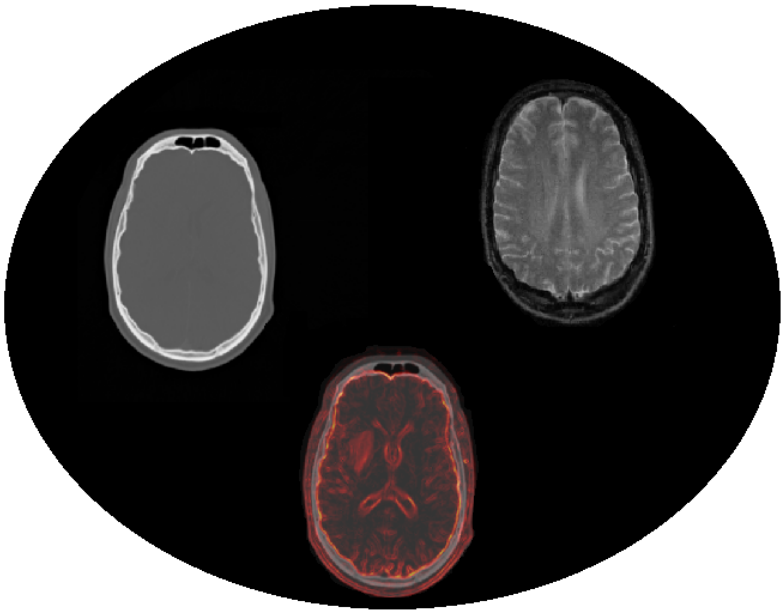
Fortgeschrittenenpraktikum (FP-95)

Medizinische Bildanalyse



Version 1.1
(7. Juni 2018)

Arbeitsgruppe
Experimentelle Strahlentherapie
Theodor-Kutzer-Ufer 1-3, 68167 Mannheim, Haus 3, Ebene 4

Anleitung und Versuch werden kontinuierlich überarbeitet. Für eine Verbesserung des Versuchs teilen Sie Kommentare und Anmerkungen bitte den Betreuern mit!

Inhaltsverzeichnis

1 Einleitung

Medizinische Bildregistrierung ist ein wesentlicher Bestandteil in der Planung von medizinischen Eingriffen oder der Diagnose von Krankheiten. Ziel der Bildregistrierung ist die geometrische Verknüpfen von Bildinformationen aus verschiedenen Aufnahmen. Patientenbewegungen, unterschiedliche interne Koordinatensysteme der bildgebenden Systeme, unterschiedliche Aufnahmeperspektiven oder Verzerrungen in den Bildaufnahmen können zum Beispiel zu unterschiedlichen geometrischen Darstellungen des aufgenommenen Objektes führen. In diesen Fällen ist es notwendig die Transformation zwischen den Bilddaten zu ermitteln, sodass korrespondierende Bildinformationen einander zugeordnet werden können. Insbesondere die Verknüpfung von Informationen verschiedener Aufnahmemodalitäten wie CT und MRT Daten spielt hierbei eine große Rolle. In diesem Praktikum sollen die Grundlagen der Bildregistrierung erarbeitet und an verschiedenen Beispielen selbst getestet werden. Aufgrund des Umfangs dieses Themas beschränkt sich der Versuch auf einige Aspekte der Registrierung anhand derer die Konzepte angewendet und diskutiert werden können.

Der Versuch kann entweder an vier Nachmittagen oder in zwei Tagen Vollzeit durchgeführt werden. Es werden 16 Stunden für die Bearbeitung vorgesehen, was bedeutet, dass die Versuchsteilnehmer sich im Vorfeld gut vorbereiten sollten, um die Aufgaben in diesem Zeitrahmen durchzuführen. Im Folgenden wird von einer Bearbeitung in vier Nachmittagen ausgegangen. Falls der Versuch in zwei Tagen abgeschlossen werden soll, so sind der erste und dritte Tag als Vormittagseinheiten zu sehen, der zweite und vierte Tag als Nachmittagseinheiten.

Noch ein wichtiger Hinweis bevor es losgeht: Da der Versuch noch in der Testphase ist sind wir dankbar für Feedback bzgl. Problemen bei der Durchführung der einzelnen Versuchsteile sowie bzgl. der Verständlichkeit dieser Versuchsanleitung.

Bei Fragen stehen Ihnen die Betreuer natürlich gerne zur Seite, jedoch soll der Versuch im Wesentlichen in selbstständiger Arbeitsweise durchgeführt werden. Die Unterstützung durch die Tutoren kann und soll daher nicht eine Einarbeitung in das Thema ersetzten.

2 Grundlagen

Dieser Abschnitt enthält eine Einführung in die Grundlagen der medizinischen Bildgebung sowie der Bildregistrierung. Hinweise zu Implementierungen finden sich in der Versuchsdurchführung und den separaten Kapiteln 7 und 8. Zur Vorbereitung auf den Versuch lohnt es sich auch den Fragenkatalog am Ende der Anleitung durchzugehen.

2.1 Bildmodalitäten

Betrachten wir zunächst die Grundlagen der medizinischen Bildgebung. Zwei entscheidende Bildgebungstechniken im klinischen Alltag sind die Magnetresonanztomographie (MRT) und die Computertomographie (CT). Da man mit den verschiedenen Bildgebungstechniken unterschiedliche Informationen hervorheben kann, ist die Wahl des Aufnahmesystems abhängig von der Fragestellung. Um das zu verstehen betrachten wir zunächst die Physik der MRT-Bildgebung.

2.1.1 MRT-Bildgebung

Entscheidend für die Magnetresonanztomographie sind die Kerne von Wasserstoffatomen, die Protonen und ihre Eigenschaft des Spins. Als rotierende Masse verfügt es über einen Drehimpuls und als Ladungsträger über ein magnetisches Moment. Somit hat es sowohl Eigenschaften eines Kreisels als auch eines Magneten. Wie bei einem Kreisel versucht das Proton die Lage seiner Rotationsachse beizubehalten und ist dabei wie ein Magnet durch andere Magnetfelder beeinflussbar. Wie bei Magneten, kann daher auch durch Bewegungen des Protons in einer Empfangsspule eine Spannung induziert werden. Somit können sowohl der Magentfeldvektor, also die Lage der Rotationsachse des Protons beobachtet werden, als auch eine Änderung von dessen Ausrichtung, durch die in einer Empfangsspule eine messbare Spannung induziert wird.

In einem äußeren Magnetfeld B_0 richtet sich ein Magnet (wie zum Beispiel bei einem Kompass) entlang des Magnetfeldes aus. Da das Proton jedoch aufgrund seines Drehimpulses die Lage seiner Drehachse beibehalten will, führt es in einem äußeren Magnetfeld B_0 eine Präzessionsbewegung aus. Diese hat eine charakteristische Frequenz, die Larmorfrequenz

$$\omega_0 = \gamma_0 \cdot B_0, \tag{1}$$

die proportional zum angelegten Magnetfeld ist. Hierbei ist ω_0 die Larmorfrequenz in MHz, γ_0 ist das gyromagnetische Verhältnis, eine materialspezifische Konstante und B_0 die Stärke des Magnetfeldes in T.

In einem äußeren Magnetfeld in z-Richtung erfolgt allmählich eine Ausrichtung der Spins. Bei dieser Ausrichtung ist sowohl eine parallele als auch eine antiparallele Ausrichtung möglich, wobei die parallele Ausrichtung energetisch umso mehr bevorzugt wird je größer das angelegte Magnetfeld ist. Dadurch dass mehr Spins parallel als antiparallel ausgerichtet sind erhält man somit eine kleine messbare Längsmagnetisierung M_Z. Will man nun in dieses System Energie hinzufügen, so

geht kann dies mit einer elektromagnetischen Welle erfolgen, die die gleiche Frequenz hat wie die Präzession der Spins im Magnetfeld, also die Larmorfrequenz. Durch diese Anregung kippen die Drehachsen der Spins wieder aus der z-Richtung heraus. Mit einem präzisen Hochfrequenzpuls mit bestimmter Leistung und Dauer können so die Drehachsen gezielt um einen bestimmten Winkel wie z.B. 90° bzw. 180° gekippt werden. Bei einem 90°-Puls kippt die Magnetisierung M von der z-Richtung in die x-y-Ebene und dreht sich dort mit der Larmorfrequenz wodurch eine Wechselspannung mit dieser Frequenz in den Empfangsspulen induziert wird. Diese Spannung ist das MR-Signal auf dem die MRT-Bildgebung beruht.

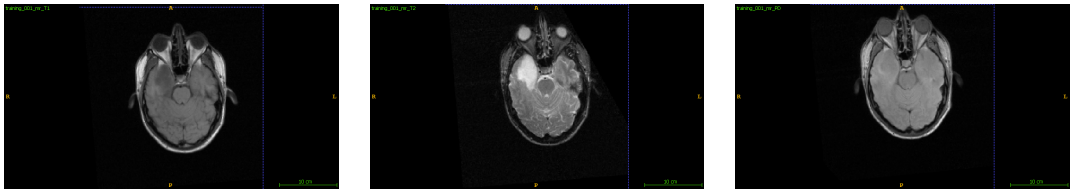
Betrachten wir nun wie die Spins aus den angeregten Zuständen wieder mit der Zeit in den stabilen Ausgangszustand zurückfallen und sich die transversale Magnetisierung in der x-y-Ebene langsam abbaut. Dies erfolgt durch die Spin-Gitter-Wechselwirkung und die Spin-Spin-Wechselwirkung. Diese Prozesse werden auch als T1- bzw. T2-Relaxation bezeichnet.

Durch die Abgabe von Energie an die Umgebung richten sich die Spins mit der Zeit wieder entlang des Magnetfeldes aus wodurch die transversale Magnetisierung abnimmt und die longitudinale Magnetisierung M_Z wieder zunimmt. Daher wird dieser Vorgang als Spin-Gitter- oder longitudinale-Relaxation bezeichnet. Die Zeitkonstante T1 für diesen Vorgang ist abhängig von der Magnetfeldstärke und von der materialabhängigen inneren Bewegung der Moleküle. Für Gewebe liegt sie bei einem angelegten Magnetfeld der Stärke 1,5 T in der Größenordnung von einer halben bis mehreren Sekunden.

Neben der Energieabgabe an die Umgebung zerstört jedoch auch noch ein zweiter Effekt die transversale Magnetisierung. Direkt nach der Anregung präzedieren alle Spins synchron haben also die gleiche Phase. Durch Energieaustausch von Spins untereinander verursacht von rasch wechselnden lokale Magnetfeldänderungen durch benachbarte Spins zerfällt die Phasenkohärenz. Dadurch heben sich die Magnetvektoren teilweise auf anstatt sich zu addieren und die transversale Magnetisierung nimmt ab. Man spricht hierbei von der Spin-Spin-Wechselwirkung. Sie hat die Zeitkonstante T2. Neben der Beeinflussung durch andere Spins führen auch Inhomogenitäten des äußeren Magnetfeldes zu einer Dephasierung der Spins. Verursacht werden diese Inhomogenitäten vom Gerät selbst sowie vom untersuchten Objekt im Magnetfeld, also z.B. einem menschlichen Körper. Diese zusätzliche Dephasierung der Spins hat die Zeitkonstante T2* , die in der Regel kürzer ist als die Spin-Spin-Relaxationszeit T2. Der T2* Effekt in den Messungen kann durch spezielle Sequenzen unterdrückt werden. Die Spin-Gitter-Wechselwirkung und die Spin-Spin-Wechselwirkung laufen gleichzeitig und unabhängig voneinander ab. Da jedoch die T2-Zeit jedoch viel kürzer ist als die T1-Zeit verschwindet die messbare transversale Magnetisierung schon lange bevor sich die Längsmagnetisierung wieder aufgebaut hat.

Drei Gewebeparametern bestimmen daher die Intensität, mit der das Gewebe in einem MRT-Bild dargestellt wird. Zum einen die Protonendichte, also der Anzahl anregbarer Spins pro Volumeneinheit. Dann die T1-Zeit, also die Zeit bis sich die Spins nach einer Anregung wieder entlang des Magnetfeldes ausgerichtet haben und zuletzt die T2-Zeit, also die Zeit bis das Signal nach einer Anregung abklingt aufgrund der aus der Phase laufenden Spins. Je nachdem welche Eigen-

schaft in einem Bild hervorgehoben wird spricht man von protonengewichteten, T1-gewichteten bzw. T2-gewichteten MR-Bildern. Beispiele hierfür sind in Abbildung 1 dargestellt.



(a) (b) (c)

Abbildung 1: Ansicht verschiedener Modalitäten: (a) MR-T1, (b) MR-T2, (c) MR-PD

Im Rahmen der Ortsauflösung sind mehrere Anregungen der selben Schicht notwendig. Die Zeit zwischen diesen Anregungen wird Repetitionszeit (TR) genannt. Je länger man zwischen zwei Anregungen wartet umso mehr Spins richten sich wieder entlang des Magnetfeldes aus und können bei der nächsten Anregung wieder zum Signal beitragen. Eine kurzes Repetitionszeit bedeutet, dass je nach Gewebe die Spins unterschiedlich stark relaxieren konnten bevor es zu einer neuen Anregung kommt. Dadurch liefern unterschiedliche Stoffe unterschiedliche Intensitäten abhängig von ihrer T1-Zeit. Wird TR jedoch sehr lang gewählt, haben die Spins in allen Geweben die Zeit sich wieder abzuregen wodurch der Bildkontrast nicht mehr von den T1-Zeiten der Gewebe abhängig ist. Eine weitere Zeit, die den späteren Bildkontrast bestimmt ist die Echozeit (TE), die Zeit zwischen der Anregung der Spins und deren Messung. Das Ein- und Ausschalten der Gradientenspulen im Rahmen der Ortskodierung erzeugt Inhomogenitäten im Magnetfeld, die die T2- und T2*-Effekte verstärken. Bei einer sehr kurzen Echozeit verglichen mit den T2-Zeiten des untersuchten Gewebes (TE $\ll$ T2) sind die T2- und T2*-Effekte noch gering und kaum messbar. Erst wenn die Echo-Zeit in der Größenordnung der vorkommenden T2-Zeiten gewählt wird sind die Signalunterschiede zwischen den verschiedenen Geweben signifikant und man erhält eine starke T2-Gewichtung. Durch die Zeiten TR und TE kann also die Gewichtung des Bildes bestimmt werden. Der Zusammenhang ist in Tabelle 1 zusammengefasst.

	Repetitionszeit TR	Echozeit TE
T1-gewichtet	kurz	kurz
T2-gewichtet	lang	lang
Protonengewichtet	lang	kurz

Tabelle 1: Bildgewichtung durch die Wahl von TR und TE.

Nachdem nun die Entstehung der MR-Signale und Möglichkeiten der unterschiedlichen Gewichtung der Bildgebung besprochen wurden, betrachten wir im letzten Schritt noch die Ortskodierung. Um ein dreidimensionales Bild aufnehmen zu können müssen den MR-Signalen Koordinaten zugeordnet werden, es muss also klar sein von wo im Körper sie ausgegangen sind. Für die Auflösung der z-Koordinate verwendet man eine zusätzliche Magnetspule, die dem Magnetfeld

einen Gradienten entlang der z-Richtung gibt. Da die Larmorfrequenz abhängig ist von der Stärke des äußeren Magnetfeldes unterscheiden sich die Larmorfrequenzen der Spins nun entlang der z-Richtung. Somit wird immer nur eine Schicht im Körper mit einer Frequenz angeregt. Durch einen starken Gradienten kann man daher feine Schichten auflösen, wohingegen man bei einem schwachen Gradienten nur grobe Schichten auflösen kann. Für die Auflösung in Y-Richtung, die Phasenkodierung, wird ein zusätzlicher Gradient in Y-Richtung angelegt. Hierdurch ist die Larmorfrequenz im oberen Bereich des MRT etwas höher als im unteren wodurch die Spins im oberen Bereich schneller kreisen als die im unteren Bereich. So kommt es zu einer Phasenverschiebung der Spins. Schaltet man den Phasengradienten kurz danach wieder ab bewegen sich alle Spins wieder gleich schnell können aber anhand ihrer Phase bzgl. ihrer Y-Position zugeordnet werden. Mit einem zusätzlichen Gradienten in X-Richtung erreicht man, dass die Spins im Bereich mit stärkerem Magnetfeld schneller präzedieren als im Bereich mit schwächerem Magnetfeld. Dadurch hat das MR-Signal nicht mehr nur eine Frequenz, sondern ein Frequenzspektrum, bei der die Höhe der Frequenz die Information über die x-Position liefert. Die Frequenzen können durch eine Fourier-Transformation analysiert werden. Zur Analyse der Phaseninformation ist jedoch eine Messung nicht ausreichend sondern es müssen mehrere Messungen mit unterschiedlichen Phasenkodierungen durchgeführt werden. Auf diese Weise kann man aus den Ergebnissen die Phaseninformationen mit Hilfe einer weiteren Fourier-Transformation dekodieren. Die Zeit zwischen diesen Messungen ist die bereits diskutierte Repetitionszeit TR.

2.1.2 CT-Bildgebung

Widmen wir uns nun mit dem Computertomographen (CT) einem anderen Bildgebungssystem. Ein CT besteht aus einer Röntgenröhre, Blenden, einem Liegetisch und einem Detektor. Für die Berechnung eines Bildes aus den Detektordaten ist zudem eine Recheneinheit nötig sowie Bedien- und Anzeigevorrichtungen.

Mit Hilfe der Blenden wird sichergestellt, dass die in der Röntgenröhre erzeugte Strahlung kontrolliert durch den zu untersuchenden Körper strahlt. Nach verlassen des Körpers trifft sie dann auf einen Detektor. Durch die Abschwächung der Strahlung im Gewebe ändert sich die Energie, die den Detektor erreicht. Durch eine Drehung von Strahlenquelle und Detektor um den Patienten erhält man eine Vielzahl von Röntgenbildern aus verschiedenen Perspektiven. Die Kontraste in den späteren Bildern sind abhängig von der Absorptionsfähigkeit der Materialien. Gemessen wird diese in der Hounsfield-Skala, die den relativen Absorptionsunterschied eines Materials verglichen mit Wasser angibt. Luft hat hierbei einen Wert von -1000HU, Knochen sind in der Größenordnung von 500-1500HU. Mit Hilfe einer Radontransformation werden aus den Detektordaten, die nur die Aufsummierung aller Absorptionen enthalten die Bilddaten rekonstruiert.

2.2 Bildregistrierung

Um Bildinformationen verschiedener Modalitäten gleichzeitig nutzen zu können müssen die Daten in ein gemeinsames Koordinatensystem überführt werden. Dies

erfolgt im Rahmen einer Bildregistrierung. Hierbei wird ein Datensatz als Referenz verwendet und dann nach einer Transformation gesucht, die den zweiten Datensatz dem ersten optimal überlagert. Zusätzlich können anhand der Referenz auch Artefakte wie Verzerrungen aus einem Datensatz herausgerechnet werden. Da für die Diagnostik meist eine Kombination aus mehreren Aufnahmemodalitäten genutzt wird ist die Bildregistrierung in medizinischen Bildverarbeitungsframeworks von großer Bedeutung. Dieser Abschnitt behandelt die Grundlagen der Bildregistrierung.

Betrachten wir zunächst ein Bild und eine Transformation, die beschreibt wie das Bild transformiert werden soll. Mit Hilfe der Transformation kann für jeden Bildpunkt direkt berechnet werden, wohin dieser im transformierten Bild abgebildet wird. Da ein Bild aufgrund seiner Pixel diskretisiert ist, wird zusätzlich ein Intepolationsverfahren benötigt um die Intensitätswerte an den Pixelpositionen im transformierten Bild zu bestimmen. Diese Art der Problemstellung wird auch Vorwärtsproblem genannt. Im Gegensatz dazu ist bei dem Registrierungsproblem nicht das Bild und die Transformation bekannt, sondern das Ursprungsbild, im Folgenden Referenzbild genannt, und ein transformiertes Bild, das im Folgenden als bewegtes Bild bezeichnet wird. Anhand dieser Bildinformationen soll die Transformation ermittelt werden, die das bewegte Bild bestmöglich in das Referenzbild überführt. Diese Art der Problemstellung wird auch als inverses Problem bezeichnet und kann im Allgemeinen nur durch ein Optimierungsverfahren gelöst werden. Dieses hat die allgemeine Form:

$$\bar{\boldsymbol{u}} = \arg\min_{\boldsymbol{u}} \left(C(\boldsymbol{u})\right), \tag{2}$$

wobei u die zu optimierenden Parameter der Bildtransformation sind, $C(\boldsymbol{u})$ die Kostenfunktion ist, die die Ähnlichkeit der Bilder definiert und $\bar{u}$ die Parameter sind, die die optimale Transformation zwischen den Bildern beschreiben.

Der Algorithmus zur Bildregistrierung besteht somit aus einer Metrik, die die Ähnlichkeit zweier Bilddatensätze definiert, einer Transformation, die festlegt wie das bewegte Bild transformiert werden darf um die Ähnlichkeit zwischen bewegtem Bild und Referenzbild zu erhöhen, einem Interpolationsverfahren und einem Optimierungsalgorithmus mit dem der Metrikwert optimiert wird. Im Folgenden fokussieren wir uns auf Ähnlichkeitsmaße und Transformationen der medizinische Bildregistrierung und ihre Einsatzmöglichkeiten.

2.2.1 Ähnlichkeitsmaße

Ähnlichkeitsmaße können zunächst in intensitätsbasierte und landmarkenbasierte Maße unterteilt werden. Bei intensitätsbasierte Maßen werden die Intensitätswerte in den Bildern für die Ermittlung der Ähnlichkeit genutzt. Für landmarkenbasierte Ansätze werden zunächst signifikante Positionen in den zu registrierenden Bildern extrahiert und für die Bestimmung der Ähnlichkeit genutzt. In diesem Praktikum fokussieren wir uns auf intensitätsbasierte Methoden, für die keine Landmarkenbestimmung notwendig ist.

Bei intensitätsbasierten Metriken unterscheidet man zwischen Metriken, die die Intensitätswerte direkt miteinander vergleichen und solchen, die nur statistische

Informationen über die Intensitäten der Bilder nutzten. Erstere lassen sich häufig nur im monomodalen Fall einsetzen, da die Annahme, dass gleiche Informationen durch gleiche Intensitäten in den Bildern repräsentiert werden im multimodalen Fall nicht garantiert ist.

Für einen direkten Vergleich der Bildintensitäten kann die Summe der Quadratischen Abweichung (Sum of Squared Differences - SSD) verwendet werden. Hier werden die quadratischen Differenzen der Intensitäten zwischen dem Referenzbild und dem bewegten Bild verglichen. Um eine Mittlere Abweichung zu erhalten wird das Ergebnis häufig noch durch mit der Anzahl betrachteter Punkte N normiert, wodurch man die mittlere quadratische Abweichung (Mean Squared Difference - MSD) erhält.

$$C_{MSD} = \frac{1}{N} \sum_{i=1}^{N} \left(R(x_i) - M^T(X_i) \right)^2 \tag{3}$$

In einigen Fällen ist ein direkter Vergleich der Bildintensitäten jedoch nicht möglich. Ein Beispiel ist die Kombination von Informationen aus unterschiedlichen medizinischen Bildgebungen wie CT und MRT Aufnahmen. Hier ist zwar in beiden Datensätzen das gleiche Objekt dargestellt, jedoch sind nicht alle Informationen in beiden Datensätzen zu sehen bzw. haben unterschiedliche Intensitäten für die gleiche Information. In diesem Fall hat sich Mutual Information (MI) als Ähnlichkeitsmaß bewährt. Hier werden nur statistische Informationen über die Bildintensitäten verwendet in der Annahme, dass die Intensitäten in den beiden Bildern korreliert sind.

Mutual Information ist definiert als

$$S_{\mathrm{MI}}(I_A, I_B) = H(I_A) + H(I_B) - H(I_A, I_B), \tag{4}$$

wobei $H(I_A)$ und $H(I_B)$ die jeweils marginalen Entropien der Bilddaten I_A und I_B sind, und $H(I_A, I_B)$ die gemeinsame der Entropie beider Bilddaten ist. Die marginale Entropie ist definiert als

$$H(I) = -K \sum_{a} p_I(a) \cdot \log\left(p_I(a)\right), \tag{5}$$

wobei I das betrachtete Bildvolumen, $p_I(a)$ die Wahrscheinlichkeit der Intensität a im Bild I und K eine positive Konstante ist. Die Konstante K hängt von der gewählten Basis des Logarithmus ab beeinflusst aber nicht die spätere Optimierung da sie die Kostenfunktion nur skaliert. Die Wahrscheinlichkeit $p_I(a)$ kann durch die Erstellung eines Histogramms der Bildintensitäten bestimmt werden. Entsprechen ist die gemeinsame Entropie $H(I_A, I_B)$ definiert als

$$H(I_A, I_B) = -K \sum_{a,b} p_{I_A,I_B}(a, b) \cdot \log\left(p_{I_A,I_B}(a, b)\right), \tag{6}$$

wobei $p_{I_A,I_B}(a, b)$ die Wahrscheinlichkeit für das Intensitätspaar (a, b) in dem Bildpaar (I_A, I_B) ist, welche aus einem gemeinsamen Histogramm der Bildvolumen bestimmt werden kann.

2.2.2 Transformationen

Es gibt zwei Arten von Transformationen, die in der Bildregistrierung betrachtet werden können. Das sind zum einen Transformationen der Bildintensitäten und zum anderen geometrische Transformationen. In diesem Praktikum beschränken wir uns auf geometrische Transformationen.

Bei den geometrischen Transformationen ist die grundlegende Unterscheidung zwischen einer globalen Transformation der Bilddaten, also einer Transformation, die für alle Bildpunkte gleich ist, und Transformationen, die für jeden Punkt im Bild eine eigene Transformation vorschreiben. Bei den globalen Transformationen ist das einfachste Modell eine Translation. In diesem Fall werden alle Punkte eines Bildes um einen konstanten Vektor verschoben. Für die transformierten Koordinaten (x', y', z') eines Punktes (x, y, z) folgt somit im Falle einer Translation mit dem Vektor (t_x, t_y, t_z) in drei Dimensionen:

$$x' = x + t_x \tag{7}$$
$$y' = y + t_y \tag{8}$$
$$z' = z + t_z \tag{9}$$

Wenn neben einer Translation auch eine Rotation zugelassen ist gibt es 6 Freiheitsgrade für die Transformation (3 Translations- und 3 Rotationsfreiheitsgrade). Diese Art der Transformation wird auch als rigide bezeichnet. Eine wesentliche Eigenschaft der rigiden Transformation ist, dass sich der Abstand zwischen zwei Punkten im Bild durch die Transformation nicht ändern. Will man zudem auch Skalierungen und Scherungen der Bilder zulassen erhöht sich die Zahl der zu optimierenden Parameter auf 12. Diese Transformation wird als affine Abbildung bezeichnet. Schreibt man die 3D-Koordinaten als 4D-Vektoren können diese Parameter in einer 4×4-Matrix dargestellt werden:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & p_{16} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{10}$$

Deutlich mehr Freiheitsgrade erhält man, wenn man die Transformation eines Bildes durch ein Transformationsfeld anstatt einer einzelnen Matrix beschreibt. Hierbei spricht man von einer deformierbaren Registrierung. In diesem Fall bestimmt die Position eines Voxels im Bildvolumen welche Transformation auf den Voxel angewendet wird. Theoretisch könnte man für jeden Voxel eigene Transformationsparameter definieren dies führt jedoch schnell zu tausenden oder millionen Parametern. Da die Transformation zwischen benachbarten Punkten kontinuierlich sein soll reicht es in der Regel aus ein deutlich gröberes Gitter an Punkten zu definieren auf der die Transformation berechnet wird und dann die Transformationsparameter für dazwischenliegende Punkte zu interpolieren. Auf diese Weise kann man die Anzahl der zu optimierenden Parameter deutlich reduzieren. Eine weitere Möglichkeit die deformierbare Registrierung zu beschleunigen ist zunächst mit einem groben Gitter mit wenigen Punkten die Transformationsparameter zu

ermitteln und diese dann als Startpunkt für die Optimierung auf einem feineren Gitter zu benutzten.

3 Versuchsteil 1: Ermittlung der Kamerabewegung aus einer Bildfolge

Zum Einstieg soll eine 2D monomodale Registrierung bei der nur Translationen erlaubt sind betrachtet werden. Mit dieser Ausgangssituation wissen wir, dass die Transformation durch zwei Parameter beschrieben werden kann, die Translationsparameter in x- und y-Richtung. Für die Aufnahme der Bilder benutzen wir eine gewöhnliche USB-Kamera, die direkt an den Computer angeschlossen wird. Die Translation der Bilder erreichen wir hierbei nicht durch eine Bewegung des aufgenommen Objekts, sondern durch eine Bewegung der Kamera auf einer Schiene.

Die Aufgabe untergliedert sich in drei Teile. Im ersten Teil muss zunächst die Kamerafunktionalität in der Anwendung implementiert und getestet werden, sodass einzelne 2D-Bilder aufgenommen werden können. Hierbei wird gleichzeitig auch der Versuchsaufbau geprüft. Im zweiten Teil wird dann eine Bildserie aufgenommen, wobei die Kamera auf der Schiene zwischen den Bildern verschoben wird. Im letzten Teil muss dann der Algorithmus für das Registrierungsverfahren implementiert und paarweise auf die Bilder aus der aufgenommenen Bildserie angewendet werden.

3.1 Aufbau

Für diesen Versuch steht ein Computer mit einer Matlab-Installation, eine USB-Kamera und eine Schiene zur Kameraführung zur Verfügung. Es ist zunächst erforderlich sich mit der Kamera und deren Anbindung an Matlab vertraut zu machen. Hierfür verbinden sie die Kamera mit dem Computer und prüfen, dass die Kamera fest auf dem beweglichen Slider auf der Schiene angebracht ist.

Der bewegliche Slider kann z.B. mit einem Seil gleichmäßig auf der Schiene bewegt werden. Testen sie die Kameraführung indem sie sich die Bilder als Video anzeigen lassen und stellen sie sicher, dass sie die Kamera kontrolliert auf der Schiene bewegen können ohne diese auf dem Slider zu verstellen. Ein Wackeln der Kamera würde die Modell-Annahme einer reinen Translation zwischen den Bildern in der Serie in Frage stellen.

Skizzieren Sie des Versuchsaufbaus in ihrem Protokoll!

3.2 Bildaufnahme mit der Kamera

Zunächst muss die Ansteuerung der Kamera in Matlab implementiert werden. Die hierfür notwendigen Matlab-Befehle finden Sie in Abschnitt 7. Konfigurieren Sie zunächst die Kamera und richten sie auf ein großes Objekt oder eine Wand mit Bildern aus.

Nehmen sie anschließend eine Bildserie auf, bei der die Kamera auf der Schiene zwischen den einzelnen Bildern leicht weiterbewegt wird. Messen Sie die Abstände zwischen den einzelnen Kamerapositionen und dokumentieren Sie diese in ihrem Messprotokoll. Es sollen mindestens 5 Aufnahmen durchgeführt werden. Zeigen Sie sich die einzelnen Bilder am Bildschirm an. Wenn Sie mit der Bildserie zufrieden sind speichern Sie sowohl die einzelnen Bilder als auch den Matlab-Workspace auf dem Computer.

3.3 Paarweise Registrierung der Aufnahmen

Machen sie sich mit dem Thema Bildregistrierung in Matlab vertraut. Hier gibt es bereits einige vorgefertigte Funktionen die für die Durchführung einer Bildregistrierung benötigt werden. Eine Zusammenstellung der wichtigsten Befehle, die Sie in diesem Versuch benötigen finden Sie in Abschnitt 7. Implementieren Sie einen Algorithmus zur monomodalen Registrierung von zwei 2D-Bildern unter Annahme einer reinen Translation zwischen den Bildern. Nutzen Sie den von Ihnen geschriebenen Algorithmus um jeweils paarweise eine Registrierung der einzelnen Bilder aus der Bildserie zueinander durchzuführen und dokumentieren sie die Ergebnisse. Speichern sie sowohl die resultierenden überlagerten Bilder als auch die ermittelten Transformationsparameter.

3.4 Hinweise

- Überlegen sie bei der paarweisen Registrierung mit welche Strategie sie die Bilder am Besten registrieren können.
- In der Online-Hilfe von Matlab finden sich sowohl Beispiele zum Thema Bildregistrierung als auch zum Thema ansteuern einer USB-Kamera. Falls Sie mit den Befehlen aus Abschnitt 7 nicht auskommen lohnt sich ein Blick auf die Beispiele.

3.5 Aufgaben

- Implementieren Sie einen Algorithmus zur Aufnahme von Einzelbildern mit der USB-Kamera in Matlab.
- Nehmen Sie 5 Bilder von verschiedenen Kamerapositionen auf (Messen und dokumentieren Sie die Translation der Kamera). Die Abstände sollten so gewählt sein, dass am Ende sowohl kleine Verschiebungen als auch große Verschiebungen zwischen Bildern zu finden sind. Wählen Sie die einzelnen Abstände also weder sehr klein noch zu groß bzw. wählen Sie unterschiedliche Abstände zwischen den Kamerapositionen. Messen und dokumentieren Sie die gewählten Abstände.
- Schreiben Sie einen Algorithmus zur paarweisen Registrierung der Bilddaten.
- Registrieren Sie die Bilder und stellen Sie die Ergebnisse in Bildern dar, bei denen die Referenzbilder mit den zugehörigen transformierten Bildern überlagert sind.

3.6 Auswertung

- Übernehmen Sie die überlagerten Bilder von Referenzdaten und transformierten Bilddaten in ihr Protokoll. Diskutieren Sie die Qualität der Registrierung für die verschiedenen Kameraverschiebungen.
- Vergleichen Sie die gemessene und durch die Registrierung ermittelte Translation. Diskutieren Sie die Ergebnisse.
- Aus den verschiedenen Messungen kann das Verhältnis zwischen der Kameraverschiebung in cm und der ermittelten Pixelverschiebung ermittelt werden. Berechnen Sie das Verhältnis und geben Sie dessen Fehler (Fehlerrechnung!) an.

4 Versuchsteil 2: Ähnlichkeit bei monomodalen und multimodalen Bilddaten

Nach dem Einstieg in das Themengebiet Bildregistrierung sollen nun verschiedene Ähnlichkeitsmaße genauer untersucht werden. Hierfür werden 3D-Bilddatensätze aus CT und MRT Aufnahmen vom Kopf genutzt. Betrachtet werden soll zunächst der Fall, dass das Referenzvolumen und das zu registrierende Volumen gleich sind. Transformiert man nun eins der Volumen sollte die ermittelte Ähnlichkeit der Bilder zueinander sinken. Für den Versuchsteil verwenden wir eine Metrik wie sie in der monomodalen Bildregistrierung eingesetzt wird und eine die für multimodale Registrierungsprobleme entwickelt wurde. Untersucht wird für beide Metriken, wie sich die Ähnlichkeit abhängig von der verwendeten Transformation ändert. Anschließend wird der Versuch wiederholt mit dem Unterschied, dass nun das CT-Volumen als Referenzbild verwendet wird und das MR-Volumen als zu registrierendes Volumen. Auch für diesen multimodalen Fall soll untersucht werden wie die Metrik-Werte sich in Abhängigkeit von der verwendeten Transformation ändern.

Für die Registrierung von medizinischen Bilddaten nutzen wir in diesem Praktikum Daten aus dem Retrospective Image Registration Evaluation Project (RIRE). Wir konzentrieren uns dabei auf die CT- und die MR-T2 Daten. Unter den vorliegenden Daten gibt es einen Trainingsdatensatz, für den die optimalen Transformationen gegeben sind. Da wir die Änderung der Ähnlichkeitswerte bei Änderung der Transformationsparameter um die optimale Transformation betrachten wollen, benötigen wir in diesem Versuchsteil die optimalen Transformationsparameter um die CT- und MR-Daten zunächst in die gewünschte Ausgangskonfiguration zu transformieren.

4.1 Aufnahme der Ähnlichkeit im monomodalen Fall

Für den monomodalen Fall betrachten wir zunächst nur das CT-Volumen. Vergleichen wir dieses mit sich selbst so sollte die Ähnlichkeit höher sein als für den Fall, das wir das Volumen mit einer verschobenen oder rotierten Version des Volumens vergleichen. Als Ähnlichkeitsmaße testen wir die Mean-Square-Metrik sowie die Mattes-Mutual-Information-Metrik aus der ITK-Bibliothek.

Der Hauptteil dieses Versuchsteils wird in C++ mit der ITK-Bibliothek implementiert. Um die Einarbeitungszeit zu verkürzen wurde bereits eine Basis implementiert auf der in diesem Versuch aufgebaut werden kann. In diesem Basiscode werden zunächst die Bildvolumen als itk::Image geladen. Darüber hinaus werden Transformationen definiert und die ein Parametervektor für den monomodalen und für den multimodalen Fall definiert. Dieser Vektor enthält drei Einträge zur Beschreibung der Rotation und drei Einträge zur Beschreibung der Translation. Da im monomodalen Fall zweimal das CT-Volumen verwendet wird sind die Volumen bereits optimal überlagert und der Parametervektor enthält nur Nullen. Darüber hinaus werden die Metriken definiert und Instanzen der Metriken erstellt. Den Metriken werden die Bildvolumen als Referenz- (Fixed-) Volumen und als bewegtes (Moving) Volumen zugeordnet. Anschließend werden die Metriken initialisiert. Mit

```
metric->GetValue()
```

kann der aktuelle Wert der Metrik *metric* abgefragt werden.

Für diese Aufgabe soll nacheinander immer einer der Translationseinträge im Parametervektor um den optimalen Wert variiert werden. Der aktuelle Wert des betrachteten Transformationsparameters sowie die Werte der beiden Metriken sollen in Vektoren zwischengespeichert werden. Die Funktionalität zum Speichern der erhaltenen Werte in eine Textdatei ist bereits in dem Beispielcode implementiert.

Anschließend sollen die erhaltenen Werte mit einem Matlab-Programm eingelesen und mit den Transformationsparameter-Werten graphisch dargestellt werden. Das einlesen erfolgt mit dem Befehl:

```
Data = importdata('output.txt',' ');
```

Da die Wertebereiche der beiden Metriken sehr unterschiedlich sind, sollten für einen gemeinsamen Plot der beiden Metrik-Werte verschiedene y-Achsen-Skalen verwendet werden. Ein Beispiel hierfür ist:

```
figure;
yyaxis left
plot(data1,data2);
yyaxis right
plot(data1,data3);
```

4.2 Aufnahme der Ähnlichkeit im multimodalen Fall

In diesem Aufgabenteil sollen die Messungen aus der vorherigen Aufgabe für den multimodalen Fall wiederholt werden. Es reicht hierbei sich auf einen Translations- und einen Rotationsparameter zu beschränken. Da das CT- und das MR-T2-Volumen nicht von Anfang an überlagert sind, enthält der Parametervektor für die optimale Transformation Einträge, die nicht Null sind. Variieren Sie anschließend die Anzahl Histogramm-Bins der Mutual Information Metrik zwischen 5 und 100 und stellen Sie in Matlab die Ergebnisse in einen gemeinsamen Plot dar.

4.3 Aufgaben

- Implementieren Sie einen Code um einen der Transformationsparameter um den optimalen Wert zu variieren und die Metrikwerte für die jeweils aktuelle Position anzugeben. Speichern Sie die Ergebnisse in eine Datei und plotten Sie die Ergebnisse in Matlab. Die Plots sollen als Bilddatei gespeichert werden. Führen Sie die Aufgabe für alle Transformationsparameter durch und beachten sie den unterschiedlichen Wertebereich der Translations- und Rotationsparameter!

- Wiederholen Sie die Messung für einen Translations- und einen Rotationsparameter im multimodalen Fall.

- Variieren Sie die Anzahl Histogramm-Bins für die Berechnung der Mutual Information und plotten Sie die Ergebnisse in einem gemeinsamen Plot.

4.4 Auswertung

- Übernehmen Sie die Plots in ihr Protokoll. Diskutieren Sie die Ergebnisse und welche Metrik in welchen Fällen geeignet ist.
- Diskutieren Sie den Einfluss der Anzahl Histogramm-Bins anhand der Ergebnisse.

5 Versuchsteil 3: Rigide 3D-Registrierung von medizinischen Bildaten

Multimodale Bildregistrierung ist ein elementarer Bestandteil der medizinischen Bildanalyse. Die Kombination von Informationen aus verschiedenen Bildaufnahmen soll in diesem Praktikumsteil anhand von CT- und MR-Kopfdaten getestet werden. Hierfür werden sowohl ein Trainingsset als auch mehrere Testsets von CT- und MR-Daten genutzt. Für das Trainingsset, das bereits im letzten Versuchsteil verwendet wurde, ist die optimale Transformation gegeben, sodass das erzielte Ergebnis mit der gegebenen Lösung verglichen werden kann. An diesem Datensatz soll ein Algorithmus entwickelt und getestet werden, um die CT- und MR-T2-Daten zu registrieren. Anschließend sollen mit Hilfe des entwickelten Programms die optimalen Transformationen für die Testdaten ermittelt werden.

5.1 Implementierung des Algorithmus

Wie bereits im letzten Versuchsteil erfolgt auch hier die Implementierung in C++ unter Zuhilfenahme der ITK-Bibliothek. Der Code der für diese Aufgabe zur Verfügung steht enthält alle notwendigen Werkzeuge um eine Registrierung durchzuführen. Für die Registrierung wird Mutual Information zusammen mit einem Quasi-Newton-Optimierer verwendet. Da bei den Daten nur von einer rigiden Transformation ausgegangen wird, wird eine solche hier für die Registrierung zugrunde gelegt. Neben einzelnen Komponenten wie Metrik, Transformation oder Optimierer bietet das ITK-Framework auch eine Klasse zur Registrierung, die die Komponenten miteinander verbindet. Für eine zuverlässige Registrierung von großen Bildvolumen wird die Registrierung häufig auf mehreren Auflösungen nacheinander durchgeführt, wobei man mit einer groben Auflösung startet und diese dann immer feiner werden lässt. Zusammen mit einer Verkleinerung der ursprünglichen Auflösung wird häufig auch eine Glättung des Bildes durchgeführt. Die auf einer Auflösung ermittelten Parameter werden nach jeder Auflösung als Startwerte für den Optimerer auf der nächsten Auflösung verwendet. Hierdurch wird das Verfahren robuster gegenüber der Konvergenz in lokalen Optima der Metrik. Die Umsetzung dieses Ansatzes ist bei ITK in der Registrierungsklasse implementiert. Angegeben werden müssen hierbei die Anzahl Auflösungen, die verwendet werden sollen, die Glättungsparameter, sowie die Faktoren, um die die Auflösung in den einzelnen Auflösungen reduziert werden sollen.

Mit diesen Werkzeugen kann bereits eine erfolgreiche Registrierung durchgeführt werden, es zeigt sich aber, dass bei so großen Bilddaten wie den hier verwendeten, die Registrierung sehr langsam ist. Um die Registrierung zu Beschleunigen kann man nur eine Teilmenge der Informationen verwenden, um den Metrikwert zu berechnen. Die Auswahl dieser Teilmenge kann zum Beispiel auf einem groben Gitter liegen oder eine zufällige Auswahl an Punkten sein. Die Registrierungsklasse bietet auch die Möglichkeit eine Sampling-Strategie auszuwählen und die Prozentzahl der Punkte die verwendet werden sollen bzgl. der maximalen Anzahl an Bildpunkten.

Variieren Sie die Sampling-Prozentzahl sowie die Sampling-Strategie und do-

kumentieren Sie die Programm-Laufzeit sowie das Registrierungsergebnis. Außerdem sollten Sie die Anzahl Auflösungen und dazugehörige Glättungsparameter und Auflösungsfaktoren so anpassen, dass Sie ein zufriedenstellendes Registrierungsergebnis erhalten.

Das Ergebnis der Registrierung wird in eine Datei gespeichert und kann z.B. mit dem freien Programm ITK-Snap betrachtet werden. Mit diesem Programm ist es auch möglich das registrierte Volumen dem Referenzvolumen zu überlagern und in verschiedenen Farben darzustellen. Hiermit können Sie die Qualität der Registrierung visuell prüfen und 2D-Schnitte der Volumen als Bilddatei speichern.

5.2 Anwendung auf die Testdaten

Wenn Sie mit dem Ergebnis auf den Trainingsdaten zufrieden sind können Sie ihren Registrierungsalgorithmus auf die Testdaten anwenden. Bestimmen Sie für die vorliegenden Testdaten die Transformationsparameter und übernehmen Sie ein Bild mit den überlagerten Volumen in ihr Protokoll.

5.3 Aufgaben

- Konfigurieren Sie den Registrierungsalgorithmus mit Hilfe der Trainingsdaten. Hierbei soll insbesondere der Einfluss der Samplingrate sowie der Konfiguration der Auflösungen auf das Registrierungsergebnis untersucht werden.
- Testen und dokumentieren Sie das Zeitverhalten und die Registrierungsgenauigkeit bzgl. der für die Metrik-Berechnung verwendeten Bildpunkte (Prozentzahl).
- Wenden Sie den Algorithmus auf die Testdaten an und dokumentieren Sie die Ergebnisse.

5.4 Auswertung

- Diskutieren Sie das Zeitverhalten und die Registrierungsgenauigkeit bzgl. der für die Metrik-Berechnung verwendeten Bildpunkte (Prozentzahl).
- Diskutieren Sie die Ergebnisse der Registrierung der Testdaten. Welche Probleme traten auf und wie ist die Qualität der Registrierung nach visueller Betrachtung der Ergebnisse?

6 Versuchsteil 4: Deformierbare Registrierung

Der letzte Versuchsteil demonstriert die Funktionsweise einer deformierbaren Registrierung. Hierfür wird ein Bild von einem weißen Rechteck auf schwarzem Hintergrund erzeugt und ein Bild von einem weißem Kreis auf schwarzem Hintergrund. Ziel der Registrierung ist das Bild vom Rechteck so zu deformieren, dass das deformierte Bild einen Kreis darstellt.

Auch für diesen Aufgabenteil nutzen wir die ITK-Bibliothek in C++. Statt der rigiden Transformation wird in diesem Versuchsteil für die deformierbare Registrierung eine Spline-basierte Transformation gewählt. Da hier deutlich mehr Parameter optimiert werden müssen als bei einer rigiden Transformation ist es auch sinnvoll einen anderen Optimierer zu wählen, der schnell eine große Menge an Parametern optimieren kann. Genutzt wird für diese Aufgabe der LBFGS-Optimierer. Da hier ein monomodales Problem vorliegt kann eine Metrik verwendet werden, die direkt die Bildintensitäten vergleicht.

Die wesentliche Eigenschaft, die hier betrachtet werden soll ist der Einfluss der Anzahl der Gitterpunkte des Transformationsgitters auf das Registrierungsergebnis sowie die Laufzeit der Registrierung. Hierfür soll die Anzahl der Gitterpunkte variiert und die Ergebnisse dokumentiert werden. Die transformierten Bilder werden direkt als Bilddatei gespeichert.

Nachdem ein Transformationsfeld ermittelt wurde, dass das Rechteck im bewegten Bild zufriedenstellend zu einem Kreis transformiert soll dieses Transformationsfeld zum Abschluss noch auf ein Bild mit Gitterlinien oder einem Schachbrettmuster angewendet werden, um den Effekt des Feldes in den einzelnen Bereichen des Bildes zu visualisieren. Diskutieren sie das Ergebnis. Falls sie noch Zeit haben und noch etwas an deformierbaren Registrierungen experimentieren wollen können sie auch noch andere geometrische Formen als Input-Daten für die Registrierung ausprobieren.

6.1 Aufgaben und Auswertung

- Variieren Sie die Anzahl Gitterpunkte die die Transformation definieren und dokumentieren Sie das Zeitverhalten sowie das Registrierungsergebnis. Starten Sie mit einer Zahl, die ein zufriedenstellendes Ergebnis liefert und reduzieren Sie die Anzahl immer weiter.

- Erstellen Sie ein Funktion, die ein Gitterbild oder Schachbrett mit den Maßen der bisherigen Bilder erzeugt.

- Wählen Sie eine Transformation, die zu einem guten Ergebnis führte und wenden Sie das ermittelte Deformationsfeld auf das selbst erstellte Gitterbild an.

- Diskutieren Sie die Ergebnisse.

7 Spezielle Matlab Befehle

7.1 Allgemeine Syntax

For Schleife

```
x=0;
for i=1:1:10
x = x+i;
end
```

7.2 Umgang mit der Kamera in Matlab

Im ersten Versuchsteil sollen die Daten einer USB-Kamera in Matlab ausgewertet werden. Dies wird durch das 'MATLAB Support Package for USB Webcams' ermöglicht. Hierfür muss in Matlab zunächst ein Kameraobjekt erstellt werden. Dies erfolgt mit:

```
camList = webcamlist;
cam = webcam(1);
```

Wenn die Kamera nicht mehr benötigt wird, kann sie mit

```
clear('cam');
```

wieder freigegeben werden.

Bei dem Kameraobjekt können auch die Einstellungen der Kamera wie zum Beispiel Auflösung oder Zoom verändert werden und der Name der Kamera eingesehen werden. Dies ist hilfreich, um zu prüfen, ob die Kamera richtig erkannt wurde. Eine komplette Liste der Kameraeigenschaften findet man in der MathWorks-Online-Hilfe unter 'Acquire Images from Webcams'. Mit

```
cam.Resolution = cam.AvailableResolutions(4);
```

kann der Kamera z.B. die Auflösung zugewiesen werden, die an der vierten Stelle in der Liste der möglichen Kameraauflösungen steht.

```
cam.Zoom = 10;
```

stellt den Zoom auf 10. Analog können auch andere Eigenschaften des Kameraobjektes verändert werden.

Um die aktuelle Kameraeinstellung betrachten zu können kann man sich das Kamerabild in einem Fenster zeigen lassen. Dies erfolgt mit:

```
preview(cam)
```

Das Fenster kann entweder mit der Maus geschlossen werden oder vom Code aus mit:

```
closePreview(cam)
```

Ist das Kameraobjekt eingerichtet und getestet kann man mit

```
bild = snapshot(cam);
```

ein Bild im Arbeitsbereich von Matlab speichern. Dieses ist als Array gespeichert und kann mit

```
figure1 = figure;
imshow(bild);
```

angesehen werden. Im Fenster in dem das Bild so gezeigt wird gibt es auch eine Option das Bild auf der Festplatte in einem gängigen Bildformat zu speichern. Diese Bilder können dann gegebenenfalls in die Auswertung eingefügt werden. Für die spätere Registrierung ist es einfacher mit Bildern mit nur einem Farbkanal (Grauwertbilder) anstelle von Farbbildern mit drei Farbkanälen zu arbeiten. Ein Farbbild ('rgb') kann in ein Grauwertbild umgewandelt werden mit:

```
bild_grau = rgb2gray(bild_rgb);
```

In vielen Fällen ist es hilfreich, wenn das Programm auf eine Nutzereingabe warten, bevor der Code weiter ausgeführt wird. Eine solche Unterbrechung erreicht man z.B. durch ein Mitteilungsfenster, bei dem auf das Klicken von 'OK' gewartet wird bevor der Code weiter ausgeführt wird. Ein einfaches Fenster kann erstellt werden mit:

```
uiwait(msgbox('Hier einen passenden Text eintragen'));
```

7.3 Bildregistrierung in Matlab

Für die Bildregistrierung im ersten Versuchsteil benötigen wir eine Metrik und einen Optimierer. Hierfür wird die Image Processing Toolbox von Matlab benötigt. Zunächst kann eine Konfiguration von Metrik und Optimierer mit

```
[optimizer,metric] = imregconfig(modality)
```

vorgenommen werden, wobei für modality 'monomodal' oder 'multimodal' eingesetzt werden kann. Ähnlich wie bei der Kamera können auch beim Optimierer Eigenschaften konfiguriert werden wie z.B. die maximale Anzahl an Iterationen.

Die eigentliche Registrierung erfolgt mit:

```
T = imregtform(bild1,bild2,transformationType,optimizer,metric);
```

Das Ergebnis der Registrierung wird als Transformation zurückgegeben. bild1 und bild2 sind hierbei die zu registrierenden Bilder. transformationType definiert welche Art von Transformation bei der Registrierung verwendet werden soll. Dabei stehen als Optionen 'translation' , 'rigid' , 'similarity' und 'affine' zur Auswahl.

Will man ein Bild mit einer Transformation in das Koordinatensystem des anderen Bildes transformieren benötigt man Informationen über das Koordinatensystem des Referenzbildes sowie die Transformation:

```
Rreferenz = imref2d(size(referenzbild));
bild_transformiert = imwarp(bewegtesbild,T,'OutputView',Rreferenz);
```

Das transformierte Volumen und das ursprüngliche Volumen können auch gemeinsam in einem Bild überlagert dargestellt werden:

```
figure
imshowpair(bild_transformiert, referenzbild);
```

8 Nützliche C++ Befehle

8.1 Hilfsfunktionen

Da viele Abläufe wie das Laden und speichern von Bilddaten immer wieder benötigt werden kann man sich oft Arbeit sparen, wenn man Funktionen schreibt in denen man den Code auslagert. Für dieses Praktikum wurden einige solcher Funktionen bereitgestellt, die im Folgenden vorgestellt werden.

ImageType::Pointer image = nrnLoadImage<ImageType> ("Pfad"); Diese Funktion wird im Praktikum verwendet um Bilddaten in das Programm zu laden. Hierbei muss zuvor der Typ des Bildes, also Dimensionalität und Datentyp der einzelnen Voxel als ImageType definiert werden und als Parameter der Pfad auf der Festplatte übergeben werden.

nrnSaveImage<ImageType>("Pfad/Name.Endung", image); Analog zu der Funktion LoadImage gibt es auch eine Funktion SaveImage um ein Ergebnis als Datei auf der Festplatte zu speichern. Auch hier muss der Typ des Bildes image als Template-Parameter angegeben werden. Neben dem zu speichernden Bild muss noch der Pfad angegeben werden wo das Bild gespeichert werden soll.

ImageType::Pointer newImage = resampleImage<ImageType, TransformType>(image,referenceImage,transform); Diese Funktion wird zur Überführung von einem Bild in das Koordinatensystem eines anderen Bilds genutzt. Zusätzlich wird eine Transformation als Parameter angegeben, die auf das Bild angewendet wird. Ist diese Transformation eine Einheitstransformation (z.B. Einheitsmatrix bei rigiden und affinen Transformationen) so wird das Bild nur in ein anderes Koordinatensystem übertragen.

8.2 ITK header für die Bereitstellung der Registrierungsfunktionalität

itkImageRegistrationMethodv4.h: Basisklasse für die Erstellung eines Bildregistrierungs-Frameworks in ITK

itkMattesMutualInformationImageToImageMetricv4.h: Header für eine Klasse zur schnellen Berechnung der Mutual Information. Standardmäßig wird hier eine kubische-BSpline Interpolation verwendet.

itkMeanSquaresImageToImageMetricv4: Header für eine Klasse zur schnellen Berechnung der mittleren quadratischen Abweichung.

itkEuler3DTransform: Implementierung einer rigiden Transformation mit Euler Winkeln.

itkVersorRigid3DTransform: Implementierung einer rigiden Transformation mit Versoren für die Definition der Rotationsparametern.

itkBSplineTransform: Implementierung einer deformierbaren Transformation mit B-Splines.

itkQuasiNewtonOptimizerv4: Implementierung eines Quasi-Newton-Optimierers.

itkLBFGSOptimizer: Implementierung eines LBFGS-Optimierers. Dieser Optimierer wird vor allem bei Problemen mit einer großen Anzahl an zu optimierenden Parametern verwendet.

8.3 Verwendung der ITK Klassen

In ITK muss für jedes Bild zunächst ein Type definiert werden der angibt wie viele Dimensionen das Bild hat und welchen Datentyp ein einzelner Voxel (3D Pixel) besitzt. Damit die Funktionen nicht für jeden Datentypen neu implementiert werden müssen ist ITK template-basiert, d.h. man kann den Klassen/Funktionen mitteilen mit welchem Datentyp man sie verwenden will. Um den Code übersichtlich zu halten definiert man zunächst die gewünschte Klasse mit den gewünschten Datentypen unter einem neuen Namen. Im Folgenden einige Beispiele für die im Abschnitt zuvor genannten Klassen:

- typedef itk::Image< double, 3 > ImageType;
- typedef itk::VersorRigid3DTransform<double> TransformType;
- typedef itk::ImageRegistrationMethodv4<ImageType,ImageType, TransformType > RegistrationType;

- typedef itk::MattesMutualInformationImageToImageMetricv4< ImageType,ImageType > MetricType;
- typedef itk::QuasiNewtonOptimizerv4 OptimizerType;

Anschließend kann man sich eine Instanz dieser Klasse erstellen mit der im weiteren Verlauf gearbeitet werden kann. Dies sieht z.B. so aus:

- TransformType::Pointer transform = TransformType::New();
- MetricType::Pointer metric = MetricType::New();
- OptimizerType::Pointer optimizer = OptimizerType::New();
- RegistrationType::Pointer registration = RegistrationType::New();

Jetzt haben wir schon eine Transformation, eine Metrik und einen Optimierer mit dem gearbeitet werden kann. In den meisten Fällen wollen wir jedoch einige Attribute unserer Instanzen ändern wie z.B. die Parameter einer Transformation, die Anzahl Bins die zur Berechnung der Mutual Information verwendet werden oder Abbruchbedingungen für den Optimierer. Die meisten dieser Eigenschaften eines Objects lassen sich mit eigenen *SetWert(Wert)*-Funktionen verändern. Neben wir zum Beispiel die Histogramm-Bins so müssten wir zum Ändern der Bin-Zahl auf den Wert 50 schreiben:

```
metric->SetNumberOfHistogramBins(50);
```

Ähnlich ist es bei der Instanz unserer Registrierungsklasse. Hier wollen wir zwar keinen Zahlenwerte zuweisen, müssen aber die Transformation, die Metrik und den Optimierer für die Registrierung festlegen und die Bilder angeben, die später registriert werden sollen. Dies erfolgt durch:

```
registration->SetMetric       (metric);
registration->SetOptimizer    (optimizer);
registration->SetFixedImage   (fixedImage);
registration->SetMovingImage  (movingImage);
registration->SetInitialTransform(transform);
```

Viele Klassen haben eine Vielzahl an Attributen mit denen man das Verhalten der Klasse und somit der späteren Registrierung beeinflussen kann. Ziel des Praktikums kann und soll es nicht sein die gesamte Bibliothek mit ihren Möglichkeiten durchzuarbeiten. Stattdessen wird für die C++ Aufgaben ein Framework vorgegeben sein, auf dessen Grundlage an einzelnen Teilen der Registrierung experimentiert und deren Einfluss auf das Registrierungsergebnis analysiert werden kann. Online kann zudem auch zu jeder der verwendeten ITK-Klassen eine detaillierte Beschreibung aller Methoden und Attribute gefunden werden.

9 Hinweise zur Entwicklungsumgebung

In diesem Praktikum wird sowohl Matlab als auch C++ für die Berarbeitung und Auswertung der Bilddaten verwendet. Dieses Kapitel bietet eine kurze Einführung in die verwendeten Entwicklungsumgebungen. Für die Erstellung der Plots in Aufgabe 2 dürfen auch andere Programme als Matlab verwendet werden.

9.1 Matlab

Im ersten Versuchsteil wird mit Matlab gearbeitet. Die Oberfläche des Programms unterteilt sich in verschiedene Unterfenster (siehe Abbildung 2). In das Command Window unten Mitte werden die Befehle eingegeben und mit Enter ausgeführt. Ein ';' am Ende der Zeile bewirkt, dass der Befehl keine Ausgaben im Command Window selbst erzeugt. Die bereits definierten Variablen und ihre Werte werden im Workspace rechts im Bild angezeigt. Durch Anklicken von Variablen im Workspace öffnet sich oberhalb des Command Windows ein weiteres Fenster, indem die Werte der Variablen in einer Matrix angezeigt werden. Will man ein größeres Programm schreiben, so empfiehlt es sich, die einzelnen Befehle nicht im Command Window einzugeben, sondern ein neues Skript (oben links im Bild) zu erstellen und die Befehle in dieses Skript zu schreiben. Ein Skript kann als Textdatei gespeichert werden und anhand seines Namens auch vom Command Window aus aufgerufen werden.

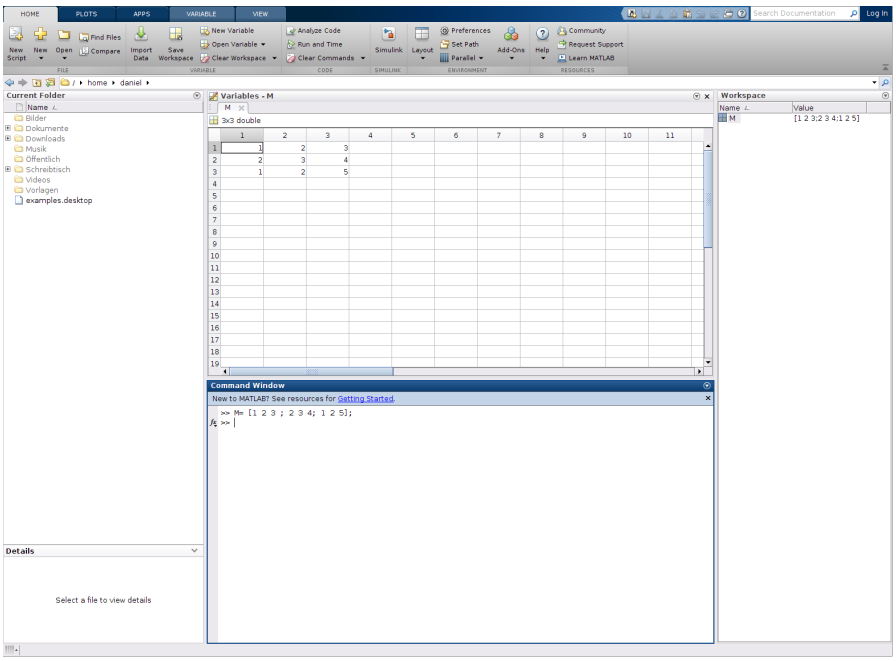


Abbildung 2: Oberfläche der Matlabumgebung

9.2 C++

In den Versuchsteilen 2-4 wird mit C++ gearbeitet. Für alle drei Aufgaben wurde ein zip-Archiv bereitgestellt, dass Code-Templates und ein Makefile für die jeweilige Aufgabe beinhaltet. Diese Ordner sollen zunächst in den eigenen Ordner kopiert und da z.B. mit 7-zip entpackt werden. Jeder Ordner enthält einen Ordner Source-Ordner und einen Build-Ordner, wobei der Build-Ordner zu Beginn leer ist. In diesem wird später das C++ Projekt generiert. Für die Kompilierung des Codes und als Entwicklungsumgebung für C++ wird in diesem Praktikum Visual Studio verwendet. Zunächst muss jedoch aus den Code-Templates für die einzelnen Aufgaben ein Visual Studio Projekt erstellt werden. Hierfür wird das Programm Cmake verwendet.

9.2.1 cmake

Cmake konfiguriert anhand von den Anweisungen die in einem Makefile stehen ein Projekt und sammelt dabei die Quellcode-Dateien für das Projekt zusammen mit den für das Projekt notwendigen Bibliotheken. Daraus wird dann ein Visual Studio Projekt generiert, dass mit Visual Studio gestartet werden kann. Die Oberfläche von cmake ist in Abbildung 3 gezeigt.

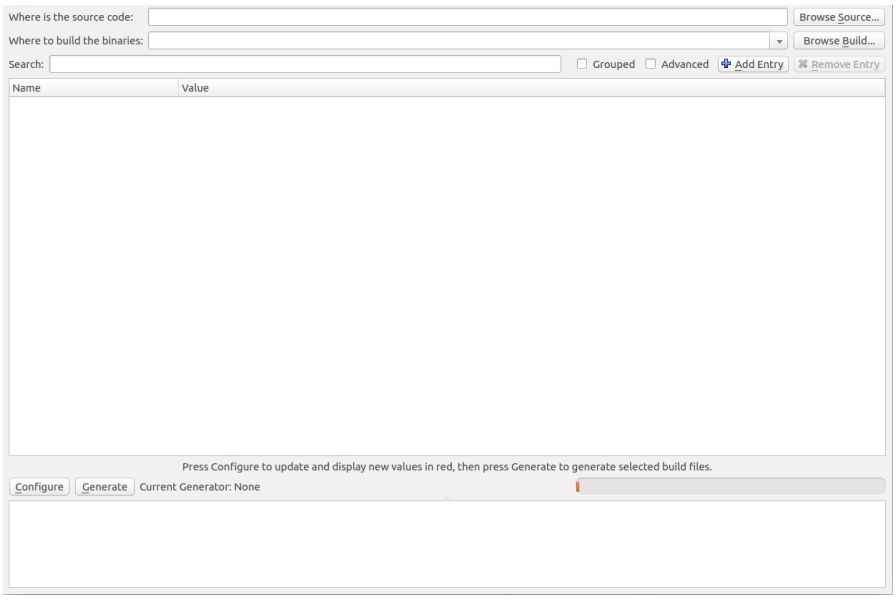


Abbildung 3: Oberfläche von Cmake

In obersten Eingabefeld wird nach dem Ordner mit den Source-Dateien des Projekts gefragt. Hier muss der Pfad zu dem jeweiligen Source-Ordner der entsprechenden Aufgabe eingefügt werden. Entsprechend muss in dem Feld darunter der Pfad zu dem zugehörigen Build-Ordner der Aufgabe eingefügt werden. Man kann auch mit 'Browse Source' und 'Browse Build' ein Fenster öffnen um nach den Ordner zu suchen. Anschließend wird durch das Drücken des Buttons **Configure** die Konfiguration des Projekts gestartet. Hierbei wird man nach dem Compiler gefragt, der verwendet werden soll. Hier muss der neuste Visual Studio Compiler

mit der Version Win64 ausgewählt werden. Die Konfiguration dauert etwas, da der Compiler und die Projektabhängigkeiten an dieser Stelle zunächst geprüft werden.

Bei den C++ Aufgaben wird die Bibliothek ITK verwendet. Es kann sein, dass Cmake den Pfad hierzu nicht automatisch findet. In diesem Fall muss der Pfad nach der Konfiguration manuell eingefügt werden. Hierfür sucht man im mittleren Feld die Variable mit dem Namen ITK und gibt auf der rechten Seite beim Feld 'Value' den Pfad 'C:/APIs/ITK-4_9_0' ein. Durch erneutes Drücken von **Configure** kann geprüft werden, dass nun alle Pfade gefunden werden.

Zum Abschluss wird durch Drücken von **Generate** das Visual Studio Projekt erstellt.

9.2.2 Visual Studio

Nach dem erstellen eines C++Projektes kann mit diesem in Visual Studio gearbeitet werden. Nach dem ersten Öffnen des Projektes müssen zunächst einige Einstellungen vorgenommen werden. Abbildung 4 zeigt die Oberfläche von Visual Studio mit einem geöffneten Projekt. Im linken Bereich sieht man dabei das Projektmanager Fenster. Hier können die zu dem Projekt gehörenden Dateien eingesehen werden. Das mittlere Fenster ist das Editor Fenster in dem einzelne Dateien bearbeitet werden können. Das Fenster rechts im Bild ist für die Kompilerausgabe. In diesem werden Fehler beim Compilieren angezeigt und der Status des Projekts nach dem Kompilieren/Erstellen.

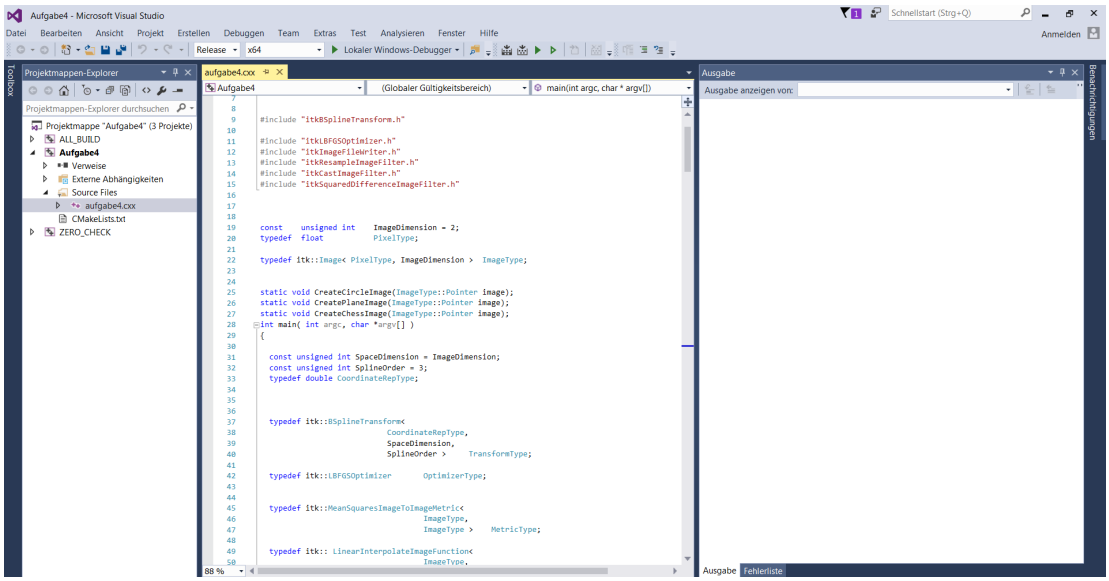


Abbildung 4: Oberfläche von Visual Studio

Die Shortcuts über dem Editor Fenster werden für das Kompilieren, Erstellen und Starten des aktiven Projektes verwendet. Das gefüllte grüne Dreieck mit der Spitze nach rechts steht für das Starten eines Projektes mit zusätzlichem Debugger. Ein Debugger kann eingesetzt werden, um den Status und Werte einzelner Variablen zu bestimmten Zeiten im Programm zu überprüfen oder um den Ursprung

eines Fehlers im Programm zu lokalisieren. Hierfür ist es notwendig, dass das Programm zuvor mit Debug-Informationen kompiliert wurde. **Dadurch wird insebsondere bei der hier durchgeführten Bildregistrierung das Programm zur Laufzeit deutlich langsamer!** Um keinen Debugger zu verwenden kann der nicht gefüllte grüne Pfeil rechts davon verwendet werden. Die Einstellung, ob das Programm mit (RELEASE) oder ohne (DEBUG) Debug-Informationen kompiliert wird, kann im weiß unterlegten Feld (im Bild mit dem Text 'Release' oberhalb des Editor-Fensters) eingestellt werden. **Wenn ihr keine Laufzeitfehler im Programm habt empfehlen wir hier darauf zu achten, dass dieses Feld auf 'Release' steht!** In dem Feld rechts daneben wird eingestellt, ob das Projekt als x86 oder x64 Projekt erstellt werden soll. Dieses Feld muss auf x64 stehen. Zwischen diesen Feldern und den Pfeilen (grünen Dreiecken) zum Starten des Programms befinden sich zwei Icons zum Kompilieren und Erstellen des Projekts.

Damit Visual Studio erkennt, welches Projekt gestartet werden soll muss zunächst das aktuell geöffnete Projekt als Startprojekt festgelegt werden. In dem Beispiel in Abbildung 4 wurde das Projekt 'Aufgabe4' geöffnet. Im Fenster des Projektmanagers können darüber und darunter noch Projekte mit den Namen 'ALL_BUILD' und 'ZERO_CHECK' gefunden werden. Mit einem Rechtsklick auf 'Aufgabe4' kann aus dem Auswahlmenu der Befehl **'Als Startprojekt festlegen'** ausgewählt werden. Danach sollte der Name 'Aufgabe4' wie in Abbildung 4 fettgedruckt im Projektmanager erscheinen. Solltet ihr diesen Schritt vergessen, erhaltet ihr beim Starten später den Fehler, dass das Projekt ALL_BUILD nicht gestartet werden kann.

Ansonsten könnt ihr nun die eigentlichen Änderungen am Code im Editor-Fenster durchführen.

10 Fragenkatalog

- Wo wird Bildregistrierung in der Medizin genutzt?
- Welche Arten von Transformation unterscheidet man in der Bildregistrierung?
- Warum macht es für die Wahl der Metrik einen Unterschied ob man ein monomodales oder ein multimodales Problem vorliegen hat?
- Warum wird eine Registrierung oftmals auf mehreren Bildauflösungen durchgeführt?
- Was ist Mutual Information?
- Was beschreibt die Sum of Squared Differenzes (SSD) Metrik?
- Warum unterscheidet man zwischen Trainings- und Testdaten?
- Wie wird das Joint-Histogramm erstellt?
- Wie funktioniert ein MRT?
- Welche Materialien werden in MRT bzw. CT besonders deutlich dargestellt?
- Was ist der Unterschied zwischen einem im Debugmodus und einem im Releasemodus erstellten Projekt?
- Nach welchen Kriterien kann ein Algorithmus zur Bildregistrierung bewertet werden?

11 Referenzen und Literatur zum Nachschlagen

- Weishaupt, D., Köchli, V. D., Marincek, B., Wie funktioniert MRI?, ISBN: 978-3-642-41616-3
- D. L. G. Hill, P. G. Batchelor, M. Holden, D. J. Hawkes, Medical image registration, Physics in Medicine and Biology 46(3) (2001) R1-R45
- F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, P. Suetens, Multimodality image registration by maximization of mutual information, IEEE Transactions on Medical Imaging 16(2) (1997) 187-198
- J. P. W. Pluim, J. B. A. Maintz, M. a. Viergever, Mutualinformation-based registration of medical images: a survey, IEEE Transactions on Medical Imaging 22(8) (2003) 986-1004
- ITK - Segmentation & Registration Toolkit (https://itk.org)