

**BACHELORTHESIS**  
Tu Nguyen Anh

# ComRec: Die Skelettextraktion eines Charakters aus einem Comic Bild

---

**FAKULTÄT TECHNIK UND INFORMATIK**  
Department Informatik

Faculty of Computer Science and Engineering  
Department Computer Science

Tu Nguyen Anh

# ComRec: Die Skelettextraktion eines Charakters aus einem Comic Bild

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung  
im Studiengang Bachelor of Science Angewandte Informatik  
am Department Informatik  
der Fakultät Technik und Informatik  
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Philipp Jenke  
Zweitgutachter: Prof. Dr. Tim Tiedemann

Eingereicht am: 16. März 2020

**Tu Nguyen Anh**

**Thema der Arbeit**

ComRec: Die Skelettextraktion eines Charakters aus einem Comic Bild

**Stichworte**

Comic, Rekonstruktion, Skelett

**abstract** - Diese Arbeit ist der erste Teil einer Kollaboration, in dem Informationen einer Figur aus einem Comicbild erarbeitet wird, um diese anschließend auf ein vorhandenes 3-dimensionales Modells der Figur zu übertragen. Ein bearbeitetes Comicbild der Größe  $n \times m$  besteht aus  $RGB\alpha$ -Pixeln. Werden die Pixeln einzeln untersucht, dann geben sie wenig Ausschluss auf die Positionen der Knochen und Gelenke des Skeletts. Aus diesem Grund werden Filter und Transformationen auf das Bild angewandt um den Pixeln einen Kontext zu geben, um daraus Schlussfolgerungen ziehen zu können.

**Title of Thesis**

ComRec: The extraction of a skeleton of a character from a comic panel

**Keywords**

comic, reconstruction, skeleton

**abstract** - This work is the first part of a collaboration, which acquires figure information from a comic panel and applies them on an existing 3-dimensional model of that figure. An edited comic panel in the size of  $n \times m$  is composed of  $RGB\alpha$  pixels. The separate examination of the pixels doesn't indicate where the locations of the bones and joints of the skeleton are. Therefore filters and transformations are used on the panel to put a significant meaning onto the pixel, so that conclusions can be made.

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>vi</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Struktur der Arbeit . . . . .	2
<b>2 Stand der Technik</b>	<b>3</b>
<b>3 Grundlagen</b>	<b>6</b>
3.1 Begriffsdefinitionen . . . . .	6
3.1.1 Silhouette . . . . .	6
3.1.2 Konturlinie . . . . .	6
3.1.3 Skelett . . . . .	7
3.1.4 8-Nachbarn . . . . .	8
3.1.5 Vektorgrafik . . . . .	8
3.2 Vektorrechnung . . . . .	8
3.2.1 Orthogonalität . . . . .	9
3.2.2 Länge eines Vektors . . . . .	9
3.2.3 Winkel zwischen zwei Vektoren . . . . .	9
3.2.4 Entfernung eines Punktes zu einer Geraden . . . . .	9
<b>4 Algorithmen</b>	<b>11</b>
4.1 Voraussetzungen an das Originalbild . . . . .	11
4.2 Filterung . . . . .	12
4.2.1 Silhouette und geschlossene Konturlinien . . . . .	13
4.2.2 Innere Linien . . . . .	13
4.2.3 Verdünnung . . . . .	14
4.2.4 Das grobe Skelett . . . . .	17
4.3 Transformation . . . . .	17
4.3.1 Die Erweiterung der geschlossenen Konturlinien . . . . .	17

4.3.2	Euclidische Distanz . . . . .	21
4.3.3	Sobel . . . . .	23
4.4	Skelettisierung . . . . .	24
4.4.1	Ermittlung der Brust und des Beckenmittelpunktes . . . . .	24
4.4.2	Ermittlung der Anbindungspunkte . . . . .	25
4.4.3	Ermittlung der Körperteile . . . . .	27
<b>5</b>	<b>Konzept</b>	<b>29</b>
5.1	Anforderungen . . . . .	29
5.2	Architektur . . . . .	30
5.3	Schnittstellen . . . . .	31
5.4	MATLAB . . . . .	32
<b>6</b>	<b>Implementation</b>	<b>33</b>
6.1	Datentypen . . . . .	33
6.1.1	BaseDataSet . . . . .	33
6.1.2	ImageTransformation . . . . .	34
6.1.3	Vectorgraphic . . . . .	39
6.1.4	Splines . . . . .	40
6.2	Umsetzung der Algorithmen . . . . .	41
6.2.1	BodylinesT . . . . .	41
6.2.2	JointEstimation . . . . .	44
6.3	ImageConverter . . . . .	48
6.4	Qualitätssicherung . . . . .	49
<b>7</b>	<b>Evaluation</b>	<b>51</b>
	<b>Literaturverzeichnis</b>	<b>53</b>
	<b>Selbstständigkeitserklärung</b>	<b>56</b>

# Abbildungsverzeichnis

3.1	Skelett mit den Gelenken und Knochenverbindungen . . . . .	7
3.2	Pixel mit 8-Nachbarn . . . . .	8
3.3	Pixel mit 8-Nachbarn am Rand . . . . .	8
4.1	Verdünnungsfilter für zwei und drei Nachbarn . . . . .	16
4.2	Kosinuswert zweier Winkel . . . . .	18
4.3	Kosinuswerte und $G_W(\text{rot})$ in einem Graphen . . . . .	19
4.4	Kosinuswerte und $G_W(\text{rot})$ in einem Graphen im Bereich $x = [3500, 3750]$ . . . . .	19
4.5	Silhouette mit Wirbelsäule (rot), horizontale (gelb) und euclidische Di- stanzen (grün) . . . . .	21
4.6	Pixel (grün) und die euclidische Distanz (blau) . . . . .	22
4.7	Pixel (rot), Nachbarpixel (grün) und die euclidische Distanzen (blau) . . . . .	22
4.8	Pixel (rot) und Nachbar (grün) und die Distanz (blau) . . . . .	22
4.9	euclidische Distanz auf der Grundlage der geschlossenen Konturlinien . . . . .	24
4.10	Rumpf als Rechteck . . . . .	25
4.11	euclidische Distanzen um den leeren Brustbereich . . . . .	26
4.12	euclidische Distanzen um den leeren Beckenbereich . . . . .	26
5.1	UML-Diagramm . . . . .	30
6.1	Beispiel 1 . . . . .	37
6.2	Beispiel 2 . . . . .	37
6.3	Beispiel 3 . . . . .	37
6.4	Vektorgrafik mit der Distanz $\Delta d = 0.5$ . . . . .	40
6.5	Vektorgrafik mit der Distanz $\Delta d = 2$ . . . . .	40
6.6	Hinzufügen eines EvaluationObject zu einer Liste . . . . .	42
6.7	Elemente zur Berechnung eines Winkels . . . . .	43
6.8	Elemente zur Ermittlung des maximalen Winkels . . . . .	43
6.9	Rotation der Listenelemente . . . . .	44

6.10	Toleranzfläche an einer Tangente . . . . .	44
6.11	Ausschnitt der euklidischen Distanz mit der einfachen Konturlinie . . . . .	46
6.12	Trennung der Kreisfläche $A_k$ durch die einfache Konturlinie . . . . .	46
6.13	fehlerhafte Einordnung der Teilflächen in die Abbildungen . . . . .	46
6.14	korrekte Einordnung der Teilflächen in die Abbildungen . . . . .	47
6.15	8-Nachbarn ohne Vorgängerrotation . . . . .	47
6.16	8-Nachbarn mit Vorgängerrotation . . . . .	47
7.1	extrahiertes Skelett . . . . .	51

# 1 Einleitung

In der Computergrafik wird der Erfassung der Bewegung und der Pose eines Objektes stetig mehr Aufmerksamkeit geschenkt. Im Hinblick der steigenden Anzahl von Anwendungen in der virtuellen Realität oder der Robotik überrascht diese Entwicklung nicht. Folglich wachsen die Anforderungen bezüglich der Genauigkeit und Schnelligkeit der Systeme, denn diese gehören bereits heute zum alltäglichen Gebrauch. Das autonome Fahren muss zum Beispiel im Straßenverkehr blitzschnell die Bewegung der Fußgänger ermitteln, um abzuschätzen, ob die Personen und das Auto zusammenstoßen werden und kann so dementsprechend reagieren.

Damit die Systeme funktionieren können, müssen eine Menge Informationen aus der Umgebung erhoben werden. Dies lässt sich mit einer Vielzahl an Sensorik und Kameras erreichen, wobei das Bild ein beliebtes Medium ist. Von Vorteil ist die große Datenmenge, womit auf viele verschiedene Sensoren verzichtet werden kann und folglich kann die Synchronisation der verschiedenen Quellen vermieden werden. Mehr noch bietet die Kamera durch das Smartphone, welches in fast jedem Haushalt vorhanden ist, eine weitreichende Unterstützung und Testbarkeit durch den Endnutzer.

Eine simple Posenaufzeichnung der Anwendung für den Verbraucher ist mit der zusammenhängenden Technik verbunden. Dabei wird vor allem zwischen der markerlosen und markerbasierten Vorgehensweise unterschieden. Bei der markerbasierten Extraktion müssen Marker definiert werden, die möglichst nicht natürlich in der Umwelt vorkommen. Dadurch wird die sofortige Nutzung des Systems schwerfällig und eine Vorbereitung des Objektes notwendig. Aus diesem Grund geht die Tendenz in Richtung der markerlosen Technik, denn sie lässt sich mithilfe der Kamera sofort starten.

Nun verschiebt sich die Schwierigkeit auf die Bestimmung der Bildsegmente und ihre Klassifizierung. Die simpelste Form der Segmentierung ist die Silhouette, welche die einfache Abtrennung eines Objektes vom Hintergrund ist. 0 und 1 repräsentieren die zwei Optionen und ersetzen den ursprünglichen Pixelwert. Damit bietet sich die Silhouette



als ein guter Ausgangspunkt für die Extraktion der Pose an. Denn obwohl der Datenreichtum zu Beginn meist bevorzugt wird, sind sie für einzelne Prozesse ein zusätzliches Laster. Es führt oft zu redundanten und intensiven Berechnungen, wenn die Filterungen nicht vorgenommen werden. Solche verringerten Informationsmengen bieten dem System ein weites Spektrum isolierter Bildbestandteile.

Dabei ist die Segmentierung lediglich die Vorbereitung, denn der Skelettisierungsalgorithmus ist der eigentliche Kern der Posenextraktion. Das resultierende Skelettmodell ist ideal für die Beschreibung der Pose, denn die komplexe Oberflächenstruktur wird nicht gespeichert, trotzdem geben die Knochen und Gelenke ausreichend Informationen her. Außerdem ist die Deformation bei nachträglichen Optimierungen einfacher, weil jedes Element die Bewegung auf den gesamten Körper überträgt.

### 1.1 Struktur der Arbeit

Die Gliederung der Arbeit in die folgenden 6 Kapiteln soll das schrittweise Heranführen an den Inhalt ermöglichen. Demzufolge wird in Kapitel 2 mit der Auflistung relevanter wissenschaftlicher Schriften begonnen. Dies verschafft einen Überblick über die existierenden Algorithmen und ihre Vor- und Nachteile. Unmittelbar danach folgt die Vermittlung der Begriffsdefinitionen und Formeln der Vektorrechnung in Kapitel 3. Kapitel 4 stellt die Algorithmen dieser Arbeit vor und erläutert den Zusammenhang zur Skelettextraktion. Anschließend wird in Kapitel 5 die Grundstrukturen für die Implementation, welche ausführlich in Kapitel 6 erarbeitet wird, festgelegt. Abschließend werden die Ergebnisse und Aussichten der Entwicklung in Kapitel 7 zusammengefasst.

## 2 Stand der Technik

In der Computergrafik ist die Skelettisierung ein altbewährtes Problem, die bereits viele Methoden hervorgebracht hat. Es kann zwischen den drei Kategorien „intelligente Systeme“, „iteratives Verdünnen“, „Distanzabbildung“ unterschieden werden.

Die aktuell häufigsten verwendeten Verfahren gehören zu den intelligenten Systemen. Sie erfordern vielfältig konfigurierte Datensätze, die sich anhand der Pose und der Kameraposition unterscheiden. Die Datensätze werden genutzt, um durch Lernmechanismen im Entwicklungsprozess eine höhere Verlässlichkeit zu erzielen. Unter anderem gehört der Markov-Chain-Monte-Carlo Algorithmus dazu[8]. Hierbei werden Skelette aus der Datenbank genutzt, um sie mit dem Input nach Übereinstimmungen zu vergleichen. Die resultierenden Ergebnisse sind Wahrscheinlichkeitswerte der möglichen Positionen einzelner Gelenkpunkte. Zusätzlich grenzen Bildinformationen, wie beispielsweise durch die Ermittlung des Gesichtes, die Möglichkeiten ein[9]. Den gleichen Ansatz verfolgt ebenfalls [7], jedoch wird ein Kantenmodell anstelle der Silhouette verwendet. Das Kantenmodell wird durch einen Detektor aus den Körpermerkmalen ermittelt und hat den Vorteil, dass die Schichten der Bildtiefe für den weiteren Verlauf erhalten bleiben.

Ein weiteres Mittel, um Körpermerkmale zu bestimmen, ist die Regression[1][16]. Die Regression ist ein Analyseverfahren, das Beziehungen zwischen verschiedenen Objekten herstellen und diese Abhängigkeiten bewerten soll. So können die Pixel einer Silhouette auf die Dichte ihres Vorkommens entlang einer Koordinatenachse aufgezeichnet werden und durch Vergleiche zu anderen Bildern oder Datenbanken bewertet werden. Wird an einer Achsenposition ein Maximum gefunden, kann es darauf hindeuten, dass sich dort ein wichtiges Körpergelenk befindet. Neben den Vergleichsdaten ist ein grundlegendes Verständnis für den menschlichen Körper erforderlich, um die Eigenschaften zu filtern, die für eine Bewertung sinnvoll sind.

Die Skelettisierung mit dem iterativen Verdünnen basiert auf der Idee, dass sich die Gelenke und die meisten Stützknochen in einer zentralen Lage befinden. Außerdem wird

oft mit einem vereinfachten Skelettmodell gearbeitet, die dieselben Funktionen abdecken können. [17] und [4] werden zu den Verfahren gezählt, die bei jedem Durchlauf den äußersten Rand des Objektes entfernt, bis nur noch ein 1-Pixel weites Objekt übrig bleibt. Es werden dabei Metriken verwendet, die dabei Aufschluss geben, ob eine weitere innenliegende Schicht existiert oder ob das Skelett schon erreicht wurde. Allerdings prüft der Algorithmus jedes Mal alle verbleibenden Pixel und erhöht dadurch die Zeitkomplexität. Obendrein ist das iterative Verdünnen sehr fehleranfällig. Das 3x3 Raster der Metriken schränkt die Sicht lediglich auf die unmittelbaren Nachbarn ein und ignoriert somit andere Teile des Objektes. Kleinste Unregelmäßigkeiten und Rauschen im Bild verfälschen das Ergebnis und kann das Skelett für anschließende Prozesse unbrauchbar machen.

Auf der gleichen Grundidee ist die Distanzabbildung aufgebaut, bei dem das Bewertungssystem die kürzeste Entfernung der Pixel zu dem Rand berechnet [12][3]. Sie zeigen, wie die Zeitkomplexität verringert werden kann, indem die Informationen der Nachbarn verwendet werden, anstatt die Entfernungen zu allen Randpixeln ermitteln und vergleichen zu müssen. Der Wert kann als Radius für die aktuelle Suche verwendet werden, um den Bereich einzugrenzen. [3] verwendet ein Datenmodell, das die Abbildung in Zeilen unterteilt, die jeweils zwei Markierung für eine Distanz haben können.

Da die 3D Darstellung in der Computergrafik immer mehr an Bedeutung gewinnt, müssen die Algorithmen, die für die 2D Welt existieren, in diese übertragen werden. Um die Importe mit jeder weiteren Dimension überflüssig zu machen, entwickelt [6] eine allgemeine Methode der Entfernungstransformation, die für k-dimensionale Bilder gelten soll. Die Euclidische Metrik wird für die Berechnung der Distanz zu sogenannten „Feature Pixel“ verwendet.

Die Ergebnisse der Distanzabbildungen geben ein Ergebnis aus, das einem Skelett nahe kommt, aber noch nicht dieselben Eigenschaften hat. Es dient einem Lebewesen als Stütz- und Bewegungsapparat, deswegen muss es Elemente besitzen, an denen Translationen oder Rotationen durchgeführt werden und diese auf die restlichen Körperteile auswirken. [14] nimmt sich die Abbildungsdaten und sagt anhand der Werte voraus, an welchen Orten sich bestimmte Gelenke befinden können. Statt einem starren Skelett, das aus vielen Pixeln besteht, die eine schwer nachvollziehbare Beziehung zu einander haben, wird durch eine Repräsentation mit Gelenken eine Bewegungsfreiheit geschenkt.

Auch wenn diese drei Ansätze sehr unterschiedlich sind, können sie durch Ergebnisse aus verschiedenen Filtertechniken bereichert und somit präzisere Resultate erzielt werden.

[2] verdeutlicht die generelle Problematik der Skelettisierung, denn ein einzelnes Bild kann unmöglich die gesamte Pose eines Objektes einfangen. Die Extraktion erfolgt ausschließlich auf Grundlage des Vordergrundes, also richtet sie sich vor allem nach der Haltung und der Kameraposition. Folglich sind Knochen und Gelenke nur aus den sichtbaren Bereichen entnehmbar, aber falls Merkmale verdeckt werden, sind bloß Schlussfolgerungen oder willkürliche Annahmen möglich.

Auch wenn viele Informationen durch die 2 Dimensionalität verloren gehen, können unter anderem durch Linien auf Züge und Formen geschlossen werden[5]. Das wird besonders wichtig, wenn es zu einer Verdeckung von Körperteilen kommt und wegen den Körperrissen weiterhin sichtbar bleiben. [11] und [15] sind Beispiele, wie aus Pixelgrafiken eine Vektorgrafik erstellt werden kann, damit eine Kalkulierbarkeit des Linienverlaufs für das System ermöglicht wird.

Außerdem wird in der Bildverarbeitung häufig eine gewisse Ungenauigkeit in Kauf genommen, um das Erlernen größerer Themengebiete zu umgehen. Genauso ist es in den meisten vorgestellten Algorithmen mit den Grundkenntnissen über den inneren Aufbau komplexer Körper und vernachlässigt hilfreiche Beziehungen. Werden jedoch Heuristiken über die Verbindungen der Gelenke hinzugefügt[10], so können Annahmen gemacht werden, welche Konstellationen die verschiedenen Merkmale einnehmen können. Ferner ermöglicht dieses Wissen eine Vereinfachung der Skelettisierung, indem zuallererst charakteristische Gliedmaßen bestimmt und die nachfolgenden Punkte hergeleitet werden.

Die Reproduktion aus einem Skeletts lässt sich unter Zuhilfenahme eines Modells in der neutralen Haltung bewerkstelligen.[13]

# 3 Grundlagen

## 3.1 Begriffsdefinitionen

### 3.1.1 Silhouette

Die Silhouette ist neben dem Hintergrund und der Konturlinien (vgl. Kapitel 3.1.2) eine der drei grundlegenden Bestandteile der verwendeten Bilder, welche hier auch als Zustände bezeichnet werden. Die Besonderheiten der genannten Ausschnitte sind, dass sie eindeutig voneinander differenziert werden können. Die Silhouette ist in einer binären Form dargestellt und verkörpert den wesentlichen Teil der Figur. Die einfarbige Abbildung sieht aus wie der Schatten der Figur, wodurch es natürlich erscheint, die schwarze Farbe für inbegriffene Bildpunkte zu verwenden.

### 3.1.2 Konturlinie

Konturlinien, oder auch Isolinien genannt, dienen im Allgemeinen dazu, Punkte mit derselben Wertigkeit zu verbinden. So werden zum Beispiel Höhenlinien auf topografischen Landkarten gekennzeichnet. In dieser Arbeit sind die Konturlinien ein Teil der Figur und beherbergen noch zwei weitere Funktionen. Aus diesem Grund werden zwischen den geschlossenen und den einfachen Konturlinien unterschieden.

Eine geschlossene Konturlinie bildet eine Grenze zwischen der Silhouette und dem Hintergrund. Dabei umrundet sie vollständig eine Fläche, deren Pixel nur einen gemeinsamen Zustand einnehmen. Befindet sich allerdings eine weitere geschlossene Konturlinie in dem Bereich, bedeutet das, dass dort ein Zustandswechsel hinter der Linie liegt.

Die einfache Konturlinie hebt wiederum einzelne Körperteile innerhalb der Figur voneinander hervor, damit sie bei einer gleichfarbigen Gestaltung auseinander gehalten werden

können. Demzufolge führt die Linie in den Körper hinein und verläuft nicht als Grenze an der Silhouette. Dennoch können die beiden Enden zwei verschiedene geschlossene Konturlinien verbinden oder an einer angebunden sein und innerhalb der Figur enden.

#### 3.1.3 Skelett

Das Skelett und die Hülle sind zwei Elemente, aus denen sich eine Figur zusammensetzt. Die Hülle wird ausschließlich als das optische Erscheinungsbild des Körpers verwendet, welches über das Skelett gezogen wird. Das Skelett hingegen hat die Aufgabe, eine Pose für den Körper vorzugeben. Für diese Funktion benötigt es ein Gerüst aus einer Menge Knotenpunkte und Verbindungslinien. Angesichts der menschlichen Comicfiguren, die als Untersuchungsobjekte für diese Arbeit ausgewählt werden, kommen die Begriffe wie Gelenke und Knochen in Verwendung.

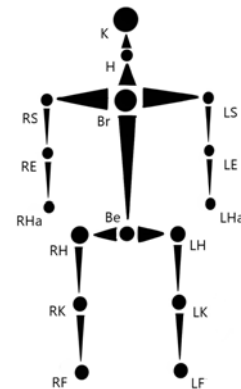


Abbildung 3.1:  
Skelett mit den  
Gelenken und  
Knochenverbindungen

Die hohe Komplexität des biologischen Skeletts ist für die Bearbeitung ungünstig, denn die Zahl der Knochen und Gelenke liegt über 200. Dabei kommen immer wieder Stellen wie die Ferse vor, an dem die Konzentration kleinerer Knochen höher als im restlichen Körper ist. Die Lokalisation der einzelnen Bestandteile erschwert sich je detailloser das Bild ist. Zudem handelt es sich bei dem Körper um ein 3D Objekt, welches auf einem 2D Bild gespeichert ist. Der daraus resultierende Verlust der Tiefeninformationen sind nicht ohne weitere Abbildungen zu ermitteln, worauf eine klare Unterscheidung von überlappenden Knochen und Gelenken mühsam ist.

Die perfekte Bestimmung ist mit diesen Rahmenbedingungen unmöglich, deshalb ist eine Reduzierung des Skeletts notwendig. Die Vereinfachung zieht jedoch eine Ungenauigkeit, die bewusst in Kauf genommen wird, mit sich. Dabei handelt es sich neben dem tatsächlichen Abbau der Knochen- und Gelenkmenge auch um eine Repositionierung. Sie werden an einen zentralen Ort verschoben, an dem sie eine simple Konstruktion fördern, statt eine Stützfunktion einzunehmen.

Die Abkürzungen und die Anordnung der Knotenpunkte und Verbindungslinien können in der Abbildung 3.1 eingesehen werden.

### 3.1.4 8-Nachbarn

Pixelinformationen sind für Filter oder ähnliche Bildmanipulationstechniken unumgänglich. Neben den eigentlichen Positionen können weitere Pixel für die Anwendung dazugezogen werden. Zu diesen gehören die sogenannten 8-Nachbarn, welche aus den direkten Nachbarn bestehen. Die Umrundung der angrenzenden Pixel um die aktuelle Position erfolgen in einer bestimmten Reihenfolge (vgl. Abbildung 3.2). Sollten sich die 8-Nachbarn außerhalb des Bildbereiches befinden, so wird ein imaginäres Padding von 1 Pixel Breite hinzugefügt. Bei den Betrachtungen werden sie als Teil des Hintergrundes angesehen und enthalten dementsprechend dieselben Daten.

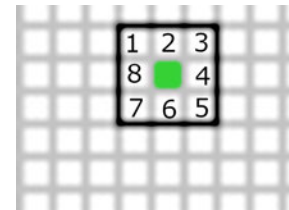


Abbildung 3.2:  
Pixel mit 8-Nachbarn

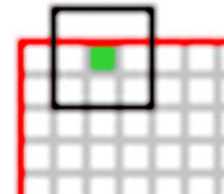


Abbildung 3.3:  
Pixel mit 8-Nachbarn  
am Rand

### 3.1.5 Vektorgrafik

Die Vektorgrafik ist eine Darstellungsart, bei der das Bild aus vielen verschiedene Objektbeschreibungen aufgebaut ist. Ein Kreis kann beispielsweise aus einem Mittelpunkt und einem Radius konstruiert werden, wogegen in der Pixelgrafik alle Positionen des Kreisumfangs bekannt sein müssen. Daher lassen sie sich in deutlich kleineren Dateien abspeichern und beliebig groß skalieren. Neben den einfachen grafischen Formen sind auch komplexere Kurvenbeschreibungen wie den Splines möglich. Deswegen befinden sich in den Bilddaten nur einzelne Formeln und Positionswerte, dessen Strukturen bei einem Aufruf des Bildes erneut berechnet werden müssen.

## 3.2 Vektorrechnung

Alle Bilder und Transformationen werden in dieser Arbeit in einem 2-dimensionalen Vektorraum verarbeitet. In der Computergrafik ist es üblich, dass sich der Ursprung des Koordinatensystem oben links befindet.

$$\vec{c} = \begin{pmatrix} x_{vec} \\ y_{vec} \end{pmatrix}.$$

### 3.2.1 Orthogonalität

Zwei Vektoren  $\vec{dist}, \vec{orth}$  sind genau dann orthogonal, wenn folgende Bedingung zutrifft.

$$\vec{orth} = n \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} * \vec{dist}, \text{ mit } n \in \mathbb{N}$$

### 3.2.2 Länge eines Vektors

Wird der Vektor als rechtwinkliges Dreieck betrachtet, wobei der Vektor die Hypotenuse und die x und y Ausrichtungen jeweils die Katheten sind, so kann der Satz des Pythagoras angewendet werden.

$$\|\vec{vec}\| = \sqrt{x_{vec}^2 + y_{vec}^2}$$

### 3.2.3 Winkel zwischen zwei Vektoren

Der Winkel  $\alpha$  zwischen zwei Vektoren  $\vec{a}, \vec{b}$  lässt sich mit

$$\cos(\alpha) = \frac{\vec{a} * \vec{b}}{\|\vec{a}\| * \|\vec{b}\|}$$

berechnen. In der Arbeit wird auf eine Umwandlung des Kosinuswert in den Winkel verzichtet, da sie dieselbe Bedeutung haben.

### 3.2.4 Entfernung eines Punktes zu einer Geraden

Seien  $\vec{A}, \vec{B}$  Punkte auf der Geraden  $g$ , dann lässt sich die Gerade  $g$  durch

$$g = n * \vec{dist} * \vec{A}, \text{ mit } \vec{dist} = \vec{B} - \vec{A}$$

beschreiben. Um daraus die Entfernung  $d_{pg}$  zu berechnen, wird die Orthogonale  $\vec{orth} \perp \vec{dist}$  benötigt. Mit diesem Vektor kann eine zweite Gerade  $h$ , die durch  $\vec{P}$  verläuft,

$$h = m * \vec{orth} + \vec{P}$$



konstruiert werden. Nun kann der Schnittpunkt  $\vec{S}$  und somit auch die Entfernung  $d_{pg}$

$$\vec{S} = n * \vec{dist} + \vec{A} = m * \vec{orth} + \vec{P}$$

$$d_{pg} = \|\vec{SP}\|$$

aus den beiden Geraden bestimmt werden. Durch eine Umstellung der y-Werte nach  $m$  und der x-Werte nach  $n$  entstehen folgende zwei Gleichungen

$$(1) m = \frac{n*y_{dist}+y_A-y_P}{y_{orth}}, (2) n = \frac{m*x_{orth}+x_P-x_A}{x_{dist}}.$$

Das Einsetzen von (1) in (2) und das anschließende Auflösen nach  $n$  führt zu

$$(2) n = \frac{(y_A-y_P)*\frac{x_{orth}}{y_{orth}}+x_P-x_A}{x_{dist}-\frac{x_{orth}*y_{dist}}{y_{orth}}},$$

womit sich

$$(1) m = \frac{y_P-y_A-n*y_{dist}}{x_{dist}}, \text{ mit } \vec{orth} = \begin{pmatrix} y_{dist} \\ -x_{dist} \end{pmatrix} \text{ und } x_{dist} \neq 0$$

$$(2) n = \frac{(y_P-y_A)*\frac{y_{dist}}{x_{dist}}+x_P-x_A}{x_{dist}+\frac{y_{dist}^2}{x_{dist}}}, \text{ mit } \vec{orth} = \begin{pmatrix} y_{dist} \\ -x_{dist} \end{pmatrix} \text{ und } x_{dist} \neq 0$$

ableiten und die Parameter  $m$  und  $n$  errechnen lassen.

## 4 Algorithmen

Das Ziel der Extraktion eines Skelett aus einem Bild ist ein komplexer Prozess, für den mehrere Zwischenschritte nötig sind. Jedes dieser nachfolgenden Algorithmen erstellt für diesen Zweck ein isoliertes Feature, dessen Informationen als Eingabe für die Ermittlung der Gelenkpositionen dienen. Dabei werden sie in eine der folgenden drei Kategorien untergeordnet.

- Filterung
- Transformation
- Skelettisierung

Die Filterung gewinnt Daten, die direkt von dem Bild kommen und werden anschließend durch Transformation interpretiert. Diese werden der Skelettisierung zugeführt, welche die Ermittlung der Gelenkpositionen beinhaltet und zugleich die finale Phase ist.

Im weiteren Verlauf des Kapitels werden die einzelnen Kategorien und ihre enthaltenen Algorithmen beschrieben und die Funktionsweise erläutert. Doch zu Beginn müssen einige Randbedingungen für das Bild gesetzt werden, damit der Anwendungsfall und die daraus resultierende Zahl der Features eingegrenzt werden können.

### 4.1 Voraussetzungen an das Originalbild

Das Skelett hat eine große Abhängigkeit zu der Pose und der Kameraposition, denn sie sind ihre definierenden Größen. Eine Eingrenzung der Bilder findet statt, denn die Arbeit befasst sich vor allem mit den Figuren, die aus einer frontalen Perspektive aufgenommen sind. Eine komplette Abdeckung der menschlichen Pose kann derzeit mit ComRec nicht erfasst werden, da jeder Sonderfall zur Erstellung von neuen Features führt. Die Erfolge

und Defizite des Systems im Bezug auf die Voraussetzungen des Originalbildes werden im Kapitel 7 wieder aufgegriffen.

Neben der unzähligen möglichen Posen, die eine menschliche Figur einnehmen kann, so bietet das Bild ebenfalls eine große Verstellbarkeit. Wegen der hohen Farbtiefe existieren unter anderem zwischen Weiß (r:255; g:255; b:255) und Schwarz (r: 0; g:0; b:0) viele Grautöne, die durch keine konkrete Trennlinie auseinander gehalten werden können. Im Bezug darauf müssen Einschränkungen für das Bild vorgenommen werden, bevor die Algorithmen funktionieren können. Die wichtigste farbliche Differenzierung in der Arbeit liegt in Weiß und Schwarz, worauf keine Toleranz in der Erkennung in Kauf genommen wird.

Die Problematik in dieser Verarbeitung liegt darin, dass außer dem Hintergrund kein Weiß und den Linien kein Schwarz existieren darf. Das setzt voraus, dass die Linien tatsächlich schwarz und der Hintergrund weiß sind, doch die meisten Bilder bieten diese Eigenschaften nicht an. Bevor die Bilder also verwendet werden, muss die Figur zuerst von dem Hintergrund getrennt werden, da keine automatische Segmentierung vorgenommen wird. Anschließend werden alle Linien schwarz gefärbt und die gesamten weißen und schwarzen Punkte in der Figur entfernt.

Damit ist das Bild für die nachfolgenden Verarbeitungen vorbereitet.

### 4.2 Filterung

Die erste Phase ist die Filterung und beginnt mit der Verarbeitung des Bildes. Auf den Betrachter kann die Grafik sehr viele Eindrücke und Interpretationsmöglichkeiten hinterlassen, doch das System findet nur eine Karte mit unterschiedlichen Farbwerten vor. Um eine erfolgreiche Posenextraktion durchzuführen, bedarf es neben dem Eingabebild noch weitere Features. Sie werden nicht in Form von neuen Daten eingeführt, sondern werden direkt aus dem Bild gewonnen.

Wie bei einer topologischen Karte, müssen Grenzen oder Gebiete definiert werden, die mindestens eine gemeinsame Eigenschaft besitzen. Die Aufgabe der Filter ist es, diese Flächen zu finden und deren Pixel sorgfältig voneinander zu trennen. Sie sind in einer gesonderten Karte dargestellt, welche im weiteren Verlauf als Abbildung bezeichnet wird. Außerdem können sie dieselben Pixel enthalten, da einige Bildpunkte mehrere Funktionen einnehmen.

Eine weitere Besonderheit der Abbildungen ist es, dass die Komplexität der Farbwerte entfernt wird. Die ursprünglichen  $RGB\alpha$ -Werte werden in eine vereinfachte numerische Darstellung gebracht. So können den Pixeln Werte zugeordnet werden, die eine Bedeutung besitzen und für das System leichter auseinanderzuhalten und interpretierbar ist.

Die Algorithmen für die Informationsfilterung werden in den folgenden Abschnitten vorgestellt.

### 4.2.1 Silhouette und geschlossene Konturlinien

Die Silhouette und die geschlossenen Konturlinien sind die Grundlagen für viele weitere Abbildungen, die während der Extraktion gewonnen werden. Wegen dem höheren Stellenwert und der simplen Abspaltung dieser Features können sie als erstes angegangen werden. Außerdem ist es nach der Definition in Kapitel 3.1 nicht möglich, sie getrennt voneinander herauszuziehen, denn diese beiden Teile legen vereint die Figur fest.

Der Ausgangspunkt ist demnach der Bereich der Figur, der nun im ersten Schritt bestimmt wird. Im Gegensatz zum Hintergrund haben ihre Pixel die Eigenschaft, dass mindestens einer ihrer Farbwerte über 0 liegt. Also muss im gesamten Bild nach Punkten gesucht werden, die nicht weiß sind.

Ist der Bereich für die weitere Untersuchung eingegrenzt, so steht die Unterscheidung für die zwei Abbildungen bevor. Unter Zuhilfenahme der 8-Nachbarn können Pixel der geschlossenen Konturlinien zugeordnet werden, indem ein Vorkommen des Hintergrundes in der Umgebung nachgewiesen ist. Streng genommen muss der Konturpunkt zusätzlich an einem Silhouettenfeld angrenzen, um die Definition zu erfüllen. Allerdings existiert die Silhouette bisher noch nicht und die Position kann ihr, wegen dem Hintergrund, nicht zugewiesen werden. Aus diesem Grund kann diese Bedingung vernachlässigt werden.

Um die Silhouette zu erstellen, werden alle restlichen Figurenpixel hinzugezogen, denn sie haben nur die Möglichkeit, einen der beiden Status anzunehmen.

### 4.2.2 Innere Linien

Die Filterung der inneren Linien nutzt die Besonderheit von Comicbildern, in dem einzelne Körperteile, Schattierungen oder auch Bewegungsabläufe anhand schwarzer Linien akzentuiert werden. Obwohl die Bezeichnung auf etwas anderes hindeutet, gehören die

Konturlinien ebenfalls hierzu, weshalb sie als Basis genommen. Die restlichen schwarzen Pixel innerhalb des Körpers werden dem hinzugefügt.

### 4.2.3 Verdünnung

Konturlinien und innere Linien sind vor dieser Filterung in einem Zustand, der nur schwer zu verarbeiten ist, denn die Linienrichtung lässt an einzelnen Pixeln durch die höhere Breite nicht so einfach ermitteln. Daher ist eine 1-pixelweite Linie erwünscht, da es sonst zu Problemen in der Pfadfindung führen kann, wenn der Suchalgorithmus in einer Schleife feststeht. Das würde bedeuten, dass jeder Punkt nur noch einen Vorgänger und Nachfolger hat. Um das zu erreichen wird die Verdünnung in zwei Phasen aufgeteilt. Zuerst wird der Algorithmus von Zhang und Suen angewandt und danach eigene Filter verwendet. Wie bereits in Kapitel 4.2.2 erwähnt, basieren die inneren Linien auf den geschlossenen Konturlinien. Deshalb muss bei der Ausführung darauf geachtet werden, dass die Pixel der inneren Linien, welche ebenfalls zu den Konturlinien gehören, von diesen Algorithmen ausgeschlossen werden. Das setzt voraus, dass es sich bereits um die verdünnten Konturlinie handelt.

#### Erste Phase - Zhang Suen

Der Algorithmus von Zhang und Suen ist ursprünglich für die Skelettextraktion vorgesehen gewesen, jedoch erfüllt das Ergebnis nicht die gewünschten Anforderungen. Dagegen sind die Eigenschaften der Verdünnung für das weitere Vorgehen optimal. In dem werden Randpixel der Abbildungen im Verlauf von mehreren Zyklen entfernt, bis eine zusammenhängende dünne Linie übrig geblieben ist. Dabei basieren die Berechnungen der Pixel der  $n$ -ten Iteration auf den Werten der 8-Nachbarn der vorhergegangenen Iteration. Demzufolge verändern sich die Werte bei der Konstruktion der neuen Abbildung nicht und die Pixel können parallel bearbeitet werden.

Die Bedingungen für die Reduzierung eines Pixel  $P_0$  werden von Zhang und Suen in folgende zwei Subiterationen aufgeteilt.

## 1. Subiteration

a)  $2 \leq B(P_0) \leq 6$

b)  $A(P_0) = 1$

$c_1) P_2 * P_4 * P_6 = 0$

$d_1) P_4 * P_6 * P_8 = 0$

## 2. Subiteration

a)  $2 \leq B(P_0) \leq 6$

b)  $A(P_0) = 1$

$c_2) P_2 * P_4 * P_8 = 0$

$d_2) P_2 * P_6 * P_8 = 0$

Durch die direkte Gegenüberstellung sind die Gemeinsamkeiten der beiden Subiterationen klar erkenntlich. Demnach können die obengenannten Bedingungen zusammengeführt oder vereinfacht werden.

Dazu gehören *a)* und *b)*, die eine Entfernung der Pixeln verhindern sollen, wenn sie bereits ein Teil des Ergebnisses sind. In den 8-Nachbarn beschäftigt sich *a)* mit der Menge  $B(P_0)$ , welche ebenfalls noch Teile der Linie sind, und *b)* konzentriert sich auf die 0-1 Übergänge  $A(P_0)$ . Handelt es sich um einen Übergang, dann verrät es die Tatsache, dass diese Position nur an einer Hintergrundfläche angrenzt. Ein Merkmal einer einfachen Linie ist es, zwei Flächen voneinander zu trennen, dementsprechend sind die Voraussetzungen in diesem Fall noch nicht erreicht. Eine wiederholte Überprüfung von *a)* und *b)* ist redundant, weil die Ergebnisse, die auf die vorherige Iteration basieren, dieselben bleiben.

Die Merkmale der Pixel werden durch die übrigen Bedingungen bestimmt, so werden die südöstlichen Ränder und die nordöstlichen Ecken durch  $c_1$  und  $d_1$  umfasst. Um die nordwestlichen Ränder und die südwestliche Eck kümmern sich folglich  $c_2$  und  $d_2$ . Mit einer Analyse von  $c_1$  und  $d_1$ , welche genauso auf  $c_2$  und  $d_2$  angewendet werden, sind folgende Vereinfachungen zu erkennen.

$$\text{wenn } P_4 == 0 \vee P_6 == 0 \vee (P_2 == 0 \wedge P_8 == 0), \text{ dann ist } c_1 == true \wedge d_1 == true$$

$$\text{wenn } P_2 == 0 \vee P_8 == 0 \vee (P_4 == 0 \wedge P_6 == 0), \text{ dann ist } c_2 == true \wedge d_2 == true$$

Diese zwei Gleichungen können ergänzend vereinigt werden.

$$P_2 == 0 \vee P_4 == 0 \vee P_6 == 0 \vee P_8 == 0$$

Zusammenfassend bewerten folgende drei Bedingungen, ob es sich um einen Bildpunkt handelt, der nicht in die nächste Iteration überführt werden soll.

- a)  $2 \leq B(P_0) \leq 6$
- b)  $A(P_0) = 1$
- c)  $P_2 == 0 \vee P_4 == 0 \vee P_6 == 0 \vee P_8 == 0$

### Zweite Phase - eigene Filter

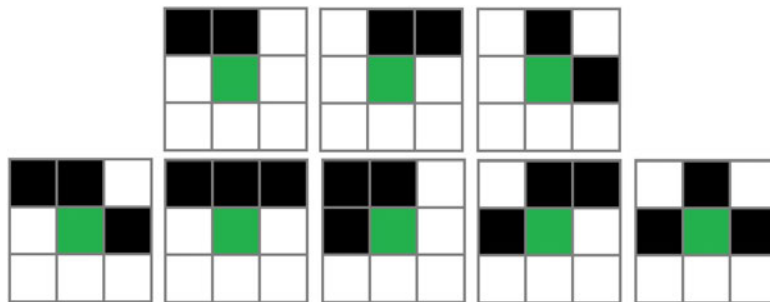


Abbildung 4.1: Verdünnungsfiler für zwei und drei Nachbarn

Die Resultate aus dem Algorithmus von Zhang und Suen sind noch nicht ideal. Sie müssen weiterhin verfeinert werden, denn einige Linienpunkte besitzen in den 8-Nachbarn immer noch zu viele Pixel. Zwar lassen sich die Nachfolgepixel dieser Punkte mit zusätzlichen Berechnungen ermitteln, doch die wollen hier vermieden werden, da das Ziel dieser Filterung eine einfache Linienverfolgung ist. Aus diesem Grund werden Pixel, dessen 8-Nachbarn eines der Muster (vgl. Abbildung 4.1) aufweist, von der jeweiligen Darstellung entfernt. Dabei entspricht das grüne Feld den Betrachtungspunkt und die schwarzen Felder die besetzten Nachbarpixel. Dank der vorherigen Phase sind aus den vielen Konstellationen bloß die Pixel mit 2 und 3 Nachbarn übrig. Die Filter für 2 Nachbarn entfernen vor allem überflüssige Ecken, deren Winkel in der 8-Nachbar Betrachtung kleiner sind als  $90^\circ$ . Diese Beseitigung kann dadurch begründet werden, dass die Nachbarpixel ebenfalls eine direkte Verbindung zueinander haben. Die Filter für 3 Nachbarn nehmen sich zusätzlich Stellen vor, die entlang einer Linie zu finden sind. So werden unter anderem Punkte entfernt, die zu Unebenheiten führen. Durch Rotation werden die Filter wiederverwendet, ohne dass sie dasselbe Schema abdecken.

### 4.2.4 Das grobe Skelett

Das grobe Skelett ist eine gefilterte Abbildung der euclidischen Distanz, die im Kapitel 4.3.2 beschrieben wird. Durch einen definierten Grenzwert, der sich von Figur zu Figur unterscheiden kann, werden alle Pixel, deren Werte sich unterhalb davon befinden, entfernt. Das hat zur Folge, dass weite Teile in Randnähe verschwinden und ein Bereich übrig bleibt, in der sich das mögliche Skelett befinden kann.

## 4.3 Transformation

Die Filterung des Bildes in seine Bestandteile fügen den einzelnen Pixeln zusätzliche Daten hinzu. Sie sagen dennoch nichts über das Skelett aus und deshalb muss ihnen mit den Transformationsalgorithmen eine Bedeutung gegeben werden. Es wird eine Beziehung zwischen den Informationen im Zusammenhang auf große Pixelgruppen oder anderen Abbildungen hergestellt.

In den nächsten Abschnitten werden die Algorithmen für die inneren Linien, euclidische Distanz und der Sobel Abbildung vorgestellt.

### 4.3.1 Die Erweiterung der geschlossenen Konturlinien

Die Extraktion der Silhouette und der geschlossenen Konturlinien bewirkt eine Beseitigung der Tiefeninformation. Sollten sich Teile der Arme im Ursprungsbild vor oder hinter dem Rumpf befinden, so lassen sie sich in den Abbildungen nicht mehr voneinander unterscheiden. Um dem entgegenzuwirken, werden die einfachen Konturlinien, die die inneren Körperstrukturen wiedergeben sollen, ergänzt. Als Teil der inneren Linien sind sie ebenda zu finden, allerdings sind noch weitere Objekte darin abgebildet, die andere Funktionen einnehmen.

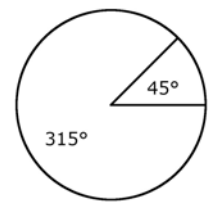
Die einfachen Konturlinien werden zur Trennung der Körperteile verwendet, insofern existieren sie erst, wenn eine Verdeckung vorliegt. An Stellen, an denen zwei Gliedmaßen aufeinandertreffen, liegen die Kreuzungspunkte in der geschlossenen Konturlinie vor. Sie werden Verschmelzungspunkte genannt. Von dort aus verlaufen die weiterführenden Linien, wobei es eine erhebliche Rolle spielt, welcher Körperteil im Vordergrund ist. Schließlich bleiben nur ihre Linien erhalten und geben demzufolge die Richtung vor.



## Verschmelzungspunkte

Der Algorithmus beginnt mit der Ermittlung der Verschmelzungspunkte, die den Anfang der einfachen Konturlinie markiert. Um dies zu erreichen, muss erst verstanden werden, welche Eigenschaften die Positionen innehaben. Als Kreuzungspunkt impliziert es natürlich, dass sich zwei Geraden an dieser Stelle treffen und einen Winkel einschließen. In der Pixelgrafik können sie nicht sofort ausgelesen werden und müssen deswegen errechnet werden. Dafür werden zwei weitere Referenzpunkte benötigt, zu denen jeweils ein Richtungsvektor erstellt wird. Selbstverständlich bestimmt die Auswahl der Pixel über die Qualität des Winkels und dürfen deshalb nicht willkürlich ausgesucht werden. Das wird durch eine Einschränkung auf dieselbe geschlossene Konturlinie geklärt. Hieraus werden Vor- und Nachfolgepixel in einem Abstand von  $K_2 \in N$  ausgesucht. Dabei gilt, je größer die Entfernung ist, desto genauer kann der Winkel werden. Das hilft auch kleinere Unregelmäßigkeiten in der Linie zu ignorieren. Ist die Distanz wiederum zu groß, dann entspricht es nicht mehr den daliegenden Eigenschaften der Figur.

Alle Punkte der geschlossenen Konturlinie haben nun einen Winkelwert erhalten, selbst wenn sie Teil einer geraden Linie sind. Darüberhinaus hat sich nichts geändert, denn es sind bisher keine Pixel aus der Betrachtung ausgeschlossen worden. Entscheidend dafür ist die Einführung eines Grenzwertes  $G_W \in [-1.0, 1.0]$ , der gleichzeitig für zwei Winkel steht (vgl. Abbildung 4.2). Bedauerlicherweise hat der Kosinus keine Unterscheidung zwischen dem großem und kleinem Wert. Es wird schlicht und einfach angenommen, dass es sich um Letzteres handelt. Sollten sie zu den spitzen Winkel gehören, kann es auf eine Überschneidung hindeuten, weil der menschliche Körper normalerweise nur vereinzelt extreme Kanten aufweist. Gleichzeitig kann die Überlappung einen geschmeidigen Übergang haben, weshalb eine große Spannweite für den Grenzwert genommen wird. Für die weitere Ausarbeitung wird  $G_W = -0.7$  gesetzt und nur Pixel mit einem Winkel, der über diesem Wert ist, werden der Menge der möglichen Verschmelzungspunkte hinzugefügt.



$$\cos(45^\circ) = \cos(315^\circ) \\ = 0.707\dots$$

Abbildung 4.2:  
Kosinuswert  
zweier Winkel

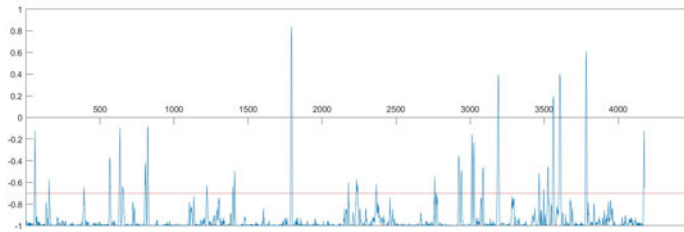


Abbildung 4.3:  
Kosinuswerte und  $G_W(\text{rot})$  in einem Graphen

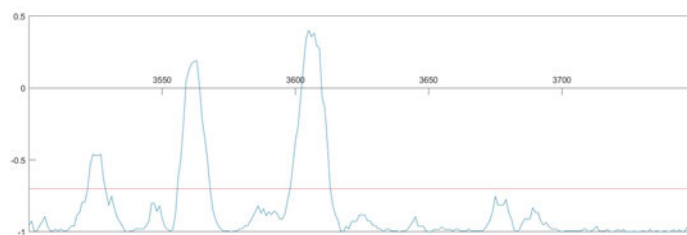


Abbildung 4.4:  
Kosinuswerte und  $G_W(\text{rot})$  in einem Graphen  
im Bereich  $x = [3500, 3750]$

Diese Grenze ergibt sich durch eine Analyse der grafischen Darstellung der Kosinuswerte (vgl. Abbildung 4.2). Bei der Suche nach spitzen Winkeln ist bei  $x \approx 1800$  ein globales Maximum zu finden. Würden sich die weiteren Arbeitsschritte ausschließlich auf diese Position konzentrieren, dann ist eine inkorrekte Auswahl möglich. Neben den gesuchten Verschmelzungspunkten existieren körperliche Kanten, wie zum Beispiel den Achseln, und Falten der Klamotten, die auf diesen Wert zutreffen können. Ein Herabsenken der Linie ermöglicht es, die lokalen Maxima aufzunehmen und somit eine höhere Varianz in der Menge zu haben.

Allerdings liegen die Werte der benachbarten Pixel ebenfalls über der Grenze und müssen entfernt werden (vgl. Abbildung 4.4). Aus diesem Grund wird ein Bereich definiert, aus dem der höchste Wert bestimmt wird. Als Intervallgrenzen werden die Pixel verwendet, die  $K_3 \in N$  vor und nach dem Punkt liegen. Zusätzlich lassen sich mit dieser Methode lokale Maxima herausnehmen, die in der Umgebung ein größeres Maximum haben. Übrig geblieben sind die Kandidaten, um die einfachen Konturlinien zu finden.

### Einfache Konturlinien

Mit den gewonnenen Verschmelzungspunkte lassen sich keine Konturlinien herleiten, weil noch andere Linien in der Figur enthalten sind. Im Gegensatz zu denen haben sie die Eigenschaft, dass ihr Verlauf in die Richtung einer Tangente weiterläuft, die jedoch nicht aus einer Pixelgrafik ausgelesen werden kann. In diesem Fall ist die Vektorgrafik die geeignetere Darstellungsform, denn dort werden sämtliche Linien mithilfe von Kurven konstruiert.

Die hier verwendeten B-Splines benötigen zwei Verschmelzungspunkte, die als Anfang und Ende fungieren. Zusätzlich werden aus dem Zwischenraum Pixel ausgewählt, um eine erfolgreiche Interpolation durchzuführen. Aus jeder Kurve können mittels der Ableitung Tangenten an den Endpunkten errechnet werden. Da Geraden nicht den menschlichen Zügen entsprechen, verläuft die Suche nach den einfachen Konturlinien in einem zylinderförmigen Areal.

Darüberhinaus befinden sich unter Umständen Schattierungen oder andere Objekte im Suchbereich, die herausgefiltert werden müssen. Der Großteil kann bereits durch eine Berechnung eines Winkels zu der Tangente ausgeschlossen werden. Die restlichen Strecken werden über den Suchbereich hinaus erweitert und auf zwei Kriterien überprüft.

Falls die Figur nur eine geschlossene Konturlinie hat, dann müssen die gefundenen Linien innerhalb des Körpers enden. Existieren mehrere geschlossene Konturlinien, dann werden zwei von der einfachen Konturlinie verbunden.

Die Bestätigung durch das zweite Kriterium ist dadurch zu begründen, dass die Hintergrundbereiche inmitten der Figur durch eine Verformung in eine verdeckende Pose entsteht. Dabei befindet sich der äußere Rand des Körperteils an dem großem Hintergrund und der innere Rand an dem kleinem Hintergrund. Eine verbindende Strecke charakterisiert die Form des Gliedmaßes. Andere Verbindungen zwischen den Hintergrundbereichen können ignoriert werden, da sie entweder an keinen Verschmelzungspunkt anbinden oder nicht in die Richtung der Tangente verlaufen. Das erste Kriterium kann nicht durch einfache Mittel bewiesen werden, somit ist die Trennung von anderen Objekten nicht gewährleistet. Trotzdem wird sie für die Erstellung der Euclidischen Distanz benötigt.

### 4.3.2 Euclidische Distanz

Die euclidische Distanz bei einer Silhouette ist ein Bewertungssystem, dessen Ergebnis eine Ähnlichkeit zum Stütz- und Bewegungsapparat des Menschen hat. Damit der Vergleich gerechtfertigt ist, muss die Abbildung viele Charakteristiken des Skeletts entsprechen.

Die Wirbelsäule ist das wichtigste Stützelement, an dem alle Gliedmaßen verbunden sind. Kann diese nicht korrekt ermittelt werden, dann nimmt die Pose eine unnatürliche Form an. Sie befindet sich auf einer zentralen vertikalen Linie des Rumpfes, wenn das Skelett frontal oder rückseitig betrachtet wird. Demzufolge werden zu den beiden Körperseiten die gleichen horizontalen Abstände gemessen. Jedoch können sie nicht in einem neuem Bild errechnet werden, da die Ausrichtung des Körpers dort unbekannt ist. Ersatzweise wird der nächstliegende Konturpunkt für die euclidische Distanz verwendet, weil die Werte annähernd gleich sind. In der Abbildung 4.5 wird die Anpassung veranschaulicht.

$$\begin{aligned} a_1 &\approx b_1 \text{ und } a_2 \approx b_2 \\ \implies a_1 &\approx a_2 \text{ mit } b_1 \approx b_2 \end{aligned}$$

In den Bereichen der Brust und des Steißbeines ist diese Vereinfachung nur bedingt anwendbar, da zum Beispiel die Strecke zu den Schultern deutlich kürzer wird als die Horizontale. Aus diesem Grund wird die Wirbelsäule nur zwischen zwei zentralen Knotenpunkten verglichen, denn da setzen die Extremitäten und der Kopf an. Ab den Becken, Schultern und Halswirbeln nehmen die Knochen einen Großteil der Masse ein. Damit sie und die dazwischenliegenden Gelenkpunkten eindeutig lokalisiert werden können, wird wie bei der Wirbelsäule die zentralen Vertikalen benutzt.

Um diese Struktur ausfindig machen zu können, muss die kürzeste Entfernung jedes Silhouettenpixels zur Kontur bekannt sein. Eine Ermittlung durch die Berechnung aller möglichen Distanzen und eine anschließende Bestimmung des Minimum führt zu erheblichen Laufzeit- und Speicherbelastungen. Diese kann durch eine Verwendung der Informationen in den 8-Nachbarn reduziert werden. Unter der Annahme, dass dort bereits ein Distanzwert  $r_1$  existiert, kann der darauf bezogene Konturpunkt für einen Suchradius  $r_2$  in der aktuellen Berechnung eingesetzt werden.

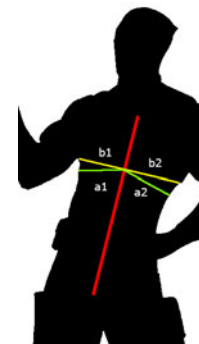


Abbildung 4.5: Silhouette mit Wirbelsäule (rot), horizontale (gelb) und euclidische Distanzen (grün)

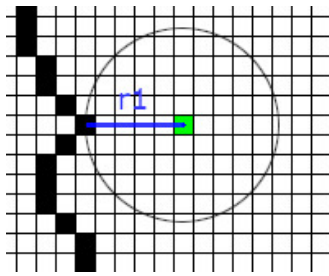


Abbildung 4.6: Pixel (grün) und die euclidische Distanz (blau)

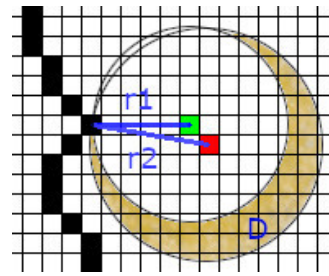


Abbildung 4.7: Pixel (rot), Nachbarpixel (grün) und die euclidische Distanzen (blau)

In der Abbildung 4.6 lässt sich erkennen, dass sich kein weiterer Randpixel in der Kreisfläche  $A_1$  befindet. Folglich kann für den Wert eines Nachbarn ebenfalls keine Referenzpunkte darin gefunden werden. Wird nun die Strecke zum selben Pixel als der Radius  $r_2$  für eine Kreisfläche  $A_2$  verwendet, entsteht ein Differenzbereich  $D$  (vgl. Abbildung 4.7). Die Suche kann auf  $D$  eingeschränkt werden, denn er liegt außerhalb des Einflussbereiches von  $A_1$  und unter der maximal möglichen euclidischen Distanz  $r_2$ .

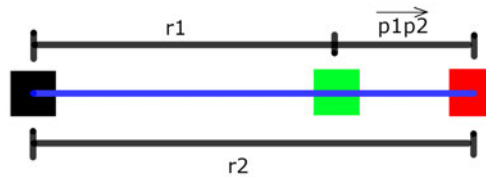


Abbildung 4.8: Pixel (rot) und Nachbar (grün) und die Distanz (blau)

Um  $r_2$  errechnen zu können, werden mindestens drei Informationen für eine Dreiecksgleichung benötigt. Eine Vereinfachung lässt sich dadurch erreichen, in dem alle Punkte auf eine Linie positioniert werden, womit die Dreiecksseite  $r_2$  zu einer zusammengesetzten Strecke wird (vgl. Abbildung 4.8). Der Teilabschnitt  $\overrightarrow{p_1p_2}$  ist die Entfernung zwischen den Pixelmittelpunkten und besitzt als Diagonale in den 8-Nachbarn einen Wert von  $\sqrt{2}$  und sonst 1. Zwar werden in vielen Fällen nicht der korrekte Wert zugewiesen, aber der liegt nie unter diesem. Das Wichtigste dabei ist, dass sich  $D$  nicht verkleinert.

$$r_2 = r_1 + \overrightarrow{p_1p_2} \text{ mit } \overrightarrow{p_1p_2} \in \{1, \sqrt{2}\}$$

Die eigentliche Suche wird von dem Pixel aus vorgenommen und es wird eine radiale Vorgehensweise verwendet. Das bedeutet, dass die Fläche  $A_2$  in Zeilen aufgeteilt wird, die vom Mittelpunkt aus untersucht werden. Pro Zeile können maximal zwei mögliche Kandidaten herausgenommen werden, die sich links und rechts, relativ zum Bezugspunkt, befinden. Alle weiteren Pixel der Reihe können ignoriert werden, da sie mathematisch keine kürzere Distanz mehr aufweisen können. Nachdem alle Werte errechnet sind, können sie miteinander verglichen und der Niedrigste unter diesen ausgesucht werden.

Dieser Algorithmus zur Berechnung der euclidischen Distanz funktioniert nur, wenn vom Rand aus begonnen wird. Die Pixel erhalten den Wert 1, da sie direkt an den Konturpixeln angrenzen. Damit ist die Grundlage gesetzt, dass die Nachfolger Informationen aus den 8-Nachbarn beziehen können. Das wird auf die Silhouette mit jeweils den geschlossenen und den erweiterten Konturlinien angewendet.

### 4.3.3 Sobel

Der Sobel ist eine grafische Abbildung der Gradienten der euclidischen Distanzen, um dort Kanten besonders deutlich hervorzuheben. In der Mathematik werden die Gradienten für gewöhnlich mit der 1. Ableitung einer Formel  $f$  berechnet, aber in diesem Fall handelt es sich um eine 2-dimensionale Karte. Aufgrund dieser Tatsache müsste eine Formel  $f_{euclid}(x, y)$  bestimmt werden, um  $f_{sobel}(x, y)$  herzuleiten. Mit einem Sobeloperator  $O_{sobel}$  lässt sich die komplexe Polynomberechnung von  $f_{euclid}(x, y)$  umgehen und außerdem eine schärfere Kante gewinnen.

$$O_{sobel} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Die Anwendung von  $O_{sobel}$  auf einen Pixel und seinen 8-Nachbarn  $M_{euclid}$  verläuft so, dass sie, entsprechend ihrer Positionen, miteinander multipliziert werden. Der Sobelwert ergibt sich aus deren Summe.

$$f_{F_{sobel}} = \sum_{i=1}^9 (a_i * b_i) \text{ mit } O_{sobel} = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} \text{ und } M_{euclid} = \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{pmatrix}$$

## 4.4 Skelettisierung

Zu guter Letzt ermittelt die Skelettisierung anhand der Ergebnisse der vorangegangenen Algorithmen die Gelenkpositionen. Es wird vor allem auf der Grundlage der euklidischen Distanzen und der Sobel Abbildung gearbeitet, da dort bereits die Eigenschaften der anderen erzeugten Elemente inbegriffen sind.

Für eine erfolgreiche Extraktion sind Ausgangspunkte nötig, von denen aus das restliche Skelett hergeleitet wird. Beim Menschen bietet es sich an, den Rumpf als zentralen Punkt auszusuchen, weil alle Körperteile mit ihm verbunden sind. Der Rumpf wird mittels der zwei Skelettpunkte, Brust und Beckenmittelpunkt, repräsentiert.

Eine Charakterisierung dieser Knotenpunkte wird durch die angebotenen Gliedmaßen erreicht, denn neben den Extremitäten ist die Brust noch zusätzlich mit dem Kopf verbunden. Nachdem sie identifiziert sind, müssen als nächstes die Anbindungspunkte gefunden und diese bis zu den Endpunkten erweitert werden.

### 4.4.1 Ermittlung der Brust und des Beckenmittelpunktes

Die Brust und der Beckenmittelpunkt sind die wichtigsten Komponenten aus dem Zielskelett, denn wie bereits erwähnt, verbinden sie alle Körperteile miteinander. Außerdem befinden sie sich im Rumpf, der das größte Volumen im menschlichen Körper hat. Wird diese Information verwendet, lassen sich die zwei Knotenpunkte aus der euklidischen Distanz, der auf der Grundlage der geschlossenen Konturlinie konstruiert ist, gewinnen.

Bei der Betrachtung der Abbildung 7.1 lässt sich die hohe Dichte der Pixel mit größeren Distanzwerten im Rumpf erkennen. Jedoch sind nicht nur zwei einzelne Punkte zu sehen, die als Brust und Beckenmittelpunkt identifizierbar sind, sondern sie zeigt eine großflächige Auswahlmöglichkeit. Aus dieser Menge müssen zwei Positionen ausgesucht werden, die eine zentrale Lage besitzen, damit sie als Knotenpunkte verwendet werden können. In der euklidischen Distanz bedeutet es, dass es die Pixel mit den maximalen Werten sind.



Abbildung 4.9:  
euklidische Distanz  
auf der Grundlage  
der geschlossenen  
Konturlinien

Mit einer Extremwertsuche kann vorerst nur einer der beiden Punkte bestimmt werden, denn die nächsttieferen Werte liegen in der unmittelbaren Umgebung. Demzufolge müssen sie zuerst entfernt werden, bevor die nächste Suche durchgeführt werden kann. Dafür wird dieselbe radiale Methode, wie sie in Kapitel 4.3.2 genutzt wird, verwendet. Der Unterschied hierbei ist, dass der Radius um 1 verringert wird, damit nach der Löschung weiterhin ein Rand übrig bleibt. Außerdem ist der Großteil des Rumpfes noch erhalten geblieben, da er in den meisten Fällen eine rechteckige Form hat (vgl. Abbildung 4.10).



Abbildung 4.10:  
Rumpf  
als Rechteck

Zusätzlich ist eine Durchführung auf die euclidische Distanz mit den erweiterten Konturlinien für die nächsten Phasen wichtig. In diesem Fall kann es jedoch vorkommen, dass die Konturlinien durch die Kreisfläche laufen und sie in mehrere Teile trennt. Sie gehören entweder dem Rumpf oder einer Extremität an, welches durch die Position des maximalen Wertes bestimmt wird. Der Bereich, der diesen Punkt enthält, gehört dem Rumpf an und kann deswegen gelöscht werden. Dabei wird ebenfalls darauf geachtet, dass ein Rand übrig bleibt.

Nachdem der erste Knotenpunkte extrahiert ist und die Flächen aus den Abbildungen entfernt sind, wird eine erneute Extremwertsuche und die anschließende Entfernung der Umgebungspixel durchgeführt. Zu diesem Zeitpunkt sind beide Knotenpunkte noch nicht gekennzeichnet, weil dafür die Anbindungspunkte benötigt werden.

### 4.4.2 Ermittlung der Anbindungspunkte

Für die Anbindungspunkte wird die zuvor bearbeitete euclidische Distanz mit den erweiterten Konturlinien gebraucht. Sie werden den Pixeln, die an den gelöschten Bereichen angrenzen, entnommen, somit wird gewährleistet, dass zwischen ihnen und den Knotenpunkten eine direkte Verbindung besteht.



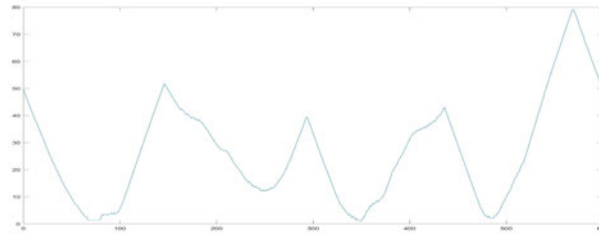


Abbildung 4.11: euclidische Distanzen um den leeren Brustbereich

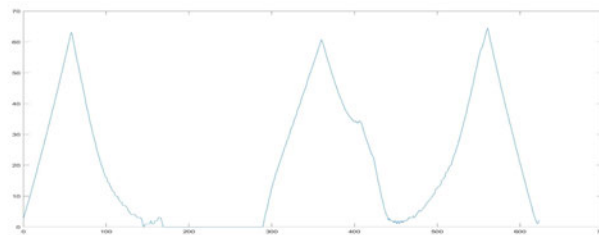


Abbildung 4.12: euclidische Distanzen um den leeren Beckenbereich

Für die korrekte Menge der Anbindungspunkte muss bekannt sein, welcher Teil des Rumpfes sich im Zentrum des Kreises befindet. Am Ende werden nur fünf Punkte für die Ermittlung der Körperteile benötigt. Sie alle besitzen die Eigenschaft, dass sie sich auf einer mittigen Linie in einem Körperteil befinden und somit ein lokales Maximum besetzen. Für eine Analyse werden die Pixel und die dazugehörigen Werte, die während der Umrundung notiert sind, grafisch Dargestellt (vgl. Abbildung 4.11 und 4.12).

Die Körperteile unterscheiden sich unter den Menschen sehr stark, deswegen ist die Festlegung eines Grenzwertes schwierig. Die Bestimmung der lokalen Maxima erfolgt anhand der Beobachtungen an den Graphenbewegungen. Es wird ein Extrempunkt gefunden, wenn der Graph sich nach einer Aufwärtsbewegung wieder in einer Abwärtsbewegung befindet. Aus beiden Umrundungen werden jeweils die vier höchsten Extrempunkte entnommen, die jedoch nicht dem entsprechen, was auf den zwei Graphen zu sehen ist. Der Punkt mit dem niedrigsten Wert ist wegen einem einfachen Ausschlussverfahren kein Anbindungspunkt, da die Knochen entlang einer Maximallinie verlaufen.

Um die restlichen überflüssigen Punkte zu eliminieren, wird in der Fläche zwischen den zwei Kreisen gesucht. Dieser Bereich ist bei dem Löschprozess übrig geblieben und hat

eine ähnliche Distanzabbildung wie im restlichen Rumpf. Das bedeutet, dass es wahrscheinlich ist, dass die höchsten Werte dem der Brust und des Beckenmittelpunktes ähnlich sind und sich in der Liste der Anbindungspunkte befinden. Unter dieser Annahme wird die Strecke zwischen den zwei Knotenpunkten für die Lokalisierung verwendet, denn die Pixel liegen auf einer Fläche die sich senkrecht dazu ausbreitet. Nachdem auch diese zwei Pixel entfernt sind, bleiben nur die tatsächlichen Anbindungspunkte erhalten.

Nun verraten die verbliebenen Pixel die Funktionen der Knotenpunkte, denn die Brust ist an drei Körperteilen angebunden und der Beckenmittelpunkt an zwei. Bei der Auflistung ist die Zugehörigkeit der Anbindungspunkte verloren gegangen, deshalb wird sie durch die kürzere Distanz zwischen ihnen und den Knotenpunkten festgelegt. Somit können auch die Knotenpunkte identifiziert werden.

### 4.4.3 Ermittlung der Körperteile

Die relative Positionen der Anbindungspunkte zu den Knotenpunkten geben Aufschluss, um welchen Körperteil es sich handelt. Bevor also die Expansion des Skeletts beendet werden kann, müssen diese festgestellt werden. Sie sind aus ihren Bezeichnungen zu beziehen, so befindet sich zum Beispiel die linke Schulter, wenn die Blickrichtung aus dem Bild zeigt, auf der rechten Seite der Brust. Nur der Nacken hat keine solche Bezeichnung, aber sie kann als Punkt, der am Weitesten von dem Beckenmittelpunkt entfernt ist, beschrieben werden. Dafür werden die Distanzen der Anbindungspunkte berechnet und daraus das Maximum gewählt.

Nun wird der Sobel gebraucht, um der Linie zu folgen, da die Kanten auf der Abbildung deutlicher zu erkennen sind. Die Werte der Mittellinie gehen gegen 0 und haben einen größeren Unterschied zu ihren Nachbarpixel. Für die Verfolgung des Verlaufes wird für jeden Pixel in den 8-Nachbarn nach dem minimalen Wert gesucht. Außerdem muss der neue Pixel die Eigenschaft erfüllen, dass die Strecke zwischen ihm und einem Vorgängerpixel größer wird, damit keine rücklaufende Strecke gefunden wird. Beendet ist die Suche durch das Erreichen des Randes im grobem Skelett, da es bei einer Minimalwertbestimmung der 8-Nachbarn immer ein Ergebnis erzielt wird. Beim Nacken als Startpunkt befindet sich der Kopf am Endpunkt.

Um die seitliche Ausrichtung der Schultern und Beckenseiten auszumachen, wird von dem Richtungsvektor zwischen den Knotenpunkt die Orthogonale errechnet. Ein Winkel zwischen dieser und einem Richtungsvektor von dem naheliegendem Knotenpunkt zum

Anbindungspunkt zeigt die Seite an, dabei ist darauf zu achten, um welche der beiden Orthogonalen es sich handelt. Eine Verdrehung ist sonst möglich, welche sich nur auf die Blickrichtung des Skeletts auswirkt.

Für die Ermittlung der Endpunkte wird genauso wie im Falle des Kopfes vorgegangen. Zusätzlich wird die Position auf halber Strecke für das Zwischengelenk markiert, denn die Extremitäten bestehen aus zwei Teilen, die ungefähr dieselbe Länge haben.

# 5 Konzept

In diesem Kapitel werden die geplanten Strukturen des Systems vorgestellt und welche Anforderungen sie für die Extraktion erfüllen müssen.

## 5.1 Anforderungen

Die Extraktion des Skelettes basiert auf die Ergebnisse der vorgestellten Algorithmen. Die Klassen müssen sich an die Vorgehensweise orientieren.

Darüber hinaus sind folgende Anforderungen für die Anwendung vorgesehen:

- Es wird eine einheitliche Datenstruktur für das Skelettobjekt benötigt, da es sich bei diesem System um den ersten Abschnitt von ComRec handelt. Für eine Zusammenarbeit wird ein gemeinsames Skelettobjekt erstellt, damit die Arbeiten getrennt voneinander bearbeitet werden können
- Eine Klasse soll alle Abbildungsobjekte sammeln, damit die Algorithmen ohne Einschränkungen auf sie zugreifen können.
- Alle Algorithmen sollen die Abbildungssammlung durch eine Dependency Injection im Konstruktor erhalten. Somit wird eine einheitliche Erstellung für die Komponenten erreicht und gleichzeitig fördert es die Austauschbarkeit.
- Die Getter und Setter der Abbildungssammlung sollen die Objekte mittels einer `copy()` - Funktion übergeben, sodass die Abbildungsdaten unverändert bleiben und die nachfolgenden Algorithmen sie verwenden können.
- Für sämtliche Abbildungen sollen dieselbe Datenstruktur verwendet werden. Die Speicherung der Informationen beschränken sich auf die relevanten Pixel und beschleunigen einen Durchlauf.
- Die Splines in der Vektorgrafik sollen austauschbar sein.

- Die Vektorgrafik und die Abbildungsstruktur werden mit demselben Interface konstruiert.
- Konvertieren der Datensätze zu Bilddaten.

## 5.2 Architektur

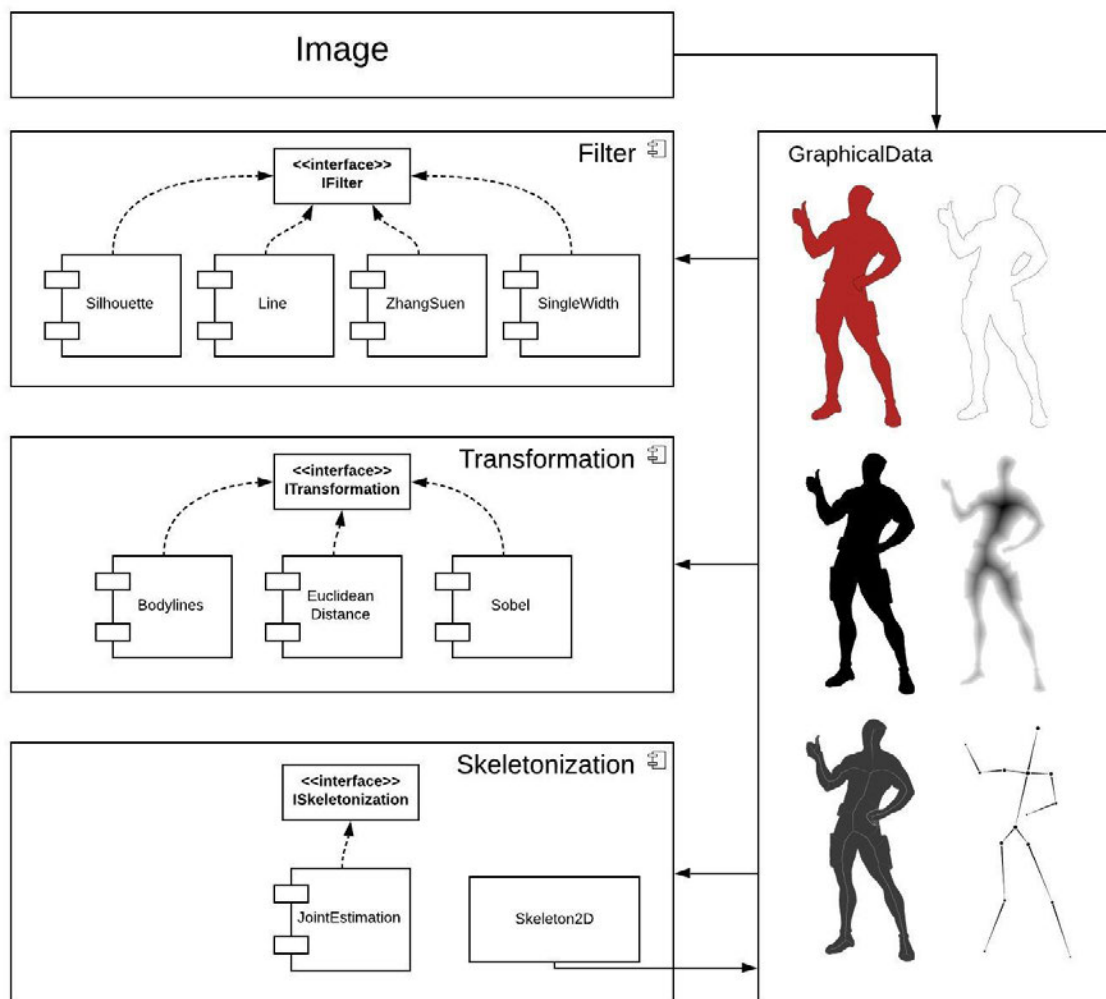


Abbildung 5.1: UML-Diagramm

## 5.3 Schnittstellen

Die Rolle der Schnittstellen in einem Softwareprojekt ist die Vereinbarung der Methoden, damit mehrere Programme miteinander kommunizieren können. Die implementierenden Klassen sind an diesen Vorgaben gebunden, aber sie können sie mit unterschiedlichen Funktionen belegen. Ein positiver Nebeneffekt ist die Klassifizierung der Komponenten mit denselben Einstiegspunkten.

Für die drei vorgestellten Themengebiete Filterung, Transformation und Skelettisierung sind sie wie folgt definiert.

### **IFilter**

```
extract();
```

### **ITransformation**

```
run();
```

### **IJointEstimation**

```
createSkeleton();
```

Die Komponenten, die die Schnittstelle zum Starten ihrer Berechnungslogik verwenden, können unter Bedingung ihrer eigenen fachlichen Abhängigkeiten beliebig ausgetauscht werden. Dafür wird zusätzlich die Allokation und die Freigabe des Speichers durch das `IComponent` - Interface mitgegeben. Eine zentrale Bindung der benötigten Abbildungsobjekte hilft dabei, eine schnelle Übersicht über die Grundlagen der Komponente zu erhalten. Um die Ergebnisse nach der Vollendung des Algorithmus nicht zu verlieren, wird von dem Interface eine Methode zur persistenten Speicherung angegeben.

### **IComponent**

```
void loadData();  
void clearData();  
void saveData();
```

Neben den Komponenten der Algorithmen bekommen die grafischen Elemente ebenfalls eine Schnittstelle bereitgestellt, die vor allem der Iteration dienen soll. Mit diesen Methoden wird ein Iterator aktiviert, der trotz ihrer unterschiedlichen Darstellungsarten alle Bildpunkte einmal durchläuft. Sie bietet währenddessen die Möglichkeit an, den aktuellen Pixel zu löschen und beendet schlussendlich, wenn kein Pixel in der Abbildung übrig geblieben ist.

## **IGraphics**

```
boolean startIteration();  
Vector next();  
int remove();
```

## **5.4 MATLAB**

MATLAB von The MathWorks <sup>1</sup> ist eine umfangreiche Software, die viele Möglichkeiten zum Lösen mathematischer Probleme bietet. Zudem können die Ergebnisse grafisch dargestellt und für eine Analyse bereitgestellt werden.

Um diese Funktionen in das System zu integrieren, hat MATLAB Bibliotheken für verschiedene Programmiersprachen entwickelt. Bei der Auswertung der Bibliothek ist aufgefallen, dass es sich dabei um ein Kommunikationsinterface für ein Engine handelt. Die Nutzung der mathematischen Formeln verläuft über ein Datenstream, der auf Strings basiert, weshalb die Funktionsaufrufe nicht als elegante Lösung angesehen sind. Doch trotz dessen ist die Entscheidung auf diese Bibliothek gefallen, da MATLAB eine große Formelsammlung besitzt, die außerdem gut dokumentiert ist.

Ein weiterer Nachteil der MATLAB Bibliothek ist die Konsolenausgabe, da sie sich nicht abstellen lässt. Denn jede Berechnung, die über den Datenstream versendet wird, löst Java's sysout aus. Viele Anfragen können die Laufzeit verlängern, weshalb bei der Implementierung darauf geachtet werden muss, dass es auf ein Minimum reduziert wird oder Teile für die eine optimierte Laufzeit abstellbar ist.

---

<sup>1</sup><https://www.mathworks>

# 6 Implementation

## 6.1 Datentypen

### 6.1.1 BaseDataSet

Die abstrakte Klasse `BaseDataSet` ist eine Grundlage für alle Datentypen, die in dem System verwendet werden. Sie ist für die Bereitstellung der `copy()` - Methode zuständig, die die Inhalte eines Objektes mithilfe der `setData()` - Methode in eine neue Instanz überträgt. In Anbetracht der Vererbung von `BaseDataSet` wird ein gleichbleibendes Verhalten für alle Datentypen vorausgesetzt, was durch die Verwendung von Java's Generics nicht ganz trivial ist.

Eine neue Instanz wird mit der zusätzlich `createNewInstance()` - Methode erstellt. Durch die besondere Eigenheit von Generics können in Klassen ein generischer Typ `<T>` deklariert werden, jedoch lassen sich damit keine neuen Objekte erstellen. Ein Schlupfloch eröffnet sich, wenn auf die Subklasse zurückgegriffen wird. Sie bietet eine tatsächlich existierende Klasse an, aus dem eine neue Instanz generiert wird.

```
T t = (T) this.getClass().newInstance();
```

`<T>` könnte bei der Deklaration einer Klasse jeden Typ annehmen, welches nicht der Subklasse entspricht. Das würde dem Ziel der `copy()` - Methode widersprechen, da die Klassen des Ursprungs- und Ergebnisobjektes sich voneinander unterscheiden. Mit dem Gleichsetzen von `<T>` und einer Subklasse muss die folgende Deklaration vorgenommen werden.

```
public class SubClass extends BaseDataSet<SubClass>{}
```



Ein wichtiger Bestandteil von `BaseDataSet` ist die abstrakte Methode `setData()`, die im Gegensatz zu `copy()` und `createNewInstance()` keine Implementierung beinhaltet. Klassen, die solche Methoden besitzen, können nicht instanziiert werden, solange sie noch nicht mit einem Inhalt gefüllt sind. Das muss in den jeweiligen Subklassen vorgenommen werden. Damit gewährleistet wird, dass der Typ `<T>` ebenfalls über die `setData()`-Methode verfügt, muss es sich in diesem Fall um eine Vererbung von `BaseDataSet` handeln. Dies unterstützt die Bedingung im vorherigen Paragraphen.

```
public class BaseDataSet<T extends BaseDataSet>{}
```

Ein ähnliches Ergebnis könnte mit einem Kopierkonstruktor erreicht werden, aber es wurde sich für eine Konstruktion mit Generics entschieden, da sich mit diesem einige Vorteile hervorheben. Eine Vererbung der Methoden bringt das gleiche Verhalten mit sich, welche durch einen einheitlichen Aufruf ausgeführt werden kann. Außerdem wird eine Implementierung von `setData()` angestoßen, da sonst keine Instanz der Subklasse möglich ist. Demzufolge ist eine Priorisierung der Attribute schon beim Klassendesign erforderlich, damit die Kopie des Objektes definiert ist.

### 6.1.2 ImageTransformation

Das hier implementierte Teilsystem von `ComRec` bearbeitet und bewertet Bildinformationen, um daraus Schlussfolgerungen für ein Skelett zu ziehen. Die Datenstruktur, die das Eingangsbild beschreibt, ist für solche Aufgaben zu umfangreich, da jeder Pixel mehrere Farbwerte und eine Transparenzkonstante enthält. Dem stehen die Zwischenergebnisse, die in den verschiedenen Algorithmen erstellt werden, gegenüber, weil einzelne Abbildungen nur einen minimalen Teil der gesamten Fläche einnimmt. Außerdem wird eine Reduzierung der Informationsmenge auf einen einfachen numerischen Wert vorgenommen, deswegen wird eine neue Datenstruktur erarbeitet, die nicht so speicherintensiv ist.

Um eine geeignete Struktur zu finden, die den Anforderungen entsprechen und dazu noch effizient ist, müssen mehrere Möglichkeiten vorgestellt werden. Jede dieser Methoden hat ihre besondere Funktionsweise, dessen Vor- und Nachteile erörtert werden. Im Anschluss darauf wird eine Gegenüberstellung der Eigenschaften vorgenommen, sodass die kompakte Ansicht bei der endgültigen Entscheidung hilft.

Die erste Vorgehensweise beruht auf der Idee, dass den Pixeln sämtliche Informationen über sich selbst bekannt sind. Die `PixelInformation` - Klasse soll ähnlich wie ein Datensatz im  $RGB\alpha$  - Farbmodell funktionieren. So wie die Werte dort die Intensität der einzelnen Farbschichten angibt, zeigen die Instanzattribute an, wie sie in den Abbildungen belegt sind. Die Größe eines Pixelobjektes kann durch eine durchdachte Auswahl der Basisdatentypen beeinflusst werden. Ist die Abbildung zum Beispiel eine binäre Darstellungsform, wie die Silhouette eine ist, dann kann sie mit einfachen booleschen Werten ausgedrückt werden.

```
class PixelInformation{
    silhouette: boolean;
    . . .
}
```

Damit die Pixelinformationen für das Gesamtbild eine Bedeutung erhält, müssen sie einer Position auf einer Karte zugeordnet werden. Anstatt Aufzeichnungen über die gesamte Länge und Breite des Bildes zu speichern, können mithilfe einer mehrdimensionalen `HashMap` nicht verwendete Pixel ausgelassen werden. Das ist wegen der besonderen Zugriffsfunktion dieser Datenklasse möglich, die ein Objekt hinter einem Schlüssel hinterlegt. Die Schlüssel entsprechen den linearen x- und y-Achsen und erlauben Sprünge zwischen den Werten.

```
private HashMap<Integer, HashMap<Integer, PixelInformation>>
    pixelInformationData;
```

Ein weiterer Vorteil dieser Methode ist es, dass durch eine Abfrage sämtliche Informationen in einem Objekt bereitgestellt sind. Das erlaubt es, Algorithmen zu erweitern, indem sie weitere Abbildungen zu den Berechnungen hinzuziehen, ohne sie vorher explizit importieren zu müssen.

Der Speicherverbrauch wird zwar durch die Nutzung der `HashMap` verringert, aber sie kann wieder durch leere Felder in den `PixelInformationen` rückgängig gemacht werden. Außerdem ist die Erstellung einer Kopie an alle `Pixelobjekte` gebunden, somit sind isolierte Abbildungen nur mit zusätzlichen Mitteln möglich. Das bedeutet, dass bei jeder Vervielfältigung der Daten, der Speicherplatz auf das doppelte vergrößert wird, wenn ein Teil der Ergebnisse manipuliert und nach den Berechnungen wieder verworfen werden sollen.

Die zweite Methode greift ebenfalls die Darstellung mit der mehrdimensionalen Hash-Map auf, nur mit dem Unterschied dass für jede Abbildung ein eigenes ImageTransformation Objekt initiiert wird. Damit die Klasse eine übergreifende Benutzbarkeit für die verschiedenen Ergebnisse hat, wird wieder auf Java's Generics zurückgegriffen. Da die gespeicherten Informationen eindimensionale Werte sind und der Typ `<T>` nur eine Vererbung erlaubt, wird die Klasse `Number` als Superklasse ausgesucht.

```
public class ImageTransformation<T extends Number & ... > ... {
    private HashMap<Integer, HashMap<Integer, T>>
        imageTransformationData;
    . . .
}
```

Wie zuvor können Pixel aus der Abbildung ausgeschlossen werden, wenn sie keine relevanten Daten repräsentieren. Da Konturen oder ähnliche Grafiken keine großflächigen Areale haben, wird durch die getrennte Speicherung der Informationen der Verbrauch auf das Minimum reduziert. Sie müssen nicht wegen anderen Abbildungen mit bedeutungslosen Daten gefüllt werden. Desweiterem lassen sich einfache Kopien anlegen, die in den Algorithmen bearbeitet werden können. Jedoch werden mit jedem Pixel ein Schlüssel und Werte Paar erstellt und vergrößert somit jedes Objekt auf das Doppelte.

Die letzte Methode verwendet neben der `HashMap` ein 2-dimensionales `int`-Array, das anders als die Vorgänger einen Bereich definiert, in dem markierte Pixel vorhanden sind. Für jede Zeile wird ein eigenes Array erstellt, dessen zweite Dimension entweder einzelne Positionen oder eine durchgehende Reihe anzeigt. Somit haben die Elemente höchstens zwei Werte. Um einen Bereich damit beschreiben zu können, werden der Anfangs- und Endpunkt, die die vielen Pixel eingrenzen, benötigt.

```
private HashMap<Integer, int[][]> rangeArray;
```

Der Vorteil dieser Methodik liegt ganz klar im Speicherverbrauch, denn es spielt hierbei keine Rolle, wieviele Pixel abgebildet werden müssen. Selbst lange Pixelreihen werden durch ein Wertepaar angegeben. Das worst-case-Szenario ist, wenn zwei benachbarte Pixel und eine Lücke im Wechsel das gesamte Bild füllen. In diesem Fall kann die Datenzahl nur auf  $2/3$  der ursprünglichen Größe verkleinert werden.

Negativ fällt auf, dass es keine individuellen Werte für die Pixel angegeben werden können, weswegen es sich nur auf binäre Abbildungen anwenden lässt.

Im Folgendem werden die Methoden an drei Beispilmuster angewendet und die Ergebnisse miteinander verglichen.

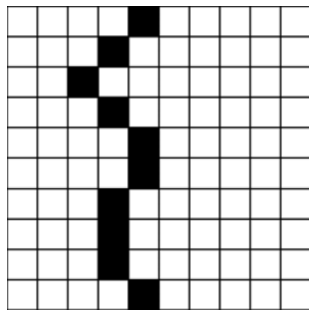


Abbildung 6.1: Beispiel 1

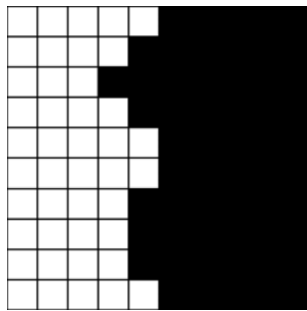


Abbildung 6.2: Beispiel 2

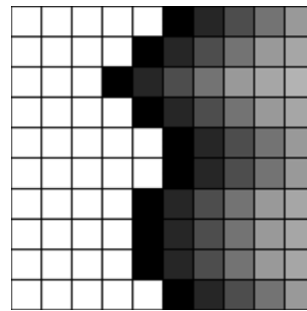


Abbildung 6.3: Beispiel 3

	Abb. 6.1	Abb. 6.2	Abb. 6.3
Methode 1	$\{0: \{4: o_1, 5: o_2, \dots\}, \dots\}$ $o_1 = \{\text{Abb. 6.1}=1, \text{Abb. 6.2}=0, \text{Abb. 6.3}=0\}$ $o_2 = \{\text{Abb. 6.1}=0, \text{Abb. 6.2}=1, \text{Abb. 6.3}=9\}$		
Methode 2	$\{0: \{4: 1\},$ $1: \{3: 1\},$ $\dots\}$	$\{0: \{5: 1, 6: 1, \dots\},$ $1: \{4: 1, 5: 1, \dots\},$ $\dots\}$	$\{0: \{5: 9, 6: 8, \dots\},$ $1: \{4: 9, 5: 8, \dots\},$ $\dots\}$
Methode 3	$\{1: [[4]],$ $2: [[3]],$ $\dots\},$	$\{1: [[5, 9]],$ $2: [[4, 9]]$ $\dots\}$	nicht möglich

Tabelle 6.1: Speicherverbrauch der Pixelstrukturen

Bei der Analyse der Tabelle 6.1 ist deutlich zu sehen, dass die Methode 3 in den ersten beiden Fällen eine deutliche Einsparung aufweist. Leider müsste bei dieser Vorgehensweise, wegen der nicht existenten Abdeckung des dritten Falles, eine weitere Struktur implementiert werden. Die weiteren Methoden erfüllen hingegen alle Anforderungen, die die Abbildungen verlangen. Die erste Methode überragt vor allem, wenn die gleichen Flächen ausgefüllt werden müssen, somit werden die Instanzvariablen nicht mit einer 0 ergänzt. Methode 2 lässt sich bei einer Vielzahl unterschiedlicher Abbildungen anwenden, wobei es vorwiegend aus Bilder, wie dem Fall 1, bestehen sollte.

Der Algorithmus für die Skelettisierung besteht aus vielen verschiedenen Modulen, die einen Zugriff auf die Daten benötigen. Der Verbrauch, der durch eine Kopie entsteht,

	Kopie der Daten aus Abb. 6.1
Methode 1	$\{0: \{4: o_1, 5: o_2, \dots\}, \dots\}$ $o_1 = \{\text{Abb. 6.1}=1, \text{Abb. 6.2}=0, \text{Abb. 6.3}=9\}$ $o_2 = \{\text{Abb. 6.1}=0, \text{Abb. 6.2}=1, \text{Abb. 6.3}=8\}$
Methode 2	$\{0: \{4: 1\}, 1: \{3: 1\}, \dots\}$
Methode 3	$\{1: [[4]], 2: [[3]], \dots\},$

Tabelle 6.2: Speicherverbrauch einer Kopie

wird mit der Auswertung der Tabelle 6.2 verdeutlicht und spielt bei der Auswahl eine entscheidende Rolle. Werden die Daten aus der Tabelle 6.1 mit diesen verglichen, dann wird bei Methode 1 der gesamte Datensatz übertragen. Methode 2 und 3 ignorieren die anderen Abbildungen, aber da Methode 3 keine vollständige Fallabdeckung besitzt, ist die Entscheidung, trotz einiger Defizite, auf Methode 2 gefallen.

Für die Verwendung der Klasse werden Methoden benötigt, die eine Manipulation der Daten zulässt. Das schließt das Hinzufügen aber auch das Entfernen von Pixelinformationen ein. Damit die Abbildung um ein weiteren Pixel ergänzen zu können, werden die x- und y-Koordinaten der Position sowie ein positiver Eingabewert erwartet. Mit den Angaben kann zuerst die Existenz der Zeile in der HashMap abgefragt werden. Wenn sich kein Objekt finden lässt, dann wird eine neue HashMap initiiert und mit der Zeilennummer als Schlüssel hinzugefügt. In dieser Zeile wird nach der Spaltenposition gesucht und der Eingabewert eingetragen.

Bei einem Wert, der kleiner oder gleich 0 ist, wird die Löschfunktion aufgerufen. Die Abfragen bis zum Pixel bleiben gleich, aber die Werte werden aus der HashMap entfernt. Sollte die Zeile zu dem Zeitpunkt keine Daten mehr enthalten, dann kann diese gelöscht werden, um den Speicher wieder freizugeben.

```
public int setPixel(int row, int col, T pixelValue);
```

Eine wichtige Funktion, um alle Daten durchzuarbeiten, sind die Methoden aus dem IGraphics - Interface für die Iteration (vgl. Kapitel 5.3). Bei einem Start wird von der HashMap ein Zeileniterator Element erstellt und dem Instanzattribut iteratorRow übergeben. Das ist wichtig, da der Iterator nicht als Ergebnis der Methode zurückgegeben wird, sondern als interner Mechanismus arbeitet. Um das nächste Element von dem Iterator zu erhalten, muss auch iteratorCol ebenfalls mit einem Spalteniterator Element belegt sein. Der wird nach dem Durchlauf eines Spalteniterator aus der nächsten Zeile

gewonnen und aktualisiert. Da dieser interne Vorgang lediglich die x- und y-Werte zurückgibt, müssen sie in den `currRow` und `currCol` Attributen gespeichert werden, da sie sonst bei dem Abschluss der Methode verloren gehen. Mit den Informationen können die Pixeldaten geholt, aber auch gelöscht werden.

```
private Iterator<Integer> iteratorRow;
private Iterator<Integer> iteratorCol;
private int currRow;
private int currCol;
```

### 6.1.3 Vectorgraphic

Die `Vectorgraphic` - Klasse wird für die Erweiterung der Konturlinien (vgl. Kapitel 4.3.1) verwendet, um die Verschmelzungspunkte und die dort entspringenden Tangenten zu gewinnen, die wichtige Bestandteile bei der Durchführung des Algorithmus sind. Denn wie bereits in den Grundlagen (vgl. Kapitel 3.1.5) beschrieben, werden die Vektorgrafiken aus verschiedenen Grundformen aufgebaut, aus denen durch mathematische Berechnungen weitere Informationen gewinnen lassen. Würde eine Pixelgrafik verwendet werden, dann ergeben sich daraus nur Annäherungen, dessen Genauigkeit darauf beruhen, wie viele Pixel für die Ermittlung hinzugezogen sind. Obwohl die Vektorisierung (vgl. Kapitel 6.1.4) auf denselben Pixeldaten arbeitet, entsteht ein Objekt, das nicht nur in einem kleinen Bildabschnitt eine Deckungsgleichheit anstrebt, sondern über eine längere Strecke.

Für die Umsetzung einer Vektorgrafik sind in der eigentlichen Definition unterschiedliche Formen anwendbar, jedoch beschränkt sich die Grundlage hier auf die B-Splines. Diese müssen der `Vectorgraphic` - Instanz in einer festgelegten Reihenfolge übergeben werden, weil der neue Spline am Endpunkt des letzten Objektes ansetzt. Damit kann nach der Umformung einer geschlossenen Konturlinie ein ganzer Rundlauf simuliert werden, in dem die Länge als Funktionsparameter verwendet wird.

```
public boolean addSpline(T spline);
```

Außerdem können somit zwei aneinanderliegende Splines zusammengeführt werden, ohne sie in der Liste suchen zu müssen. Das wird genau dann ausgeführt, wenn ein neu hinzugefügtes Objekt zu wenige Linienpunkte besitzt. Sie werden zu den Linienpunkten des letzten Splines angehängt und eine neue Berechnung der Knotenpunkte wird angestoßen.

```
public boolean splineConnect();
```

Natürlich müssen die Methoden des IGraphics - Interface ebenfalls implementiert werden. Wie bei der HashMap in der ImageTransformation - Klasse (vgl. Kapitel 6.1.2) wird von der Splinelist ein Iterator erstellt und die Elemente nacheinander abgearbeitet. Um die Pixel durch die Formeln zu finden, werden in gleichmäßigen Abständen zwischen dem Startwert  $t_0$  und dem Endwert  $t_n$  die Positionen berechnet. Dabei sind die Abstände möglichst klein zu wählen, damit keine Lücken in den Kurven entstehen. Das wird an der Gleichung  $f(x) = \Delta d * x$  verdeutlicht.

mit  $\Delta d = 0.5$

$$f(0) = 0$$

$$f(1) = 0.5$$

...

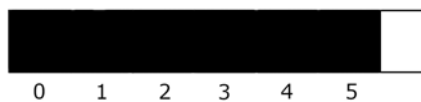


Abbildung 6.4: Vektorgrafik  
mit der Distanz  $\Delta d = 0.5$

mit  $\Delta d = 2$

$$f(0) = 0$$

$$f(1) = 2$$

...



Abbildung 6.5: Vektorgrafik  
mit der Distanz  $\Delta d = 2$

### 6.1.4 Splines

Die Beschreibung einer Kurve durch Splines ist von vielen Parametern abhängig, da sie dadurch viele Formen annehmen kann. Der Grund dafür liegt in der Zusammensetzung mehrere Polynome, die eine Position über die Gewichtung der Knotenpunkte errechnet. Mit einer Ordnungszahl kann bestimmt werden, wieviele Punkte Einfluss auf die Berechnungen haben.

```
public void setOrder(int k);
public int setStartPoint(Vector startPoint);
public int setEndPoint(Vector endPoint, int tn);
public int addSplinePoint(Vector splinePoint);
```

Es lassen sich zu diesem Zeitpunkt keine direkten Angaben über die Knotenpunkte gewinnen, weshalb auf die Linie zurückgegriffen wird, die der Spline ersetzen soll. Daraus

werden in gleichen Abständen Punkte entnommen, um eine Interpolation durchzuführen. Die Implementation der bisherigen Spline-logik wird von einer abstrakten Klasse abgedeckt und die Interpolation und jegliche Berechnungen sind in den Subklassen enthalten. Die Entwicklung hat sich in diese Richtung bewegt, weil die eigene Programmierung keine verwendbaren Ergebnisse erzielt haben. Deswegen ist das Fundament für Bibliotheken entwickelt worden, um sie einfacher austauschen zu können.

```
public abstract boolean initialize();
public abstract Vector calculate(double t);
public abstract Vector getStartDerivative();
public abstract Vector getEndDerivative();
```

Bei der Verwendung der Matlab Bibliothek (vgl. Kapitel 5.4) lässt sich keine Splineinterpolation mit beiden Achsen durchführen. Deswegen sind dem Interface die jeweiligen Daten der x- und y-Achse aufgespalten übergeben und dort wird mit seinen Befehlen sowohl der Spline als auch die Ableitung erstellt.

```
spapi()
fnder()
```

## 6.2 Umsetzung der Algorithmen

Das folgende Kapitel zeigt die Schwierigkeiten und Überlegungen, die während der Implementationen der Algorithmen aufgekommen sind. Da es sich bei den Filterungen und Transformationen oft um eine Iteration durch die Abbildungen in der Kombination mit einfachen bool'schen Abfragen handelt, werden keine Erläuterungen für die Vorgehensweise gegeben, da sie teilweise aus dem Kapitel 6.1 herzuleiten sind. `BodyLinesT` und `JointEstimation` haben im Vergleich einen größeren Umfang und eine höhere Komplexität, da neue interne Strukturen und Verarbeitungsprozesse erstellt werden.

### 6.2.1 `BodyLinesT`

`BodyLinesT` ist die Implementation zur Ermittlung der einfachen Konturlinie (vgl. Kapitel 4.3.1). Dafür sind die Schritte in die folgenden drei Methoden aufgeteilt, deren Einzelheiten in den darauffolgenden Passagen erläutert werden.



```
createVectorgrafic(...);
findBodyPartLineStartingPoints(...);
completeLines(...);
```

Da die Grundbausteine dieser Vektorgrafik Splines sind, muss der Algorithmus die geschlossenen Konturlinien in Abschnitte unterteilen, die sich durch ein solches Objekt beschreiben lassen. Für diesen Prozess wird jede einzelne Konturlinie umrundet, sodass die Punkte in der richtigen Reihenfolge aufgezeichnet sind. Das ist besonders wichtig, denn nur so wird eine annähernde Überdeckung der Pixel gewährleistet. Der erste Punkt wird durch das Starten einer Iteration erhalten und danach wird der restliche Pfad per Linienverfolgung ermittelt. Die abgearbeiteten Positionen werden dabei von der Abbildung entfernt, damit sie kein weiteres Mal aufgezeichnet werden können. Damit sie trotzdem für weitere Bearbeitungen erhalten bleiben, werden sie als ein `EvaluationObject` einer Liste an der vorderster Stelle hinzugefügt (vgl. Abbildung 6.6).

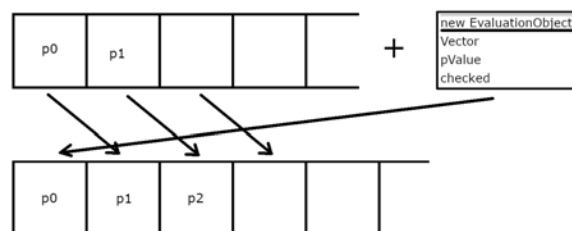


Abbildung 6.6: Hinzufügen eines `EvaluationObject` zu einer Liste

Das hat den Vorteil, dass die Zeigerposition sich nicht für die verschiedenen Abfragen, bei denen mehrere Pixel in bestimmten Abständen verwendet werden, durch die Liste arbeiten muss. So wird zum Beispiel für die Berechnung eines Winkels eines Pixels  $p_i$  die beiden Pixel  $p_{i-K_2}$  und  $p_{i+K_2}$ , mit  $K_2 \in \mathbb{N}$  benötigt. Dementsprechend wird vorausgesetzt, dass die Liste mindestens  $len(liste) = 2 * K_2 + 1$  Elemente beinhaltet. Wegen der Listenaktualisierung, die sich nun am Anfang befindet, müssen die Werte ebenfalls aus dem Bereich entnommen werden (vgl. Abbildung 6.7). Jedoch ist für  $p_{K_2}$  die erste Kalkulation möglich, denn dort werden die Randbedingungen für einen Winkel erfüllt. Der Winkel wird im `EvaluationObject` aktualisiert und ist für die Prüfung nach dem maximalen Wert in der Umgebung bereit.

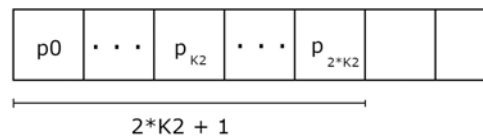


Abbildung 6.7: Elemente zur Berechnung eines Winkels

Hier gilt das gleiche Prinzip wie zuvor, nur dass statt  $K_2$  ein  $K_3 \in N$  verwendet wird. Das bedeutet also, dass  $p_{K_3}$  ein Maximum ist, wenn alle Werte  $V_{p_i}$  die Bedingung erfüllen  $V_{p_i} \leq V_{p_{K_3}}$  mit  $p_0 \leq p_i \leq p_{2*K_3}$ . Wird außerdem der Fakt beachtet, dass die Elemente für die Berechnung einen Wert beinhalten müssen, dann kann ein Offset von  $K_2$  vor dem Bereich eingeführt werden (vgl. Abbildung 6.8). Das würde zu einer kürzeren Laufzeit führen, denn die Überprüfung kann sofort beginnen, sobald es genügend Informationen gibt, andernfalls ist bis zum nächsten Durchlauf zu warten. Unglücklicherweise weichen wegen Implementationsfehler die Ergebnisse von denen der primären Vorgehensweise (vgl. Abbildung 6.7) ab und aus dem Grund wird darauf verzichtet.

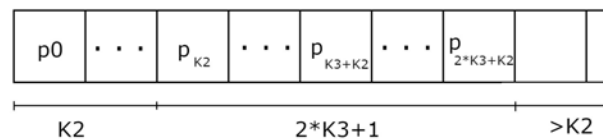


Abbildung 6.8: Elemente zur Ermittlung des maximalen Winkels

Nachdem die Linienverfolgung am letzten Punkt angelangt ist, sind auch keine weiteren Pixel dieser Konturlinie zu finden, aber noch nicht alle haben zu diesem Zeitpunkt einen Winkel errechnet bekommen. Um die Kalkulation fortzusetzen, werden die Elemente am Ende der Liste an den Anfang gesetzt (vgl. Abbildung 6.9) und beendet, wenn ein Maximum ein zweites Mal besucht wird. Mit diesen Daten lassen sich Splines erstellen, denn der Start- und der Endpunkt ergeben sich aus zwei aufeinanderfolgende Maxima und die Splinepunkte werden in gleichmäßigen Abständen aus der dazwischenliegenden Menge entnommen. Damit ist eine Konturlinie zur Vektorgrafik hinzugefügt, um die weiteren Elemente umzuwandeln, wird dieser Vorgang wiederholt, bis es keine Pixel nach einem Iterationsstart zu finden sind.

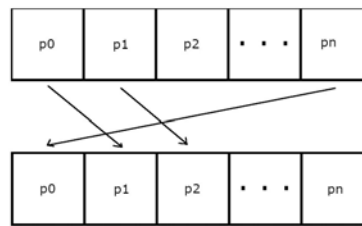


Abbildung 6.9: Rotation der Listenelemente

Nach der Zusammensetzung der Vektorgrafik werden jeweils die Tangenten an den zwei Endpunkten der Splines errechnet, denn an diesen Stellen lassen sich die einfachen Konturlinien finden. Die Suche kann sich nicht allein darauf stützen, weil sich die Linien während der vorherigen Schritte leicht verschoben haben können. Damit sie trotz dessen gefunden werden, wird entlang der Tangente mit einer Toleranz gearbeitet, dadurch entsteht eine Dreiecksfläche entsprechend einem Verhältnis (vgl. Abbildung 6.10). Die Pixel in diesem Bereich haben die Möglichkeit ein Teil der einfachen Konturlinie zu sein, dafür muss das Objekt in die gleiche Richtung wie die Tangente laufen. Die Ausrichtung definiert sich durch die Lage der Endpunkte, die sich noch innerhalb der Toleranzfläche befinden. Ein Winkel gibt Aufschluss, ob es sich dabei um einen Kandidaten handelt. Sollte dies der Fall sein, erfolgt eine Linienverfolgung, bei der auf die Bedingungen der einfachen Konturlinien in Kapitel 3.1.2 geachtet wird.

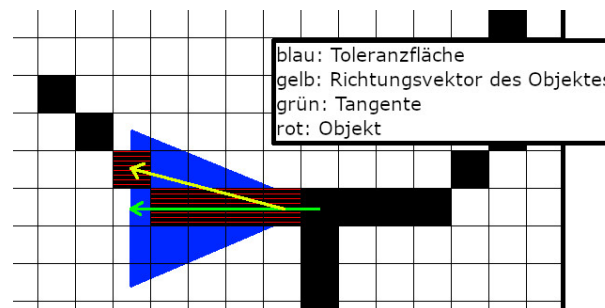


Abbildung 6.10: Toleranzfläche an einer Tangente

## 6.2.2 JointEstimation

Die Klasse JointEstimation verarbeitet alle bisherigen Abbildungen um die Gelenkpositionen zu ermitteln und stellt damit die finale Komponente der Skelettextraktion dar.

Auch sie wird in mehrere Teilschritte getrennt, sodass die Übersichtlichkeit und Verständlichkeit erhalten bleiben.

Der Start wird mit der Erfassung der Knotenpunkte, der Brust und dem Beckenmittelpunktes, eingeleitet, welche mit einer Iteration durch das gesamte Objekt erreichen lässt und gleichzeitig nach dem höchstem Wert gesucht wird. Die örtliche Trennung der beiden Pixel ist hier entscheidend, denn sonst liegen die beiden Punkte in der direkten Nachbarschaft und führt zu einer Pose, die keinem stehendem Menschen entspricht (vgl. Kapitel 4.1). Aus der Algorithmusbeschreibung in Kapitel 4.4.1 ist definiert worden, dass die Punkte nicht innerhalb der euklidischen Distanz  $d_e$  des jeweils anderen Punktes liegen kann. Um die Elemente aus diesem Bereich zu entfernen, wird eine Quadratfläche  $A_q$  genommen, bei der der Knotenpunkt im Schwerpunkt liegt und die Seiten eine Länge von  $a = (d_e - 1) * 2$  besitzen. Wird nun dieselbe Länge  $d_e - 1$  als Radius  $r$  für eine Kreisfläche  $A_k = \pi * r^2$  verwendet, dessen Inhalt gelöscht werden soll, bedeutet es, dass die Schnittmenge  $A_{q-k} = A_q \setminus A_k$  von der Löschung ignoriert werden soll.  $A_{q-k}$  hat die Eigenschaft, dass alle Pixel eine größere Distanz  $d_p$  haben als  $d_e - 1$ , aber da noch ein Rand für die nächsten Schritte übrigbleiben soll, wird folgendes implementiert.

$$(d_e - 1) > d_p \begin{cases} \text{delete}(p) & \text{if } true \\ \text{remain}(p) & \text{if } false \end{cases}$$

Mit der Verwendung der quadratischen Distanz kann der Berechnungsaufwand verringert werden, weil das Ziehen einer Wurzel in den meisten Programmiersprachen ein komplexer Mechanismus ist und beide Werte den gleichen Zweck erfüllen. Aus demselben Grund wird statt der `Math.pow()` - Methode die einfache Multiplikation genutzt.

```
if (((value - 1) * (value - 1)) >
    new Vector(x, y).subtract(greatest).getSqrNorm()) {...}
```

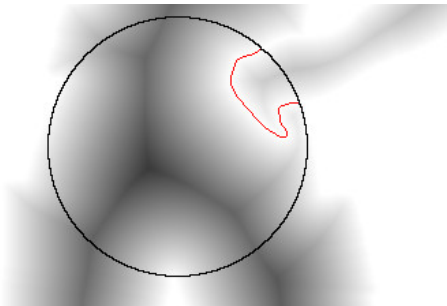


Abbildung 6.11: Ausschnitt der euclidianischen Distanz mit der einfachen Konturlinie

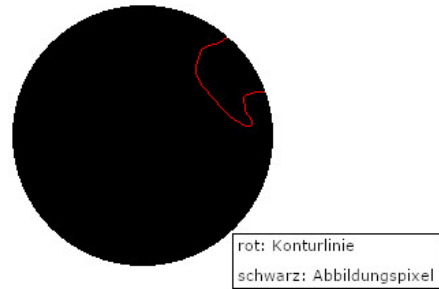


Abbildung 6.12: Tennung der Kreisfläche  $A_k$  durch die einfache Konturlinie

Bei dem Lösungsverfahren für die Distanzabbildung mit den einfachen Konturlinien kann die Kreisfläche  $A_k$  durch die Linie in zwei Gebiete getrennt werden (vgl. Abbildung 6.11 und 6.12). Sie werden in einer Abbildungsliste mit zwei Elementen aufgezeichnet, da sie außerhalb dieser Methoden keine Relevanz mehr hat. Zur Kategorisierung der Pixel in die Abbildung wird ein bool'scher Wert *color* eingeführt, dessen Wahrheitswert den Status des vorangegangenen Pixels angibt. Sie erhält den Wert *true*, wenn es sich um Teile der Konturlinie handelt und sonst ist sie *false*. Nach einem Wechsel von *false* auf *true*, also wenn Pixel mit einer euclidianischen Distanz nach einer einfachen Konturlinie folgt, dann wird ein Zeigerwechsel in der Liste ausgelöst.

```
z = (z + 1) & 1;
```

Da die Differenzierung mit einem zeilenweisen Durchlauf vorgenommen wird, ist sie nur bei vertikal verlaufenden Konturlinien zuverlässig. Horizontale Konturlinien können an beiden Enden an die gleiche Fläche angrenzen, was von dem Algorithmus nicht abgedeckt wäre, da er einen Abbildungswechsel erwartet.



Abbildung 6.13: fehlerhafte Einordnung der Teilflächen in die Abbildungen

Um diesen Fehler zu korrigieren, werden die Informationen aus den Pixel 1,3 und 8

der 8-Nachbarn (vgl. Abbildung 3.2) verwendet, denn durch bestimmte Konfigurationen können folgende Zuordnungskriterien erschaffen werden.

$p_0$  gehört zur *Abbildung*<sub>1</sub>, wenn:

- $p_3$  zu *Abbildung*<sub>1</sub> gehört
- $p_8$  zu *Abbildung*<sub>1</sub> gehört
- $p_1$  zu *Abbildung*<sub>2</sub> gehört UND  $p_3$  und  $p_8$  Konturpunkte sind



Abbildung 6.14: korrekte Einordnung der Teilflächen in die Abbildungen

Nachdem die eindeutige Trennung erfolgreich abgeschlossen ist, wird durch die Position des Knotenpunktes entschieden, welche Fläche von der euklidischen Distanz entfernt wird. Die Randlinienkonstruktion beginnt mit einem zufälligem Punkt, der als erstes von dem Knotenpunkt in eine der vier Himmelsrichtungen erreicht wird. Von diesem Pixel werden die 8-Nachbarn genommen und so rotiert, dass der Vorgängerpixel bei einem Suchlauf in Uhrzeigerichtung an der letzten Position ist (vgl. Abbildung 6.15 und 6.16). Das Verhalten der Reihenfolge bewirkt, dass das erste Objekt mit einer euklidischen Distanz der Nachfolgapixel ist. Dieser Vorgang wird solange wiederholt, bis beide Randlinien für die Maximalbestimmung (vgl. Kapitel 4.4.2) bereitstehen.

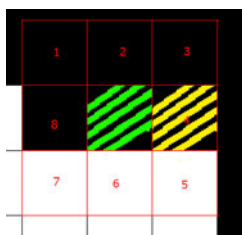


Abbildung 6.15: 8-Nachbarn ohne Vorgängerrotation

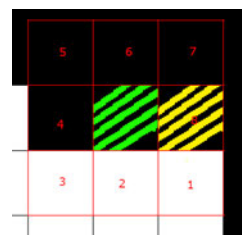


Abbildung 6.16: 8-Nachbarn mit Vorgängerrotation

Die daraus resultierenden Anbindungspunkte können durch eine Sortierung und Auswahl des niedrigsten Wertes reduziert werden. Außerdem wird eine Entfernungsmessung zu allen Punkten von der Strecke zwischen den Knotenpunkten vorgenommen, um zu entscheiden, ob diese ebenfalls gelöscht werden müssen (vgl. Kapitel 3.2.4). Der Parameter  $n$  wird zur Berechnung der Stelle verwendet, an dem die Orthogonale zur Distanzmessung ansetzt. Ist der Wert in dem Intervall  $0 < n < 1$ , dann befindet sich der Pixel in der Fläche zwischen den beiden Knotenpunkte. Obwohl es theoretisch nicht möglich ist, dass sich mehr als zwei Punkte dort befinden können, werden die zwei Punkte mit der kürzesten Distanz entfernt und übrig bleiben die Anbindungspunkte.

Die Erweiterung der Körperteile läuft entlang der Maximallinie der euclidischen Distanz. Das natürliche Verhalten einfachen Linie ist es, dass der nächste Punkt eine größere Entfernung zum Vorgänger hat als der aktuelle Pixel. Diese Information wird verwendet um in den 8-Nachbarn nach gültigen Positionen zu suchen und sie nach ihren euclidischen Werten zu ordnen. Das größte Element wird zu einer Liste hinzugefügt und dieser Vorgang endet, wenn es sich außerhalb des groben Skeletts befindet.

```
neighbours = neighbours.stream().sorted(this::compareTo)
                .collect(Collectors.toList());
. . .
skeletonLine.add(neighbours.get(0));
```

Die Gelenke befinden sich am Ende und auf der Hälfte der Liste und werden anhand des Anbindungspunktes zugeordnet.

### 6.3 ImageConverter

Während der Arbeit werden immer wieder neue Zwischenergebnisse erzeugt, die durch numerische Werte und Positionsangaben in den Abbildungen beschrieben werden. Beim Lesen sind diese Informationen für den Menschen nur schwer zu verstehen. Um es zu vereinfachen, wird ein ImageConverter erstellt, der eine Visualisierung für die Objekte vornimmt. Im Wesentlichen bedeutet es, dass eine Umwandlung für die drei Klassen ImageTransformation, VectorGraphic und Skeleton2D im Bedarfsfall ausgeführt werden. Denn die Bilder sind für die Prozesse keine Notwendigkeit, sondern dienen lediglich zur manuellen Überprüfung der Korrektheit.

Für ImageTransformation und Vectorgraphic werden ihre internen Iteratoren benutzt, um alle möglichen Pixel aufzusammeln und in eine Bitmap zu übertragen. Da es bei der Vectorgraphic keine numerischen Werte zu erwarten sind, bekommen sie den Wert 1. Somit hat sie das gleiche Verhalten wie eine binäre Abbildung in der ImageTransformation und kann als Spezialfall von ihr betrachtet werden. Den Zahlen muss noch eine Farbe zugeordnet werden, damit sie abgebildet werden können. Um ein Graustufenbild zu erstellen, bekommt der höchste Wert in der Abbildung Schwarz zugeordnet. Die restlichen Pixel bekommen einen prozentualen Anteil der Farbintensität, der durch ihren eigenen Wert errechnet wird.

$$\text{Pixel } RGB_i = (r_i : x, g_i : x, b_i : x) \text{ mit } x = V_{p_i}/V_{max} * 255$$

Die Konstruktion des Skeletts ist nur mit Vorwissen möglich, denn sonst können die Gelenke nicht korrekt miteinander verbunden werden. Außerdem enthält Skeleton2D lediglich die Koordinaten der Gelenke und der einzelne Pixel wäre auf einem Bild schwer zu erkennen. Mit der Verwendung einer Kreisfläche wird diesem Punkt eine größerer Bereich zu Verfügung gestellt, die nach dem Skelettplan verbunden werden (vgl. Kapitel 3.1.3).

### 6.4 Qualitätssicherung

In größeren Projekten ist die Softwarequalität ein wichtiger Aspekt, dessen Prüfung sich durch den gesamten Entwicklungsprozess zieht. Einzelne Punkte sind bereits in der Anforderungsanalyse und der Modellierung erwähnt und ausführlich erläutert worden, weswegen sie in diesem Kapitel nur kurz wiederholt und nicht weiter darauf eingegangen werden.

So wird bei der Entwicklung der Datentypen auf hohe Wiederverwendbarkeit gesetzt, die, wie im Falle der ImageTransformation, abbildungsübergreifend genutzt werden soll. Komponenten hingegen sind durch den Einsatz der Interfaces leicht austauschbar und während der Implementierung ist darauf geachtet worden, dass die Methoden verständlich aufgebaut sind. Zwar ist prinzipiell eine gute Performanz wünschenswert, jedoch lässt sich dies nicht in allen Situationen erreichen. Es können bei den Filtertechniken auf die wiederholten Iterationen durch die Abbildungen verzichtet werden, indem viele Klassen



zusammengefasst werden, aber dadurch lassen sie sich nur umständlich austauschen und schwer zu verstehen.

Die Richtigkeit der Software ist ebenfalls ein Merkmal für die Qualität, dessen Sicherstellung durch das Instrument, den Tests, durchgesetzt wird. Die Whitebox-Tests, die eine Analyse des statischen Quellcodes auf Fehler ist, wird in weiten Teilen bereits durch den Java Compiler abgedeckt. Die Überprüfung der korrekten Funktionalitäten gehört zu den Aufgaben der Blackbock-Tests, die in UnitTests und manuelle Tests unterteilt sind. Um einen UnitTest durchzuführen, werden die Eingangsdaten und die Ausgangsdaten benötigt, damit das Verhalten der Methoden verifiziert werden kann. In der Theorie lässt sich das für alle Klassen einrichten, aber die Modellierung der Testumgebungen sind für die die Transformations- und Skelettisierungsalgorithmen zu umfangreich, weshalb ihre Ergebnisse manuell angeschaut werden.

## 7 Evaluation

Nachdem das System vollständig implementiert ist, können weitere Bilder für eine Auswertung der Robustheit hinzugezogen werden. Denn neben dem einen Testbild, das bei der Entwicklung verwendet wurde, muss es natürlich eine breite Menge an Fällen bewältigen müssen. Zu der Menge an Daten werden bearbeitete Bilder genommen, die eine Ähnlichkeit zum Testbild vorweisen und welche die noch andere Überschneidungen von Körperteilen vorweisen. Selbstverständlich entsprechen die Bilder weiterhin den Bedingungen, die in Kapitel 4.1 festgelegt sind.

Zuerst lässt sich feststellen, dass das System zuverlässig Ergebnisse erzielt, wenn es um simple Posen handelt, die keine oder nur kleine Überschneidungen vorweisen. Obwohl die einzelnen Gelenke oftmals nicht ganz ihrer idealen Positionen entsprechen, weil die Verse zum Beispiel in den Zehbereich verschoben wird und dadurch der Knochen teilweise außerhalb des Körpers ist, ändert es die mit dem Skelett verbundenen Pose nicht.



Abbildung 7.1:  
extrahiertes  
Skelett

Gravierendere Auswirkungen auf die Endergebnisse haben dafür Kleinigkeiten, die bei der Entwicklung nicht beachtet wurden, denn Menschen tragen im öffentlichen Leben Klamotten und haben eine Frisur, die ihre eigene Persönlichkeit widerspiegeln sollen. Das ist insofern ein Problem, da sie Positionen und Formen annehmen können, die nicht auf das Skelett zurückzuführen sind. Deshalb können Haare den Kopfbereich stark deformieren, da es sich um ein System handelt, dass in erster Linie mit Silhouetten arbeitet. Zusätzlich fangen ihre Ansätze in der Stirnregion an, wodurch eine klare Abgrenzung der Haarwurzeln an der Kopfhaut, also der Bereich des Schädelknochens, selbst bei menschlichen Begutachtungen schwer zu erkennen sind.

Diese Eigenschaften haben Klamotten auch, dass sie die Figur überdecken. Der Unterschied ist, dass sie in den meisten Fällen, den Körperlinien folgen und somit keinen

großen Unterschied erzeugen. Daraus lässt sich schlussfolgern, dass unnatürliche Linien, wie zum Beispiel Falten in der Hose, zu unerwünschten Resultaten führen. Bei der Ermittlung der Anbindungspunkte kann eine große Falte, die sich in Rumpfbereich befindet, eine falsche Berechnung auslösen, weil die euklidischen Distanzen einen höheren Wert haben. Desweiterem erzeugen sie viele zusätzliche schwarze Linien in der Figur, die als einfache Konturlinie wahrgenommen werden. Steigt die Zahl der Unregelmäßigkeiten, dann existieren mehr Kanten, die bei der Erstellung der Vektorgrafik als Endpunkte der Splines verwendet werden. Die Splines werden demnach immer kürzer und werden von dem Algorithmus miteinander verbunden, damit es genügend Knotenpunkte für die Beschreibung initialisiert werden. Wenn sich nun ein Verschmelzungspunkt zwischen zwei solcher Splines befindet, dann wird er durch diese Verbindung übersprungen und kann nicht gefunden werden.

Neben den Ausbesserungen der Fehlerquellen sind Optimierungsmöglichkeiten vorhanden, die schon bei der Segmentierung der Figur oder nach der Extraktion vorgenommen werden können. Bevor das Bild in jeglicher Weise bearbeitet wird, befindet sich die Figur in einer Szenerie, die auf eine bestimmte Pose hinweist. Ein Stuhl, der sich in direktem Kontakt mit der Figur befindet, deutet stark darauf, dass sich die Person in einer sitzenden Haltung befindet. Für diese Erweiterung wird eine gute Klassifizierung der Objekte vorausgesetzt, die außerdem die manuelle Bearbeitung des Bildes ablöst.

Die erfolgreiche Weiterentwicklung solcher Skelettisierungsalgorithmen hängt von vielen Faktoren ab, die erst durch Analysen und Tests zum Vorschein kommen. Es ist ganz klar interessant, Beziehungen zwischen den Werten zu finden und sie in einem robusten Ermittlungsverfahren einzuarbeiten, dennoch kann der Mensch sehr viele Formen und Farben annehmen, dessen Features durch zusätzliche Software erkannt werden müssen. Bei der Implementierung ist außerdem aufgefallen, dass die Algorithmen sehr an der Erfüllung der Bildbedingungen gebunden sind, wodurch eine flexible Einschätzung fast unmöglich ist. Es lässt sich abschließend zusammenfassen, dass der Aufwand, ein System zu erstellen, das alle Sonderfälle abdeckt, zu groß ist und die Laufzeit keine Konkurrenz zu anderen Anwendungen darstellt. Zusätzlich stehen ihnen die zukunftsorientierten intelligenten Systeme gegenüber, die mit ihren schnellen und zuverlässigen Berechnungen überzeugen.

# Literaturverzeichnis

- [1] AGARWAL, A. ; TRIGGS, B.: Monocular Human Motion Capture with a Mixture of Regressors. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, URL <https://ieeexplore.ieee.org/abstract/document/1565379>, Sep. 2005, S. 72–72. – ISSN 2160-7508
- [2] BIN FAN ; ZENG-FU WANG: Pose estimation of human body based on silhouette images. In: *International Conference on Information Acquisition, 2004. Proceedings.*, URL <https://ieeexplore.ieee.org/document/1373373>, June 2004, S. 296–300
- [3] CHEN, S. ; LI, J. ; WANG, X.: A Fast Exact Euclidean Distance Transform Algorithm. In: *2011 Sixth International Conference on Image and Graphics*, URL <https://ieeexplore.ieee.org/document/6005530>, Aug 2011, S. 45–49
- [4] CHEN, W. ; SUI, L. ; XU, Z. ; LANG, Y.: Improved Zhang-Suen thinning algorithm in binary line drawing applications. In: *2012 International Conference on Systems and Informatics (ICSAI2012)*, URL <https://ieeexplore.ieee.org/abstract/document/6223430>, May 2012, S. 1947–1950. – Skeletonization: Thinning (doing)
- [5] FUJITA, R. ; HAYASHI, T.: Vector image retrieval based on approximation of Bezier curves with line segments. In: *Proceedings of 2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, URL <https://ieeexplore.ieee.org/document/6032932>, Aug 2011, S. 431–436. – new, Bezier. – ISSN 1555-5798
- [6] MAURER, C. R. ; RENSHENG QI ; RAGHAVAN, V.: A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (2003), Feb, Nr. 2, S. 265–270. – URL <https://ieeexplore.ieee.org/document/1177156>. – ISSN 0162-8828

- [7] MORI, G. ; MALIK, J.: Recovering 3D human body configurations using shape contexts. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (2006), July, Nr. 7, S. 1052–1062. – URL <https://ieeexplore.ieee.org/document/1634337>. – ISSN 0162-8828
- [8] MUN WAI LEE ; COHEN, I.: Proposal maps driven MCMC for estimating human body pose in static images. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*. Bd. 2, URL <https://ieeexplore.ieee.org/document/1315183>, June 2004, S. II-II. – ISSN 1063-6919
- [9] MUN WAI LEE ; COHEN, I.: A model-based approach for estimating human 3D poses in static images. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (2006), June, Nr. 6, S. 905–916. – URL <https://ieeexplore.ieee.org/document/1624355>. – ISSN 0162-8828
- [10] NGUYEN, H. A. ; MEUNIER, J.: Reconstructing 3D human poses from monocular image. In: *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*, URL <https://ieeexplore.ieee.org/document/6310523>, July 2012, S. 1426–1427
- [11] PAL, S. ; BISWAS, P. K. ; ABRAHAM, A.: Face recognition using interpolated Bezier curve based representation. In: *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004*. Bd. 1, URL <https://ieeexplore.ieee.org/document/1286424>, April 2004, S. 45–49 Vol.1. – new, Bezier. – ISSN null
- [12] PENGFEI, W. ; FAN, Z. ; SHIWEI, M.: Skeleton extraction method based on distance transform. In: *2013 IEEE 11th International Conference on Electronic Measurement Instruments* Bd. 2, URL <https://ieeexplore.ieee.org/document/6743120>, Aug 2013, S. 519–523
- [13] POHLAND, T.: *Comic Rekonstruktion - 3D Rekonstruktion eines Comic Charakters*. 2020
- [14] QUOC, P. B. ; BINH, N. T. ; TIN, D. T. ; KHARE, A.: Skeleton Formation From Human Silhouette Images Using Joint Points Estimation. In: *2018 Second International Conference on Advances in Computing, Control and Communication Technology (IAC3T)*, URL <https://ieeexplore.ieee.org/document/8674026>, Sep. 2018, S. 101–105. – Verarbeitet, Joint Estimation

- [15] SEKITA, Iwao ; TORAICHI, Kazuo ; MORI, Ryoichi ; YAMAMOTO, Kazuhiko ; YAMADA, Hiromitsu: Feature extraction of handwritten Japanese characters by spline functions for relaxation matching. In: *Pattern Recognition* 21 (1988), Nr. 1, S. 9 – 17. – URL <http://www.sciencedirect.com/science/article/pii/0031320388900672>. – new, Bezier. – ISSN 0031-3203
- [16] STENTIFORD, F. W. M. ; MORTIMER, R. G.: Some new heuristics for thinning binary handprinted characters for OCR. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13 (1983), Jan, Nr. 1, S. 81–84. – URL <https://ieeexplore.ieee.org/document/6313034>. – ISSN 0018-9472
- [17] ZHANG, T. Y. ; SUEN, C. Y.: A Fast Parallel Algorithm for Thinning Digital Patterns. In: *Commun. ACM* 27 (1984), März, Nr. 3, S. 236–239. – URL <http://doi.acm.org/10.1145/357994.358023>. – Skeletonization: Thinning (done). – ISSN 0001-0782

## Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.“

*Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI*

## Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,


Name: \_\_\_\_\_

Vorname: \_\_\_\_\_

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

### **ComRec: Die Skelettextraktion eines Charakters aus einem Comic Bild**

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

\_\_\_\_\_  \_\_\_\_\_  
Ort Datum Unterschrift im Original