Freie Universität Berlin
Fachbereich Mathematik und Informatik
Takustr. 9
14195 Berlin

Bachelorarbeit

Maschinelles Generieren von impressionistischen Bildern im Stil von Leonid Afremov

Autor:
Jana Cavojska

Gutachter:
Prof. Dr. Daniel Göhring
Prof. Dr. Raúl Rojas

6. Oktober 2015

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, 21. Oktober 2015 _____

Jana Cavojska

Zusammenfassung

In dieser Arbeit wird ein Verfahren vorgestellt und detailliert beschrieben, mit welchem es möglich ist, aus einem Eingabebild ein impressionistisch aussehenden Ausgabebild im Stil des Malers Leonid Afremov zu erzeugen. Dieses Verfahren verwendet als Grundlage ein initiales Clustering des Bildes, und nutzt die Größe, Form und Ausrichtung der Cluster, um die passende Pinselstrich-Art zu wählen und um die Attribute der Pinselstriche zu bestimmen. Ähnlich wie auf Afremovs Bildern werden je nach erwünschtem Detailgrad in einer Bildregion entweder kleine und simple oder große und mehrfarbige Pinselstriche eingesetzt. Zum Schluss werden die Ergebnisse mit einem statistischen Mittel validiert.

Inhaltsverzeichnis

1 Einleitung

1.1 Problembeschreibung

Das Ziel dieser Arbeit ist die Entwicklung und Implementierung eines Algorithmus, welcher, gegeben ein Eingabebild, aus diesem ein impressionistisch aussehendes Ausgabebild im Stil von Leonid Afremov erzeugt.
Dabei sollen sowohl die für den Impressionismus im Allgemeinen geltenden Prinzipien eingehalten werden, als auch die für die Werke Leonid Afremovs spezifischen Charakteristika möglichst gut in das resultierende Bild eingebracht werden.

Für den Impressionismus als Kunstrichtung ist das Darstellen der subjektiven Wahrnehmung des Künstlers das zentrale Problem. So werden z. B. auf einem Gemälde diejenigen Teile einer Szene detailliert dargestellt, welche die Aufmerksamkeit des Künstlers beim kurzen Draufblicken auf die Szene gefesselt haben, solche also, die er als wichtig empfand.
Diese subjektive Komponente der Kunstrichtung scheint schon vom Prinzip her ein maschinelles Nachbauen eines impressionistischen Bildes unmöglich zu machen. Das Einzige, was uns übrig bleibt, wenn man dies trotzdem versucht, ist die malerischen Effekte nachzumachen, die sich aus diesem kurzen Draufblicken und selektiven Wahrnehmen des Bildes ergeben.
Wie dem Kapitel "Impressionismus" zu entnehmen ist, gehört zu solchen Effekten eine Art des Ausmalens des Bildes, bei welcher nicht das gesamte Bild mit dem gleichen Detailgrad gemalt werden darf, sondern mit einem geeigneten Verfahren Bildflächen identifiziert werden müssen, die sich in ihrer Wichtigkeit von den benachbarten Flächen abheben. Solche "wichtigeren" Flächen müssen dann auf eine solche Art ausgemalt werden, dass auf ihnen mehr Details beibehalten werden als auf benachbarten Flächen.

1.2 Problemrelevanz und Ziele

Maschinelle Generierung von Bildern, die nicht die Realität darstellen, sondern sich im Gegenteil um eine Abstraktion dieser bemühen, kann viele Gründe haben. So ist es z. B. für Menschen einfacher, Inhalte anhand von vereinfachten, abstrahierten Skizzen als anhand von komplizierten Abbildungen zu lernen, die den Beobachter mit Details überfluten.
Abstrakte Darstellung bzw. abstrakte Kunst muss jedoch keinem praktischen

Zweck dienen; es kann etwas sein, womit man sich gerne umgibt, um sich daran einfach zu erfreuen. Dies hat den Nebeneffekt, dass abstrakte Kunst zu einem Mittel werden kann, mit Hilfe dessen das Interesse von Menschen an mit konkreten Kunstwerken verwandten Themen geweckt wird: Ein gutes Beispiel hierfür ist Googles neue Methode zur Visualisierung von neuronalen Netzwerken, auch bekannt als "Deepdream" [1]. Ursprünglich dazu gedacht, unser Verständnis des Lernprozesses neuronaler Netzwerke zu verbessern, gewann das Projekt sehr schnell an Popularität und verbreitete sich unter anderem über soziale Netzwerke, da die visualisierten Daten traumähnliche Landschaften entstehen lassen, von welchen Menschen fasziniert waren.
Die Werke Afremovs teilen diese Qualitäten: Sie ziehen den Blick an und fesseln selbst Kunst-Laien durch ihre lebendige Farbpalette und traumhaft schöne Szenerien. In dieser Arbeit soll untersucht werden, ob und inwieweit es möglich ist, einen Algorithmus zu entwickeln, der die Essenz dieser Werke auffängt und wiedergibt, um einen ähnlichen Eindruck beim Beobachter wie beim Anblick der Original-Werke zu hinterlassen.

1.3 Beitrag dieser Arbeit

Der Beitrag dieser Arbeit liegt darin, dass sie als erste versucht, nach impressionistischen Prinzipien vorgehend "wichtige" (siehe Kap. "Problembeschreibung") Flächen im Eingabebild grundsätzlich anders zu behandeln als weniger wichtige Flächen.
So werden "wichtigere" Bildflächen auch mit einem anderen Detailgrad und auf eine andere Art ausgemalt als weniger wichtige Stellen. Zu diesem Zweck werden beim Generieren des Ausgabebildes zwei Arten von Pinselstrichen eingesetzt, die grundsätzlich unterschiedlich sind:

- simple, einfarbige Pinselstriche, die durch ihre Position, Farbe, Größe, Richtung und Form eindeutig definiert sind, so wie man dies aus vielen anderen Arbeiten kennt [2] [3] [4] [5]. Das Verfahren zur Ermittlung optimaler Werte für diese Attribute ist dabei quadratische Regression über den Datenpunkten jedes Bildsegments, welcher durch einen Pinselstrich ersetzt werden soll. Der Einsatz einer Regression zur Ermittlung der Ausrichtung der Pinselstriche ist ebenfalls noch in keiner der untersuchten verwandten Arbeiten vorgekommen.

- mehrfarbige Pinselstriche mit Gradienten zwischen ihren einzelnen Farbkomponenten. Der Einsatz von solchen mehrfarbigen Pinselstrichen ist

ein dritter wichtiger Punkt, in welchem sich diese Arbeit von anderen Arbeiten mit einem ähnlichen Thema unterscheidet.

Kleine Bildsegmente (diese werden als Flächen behandelt, denen der Künstler bei einem Gemälde eine erhöhte Aufmerksamkeit geschenkt hätte) werden mit den simplen Pinselstrichen ausgemalt, die sich in ihrer Form dem Verlauf ihrer zugehörigen Segmente anpassen, um detailreiche Regionen mit möglichst wenig Informationsverlust darzustellen.
Große Bildsegmente werden mit zufällig positionierten mehrfarbigen Pinselstrichen ausgemalt, die innerhalb eines Bildsegments einander sehr ähnlich sind, sich aber stark von Pinselstrichen benachbarter (großer und kleiner) Segmente unterscheiden. Hierdurch gehen die Details innerhalb jedes großen Segments weitestgehend verloren, dies ist jedoch erwünscht, da jedes Segment als ein für die Wahrnehmung atomarer Baustein betrachtet wird und lediglich seine Abgrenzung zu benachbarten Segmenten realisiert werden muss.

2 Grundlagen

2.1 Impressionismus

Mit dem Begriff "Impressionismus" bezeichnet man einen Stil der Kunst, u.a. der Malerei, der sich in der 2. Hälfte des 19. Jahrhunderts[6] in Frankreich ausbildete und sich in viele andere Länder ausgebreitet hat.
Zu den führenden Vertretern gehörten[7]:

- Edouard Manet (1832 - 1883)
- Edgar Degas (1834 - 1917)
- Auguste Renoir (1841 - 1919)

Impressionismus legt den Nachdruck nicht auf die Darstellung der Dinge, wie sie wirklich sind, und wie alle sie wahrnehmen können, sondern im Gegenteil darauf, wie der Einzelne die Wirklichkeit wahrnimmt:

"Was alle haben und erfahren können, das bereichert nicht mehr"[6]

Dabei bleibt die Natur der Hauptgegenstand dieser Wahrnehmung.

Die Art, auf welche sich die Seheindrücke des Einzelnen von denen aller anderen Individuen unterscheiden, könne nur festgehalten werden, indem man die "Dauerbetrachtung meidet", also die Dauer der Betrachtung einer Szene oder eines Gegenstandes verkürzt. Nur, wenn Objekte möglichst rasch gesehen werden, bleiben jene individuellen Tönungen der Erfahrung erhalten; wenn man ein Objekt zu oft wiederholt betrachte, führe dies nur zur Feststellung dessen, was die Mehrheit der Menschen dieser Szene entnehmen würde. Dieses Bestreben nach der Moment-Betrachtung äußerte sich im Impressionismus auf zwei Arten: bei der Form und bei der Farbe.

Umgang des Impressionismus mit der Farbe:
Die streng in sich geschlossenen Grenzen der Farbflächen wurden immer mehr aufgelöst, um sie auf eine solche Art nebeneinander stehend darzustellen, wie sie sich bei raschestem Blick in freier Luft und freiem Licht zeigen.
Wenn man beim Betrachten eines Gegenstandes seine Farbe kurz in den Mittelpunkt stellt, statt den gesamten Gegenstand lange und genau zu untersuchen, so scheinen seine formalen Grenzen zurückzuweichen, zu verschwimmen - genau diesen Effekt wollte man erreichen.
Hierbei spielte das Licht eine ganz besondere Rolle: Es wurde zum verbindenden Element, es überflutete die Dinge und hebte damit feste Grenzen auf. Da es an sich farblos ist, wurde es dargestellt, indem man Gegenstände, auf welche es traf, hell aufleuchten ließ.
An Stellen, an welchen trübe Mischfarben erforderlich gewesen wären, vermied man es, die Farben bereits auf der Farbpalette zu mischen, was zu ihrer Trübung geführt hätte, und setzte sie stattdessen auf der Malfläche knapp nebeneinander, wodurch sie ihre Helligkeit und Schönheit behielten.
Räumliche Tiefe wurde, da nun scharfe Trennlinien zwischen Gegenständen fehlten, auf der Ebene der Farbe dadurch simuliert, dass weiter entfernt liegende Teile des Bildes mit einem größeren Blauanteil gemalt wurden, mit dem Hintergedanken, dass je dicker eine Luftschicht ist, umso bläulicher die Gegenstände in der realen Welt erscheinen, von welchen sie uns trennt. ("Luftperspektive")

Umgang des Impressionismus mit der Form:
Die festen Grenzen alles Körperlichen wurden immer mehr aufgelöst bis völlig negiert, um ein extremes Augenblicks-Erlebnis raschtest bewegter Dinge und Vorgänge beim Betrachter zu erzeugen.
Beim Betrachten unserer Umwelt sehen wir nämlich nie alle Teile einer Ge-

samtszene zum selben Zeitpunkt scharf, sondern wir fixieren nacheinander alle Teilszenen, und setzen sie in unserem Kopf zu einem Ganzen zusammen. Beim kurzen Draufblicken auf eine Szene können gar nicht all ihre Teile gleichzeitig scharf gesehen werden; zu den Rändern des Sehfeldes hin wirkt alles immer mehr verwaschen. Daher werden auch nicht alle Teile eines impressionistischen Gemäldes mit dem gleichen Detailgrad gemalt. [6]

Zusammenfassend kann man sagen, dass das Motiv zurückgetreten ist, und dass Bewegung, Licht und Farbe zu zentralen Problemen wurden, die die Fragen der Komposition, der strengen Form und Statik überklangen.[7]

2.2 Leonid Afremov

Leonid Afremov (geboren am 12. Juli 1955 in Vitebsk, Weißrussland) ist ein russisch–israelischer moderner impressionisticher Künstler, der am häufigsten mit einer Malspachtel und mit Ölfarben arbeitet. Er entwickelte seine eigene, einmalige Technik und Stil, welche nur schwer mit Techniken anderer Maler verwechselt werden könnten. Als Künstler repräsentiert er sich selbst und verkauft seine Werke ausschließlich übers Internet, mit nur sehr wenigen Ausstellungen oder Beteiligung von Händlern und Gallerien.
Bevor Online-Einkäufe zum Alltag wurden, war er ein armer Künstler. Er lebte in Vitebsk, seiner Geburtsstadt, bis 1990. Von 1990 bis 2002 lebte er in Israel, und danach bis 2010 in Boca Raton, Florida. Heute hat er seinen Wohnsitz im populären Urlaubsort Playa del Carmen, Quintana Roo, Mexico, nahe Cancun.
Zu seinen Motiven gehören hauptsächlich Landschaften, Szenen aus der Stadt, Meereslandschaften, Blumen und Portraits. Die meisten seiner Werke sind sehr farbenfroh und politisch neutral. [8]

Wie dem Kapitel “Impressionismus” zu entnehmen ist, ist auf impressionistischen Bildern ein Variieren des Detailgrads zu beobachten. Dies ist auch auf Afremovs Bildern sofort ersichtlich. Während Menschenfiguren und Gegenstände wie Straßenlaternen oder Sitzbänke eher detailliert dargestellt werden, mit kleineren, eher einfarbigen Pinselstrichen, sieht man anstelle von Baumkronen nur Gruppierungen von sehr breiten (mehrfarbigen, rechteckigen) Pinselstrichen, die fast alle Details verschwinden lassen. Die einzige Ausnahme bilden Baumäste, die hier und da zu sehen sind. Generell bleiben

dünne und lange Linien (Baumstämme, Gebäudeecken, Ränder von gepflasterten Gehwegen) gut sichtbar.
Eine andere Art, große Flächen auszumalen, die Afremov gerne verwendet, ist das Malen von vielen ovalen und parallelen Pinselstrichen, die große Flächen befüllen. Dieser Ansatz wird z. B. zur Darstellung von Wasser oder vom Boden gewählt (Gras, Gehwege).
Lichtquellen werden dargestellt, indem helle Pinselstriche der gleichen Farbe wie die Lichtquelle selbst in einer bestimmten Distanz von dieser Quelle platziert werden, inmitten anderer Objekte, oft in die Mitte von Wasserflächen. Die Lichtreflektion, die man in der realen Welt beobachten würde, ist auf seinen Bildern immer stark übertrieben.
Alle verwendeten Farben sind insgesamt sehr hell und lebhaft. Die nachfolgenden zwei Bilder sind gute Beispiele für die genannten Charakteristika:

Abbildung 1: *Gemälde "Sounds of the Fall", Leonid Afremov*

Abbildung 2: *Gemälde "Alley by the Lake", Leonid Afremov*

2.3 Non-Photorealistic Rendering und verwandte Arbeiten

Seit in den 60er Jahren das Feld der Computergrafik entstanden ist, war sein Hauptziel der Fotorealismus. Aus diesem Feld heraus hat sich später ein ein anderes Feld mit einem genau entgegengesetzten Ziel abgespalten: **Non-Photorealistic Rendering (NPR)**. Statt sich um eine exakte Repräsentation der Realität zu bemühen, konzentriert sich NPR darauf, wie man Details abstrahieren und expressive und künstlerische Stile emulieren kann. [9]
Da es in dieser Arbeit darum geht, einen Stil der Malerei nachzuahmen, der eben die Details einer Szene auf eine künstlerische Art abstrahiert, und dabei auf Fotorealismus verzichtet, wäre diese Arbeit wissenschaftlich genau in das Gebiet des NPR einzuordnen.

Es wurden bis heute mehrere Arbeiten im Bereich Non-Photorealistic Rendering geschrieben, die sich spezifisch mit dem Problem des "Painterly Rendering", also der Generierung von gemäldeähnlichen Bildern beschäftigen. Dabei ist zwischen solchen Arbeiten zu unterscheiden, die beim Generieren der Bilder teilweise auf Benutzer-Input angewiesen sind und solchen, die Bilder voll automatisiert erzeugen.

- *Methoden mit Benutzer-Input:*
  Zu dieser Gruppe gehört z. B. **Zhaos** semantisch-orientiertes "From Image Parsing to Painterly Rendering" [10] mit seiner interaktiven Segmentierung, bei welcher der Benutzer angeben soll, welche Bildteile Menschen, Kleidungsstücke, und welche der Hintergrund sind, oder die berühmte Arbeit von **P. Haeberli**, "Paint By Numbers: Abstract Image Representations"[2], deren Ziel das Erzeugen von impressionistischen Bildern ist. Sie geht davon aus, dass ein Gemälde eine Serie von Pinselstrichen ist, die die Attribute Position, Farbe, Größe, Richtung und Form haben. Die konkreten Werte dieser Attribute für jeden Pinselstrich werden in Interaktion mit dem Benutzer festgelegt, und das abhängig davon, wie schnell, wo, in welche Richtung etc. er den Mauszeiger über die "Leinwand" bewegt.

- *Voll automatisierte Methoden:*
  Da meine Arbeit sich damit beschäftigt, Bilder voll automatisiert zu generieren, werde ich mich im Rest der Zusammenfassung auf diese zweite Gruppe konzentrieren.

  Einige NPR-Arbeiten verfolgen das Ziel, ganz konkrete Stilrichtungen der Malerei nachzuahmen, wie z. B. den Kubismus (Montin [9]), Pointillismus (Yang u. Yang [11]), oder Impressionismus (Haeberli [2], Litwinowicz [3], Sparavigna u. Marazzato [12]).
  Andere (Hertzmann [5], Gooch, Coombe und Shirley [13]) wiederum befassen sich entweder mit dem Painterly Rendering im Allgemeinen, d.h. die resultierenden Bilder sollen an keinen konkreten Stil erinnern, oder aber es können durch eine Anpassung der Parameter Ergebnisbilder in einem von mehreren Kunststilen entstehen.

Einige der in diesen Arbeiten verwendeten Grundideen und Vorgehensweisen sind auch für meine Arbeit durchaus relevant und werden von mir zur Lösung einiger der Teilprobleme angewandt, wie z. B. die Idee, dass der allererste Schritt beim Erzeugen des resultierenden Bildes eine Segmentierung des Eigabebildes sein muss, oder dass die Farbpalette vor dem Ausmalen reduziert werden sollte.
Andere jedoch sind aus ganz offensichtlichen Gründen nicht anwendbar, weil nämlich die Problemstellung, mit der sich diese Arbeiten befassen, doch eine etwas andere ist als die in dieser Arbeit.

- *Kunststil-orientierte Arbeiten:*
  So liegt der Fokus beim Generieren **kubistischer Bilder** [9] darauf, saliente Features (für die Wahrnehmung wichtige Features wie Augen, Mund, Ohren des Menschen auf dem Originalbild) zu extrahieren und auf das resultierende Bild auf eine randomisierte Art so zu bringen, dass sich diese Features nicht wiederholen oder überlappen. Die Problematik des Ausmalens des Bildes tritt eher in den Hintergrund.

  Bei der Malrichtung des **Pointillismus** [11] besteht das Gesamtbild aus vielen kleinen Bildpunkten ungefähr der gleichen Größe. Dies ist ein wesentlicher Unterschied zu den impressionistischen Werken Afremovs, die in einigen Bereichen (Personen im Vordergrund) einen viel höheren Detailgrad aufweisen als in anderen (Bäume, weit entfernte Gegenstände). Aus diesem Grund ist bei seinen Werken eine ganz andere Herangehensweise erforderlich als bei pointillistischen Gemälden.

  Nach dem Paper von **Sparavigna und Marazzato** [12] erzeugt man impressionistische Bilder, indem für jeden gewählten Pixel ein zweiter Pixel (innerhalb einer zufällig bestimmten kurzen Entfernung) gewählt wird, und die kreisförmige Umgebung des zweiten Pixels mit der Farbe eingefärbt wird, die der erste Pixel hatte. Dies wird iterativ gemacht, z. B. bis das ganze Bild übermalt wurde. Dieses Verfahren hat ähnlich wie das Pointillismus-Paper den Mangel, dass die generierten Pinselstriche alle sehr gleichberechtigt sind; es sind keine Bereiche im Bild erkennbar, die sich durch einen höheren Detailgrad auszeichnen als der Rest.

Litwinowicz [3] geht beim Erzeugen impressionistischer Bilder folgendermaßen vor:

1. Auf das Eingabebild wird ein gausscher Filter angewandt
2. Es werden Farbgradienten berechnet und Bildkanten deteziert
3. Pinselstriche werden generiert, indem sie von ihrem Startpunkt entlang des berechneten Gradienten in beide Richtungen verlängert werden, bis entweder auf eine Kante gestoßen wird, oder die maximale Pinselstrichlänge erreicht ist

Dies produziert sehr schöne Ergebnisse, die einen handgemalten Eindruck machen. Jedoch ist die Behandlung der Kanten für die Zwecke dieser Arbeit wiederum viel zu präzise und viel zu perfekt.

– *Arbeiten, die sich nicht nach einem Kunststil orientieren:*

Das **Hertzmann**-Paper "Painterly Rendering with Curved Brush Strokes of Multiple Sizes" [5] geht auf den Bedarf nach Pinselstrichen unterschiedlicher Größen durchaus ein. Das resultierende Bild wird hier in mehreren Durchgängen gemalt; es wird mit großen Pinselstrichen angefangen, und an Stellen, an welchen sich das so entstandene Bild allzu sehr vom Originalbild unterscheidet, wird das Bild iterativ mit immer kleiner werdenden Pinselstrichen präzisiert.

Die Arbeit "Artistic Vision: Painterly Rendering Using Computer Vision Techniques" von **Gooch, Coombe und Shirley** [13] nimmt zunächst ein Clustering des Ausgangsbildes vor, und approximiert dann für jeden Cluster seine mediale Achse, die die Richtung des Pinselstrichs führen wird, mit welchem der Cluster ausgemalt wird. (Die mediale Achse eines Clusters ist eine Linie, die entlang der Länge und ungefähr durch die Mitte des Clusters führt, und den Cluster somit beschreibt.) Von dieser Methode wurde der in dieser Arbeit verwendete (etwas einfachere) Ansatz zum Ausmalen sehr kleiner Regionen inspiriert: für jede auszumalende Region wird seine Regressionslinie ermittelt, und der entsprechende Pinselstrich wird entlang dieser Linie gelegt. Siehe Kapitel "Schritt 2 - Generieren der Pinselstriche".

Ein weiteres Mittel, von welchem [13] Gebrauch macht, ist die Verwendung von Depth Maps. Die Depth Map eines Bildes liefert Informationen darüber, welche Bildflächen im Vordergrund stehende Gegenstände enhalten, und welche Flächen den Hintergrund darstellen. Die weiter im Vordergrund stehenden Gegenstände werden feiner segmentiert als diejenigen im Hintergrund. Interessanterweise haben Depth Map verwendende Bilder einen weniger impressionistischen Charakter als diejenigen, die auf Depth Maps verzichten, was sich jedoch dadurch erklären ließe, dass die Depth Maps nicht nur zur Bestimmung der Feingranulierung der Segmente genutzt werden, sondern auch um die Richtung der Pinselstriche zu leiten, was auf impressionistischen Bildern so nicht vorzufinden ist. Dennoch ist die Idee der feineren Segmentierung der im Vordergrund stehenden Objekte hoch interessant und stellt eine mögliche Antwort auf die Frage dar, welche Flächen in einem Bild mit einem höheren Detailgrad ausgemalt werden sollten.

Einen ganz anderen Ansatz verfolgt das vor kurzem erschienene Paper von **Gatys, Ecker** und **Bethge** [14], welches ein faltendes neuronales Netzwerk (**Convolutional Neural Network**, CNN) verwendet. In einem CNN kann man die Neuronen jeder Schicht als eine Reihe von hintereinander geschalteten Filtern verstehen, von welchen jeder auf andere Features im Eingabebild reagiert.
Die o. g. Arbeit verwendet CNNs, um aus dem Eingagebild (z. B. ein Foto) ein Ausgabebild zu produzieren, welches mit seinem Stil an den Stil ganz konkreter malerischer Kunstwerke erinnert. Dies wird erreicht, indem das Netzwerk zum einen aus dem Eingabebild seine "Content Representation" isoliert, also seinen eigentlichen Inhalt, und zum anderen aus dem Kunstwerk, welches einem als Stil-Vorlage für das Ausgabebild dient, seine "Style Representation" isoliert.
Dabei sind unter "Content Representation" die Reaktionen der Neuronen der höheren Schichten des Netzwerks auf das Eingabebild zu verstehen, also die Daten, für derer Gewinnung man CNNs typischerweise verwendet.
Für die Extraktion der "Style Representation" eines Bildes wird dagegen ein Feature-Raum verwendet, der aus den Reaktionen der Neuronen einer jeden Schicht zusammengesetzt wird und aus Kor-

relationen zwischen den unterschiedlichen Filter-Reaktionen besteht. Aus Feature-Korrelationen unterschiedlicher CNN-Schichten erhält man Informationen über das Bild, die seine Textur beschreiben, aber nicht seine globale Anordnung.
Im Ausgabebild wird die "Content Representation" des Eingabefotos mit der "Style Representation" des Eingabe-Kunstwerks kombiniert.

2.4 Bildsegmentierung

In Arbeiten, in welchen das Thema dieses Kapitels im Hauptfokus liegt, wird zwischen den Begriffen "Segmentierung" und "Clustering" unterschieden. Da aber diese Unterscheidung stellenweise problematisch sein kann und für die Zwecke dieser Arbeit unwichtig ist, werden hier beide Begriffe als gleich bedeutend behandelt.

Die Segmentierung eines Bildes kann als eine Unterteilung des Bildes in disjunkte, homogene Regionen definiert werden. Solche Regionen enhalten i. d. R. ähnliche Bildobjekte oder Objektteile. Das Ausmaß der Homogenität der segmentierten Regionen (Ähnlichkeit der Pixel einer jeden Region zu anderen Pixeln derselben Region) wird mit Hilfe einer vorher bestimmten Bildeigenschaft gemessen, wie z. B. der Pixelfarbe. Bildsegmentierung ist ein fundamentaler Schritt, welcher gewöhnlich am Anfang komplexerer Computer Vision-Verfahren steht. [15]
Durch Bildsegmentierung entstandene Regionen sollten gleichzeitig die Eigenschaft haben, dass die in ihnen enthaltenen Pixel auch vom Menschen als zusammen gehörend wahrgenommen werden.

Es gibt zahlreiche Verfahren, mit welchen dies erreicht werden kann. Unterschiedliche Verfahren betrachten jeweils unterschiedliche Segmentierungskriterien als ihre Priorität; daher sind einige Verfahren für die Zwecke dieser Arbeit besser geeignet als andere.
Zu Kriterien, die ein Segmentierungsalgorithmus erfüllen sollte, können gehören [16]

- Adhäsion zu Regionsgrenzen innerhalb des Bildes (d. h. Grenzen der Segmente überdecken sich mit tatsächlichen Grenzen der Bildregionen)

- Die Größe und / oder Anzahl der Segmente in einem Bild sollte einstellbar sein
- Der Algorithmus sollte schnell sein, vor allem wenn er als ein Pre-Processing-Schritt vor weiterer Bildbearbeitung verwendet wird

Für die Zwecke dieser Arbeit ist ein Segmentierungs-Algorithmus erwünscht, der sich stark an das erste genannte Kriterium hält. Die Einstellung der Anzahl der Regionen ist z. B. unwichtig. Hohe Geschwindigkeit wäre natürlich erwünscht, aber nicht zwingend erforderlich.

- In einem Versuch, einen solchen Algorithmus zu finden, habe ich einen simplen, iterativen, Graph-basierten Algorithmus implementiert, bei dem ich mich von dem aus der Bioinformatik bekannten Neighbor-Joining-Algorithmus [17] zum Erzeugen phylogenetischer Bäume inspirieren ließ: Ich modellierte alle Bildpixel als Graphknoten, und zwischen jeden Knoten und seine (maximal 4) direkten Nachbarn habe ich jeweils Kanten gesetzt. Jede Kante zwischen den Pixeln $p1$ und $p2$ mit den RGB-Werten $(p1_r, p1_g, p1_b)$ und $(p2_r, p2_g, p2_b)$ hatte das Gewicht g:

  $$g = abs(p1_r - p2_r) + abs(p1_g - p2_g) + abs(p1_b - p2_b)$$

  Nach dem Neighbor-Joining Prinzip habe ich alle Kanten nach ihren Gewichten aufsteigend sortiert, und iterativ immer das Pixelpaar mit dem kleinsten Kantengewicht zu einem Cluster gemerged, solange noch Kantengewichte existierten, die unter einem fest gewählten, konstanten Threshold lagen. Nach einem solchen merge-Schritt mussten für den neu entstandenen Cluster jeweils die Distanzen zwischen diesem neuen Cluster und all seinen Nachbarn berechnet werden. Die Menge der Nachbarn des neuen Clusters ergab sich dabei aus einer Vereinigung der Nachbarmengen der zwei alten Cluster, die gemerged wurden, und die neuen Distanzen wurden immer zwischen dem Pixelrepräsentanten des neuen Clusters und dem Repräsentanten eines Nachbarn berechnet. Die folgende Graphik ist ein Beispiel für die Resultate dieses Algorithmus. Zum Einfärben der Cluster wurden zufällige Grautöne verwendet.

Abbildung 3: *Eigener Graph-basierter Clustering-Algorithmus*

Dieses Verfahren hatte den Nachteil, dass die entstandene Segmentierung stellenweise viel zu fein war (zu viele kleine Cluster in nach der menschlichen Wahrnehmung sehr homogenen Regionen), oder viel zu grob (viel zu große Segmente, die wichtige Details verschwinden ließen). Außerdem waren die Segmentgrenzen viel zu uneben. All dies sind Anzeichen eines viel zu simplen Clustering-Verfahrens gewesen, welche sich nur mit hohem Post-Processing-Aufwand, wenn überhaupt, beseitigen ließen.

Ideal wäre ein Algorithmus, der in der Lage ist, Objekte als Regionen zu erkennen wie etwa eine Baumkrone, eine Wiese, das Hemd eines Menschen, ein Baumast etc. Da mein Clustering-Algorithmus noch einen sehr weiten Weg vor sich gehabt hätte, bis er diese Anforderungen erfüllt, entschied ich mich, nach einem bereits existierenden Clustering-Verfahren zu greifen. Algorithmen, die meinen Zielen nahe gekommen sind, sind z. B. SRM (Statistical Region Merging) [18], oder das Verfahren von Felzenszwalb [19], da beide Verfahren innerhalb hochfrequenter Bildbereiche anders clustern als innerhalb sehr homogener Bereiche. (So wird z.B. nicht jeder Grashalm zu seiner eigenen Region, sondern die gesamte Wiese bildet eine Region, weil die Wiese als ein hochfrequenter Bereich erkannt wurde.)

- **SRM (Statistical Region Merging)** [18] geht folgendermaßen vor: Es wird über alle adjazenten Pixelpaare iteriert, und die zwei Regionen $R1$, $R2$, in welche die zwei Pixel $p1$, $p2$ des aktuellen Pixelpaares gehören, werden genau dann zu einer Region zusammengefasst, wenn das statistische Prädikat P, welches die Ähnlichkeit dieser Regionen misst, $True$ zurückliefert. wobei

$$P(R1, R2) = \begin{cases} true & \text{if } \forall a \in \{R, G, B\} : \\ & |R2_a - R1_a| \leq \sqrt{b^2(R1) + b^2(R2)} \\ false & \text{otherwise} \end{cases}$$

  wo $\{R, G, B\}$ den drei Farbchannels "rot, grün, blau" entsprechen und b eine Konstante ist.
  Folgendes Bild stellt die Resultate dieses Verfahrens dar:

Abbildung 4: *SRM-Clustering [18]*

- **Felzenszwalbs Paper** "Efficient Graph-Based Image Segmentation" [19] fängt mit einer initialen Segmentierung an, in welcher jeder Pixel in sein eigenes Segment (Komponente) gehört. Es werden zunächst alle Kanten zwischen benachbarten Pixeln (also Pixelpaare) aufsteigend nach ihrem Kantengewicht sortiert. Es wird über diese sortierten Pixelpaare iteriert, und falls für das aktuelle Pixelpaar v_i, v_j gilt,

dass das Gewicht $w(v_i, v_j)$ klein ist im Vergleich zu der internen Differenz $MInt(C1, C2)$ beider Komponenten $C1, C2$, zu welchen die zwei betrachteten Pixel gehören, werden diese Komponenten zusammengefasst. Diese interne Differenz wird berechnet als

$$MInt(C1, C2) = min(Int(C1) + \tau(C1), Int(C2) + \tau(C2))$$

$Int(C1)$ ist die interne Differenz zwischen Pixeln der Komponente $C1$ und ist definiert als das größte Kantengewicht e im minimalen Spannbaum der Komponente $C1$ mit der Kantenmenge E:

$$Int(C1) = max(w(e)), e \in MST(C1, E)$$

τ ist die Threshold-Funktion, welche das Ausmaß kontrolliert, in welchem die Differenz zwischen zwei Komponenten größer sein muss als ihre internen Differenzen Int, damit sie vom Algorithmus tatsächlich als zwei unterschiedliche Komponenten gesehen werden (also die Evidenz für eine Grenze zwischen ihnen existiert).

$$\tau(C1) = \frac{k}{|C1|}$$

wo $|C1|$ der Größe von $C1$ entspricht und k eine Konstante ist.

Abbildung 5: *Felzenszwalb-Clustering [19]*

- Eher schlechter geeignet wären sog. Superpixel-Methoden wie **SLIC (Simple Linear Iterative Clustering)** [16]. Die Regionen, die durch diese Verfahren produziert werden, erinnern an große Pixel: sie sind alle ungefähr gleich groß und mehr oder weniger rund. Damit diese Größen- und Formanforderung überhaupt eingehalten werden kann, kommt es öfter dazu, dass eine Grenze zwischen zwei optisch unterschiedlichen Bereichen mitten durch einen Superpixel durchläuft, und nicht entlang seiner Grenze. Selbst wenn man die Superpixelgröße klein genug wählen würde, dass dies nicht passiert, stünde man nachher immer noch vor dem Problem, diese Superpixel weiter gruppieren zu müssen, welches die weiter oben genannten Verfahren nicht haben.

Abbildung 6: *SLIC-Superpixel [16]*

3 Der Algorithmus

3.1 Überblick über den Algorithmus

1. Bildsegmentierung mit dem Felzenszwalb-Algorithmus
2. Als Hintergrund für das Ausgabebild wird entweder das segmentierte Bild oder das saturierte und verwischte Eingabebild verwendet
3. Abspeicherung der Bildsegmente in einer Liste S, absteigend nach Größe sortiert
4. ```
   for segment in S:
       if segment.size < threshold:
           paintSmallSegment(segment)
       else:
           paintLargeSegment(segment)
   ```

Die Funktion `paintSmallSegment()` ist im Kapitel "Simple, einfarbige Pinselstriche" beschrieben, die Funktion `paintLargeSegment()` im Kapitel "Komplexe, mehrfarbige Pinselstriche".

3.2 Schritt 1 - Bildsegmentierung

Von den vorgestellten Algorithmen hat sich der Felzenszwalb-Clustering-Algorithmus erwiesen. Siehe Unterkapitel "Bildsegmentierung" im Kapitel "Grundlagen".

3.3 Schritt 2 - Generieren der Pinselstriche

Afremovs spezifischer Malstil (Malen mit der Spachtel) schließt nicht nur existierende Malprogramme aus, sondern auch Rendering-Algorithmen aller für diese Arbeit untersuchten Paper. So haben z. B. Pinselstriche, mit welchen Baumkronen auf jedem Bild ausgemalt werden, typischerweise eine gerade Unterkante, eine Oberkante, die mehrere Ausbuchtungen und Dellen aufweist, und eine Krümmung, die sich nach oben hin abschwächt, nicht unähnlich einer Logarithmus-Kurve. Sowohl entlang des Pinselstrichs als auch querverlaufend sind oft mehrere Farbgradienten zu beobachten.
   ```

Das Programm unterscheidet, wie auch Afremovs Werke selbst, zwischen zwei Arten von Pinselstrichen: simple zum Ausmalen von detailreichen Flächen (`paintSmallSegment()`), und mehrfarbige, viereckige, zum Ausmalen von großen Flächen wie Baumkronen, Wiesen, Gebäudewänden... (Funktion `paintLargeSegment()`).

3.3.1 Simple, einfarbige Pinselstriche

In detailreichen Regionen des Eingabebildes erkennt der Clustering-Algorithmus eine größere Anzahl von kleineren Segmenten als in weniger detailreichen Regionen. In dieser Arbeit wird jeder solche kleine Cluster mit einem gekrümmten Pinselstrich ausgemalt, dessen Krümmung durch eine Linie geleitet wird, die durch polynomielle Regression 2. Grades über alle Pixel dieses Segments berechnet wird.

Polynomielle Regression
Polynomielle Regression ist ein Verfahren, bei dem solche Koeffizienten für die Polynomgleichung

$$p(x) = p_0 \cdot x^{deg} + ... + p_{deg}$$

vom Grad *deg* gesucht werden, die den quadratischen Fehler ("mean squared error", MSE) minimieren.[20]

$$MSE = \sum_{j=0}^{k} |p(x_j) - y_j|^2$$

wobei jedes x_j der x-Koordinate eines Pixels entspricht, und y_j der zugehörigen y-Koordinate dieses Pixels auf diesem Bild.
Es wird also nach einem Polynom gesucht, dessen Funktionskurve eine möglichst kleine Abweichung zu allen Pixeln hat, und das Segment somit am besten beschreibt.
Im Gegensatz zur linearen Regression gibt es keine geschlossene Formel zur Berechnung dieser Koeffizienten; sie werden iterativ ermittelt.[21]

Auf dem folgenden Lena-Bild wurde durch jedes erkannte Segment seine Regressionslinie gelegt und gelb gefärbt:

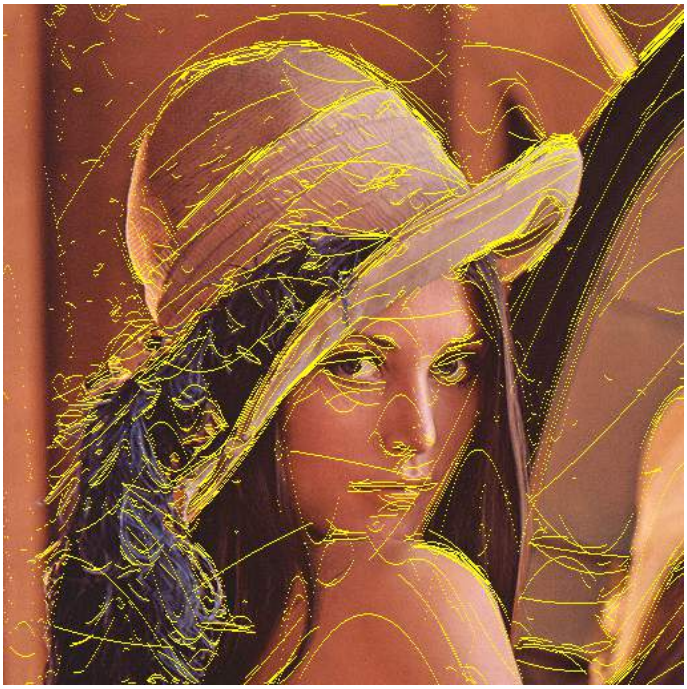

Abbildung 7: *Lena, Regressionslinien*

Auf den folgenden zwei Abbildungen wurde jeder Pinselstrich wie ein "simpler" Pinselstrich gemalt. Man sieht, dass die quadratische Regression für die größeren, konkaven Regionen eine eher schlechte Approximation liefert, und für kleinere eine etwas bessere. Der einzige Unterschied zwischen den folgenden zwei Abbildungen ist die Breite der verwendeten Pinselstriche. In der ersten Abbildung wird als Breite jedes Pinselstrichs der Wert

$$max\left(1, \frac{Anzahl\_Pixel\_im\_Segment}{Pinselstrichlaenge}\right)$$

verwendet, in der zweiten die Breite der rotierten Bounding Box des Segments. (Eine rotierte Bounding Box einer Region ist das kleinste Rechteck, in welches diese Region reinpasst. Eine nicht-rotierte Bounding Box ist achsenparallel, eine rotierte ist so ausgerichtet, dass sie die kleinstmögliche Fläche einnimmt.)
Auf beiden Lena-Bildern wurde beim Ausmalen mit den größten Segmenten angefangen.

Abbildung 8: *Lena, dünne, simple Pinselstriche*

Abbildung 9: *Lena, breite, simple Pinselstriche*

Vorgehen:
Jeder simple Pinselstrich wird durch das Nebeneinanderlegen von senkrechten, von oben nach unten wachsenden Pixellinien gemalt und anschließend rotiert und an die richtige Stelle im resultierenden Bild eingefügt. Der Winkel *angle*, um welchen der Pinselstrich rotiert wird, entspricht dem Winkel zwischen der Strecke r, die die Endpunkte der Regressionslinie verbindet, und einer senkrechten Linie, die durch den weiter unten liegenden der beiden Endpunkte der Regressionslinie r verläuft:

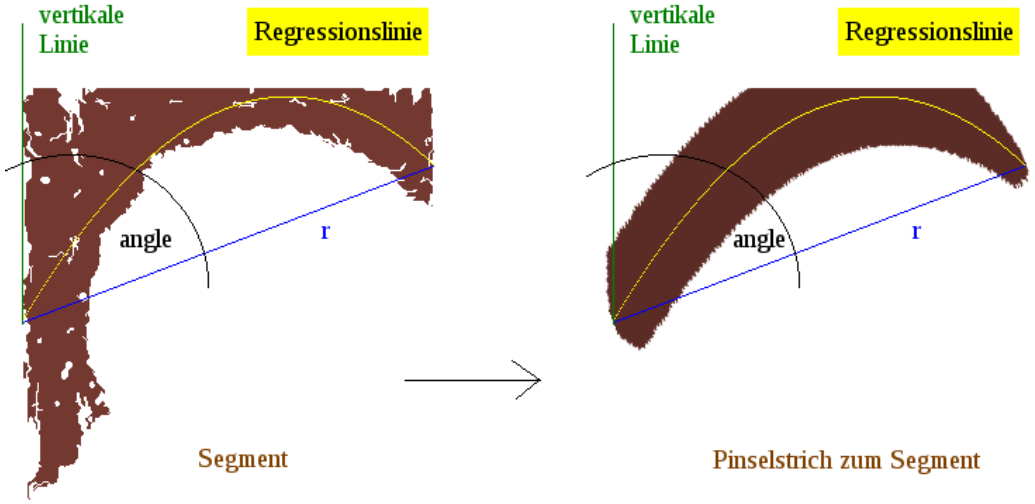

Abbildung 10: *Simpler Pinselstrich*

Form des simplen Pinselstrichs:
Biegung:
Für jede x-Koordinate x_i des Pinselstrichs wird eine dünne (Pixelbreite 1), sich nach unten ziehende Pixellinie eingefärbt, die bei x_i anfängt. Die y-Koordinaten y_j dieser Pixellinie nehmen Werte von 0 bis $length(r)$ an. Die x-Koordinaten der Pixellinie sind nicht konstant, was einen geraden, viereckigen Pinselstrich erzeugen würde, sondern haben jeweils einen Abstand von der linken Seite des Pinselstrichs, der für jeden y-Wert y_j eines Pixellinie anders ist, und dem Abstand gleich ist, welchen der y_j-te Pixel der Regressionslinie von der Strecke r hat, die die Endpunkte der Regressionslinie verbindet. Das Array, das die Abweichungen einer Pixellinie von einer geraden, nach unten laufenden Pixellinie enthält, und der somit das Einfärben leitet, wird in dieser Arbeit als *arc* bezeichnet.

Form der Ober- und Unterseite:
Die Ecken der Pinselstriche werden abgerundet, indem von der Ober- und Unterseite ein Teil des Pinselstrichs abgeschnitten wird. Die Angabe darüber, wieviele Pixel bei welcher x-Koordinate abgeschnitten werden, ist in den Arrays *upperTemplate* (für die Oberseite) und *lowerTemplate* (für die Unterseite des Pinselstrichs) gespeichert. Die Werte in jedem dieser Arrays sind ganz einfach die Kosinuswerte für Winkel zwischen -90 und 90 Grad (zum Ausschneiden eines bogenförmigen Bereichs), multipliziert mit einem zufälligen Skalierungsfaktor, um etwas mehr Diversität reinzubringen. Für die Wirkung der Templates siehe folgende Grafik:

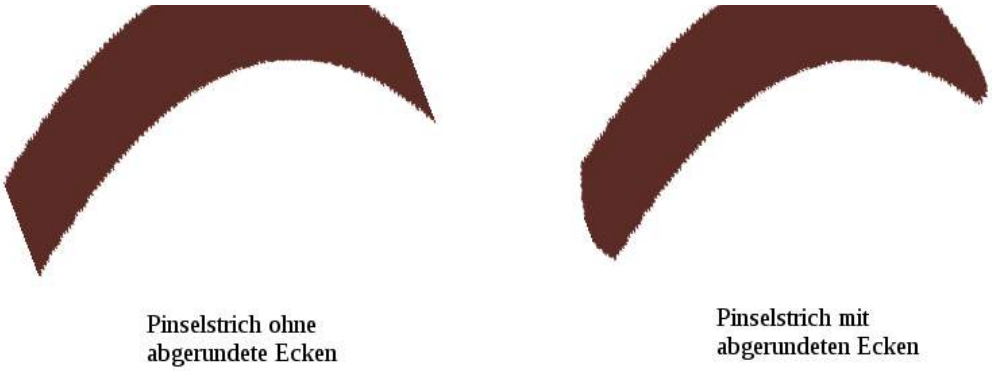

Abbildung 11: *Abrundung der Ecken*

Farben:
Zum Ausfärben des simplen Pinselstrichs wird eine einzige Farbe verwendet.

Position des simplen Pinselstrichs:
Der Pinselstrich wird so in das resultierende Bild eingefügt, dass die Regressionslinie, welche das zum Pinselstrich gehörige Segment im Eingabebild beschreibt, auch den eingefügten Pinselstrich im Ausgabebild gut abstrahieren würde. Dies wird erreicht, indem beim senkrecht ausgerichteten Pinselstrich der horizontale Mittelpunkt seiner oberen (oder unteren) Kante ermittelt wird. Nach dem Rotieren um *angle* wird dieser Mittelpunkt erneut gefunden und der nun richtig ausgerichtete Pinselstrich wird an eine solche Position im Ausgabebild gelegt, dass dieser Mittelpunkt sich mit dem Anfang der Re-

gressionslinie deckt. Falls *angle* negativ ist, wird der horizonale Mittepunkt der Pixel der oberen Kante bestimmt, beim negativen *angle* wird die untere Kante genommen. Beim negativen Winkel der Regressionslinie weiß man nämlich, dass der Anfang der Regressionslinie mit dem linken oberen Ende des Segments zusammenfallen wird (dies war eine Design-Entscheidung).

3.3.2 Komplexe, mehrfarbige Pinselstriche

Komplexere, mehrfarbige Pinselstriche haben eine grob viereckige Form und einen Aufbau, der durch die folgende Grafik veranschaulicht wird:

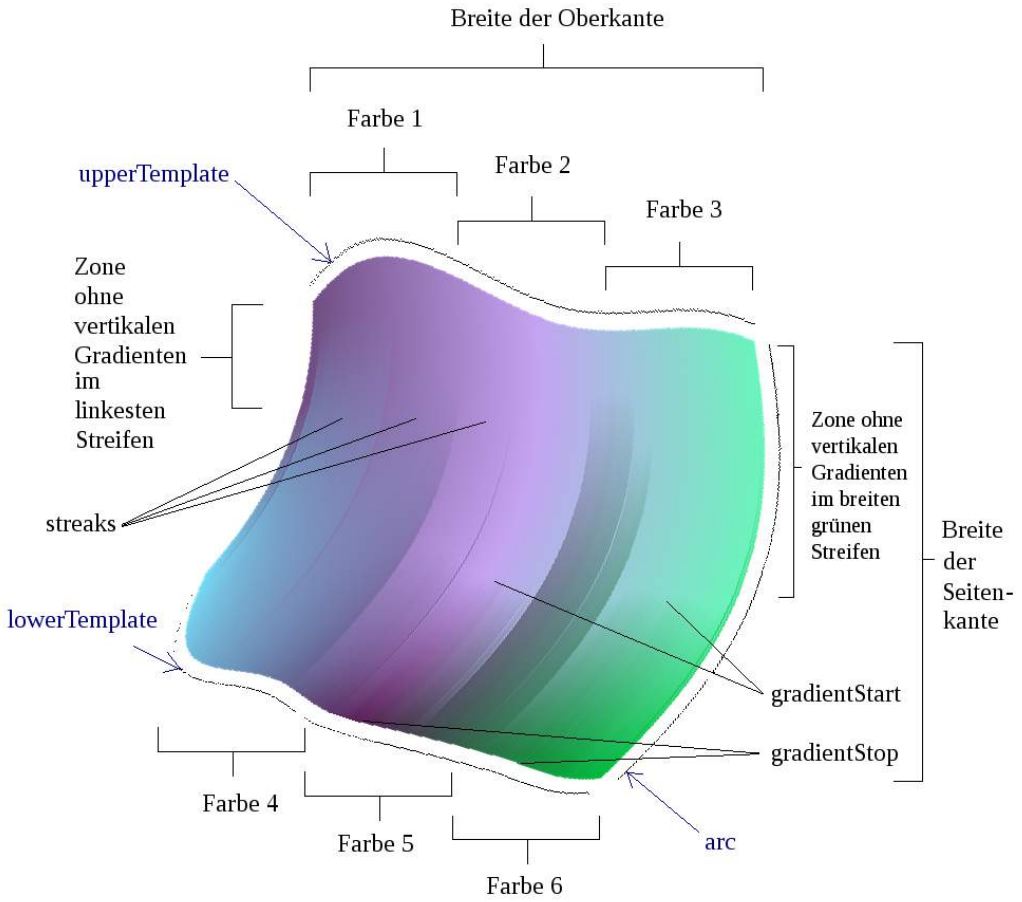

Abbildung 12: *Aufbau eines Pinselstrichs*

Komplexe Pinselstriche werden ebenfalls durch das Nebeneinanderlegen von senkrechten, von oben nach unten wachsenden Pixellinien gemalt, rotiert und an die richtige Stelle im resultierenden Bild eingefügt.

Form des komplexen Pinselstrichs:

Biegung:

Ähnlich wie bei den simplen Pinselstrichen werden auch hier die 1 Pixel breiten senkrechten Pixellinien anhand der Werte im *arc*-Array berechnet, das für jede y-Koordinate ihren Offset von der linken Seite des Pinselstrichs.
Anders als bei den simplen Pinselstrichen ergeben sich die Werte in *arc* aus keiner Regressionslinie, sondern folgen entweder einer Logarithmus- oder Kosinus-Kurve. Dies verleiht jeder vertikalen Pixellinie eine leicht gebogene Form. Die Stärke der Biegung kann durch einen Parameter festgesetzt werden (*bulgeSize*). Der folgende Pseudocode stellt die Erzeugung eines logarithmusförmigen *arc*-Arrays dar. $sizeY$ bezeichnet die vertikale Länge des Pinselstrichs.

```
1:  function createLogArc(sizeY, bulgeSize)
2:      if sizeY = 1 then
3:          return [0]
4:      end if
5:      if sizeY = 2 then
6:          return [0, 0]
7:      end if
8:      arc ← zeros[sizeY]
9:      stepsize ← 10
10:     logparam ← 1.1
11:     i ← 0
12:     while i ≤ length(arc) do
13:         arc[i] ← log_2(logparam)
14:         logparam ← logparam + stepsize
15:         i ← i + 1
16:     end while
17:     arcMax ← max(arc)
18:     arcScale ← bulgeSize / arcMax
19:     arc ← arc · arcScale
20:     return arc
```

21: **end function**

Die folgende Grafik verdeutlicht die Wirkung der Biegung auf den Pinselstrich:

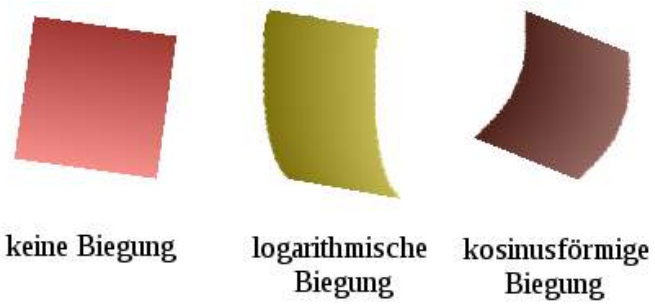

Abbildung 13: *Pinselstrich-Biegung*

Form der Ober- und Unterseite:
Die Wellenform der Ober- und Unterseite wird auch hier durch das Abschneiden des Ober- und Unterteils jedes Pinselstrichs erreicht. Die Angaben zur abzuschneidenden Pixelanzahl für jede x-Koordinate sind in den Arrays *upperTemplate* (für die Oberseite) und *lowerTemplate* (für die Unterseite des Pinselstrichs) gespeichert. Die Werte in jedem dieser Arrays entstehen durch kubische Spline-Interpolation der Werte in einem kürzeren Array, das nur die gewünschten Spitzpunkte (Anzahl der Pixel, die abgeschnitten werden sollen) enthält.

Die *kubische Spline-Interpolation*[22] ist ein Interpolationsverfahren, bei welchem von einer endlichen Menge von Punkten

$$M = [x_i, y_i], \text{ für } i = 0..n$$

ein Polynom gebildet wird, dessen Funktionskurve durch all diese Punkte durchläuft. Dabei wird jeder Abschnitt zwischen zwei (auf der x-Achse benachbarten) Punkten aus M lokal durch seine eigene kubische Polynomgleichung beschrieben, und am Ende werden all diese Teilpolynome zu einem einzigen Polynom verbunden.
Für jedes Intervall existiert eine andere Polynomgleichung, mit ihren eigenen Koeffizienten:

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i,\ x \in [x_i, x_{i+1}]$$

Durch Zusammensetzung all dieser Gleichungen für die Teilabschnitte entsteht eine Gesamtgleichung $S(x)$ mit einer kontinuierlichen Funktionskurve. Dies wird durch das Einhalten der folgenden Bedingungen erreicht:

- Jedes kubische Polynom muss an seinen beiden Enden durch Punkte aus M durchgehen, die es von beiden Seiten beschränken (dies stellt die Kontinuität der Funktion sicher):

$$S_i(x_i) = y_i, \quad S_i(x_{i+1}) = y_{i+1}$$

- Die erste und zweite Ableitung müssen auch kontinuierlich sein (hierdurch entsteht eine optisch zufriedenstellende Kurve ohne spitze Umbruchstellen):

$$S'_{i-1}(x_i) = S'_i(x_i), \quad S''_{i-1}(x_i) = S''_i(x_i)$$

Die Anzahl solcher Spitzpunkte ist momentan auf 5 für *upperTemplate* und 6 für *lowerTemplate* festgesetzt. Die konkreten Werte für die Spitzpunkte sind Zufallszahlen. Für den ersten und den letzten Spitzpunkt werden jeweils größere Zufallszahlen genommen, um an den Ecken mehr vom Pinselstrich abzuschneiden, damit dieser abgerundeter wirkt. Für die Wirkung der Templates siehe folgende Grafik:

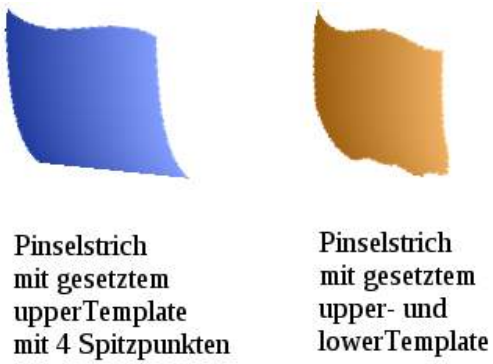

Abbildung 14: *upper- und lowerTemplate*

Farben:
Entlang der Breite des Pinselstrichs kann eine beliebige Anzahl von Farben definiert werden. Diese können für die Oberseite und die Unterseite unterschiedlich sein, auch in ihrer Anzahl.

Breitengradient:
Für alle Farben, die für die Oberseite definiert wurden, wird entlang der Breite des Pinselstrichs mit linearer Spline-Interpolation ein Farbgradient berechnet, der alle definierten Farben enthält und für sanfte Übergänge zwischen ihnen sorgt. Das gleiche gilt für die Farben der Unterseite. Die unten stehende Grafik verdeutlicht, wie es aussehen würde, wenn entlang der Breite kein Gradient berechnet wird, sondern harte Farbübergänge zwischen den Farben am *upperTemplate* untereinander und zwischen den Farben am *lowerTemplate* untereinander verwendet werden.

Längengradient:
Darüber hinaus wird auch entlang der Länge des Pinselstrichs ein Gradient berechnet, nämlich für den Übergang zwischen der Farbe, die an der oberen Kante bei einem konkreten x-Wert anliegt, und der Farbe, die an der unteren Kante anliegt, an der Stelle, die mit dem obigen x-Wert durch *arc* verbunden wurde.
Beide Pinselstriche in der folgenden Grafik bestehen aus drei zufällig generierten Farben am *upperTemplate* und drei Farben am *lowerTemplate* und verwenden Längengradienten zwischen allen drei Farbenpaaren, jedoch nur der zweite Pinselstrich verwendet auch Breitengradienten.

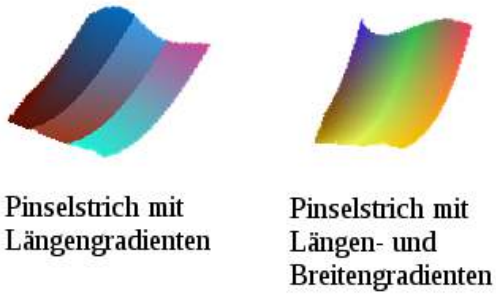

Abbildung 15: *Farbgradienten*

Farbstreifen:
Um noch bessere Ergebnisse zu erreichen, ist dieser Längengradient nicht einheitlich, sondern fängt für jeden x-Wert bei einem anderen Abstand von der Oberkante an. Dieser Abstand ist zwar randomisiert, aber für benachbarte x-Werte nimmt er mit hoher Wahrscheinlichkeit sehr ähnliche Werte an, wodurch entlang des Pinselstrichs deutlich sichtbare Streifen (in Abbildung 12"streaks" genannt) entstehen. Diese Streifen haben eine Pixelbreite zwischen 1 und einem Maximalwert, der ein Viertel der sichtbaren Pinselstrichbreite ("Breite der Oberkante" in Abbildung 12) beträgt. Mit einer Wahrscheinlichkeit von $0.5^{(excess)}$ können sie jedoch um *excess* viele Pixel breiter werden.
Wenn der Abstand des Gradientenanfangs von der Oberkante des Pinselstrichs > 0 ist, entsteht für diesen x-Wert im Bereich dieses Abstands eine Zone ohne vertikalen Gradienten (siehe Abbildung 12). Da der erwähnte Abstand für Werte des gleichen Farbstreifens ähnlich sind, sich aber stark von den Abständen der benachbarten Streifen unterscheidet, entsteht auf diese Art eine gut sichtbare Grenze zwischen einzelnen Streifen.

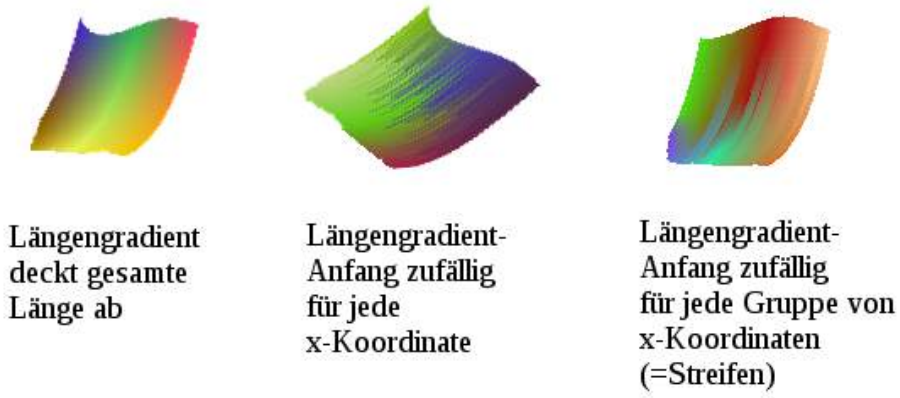

Abbildung 16: *Farbstreifen*

3D-Effekt:
Bei einigen Pinselstrichen auf Afremovs Bildern ist an der Unterkante eine Anhäufung von Farbe zu beobachten, die sich als ein dünner heller Streifen entlang der Unterkante des Pinselstrichs äußert, begleitet durch einen noch dünneren dunklen Streifen unterhalb von dem hellen:

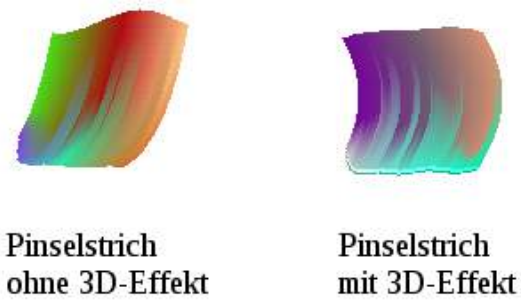

Abbildung 17: *3D-Effekt*

3.3.3 Farben der Pinselstriche

Noch vor dem Clustering-Schritt wird die Farbsättigung des Eingabebildes auf das Dreifache erhöht. Das auf diese Art entstandene gesättigte Bild dient als Farbgrundlage für das Ausgabebild.

Jeder simple, einfarbige Pinselstrich wird mit der Farbe eingefärbt, die der Pixel des gesättigten Bildes hat, bei welchem die Regressionslinie dieses Segments anfängt. Da alle Pixel desselben Segments relativ ähnliche Farben haben, produziert diese simple Methode zufriedenstellende Ergebnisse.

Jeder komplexe Pinselstrich wird unter Angabe von vier Farben generiert. Die Färbung des ganzen Pinselstrichs ergibt sich aus der Interpolation der Farbwerte zwischen diesen vier Eingabefarben (zwei $startColors$, zwei $endColors$. Siehe Kapitel "Komplexe, mehrfarbige Pinselstriche") Als Quelle für diese Farben werden vier Pixel verwendet, die sich auf dem gesättigten Bild in der Umgebung der vorgesehenen Koordinate für diesen Pinselstrich befinden. Somit spiegelt der komplexe Pinselstrich grob die Färbung der unter ihm liegenden Pixel wider, was eine ungefähre Erkennung von Strukturen im Bild ermöglicht.

3.4 Schritt 4 - Ausmalen des Bildes mit generierten Pinselstrichen

Die Grundidee des Ausmalens des Bildes besteht darin, über eine nach der Größe absteigend sortierte Liste aller Felzenszwalb-Segmente zu iterieren und jedes Segment mit den zu den Segmentattributen passenden Pinselstrichen auszumalen.

3.4.1 Hintergrund

Sowohl beim Produzieren echter Gemälde als auch bei ihrem maschinellen Generieren [12] ist es üblich, das Ausgabebild mit einer Hintergrundfarbe auszumalen, bevor die Pinselstriche aufgetragen werden, die am Ende sichtbar bleiben. Beim maschinellen Generieren von Bildern kann der Hintergrund eine einzige Farbe sein (weiß, schwarz, die auf dem Originalbild am häufigsten vorkommende Farbe), oder das Originalbild selbst (entweder 1:1 übernommen oder verschwommen).

Bei diesem Algorithmus hat man die Möglichkeit, entweder das verschwommene ("blurred") Eingabebild als Hintergrund zu nehmen, oder das gefärbte, mit dem Felzenszwalb-Verfahren geclusterte Bild, in welchem jeder Cluster mit der Farbe eines zufälligen Pixels aus diesem Cluster gefärbt wurde. Beide Möglichkeiten führen zu optisch zufriedenstellenden Ergebnissen.

3.4.2 Große Segmente

Segmente, deren Größe (Anzahl der Pixel, aus welchen sie bestehen) über einer bestimmten Grenze liegt, werden mit multiplen, mehrfarbigen Pinselstrichen ausgemalt. Dazu müssen zunächst für jedes Segment die Positionen ermittelt werden, an welchen die komplexen Pinselstriche zu platzieren sind. Der folgende Pseudocode beschreibt diesen Prozess. $minX$, $minY$, $maxX$ und $maxY$ sind die Koordinaten der nicht-rotierten Bounding Box des Segments, *segment* ist eine Liste aller Koordinaten des Segments, und *stroke_density* ist die Größe der Schritte, in welchen nach passenden Koordinaten gesucht wird.

```
1: function getStrokePositions(segment, stroke_density, minX, minY,
   maxX, maxY)
2:     coordinates ← []
3:     x ← minX
4:     while x ≤ maxX do
5:         y ← minY
6:         while y ≤ maxY do
7:             if [x, y] ∈ segment then
8:                 coordinates.append([x, y])
9:             end if
10:            y ← y + stroke_density
11:        end while
12:        x ← x + stroke_density
13:    end while
14:    return randomize(coordinates)
15: end function
```

In der Praxis produziert dieses Verfahren bessere Ergebnisse, als über *segment* zu iterieren und mit einer bestimmten Schrittgröße dort Koordinaten auszuwählen. Das gewählte Verfahren hat den Effekt, dass wenn eine potentielle Koordinate untersucht wird, die doch nicht zum *segment* gehört, um diese herum ein größerer Raum frei gelassen wird (da die nächste Koordinate, die untersucht wird, *stroke_density* viele Pixel entfernt ist, und nicht in direkter Nachbarschaft).

Komplexe Pinselstriche innerhalb eines Segments haben immer die gleiche Größe, sowie eine sehr ähnliche Ausrichtung (±40 Grad). Komplexe Pinselstriche anderer Regionen können eine ganz andere Ausrichtung haben, und je nach Parametern, mit welchen der Algorithmus gestartet wird, auch eine andere Pinselstrichgröße. Dies dient einer besseren optischen Unterscheidung der Bildsegmente.

3.4.3 Kleine Segmente

Jedes Segment, dessen Größe unter einer bestimmten Grenze liegt, wird mit einem einzigen, einfarbigen Pinselstrich ausgemalt. Sie sorgen dafür, dass feine Umrisse von Gegenständen sichtbar sind, sowie kleine Regionen, die sich

durch ihre Farbe und andere Eigenschaften stark von ihrer Umgebung abhoben (und somit einem menschlichen Beobachter auffallen würden), weswegen sie vom Clustering-Algorithmus als eigene Segmente erkannt wurden. Jeder simple Pinselstrich ist in seiner Farbe, Position, Form und Ausrichtung eine Approximation des Segments, welches er ausmalt. Für Details siehe Kapitel "Simple, einfarbige Pinselstriche".
Simple Pinselstriche für Segmente, deren Farbe sich nicht allzu stark von der Farbe der Umgebung abhebt, werden gar nicht angefärbt. Der Zweck dieser Maßnahme ist das Eliminieren von überflüssigen kleinen Segmenten, die zwar durch das Clustering erkannt wurden, jedoch einem menschlichen Beobachter wahrscheinlich kaum als eigenständige Regionen auffallen würden. Da diese Segmente nicht ausgemalt werden, wird ihre Fläche entweder durch einen komplexen Pinselstrich überdeckt (der zu einem benachbarten großen Segment gehört), oder diese Fläche bleibt dank des Hintergrundbildes sichtbar (welches eine abgewandelte Form des Eingabebildes ist).
Kleine Segmente werden nach großen gemalt, um von diesen nicht überdeckt zu werden.

3.4.4 Haarlinien-Segmente

Eine dritte Gruppe von Pinselstrichen bilden haardünne Segmente, deren Breite oder Länge 1 Pixel beträgt. Diese werden nicht approximiert, sondern direkt in das Bild gemalt. Sie kommen selten genug vor, dass sie die Illusion eines Gemäldes gut ergänzen und wichtige Details unterstreichen, ohne allzu störend zu wirken.
Haarlinien-Segmente werden als letzte gemalt, um von den größeren nicht überdeckt zu werden.

4 Laufzeitanalyse

Die Gesamtlaufzeit des in dieser Arbeit vorgestellten Algorithmus liegt in $O(n \cdot log(n))$, wobei n die Anzahl der Pixel im Eingabebild ist.

Zum Zweck der Laufzeitanalyse könnte man den Gesamtalgorithmus in 2 Hälften teilen:

1. Der Felzenszwalb-Clustering-Schritt [19], der ganz am Anfang ausgeführt wird, hat nach den Angaben in [19] eine Laufzeit von $O(n \cdot log(n))$ mit

n = Anzahl der Pixel

2. Der Rest des Algorithmus, also alles bis auf den Clustering-Schritt, läuft ebenfalls in $O(n \cdot log(n))$.
   Für diese obere Schranke ist der Schritt verantwortlich, in welchem die erkannten Felzenszwalb-Cluster nach ihrer Größe sortiert werden. Die Anzahl dieser Cluster ist zwar nach oben durch die Anzahl der Pixel beschränkt, liegt jedoch in der Praxis deutlich darunter.
   Alle anderen Schritte des Algorithmus haben lineare Laufzeit in der Anzahl der Pixel n, darunter:

   (a) Die polynomielle Regression 2. Grades, die die Leitlinien für die simplen Pinselstriche erzeugt, läuft für die Pixel jedes Felzenszwalb-Clusters in linearer Zeit. Die hierfür verwendete Methode der kleinsten Quadrate macht sich sie sog. SVD (Single Value Decomposition) zunutze, die nach [23] in ihrem ersten Schritt (Reduktion der Eingabematrix auf eine bidiagonale Matrix) in $O(mn^2)$ läuft, wobei m in unserem Fall die Anzahl der Pixel im jeweiligen Cluster ist, und n der Grad der Regression (2 bei quadratischer Regression, damit konstant), und in $O(n)$ in ihrem zweiten Schritt (iterative Berechnung der SVD für die bidiagonale Matrix aus Schritt 1, mit konstanter Präzision).
   Aus der linearen Laufzeit für jeden Cluster, und aus der Tatsache, dass die Cluster disjunkt sind, ergibt sich eine lineare Laufzeit für die polynomielle Regression für das Gesamtbild.

   (b) Die Ermittlung der Stellen, an welchen komplexe Pinselstriche innerhalb von Clustern platziert werden sollen, kann in $O(n)$ realisiert werden, wenn für jeden Cluster die Liste seiner Pixel durchgegangen wird und an Koordinaten in regelmäßigen Abständen der Größe $stepSize$ Pinselstriche platziert werden.
   Ein solcher Ansatz führt aber zu optisch weniger zufriedenstellenden Ergebnissen und hat in der Praxis eine schlechtere Laufzeit als der in dieser Arbeit tatsächlich verwendete Ansatz, der asymptotisch in $O(n^2)$ läuft: Die Positionen für komplexe Pinselstriche werden ermittelt, indem durch alle Koordinaten innerhalb der Bounding Box des Clusters iteriert wird, und für jede $stepSize$-te Koordinate überprüft wird, ob sie zum Cluster gehört. Wenn ja, wird an ihrer Position ein komplexer Pinselstrich gemalt.

(c) Eine letzte potenziell problematische Stelle ist die Interpolation, die zum Erzeugen der Cutoff-Templates für die komplexen Pinselstriche dient (kubische Interpolation), sowie für die Farbübergange innerhalb der komplexen Pinselstriche (lineare Interpolation). Dieser Schritt hat jedoch eine Laufzeit, die linear in der Anzahl der zu interpolierenden Punkte liegt, da hier die Spline-Interpolation verwendet wird.

5 Resultate

Der Algorithmus erreicht relativ gute Ergebnisse bei Bildregionen mit einer hohen Anzahl an **komplexen Pinselstrichen**. Diese zeichnen sich durch eine traumähnliche Atmosphäre aus, die an Afremovs Werke erinnert, nicht nur durch eine Ähnlichkeit der Pinselstrichformen, Farbübergänge und Lagebeziehungen zueinander, sondern auch durch den Gesamteindruck, den sie hinterlassen.

Genauso gut gelungen ist die Anpassung der **Farbpalette**: Die Lebhaftigkeit der Farben, die beim Betrachten der Bilder Afremovs sofort unsere Aufmerksamkeit auf sich zieht, konnte durch eine simple Erhöhung der Farbsättigung erreicht werden.

Die Abbildung von **Wasserflächen** erinnert schon etwas weniger an Afremov, der diese auf eine sehr charakteristische Art malt, mit dünnen, parallelen und leicht unregelmäßigen Pinselstrichen, wobei blaue und hell leuchtende Flächen im starken Kontrast gegenüber gestellt werden. Die von diesem Algorithmus verwendeten Verfahren führen zwar zu Resultaten, die gut genug sind, dass Wasserflächen als solche erkannt werden (und damit zum Gesamteindruck eines impressionistischen Bildes beitragen), aber damit es möglich wäre, dass diese unmissverständlich wie von Afremov gemalt aussehen, wäre es erforderlich, Wasserflächen als solche zu erkennen und gesondert zu behandeln, was den Rahmen dieser Arbeit sprengen würde.

Etwas weniger zufriedenstellend ist die Darstellung von detailreichen Regionen, insbesondere **Personen** gelungen. Im Rahmen dieser Arbeit wollte ich untersuchen, ob es möglich wäre, hinreichend kleine Regionen durch Pinselstriche zu approximieren, deren Parameter sich aus den Eigenschaf-

ten dieser Regionen ableiten (eine Idee, auf die mich die Arbeit von Gooch, Coombe und Shirley [13] gebracht hat, nur ist meine Approximation durch Regression eine viel simplere gewesen). Meine Annahme war, dass sich hinreichend kleine Segmente immer gut genug durch quadratische Regression all ihrer Pixel approximieren lassen (und dass durch die Wahl eines passenden Clustering-Algorithmus die Entstehung hinreichend kleiner Segmente sichergestellt werden kann). Diese Annahme hat nur teilweise zum Ziel geführt: Die Approximation funktioniert meistens gut genug, dass zumindest Personen als solche auf dem resultierenden Bild zu erkennen sind. Jedoch versagt sie dabei, Personen handgemalt aussehen zu lassen. Sowohl die kantigen Haarlinien als auch die selbst nach Randomisierung der Umrisse viel zu regelmäßig wirkenden bogenförmigen Pinselstriche sehen doch eher computergeneriert aus. Dies ist ein ähnliches Problem wie das der Wasserflächen; eine Erkennung von Personen und eine Sonderbehandlung von Regionen, die Personen enthalten, könnte dieses Problem lösen.

Zusammenfassend kann man sagen, dass ich das Ziel erreicht habe, welches ich mir für diese Arbeit vorgenommen habe: zu untersuchen, ob und inwieweit es möglich wäre, Bilder, die dem Stil Afremovs ähneln, mit einem Programm zu generieren. Zumindest für eine Untergruppe der Eingabebilder (Landschaften, die möglichst wenige Personen enthalten), ist es über meine Erwartung hinaus gelungen. Die Merkmale, welche auf Afremovs Bildern unsere Aufmerksamkeit als erste auf sich ziehen, sind auch auf den computergenerierten Bildern wiederzufinden: die lebendigen Farben, die breiten, schönen Pinselstriche, die einander völlig zufällig überdecken und einem harmonisch vor den Augen tanzen.
Für Bilder, die Personen enthalten, oder Gegenstände, die einer völlig anderen Behandlung bedürfen als die umgebende Landschaft, ist der hier vorgestellte Algorithmus ohne Anpassungen eher ungeeignet.
In der Gallerie im letzten Kapitel sind einige von diesem Algorithmus produzierte Beispielbilder zu finden.

6 Validierung

In diesem Kapitel werden die Ergebnisse eines simplen Verfahrens präsentiert, welches zur Validierung der Ergebnisse dieser Arbeit gewählt wurde.
Bei diesem Verfahren werden drei Gruppen von Bildern mit einem auf der

Shannon-Entropie basierenden Verfahren miteinander verglichen, und es wird untersucht, ob nach diesem Verfahren einige dieser Gruppen eine größere Ähnlichkeit zueinander aufweisen als andere.

6.1 Shannon-Entropie

Die Shannon-Entropie beschreibt den Informationsgehalt bzw. die Unsicherheit eines Ereignisses oder einer Nachricht durch eine rationale Zahl, welche aussagt, wie viel Information in dieser Nachricht enthalten ist.
In der Informationstheorie ist die Shannon-Entropie ein Maß dafür, wie unsicher der Ausgang einer Situation ist, in welcher ein Ereignis aus einer Menge von möglichen Ereignissen gewählt wird.
Je unwahrscheinlicher ein solches Ereignis, umso mehr Information bringt es, wenn es vorkommt. Entropie kann man also auch als ein Informationsmaß verstehen.[24]
Shannon definierte die Entropie H einer diskreten Zufallsvariable X mit den möglichen Werten $\{x_1, ..., x_n\}$ und der Wahrscheinlichkeitsfunktion $P(X)$ als:

$$H(X) = -K \sum_i [P(x_i) \cdot log(P(x_i))]$$

wobei

x_i	- ein distinktes Ereignis in unserer Verteilung
$P(x_i)$	- Wahrscheinlichkeit, mit welcher Ereignis x_i vorkommt
K	- positive Konstante

In unserem Fall ist jeder mögliche Farbwert zwischen 0 und 255 ein solches Ereignis x_i, und die Wahrscheinlichkeit $P(x_i)$, dass ein solches Ereignis vorkommt, wird berechnet als

$$P(x_i) = \frac{count(x_i)}{imsize}$$

wobei $count(x_i)$ die Anzahl der Vorkommen des konkreten Farbwertes x_i im Bild ist, und $imsize$ Anzahl der Bildpixel. Die Konstante K wird gleich 1 gesetzt.

6.2 Durchführung des Experiments

Die untersuchten Bilder wurden in drei Gruppen unterteilt:

1. Gruppe 1: Fotos, die als Eingabebilder für den Algorithmus in dieser Arbeit verwendet wurden.

2. Gruppe 2: Bilder, die von dem in dieser Arbeit vorgestellten Algorithmus produziert wurden (mit Bildern aus der ersten Gruppe als Eingabe).

3. Gruppe 3: Fotos von Gemälden Afremovs, die in der Bildergallerie auf seiner offiziellen Webseite[25] veröffentlicht wurden.

Vorgehen:
Für jede dieser drei Gruppen wurden 4 Zahlenwerte berechnet:

- *rEntropy* - durchschnittliche Entropie des rot-Channels aller Bilder einer Gruppe

- *gEntropy* - durchschnittliche Entropie des grün-Channels aller Bilder einer Gruppe

- *bEntropy* - durchschnittliche Entropie des blau-Channels aller Bilder einer Gruppe

- *grEntropy* - durchschnittliche Entropie aller Graubilder einer Gruppe

Das Experiment wurde mit jeweils 912 Bildern in jeder der 3 Gruppen durchgeführt. Um ihre Entropie-Werte vergleichbar zu machen, wurden alle Bilder auf die gleiche Größe gebracht, 500x500 Pixel.

6.3 Resultate des Experiments

	Gruppe 1	**Gruppe 2**	**Gruppe 3**
rEntropy	7.38119516389	7.27519618364	7.80140402048
gEntropy	7.40860351111	7.42822126323	7.71538216276
bEntropy	7.19426102764	6.71987228677	7.22865966951
grEntropy	7.32791091292	7.34646118526	7.73412071998

6.4 Interpretation der Resultate

Gruppe 3: Es fällt auf, dass die Gruppe der Afremov-Gemälde in allen untersuchten Farbchannels die größte Entropie aufweist. Dies ließe sich dadurch erklären, dass Afremov eine breite Skala an Farben verwendet, von welchen keine viel häufiger vorkommen als andere (im Vergleich zu Fotos von Landschaften zumindest). Die Reduktion der Farben, die dadurch zustande kommt, dass innerhalb jedes Pinselstrichs nur ganz wenige Farben vorkommen, scheint hier also eher in den Hintergrund zu treten.
Gruppen 1 und 2: Einen Anstieg der Entropie-Werte beim Übergang von den Eingabefotos der 1. Gruppe zu den Ausgabebildern der 2. Gruppe kann man im Grün-Channel und im Grau-Bild beobachten, im Rot- und Blau-Channel dagegen sieht man einen Abstieg. Da es sich größtenteils um Fotos von Landschaften handelt, in welchen der Grün-Anteil überwiegt, könnte man die Ergebnisse so beschreiben, dass zumindest der für die Wahrnehmung wichtigste der drei Channels sich in die richtige Richtung zu entwickeln scheint.

Abschließend muss man aber sagen, dass all diese beobachteten Unterschiede nur ganz marginal sind, und dieses Verfahren sich somit nur beschränkt zur Untersuchung der Unterschiede der drei gewählten Bildergruppen eignet.

7 Ausblick

Wie im Kapitel "Resultate" geschildert, besteht vor allem bei der Behandlung von Personen und Wasserflächen Verbesserungspotential. Es wäre interessant, zu sehen, was man erreichen könnte, wenn man die Ergebnisse dieser Arbeit mit Verfahren kombinieren würde, die sich speziell mit diesen Problemen beschäftigen. Bei Erkennung von Personen oder Objekten würden sich z. B. neuronale Netzwerke anbieten, und beim Ausmalen von Personen Verfahren, die hierzu besser geeignet sind, weil sie präziser arbeiten als die in dieser Arbeit verwendete quadratische Regression (z. B. die Technik von Gooch, Coombe und Shirley [13], oder die von Hertzmann [5]).

8 Literatur

[1] Alexander Mordvintsev, Christopher Olah und Mike Tyka. *Inceptionism: Going Deeper into Neural Networks.* URL: `http://googleresearch.`

blogspot.de/2015/06/inceptionism-going-deeper-into-neural.html.

[2] Paul Haeberli. "Paint By Numbers: Abstract Image Representations". In: *Proceedings of SIGGRAPH 90*. Bd. 24. 4. 1990.

[3] Peter Litwinowicz. "Processing Images and Video for an Impressionist Effect". In: *Proceedings of SIGGRAPH 97*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997. ISBN: 0-89791-896-7.

[4] A. Atencia u. a. *Scalable Impressionist Rendering*. Université Paris. 2001.

[5] Aaron Hertzmann. "Painterly Rendering with Curved Brush Strokes of Multiple Sizes". In: *Proceedings of ACM SIGGRAPH 98*. 1998.

[6] Max Deri. *Die Malerei im XIX. Jahrhundert*. Bd. 1. Berlin: Paul Cassirer, 1919.

[7] Doris Wild. *Moderne Malerei*. Zürich: Büchergilde Gutenberg, 1950.

[8] *Leonid Afremov Bio*. URL: http://afremov.com/Leonid-Afremov-bio.html.

[9] Henri Montin. *Cubist Cameras*. The University of Bath. 2008.

[10] Kun Zeng u. a. "From image parsing to painterly rendering". In: *ACM Trans. Graph.* 29.1 (2009).

[11] Hui-Lin Yang und Chuan-Kai Yang. *A Non-photorealistic Rendering of Seurat's Pointillism*. National Taiwan University of Science and Technology, Taipei, 106, Taiwan, ROC.

[12] Amelia Carolina Sparavigna und Roberto Marazzato. *Non-photorealistic image processing: an Impressionist rendering*. Politecnico di Torino, Italy.

[13] Bruce Gooch, Greg Coombe und Peter Shirley. *Artistic Vision: Painterly Rendering Using Computer Vision Techniques*. University of Utah, University of North Carolina at Chapel Hill.

[14] Leon A. Gatys, Alexander S. Ecker und Matthias Bethge. *A Neural Algorithm of Artistic Style*. University of Tübingen. 2015.

[15] Swagatam Das und Amit Konar. "Automatic Image Pixel Clustering with an Improved Differential Evolution". In: *Appl. Soft Comput.* 9.1 (2009).

[16] Radhakrishna Achanta u. a. “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.11 (2012).

[17] Naruya Saitou und Masatoshi Nei. *The Neighbor-joining Method: A New Method for Reconstructing Phylogenetic Trees*. Center for Demographic and Population Genetics, The University of Texas. 1987.

[18] Richard Nock und Frank Nielsen. “Statistical Region Merging”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.11 (2004).

[19] Pedro F. Felzenszwalb und Daniel P. Huttenlocher. “Efficient Graph-Based Image Segmentation”. In: *Int. J. Comput. Vision* 59.2 (2004).

[20] *numpy.polyfit, Numpy and Scipy Documentation*. URL: `http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html`.

[21] *Nonlinear Least Squares Regression. NIST/SEMATECH e-Handbook of Statistical Methods*. URL: `http://www.itl.nist.gov/div898/handbook/pmd/section1/pmd142.htm`.

[22] *Cubic Spline Interpolation. Department of Physics and Astronomy, The University of Utah*. URL: `http://www.physics.utah.edu/~detar/phys6720/handouts/cubic_spline/cubic_spline/node1.html`.

[23] Lloyd N. Trefethen und III David Bau. *Numerical Linear Algebra*. USA, Philadelphia: Society for Industrial und Applied Mathematics, 1997.

[24] C. E. Shannon. “A Mathematical Theory of Communication”. In: *The Bell System Technical Journal* XXVII.3 (1948). URL: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6773024`.

[25] *Leonid Afremov Homepage*. URL: `http://afremov.com`.

9 Gallerie

Abbildung 18: *Haus in Dahlem, Original*

Abbildung 19: *Haus in Dahlem*

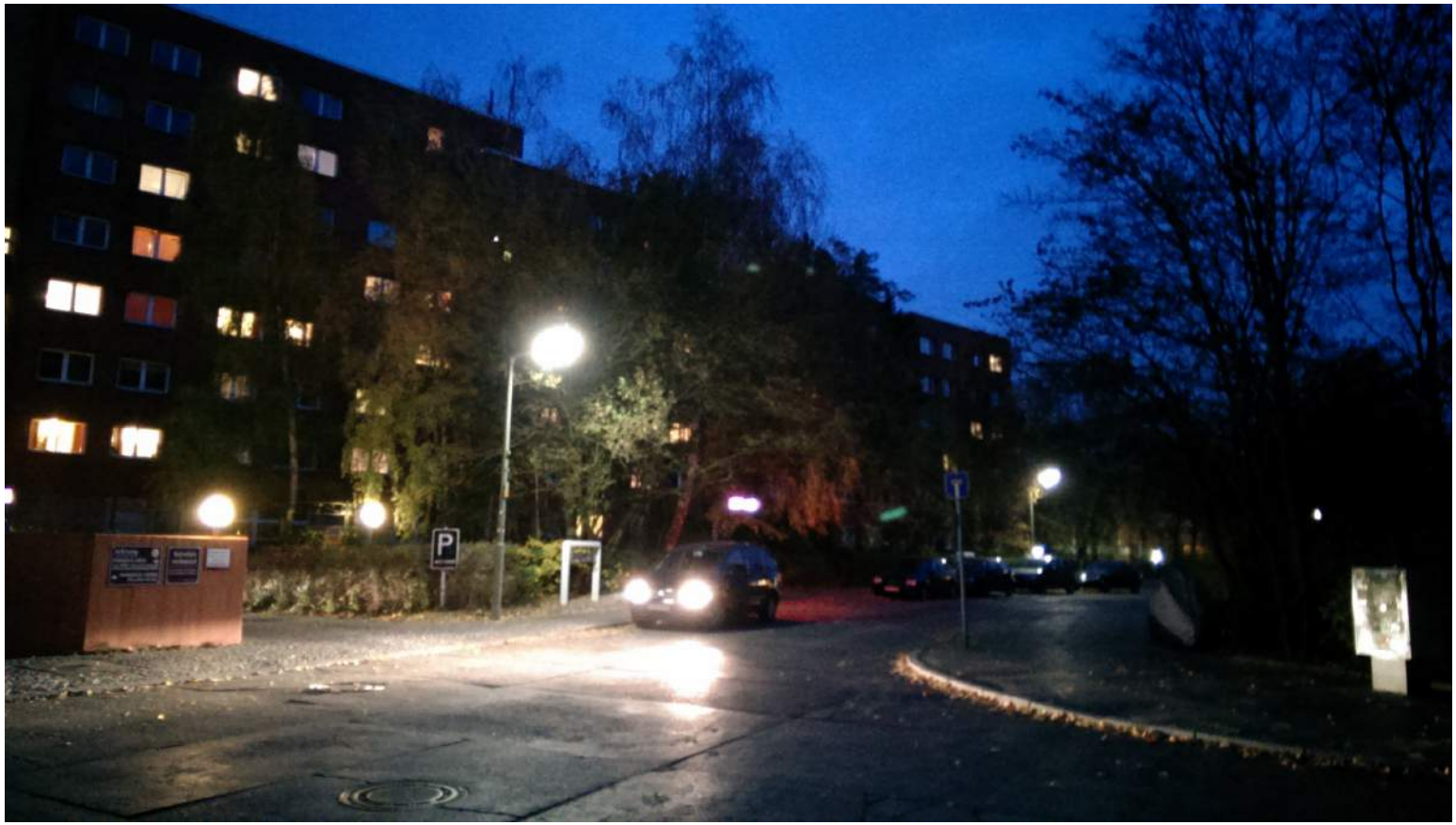

Abbildung 20: *Parkplatz*

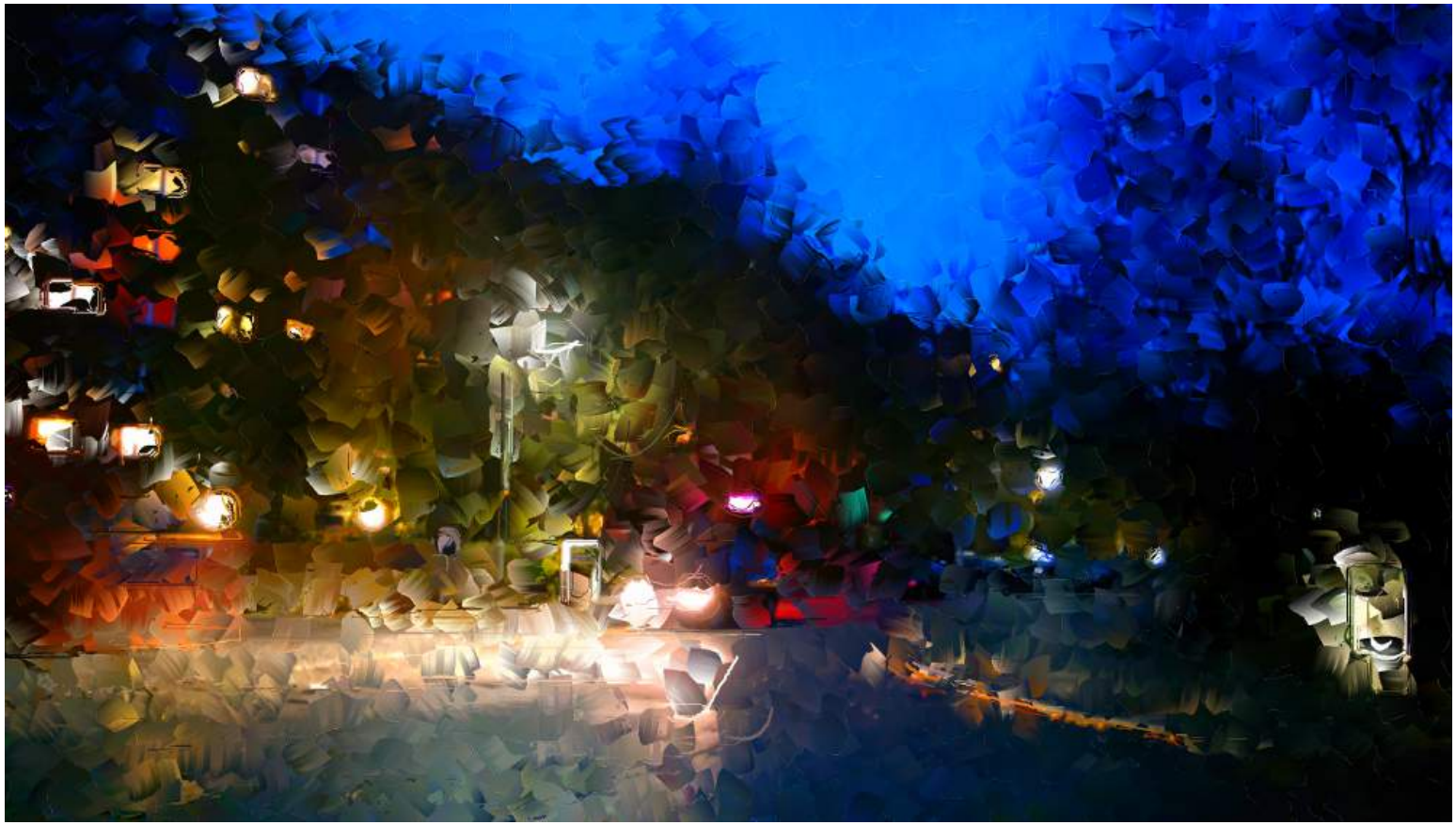

Abbildung 21: *Parkplatz*

Abbildung 22: *Im Park, Original*

Abbildung 23: *Im Park*

Abbildung 24: *Teufelssee, Original*

Abbildung 25: *Teufelssee*

Abbildung 26: *Denisa, Original*

Abbildung 27: *Denisa*

Abbildung 28: *Abend in Brno*