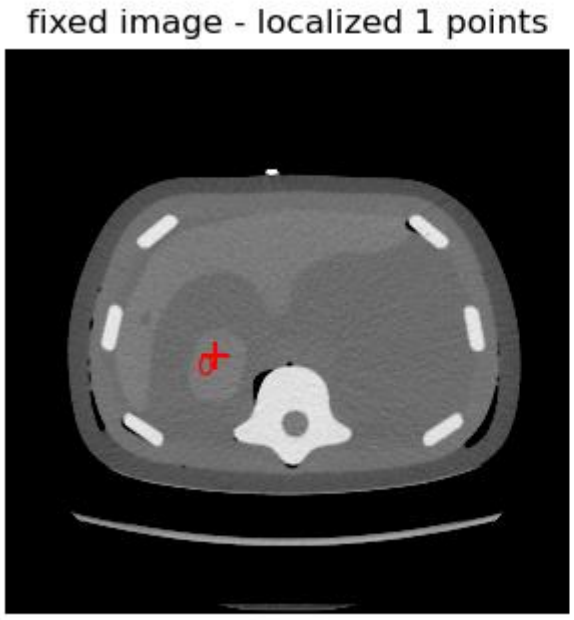


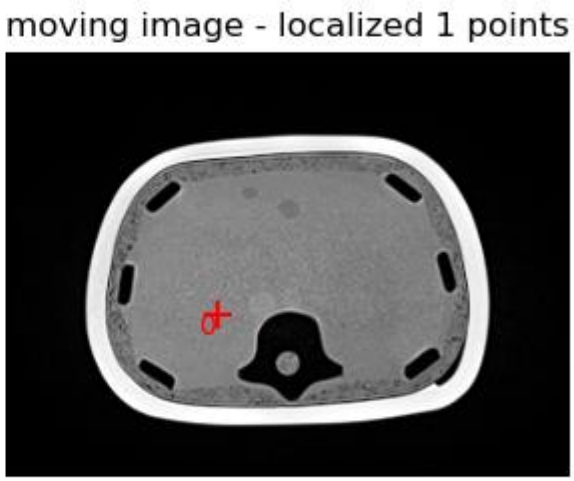
Após a inicialização, o script executa o registro multi-resolução através da função multires_registration, obtendo a transformação final e exibindo informações sobre a métrica final e a condição de parada do otimizador. Em seguida, utiliza-se uma interface de aquisição de pontos de registro (gui.RegistrationPointDataAquisition) para coletar pontos de referência manualmente ou utilizar pontos previamente localizados para avaliar a transformação.





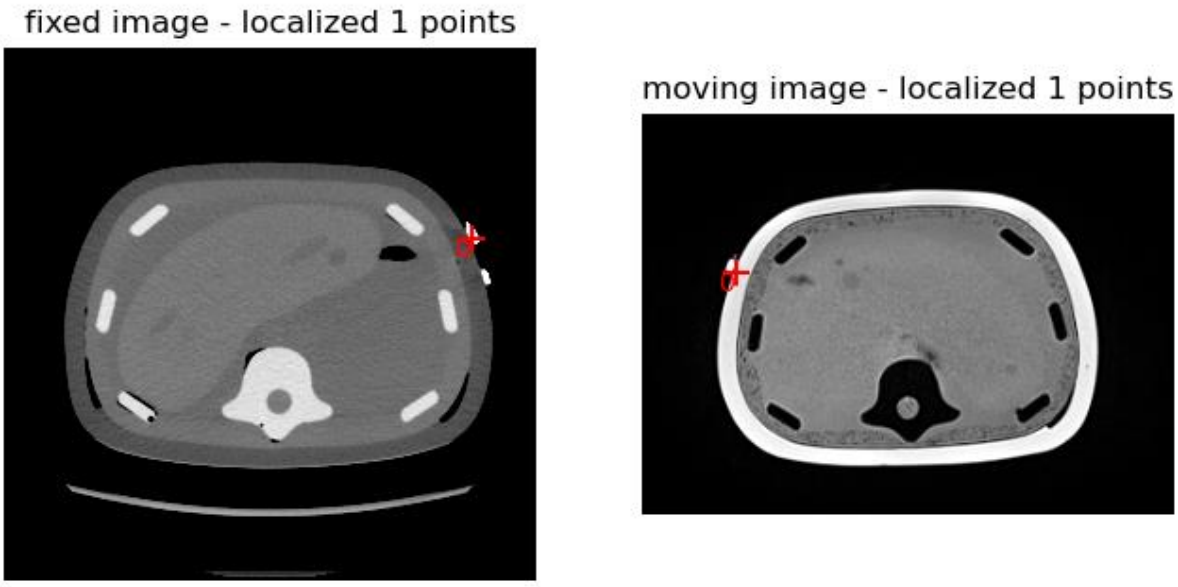
Uma característica notável deste script é a definição de um dicionário all_orientations que contém várias orientações de rotação pré-definidas. O script avalia a métrica de similaridade para cada uma dessas orientações, utilizando uma abordagem de amostragem em espaço de parâmetros de rotação. Inicialmente, seleciona a melhor orientação com base na métrica de similaridade mínima e, posteriormente, utiliza a multiprocessamento (ThreadPool) para avaliar de forma eficiente todas as orientações possíveis, identificando a orientação que proporciona a melhor métrica de similaridade.



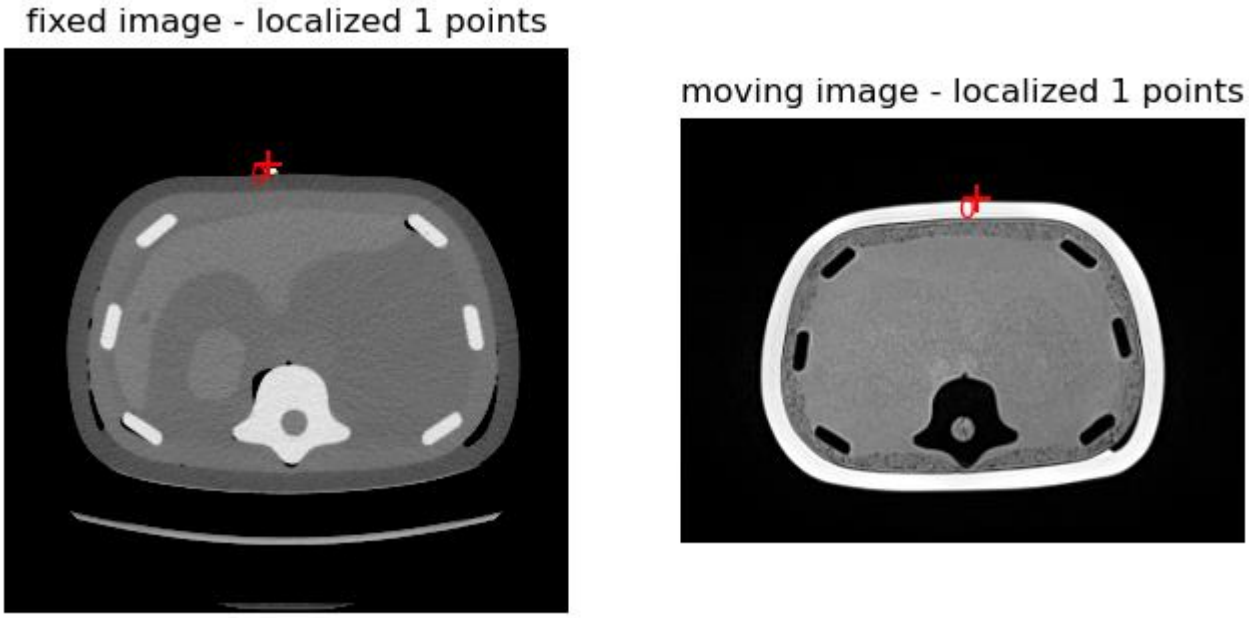


Após determinar a melhor orientação, o script atualiza a transformação inicial com essa orientação otimizada e realiza novamente o registro multi-resolução para refinar ainda mais a transformação. Em seguida, realiza uma busca exaustiva de parâmetros de otimização utilizando o otimizador Exhaustive, que explora uma grade de passos para

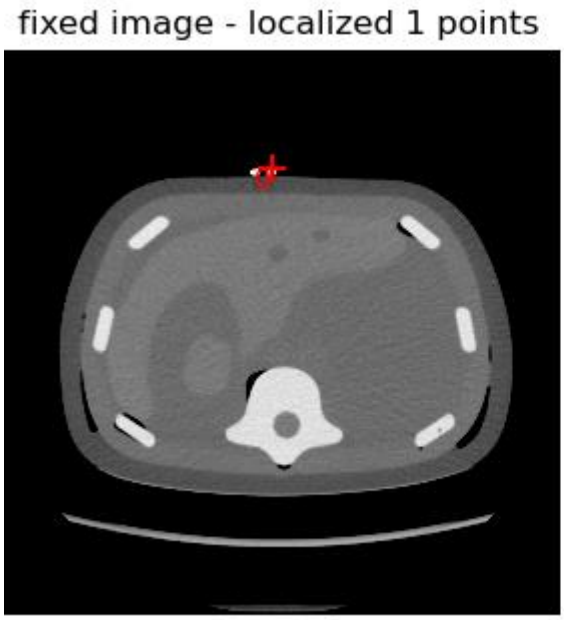
rotação e tradução, garantindo que a melhor configuração inicial seja encontrada para o registro subsequente.

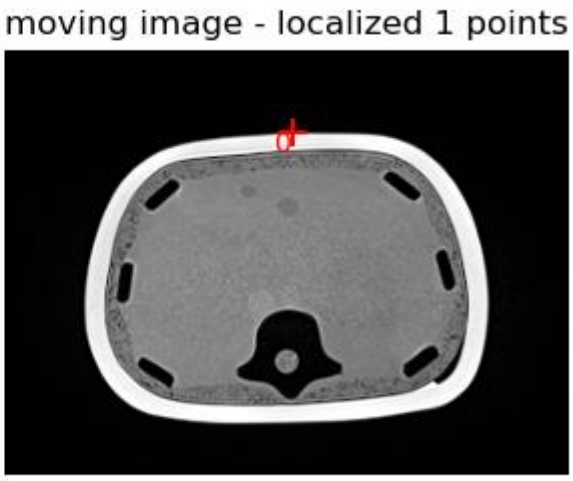


O script também implementa uma fase de exploração e exploração, onde durante a fase de exploração, observa-se os valores da métrica e os parâmetros da transformação em cada iteração, armazenando esses dados para análise posterior. Na fase de exploração, seleciona-se os melhores parâmetros obtidos e realiza-se um registro final com base nesses parâmetros, garantindo que a transformação final seja a mais precisa possível.

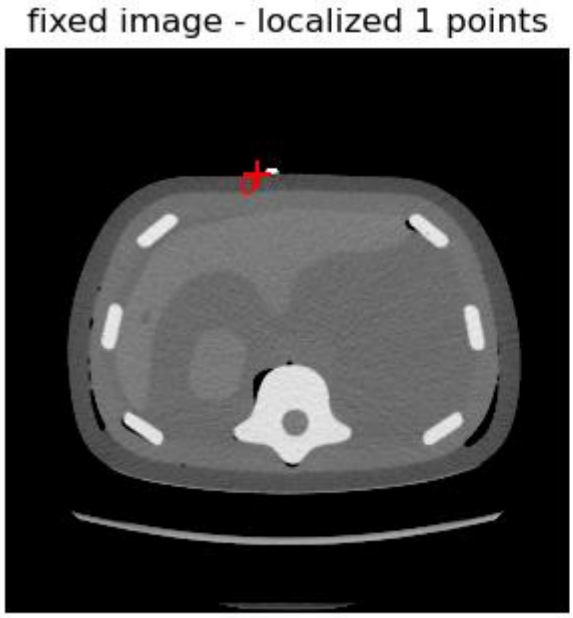


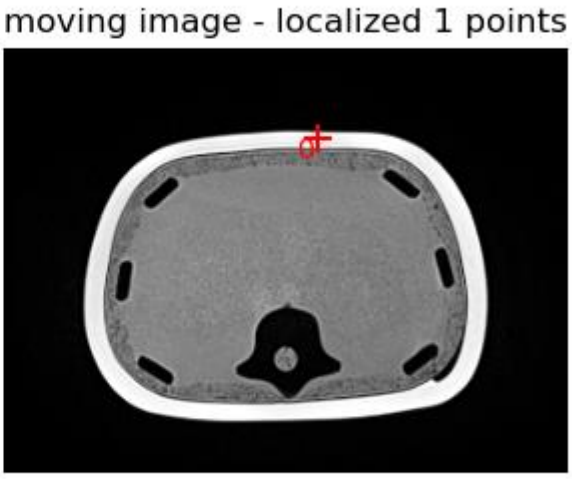
Além disso, o script inclui uma etapa de aquisição de pontos manuais, onde pontos de referência específicos são selecionados manualmente nas imagens fixa e móvel para inicializar uma transformação baseada em marcos (LandmarkBasedTransformInitializer). Essa transformação baseada em marcos é então convertida para uma transformação Euler 3D para facilitar a otimização.





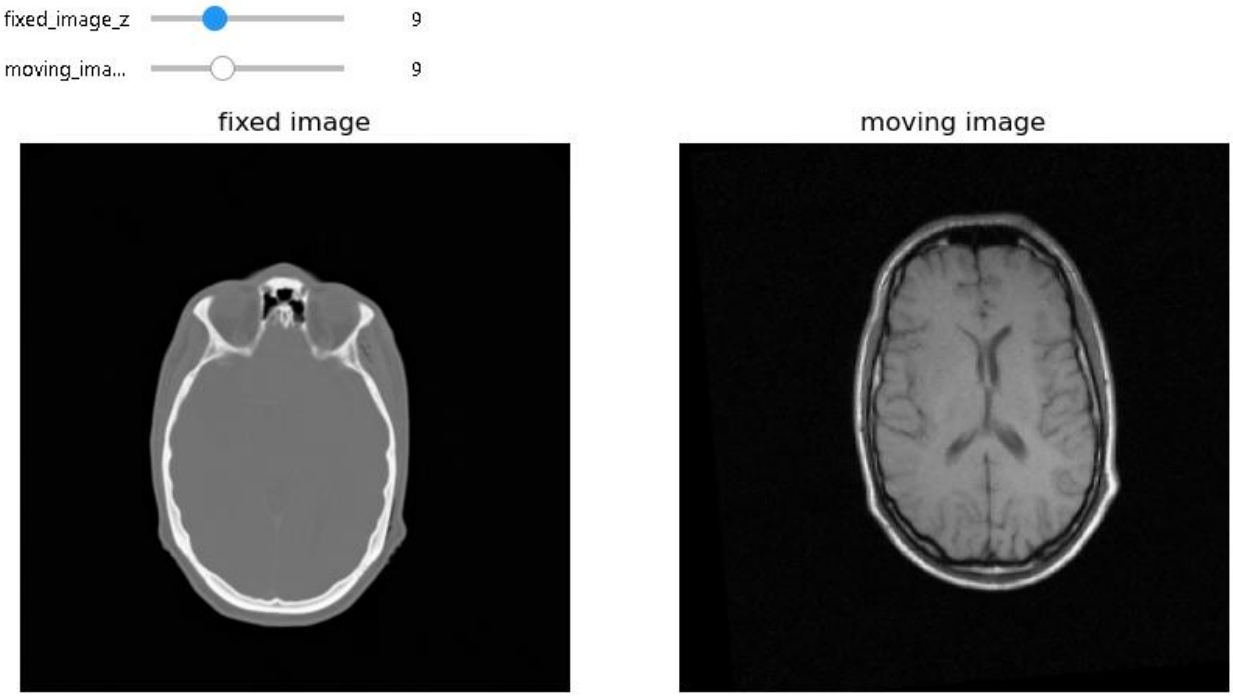
Por fim, o script realiza o registro de uma fatia expandida da imagem fixa, ajustando a transformação inicial com base no centro da fatia e na rotação conhecida, e executa o registro multi-resolução para alinhar essa fatia específica com a imagem móvel.



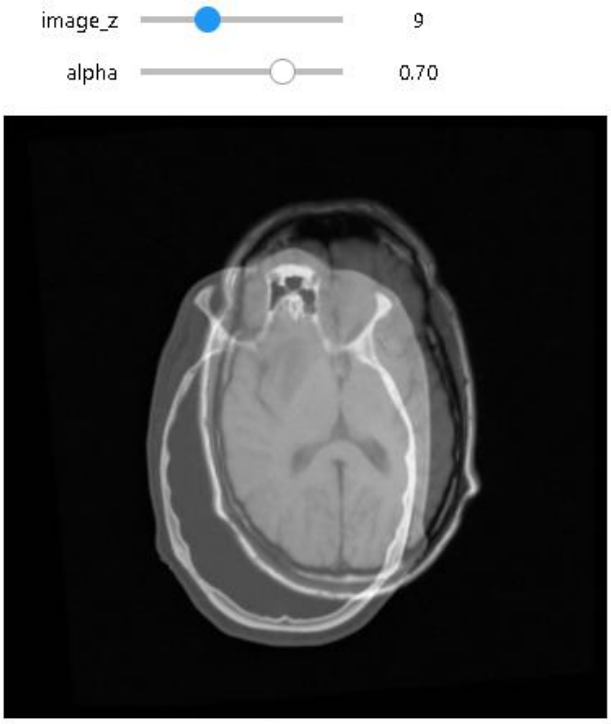


5) Registration_Introduction.ipynb

O método apresentado demonstra um registro entre uma imagem de tomografia e ressonância de cabeça. Desta forma, o script em um primeiro momento tem funções para obtenção das imagens da internet e defini métodos utilitários para fazer a visualizações das imagens. Posteriormente é possível visualizar estas imagens a partir de uma GUI e facilmente se percebe o deslocamento entre as fatias.

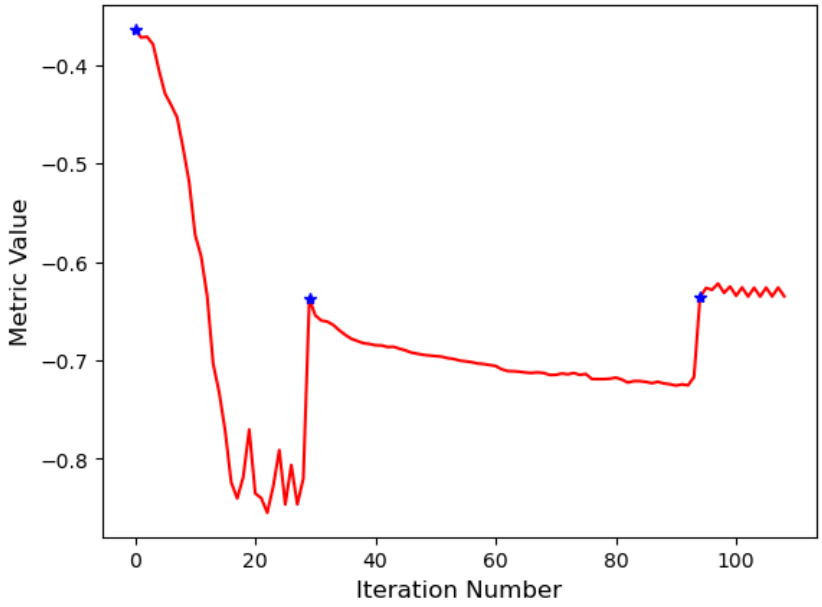


Logo, é feito uma primeira transformação da imagem móvel de forma a trazer um alinhamento inicial entre os centros geométricos das imagens usando o método CenteredTransformInitializer, como resultado foi possível visualizar da seguinte forma:

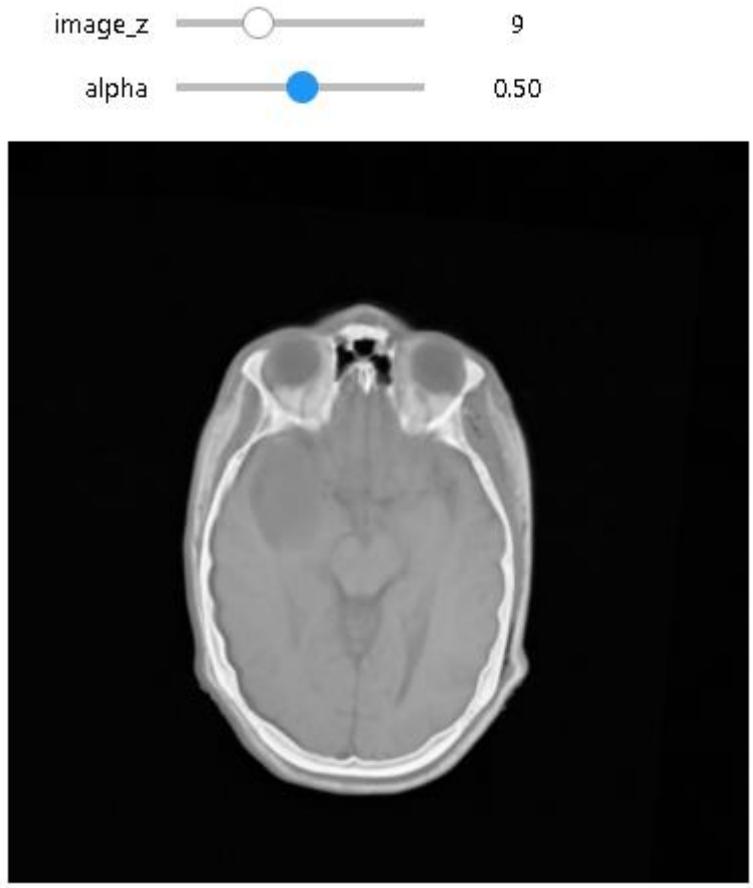


Ainda é muito claro que as imagens não estão registradas, isto porque o método não conseguiu assimilar as correspondências entre as duas imagens devido que são de modalidades diferentes, resultado em contraste de estruturas diferentes. Desta forma, um segundo método de registro é aplicado, este que funciona por interações e busca a convergência a partir da redução de uma função custo, chamado de Descida de Gradiente na qual a função custo utilizada é baseado na técnica chamada de Informação Mútua de Mattes que consegue identificar as similaridades das estruturas entre as diferentes modalidades de imagens com base na co-ocorrência das

intensidades entre as duas imagens e depende de um histograma conjunto. O valor da métrica utilizada ao longo das interações é plotado e pode ser visualizado abaixo:



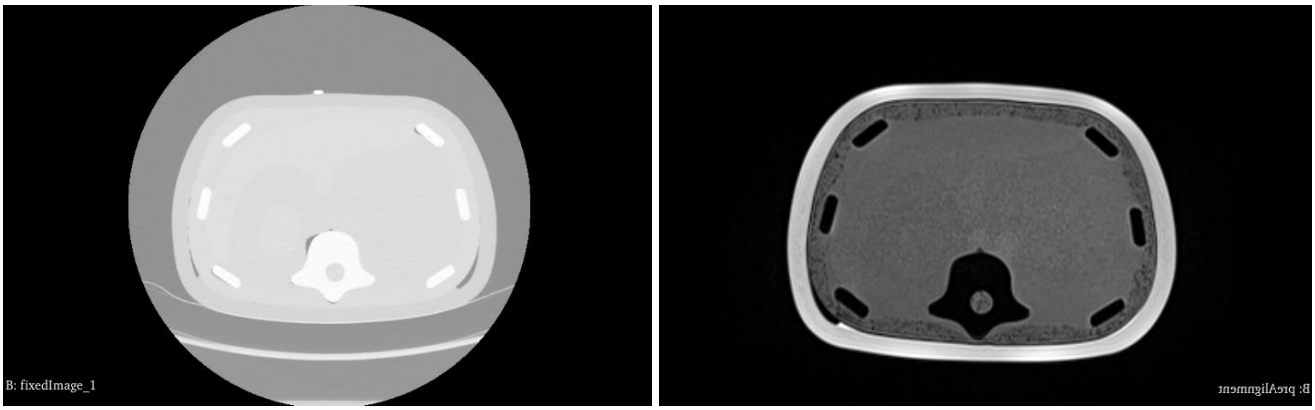
Por fim o registro entre as imagens foram feitos e os resultados são satisfatórios como pode ser visto abaixo:



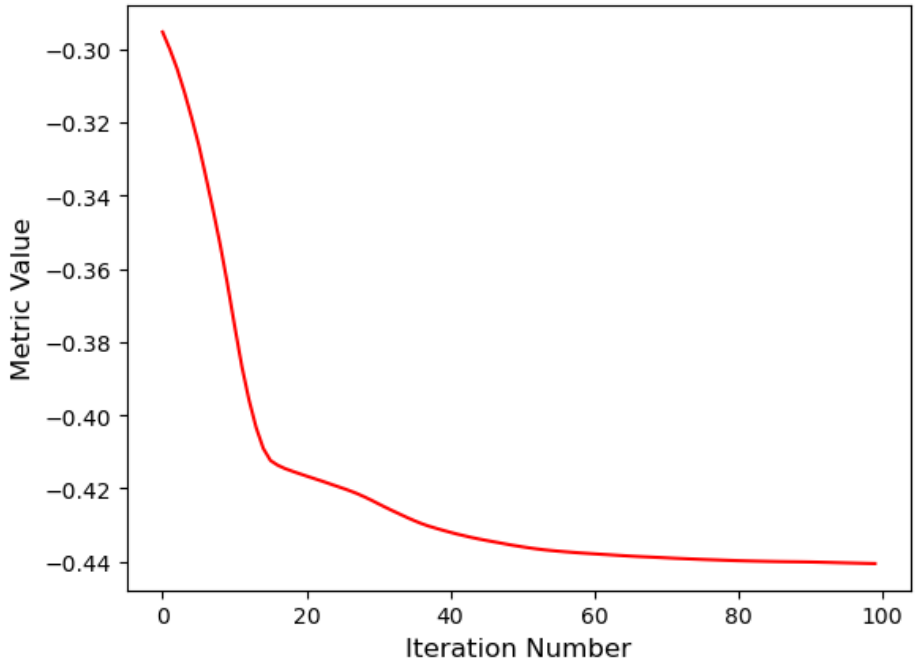
6) Registration_Introduction_Continued.ipynb

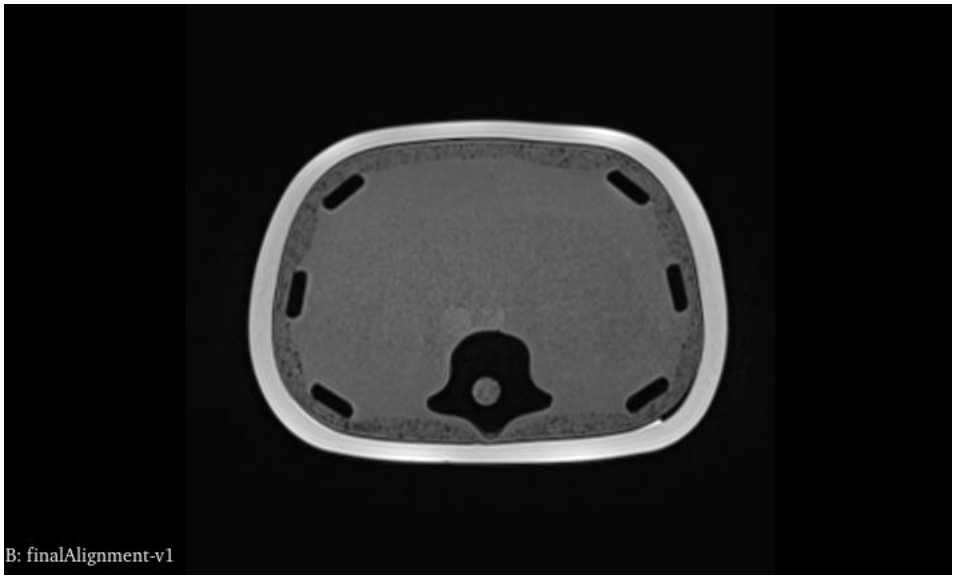
Neste notebook é demonstrado uma continuação da introdução de registro do tópico anterior, aqui é demonstrado 3 modelos de registro, cada abordagem com suas características próprias em relação à transformação inicial, final e às estratégias de otimização, incluindo o uso de multi-resolução e composição de transformações. Além

disto, não existe neste notebook funções de visualização de imagem, sendo recomendado pelo próprio utilizar ferramentas como ITK-Snap e 3D Slicer. Abaixo estão as imagens a serem registradas, sendo a da esquerda a fixa e a da direita a que será alinhada.

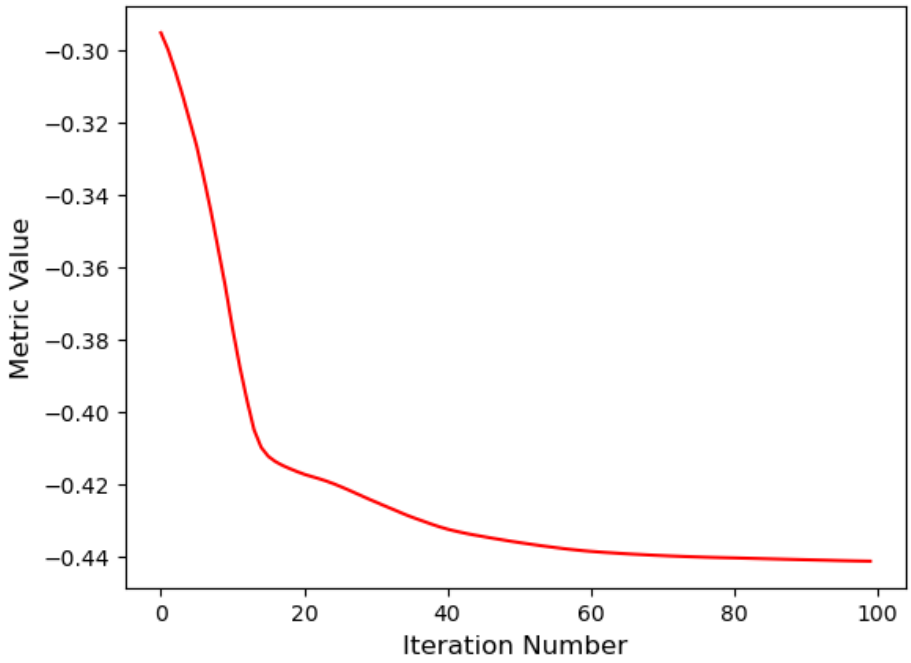


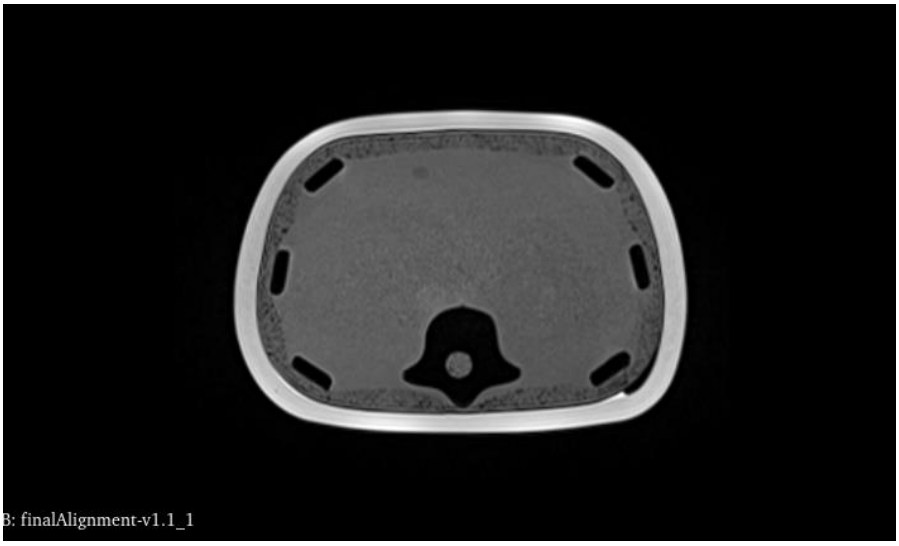
No primeiro script (Versão 1), a transformação inicial é definida e passada para o método SetInitialTransform. A transformação final é obtida diretamente após a execução do registro, sendo uma versão refinada da transformação inicial. Essa abordagem é relativamente simples e direta, sem composição de múltiplas transformações ou o uso de multi-resolução. O otimizador utilizado é o GradientDescent, que ajusta as etapas fisicamente, mas não há parâmetros adicionais como suavização ou mudança de escala por níveis de resolução. O processo também inclui o uso de callbacks que permitem o monitoramento do progresso do registro, como a plotagem dos valores da métrica em cada iteração. Resultado:



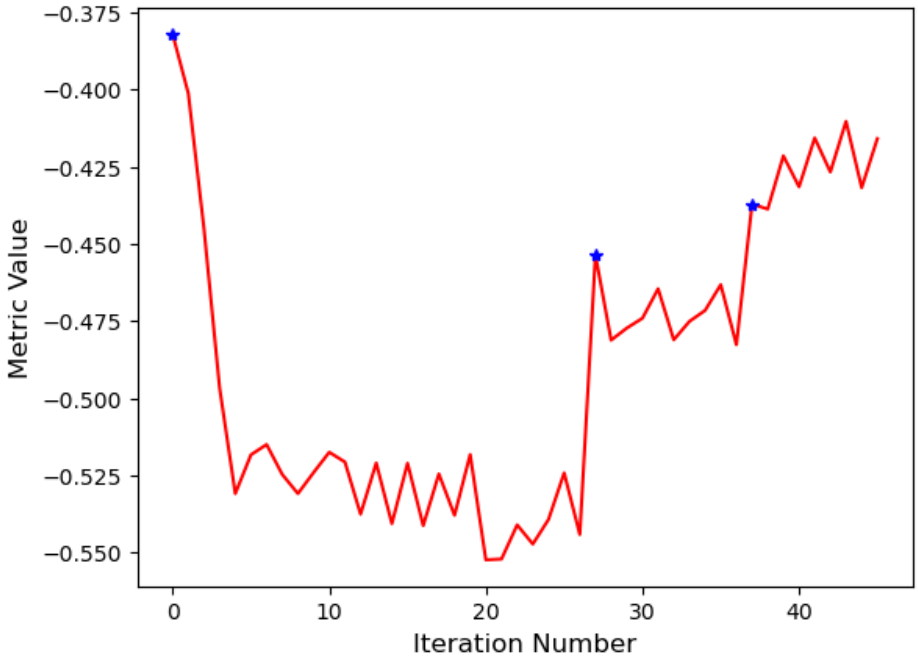


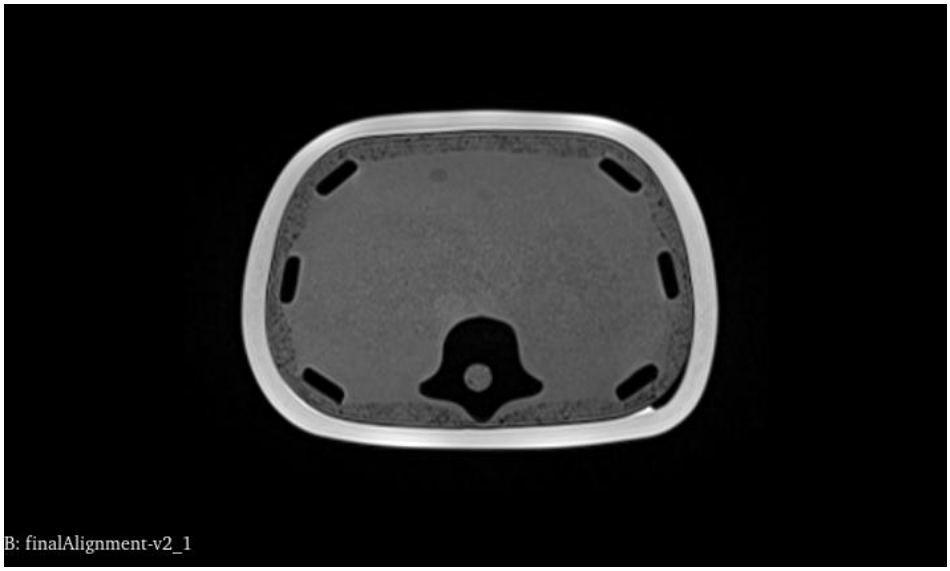
No segundo script (Versão 1.1), a diferença principal está na maneira como as transformações inicial e final são tratadas. Aqui, a transformação inicial é separada da transformação otimizada. O SetMovingInitialTransform é usado para definir a transformação inicial no espaço da imagem móvel, enquanto uma nova transformação, um Euler3DTransform, é criada para ser otimizada. Após a execução do registro, a transformação final é composta a partir de duas transformações: a otimizada e a inicial. Isso significa que a transformação resultante é a combinação dessas duas, o que é útil em situações onde se deseja preservar o histórico das transformações aplicadas. Esse método introduz uma estratégia de composição que não está presente no primeiro script. Resultado:





Já o terceiro script (Versão 2) introduz a estratégia de multi-resolução, tornando-o mais adequado para registros mais complexos que exigem maior precisão e eficiência. Nessa abordagem, a resolução da imagem é ajustada em diferentes níveis por meio de fatores de encolhimento (shrink factors) e sigmas de suavização, especificados fisicamente. Ao usar diferentes níveis de resolução, o registro pode ser realizado de forma mais eficiente, começando em uma versão mais grosseira da imagem e refinando gradualmente. Assim, a transformação final é novamente um Euler3DTransform, mas ela é inicializada com a transformação inicial, como no primeiro script. Além disso, o método aplica suavização em unidades físicas, o que garante que o processo seja mais robusto em imagens com diferentes resoluções. Embora o otimizador continue sendo o gradiente descendente, o uso de múltiplas resoluções e suavização controlada torna essa abordagem mais sofisticada. Resultado:





7) Registration_Memory_Time_Tradeoff.ipynb

Inicialmente, o script importa as bibliotecas necessárias. Em seguida, é definida a função register_images, que encapsula a configuração e execução do método de registro multi-resolução. Dentro dessa função, um objeto ImageRegistrationMethod é criado e configurado para utilizar a métrica de informação mútua de Mattes com 50 bins de histograma. A estratégia de amostragem da métrica é definida como regular, utilizando 1% dos pixels da imagem para avaliar a similaridade, o que balanceia entre desempenho e precisão. O interpolador escolhido é o linear (sitkLinear), que proporciona uma interpolação mais suave e precisa em comparação com o vizinho mais próximo. O otimizador utilizado é o de gradiente descendente, configurado com uma taxa de aprendizado de 1.0 e 1000 iterações, permitindo uma otimização mais detalhada. A transformação inicial é definida, mas não é modificada no local (inPlace=False), permitindo a execução repetida do registro sem alterar a transformação original.

O script então localiza o diretório de dados e lê as séries de imagens fixa e móvel utilizando o ImageSeriesReader do *SimpleITK*. As imagens são carregadas no formato sitkFloat32 para garantir a precisão nos cálculos subsequentes. Pontos fiduciais conhecidos são carregados a partir de um arquivo padrão (ct_T1.standard), e uma transformação de referência é estimada utilizando a orientação absoluta baseada nesses pontos. Esta transformação inicial é um objeto Euler3DTransform, que inclui uma matriz de rotação e um vetor de translação, ajustando a orientação geométrica das imagens para um alinhamento inicial.

Para avaliar a precisão do registro, um conjunto de pontos aleatórios é gerado a partir da imagem fixa, e esses pontos são transformados utilizando a transformação de referência. Isso cria um conjunto correspondente de pontos na imagem móvel, permitindo a medição dos erros de registro (TRE - Target Registration Error) antes da aplicação do registro real. O uso da função interact com *ipywidgets* permite a visualização interativa das fatias das imagens fixa e móvel, facilitando a análise visual do alinhamento.

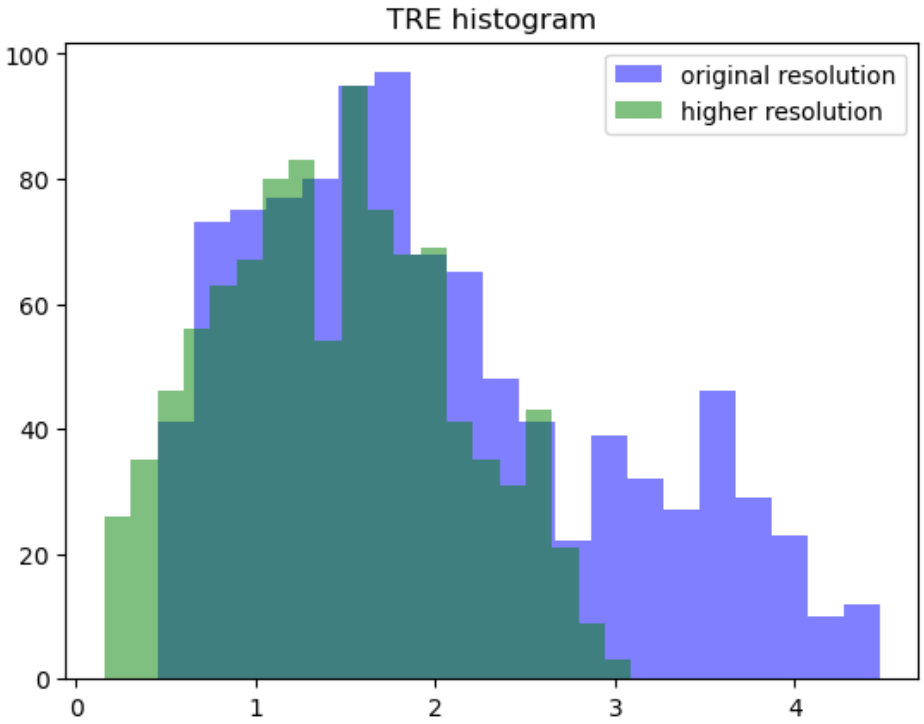
Posteriormente, o script realiza um reamostramento da imagem móvel para garantir que os voxels sejam isotrópicos com espaçamento de 1mm. Isso é feito ajustando o

espaçamento e o tamanho da imagem, e utilizando um interpolador BSpline de ordem 3 para preservar a qualidade da imagem durante o reamostramento. A relação de memória resultante é impressa, destacando a diferença entre as resoluções original e reamostrada.

Com as imagens preparadas, a transformação inicial é reestabelecida utilizando o CenteredTransformInitializer, que centraliza a transformação baseada na geometria das imagens. O processo de registro é então temporizado usando a magia %%timeit do Jupyter Notebook, permitindo medir o tempo de execução da função register_images. Após o registro, os erros de registro são calculados utilizando a função registration_errors, que compara os pontos fixos e móveis transformados, fornecendo estatísticas como o erro médio, desvio padrão e erro máximo.

Em seguida, o script repete o processo de registro, mas desta vez utilizando a imagem móvel reamostrada e um interpolador de vizinho mais próximo (sitkNearestNeighbor). Novamente, os erros de registro são calculados e impressos, permitindo uma comparação direta entre os resultados obtidos com diferentes interpoladores e resoluções de imagem.

Por fim, o script plota histogramas dos erros TRE para ambas as resoluções (original e reamostrada), utilizando o *matplotlib*. Esses histogramas visualizam a distribuição dos erros, facilitando a comparação e a avaliação da eficácia das diferentes abordagens de registro.



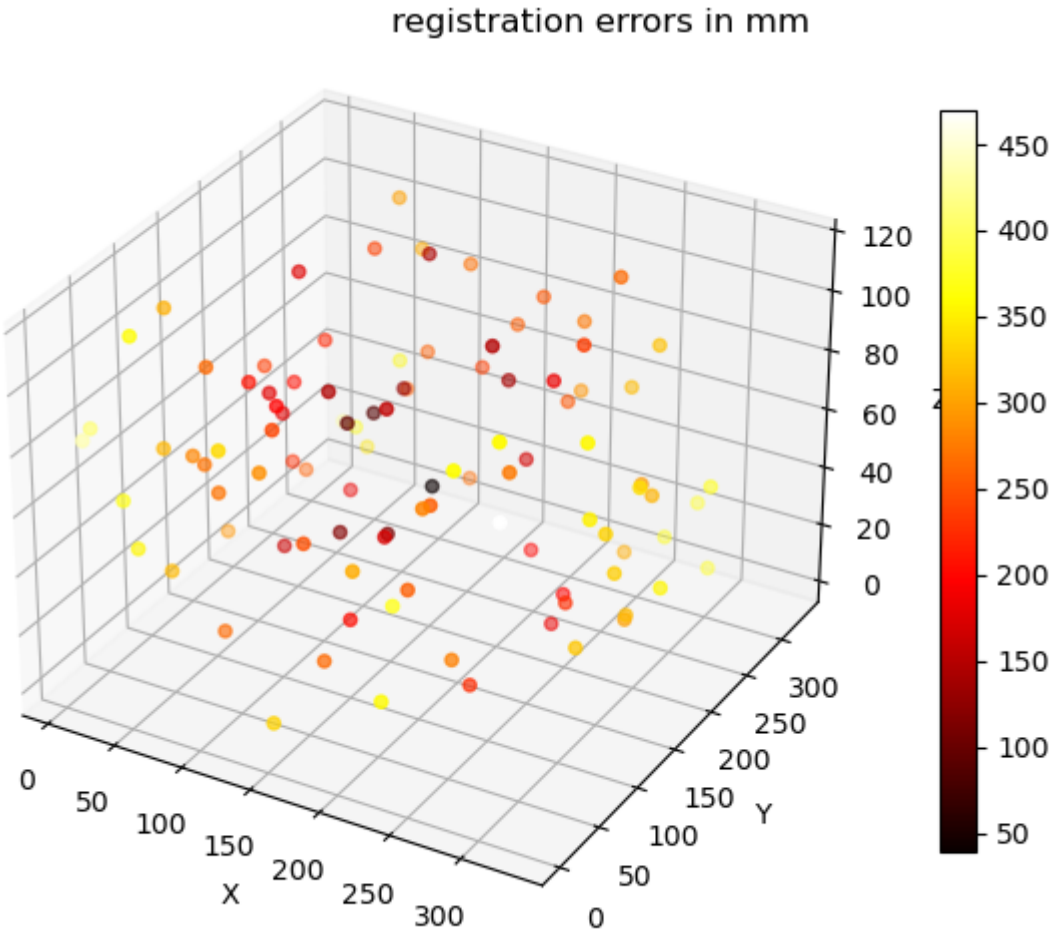
8) Registration_Semiautomatic_Homework.ipynb

Inicialmente, o script lê as imagens fixa e móvel a partir de arquivos especificados. Em seguida, carrega os pontos fiduciais correspondentes das duas imagens utilizando a função load_RIRE_ground_truth. O script considera a possibilidade de as imagens terem orientações diferentes. Se a variável rotate estiver definida como True, a imagem móvel é rotacionada em 180 graus ao redor do eixo z para alinhar sua orientação com a imagem fixa. Essa rotação é realizada utilizando uma transformação

Euler3DTransform, que inclui uma rotação de pi radianos (180 graus). Após a rotação, a imagem móvel é reamostrada utilizando interpolação linear para garantir que a transformação seja aplicada corretamente.

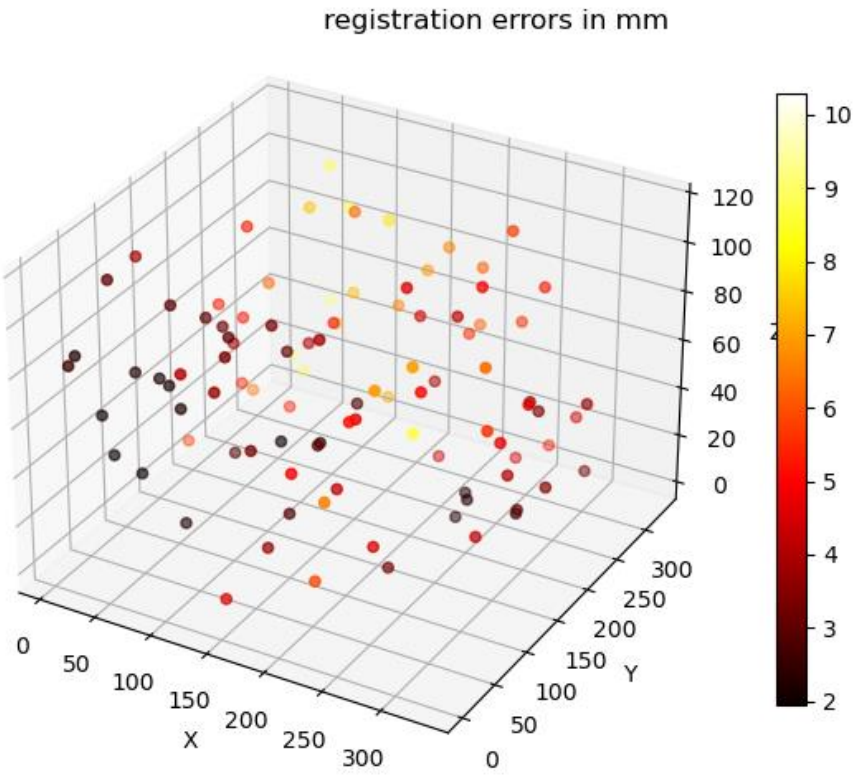
Em seguida, o script inicializa uma transformação rígida (VersorRigid3DTransform) baseada nos pontos fiduciais carregados. Isso é feito utilizando a função LandmarkBasedTransformInitializer, que calcula a melhor transformação rígida que alinha os conjuntos de pontos fiduciais das imagens fixa e móvel. A transformação resultante inclui uma matriz de rotação e um vetor de translação, ajustando a orientação geométrica das imagens para um alinhamento inicial.

Para avaliar a precisão do registro, o script gera um conjunto aleatório de pontos a partir da imagem fixa utilizando a função generate_random_pointset. Esses pontos são então transformados utilizando a transformação de referência, criando um conjunto correspondente de pontos na imagem móvel. Antes de realizar qualquer registro real, os erros de registro (TRE - Target Registration Error) são calculados utilizando uma transformação identidade (Euler3DTransform), fornecendo uma medida inicial de quão bem as imagens estão alinhadas antes da otimização. O resultado em valor foi de 268mm de erro em média.

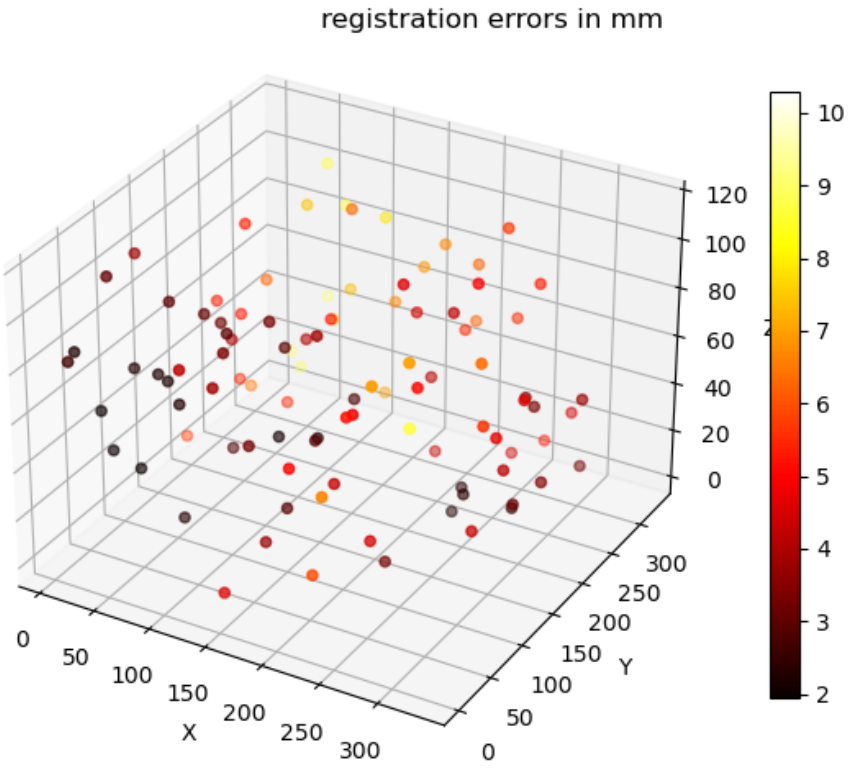


O script então utiliza pontos específicos selecionados nas imagens fixa e móvel para refinar a transformação inicial. Pontos de referência manualmente especificados são utilizados para inicializar uma transformação rígida baseada em marcos (VersorRigid3DTransform), que é posteriormente convertida para uma transformação Euler3DTransform para facilitar a otimização.

Após a aplicação da transformação, os erros de registro são recalculados, permitindo a avaliação da melhoria no alinhamento das imagens. O script também oferece uma etapa de localização semiautomática, onde pontos de referência atualizados são utilizados para inicializar uma transformação rígida adicional. Novamente, os erros de registro são calculados para avaliar a eficácia dessa abordagem. Resultado com **manual landmark localization:**

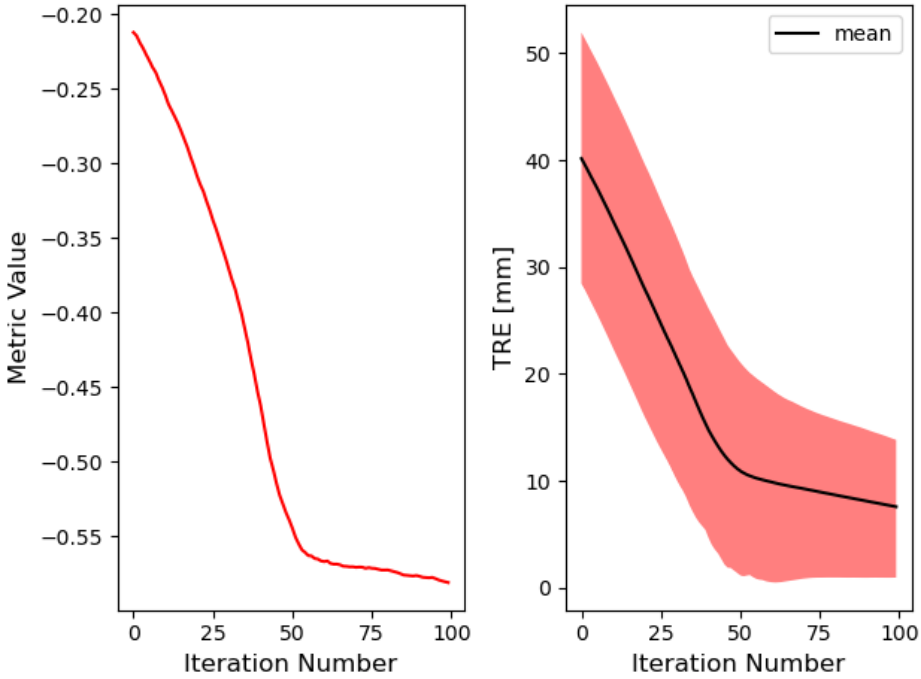


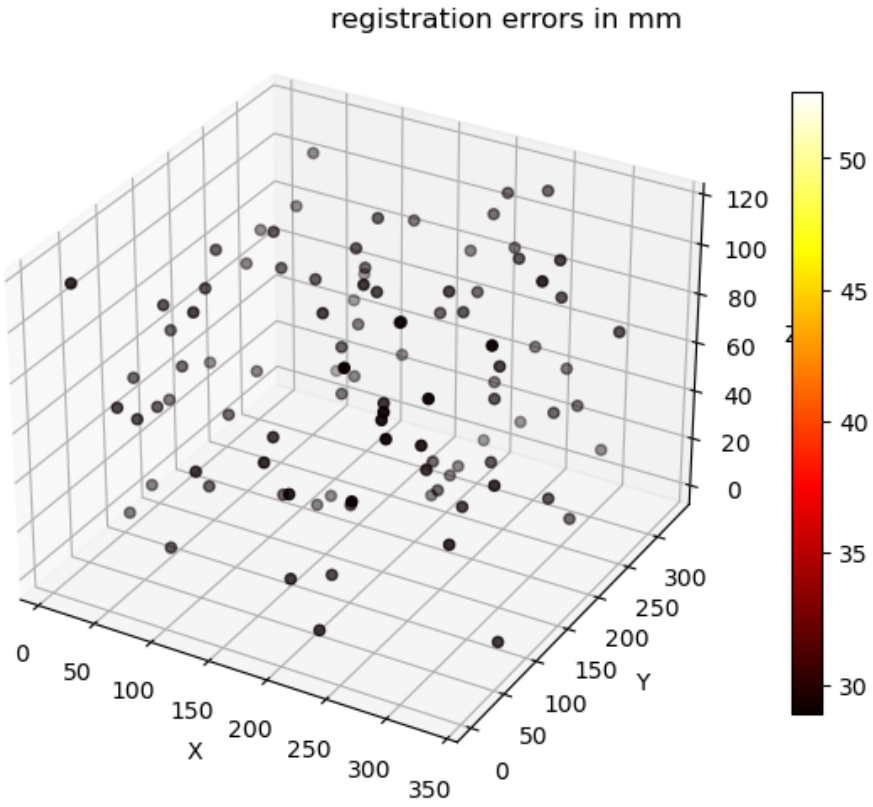
Resultado com semiautomatic landmark localization:



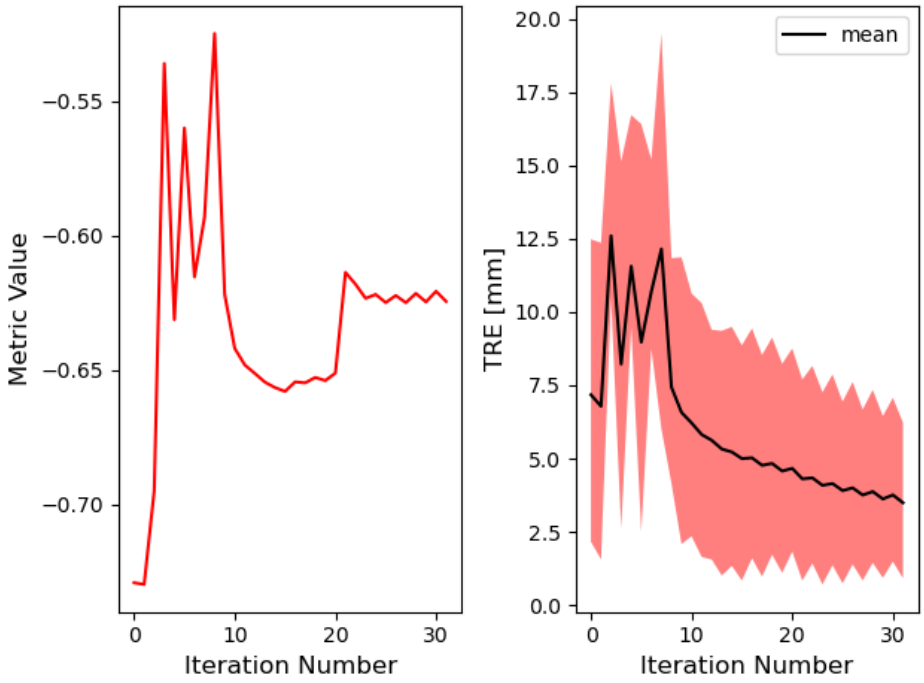
9) Registration_Tuning.ipynb

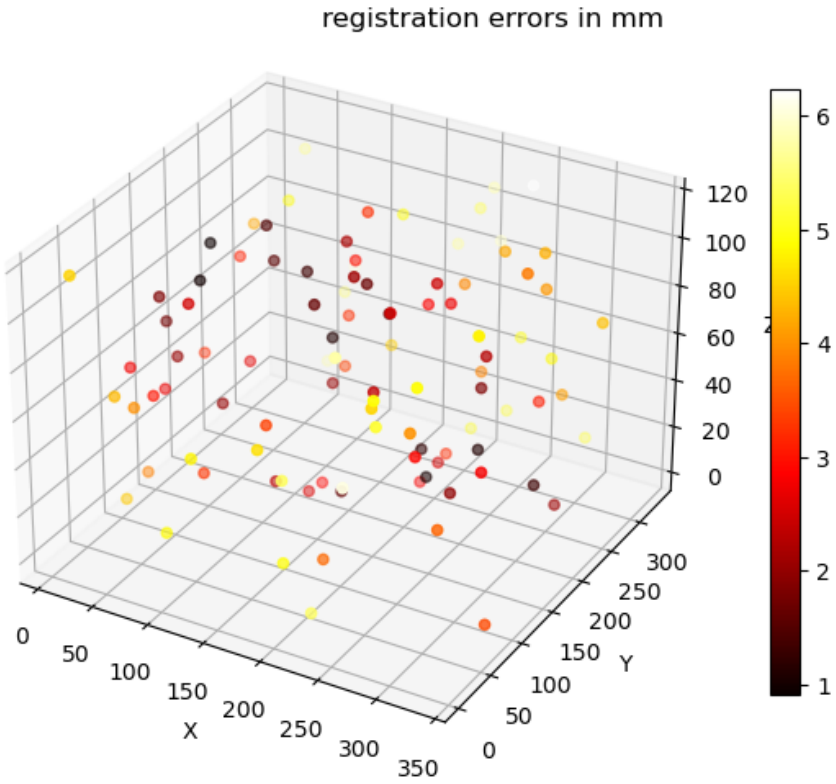
Há 3 modelos de registro apresentados e foi utilizado uma métrica de cálculo de erro dado a presença de marcadores fiduciais. Logo, o primeiro script implementa um registro simples baseado em informação mútua (Mattes Mutual Information), utilizando apenas 1% dos pixels das imagens com uma estratégia de amostragem aleatória. Ele faz uso do interpolador de vizinho mais próximo, o que torna o processo mais rápido, mas menos preciso em relação à interpolação linear, que é mencionada como uma alternativa. O otimizador utilizado é o de gradiente descendente, com 100 iterações. A transformação inicial não é alterada durante o processo de otimização, o que permite que o script seja executado várias vezes sem modificar a transformação original. Esse primeiro script é focado na simplicidade, sem o uso de técnicas mais avançadas como multi-resolução ou suavização. O erro médio obtido foi de 7,58mm.



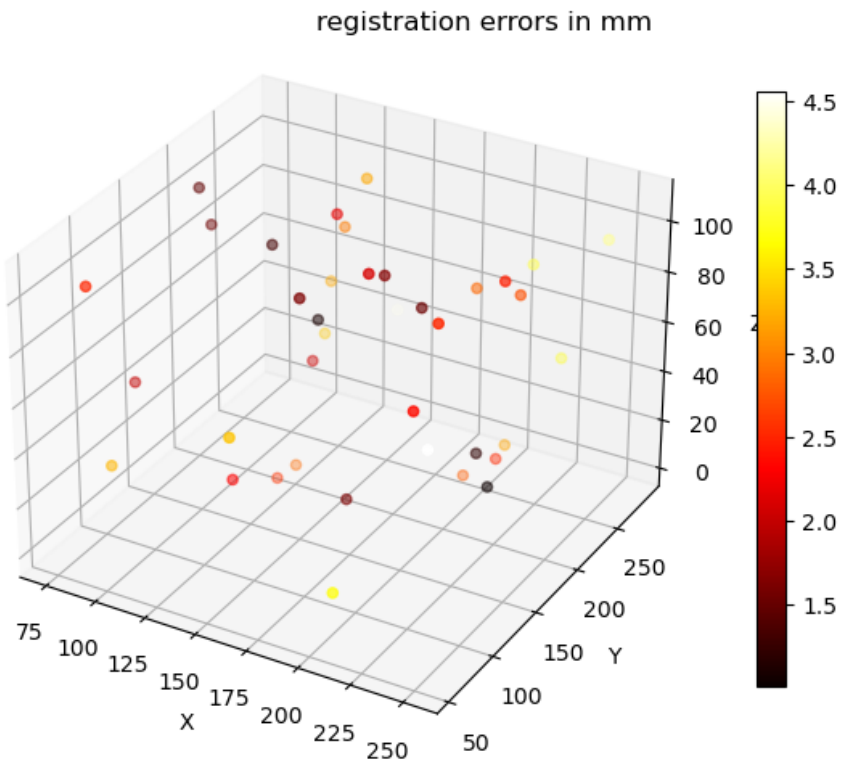


O segundo script, por sua vez, expande o processo de registro ao introduzir uma estratégia de multi-resolução e um interpolador linear. Ao contrário do primeiro, ele utiliza 10% dos pixels das imagens, o que melhora a precisão da métrica de similaridade, embora torne o processo mais demorado. O interpolador linear proporciona resultados mais suaves e precisos em comparação ao vizinho mais próximo. Além disso, este script introduz a técnica de multi-resolução, onde a imagem é registrada em diferentes níveis de resolução, utilizando fatores de encolhimento e suavização progressiva. Isso melhora significativamente a eficiência do processo, permitindo um registro mais preciso e eficiente, especialmente útil para imagens complexas. O uso de múltiplos níveis de suavização também garante que o registro se adapte melhor a diferentes escalas da imagem. O erro médio foi de 3,49mm.





O terceiro script adota uma abordagem mais específica, focando na região de interesse (ROI) da imagem fixa. Inicialmente, ele segmenta a imagem fixa com base em um limiar para criar uma máscara binária que isola a área de interesse, como a cabeça, no contexto de um exame médico. Em seguida, a análise da forma do rótulo é realizada para identificar a caixa delimitadora ao redor da região de interesse, o que permite restringir o registro apenas aos pontos dentro dessa área específica. Ao final, o erro de registro é calculado apenas para os pontos localizados dentro da caixa delimitadora. Essa abordagem é útil quando o objetivo é focar o registro em uma região específica da imagem, ignorando áreas que podem não ser relevantes ou que podem prejudicar a precisão do registro. O erro médio foi de 2,58.



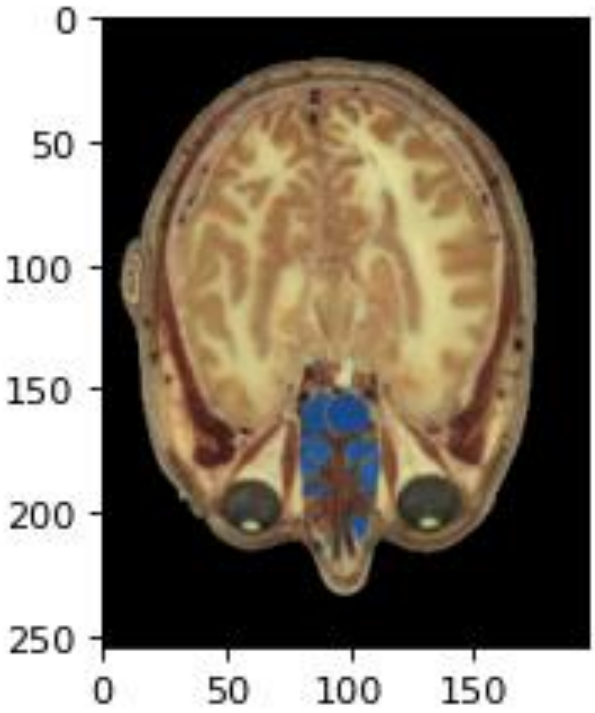
10) VH_Registration1.ipynb

O objetivo principal é alinhar duas imagens provenientes de diferentes modalidades ou capturadas em momentos distintos, garantindo que estruturas anatômicas semelhantes estejam corretamente sobrepostas. A seguir, descrevo detalhadamente cada etapa do script e suas funcionalidades.

Em seguida, o script lê uma imagem RGB fixa (fixed_rgb). A imagem é recortada para uma região de interesse específica (fixed_rgb = fixed_rgb[735:1330, 204:975, :]), reduzindo o tamanho da imagem para focar em uma área particular. Após o recorte, a imagem é encolhida utilizando a função sitk.BinShrink, que reduz a resolução da imagem em fatores especificados [3, 3, 1], mantendo a dimensão de profundidade inalterada.

Para segmentar regiões de interesse na imagem fixa, o script utiliza a função sitk.VectorConfidenceConnected, que implementa uma técnica de crescimento de região baseada em confiança. Inicialmente, são definidos pontos sementes (seeds = [[10, 10, 10]]), a partir dos quais o algoritmo inicia o crescimento das regiões segmentadas. O parâmetro initialNeighborhoodRadiusc define o raio inicial para a exploração, e numberOfIterations especifica quantas iterações o algoritmo deve realizar. O parâmetro multiplier ajusta a sensibilidade do crescimento da região. Após a segmentação, a máscara resultante (fixed_mask) é processada para inverter a segmentação e selecionar apenas o maior componente conectado, garantindo que apenas a região de interesse principal seja considerada.

A máscara segmentada é então aplicada à imagem fixa utilizando a função sitk.Mask, e a imagem resultante é exibida novamente com a sobreposição da máscara, facilitando a visualização das regiões segmentadas.



A transformação inicial para o registro é definida utilizando sitk.CenteredTransformInitializer, que estima uma transformação rígida (Euler3DTransform) baseada nos momentos geométricos das imagens fixa e móvel. Este passo centraliza a transformação, facilitando o alinhamento inicial das imagens.

O método de registro (R) é então configurado utilizando sitk.ImageRegistrationMethod. A métrica escolhida é a Informação Mútua de Mattes (MattesMutualInformation) com 50 bins de histograma, adequada para registros multimodais. O otimizador utilizado é o Gradiente Descendente com Busca de Linha (GradientDescentLineSearch), configurado com uma taxa de aprendizado de 1.0 e um número máximo de 100 iterações. A função SetOptimizerScalesFromIndexShift ajusta as escalas do otimizador com base nos deslocamentos de índice, facilitando a convergência do otimizador.

O registro é configurado para operar em múltiplas resoluções (SetShrinkFactorsPerLevel([4, 2, 1])) com sigmas de suavização progressiva (SetSmoothingSigmasPerLevel([8, 4, 2])), o que permite uma abordagem hierárquica onde o registro é refinado em diferentes níveis de detalhe. A interpolação linear (sitk.sitkLinear) é utilizada para interpolar os valores de pixel durante o registro.

Para eliminar variabilidade associada à amostragem aleatória da métrica, a amostragem é configurada com uma porcentagem fixa (SetMetricSamplingPercentage(percentage=0.1, seed=42)), garantindo reprodutibilidade dos resultados.

A transformação inicial definida anteriormente (tx = initialTransform) é atribuída ao método de registro, e comandos de callback são adicionados para monitorar o progresso do registro através da função command_iteration. O registro é então executado entre as imagens fixa e móvel, resultando em uma transformação final (outTx) que alinha a imagem móvel à imagem fixa.

Em seguida, a transformação inicial é combinada com uma transformação afim (AffineTransform) para criar uma transformação composta (sitk.CompositeTransform), que é utilizada para um segundo estágio de registro com fatores de encolhimento e sigmas de suavização ajustados ([2, 1] e [4, 1], respectivamente). O registro é novamente executado com esta transformação composta, refinando ainda mais o alinhamento entre as imagens.

Para visualizar os resultados do registro, o script utiliza o sitk.ResampleImageFilter para aplicar a transformação final à imagem móvel, reamostrando-a para coincidir com a imagem fixa. A interpolação BSpline de ordem 3 (sitk.sitkBSpline3) é utilizada para preservar a qualidade da imagem durante o reamostramento.

Após o reamostramento, a imagem resultante (out) é convertida para RGB (out_rgb) e uma visualização de checkerboard é criada utilizando sitk.CheckerBoard, que combina a imagem fixa e a imagem transformada em padrões alternados, facilitando a inspeção visual do alinhamento.