Generierung eines Durchschnittsobjekts und Quantifizierung von Unstimmigkeiten zwischen Annotationen mithilfe statistische Formanalyse

Ahmet Efe[1]
Betreuer: Dr. Daniel Kondermann [2]

Zusammenfassung
Die Ziele dieses Fortgeschrittenenpraktikums sind das Generieren eines Durchschnittsobjekts aus mehreren Annotationen und das Quantifizieren der Unstimmigkeiten zwischen diesen Annotationen. Zum Erreichen der Ziele wurde mithilfe der Software *Deformetrica* statistische Formanalyse an einem Beispieldatensatz durchgeführt. Ein erweitertes Programm konnte die Unstimmigkeiten quantifizieren, ob das Generieren eines Durchschnittsobjekts erfolgreich war, konnte nicht abschließend geklärt werden und Bedarf einen weiteren Versuch mit einem anderen Datensatz. Grundsätzlich lässt sich aus den Ergebnissen ableiten, dass statistische Formanalyse für diese Art von Aufgaben geeignet ist.

Code zu finden unter:
`https://github.com/ahmetefe98/APDeformetrica`

[1] *ahmet.efe@stud.uni-heidelberg.de*
[2] *daniel@kondermann.de*

Inhaltsverzeichnis

1. Einleitung

1.1 Motivation

Maschinelles Lernen spielt heutzutage eine sehr wichtige Rolle in der Wissenschaft und Wirtschaft. In den letzten Jahrzehnten wurden gute Methoden erforscht und im Anschluss von der Industrie umgesetzt. Jedoch benötigen auch die besten Methoden zahlreiche qualitativ hochwertige Daten, die teilweise manuell erstellt werden müssen. Das Generieren dieser großen Datensätze ist sehr mühsam und zeitaufwendig. Im Falle von Bild Annotationen müssen Menschen die Objekte markieren und benennen. Für das Markieren eines Objekts innerhalb eines Bildes stehen verschiedene Möglichkeiten zur Verfügung. Die einfachste Vorgehensweise ist das Umrunden des Objektes durch eine *bounding box*, dabei zeichnet ein Annotator um das Objekt ein Rechteck (im zweidimensionalen) oder ein Quader (im dreidimensionalen). Die *bounding box* soll so klein wie möglich sein, jedoch das ganze Objekt beinhalten. Manche Anwendungsfälle benötigen detailliertere Annotation u. a. *polygon segmentation*, bei dem die Umrisse des gesamten Objektes nachgezeichnet werden, sodass die Markierung nur das Objekt beinhaltet. Diese Annotation ist wesentlich komplexer, aber auch

viel genauer als *bounding box*, daher wird jedes Objekt von mehreren Annotatoren markiert. Aufgrund der Komplexität gibt es leider Unstimmigkeiten zwischen den Annotationen, sodass diese eine zusätzliche Bearbeitung und Analyse benötigen. [1]

1.2 Ziele des Fortgeschrittenenpraktikums

Aufgrund der Unstimmigkeiten zwischen den Annotationen ist ein Ziel dieses Fortgeschrittenenpraktikums ein Durchschnittsobjekt aus allen Annotationen zu generieren, sodass dieser fürs Maschinelle Lernen verwendet werden kann. Ein weiteres Ziel ist das Herausfinden der Stellen des Objekts, an dem Unstimmigkeiten zwischen den Annotationen existieren und wie stark diese sind. Zum Erreichen beider Ziele wird statistische Formanalyse mithilfe der Software *Deformetrica* [2] auf einem Beispieldatensatz angewendet. Der dafür benötigte Code soll so konzipiert sein, damit dieser mit wenig Aufwand auf einem anderen Datensatz anwendbar ist.

1.3 Aufbau des Berichts

Kapitel 2. Theoretische Aspekte umfasst die theoretischen Grundlagen, die für dieses Fortgeschrittenenpraktikum nötig sind. Der Hauptteil dieses Kapitels behandelt die Software *Deformetrica. Kapitel 3. Praktische Umsetzung und Ergebnisse* beinhaltet die Vorbereitung und Durchführung der Berechnung und zudem die Auswertung und Analyse der Ergebnisse. Das Kapitel endet mit einer Anleitung zur Benutzung des während dem Fortgeschrittenenpraktikum entstandenen Programms. *Kapitel 4* rundet diesen Bericht mit einem Fazit und Ausblick ab. Zur Übersichtlichkeit sind fast alle Abbildungen im Anhang platziert.

Der dem Bericht zugrunde liegende Code ist zu finden unter: `https://github.com/ahmetefe98/APDeformetrica`

2. Theoretische Aspekte

Das zweite Kapitel befasst sich mit den theoretischen Hintergründen dieses Projektes und beginnt mit einer kurzen Beschreibung des Datensatzes. Der Großteil dieses Kapitels behandelt die genutzte Software *Deformetrica* und liefert neben einer allgemeinen Erklärung auch die mathematischen Hintergründe. Zudem wird auf die öffentlich verfügbare Anleitung der Software eingegangen und das Kapitel mit einer Erklärung über eine Auswahl von Parametern abgeschlossen.

2.1 Eingabedaten

Dieser Ansatz untersucht fünf Annotationen eines Motorrads. Dafür wurden die Umrisse des Motorrads von fünf Annotatoren nachgezeichnet und liegen als *PNG* Dateien vor. Das Ergebnis ist im Anhang in Abbildung 2. Auf dem ersten Blick sehen die verschiedenen Annotationen sehr ähnlich aus, bei genauerem Betrachten, vor allem beim Aufeinanderlegen, sind jedoch die Unterschiede bemerkbar. Siehe dazu Abbildung 3. Es wird deutlich, dass manche Annotatoren detaillierter als andere gezeichnet haben. Dabei stechen zwei Bereiche ins Auge: 1. das Lenkrad und 2. die Verbindung zwischen Vorderrad und Rest des Motorrads. Hier sind die Unterschiede deutlich.

2.2 Deformetrica

Die statistische Formanalyse wird mithilfe der Open Source Software *Deformetrica* [2], das auf dem LDDMM[1] Framework basiert, durchgeführt. *Deformetrica* bietet drei Funktionalitäten an:

- *registration:* berechnet die bestmögliche Verformung von einem Start- zu einem Zielobjekt
- *atlas construction:* berechnet ein durchschnittliches Objekt aus allen Daten und die Entstehung der einzelnen Daten aus diesem Durchschnitt
- *geodesic regression:* „interpoliert“ Daten, die durch die Zeit indiziert sind, um Zwischendaten berechnen zu können

Zum Erreichen der beiden Ziele, das Generieren eines Durchschnittsobjekts und das Herausfinden der Stellen mit Unstimmigkeiten, wird die Funktion *atlas construction* genutzt. Diese Funktion liefert zum einen das gewünschte Durchschnittsobjekt und zum anderen die Möglichkeit, aus diesem Durchschnittsobjekt alle Objekte durch Verformung zu erzeugen. Letzteres kann bei der Bestimmung der Unstimmigkeiten helfen. [2]

2.2.1 LDDMM-Framework

LDDMM vergleicht Objekte mithilfe diffeomorpher Transformation des Umgebungsraums. Die Voraussetzungen dafür sind das Parametrisieren einer großen Familie von Transformationen und die Berechenbarkeit der Abstände zwischen Objekten. [3]

[1] Large deformation diffeomorphic metric mapping

Definition Diffeomorphimus Eine bijektive Abbildung heißt Diffeomorphismus, falls die Abbildung und ihre Umkehrabbildung stetig differenzierbar sind. [4]

Definition Transformation (Hier Koordinatentransformation) Als Koordinatentransformation wird die bijektive und beliebig differenzierbare Übertragung von Koordinaten von einem in ein anderes Koordinatensystem bezeichnet. [5]

Parametrisierung der Transformation

Deformetrica benutzt für die Verformung n Kontrollpunkte $(q_i)_{i=1,\dots,n}$ und Vektoren $(\mu_i)_{i=1,\dots,n}$ die ihren Ursprung an den Kontrollpunkten haben. Im Bericht werden diese Vektoren als *Momenta* bezeichnet. (Zur Veranschaulichung ist in der Abbildung 1 eine Skizze mit Kontrollpunkten und *Momenta* zu sehen.) Mithilfe der Kontrollpunkte und *Momenta* erzeugt *Deformetrica* ein Vektorfeld X auf dem gesamten Raum. Zur Auswertung des Vektorfelds X an der Stelle x:

$$X(x) = \sum_{i=1}^{p} K(x, q_i) * \mu_i \tag{1}$$

Wobei

$$K(x,y) = exp\left(\frac{\|x-y\|^2}{\sigma^2}\right) \tag{2}$$

ein Gauß-Filter mit der Breite σ ist. σ ist eines der Parameter, die der Benutzer im Vorfeld festlegen muss. Aus einem hohen Wert resultieren glatte und globale Verformungen, wohingegen bei einem niedrigen Wert die Verformungen eine geringere Genauigkeit haben.

Durch die Kontrollpunkte q und die *Momenta* μ ist die Verformung des Umgebungsraums vollständig parametrisiert. Diese Transformation hat die Bezeichnung $\Phi_{(q,\mu)}$. Für Bilder gilt:

$$\Phi_{(q,\mu)}(I) = I \circ \Phi^{-1}_{(q,\mu)} \tag{3}$$

Dabei ist I eine Funktion von $\mathbb{R}^2$ auf $\mathbb{R}$. Das heißt, dass die Verformung eines Bildes durch die Faltung zwischen den Kontrollpunkten und *Momenta* und den Pixeln des Bildes berechnet wird. Bei einer Repräsentation des Objekts durch ein Polygonnetz dienen die Eckpunkte des Netzes zur Berechnung der Verformung. [3]

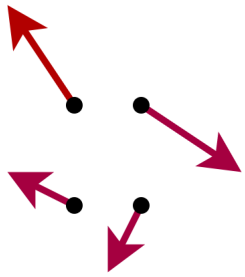


Abbildung 1. Skizze von Kontrollpunkten (schwarz) und *Momenta* (rot)

Abstand zwischen Objekten

Der Abstand zwischen Objekten bestimmt die Ähnlichkeit dieser Objekte zueinander.
Im Falle zweier Bilder x und y erfolgt die Abstandsberechnung durch den euklidischen Abstand $d(x,y)$.

$$d(x,y) = ||x-y||_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4}$$

Wobei n die Anzahl der Pixel ist.
Im Falle von Polygonnetzen steht entweder der *„current“* oder der *„varifold“* Abstand zur Verfügung. Die mathematischen Formeln beider Berechnungen sind zu finden unter [3]. Dafür muss der Benutzer ein σ im Vorfeld festlegen, in diesem Fall steht dieser Parameter für die Höhe der Übereinstimmung. Bei einem niedrigen Wert ist eine sehr genaue Übereinstimmung erwünscht, wohingegen bei einem hohen Wert eine global gute Übereinstimmung erzielt wird. [3]

2.2.2 Deformetrica Anleitung

Unter [6] ist eine Anleitung zur Benutzung der Funktionalität *atlas construction* an einem Beispieldatensatz mit Schädelbildern. Zuerst installiert die Anleitung *Deformetrica (enviroment setup)* und visualisiert die Daten *(explore the data)*. In der Ausgabe von *explore the data* sind neben den fünf Schädeln auch eine Vorlage zu sehen, die *Deformetrica* als Startpunkt für die Erzeugung eines Durchschnittsobjekts benötigt. Nach Auswahl der Parameter startet die Berechnung *(import and run Deformetrica)*. Die Codezelle *plot the resulting templates* wertet die Berechnung aus und visualisiert die Ergebnisse. Das erste Bild *initial template* ist die Vorlage, die bereits in *explore the data* zu sehen war. Das zweite Bild *estimated template* ist das aus der Vorlage entstandene Durchschnittsobjekt und erfüllt eines der Ziele dieses Fortgeschrittenenpraktikums. Auf dem dritten Bild sind

verschiedenfarbige *Momenta* zusehen, mit denen ausgehend vom Durchschnittsobjekt die einzelnen Schädel rekonstruiert werden. In der zweiten Reihe sind in Schwarz die originalen und in Farbe die aus dem Durchschnittsobjekt und den *Momenta* rekonstruierten Bilder zu sehen. Aus diesen Visualisierungen lässt sich ablesen, dass in diesem Beispiel die Rekonstruktion nicht gelungen ist, jedoch das Durchschnittsobjekt sehr gut den Datensatz abbildet. Das zweite Ziel dieses Fortgeschrittenenpraktikums, das Erkennen der Unstimmigkeiten zwischen den Annotationen, lässt sich mithilfe dieser Visualisierungen nicht erreichen. Daher dient diese Anleitung als Startpunkt und benötigt weitere Anpassungen und Erweiterungen.

2.2.3 Parameter

Im Vorfeld der Berechnung kann beziehungsweise muss der Benutzer von *Deformetrica* Parameter bestimmen. Auflistungen aller Parameter für *Deformetrica* sind zu finden unter: [7] [8] [9]. Viele Parameter beziehen sich auf die Daten und die Aufgabe, daher sind diese vorgegeben. Es folgt eine Beschreibung von Parametern, die für eine Optimierung des vorliegenden Falls variiert werden können.

- *kernel-width (template):* steht für die Höhe der Übereinstimmung. Mit einem niedrigen Wert anfangen und gegebenenfalls erhöhen
- *kernel-width (deformation):* typische Breite der Verformung. Mit einem hohen Wert anfangen und gegebenenfalls verringern.
- *noise-std:* steuert wie gut das Objekt angepasst werden soll.
- *number-of-timepoints:* steuert die Anzahl an Diskretisierungsschritten. Voreinstellung ist 11, Berechnung wird genauer aber dauert bei einem höheren Wert auch länger.
- *attachment-type:* die Art der Abstandberechnung zwischen zwei Objekten. Zur Auswahl stehen *current* und *varifold*.

3. Praktische Umsetzung und Ergebnisse

Dieses Kapitel thematisiert die praktische Umsetzung zum Erfüllen beider Ziele. Dies beinhaltet auch das Vorbereiten der Daten und das Erstellen eigener Anfangskontrollpunkte. Ein wichtiger Bestandteil ist die Auswahl der Parameter. Nach der Berechnung von *Deformetrica* , werden die Ergebnisse ausgewertet und analysiert. Hierbei wird diskutiert, ob die Ergebnisse die Ziele erfüllen. Am Ende dieses Kapitels ist eine Anleitung zur Benutzung des im Laufe des Fortgeschrittenenpraktikums entstandenen Programms zu finden. Dieses Programm ist mit leichten Modifizierungen auf andere Datensätze anwendbar.

3.1 Vorbereitung der Daten

Deformetrica benötigt als Eingabe die Bilder als *VTK* Dateien. Daher konvertiert die vom Betreuer zur Verfügung gestellte Datei *png_to_vtk.ipynb* die als *PNG* Dateien vorliegenden Bilder. In Abbildung 4 ist ein Beispiel dieser Konvertierung zu sehen, a) ist das Bild als *PNG* Datei und b) dasselbe Bild als *VTK* Datei.

3.2 Anfangskontrollpunkte

Deformetrica setzt auf dem gesamten Raum Kontrollpunkte, um den Raum und damit alle Objekte von diesen Punkten aus zu verformen. Diese Punkte sind gleichverteilt, siehe graue Punkte in Abbildung 5. Jedoch hat dies zur Folge, dass viele Punkte außerhalb des interessanten Bereichs sind. Eine Möglichkeit, um dieses Problem zu beseitigen, ist das Bestimmen eigener Kontrollpunkte, sodass diese in der Nähe des Objektumrisses liegen. Da die Objekte durch miteinander verbundenen Punkte repräsentiert werden, könnten diese Punkte als Anfangskontrollpunkte dienen. Jedoch ist in Abbildung 6 zusehen, dass es einerseits zu viele Punkte ca. 700 und andererseits die Punkte nicht gleichverteilt sind. Dies hat zur Folge, dass manche Bereiche durch zu viele und andere Bereiche durch zu wenige Punkte repräsentiert sind.

Die Klasse *LineString* aus dem Modul *geometry* der Python-Bibliothek *shapely* kann durch eine Interpolation gleichverteilte Punkte in der gewünschten Anzahl generieren. Das Ergebnis mit 200 Kontrollpunkten ist zu sehen in Abbildung 7. Ein Vergleich zwischen den Koordinaten der Vorlage und den selbst erstellten Kontrollpunkten ist zu sehen in der Abbildung 8. Die Datei *create_own_controlpoints.ipynb* kann für jede beliebe *VTK* Datei eigene Kontrollpunkte in der gewünschten Anzahl generieren. Einfachheitshalber erzeugt das Programm für alle Bilder eigene Kontrollpunkte, sodass der Benutzer in *Bike.ipynb* als Parameter, die Kontrollpunkte, mit denen *Deformetrica* arbeiten soll, festlegen muss. Siehe dazu Unterabschnitt 3.7.

3.3 Parameterauswahl

Im Unterunterabschnitt 2.2.3 ist eine Erklärung über die Parameter, die zur Optimierung variiert werden können, zu finden. Um herauszufinden, welche Parameterwerte die Besten für diesen Datensatz sind, benötigt jeder Parameter eine isolierte Betrachtung. Das Ziel ist das Verringern des Abstandes zwischen dem originalen und dem rekonstruierten Objekt. Daher berechnet das Programm nach jeder Berechnung die Abstände für alle fünf Objekte und bestimmt den Durchschnittswert.

Da die Anzahl der Koordinatenpunkte zwischen dem originalen und dem rekonstruierten Objekt nicht übereinstimmen, ist der *Euklidische Abstand* zur Berechnung nicht geeignet. Daher ordnet die *Dynamische Zeitnormierung* die zusammen gehörigen Punkte zueinander und berechnet im Anschluss die Distanz zwischen diesen Zuordnungen.

Dynamische Zeitnormierung Im Gegensatz zum *Euklidischen Abstand* verbindet die *Dynamische Zeitnormierung* die Punkte zweier Listen/Kurven nicht $1:1$ sondern $1:n$ bzw. $n:1$, dabei ist das Ziel, die Punkte so zu verbinden, sodass der Abstand minimal ist. Zudem stellt der Algorithmus sicher, dass jeder Punkt eine Verbindung eingeht und das sich keine Verbindungen überkreuzen. [10]

Die Abstandsberechnung erfolgt mithilfe des Moduls *dtw_ndim* aus der Python-Bibliothek *dtaidistance*. In der Abbildung 9 ist ein Vergleich der Zuordnung einzelnen Punkte zweier Kurven durch die *Euklidische Zuordnung* und *Dynamische Zeitnormierung* zu sehen.

Das Testen und Auswerten der Parameter erfolgt durch die Datei *Bike_parameter_testing.ipynb*. In der ersten Codezelle bestimmt der Nutzer die Parameter, sodass die zweite Codezelle *Deformetrica* ausführen und im Anschluss die dritte Codezelle die Ergebnisse auswerten kann. Nach Testen und Auswerten verschiedener Parameterwerte, kann die vierte Codezelle eine *xlsx* Datei erstellen. In der Abbildung 10 sind die Informationen die diese Datei beinhaltet zu sehen. Im ersten Block stehen die aktuellen Parameter, im zweiten eine statistische Auswertung der Standardabweichungen der *Momenta* und im letzten die Abstände zwischen den originalen und rekonstruierten Bildern. Das Ziel ist den Wert der Zeile *average of distance*, den durchschnittlichen Abstand zwischen Originalen und Rekonstruktionen, zu minimieren, daher wird der Parameterwert gewählt, bei dem der durchschnittliche Abstand am geringsten ist.

3.4 Parameterauswertung

Dieser Abschnitt thematisiert das Testen und Auswählen der Parameter aus Unterunterabschnitt 2.2.3, wie in Unterabschnitt 3.3 beschrieben.
Folgende Werte wurden getestet:

- *kernel-width (template):* 10, 20, 30, 40, 50
- *kernel-width (deformation):* 10, 20, 30, 40, 50
- *noise-std:* 0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0
- *number-of-timepoints:* 7, 9, 11, 13, 15
- *attachment-type: current*, *varifold*.

Im Gegensatz zu der *Deformetrica* Anleitung und dem Beispiel mit den Schädelbildern, existiert keine Vorlage für das Motorrad. Daher dient eines der fünf Bilder als Vorlage.

Für *kernel-width (template)* werden anhand der obigen Liste die Werte $\{10, 20, 30, 40, 50\}$ getestet. Die restlichen Parameter bleiben konstant. Wie in Abbildung 11 zu sehen, ist der durchschnittliche Abstand zwischen den originalen und den rekonstruierten Objekten bei einem Parameterwert von 10 am niedrigsten. Daher wird dieser Wert für die restlichen Berechnungen verwendet.

Für *kernel-width (deformation)* werden anhand der obigen Liste die Werte $\{10, 20, 30, 40, 50\}$ getestet. Die restlichen Parameter bleiben konstant. Wie in Abbildung 12 zu sehen, ist der durchschnittliche Abstand zwischen den originalen und den rekonstruierten Objekten bei einem Parameterwert von 10 am niedrigsten. Daher wird dieser Wert für die restlichen Berechnungen verwendet.

Für *noise-std* werden anhand der obigen Liste die Werte $\{0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$ getestet. Die restlichen Parameter bleiben konstant. Wie in Abbildung 13 zu sehen, ist der durchschnittliche Abstand zwischen den originalen und den rekonstruierten Objekten bei einem Parameterwert von 0.05 am niedrigsten. Daher wird dieser Wert für die restlichen Berechnungen verwendet.

Für *number-of-timepoints* werden anhand der obigen Liste die Werte $\{7, 9, 11, 13, 15\}$ getestet. Die restlichen Parameter bleiben konstant. Wie in Abbildung 14 zu sehen, hat die Veränderung kaum Auswirkungen auf den durchschnittlichen Abstand zwischen den originalen und den rekonstruierten Objekten. Da der Abstand beim Wert 11 am niedrigsten ist, wird dieser Wert für die restlichen Berechnungen verwendet.

Für *attachment-type* werden anhand der obigen Liste die Werte $\{$*current*, *varifold*$\}$ getestet. Die restlichen Parameter bleiben konstant. Wie in Abbildung 15 zu

sehen, ist der durchschnittliche Abstand zwischen den originalen und den rekonstruierten Objekten bei dem Parameterwert *varifold* am niedrigsten. Daher wird dieser Wert für die restlichen Berechnungen verwendet.

Zum Schluss werden alle fünf Bilder als Vorlage getestet, um den besten Startpunkt für die Verformung herauszufinden. (Abbildung 16). Das beste Ergebnis erzielt die Vorlage *mask_03*.

Die Auswertung führt zu folgenden Parametern:

- *kernel-width (template):* 10
- *kernel-width (deformation):* 10
- *noise-std:* 0.05
- *number-of-timepoints:* 11
- *attachment-type: varifold.*
- *template: mask_03.vtk.*

Durch diese Parameter ist der Abstand zwischen den originalen und den rekonstruierten Objekten minimal. Daher lohnt sich an dieser Stelle ein Blick auf die Rekonstruktion (Abbildung 17). Im Hintergrund sind in Schwarz die originalen und im Vordergrund in Farbe die rekonstruierten Objekte zu sehen. Die Bilder beinhalten nur wenige schwarze Linien, da die farbigen Linien diese überdecken. Das bedeutet, dass *Deformetrica* mithilfe des Durchschnittsbilds und den *Momenta* die Objekte sehr gut rekonstruieren kann. Nur in den Bildern *mask_02*, *mask_04* und *mask_05* existieren minimale Abweichungen am Motorradständer. Zudem sind im Bild *mask_05* Abweichungen am Lenkrad zusehen. Daraus lässt sich ableiten, dass die Rekonstruktion durch diese Parameter sehr gut ist.

3.5 Visualisierung Unstimmigkeiten

Der vorherige Unterabschnitt zeigt, wie gut *Deformetrica* mithilfe der *Momenta* die Objekte rekonstruieren kann. Da an jedem Kontrollpunkt für jedes Objekt ein *Momenta* existiert, können aufgrund dieser Werte Rückschlüsse darauf gezogen werden, ob Unstimmigkeiten zwischen Annotationen vorhanden sind. Sollten die Annotationen in einem Bereich übereinstimmen, dann haben die *Momenta* einen ähnlichen Zahlenwert und somit eine geringe Standardabweichung. Sollten jedoch die Annotationen in einem Bereich abweichen, dann haben die *Momenta* unterschiedliche Werte und somit eine hohe Standardabweichung. Durch diese Abhängigkeit helfen die Standardabweichungen zum Auswerten und Darstellen der Unstimmigkeiten.

Zur Umsetzung wurde für jeden Kontrollpunkt die Standardabweichung der fünf *Momenta,* die an diesem Kontrollpunkt anhängen, berechnet. Zur Visualisierung dienen Kreise, deren Farben abhängig von der Standardabweichung sind. Zum Beziffern dieser Werte steht eine Farbskala zur Verfügung. Diese Methode mit der Farbskala hat die Absicht, dass der Betrachter dieser Visualisierung selbst entscheiden kann, ab welchen Wert er die Abweichung bzw. die Unstimmigkeit als zu groß empfindet. Es ist nicht möglich, einen Pauschalwert als Grenze für eine zu hohe Unstimmigkeit zu definieren. Zudem soll es auch möglich sein, das Programm mit anderen Daten zu verwenden, bei dem dieser Pauschalwert nicht zutreffen würde.

Die Abbildung 18 c) beinhaltet die Visualisierung der Unstimmigkeiten. Aus diesem Bild lässt sich ablesen, das im Bereich des Lenkrads sowie an der Unterseite des Motorrads Unstimmigkeiten zwischen den Annotationen existieren. Ersteres wurde bereits in Unterabschnitt 2.1 aufgrund der Abbildung 3 vermutet, Letzteres jedoch nicht. Daher lohnt sich an dieser Stelle ein Vergleich mit den einzelnen Annotationen aus Abbildung 2. Beim detaillierten Betrachten der Unterseite, fällt auf, dass alle Annotatoren die Unterseite und vor allem den Motorradständer unterschiedlich gezeichnet haben. Dasselbe gilt auch für das Lenkrad.

Anhand dieser Visualisierung lassen sich Unstimmigkeiten zwischen den Annotationen erkennen und somit eines der beiden Ziele dieses Fortgeschrittenenpraktikums erfüllen.

3.6 Ergebnis Durchschnittsobjekt

Die Abbildung 18 a) ist dafür geeignet, um herauszufinden, ob das zweite Ziel, ein Durchschnittsobjekt aus allen Annotationen zu generieren, erfolgreich war. In dieser Visualisierung ist in Rot die Vorlage, die als Startpunkt für die Berechnung gedient hat und in Blau das ausgegebene Durchschnittsobjekt, das durch eine Verformung aus der Vorlage entstanden ist. Bis auf sehr wenigen Stellen sind die beiden Objekte identisch, daraus lässt sich schließen, dass keine bemerkbare Verformung stattgefunden hat. Im Vergleich dazu ist in der Abbildung 19 die Verformung aus der *Deformetrica* Anleitung (Unterunterabschnitt 2.2.2) zu sehen. Daher stellt sich die Frage, ob die Generierung des Durchschnittsbilds fehlgeschlagen und damit das zweite Ziel nicht erfüllt wurde.

Die Parameterauswahl im Unterabschnitt 3.4 hatte das Ziel den Rekonstruktionsfehler zu minimieren. Die Rekonstruktion entsteht durch die Verformung des Durchschnittsbilds mithilfe der *Momenta,* daher resultiert aus einem minimalen Rekonstruktionsfehler nicht in allen Fällen ein gutes Durchschnittsobjekt. Es könnte sein, dass das Durchschnittsobjekt *'schlecht'* ist und der minimale Rekonstruktionsfehler sich aus den *'guten' Momenta* ergibt. Ein Beispiel für diese Vermutung ist der Ständer des Motorrads. Alle Annotationen außer *mask_03*, welches auch gleichzeitig die Vorlage ist, haben einen Motorradständer (Abbildung 2). Auf einem erfolgreichen Durchschnittsobjekt müsste auch ein Motorradständer erkennbar sein, jedoch ist dies nicht der Fall. Das lässt auch das Ziel den Rekonstruktionsfehler zu minimieren hinterfragen, weil in der *Deformetrica* Anleitung die Rekonstruktion nicht so gut ist, aber dafür das Durchschnittsbild die Daten sehr gut repräsentiert. Daher stellt sich die Frage, ob das Optimierungsziel, ein minimaler Rekonstruktionsfehler, für die Auswahl der Parameter nicht geeignet ist.

Das Durchschnittsobjekt soll, wie der Name sagt, ein Durchschnitt aus allen Objekten bilden, wenn jedoch die Objekte sehr ähnlich zueinander sind, darf es nicht verwunderlich sein, dass das Durchschnittsobjekt der Vorlage entspricht. Hier lohnt sich ein Blick auf Abbildung 3, die Annotationen sind sehr ähnlich und es existieren nur wenige globale Unstimmigkeiten, daher ist es nicht verwunderlich, das keine großen Abweichungen zwischen der Vorlage und dem Durchschnittsbild zu sehen sind. Im Vergleich dazu hatte der Beispieldatensatz aus der *Deformetrica* Anleitung sehr große Unterschiede zwischen den Objekten. Zum Validieren dieses Ziels bietet es sich an, das Programm mit einem anderen Datensatz, bei dem sich die Objekte auch global unterscheiden, zu testen. Es könnte sein, dass es *Deformetrica* nur bei einem Datensatz, deren Objekte sich unterscheiden, gelingt, ein *gutes* Durchschnittsobjekt zu generieren. Eine andere Möglichkeit wäre eine Berechnung mit einer Vorlage, dass kein Teil des Datensatzes ist bzw. sich von Ihm unterscheidet. Im Beispiel mit den Schädelbildern hat sich die Vorlage von den Daten unterschieden. Eventuell ist dies ein Hindernis bei der Generierung des Durchschnittsobjekts.

Es konnte nicht abschließend geklärt werden, ob das zweite Ziel erfüllt wurde. Es bedarf weitere Test mit anderen Datensätzen, um dieses Ziel zu validieren.

3.7 Anleitung *Bike.ipynb*

Wie bereits im Unterunterabschnitt 2.2.2 beschrieben dient die *Deformetrica* Anleitung [6] als Startpunkt dieses Fortgeschrittenenpraktikums und benötigt zum Erreichen der Ziele eine Erweiterung. Im Laufe des Fortgeschrittenenpraktikums entstand dadurch das Programm *Bike.ipynb* und beinhaltet den ganzen Prozess, angefangen vom Installieren der benötigten Python Bibliotheken und dem Vorbereiten der Eingabedaten bis hin zum Auswerten und Visualisieren der Ergebnisse. Das Ziel beim Konzipieren dieses Programms war eine Datei zu haben, die mit wenigen Änderungen auch auf andere Datensätze anwendbar ist.

Das Ausführen der ersten Codezelle lädt die benötigten Python-Bibliotheken herunter und installiert sie anschließend. Dies ist nur ein einziges Mal für ein System nötig und dauert eine Weile. Der Code innerhalb der zweiten Codezelle bereitet die Daten vor, dazu gehört das Erstellen von *VTK* Dateien für jedes Bild und das Erzeugen eigener Kontrollpunkte. Die Ausführung dieser Zelle ist für jeden Datensatz nur ein Mal nötig. Im Falle eines neuen Datensatzes müssen die Bilder aus dem Ordner *images/mask_png* ersetzt werden. Die Hintergründe zum Thema eigener Kontrollpunkte sind zu finden in Unterabschnitt 3.2. Im nächsten Schritt, wie auch in der *Deformetrica* Anleitung, visualisiert das Programm die zu *VTK* Dateien umgewandelten Daten. Der Nutzer kann in diesem Punkt überprüfen, ob die Umwandlung in den neuen Datentyp erfolgreich war.

Der Abschnitt *Run Deformetrica* führt die Berechnung durch, hierfür benötigt das Programm Parameter, die der Nutzer festlegen muss. In Unterunterabschnitt 2.2.3 und Unterabschnitt 3.4 wird erklärt, welche Parameter dem Nutzer zur Verfügung stehen und wie er diese Auswählen kann. Bei einem anderen Datensatz sind hier einige Änderungen notwendig. Die aktuellen Parameter sind für genau diesen Datensatz bestimmt. In der siebten Zeile ist ein Parameter mit dem Namen des Objekts (*object_id*), für den aktuellen Datensatz *'bike'*, bei Bedarf kann der Nutzer diesen Namen verändern. *dataset_specifications* beinhaltet Informationen über den Datensatz. Jede Datei benötigt einen Eintrag in *dataset_filenames*, dieser Eintrag besteht aus dem Namen des Objektes *object_id* und dem Pfad zu der Datei. Beispiel:
[{object_id: os.path.join(data_base, 'mask_01.vtk')}]

Bei einem anderen Datensatz muss hier *'mask_01.vtk'* durch den Namen der Datei ersetzt werden. (Wichtig: Die Endung *.vtk* nicht vergessen.) Zudem bekommt jede Datei einen Eintrag in *subject_ids*, einfachheitshalber Namen der Datei ohne Endung benutzen. In *template_specification* kann der Benutzer die Parameter *kernel_width, noise_std* und *attachment_type* nach Unterabschnitt 3.4 setzen. Der Parameter *kernel-type* bekommt den Wert *torch*, da es sich hier um kleine Objekte handelt, bei großen Objekten müsste dies *keops* sein. *filename* beinhaltet den Namen der Vorlage, wie auch bei *dataset_filenames* ist hier nur *mask_03.vtk* zu ändern. *estimator_options* benötigt keine Veränderungen. In *model_options* sind *deformation_kernel_width* und *number_of_time_points* Parameter die nach Unterabschnitt 3.4 zu setzen sind. Falls eigene Anfangskontrollpunkte erwünscht sind, siehe Unterabschnitt 3.2, ist *mask_03_initial_control_points.txt* zu ersetzen. Falls jedoch dies nicht der Fall ist, muss der komplette Eintrag mit *'initial_control_points'* gelöscht werden. (Wichtig: Die eigenen Anfangskontrollpunkte müssen aus der Vorlage aus *template_specifications* entstanden sein, daher lautet der Name für die eigenen Kontrollpunkte <Name des Templates> + '_initial_control_points.txt'. Diese Datei ist im Ordner *images/mask_initial_control_points*) zu finden. *deformetrica.estimate_deterministic_atlas* führt anhand der Eingabe die Berechnung durch, dessen Ergebnisse die letzte Codezelle auswertet und visualisiert.

Die letzte Codezelle *Plot results* beinhaltet das Laden, Auswerten und Visualisieren der Ergebnisse. Zuerst lädt das Notebook die relevanten Ein- und Ausgabedaten. Das erste Bild in der Ausgabe visualisiert einerseits die ursprüngliche Vorlage und anderseits das erzeugte Durchschnittsobjekt. Dadurch das dieses Bild beide Objekte beinhaltet, lässt sich leicht erkennen, ob und an welchen Stellen *Deformetrica* die Vorlage verformt hat. Im zweiten Bild der Ausgabe ist das Durchschnittsbild mit den ausgehenden *Momenta* zu sehen. Im Bild werden die *Momenta* durch Pfeile, die von den Kontrollpunkten ausgehen, gezeichnet. Wie bereits bekannt, repräsentiert ein *Momenta* wie ein Kontrollpunkt verschoben werden muss, um ein Objekt aus dem Durchschnittsobjekt zu rekonstruieren. Daraus folgt, dass aus jedem Kontrollpunkt für jedes Objekt ein Pfeil ausgeht. Alle gleichfarbigen Pfeile gehören zu einem Objekt. Das letzte Bild in der ersten Zeile der Ausgabe visualisiert die Unstimmigkeiten der Annotation, wie bereits in Unterabschnitt 3.5 erklärt. Die Werte der einzelnen Kontrollpunkte, Kreise in der Abbildung, lassen sich aufgrund der Farbe aus der Farbskala auslesen. Dieser Wert steht für die Standardabweichung aller *Momenta* eines Kontrollpunkts. In der zweiten Zeile sind die originalen und die aus dem Durchschnittsbild und den *Momenta* rekonstruierten Objekte zu sehen. Die Originalen sind in Schwarz gezeichnet und die Rekonstruierten in Farbe, sodass der Betrachter direkt erkennen kann, wie gut *Deformetrica* die Objekte rekonstruiert.

4. Fazit und Ausblick

Das Ziel dieses Fortgeschrittenenpraktikums war es, Unstimmigkeiten zwischen mehreren Annotationen eines Objekts zu erkennen und ein Durchschnittsobjekt aus diesen Annotationen zu generieren. Zum Erreichen der Ziele wurde statistische Formanalyse mithilfe der Software *Deformetrica* auf einem Beispieldatensatz angewendet. Ausgehend der *Deformetrica* Anleitung [6] entstand ein Programm, das durch kleine Änderungen auch auf andere Datensätze anwendbar ist. In den Ausgaben dieses Programms sind unter anderem eine Visualisierung der Unstimmigkeiten und ein Durchschnittsobjekt zu finden. Das Programm hat sehr gut die Unstimmigkeiten erkannt und visualisiert, jedoch konnte nicht endgültig geklärt werden, ob das Durchschnittsobjekt auch erfolgreich den Durchschnitt der Daten darstellt. Dafür benötigt es weitere Test mit anderen Daten, bei denen die Unterschiede größer sind und/oder die Vorlage sich von den Daten unterscheidet. Es wurde mindestens eins von zwei Zielen erreicht und gezeigt, dass statistische Formanalyse für diese Art von Problematik geeignet ist.

Literatur

[1] Sabina Pokhrel: Image Data Labelling and Annotation - Everything you need to know. towards data science. 11.03.2020. https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1 zuletzt besucht am 03.08.2021

[2] Deformetrica. Deformetrica – learn from shapes. http://www.deformetrica.org/ zuletzt besucht am 02.08.2021

[3] AramisLab: Deformetrica Wiki: 1_lddmm. https://gitlab.com/icm-institute/aramislab/deformetrica/-/wikis/1_lddmm zuletzt besucht am 01.08.2021

[4] Prof. Dr. Claude Portenier: Analysis. Kapitel 13.1. Diffeomorphismen. Philipps-Universität Marburg: Fachbereich Mathematik und Informatik. Marburg. S.346. 2006. https://www.mathematik.uni-marburg.de/~portenier/Analyse/Skript/unter-mgf.pdf zuletzt besucht am 30.07.2021

[5] Dr. Carsten Rohr: Höhere Mathematik III für Physiker Analysis 2. Kapitel 3. Koordinatentransformation. Technische Universität München: Fakultät für Physik. München. S. 18 . 2008. https://www.ph.tum.de/academics/bsc/break/2008w/fk_MA9203_03_course.pdf zuletzt besucht am 30.07.2021

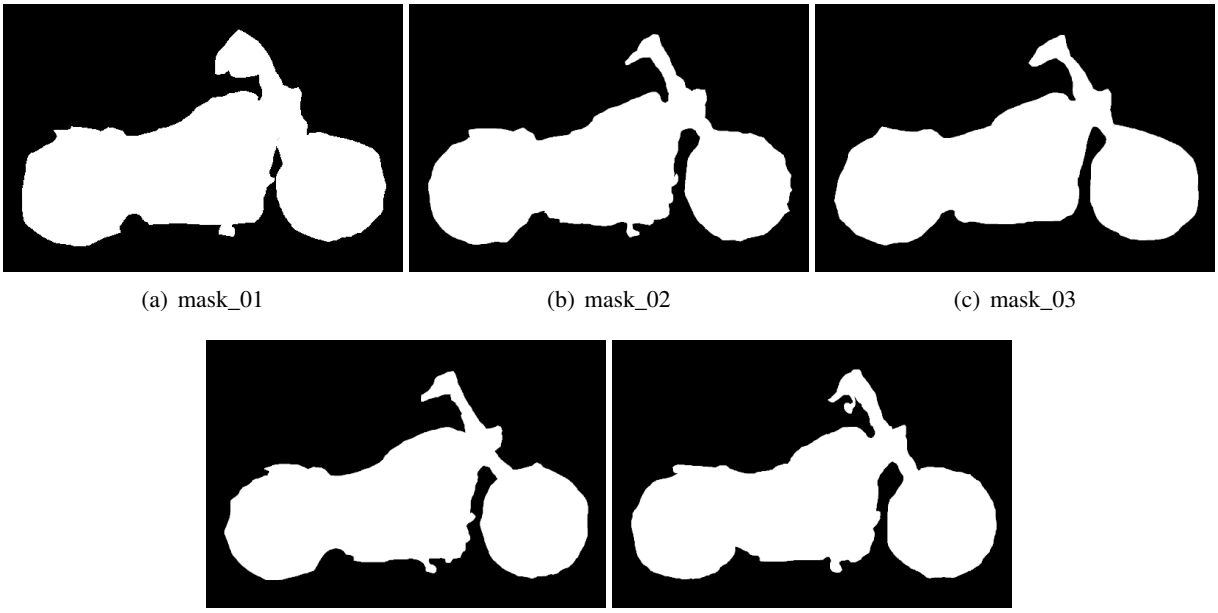
[6] Deformetrica: deformetrica_api_demo.ipynb. https://colab.research.google.com/drive/1ZYArpukrdp_SsRh-cW6PXJNaLEt1Tafr#scrollTo=E7xGFoJ-w_j3 zuletzt besucht am 03.08.2021

[7] AramisLab: Deformetrica Wiki: 3.2_model_xml_file. https://gitlab.com/icm-institute/aramislab/deformetrica/-/wikis/3_user_ual/3.2_model_xml_file zuletzt besucht am 01.08.2021

[8] AramisLab: Deformetrica Wiki: 3.3_data_set_xml_file. https://gitlab.com/icm-institute/aramislab/deformetrica/-/wikis/3_user_manual/3.3_data_set_xml_file zuletzt besucht am 01.08.2021

[9] AramisLab: Deformetrica Wiki: 3.4_optimization_parameters_xml_file. https://gitlab.com/icm-institute/aramislab/deformetrica/-/wikis/3_user_manual/3.4_optimization_parameters_xml_file zuletzt besucht am 01.08.2021

[10] Jeremy Zhang: Dynamic Time Warping. Explanation and Code Implementation. towards data science. 01.02.2020. https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd zuletzt besucht am 02.08.2021



(a) mask_01 (b) mask_02 (c) mask_03

(d) mask_04 (e) mask_05

Abbildung 2. Fünf Annotationen eines Motorrads

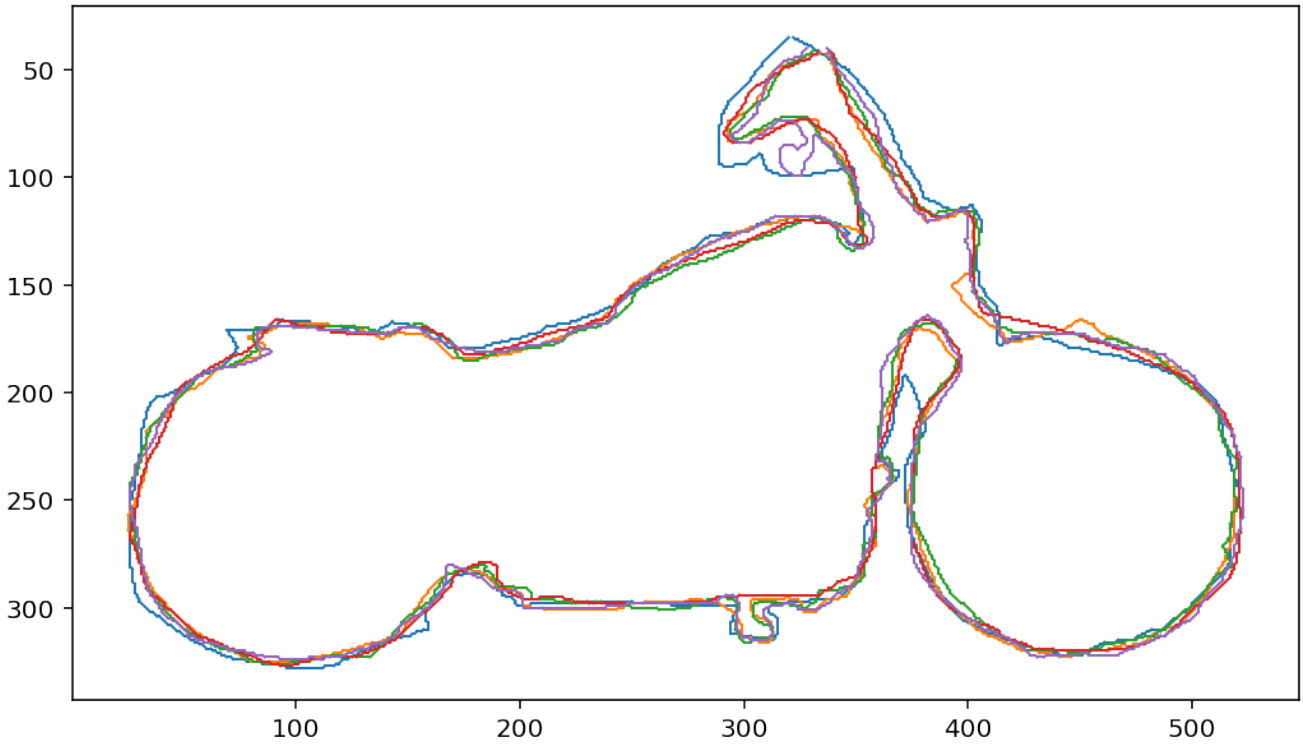
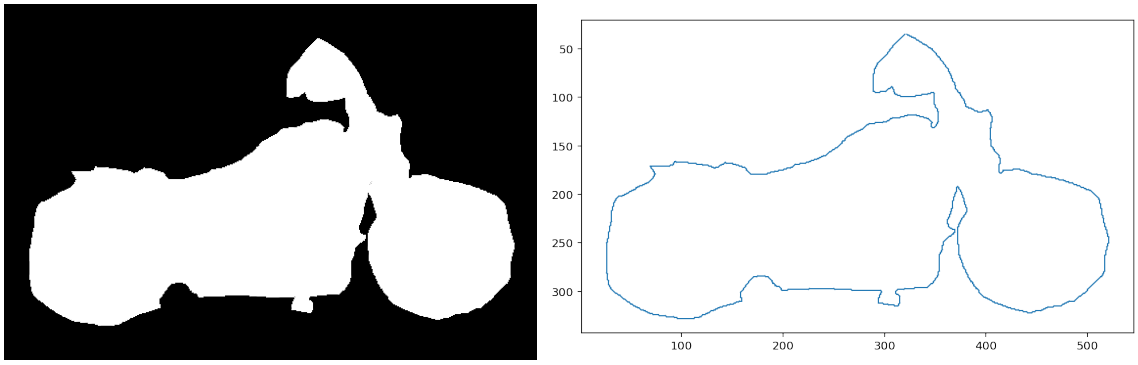


Abbildung 3. Alle fünf Annotationen in einem Bild



(a) mask_01.png (b) mask_01.vtk

Abbildung 4. Dieselbe Annotation als (a) PNG und (b) VTK Datei.

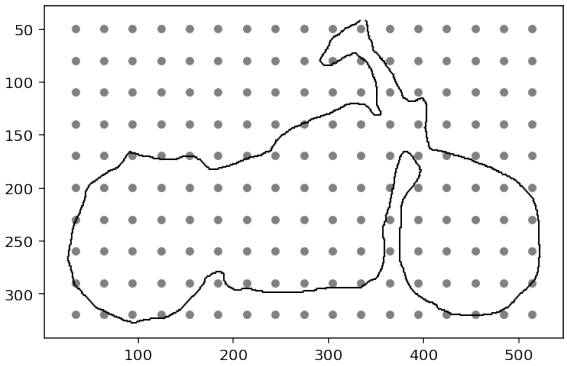


Abbildung 5. Gleich verteilte Kontrollpunkte

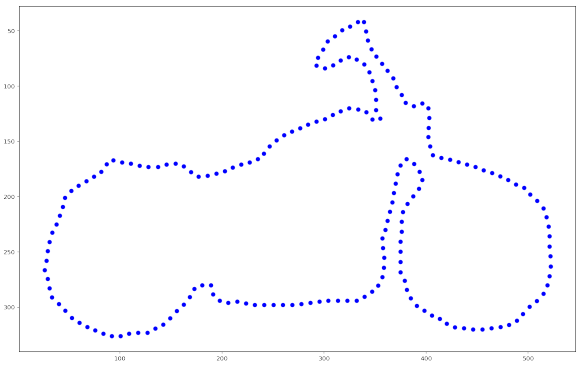


Abbildung 7. Eigene Anfangskontrollpunkte

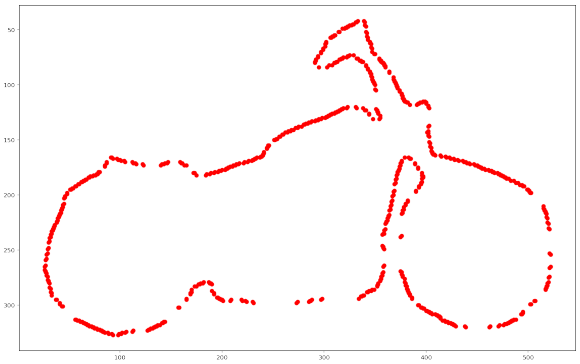


Abbildung 6. Koordinate der VTK Datei

Abbildung 8. Vergleich zwischen den Punkten der VTK Datei (rot) und den generierten Kontrollpunkten (blau)

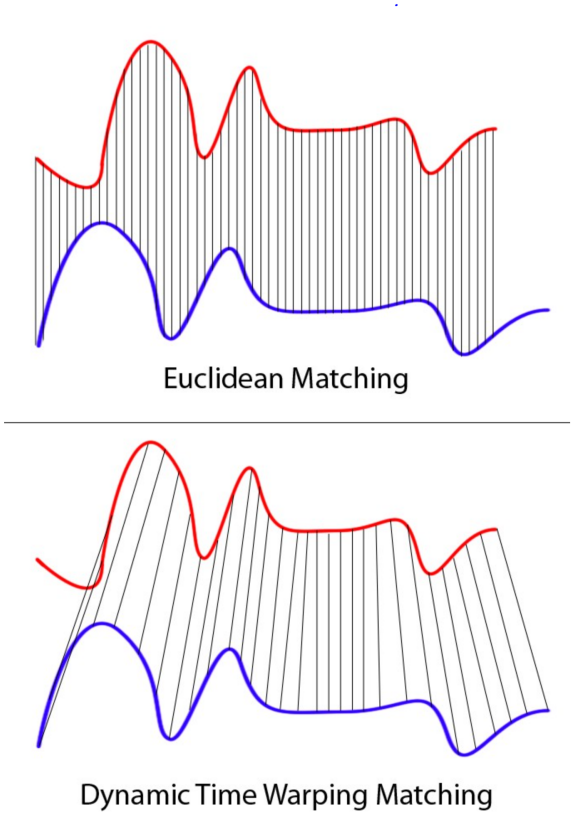


Abbildung 9. Vergleich *Euklidischen Zuordnung* und *Dynamische Zeitnormierung* [10]

parameter:
attachmentType:
kernelWidth:
deformationKernelWidth:
noiseStd:
numberOfTimePoints:
templateName:
list of momenta standard deviation:
count:
mean:
std:
min:
25%:
50%:
75%:
max:
dynamic time warping distance between raw and reconstruct:
1:
2:
3:
4:
5:
average of distance:

Abbildung 10. Informationen die in *results_parameter_testing.xlsx* gespeichert sind

parameter:					
attachmentType:	current	current	current	current	current
kernelWidth:	10	20	30	40	50
deformationKernelWidth:	20	20	20	20	20
noiseStd:	1	1	1	1	1
numberOfTimePoints:	11	11	11	11	11
templateName:	mask_03.v	mask_03.v	mask_03.v	mask_03.v	mask_03.v
list of momenta standard deviation:					
count:	200.0	200.0	200.0	200.0	200.0
mean:	0.5887185	0.4292760	0.3377715	0.3024942	0.2804953
std:	0.4719792	0.3536087	0.2909835	0.2627477	0.2503732
min:	0.0462728	0.0188599	0.0473804	0.0207457	0.0211768
25%:	0.2744388	0.2115796	0.1692132	0.1315716	0.1196857
50%:	0.4388672	0.3163522	0.2352287	0.2010590	0.1837314
75%:	0.7411880	0.5175982	0.3833091	0.3754090	0.3464344
max:	2.7612733	2.1005125	1.7286513	1.4211817	1.2678550
dynamic time warping distance between raw and reconstruct:					
1:	102.29923	124.05002	144.25552	157.64960	168.60361
2:	101.47281	104.82077	113.62129	119.11127	124.52670
3:	30.194949	27.507301	23.271955	23.593207	26.950587
4:	93.353459	98.848968	112.04720	118.19365	124.57046
5:	103.11324	103.92232	114.90253	122.24129	128.14283
average of distance:	86.086739	91.829879	101.61970	108.15780	114.55884

Abbildung 11. Auswertung verschiedene Parameterwerte für *kernel width (template)*

parameter:					
attachmentType:	current	current	current	current	current
kernelWidth:	10	10	10	10	10
deformationKernelWidth:	10	20	30	40	50
noiseStd:	1	1	1	1	1
numberOfTimePoints:	11	11	11	11	11
templateName:	mask_03.v	mask_03.v	mask_03.v	mask_03.v	mask_03.v
list of momenta standard deviation:					
count:	200.0	200.0	200.0	200.0	200.0
mean:	0.8920308	0.5887185	0.3554862	0.2550718	0.1973344
std:	0.7377772	0.4719792	0.1784162	0.1166078	0.0919372
min:	0.0971080	0.0462728	0.0908753	0.0594192	0.0447306
25%:	0.4271829	0.2744388	0.2233027	0.1603291	0.1322597
50%:	0.6582483	0.4388672	0.3070176	0.2203910	0.1731684
75%:	0.9917952	0.7411880	0.4507632	0.3290918	0.2511220
max:	3.9942436	2.7612733	0.9017329	0.5529805	0.5154665
dynamic time warping distance between raw and reconstruct:					
1:	123.76102	102.29923	150.02669	176.25316	190.07744
2:	89.395299	101.47281	118.21624	124.48724	126.60489
3:	25.987490	30.194949	28.385939	26.819854	29.833723
4:	88.990418	93.353459	109.06784	117.55716	121.71176
5:	94.153186	103.11324	122.79887	132.35744	136.34211
average of distance:	84.457482	86.086739	105.69912	115.49497	120.91398

Abbildung 12. Auswertung verschiedene Parameterwerte für *kernel width (deformation)*

parameter:								
attachmentType:	current	current	current	current	current	current	current	current
kernelWidth:	10	10	10	10	10	10	10	10
deformationKernelWidth:	10	10	10	10	10	10	10	10
noiseStd:	2.0	1.5	1.0	0.75	0.5	0.25	0.1	0.05
numberOfTimePoints:	11	11	11	11	11	11	11	11
templateName:	mask_03.v	mask_03.v	mask_03.v	mask_03.v	mask_03.v	mask_03.v	mask_03.v	mask_03.v
list of momenta standard deviation:								
count:	200.0	200.0	200.0	200.0	200.0	200.0	200.0	200.0
mean:	0.4725766	0.6295920	0.8920308	1.0226690	1.1608251	1.2281526	1.3382625	1.3884620
std:	0.3183436	0.4529012	0.7377772	0.8684289	0.9527451	0.9708307	1.0599573	1.0866101
min:	0.0765660	0.0900834	0.0971080	0.0759939	0.2068877	0.2346582	0.2031200	0.2229384
25%:	0.2671492	0.3392963	0.4271829	0.4968353	0.5722737	0.6289001	0.6787288	0.7005582
50%:	0.3866159	0.4834298	0.6582483	0.7257316	0.8205772	0.8425052	0.9216653	1.0075157
75%:	0.5742367	0.7742701	0.9917952	1.1098420	1.3314297	1.4916235	1.5831448	1.6390010
max:	2.1256309	3.1659216	3.9942436	5.0101404	5.3716807	5.3645952	5.9759148	6.2767950
dynamic time warping distance between raw and reconstruct:								
1:	162.78922	151.94822	123.76102	114.46129	103.88520	100.43586	91.815610	86.625017
2:	82.920336	97.886691	89.395299	86.851545	86.533796	88.326892	85.762450	83.361545
3:	53.473993	30.958586	25.987490	22.301854	17.738721	16.138632	16.320269	16.899364
4:	81.882102	97.259302	88.990418	85.749774	80.701792	84.169700	78.797877	76.260613
5:	98.726612	102.98769	94.153186	89.436040	87.326431	88.714592	85.814015	84.325557
average of distance:	95.958453	96.208099	84.457482	79.760101	75.237190	75.557135	71.702044	69.494419

Abbildung 13. Auswertung verschiedene Parameterwerte für *noise-std*

parameter:					
attachmentType:	current	current	current	current	current
kernelWidth:	10	10	10	10	10
deformationKernelWidth:	10	10	10	10	10
noiseStd:	0.05	0.05	0.05	0.05	0.05
numberOfTimePoints:	7	9	11	13	15
templateName:	mask_03.v	mask_03.v	mask_03.v	mask_03.v	mask_03.v
list of momenta standard deviation:					
count:	200.0	200.0	200.0	200.0	200.0
mean:	1.3120661	1.3152255	1.3884620	1.3595025	1.3603897
std:	1.0369144	1.0392318	1.0866101	1.0731497	1.0740481
min:	0.2159036	0.2153559	0.2229384	0.2032947	0.2034355
25%:	0.6708057	0.6702358	0.7005582	0.6897907	0.6902683
50%:	0.9360376	0.9363360	1.0075157	0.9570311	0.9579299
75%:	1.5995797	1.5961262	1.6390010	1.5964423	1.5956962
max:	5.7663880	5.7826410	6.2767950	6.1089553	6.1122287
dynamic time warping distance between raw and reconstruct:					
1:	94.507694	94.224384	86.625017	90.247243	90.428631
2:	86.855930	86.804174	83.361545	84.908760	84.922576
3:	16.411082	16.427411	16.899364	16.651932	16.667560
4:	79.876846	79.813905	76.260613	77.827754	77.852840
5:	87.057143	87.008113	84.325557	85.343004	85.347343
average of distance:	72.941739	72.855597	69.494419	70.995738	71.043790

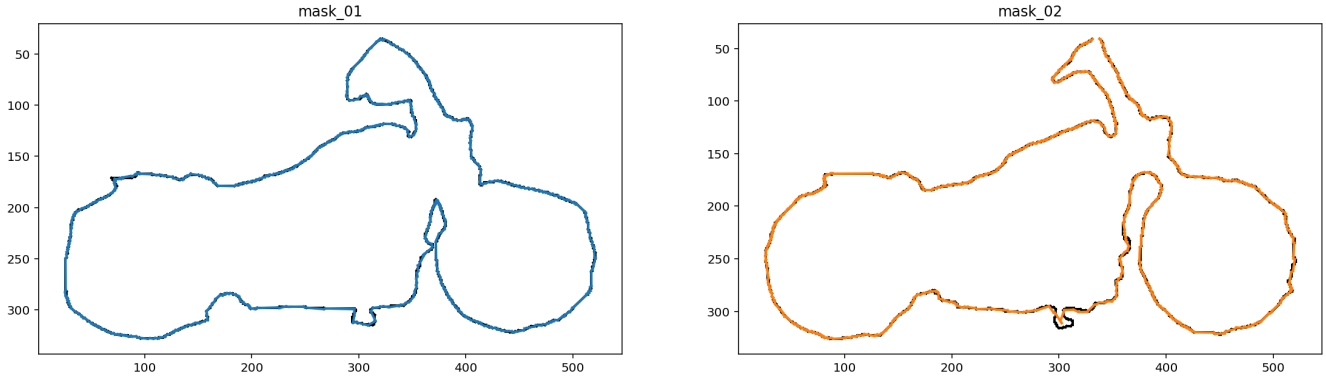
Abbildung 14. Auswertung verschiedene Parameterwerte für *number-of-timepoints*

parameter:		
attachmentType:	current	varifold
kernelWidth:	10	10
deformationKernelWidth:	10	10
noiseStd:	0.05	0.05
numberOfTimePoints:	11	11
templateName:	mask_03.v	mask_03.v
f momenta standard devia		
count:	200.0	200.0
mean:	1.3884620	1.7568189
std:	1.0866101	1.3666386
min:	0.2229384	0.1788947
25%:	0.7005582	0.8497399
50%:	1.0075157	1.3178937
75%:	1.6390010	2.0162563
max:	6.2767950	7.7777270
dynamic time warping dist		
1:	86.625017	58.995280
2:	83.361545	65.386107
3:	16.899364	19.907266
4:	76.260613	58.179109
5:	84.325557	55.409472
average of distance:	69.494419	51.575447

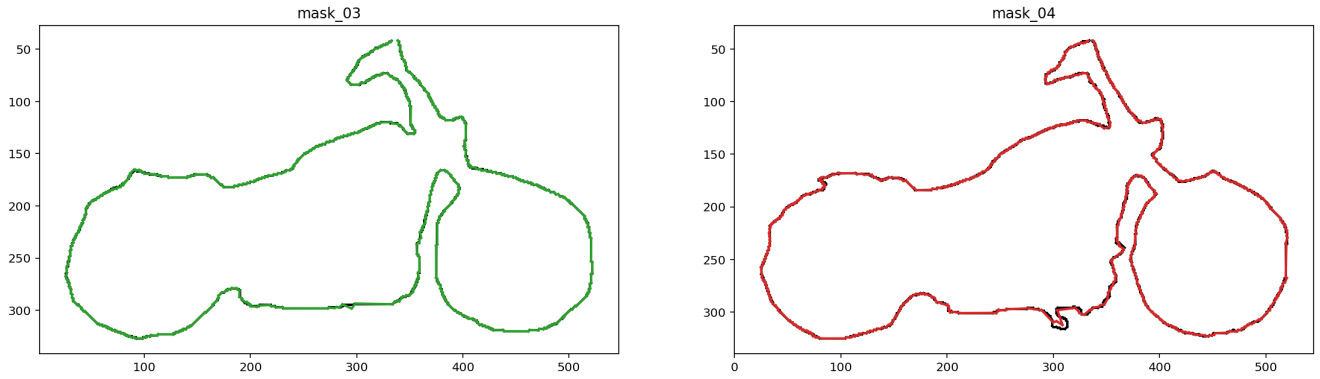
Abbildung 15. Auswertung verschiedene Parameterwerte für *attachment-type*

parameter:						
attachmentType:	varifold	varifold	varifold	varifold	varifold	
kernelWidth:	10	10	10	10	10	
deformationKernelWidth:	10	10	10	10	10	
noiseStd:	0.05	0.05	0.05	0.05	0.05	
numberOfTimePoints:	11	11	11	11	11	
templateName:	mask_01.v	mask_02.v	mask_03.v	mask_04.v	mask_05.vtk	
list of momenta standard deviation:						
count:	200.0	200.0	200.0	200.0	200.0	
mean:	1.3926833	1.4373899	1.7568189	1.4338833	1.5294790284563924	
std:	1.0314726	1.1170845	1.3666386	1.0149048	1.0835121051448278	
min:	0.2648978	0.3211210	0.1788947	0.3236543	0.2601479275466791	
25%:	0.7600476	0.6902016	0.8497399	0.7662460	0.8182787769669063	
50%:	1.0277075	1.1030417	1.3178937	1.1448422	1.1391222991510368	
75%:	1.7724981	1.6429802	2.0162563	1.6667852	1.9703321066752162	
max:	5.3305547	6.4314076	7.7777270	6.0562791	7.040165472553342	
dynamic time warping distance between raw and reconstruct:						
1:	20.272240	72.641659	58.995280	65.216252	62.81028922143926	
2:	71.625982	19.795893	65.386107	60.094306	53.00087159658319	
3:	94.102471	93.300296	19.907266	95.304080	85.34230186030939	
4:	69.102591	56.119298	58.179109	16.222606	59.64942912310182	
5:	68.151917	62.382438	55.409472	65.948577	18.23559810113121	
average of distance:	64.651040	60.847917	51.575447	60.557164	55.80769798051297	

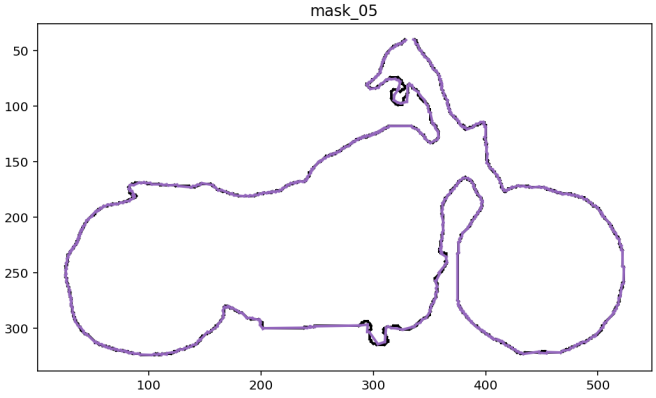
Abbildung 16. Auswertung verschiedene Vorlagen



(a)

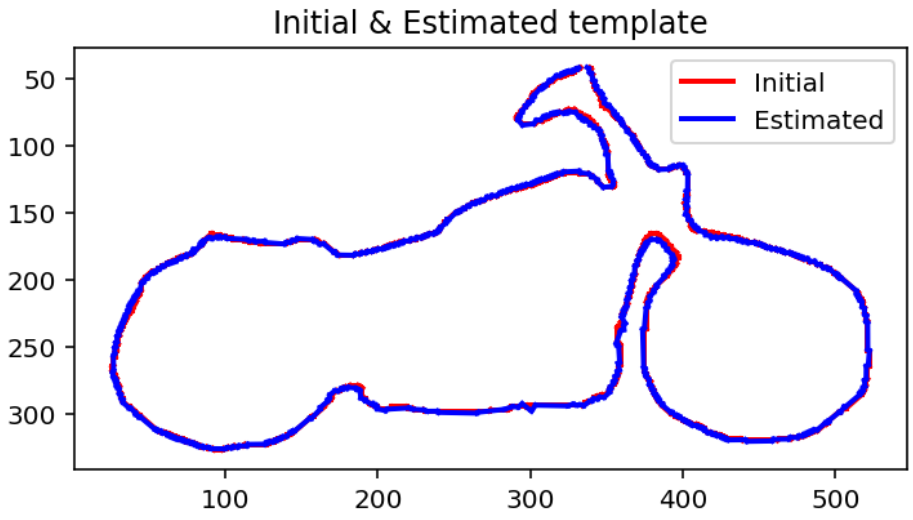


(b)

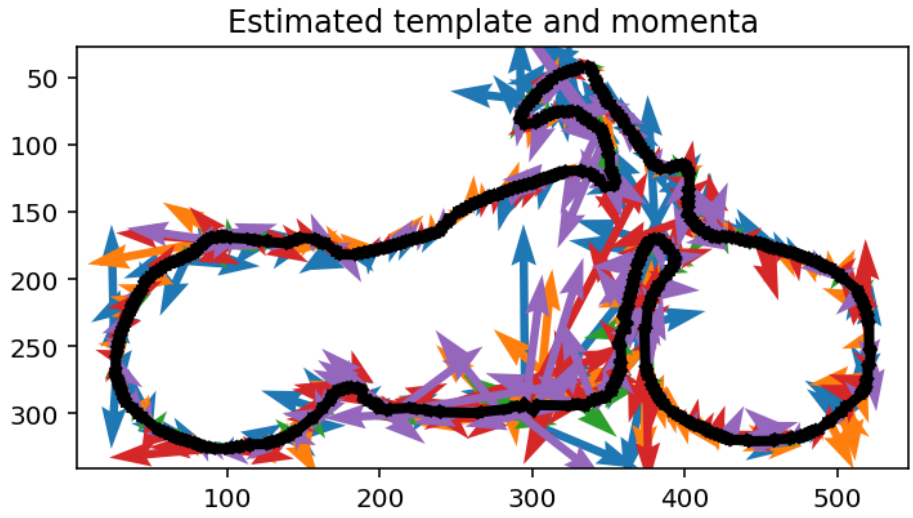


(c)

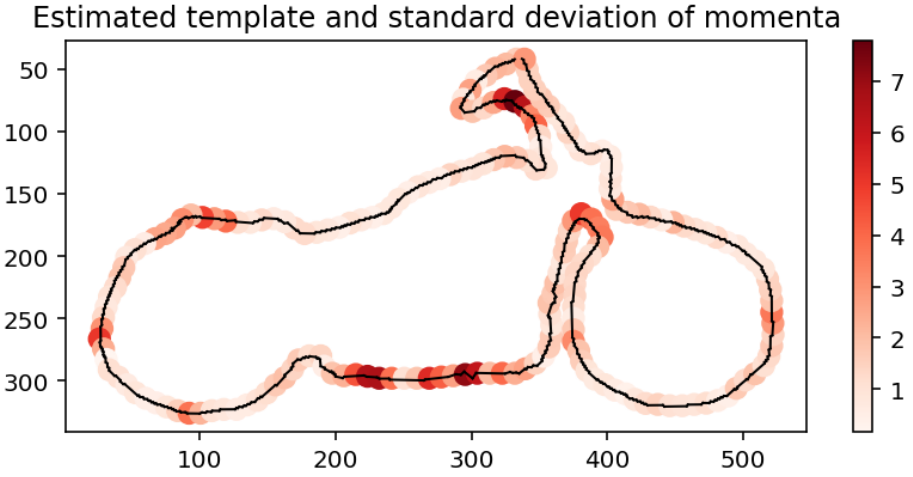
Abbildung 17. Vergleich zwischen den originalen (Schwarz) und den rekonstruierten (Farbe) Objekten



(a)



(b)



(c)

Abbildung 18. Ergebnisse

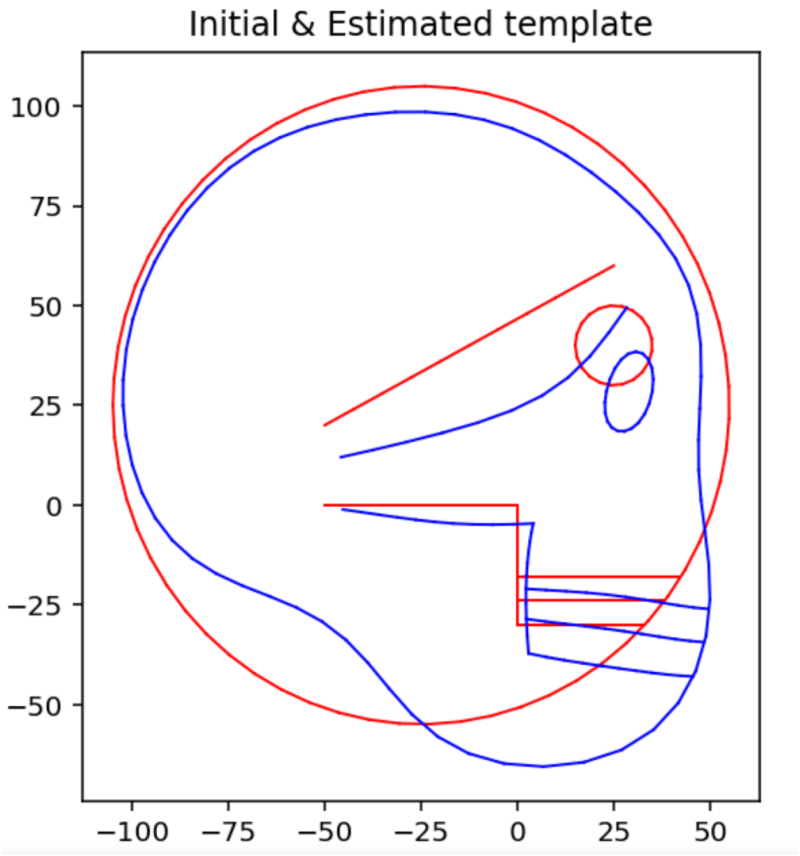


Abbildung 19. Vorlage(rot) und Durchschnittsobjekt(blau) aus [6]