Fakultät
Gestaltung
Würzburg

ART(IFICIAL) BACHELORTHESIS TITUS EBBECKE

Art(ificial)

Erzeugung maschineller Kunst durch Machine Learning basierte Steuerung eines Industrieroboters

BACHELORTHESIS TITUS EBBECKE

Erzeugung maschineller Kunst durch Machine Learning basierte Steuerung eines Industrieroboters

Bachelorthesis von Titus Ebbecke

Erstprüfer

Prof. Erich Schöls

Zweitprüfer

Manuel Michel

8. Semester SS 2019

Abgabetermin 8.7.2019

Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt

Fakultät Gestaltung

Abstract

In dieser Arbeit wird Kunsterstellung als Metrik genutzt, um Kreativität als Eigenschaft von autonomen Maschinen zu erforschen. Im Streben nach einem Replikat des menschlichen Geistes ist es angemessen, Kompetenzen zu untersuchen, die auf essenziellen Aspekten der geistigen Sonderstellung des Menschen basieren. Kunstgenerierung besitzt aufgrund ihrer Voraussetzung eines Verständnisses von komplexen und kaum zu definierenden Variablen, ihrer außerordentlichen Subjektivität und der Absenz im Tierreich eine sehr humane Charakteristik. Ziel ist es, einen Beitrag zu leisten, der demonstriert, dass Computer und Roboter zumindest symbolisch Tätigkeiten übernehmen können, die sich außerhalb linearer Aufgaben und streng rationaler Motivationen bewegen. Konkret wird eine Maschine als explorative Plattform erstellt, welche erlernt, digitale Kunst zu generieren und diese Werke mithilfe eines Industrieroboters auf Leinwand malt, um einen Vergleich mit menschlicher Kunst ermöglichen. Dabei wird eine hohe Unabhängigkeit vom menschlichen Überwacher angestrebt, wodurch die Autorenschaft der Maschine maximiert wird.

Danksagung

Diese Arbeit wäre nicht möglich gewesen, ohne die zahlreiche Unterstützung von Personen, die keine Mühe gescheut haben, mir in der Planung, Bereitstellung und Installation zu helfen. Diese Arbeit war für mich beispielhaft in ihrer internationalen Zusammenstellung aus Experten, die mich mit Antworten und Empfehlungen beraten haben. Allen voran möchte ich der KUKA Deutschland GmbH danken, dass Sie innerhalb kürzester Zeit eine Zusammenarbeit eingegangen ist. Ich danke den Beteiligten für ihre Geduld und die Bereitschaft ihre unverzichtliche Expertise zur Verfügung zu stellen, wenn ich darauf angewiesen war. Ein Projekt dieser Größe wäre ohne folgende Personen nicht denkbar gewesen.

Dr. Rainer Bischoff	KUKA Deutschland GmbH
Nadine Bender	KUKA Deutschland GmbH
Jürgen Wunder	KUKA Deutschland GmbH
Martin Stark	KUKA Deutschland GmbH
Martin Löser	Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt
Nigel Stanford	
MIMIC Team	Autodesk Robotics Lab

Ich bedanke mich zusätzlich bei den Korrekturlesern, meinen Kommilitonen, die mir bei Installation und Betrieb geholfen haben und allen Beteiligten der FHWS, die der Leihgabe zugestimmt und Räume organisiert haben.

Inhalt

Einleitung

Die unproportionale Verteilung von Intelligenz im Tierreich, insbesondere die beinahe Monopolstellung der menschlichen Geistesfähigkeiten sind ein Grund für den Glaube an den Menschen als einzigartiges Wesen. Dieser findet sich in Aristoteles' epistemischen Stufen, in denen nur der Mensch „Wissen" besitzt, im Mythos des Titan Prometheus, der effektiv Tier vom Menschen trennte und nicht zuletzt im christlichen Glauben an den Apfel der Erkenntnis, welcher Adam und Eva Bewusstsein gab, wieder. Die scheinbare Einzigartigkeit des bewussten Wesens Mensch dominierte lange Philosophie, Religion und Teile der Wissenschaft. Letztere jedoch, legt spätestens seit Darwin nahe, dass diese Einzigartigkeit nicht inhärent und damit replizierbar ist. Mit der Erfindung des Computers und seiner Möglichkeit, fast alles zu simulieren, rückte der Versuch, den Geist des Homo Sapiens zu rekreieren, in absehbare Nähe. Durch die signifikanten Fortschritte der letzten Jahre in der Machine-Learning-Forschung wächst die Domäne an menschlichen Fähigkeiten, die von Computern schneller und besser ausgeführt werden können. Im Kontext des Strebens nach künstlicher Intelligenz gilt es, Disziplinen zu erforschen, die ebenfalls mit einer Sonderstellung in der Tierwelt belegt sind und erweiterte Aspekte des Intelligenzbegriffs wie Reflektion und Bewusstsein voraussetzen.

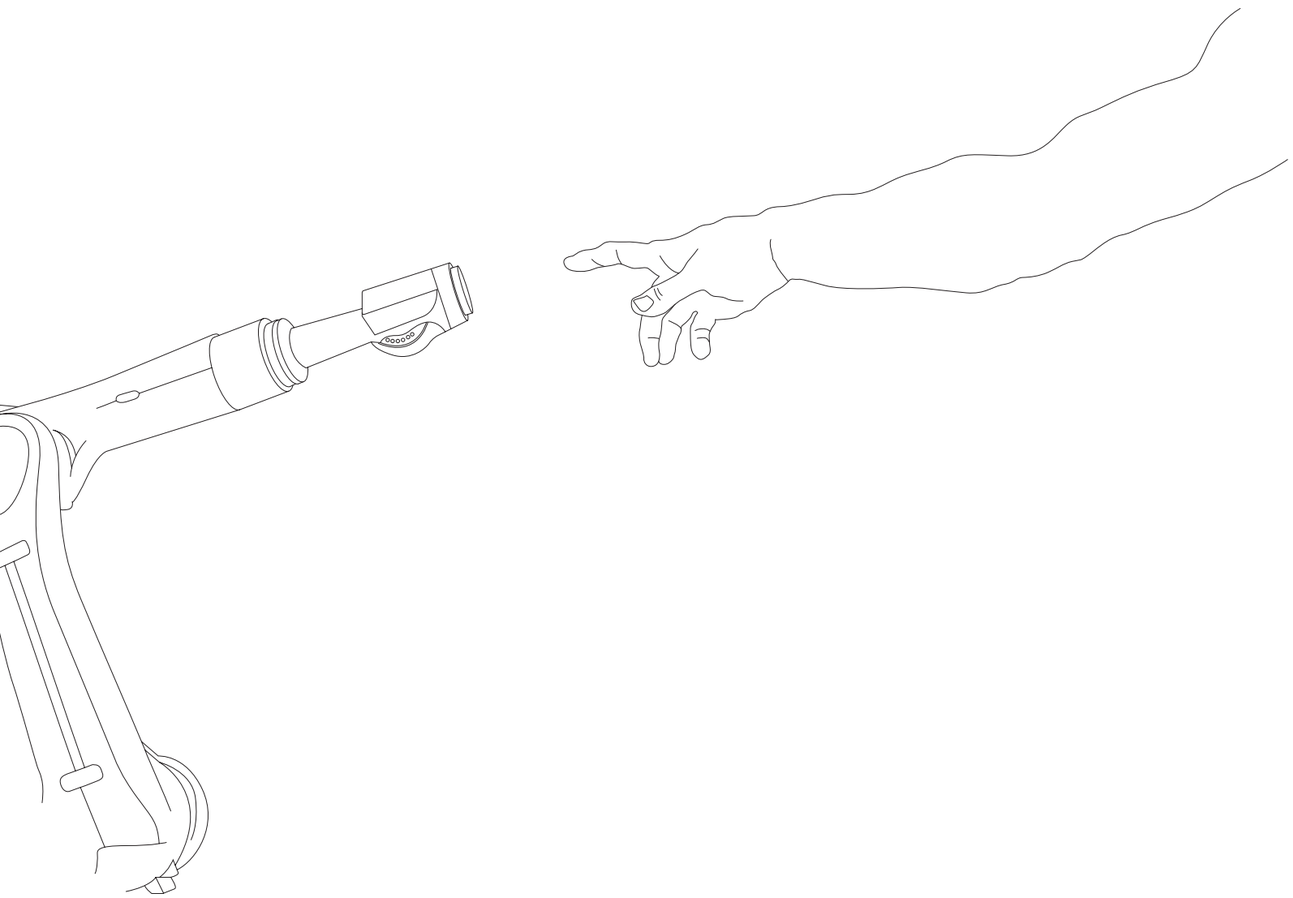
Neben der klassischen Kommunikation (Sprache, Gestik und Mimik), resultierte auch das Schaffen von Kunst aus dem Verlangen, Affektion, Erfahrungen, Konflikte, Ängste und sonstige Aspekte der tiefsten Ecken der menschlichen Gefühlswelt auszudrücken. Zu den ältesten Artefakten zählen Höhlenmalereien, die erfolgreich die Welt der Urheber zehntausende Jahre später den Betrachtern näherbringen. Praktisch simultan zur Entwicklung vom Primaten zum modernen Menschen entwickelte sich auch die Kunst mit beachtlicher Ähnlichkeit in Vielfalt und Komplexität. Eine weitere Parallele zur geistigen Vormachtsstellung des Menschen findet sich in der Verbreitung der kunsterstellenden Wesen. Kaum ein Tier erstellt auf beabsichtigter Basis Kunst, eine Tätigkeit, die weit außerhalb von primitiven Trieben und Überlebensnotwendigkeit agiert – ganz ähnlich dem großen Teil der Aktivitäten des menschlichen Alltags. Die lange Geschichte der Kunst, insbesondere der Malerei und ihre enge Korrelation mit Intelligenz macht das Erschaffen von Kunst zu einer Disziplin, deren Meisterung maßgeblicher Faktor im Streben nach menschenähnlichen Maschinen ist.

2. Forschungsstand

Die Kombination von algorithmisch gesteuerten Robotern, die Kunst erstellen, ist keineswegs neu. Bereits in der 1970er Jahren begann Harold Cohen Computerprogramme zu schreiben, die Konturen generierten. Sogenannte „Turtles“, auf Papier fahrende Roboter mit fixierten Malstiften, brachten die Formen auf das gewünschte Medium, woraufhin Cohen die umrandeten Flächen füllte. Das Programm trug den Titel AARON und zählt zu den Pionierwerken der künstlichen Kreativität. Dabei basierte die Software auf fest definierten Regeln und ist weit von aktuellen Lernalgorithmen entfernt. Diese symbolische KI diente damit eher der Diskussion als dem Beweis von künstlicher Intelligenz wie wir sie uns heute vorstellen. Die ”ArtBots: The Robot Talent Show” lässt kunstschaffende Roboter gegeneinander antreten. Häufige Gäste sind Turtles, die ähnlich wie Cohen’s Maschinen mit Stiften auf Papier fahren und so farbige Zufallsnetzwerke aus Linien erschaffen. Auch diese leiden an einer geringen Autorenschaft und zählen eher zur Zufallskunst als zu den autonomen und kreativen Agenten.

Pindar Van Arman programmiert kompakte Desktop-Roboterarme, um Machine Learning generierte oder konvertierte Bilder auf Leinwand zu malen. Viele seiner Werke greifen dabei auf Stiltransfers (Vgl. 3.3.2, S. 19) zurück, ein Prozess, der nicht vollsynthetisch ist und eher das Imitieren von existenten Kunststilen als Ziel hat, als das Erstellen von Grund auf neuer Kunst. Ungleich war das Generierungsverfahren des Portraits von Edmond De Belamy, einer fiktiven Person, erstellt von einem Machine Learning Algorithmus. Obvious Art, die menschlichen Urheber des Werkes, versteigerten das Bild im Oktober 2018 für einen spektakulären Preis von 432.000 US-Dollar (Anonym 2018). Dabei entzogen das Ausdrucken und Einrahmen des rein digitalen Bildes dem Werk partiell den maschinellen Autor. Der künstliche Agent agierte lediglich virtuell und die Konvertierung in ein physisches Bild kann als reine Zweckmäßigkeit zur Versteigerung und damit als unauthentisch angesehen werden. Die Künstler, denen vorgeworfen wurde geklauten Code ohne nennenswerten Eigenaufwand verwendet zu haben (Vincent 2018), verfestigten dennoch mit der Auktion endgültig das große Interesse, Computer zu kreativen Maschinen zu machen. Dabei ist das Streben nach maschinellen Künstlern ein natürlicher Schritt in Richtung künstlicher Intelligenz. Die Notwendigkeit für die Kunsterstellung zu erfahren und Erfahrendes zu reflektieren, neu zu ordnen und dadurch komplexe innere Gedankenspiele – grundlegend für das menschliche Wesen – zu kommunizieren, macht Kunst in Teilen zu einem Manifest des menschlichen Geistes.

Der aus dem Slawischen stammende Begriff „Roboter“ bedeutet so viel wie „Frohndienst“: heutzutage synonym mit aufgezwungener und anstrengender Arbeit. Der Roboter hat dementsprechend seine Wurzel in der Dienerschaft, der harten und vor allem nicht intellektuellen Arbeit, die unfreiwillig, also ohne freien Willen geschieht. Aufgaben für Roboter, die jenseits von repetitiven und festgeschriebenen Programmen liegen, bieten also ein wenig erforschtes Feld. Die Kombination von maßgeblichen Kompetenzen intelligenter Wesen mit mächtigen motorischen Geräten ergibt eine anthropomorphe Installation, die das Potenzial hat, diverse Argumente zur aktuellen Diskussion um Kreativität und künstliche Intelligenz beizutragen.



3. Methodik

Philosophie

Die Wahl der Methodik, insbesondere die der digitalen Bildsynthese, orientiert sich an einer gewissen Grundphilosophie.

Zu den zwei primären Zielen zählen eine starke maschinelle Autorenschaft und eine hohe Qualität der generierten Bilder. Die Unabhängigkeit von menschlichem Eingreifen in den Lernprozess ist wichtig, um die Bilder eher als Werke künstlicher Intelligenz zu bezeichnen. Die hohe Qualität der Bilder wiederum ist notwendig, um einen fairen Vergleich zwischen Maschinenkunst und menschengemachten Werken zuzulassen. Möglich ist dies nur, wenn Geschmack und Vorlieben von Menschen bedient werden.

In der Realität sind diese zwei Kriterien tendenziell gegensätzlich. Sollen die Werke auf den Vorlieben von menschlichen Betrachtern basieren, so werden sich latente Variablen wie Ästhetik dem annähern, was Menschen präferieren und dementsprechend meist auch, was sie bereits kennen. Gibt man beispielsweise einem generativen Machine Learning Algorithmus vor, menschenähnliche Kunst zu generieren, wird sein Ziel sein, den Output so nahe an den Input anzunähern wie möglich. Da Variablen wie Qualität nur durch den Input definiert werden, wird das Modell ein perfektes Imitat als qualitatives Maximum verstehen.

Der offensichtliche Gegensatz dazu wäre eine hohe Autorenschaft des generativen Prozesses. Imitate und Kopien, entweder als Resultat oder zumindest als Ziel des Lernprozesses, sind der Widerspruch zu einer hohen Autonomie in der Bildsynthese. Letzteres spielt im Verständnis von Kreativität und insbesondere Originalität eine große Rolle. Zu unterstreichen ist dabei, dass nicht ohne weiteres beide Anforderungen als prinzipiell gegensätzlich beschrieben werden können. Ein Verfahren, dass trotz oder sogar durch maximale Eigenständigkeit im Lernprozess maximale Qualität im Output erzielt, ist nicht inhärent ausgeschlossen.

Grundsätzlich gilt, dass die Maschine Dinge mit ähnlichen Grundprinzipien wie Menschen generieren sollte. Das Prinzip des Lernens ist eng mit dem Intelligenzbegriff verbunden und ist notwendig, um abseits von Zufallsparametern eigenständige und kontrollierte Leistun-

gen zu erbringen. Menschen lernen ihr Leben lang. Häufig beabsichtigt, noch häufiger ohne Intention oder Bewusstsein. Lange bevor permanente Erinnerungen gebildet werden, beginnen Menschen eine Vielzahl an Dingen über ihre Umwelt, sich und andere zu erlernen. Um diesem gerecht zu werden, liegt eine signifikante Präferenz zugunsten von Computerprogrammen vor, die mit maschinellem Lernen arbeiten.
Zusätzlich reduzieren Sekundärkriterien die Auswahl und Implementierungen an Modellen. Allen voran müssen Faktoren wie Lerndauer oder Berechnungsaufwand moderat und mit allgemein zugänglicher Hardware möglich sein. Zuletzt sollten vom Modell geforderte Trainingsgrundlagen wie ausreichend aufgearbeitete Datenset's vorhanden und zugänglich sein. Die Erstellung eines neuen Modells liegt außerhalb des Rahmens dieser Arbeit.

Aufbau und Kybernetik des Systems

Um ein möglichst menschenähnliches Generierungsprinzip zu erzielen, eignet sich eine Orientierung an dem Funktionsprinzip und den kognitiven Prozessen eines menschlichen Malers. Während Druckmaschinen eine hohe Flexibilität und Universalität im Druckprozess bieten, eignet sich ein anthropomorpher Ansatz besser. Entfernt sich die Maschine zu sehr von den motorischen Schaffungsprozessen des Menschen, schwindet die Möglichkeit, nicht nur die Resultate, sondern auch die Maschine als Installation zum Vergleichsobjekt zu machen. Das in dieser Arbeit gewählte Schaffungsverfahren beruht auf der Bedienung von herkömmlichen Malutensilien durch einen Industrieroboter. Dieser entzieht sich einer rein symbolischen Aufgabe und wird stattdessen zum oberflächlichen Imitat des menschlichen Körpers. Maß dieser Arbeit ist die Nähe zwischen Mensch und Maschine. Die Möglichkeit, die Bewegungsprinzipien des menschlichen Armes und die physischen Eigenschaften des finalen Werkes akkurat zu übernehmen, machen 6-Achsen Roboter zur bevorzugten Lösung. Offensichtliche Grenzen zwischen organischem und mechanischem Agenten verschwinden. Dieser Umstand qualifiziert diese Installation, eine direkt ersichtliche Konkurrenz zum

Training

Computer erlernt Erzeugung von 2-dimensionaler Kunst

Generierung

Computer generiert Kunst

Konvertierung

Computer erzeugt Koordinaten aus generiertem Werk

Interpretation

Robotersteuerung interpretiert Daten und handelt

Malprozess

Roboter malt generiertes Werk auf Leinwand

menschlichen Beobachter und Schöpfer zu provozieren und zu diskutieren. In dieser Arbeit wird ein KUKA KR 10 R1420 verwendet, an dessen Flansch Malwerkzeug fixiert wird, um ein generiertes Bild auf Leinwand zu bringen.
Der Roboter operiert zusammen mit einer KUKA KR C4 compact Steuerung. Zur Befestigung wurde ein Betonfundament mit 120 Liter Betonestrich gebaut. Die Holzform wurde auf einer Europoolpalette installiert und der Beton durch Gewindestangen aus Stahl mit der Form und damit der Palette verbunden. Das mobile Fundament wiegt zwischen 300 und 350kg. Durch vier Bohrungen in den Beton und darin fixierte Anker wurde der Roboter auf dem Fundament fixiert. Als Kippschutz wurden weitere 100kg Beton auf einer benachbarten Palette fixiert und beide Konstruktionen verbunden.

Abbildung 1

Physische Installation.

Neben dem kybernetischen muss auch der kognitive Teil des Systems entschieden werden. Die gewünschte Nähe zu Grundprinzipien der menschlichen Autonomie erfordert ein hohes Maß an lernbasierten Aktionen. In der Realität lösen viele Handlungen eines Menschen ein meist sensorisches Feedback aus, aus dem dieser lernt. Während die grafische Basis des gemalten Bildes problemlos durch moderne Machine Learning Algorithmen erstellt werden kann, ist das Erlernen der Motorik der Malerei stark erschwert. Optische Systeme sind zwar in der Lage, sensorisches Feedback zu geben, doch ein vollständiges Erlernen des Malprozesses erfordert enorme Ressourcenmengen. So müssten entweder Utensilien für hunderte bis tausende Malversuche oder ein physisch korrektes Simulationsmodell für einen virtuellen Lernprozess bereitgestellt werden. Die Unfähigkeit der meisten menschlichen Gehirne, selbst nach jahrelangem Hantieren mit Stift und Pinsel hochwertige Kunstwerke zu erstellen, demonstriert diesen Aufwand.

Statt einer vollständigen Feedback-Schleife wird in dieser Arbeit ein asynchroner Lernprozess genutzt. Dabei geschieht die digitale Bildsynthese vor und unabhängig vom mechanischen Prozess. Nach erfolgreichem Training auf das Generieren von menschenähnlicher Kunst erstellt das jeweilige Programm auf einem lokalen Rechner zeitlich entkoppelt Bilder nach den erlernten Regeln. Die Bilder wiederum werden unmittelbar nach der Generierung an die Robotersteuerung gegeben und vom Manipulator gemalt. Alle Lernprozesse geschehen damit vor und unabhängig vom Malverfahren des Roboters.

Algorithmik der Bildsynthese

Beurteilungsgrundlage

Ein Überblick über verschiedene Generierungsverfahren ist notwendig, um diese im Sinne der gewählten Grundphilosophie zu beurteilen. Der geforderten Nähe zur menschlichen Intelligenz wegen wird ein strenger Fokus auf Prozesse, die mit maschinellem Lernen arbeiten, gesetzt. Architektonische Probleme in Qualität, Leistung und Eigenständigkeit verschiedener Machine Learning Modelle zu beweisen liegt außerhalb der Reichweite dieser Arbeit. Die Anzahl an expliziten und impliziten Modellen und die massiven Fortschritte in verschiedenen Sub-Domänen des Machine Learning's lassen kaum eine zufriedenstellende empirische Beweislage zu. Anstatt anhand von Funktionsunterschieden zwischen beispielsweise autoregressiven (van den Oord et al. 2016) und flow-basierten Modellen (Dinh et al. 2014; Kingma und Dhariwal 2018) eine Eignung für dieses Projekt zu belegen, konzentriert sich die Beurteilung auf den Output von populären Ansätzen der Bildsynthese.

Neben der Nähe zur menschlichen Kreativität spielt auch die Qualität pro Leistungsaufwand eine signifikante Rolle. Je performanter Hardware und Modelle mit der Zeit werden, desto geringer wird die Priorität von Berechnungsgeschwindigkeiten. Wenn der (Kosten)aufwand für fotorealistische generierte Bilder unter eine gewisse Schwelle fällt und damit trivial wird, wird es interessanter, die prinzipielle Nähe zwischen maschineller und menschlicher Kreativität zu demonstrieren. Diese Arbeit agiert jedoch unter der ästhetischen Parallelität von menschlichen und maschinellen Werken und nur eine Hand voll Implementierungen aktueller Modelle sind in der Lage, hoch realistische Bilder in einem angemessenen Zeit- und Kostenrahmen zu generieren.

Stiltransfer mit Convolutional Neural Networks

Stiltransfer-Modelle übertragen Teilaspekte eines Bildes auf ein anderes Bild. Typischerweise wird dabei der Stil von Bild B auf das Motiv beziehungsweise die Komposition von Bild A übertragen. Diese Art von Bildmanipulation erzeugt Bilder, die mitunter kaum als unauthentische, generische Werke zu erkennen sind. Sie zählen damit zu den

stärksten Methoden, ein Bild nach oberflächlichen Kriterien in ein Werk zu verwandeln, das in der Kunstwelt geschätzte Aspekte beinhaltet. Gleichzeitig demonstrieren Stiltransfers die Fähigkeit, schwer bis gar nicht zu artikulierende Regeln zu erkennen und anzuwenden. Die konkrete mathematische Formulierung von Stileigenschaften ist problematisch. Da Motiv und Stil differenziert werden müssen, sind simple Ansätze für komplexe Stile selten erfolgreich. Um beispielsweise eine Fotografie optisch in ein Gemälde zu verwandeln, müssen zuerst die möglichst universellen Eigenschaften einer pinselbasierten Darstellung bekannt sein. Zu beschreiben, was einen Pinselstrich ausmacht, wenn dieser tief in ein Motiv mit diversen Farben und Formen integriert ist, fällt selbst Menschen schwer. Deep Learning Stiltransfers greifen in der Regel auf Convolutional Neural Networks (CNN) zurück. Ihre Fähigkeit, simple Features – Eigenschaften wie Mund, Nase und Augen – aus komplexen Quellbildern zu extrahieren, qualifiziert sie unter anderem zur Gesichts- oder Objekterkennung.

Obwohl CNN‘s primär zur Bildanalyse geschaffen sind, ist es möglich, die sogenannten Feature Maps zu visualisieren und damit das Originalbild oder Features aus dem Originalbild visuell zu rekonstruieren (Mahendran und Vedaldi 2014). Durch Manipulierung eines 19 Ebenen tiefen Netzwerkes gelang es, unterschiedlich konkrete Features aus dem gegebenen Input zu extrahieren und zu kombinieren, um komplexe Stile auf ein beliebiges Bild zu übertragen (Gatys et al. 2015).

Während niedrige Ebenen in diesem CNN kaum Informationsverlust haben, verliert der Input durch Downsampling in höheren Ebenen feine Details. Die Weitergabe von immer stärker komprimierten Featuremaps an folgende Ebenen geht einher mit einer immer gröber, weil abstrakter werdenden Kategorisierung der Bildinhalte. Das Verwerfen von „unwichtigen“ Informationen trainiert das Netzwerk auf die verbleibenden gröberen Bildelemente. Hohe Ebenen können dadurch das unterliegende Motiv eines Bildes A erkennen um damit präzise Details zu maskieren und für den zu übertragenden Stil freizugeben. Ebenso lassen sich dadurch Features wie Pinselstriche, Farbwahl oder Lichtverhältnisse vom stilgebenden Bild B extrahieren und übertragen, während die groben Bildelemente ignoriert werden. Überträgt man feine Details (in der Summe der Stil) von B auf die Grundstruktur eines Bildes A, kann letzteres in ein scheinbar authentisches Kunstwerk mit beachtlicher Qualität umgewandelt werden. Der Ausschluss von unwichtiger Information im stilempfangenden und stilgebenden Bild ist äußerst präzise und nach Konvertierung wirken arbiträre Fotos wie bedeutende Werke der entsprechenden Künstler.

Abbildung 2

Transfer von Stilen populärer Kunstwerke auf das Foto A mit CNN’s. Gatys et al. 2015

Zur Erzielung einer hohen maschinellen Autorenschaft ist dieses Verfahren jedoch grundlegend ungeeignet. Prinzipiell zählt der Ansatz nur beschränkt zu der generativen Algorithmik. Eine vollständige Synthese entsteht hier nicht, da das Ziel eine oberflächliche Veränderung des Inputs ist. Der Bildinhalt bleibt äußerst eng mit dem jeweiligen Quellbild verknüpft. Eine Autorenschaft kann dementsprechend fast ausschließlich den Urhebern der zu kombinierenden Bilder zugesprochen werden.
Die grundsätzliche Methodik von CNN‘s besitzt dennoch nennenswerte Parallelitäten zu neurobiologischen Prozessen im Gehirn. Die Feed-Forward Informationsarchitektur und die hierarchische Ebenenstruktur zwecks steigender Abstraktion der empfangenen Informationen ähnelt der optischen Rezeption von Menschen (Kheradpisheh et al. 2016). Kantenerkennung zur Orientierung und visueller Interaktion ist ein signifikanter Mechanismus für Organismen mit hoher Abhängigkeit von visueller Sensorik (Bhagavatula et al. 2009). Da viele lebensgroße geometrische Figuren in der Natur relativ scharfe Kanten haben, ist dies jedoch ohnehin zu erwarten.
Dennoch sind CNN‘s als Architektur der optischen Informationsverarbeitung und damit als Inspirationsquelle für neue Bilder relevant. Ihre Fähigkeit, konkrete Objekte und wage Aspekte zu identifizieren und zu klassifizieren, machen sie zu wichtigen Werkzeugen für visuell lernende Agenten. Bedingt man die Wahrnehmung der Umwelt einer Maschine als Basis zum Generieren von neuen Bildern, so wären Convolutional Neural Networks ein durchaus wichtiger Bestandteil für einen beobachtenden, maschinellen Künstler.

Stiltransfers erschaffen Bilder, die in ihrem Stil täuschend echt nach menschlichen Werken aus vergangenen Kunstepochen aussehen. Diese Technik kann jedoch kaum als vollständig generativ bezeichnet werden. Der lernende Teil dieser Methodik fokussiert sich auf oberflächliche Eigenschaften des Inputs wie Kanten und Farbstimmung. Während generative Verfahren originelle Bilder erstellen können, handelt es sich bei Stiltransfers primär um Transformationsverfahren. Der Gegensatz dazu sind die streng generativen Verfahren.
Zu den prominenten Ansätzen generativer Machine-Learning-Algorithmem gehören autoregressive Modelle (van den Oord et al. 2016; Salimans et al. 2017; Reed et al. 2017), Variable Auto Encoders (VAE) (Kingma und Welling 2013; Burgess et al. 2018; Maaløe et al. 2019) und Generative Adversarial Networks (GAN) (Goodfellow et al. 2014). Autoregressive Modelle sind im Kern stochastisch, weil sie Folgewerte auf Basis von vorherigen Werten voraussagen. Dies macht diese Modelle in der Bildsynthese attraktiv für das Vervollständigen von Bildern, wenn diese geschwärzt sind oder Bildelemente beinhalten, die durch den verdeckten Hintergrund ersetzt werden sollen. Das Vervollständigen von Bildern ist jedoch kein ausreichender Prozess für eine vollständige Bildgenerierung. VAE’s können Elemente wie Mimik, Brillen oder Azimut von Gesichtern erlernen und in generierten Bildern variieren. Ihre Fähigkeit, latente Aspekte zu identifizieren und zu kategorisieren, spricht für ein menschenähnliches Verständnis von zu unterscheidenden Teilelementen eines größeren Ganzen. VAE’s produzieren allerdings oft unscharfe oder verrauschte Bilder. GAN’s sind Algorithmen, die speziell auf generative Aufgaben wie Bildsynthese optimiert sind und seit einiger Zeit große Aufmerksamkeit erhalten haben. Moderne GAN’s sind in der Lage, originelle Bilder zu generieren, die für einen menschlichen Betrachter nicht mehr als synthetisch zu erkennen sind. Lange Zeit galten diese Modelle als instabil und aufwändig zu trainieren, allerdings wurden in letzter Zeit signifikante Fortschritte verbucht (Radford et al. 2015; Karras et al. 2017).
GAN‘s gelten mittlerweile als mächtige Werkzeuge, um Maschinen ein hoch-detailliertes Konzept einer grafischen Umwelt anzulernen. Sie werden genutzt, um Spieletex-

turen hoch zu skalieren (Ledig et al. 2016), Schriftarten (Tian 2017) und Anime Figuren (Jin et al. 2017) zu generieren oder diverse Herausforderungen in der Medizin zu meistern (Iqbal und Ali 2018; Frid-Adar et al. 2018; Kazemifar et al. 2019; Hiasa et al. 2018; Armanious et al. 2018; Yu et al. 2017). Besonders signifikant ist die Rolle, die GAN's in der Kunstgenerierung eingenommen haben (Barrat 2017; Vincent 2018). Vor allem aber gibt es neben ihnen derzeit kaum ein besseres Verfahren, welches originelle Bilder mit derart hoher Auflösung und Detailschärfe synthetisiert. GAN's sind hervorragende Modelle, um zu demonstrieren, wie künstliche Intelligenz bereits heute ein Verständnis von (sehr kleinen) Teilelementen der Realität haben kann, das weit über dem des Menschen liegt.

Generative Adversarial Networks

Die Qualität der von aktuellen GAN's generierten Bilder, die Berechnungsgeschwindigkeit und der generelle Forschungsaufwand in diesem Feld der generativen Algorithmik machen sie zu den in dieser Arbeit bevorzugten Generierungsverfahren.
Klassische Generative Adversarial Networks bestehen aus zwei neuronalen Netzwerken, die im Wettbewerb stehen. Ein „Diskriminator" erlernt zuerst die Eigenschaften des Inputs, wie zum Beispiel von Fotos menschlicher Gesichter. Sein Ziel ist es, zwischen echten Bildern des Inputs und Bildern des Generators zu unterscheiden. Der „Generator" generiert zuerst Zufallsrauschen mit dem Ziel, Bilder zu erschaffen, die der Diskriminator nicht mehr als künstlich erstellt erkennt. Der entstehende Wettbewerb trainiert beide Parteien gegenseitig, ohne dass ein Überwacher präsent sein muss, um den Lernprozess zu lenken. Da der Generator keine Vorstellung darüber hat, was zu tun ist, um den Diskriminator zu überwinden, sind seine anfänglichen Bilder einfach als falsch zu erkennen. Jedes Mal, wenn der Diskriminator diese Bilder als unecht identifiziert, variiert der Generator seinen Generationsprozess und überprüft, ob der Gegenspieler nun mehr oder weniger Schwierigkeiten hat, den Output als künstlich zu identifizieren. Ist letzteres der Fall, hat der Generator sich der Schaffungsqualität des echten Inputs angenähert. Speichert er, welcher Ansatz zu diesem verbesserten Ergebnis geführt hat, hat er in kleinen Teilen erlernt, was den Input als echt qualifiziert und wie man ihn repliziert.

Gleichzeitig wird der Diskriminator bestraft, wenn er ein falsches Bild als echt markiert hat. Er lernt dadurch umso mehr, was ein Bild falsch und echt macht. Die ewige Konkurrenz sorgt im Gegenzug dafür, dass der Generator sich ebenfalls verbessern muss, um den noch kritischeren Diskriminator zu überzeugen. Dieses Nullsummenspiel ähnelt dem Innovationsdrang durch Konkurrenz in der freien Marktwirtschaft. Konvergieren beide Akteure, wird der Generator irgendwann Bilder erschaffen, die für den Beobachter kaum noch als synthetisch zu identifizieren sind. GAN's sind dementsprechend einfach zu initialisieren, da sie keine Labels für den Input benötigen und praktisch keine Assistenz eines menschlichen Überwachers erfordern. Dieses Maß an Eigenständigkeit nicht nur in einem optimierenden, sondern in einem von Grund auf erlernenden Prozess, ist ein signifikanter Schritt in Richtung praktikabler Künstlicher Intelligenz.

Progressiv wachsende GAN's

Unter der Leitung von NVIDIA wurde 2017 ein GAN-Modell vorgestellt, welches die Pixelanzahl des Inputs im Lernprozess kontinuierlich erhöht, damit grobe Variablen schnell in der niedrigauflösenden Lernphase erfasst und an den nächst höher auflösenden Schritt übergeben werden können (Karras et al. 2017). Die durch das Wachstum erzielte Lerngeschwindigkeit und Stabilität macht dieses GAN Modell zu den aktuell fortschrittlichsten und effizientesten GAN's überhaupt. Durch eine damals rekordträchtige Inception Punktzahl von 8.80 auf CIFAR10 und wenigen Tagen Lerndauer für 512^2 Pixel große Bilder auf High End GPGPU's, eignet sich dieses GAN zum Generieren von fotorealistischen Bildern in angemessener Zeit und wurde entsprechend als bevorzugtes Modell ausgewählt.

4. Training

Datenset

Grundlage eines erfolgreichen Lernprozesses ist eine angemessene Datengrundlage. Um die Schwierigkeiten der Kunstgenerierung mit GANs zu demonstrieren, ist ein Vergleich mit dem Generieren anderer Bildarten angemessen. Zur Synthese von fotorealistischen Menschenportraits wird in aller Regel das CelebA Datenset verwendet (Liu et al. 2016). Das Set besteht aus rund 200.000 Bildern unterschiedlicher Gesichter. Die schiere Anzahl an Portraitfotos von prominenten Persönlichkeiten erlaubt es, eine enorm große Datenbasis mit hoher visueller Ähnlichkeit des Inhalts zu erstellen. Während das CelebA Datenset häufig als Benchmark von GAN's verwendet wird, sind Portraitsets wie FFHQ (Karras et al. 2018) besser geeignet, um mit einer hohen Diversität in Gesichtsmerkmalen, Belichtung und Hintergrund zu arbeiten. Dies steht in Kontrast mit Personenabbildungen aus öffentlichen Veranstaltungen (CelebA), wo Blitzlicht und soziale Selektion eine realistische Diversität von menschlichen Eigenschaften verhindert. Beide Datensets spielen eine signifikante Rolle in der Qualität von durch GAN generierten Gesichtern. Obwohl FFHQ auf 70.000 Bilder reduziert ist, besitzen beide Sets Eigenschaften, die für eine parametrische Analyse wesentlich besser geeignet sind als Datensets aus Kunstwerken.

Allen voran sind Quellen für große Mengen gemeinfreier signifikanter Kunstwerke rar. Während viele große Museen ihr Inventar in Form von Metadaten veröffentlichen, ist wikiart.org einer der wenigen Ansammlungen von fotografischen Abbildungen wichtiger Kunstwerke. Mit weit über 100.000 Werken sämtlicher Genres und Epochen zählt es zu den wenigen ausreichend großen, digitalen Sammlungen von signifikanter Kunst. Die gleiche Quelle wurde bereits von Robbie Barrat genutzt, um ein GAN auf klassische Kunst zu trainieren. Der Code und das trainierte Modell wurden später von Obvious Art verwendet, um ein Portrait zu generieren, welches für fast eine halbe Millionen Dollar versteigert wurde (Vincent 2018). Während der Erstellung dieser Arbeit wurde wikiart.org mit einem asynchronen Ladeverfahren versehen und die URL-Seitenstruktur zu komplex gehalten, um ein einfaches Crawlen der Bilder zu ermöglichen.

Ein zuvor entstandenes Datenset der Seite ist dennoch öffentlich zugänglich[1]. Dieses umfasst nicht alle verfügbaren Werke, sondern ist auf 80.000 Bilder begrenzt. Davon sind 7.360 Bilder dem Genre Abstrakt zugeordnet und 14.125 Bilder zählen zu den Portraits.

Die Anzahl an Daten ist damit signifikant geringer als CelebA und FFHQ. Ein noch größerer Faktor ist jedoch ein wesentlich komplexerer und größerer, latenter Vektorraum. Fotografien menschlicher Gesichter weisen deutlich mehr Parallelen auf als künstlerische Darstellungen. Während der realistische Stil und die Proportionen in Fotografien von Gesichtern eine Mustererkennung relativ leicht machen, muss ein Lernalgorithmus mit einer hohen Vielfalt an verschiedenen künstlerischen Darstellungsweisen kämpfen. So sind kubistische und vielfach-perspektivische Gesichtsinterpretationen von Picasso enorm weit von realistischen Barockportraits entfernt. Das Herausarbeiten von zu ignorierenden Einzigartigkeiten und Identifizieren von universellen Regeln wird damit zur Herausforderung.

Zusätzlich wird im Gegensatz zu CelebA und FFHQ kein Bildbeschnitt auf Hauptmotive vorgenommen. Die Komplexität in diesen Datensets wird nochmals dadurch reduziert, dass die zu erlernenden Eigenschaften eines Gesichtes räumlich stets an ähnlicher Stelle stehen. Augen, Mund und Nase sind für einen Algorithmus dadurch schnell zu verordnen. Kombiniert man diese Restriktionen, wird deutlich, dass die Anzahl an Bildern zu gering ist, um mit heutigen GAN's stabil realistische Portraits zu erzeugen, die von Menschen nicht als maschinell erkannt werden.

Das direkte Gegenargument zu diesem Vorgehen wäre eine Selektion der Daten. Die Künstlergruppe Obvious Art (vgl. Seite 24) gibt an ihren GAN Input „sorgfältig […] zu sortieren", um „gemeinsame visuelle Eigenschaften" (Fautrel et al. 2019) im Datenset sicherzustellen. Die Resultate sind in ihrem Detailreichtum zwar dennoch arm, doch dürfte dies durch das mittlerweile veraltete DCGAN (Radford et al. 2015) zu erklären sein, dass die Künstler nutzten. Dass eine signifikante Qualitätsverbesserung durch Selektion von beispielsweise realistischen Portraits aus den Epochen Barock, Romantik

1) Vgl. https://www.kaggle.com/c/painter-by-numbers/data

und Realismus stattfinden würde, ist kaum zu anzuzweifeln. Betrachtet man jedoch die Forschungsgrundlage dieser Arbeit, wird ein Eingreifen entmutigt. Solange die Selektion des Inputs den Output maßgeblich kontrolliert, wird dem Eingreifenden eine Autorenhandlung zugesprochen. Ähnlich wie die Wahl der Farbe beim Malen verändert das Auswählen der Trainingsgrundlage das daraus resultierende Kunstwerk. Besonders dramatisch wird dies, wenn die Selektion nach eigenen Wunschvorstellungen über das Resultat geschieht. Der Maschine entzieht man so die Beliebigkeit und degradiert den Lern- und Generierungsprozess umso mehr zum Werkzeug eines menschlichen Urhebers.

Wahl des Inputs

Trotz der Grunddevise, auf Selektion zwecks Outputoptimierung möglichst zu verzichten, um dem Gesamtwerk eher eine Maschine als Autor zuzuordnen, reicht der Überparameter „Kunst“ nicht als Trainingsbasis aus. Ein Ziel dieser Arbeit ist ein Vergleich mit menschlichen Urhebern. Dies setzt ein Qualitätsminimum der Ästhetik voraus, welches nur schwer erreicht wird, wenn die gesamte Domäne der Kunst erlernt werden muss, gerade wenn dafür nur wenige Daten zur Verfügung stehen. Gleichzeitig ist eine gewisse Einschränkung der Lerngrundlage gut zu rechtfertigen. Aus der Perspektive eines Agenten, der zweitausend Jahre Kunsthistorie erlernen soll, ist eine Umwelt aus 7.000 bzw. 14.000 Bildern prinzipiell informationsarm. Soll der Agent auf dieser Basis aber seine Umwelt unter Konkurrenz durch einen Diskriminator replizieren, kann dies nur so akkurat geschehen, wie seine Umwelt universell relevante Variablen enthält. Kunst als bereits mehrdeutiger Begriff mit hoher Allgemeingültigkeit, selbst in engeren Definitionen, beinhaltet einen Informationsraum, der schwer durch nur einige tausend Bilder mit niedriger Auflösung beschrieben werden kann. Dies geht einher mit der Tatsache, dass das GAN damit beauftragt wird, wichtige und triviale Variablen zu unterscheiden, um mit ersteren Rekombinationen seiner Umwelt zu generieren. Zusätzlich zur vollständigen Informationsrepräsentation muss also eine signifikante Redundanz von allgemeingültigen Variablen und eine minimale Anzahl von unwichtigen Variablen in der Lerngrundlage vorhanden sein. Nur so kann eine Gewichtung von wichtigen Aspekten (Bsp. Position und Schattierung von Mund, Nase, Augen) und unwichtigen (Bsp. Blätter im Hintergrund von Portraits) stattfinden.

Aus dieser Motivation heraus wurde eine Genreeinteilung vorgenommen. Dabei wurden die Kategorien „Portrait“ und „Abstrakt“ als jeweiliges Datenset festgelegt. Abstrakte Kunst eignet sich gut für einen Modell, dass aufgrund von Knappheit und Komplexität des Inputs tendenziell underfittet ist. Gegenstandslose Kunst impliziert bereits im Namen eine Distanzierung von Notwendigkeiten wie der oberflächlichen Erkennbarkeit von natürlichen Objekten. Diese folgen trotz der scheinbaren Vielfältigkeit in der realen Welt strengen und damit leicht zu verfehlenden Definitionen. Gut erkennbar wird dies an der Illustrationskompetenz von Menschen. Während simple Körper wie Tische oder kubische Hochhäuser von durchschnittlichen Personen noch gut gezeichnet werden können, ist das realistische Darstellen von Flugzeugen, Pferden oder Fahrrädern für viele Erwachsene beinahe unmöglich. Dabei sind adoleszente Menschen mit jahrzehntelangem visuellen Training auf diese Objekte ausgestattet. Obwohl dieses Training nicht besonders intensiv ist, taugt es dennoch als ausreichender Indikator für die Schwierigkeit, Objekte visuell zu replizieren, die sich Mensch und Maschine momentan noch teilen. Wird die Gegenständlichkeit entnommen, verfällt ein wichtiger Parameter zur Beurteilung eines Werkes, insbesondere wenn diese ohne Kontextwissen geschieht. Stattdessen dominieren weniger strenge Regeln die Ausführung abstrakter Kunst, Farbharmonie beispielsweise. Gerade für die Betrachtung von abstrakter Kunst ohne Hintergrundinformation eignet sich also dieses Genre, um einen kompetitiven Vergleich zwischen menschlicher und GAN-erstellter Kunst zu ermöglichen.

Portraits hingegen bieten sich durch andere Parameter an. Soziale Prägungen und evolutionäre Determinierung machen Menschen äußerst sensibel in der Wahrnehmung von menschlichen Gesichtern. Unnatürliche Proportionen und grobe Asymmetrie wirken unattraktiv oder sogar abstoßend auf den Betrachter (Zaidel und Hessamian 2010; Perrett et al. 1999). Symmetrie und Größenverhältnisse von Organen lassen eine intuitive Teilbeurteilung der Qualität der generierten Bilder zu. Dies steht in Kontrast zur stark abstrahierten und gänzlich abstrakten Kunst, bei der oft große Differenzen in der Wertschätzung von Laien und Kunstexperten die Objektivität erschweren.

„Is that ladder for climbing, or is this for appreciating?“

(Safer 1993)

Der Journalist Morley Safer, in seinem Versuch, moderne Kunst von Alltagsgegenständen im Lagerraum des Auktionshauses Sotheby's zu unterscheiden.

Das Gesicht im Portrait als äußerst wichtiger Teil der menschlichen Umwelt einerseits und der offene und unabhängige Interpretationsraum, den das Abstrakte liefert, andererseits, sind zwei relevante Aspekte des menschlichen Lebens. Diese Umstände empfehlen die Wahl der beiden Genres „Abstrakt“ und „Portrait“.

Präparierung

Das gewählte wikiart.org Datenset ist mit einer CSV Datei katalogisiert. Die darin enthaltenden Dateinamen der entsprechenden Genres wurden selektiert und in einer Textdatei gespeichert. Zur Sortierung wurde für diese Arbeit ein Powershell Skript geschrieben, welches die Dateien mit Namen aus dieser Textdatei sucht und die gewählten Bilder in den gewünschten Zielordner kopiert. Daraus resultiert eine nach entsprechenden Metadaten selektierte Ordnerstruktur.

Das gewählte GAN setzt quadratische Bilder voraus. Um die Kunstwerke mit diversen Seitenverhältnissen anzugleichen, gibt es zwei Wege. Ein additives Verfahren wäre der Zusatz von Weißraum um das Bild, sodass die kürzere Seitenlänge durch einen weißen Rahmen ausgeglichen wird. Die Alternative wäre der subtraktive Anschnitt. Hier wird ein Teil der größeren Seitenlänge schlicht abgeschnitten, um ein quadratisches Seitenverhältnis zu erlangen. Beide Verfahren verändern den Input und stellen damit einen gewissen Eingriff in den Lernprozess dar. Für diese Arbeit wurde das subtraktive Verfahren gewählt. Die Algorithmik interpretiert die Pixel des im additiven Verfahren erstellen Weißraums als legitimen Teil des Bildes und geht dadurch fälschlicherweise davon aus, dass Menschen gerne weiße Flächen an den Seiten Ihrer Bilder platzieren. Versuche, das Seitenverhältnis durch Addition von Alpha-Kanälen zu korrigieren, scheiterten an der fehlenden Unterstützung von RGBA. Eine optimale Lösung wäre ein Modell, das unterschiedliche Bildproportionen unterstützt. Da 2er-Potenzen fundamentale Relevanz in der Computerwissenschaft haben, ist jedoch nicht davon auszugehen, dass dies problemlos möglich ist. Obwohl ein Beschnitt Datenverlust mit sich bringt, wurde diese Methode ausgewählt. Ein Zusatz von Bildelementen macht die für den Zusatz verantwortliche Person immer zum Teilautor des (neuen) Werkes. Nimmt man jedoch einen mikroskopischen Ausschnitt eines Gemäldes, so bleibt der Autor dieses Ausschnittes dem Autor des vollen Bildes eher gleich.

Bei der sequenziellen Beschneidung von Portraits tritt jedoch ein Problem auf. Ist kein Gesicht im Ausschnitt zu erkennen oder ist dieses angeschnitten, so wird das Erlernen von menschlichen Gesichtsproportionen signifikant erschwert. Diese Problematik ist besonders sensibel, da bereits die Größe des Datensets und die massive Komplexität der künstlerischen Darstellung von Gesichtern (Vgl. S. 25) einen Verständnisaufbau der Proportionen hemmen. Ein Ansatz wäre, den Beschnitt relativ zur oberen Bildkante zu tätigen. Der menschliche Körper motiviert in der Regel den Künstler, den Kopf als vertikale Extremität in den oberen Teilen des Bildes zu platzieren. Neben der spekulativen Natur dieses Ansatzes ist er allerdings auch in Querformaten ungeeignet. Stattdessen wurde Gesichtserkennungssoftware verwendet, um die Wahrscheinlichkeit zu maximieren, ein Gesicht im finalen Beschnitt vorzufinden. Einige prominente Lösungen wurden ausgetestet, wobei sich die „face_recognition“ Bibliothek[2)] als akkurat und kompatibel bewiesen hat. Das für die Arbeit angefertigte Skript berechnet die Orientierung des Bildes (Hoch- oder Querformat), erkennt ein Gesicht und reduziert die größere Seitenlänge auf den Wert der kürzeren. Dabei wird eher selten eine perfekte Zentrierung des Gesichtes erreicht. Würde ein zwei- statt eindimensionaler Beschnitt durchgeführt werden, sodass die Gesichter einen ähnlichen Prozentsatz des bearbeiteten Bildes ausfüllen, würde das Training optimiert werden, da so fast ausschließlich Gesichter ähnlicher Größe und gleicher Position den Input darstellen würden. Die Proportionen wären folglich einfacher zu identifizieren und damit zu erlernen. Im Hintergrund des Leitsatzes, dem Menschen die Autorenschaft möglichst zu entziehen, wird darauf aber bewusst verzichtet. Zuletzt verkleinert das Skript die Bilder auf eine gewünschte Größe.

Von den >14.000 verarbeitenden Portraits scheiterte die Konvertierung bei 75 Werken. Grund dafür dürfte bei einigen Bildern ihre Größe (bis zu 20.000 x 30.000 Pixel) sein, bei

2) Vgl. https://github.com/ageitgey/face_recognition

anderen kann auf problematische Metadaten oder Downloadfehler spekuliert werden. Die fehlerhaften Bilder wurden manuell mit Bildbearbeitungssoftware beschnitten.
Lokale Tests mit diesem Datenset scheiterten vorerst. Bei der Konvertierung des Bilderstapels in serialisierte Datenpakete mittels dem dafür bereitgestelltem Python Tool brach der Prozess nach einigen tausend Bildern ab. Die Eliminierung diverser Fehlerquellen erwies sich als schwierig, weil die Komplexität und Log-Armut des Tools wenig Aufschluss über das Problem gab. Da die Reihenfolge der eingelesenen Bilder nicht ermittelt werden konnte, wurden die fehlerhaften Bilder im Einzelnen nicht gefunden. Als bevorzugter Lösungsweg wurde eine Angleichung aller Zusatzwerte des Datensets gewählt. Neben verschiedenen Metadaten und unterschiedlicher Pixeldichte wurden auch uneinheitliche Farbräume als Fehlerquelle in Betracht gezogen. Die letzten zwei Eigenschaften wurden mithilfe der Batch-Massenverarbeitung von Photoshop auf gleiche Werte gesetzt. Alle bearbeitbaren Metadaten wurden mit dem Windows Explorer entfernt. Dieser Aufarbeitungsprozess war ausreichend. Dies ist durchaus überraschend, da das Auslesen von einigen tausend Bildern zunächst erfolgreich war, obwohl diese mit hoher Wahrscheinlichkeit unterschiedliche Zusatzwerte besaßen. Es kann darauf spekuliert werden, dass eine gewisse Toleranz gegenüber Metadaten oder Farbraum- und Pixeldichtenunterschiede existiert, diese aber begrenzt ist und bei äußerst wenig unbekannten oder fehlerhaften Werten den Serialisierungsprozess abbricht.

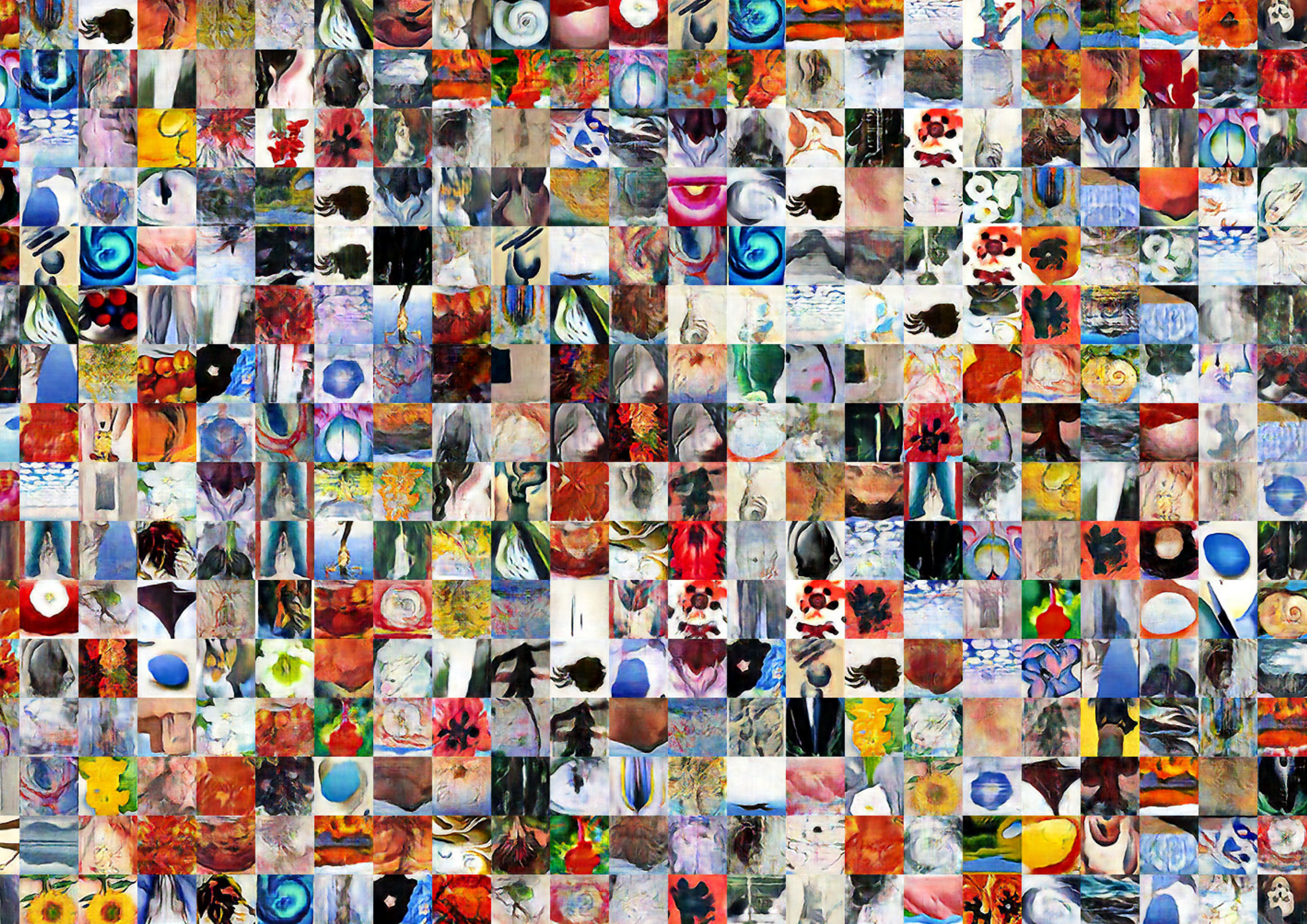
Setup und Training

Das GAN wurde zuerst lokal getestet. Der dafür benutzte Computer war mit einer NVIDIA GTX 1070 (ca. 6,5 TFLOPS Single Precision), einer i5-3570K CPU bei 4 GHz und 8GB Ram ausgestattet. Das System wurde mit aktuellen NVIDIA Treiber-, cuDNN und Cuda Toolkit Versionen betrieben. Das Trainingsset bestand aus 237 Gemälden der US-amerikanischen Künstlerin Georgia O'Keeffe. Ihre Grenzgänge zur gegenstandslosen Malerei und die stark definierten Formen des Präzisionismus eignen sich hervorragend für kontextschwach-lernende Algorithmen. Dabei ist zu unterstreichen, dass 200 Bilder um ein Vielfaches zu gering in ihrer Anzahl sind, um neuartige Bilder zu generieren.

Ein generalisiertes Verständnis von Form und Farbe ist de facto unmöglich, wenn jedes Element der Trainingsbasis praktisch einzigartig ist. Da dieser Schritt jedoch zum Benchmarking und allgemeinen Testzwecken dient, ist dies zu vernachlässigen. Aufgrund der schwachen Hardware beschränkte sich die Auflösung auf 64² Pixel. Das Training wurde nach 31 Stunden beendet. Zu diesem Zeitpunkt waren Qualitätsunterschiede zwischen echten und generierten Bildern mäßig gut mit dem bloßen Auge zu erkennen. Aufgrund der enorm limitierten Größe des Datensets ist das Overfitting wenig überraschend. Der Generator wurde aufgrund der geringen Gemeinsamkeiten aller Bilder auf triviale Variablen trainiert und lernte so das direkte Imitieren des Inputs. Zu beachten ist dabei das beinahe exakte Kopieren einzelner Bilder. Vergleicht man das Raster aus falschen Bildern (Vgl. Abb. 3) mit populären Werken von O'Keeffe, so erkennt man schnell die zugehörigen Replikate bzw. Originale. Dennoch sind einige Variationen zu erkennen, was einen guten Einblick in die Herausarbeitung von latenten Variablen gibt. Die generierten Bilder sind ohne Zweifel Kopien des Inputs und keineswegs als rein maschinell zu erachten. Die Abhängigkeit zum Input ist dramatisch und ein generalisiertes „Verständnis" von Kunst ist zweifelsohne nicht präsent.

Abbildung 3, folgende Seiten

Auf Kunst von Georgia O'Keeffe trainiertes GAN. Keines der gezeigten Bilder stammt von der Künstlerin direkt, sondern wurde von einem Algorithmus erzeugt. Aufgrund von Urheberrechtsbestimmungen, ist es nicht mehr möglich die Originale zu publizieren. Die Ähnlichkeit zwischen O'Keeffe's geschützten Werken und den synthetischen Imitaten wirft die Frage auf, wie ähnlich rechtlich das GAN der eigentlichen Urheberin ist und ob die maschinellen Werke entsprechend unter dem gleichen Schutz stehen.



Obwohl nach 31 Stunden Training auf dem lokalen System der Generator zufriedenstellende Ergebnisse lieferte, ist die genutzte Hardware keine Option für größere Auflösungen. 512² Pixel sind als angemessenes Mittelmaß zwischen optischer Qualität und Berechnungsaufwand angestrebt. Diese Auflösung beinhaltet 64-mal mehr Pixel als der lokale Test mit 64² Pixel großem Input. Die Trainingsdauer eines linearen GAN's würde um einen ähnlichen Faktor wachsen. Die für das gewählte GAN empfohlenen NVIDIA Tesla V100 GPU's waren im Bearbeitungszeitraum dieser Arbeit die stärksten konventionellen Grafikkarten der Welt. Trotz der allgemeinen Leistungsüberlegenheit (14 TFLOPS vs. 6.5 TFLOPS @GTX 1070) erschwert die Volta Architektur der V100 eine klare Prognose über die Trainingszeit. Volta liefert Tensor Kerne, die Hardware-Beschleunigung für Training mit gemischter Präzision (Micikevicius et al. 2017) ermöglichen. Potenzielle Engpässe wie VRAM, DRAM, CPU Leistung und Verbindungsbandbreiten regulieren ebenfalls die Trainingsdauer. Um die Wahrscheinlichkeit für einen reibungslosen Lernverlauf zu maximieren und FP16 Mixed-Precision Training zu benchmarken, wurde auf einer Cloud VM, bestehend aus 1x Tesla V100, 30 GB Ram und 8 vCPUs (Xeon E5-2623), Ubuntu 16.04 aufgesetzt. Lösungen wie TPUs, Computing Pipelines als VM Alternativen oder Jupyter Notebooks wurden kurzeitig evaluiert, aber vorerst als zu komplex, leistungsschwach oder teuer eingeschätzt.

Eine dem lokalen System ähnliche Softwarekonfiguration erwies sich als ungenügend. Das Training startete zwar, doch zeigte das Programm keinen Fortschritt. Sowohl im Server als auch im lokalen System wurde während der ersten Berechnungsphase keine GPU Auslastung und eine hohe CPU Auslastung gemessen. Dies lässt darauf schließen, dass anfangs eine einmalige „Setup" Phase aus regulären CPU Berechnungen durchlaufen wird. Dementsprechend konzentrierten sich die Lösungsansätze auf CPU basierte Problemfelder. Die generischen Tensorflow Binärdaten aus öffentlichen Datenbanken beinhalten keine Unterstützung für zusätzliche Instruktionen, die manche CPU Architekturen verwenden können – in diesem Fall AVX und SSE4. Dass GPU basiertes Training davon profitiert, ist zweifelhaft. Da die Problematik aber in einem CPU-lastigen Teil des Trainings auftrat, sind fehlende CPU Instruktionen eine plausible Problemquelle. Da keine Paketsetups für die präferierten CUDA und cuDNN Versionen auf Basis der gewählten CPU Architektur vorzufinden waren, wurde Tensorflow auf dem Server von Grund auf kompiliert.

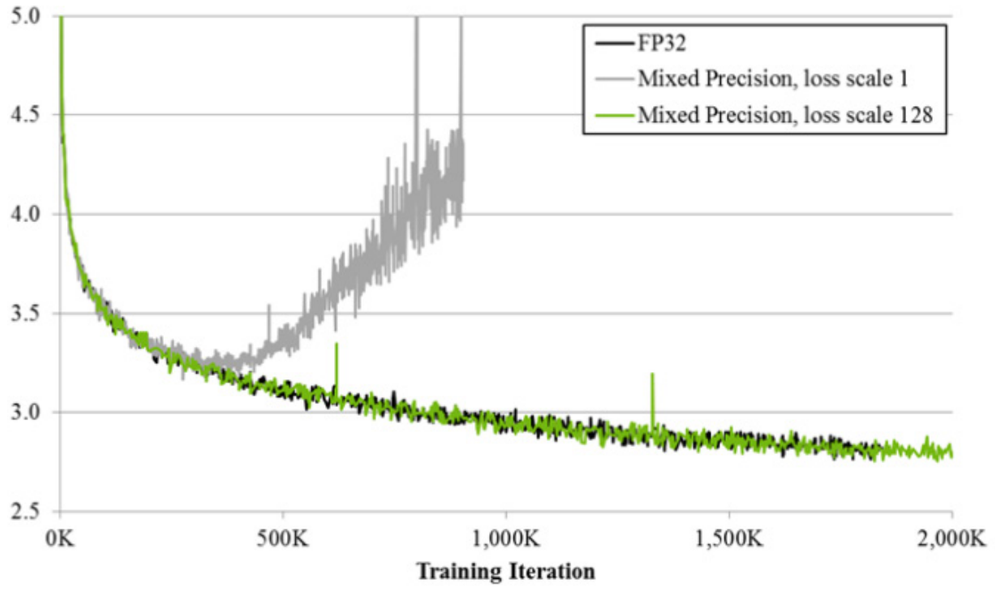


Abbildung 4

Lernkurve für LSTM Sprachmodell. Y-Achse ist Trainings-Loss. Die Divergenz bei Mixed Precision ohne Loss-Skalierung (grau) ist deutlich zu sehen (NVIDIA, 2019).

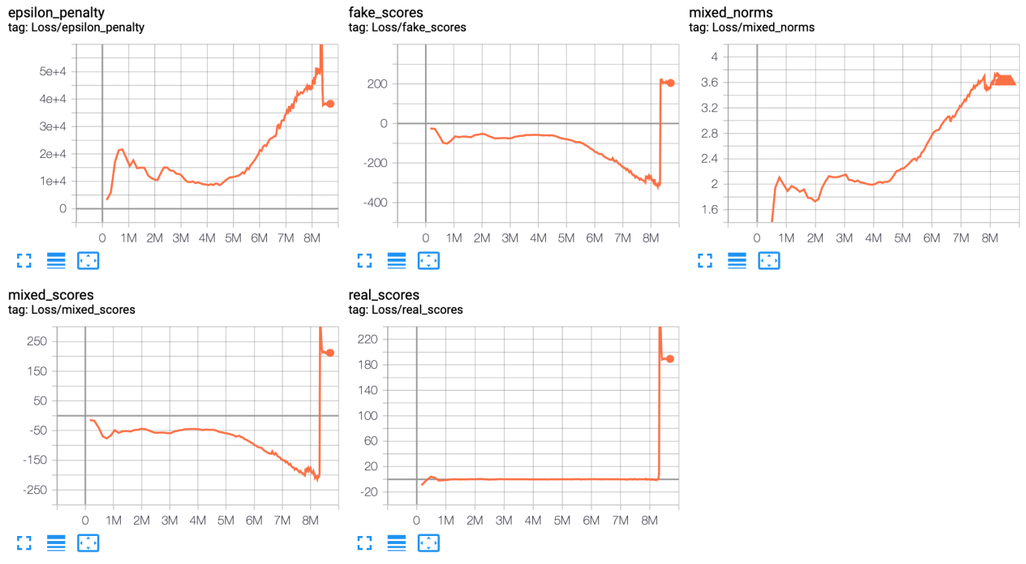


Abbildung 5

Diverse Metriken während des Trainings auf abstrake Kunst in Tensorboard.

Alle verfügbaren und von Tensorflow nutzbaren CPU Instruktionen wurden dabei einbezogen. Zusätzlich wurde die Leistung von regulärem FP32 mit FP16 Mixed-Precision Training verglichen. Wie von den GAN Entwicklern angekündigt, war hier eine signifikante Zeitreduktion mit FP16 Training im GAN Benchmark zu erkennen. Die Leistungssteigerung ist durch die Tensor Kerne und Unterstützung für Mixed Precision Training der Tesla V100 zu erklären.

Die Datensets „Abstrakt“ und „Portrait“ wurden für jeweils 38 Stunden, 19 Minuten und 62 Stunden, 51 Minuten trainiert, bevor der Prozess manuell unterbrochen wurde. Gegen Ende des Trainings wurden die Bilder sichtlich verfälscht generiert, bekamen einen intensiven Orange- bzw. Blaustich und veränderten sich kaum noch. Die meisten Trainingsparameter schlugen in ein Extrem aus und korrigierten sich nicht mehr (Vgl. Abbildung 5 S. 35). Einige Werte wurden maschinell unleserlich (Vgl. Abbildung 5 „mixed_norms“ S. 35). Die tatsächliche Fehlerquelle konnte nicht identifiziert werden. Plausibel ist, dass einige Trainingskurven auf Grund des Mixed Precision (FP16) Trainings divergierten. Einige Gradient-Aktivierungswerte können von FP16 nicht repräsentiert werden und werden dadurch auf 0 gesetzt (Micikevicius et al. 2017). Dies kann generell durch die Multiplizierung von Trainingsloss mit einem Skalierungsfaktor gekontert werden. Das Training wurde entsprechend mit Loss-Skalierung durchgeführt. Allerdings gilt es zu beachten, dass die Loss-Skalierungswerte von Diskriminator und Generator zur gleichen Zeit in negative Extreme abstürzten. Ob dies Ursache oder Reaktion der Gesamtproblematik ist, konnte nicht ermittelt werden. Die direkt oder mit einem anfänglichen Zwischenschritt anfangende, lineare und plötzliche Divergenz aller betroffenen Kurven unterscheidet sich von NVIDIA‘s Beispiel für Divergenz des Training-Loss in Abbildung 4 (Vgl. S. 35) ein weiterer Grund, Mixed-Precision als Fehlerursache anzuzweifeln.

Generell zählt die Stabilität insbesondere hinsichtlich Konvergenz und Mode Collapse zu den großen Problemen von GAN’s (Mescheder et al.; Arjovsky und Bottou 2017; Arjovsky et al. 2017; Thanh-Tung et al. 2019; Thanh-Tung und Tran 2018). Die plötzliche Radikalisierung einiger Werte (allerdings nicht der Resultate) spricht aber nicht unbedingt für ein architektonisches Problem des Models. Da Serverprobleme fast ausgeschlossen wurden, Hardware Limitierungen als unplausibel eingeschätzt wurden und die Komplexität hinsichtlich CUDA und cuDNN Versionen in Kombination mit FP16 zu hoch für eine eindeutige Fehleranalyse war, wurde das Problem nicht weiter ermittelt. Der trainierte Generator lieferte befriedigende Ergebnisse, auch wenn die Trainingszeit nur 60% der grob empfohlenen Dauer betrug. Das Training wurde mit FP32 am letzten „problemfreien“ Zwischenstand fortgesetzt. Obwohl es in dieser Konfiguration etwas länger stabil schien, destabilisierte es sich ebenfalls nach einigen Stunden.

Abbildung 6

Beispiel für ein zufällig generiertes Bild aus dem Modell ”Abstrakt”.

Abbildung 7

Beispiel für ein zufällig generiertes Bild aus dem Modell ”Portrait”.

5. Konvertierung in Koordinaten

Eine signifikante Herausforderung liegt in der Konvertierung von 2-dimensionalen Bildern in 3-dimensionale Koordinaten, die der Roboter einlesen und anfahren kann. Um dies zu ermöglichen, wurde in dieser Arbeit ein Algorithmus entwickelt, der flexibel und verlässlich beliebige Bilder in Koordinaten umwandelt, die von einem Werkzeug abgefahren werden können. Das vorgestellte Programm ermöglicht es dadurch, mithilfe jeder 2- oder 3-dimensional arbeitenden CNC-Maschine ein Bild zu malen. Dabei unterteilt sich dieses Verfahren in vier wesentliche Abschnitte. Die Reduktion der Farbkomplexität und das anschließende Anpassen der reduzierten Farbwerte an eine vordefinierte Farbpalette, die Algorithmik zur Berechnung von Pfaden auf Basis von Eigenschaften des Malwerkzeugs, das Übertragen der Koordinaten und Logik Parameter und zuletzt die roboterseitige Interpretation dieser Daten. Obwohl Ansätze zur Konvertierung von Bildern in Malpfade bereits existieren (Huang et al. 2019; Gülzow et al. 2018), ist das Generieren von CNC-Koordinaten aus Bilddaten zum Zwecke eines Malverfahrens ein Prozess, der kaum Forschung genießt. Dementsprechend wurde in dieser Arbeit ein neuartiger Algorithmus erstellt, der ein stabiles Gleichgewicht zwischen Komplexität und Universalität anstrebt.

Farbquantisierung

Da diese Arbeit eine vordefinierte Anzahl an Malfarben bedingt und die Malerei signifikant weniger Mischfarben nutzt als die 16,7 Millionen Farben des 8 Bit RGB-Farbraums, muss eine Farbquantisierung zur Farbreduktion vorgenommen werden. Während diverse Verfahren zur Vektorquantisierung wie Octree, Median-Schnitt oder Minimalvarianz existieren, erwies sich der k-Means-Algorithmus (Lloyd 1982) als zuverlässig und durch OpenCV gut implementiert. Um die minimal notwendige Anzahl an Farbclustern k zu ermitteln, wurden die generierten Portraits und abstrakten Werke in unterschiedliche Anzahlen an Clustern reduziert. Zwölf Farben, minus Weiß als Grundierung, wurden

dabei als optimaler Kompromiss zwischen Qualität und Komplexität bewertet. Da jedes Bild unterschiedliche Durchschnittswerte beinhaltet, ist es notwendig, diese Werte an eine vordefinierte Farbpalette anzupassen. Dazu wurden zwei Raster aus jeweils 84 zufällig generierten Portraits und abstrakten Werken erstellt und die Farbquantisierung durchgeführt. Das Resultat sind zwei zwölffarbige Durchschnittsfarbpaletten, im Folgenden ***colorPaint*** genannt, die die zwei Bildgenres angemessen gut repräsentieren.

Der nachfolgende Code ist zwecks Leserlichkeit und Universalität gekürzt. Eine vollständige Dokumentation des eingesetzten Algorithmus mit der Deklaration aller genutzten Variablen findet sich im Anhang.

```
img = cv2.imread(inputFile)
```

Einlesen der Datei

```
Y = img.reshape((-1, 3))
Y = np.float32(Y)
```

Umformatierung des Pixelarrays

```
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_
MAX_ITER, 10, 1.0)
K = 12
```

Definieren von Parametern. K ist Anzahl der gewünschten Farben

```
ret,label,center=cv2.kmeans(Y,K,None,criteria,10,cv2.KME-
ANS_RANDOM_CENTERS)
```

Anwendung k-Means

```
center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape))
```

Umformatierung in Bildarray

Um das passende Farbpaar aus Palette und Input-Bild zu finden, wird für jede Farbkombination der euklidische Abstand berechnet.

$$d(p,q) = \sqrt{(B_p - B_q)^2 + (G_p - G_q)^2 + (R_p - R_q)^2}$$

Wobei d der Abstand ist. p sind die Werte des Inputs und q die Werte der vordefinierten Palette. $\mathbf{B,G,R}$ sind die Blau-, Grün- und Rotwerte.

```
for i in range(K):                                                    Iteration durch sämtliche Farben
    for j in range(K):                                                Sub-Iteration durch sämtliche Farben
        d = math.sqrt((((center[i][2])-(colorPaint[j][2]))*0.3)**2 + (((center[i][1])-(color
        ↵ Paint[j][1]))*0.59)**2 + (((center[i][0])-(colorPaint[j][0]))*0.11)**2)   Abstandsberechnung d mit Gewichtung
        a.insert(j, int(d))                                           Eintragen des Abstands d in Liste a
    if a[np.argmin(a)] < colorDifferenceMax:                          Schleife für Werte unter Abstandsmaximum
        b.insert(i, np.argmin(a))                                     Eintragen des Index des kleinsten Abstands aus a
        a.clear()                                                     Freigeben von a für nächste Iteration
    else:                                                             Ausnahmefall, Farbabstand über Maximum
        b.insert(i, "null")
        a.clear()
```

Damit die bildeigene Farbe mit der geringsten Distanz zur vordefinierten Farbe zur letzteren verändert werden kann, werden iterativ die euklidischen Abstände verglichen und die Farben anschließend angepasst.

```
for i in range(K):
    res2[np.where((res2== [center[i]]).all(axis=2))] = [colorPaint[b][i]]
```

Suche die Farbe K aus Bild $res2$ und ersetze sie mit der Farbe K aus der Farbpalette $colorPaint$, anhand der Liste b

Pfadgenerierung

Das Verfahren von Gülzow et al. (2018), welches ebenfalls das Pinselmalen durch einen Industrieroboter zum Ziel hat, basiert auf der Skelettierung von Farbbereichen durch Zhang und Suen's Thinning-Algorithmus (Zhang und Suen 1984). Versuche, dieses Verfahren für dieses Projekt zu nutzen, scheiterten an der Tatsache, dass Skelettierungsverfahren einen unregelmäßigen Abstand zwischen Skelett und Kante des Körpers erstellen. Unabhängig von der Größe des Farbfeldes wird nur eine Skelettlinie in der Mitte des Feldes erstellt. Benötigt werden jedoch Linien, deren Abstand zur Kante des Farbfeldes und der jeweils nächsten Linie gleich der Pinselbreite ist. Dafür wurde ein Algorithmus entwickelt, der die Konturen von Farbfeldern iterativ um die Pinselmaße erodiert und diese Konturen als Pfade speichert. Zuerst wird eine der k Farben maskiert und diese Maske mit einem Gaußschen Weichzeichner verschwommen. Dies reduziert die Komplexität der Kanten um einen beliebigen Wert, damit eine angemessene Anzahl an Punkten und eine moderate Geschwindigkeit des Malprozesses erreicht werden. Anschließend wird die Kontur dieser Maske durch ein Detektionsverfahren (Suzuki und Abe 1985) gefunden und ausgegeben. Daraufhin wird die Maske durch Erosion geschrumpft. Der dafür verwendete Kern basiert auf dem vorher definierten Radius des Pinsels und wird durch Gewichtung angepasst. Beides kann manuell konfiguriert werden. Die Gewichtung sorgt für überlappende Malstriche, um lückenlose Farbbereiche zu garantieren. Ist die Maske vollständig erodiert, stoppt die Schleife und die Iteration wird mit dem nächsten Farbwert wiederholt. Dieser Prozess ähnelt dem Schälen einer Zwiebel, wobei jede Schale der Dicke des Pinsels entspricht. Ist die Zwiebel geschält, wird zum nächsten Farbfeld übergegangen.


```
for i in range(K):
    mask = cv2.inRange(res2, colorPaint[i], colorPaint[i])      Maskiert Farbe
    while(not done):
        mask = cv2.GaussianBlur(mask,(5,5),0)                   Weichzeichner für
                                                                Eckenreduktion
        ret, thresh = cv2.threshold(mask, 127, 255, 0)          Schwellenwert
        im2, contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
        ↵ cv2.CHAIN_APPROX_SIMPLE)                              Suche Konturen in Maske
        cv2.drawContours(res2, contours, -1, (0,255,0), 1)
        mask = cv2.erode(mask,kernelBrush,iterations = 1)       Schrumpft Kontur
        zeros = size - cv2.countNonZero(mask)
        if zeros==size:                                         Beende Schleife, wenn
                                                                Maske vollständig ge-
            done = True                                         schrumpft wurde
        contourExists == False
    done = False
```

contours ist ein mehrdimensionales Array. Die darin gelegenen Array's beinhalten zweidimensionale Koordinaten, die eigentlich als Vektoren zum Darstellen der gefundenen Kontur dienen. Diese Koordinaten können jedoch auch als kartesische Zielpunkte gesehen und vom Roboter angefahren werden. ***contours*** enthält dementsprechend die gewünschten Werte, welche beliebig weiterverarbeitet werden können.

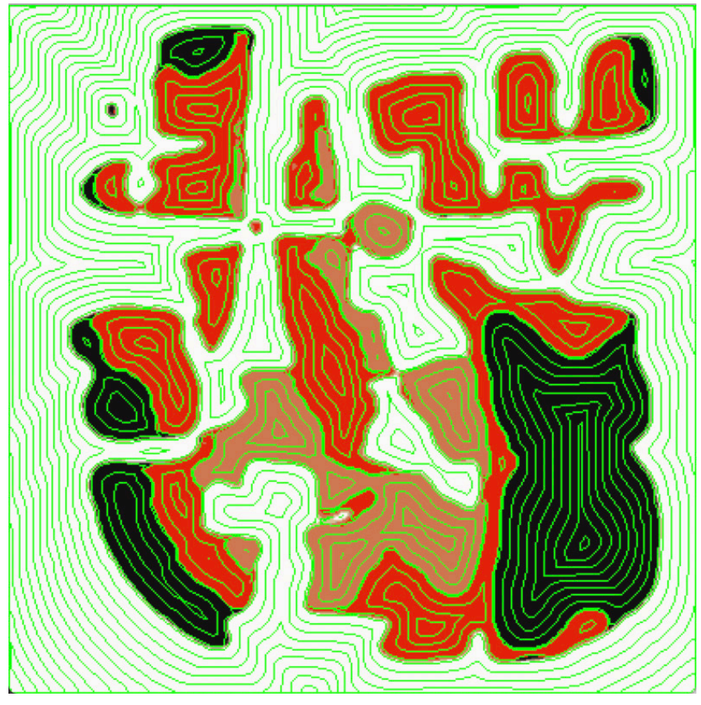
Abbildung 8,9,10,11, folgende Seiten

V.l.n.r.: Originalbild, mit k-Means auf 12 Farben reduziert, an Farbpalette angepasst, Zwiebelschalenkonturen

1

2

3



4

Kommunikation mit Robotersteuerung

Der physische Teil der Kommunikationsinfrastruktur von Computer und Robotersteuerung wird durch Ethernet realisiert. Zum schnellen Austausch von Daten zwischen beiden Komponenten bei KUKA Systemen wird das Softwarepaket „Ethernet-KRL“ benötigt. Dieses wird durch eine Ethernet-Leitung und dem KUKA Service Interface (KLI) von einem Laptop aus auf der KUKA Robot Control (KRC) installiert. KLI wird ebenfalls genutzt, um die Kommunikation zwischen Roboter und Computer in nahezu Echtzeit zu ermöglichen. Mithilfe des genannten Pakets können XML und RAW Daten in einem Netzwerk gesendet und empfangen werden und die Vektordaten des GAN-generierten Bildes als Koordinaten an die Robotersteuerung gegeben werden. Dafür wird neben dem Generierungsprozess und dem Konvertierungsprozess noch ein Socket zur Nachrichtenübergabe in das zuvor beschriebene Programm implementiert. Um Pfadkorrekturen für potenziell sensorische Systeme zu vereinfachen und eine menschenähnliche Kommunikation zwischen Hard- und Software zu erzielen, wird jeweils nur eine Koordinate bzw. Handlung an den Roboter gegeben. Erst wenn dieser die Handlung getätigt oder den gegebenen Punkt angefahren hat, wird das nächste Informationsstück ausgegeben. Der Computer wird in diesem Fall als Server konfiguriert, der während des Durchlaufens Daten an die Steuerung sendet und bei erfolgreicher Antwort fortfährt.

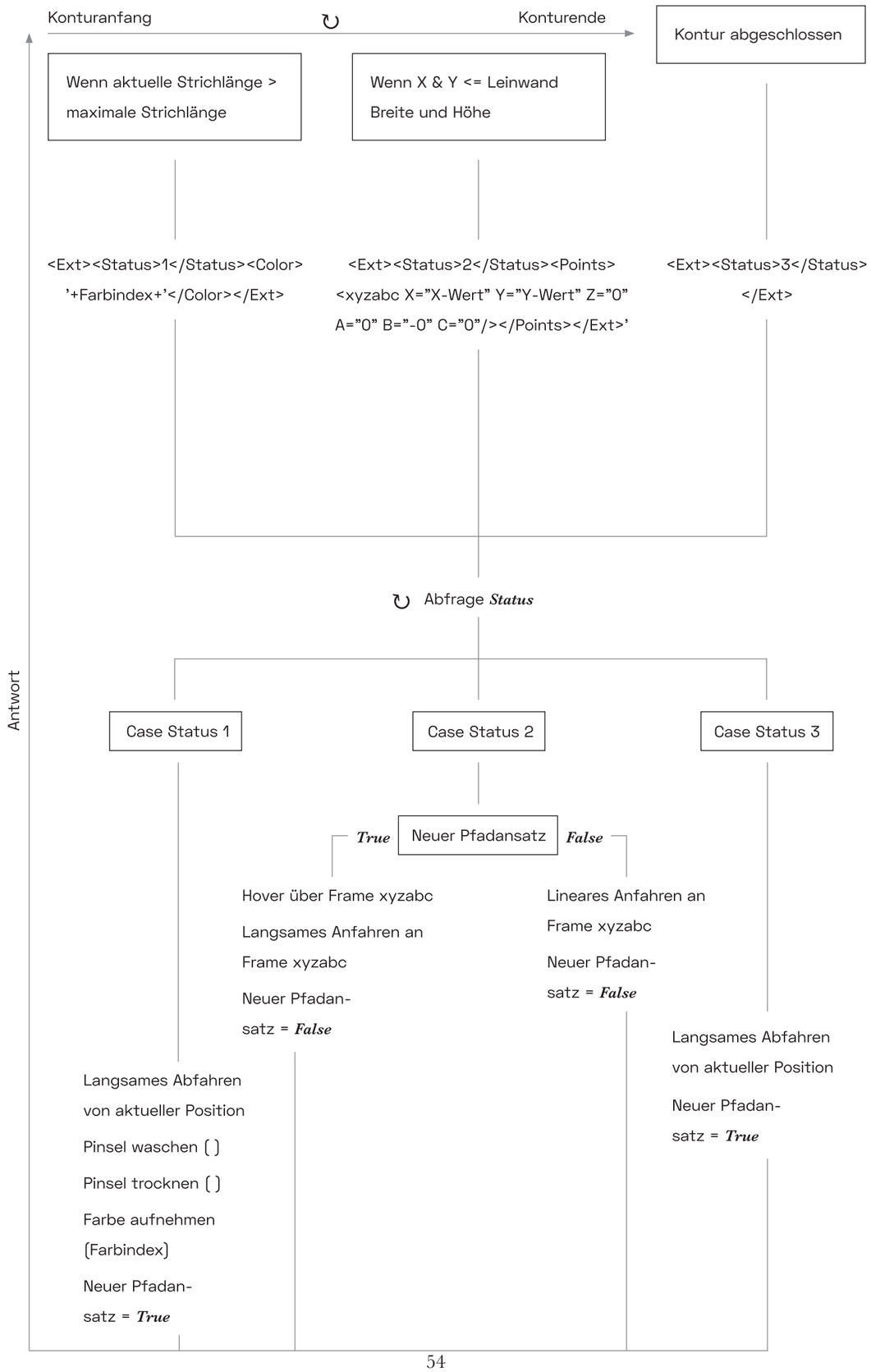
Der Malprozess erfordert neben dem Anfahren an diese Punkte die zuvor erwähnten zusätzlichen Handlungen. Diese werden als Status definiert und in Form eines Integers an die Robotersteuerung übergeben, welche beim Auslesen des Status das jeweilige Roboterunterprogramm auswählt. Die Unterprogramme unterteilen sich in Punktanfahrt, Pfadsprung und Farbwahl.

Bei Ersterem handelt es sich um ein simples Anfahren der ausgegebenen Koordinate im linearen Verfahren. Die lineare Bewegung sorgt für die Berechnung einer Geraden als Weg zwischen Start- und Endpunkt. Zu beachten ist bei der Punktanfahrt, ob sich der vorherige Punkt ebenfalls auf der Leinwand befand oder außerhalb, z.B. zu Beginn des Hauptprogrammes oder nach dem Wechseln der Farbe. Ist dies der Fall, muss der Roboter den Punkt langsam Anfahren, damit der Pinsel beim erstmaligen Auftreffen auf die Leinwand in der Z-Dimension nicht beschädigt wird.

Ein Pfadsprung tritt ein, wenn eine in sich geschlossene Kontur zu Ende befahren wurde und der Roboter zu einer neuen Kontur springt, ohne die Leinwand durch Farbwahl zu verlassen. Geschähe dies im selben Verfahren wie die Punktanfahrt, würde der Roboter einen ungewollten Strich zwischen dem Ende der Kontur A und dem Beginn der Kontur B malen. Beim Pfadsprung wird durch einen geometrischen Operator ausschließlich die Z-Achse verändert, um den Pinsel von der Leinwand zu entfernen und zum nächsten Punkt zu bringen, ohne die XY-Werte der generierten Punkte zu verändern.

Bei der Farbwahl werden verschiedene, durch Inline Commands geteachte (per Hand angefahren und gelehrte) Unterprogramme kombiniert. In diesem Fall wird der Roboter zuerst in eine sichere Home-Position gefahren. Anschließend führt er diverse Bewegungen in einer Wasserschüssel aus, um die Farbe zu entfernen. Das Abtupfen des Pinsels in einer danebenliegenden, mit Stoff befüllten Schüssel sorgt für ein trockenes Werkzeug. Nach der Rückfahrt zu der Home-Position wird der Pinsel in eine der Farbbehälter gefahren. Die Auswahl der Farbe wird dabei durch die Übergabe eines Integers durch das Server Programm bestimmt. Dieser entscheidet, welches Unterprogramm angewählt wird, wobei jedes dieser per Hand geteachten Programme einen unterschiedlichen Behälter anfährt.

Die Farbaufnahme respektiert die Eigenschaften von Acrylfarbe und erzielt durch unterschiedliche Geschwindigkeiten, Verweildauer zum Aufsaugen und Abtropfen der Farbe und ruckartige Ausfahrgeschwindigkeit einen Pinsel, der gesättigt ist, aber das Kontaminieren von benachbarten Farbbehälter durch ungewolltes Tropfen verhindert. Die Farbaufnahme ist dabei selten durch das Wechseln der Farbe motiviert und wird wesentlich öfter durch das Erreichen der maximalen Strichlänge ausgelöst. Diese wird im Server-Programm durch die Berechnung und Addition des euklidischen Abstands zwischen den Koordinaten kalkuliert. Ein beliebig definiertes Längenmaximum sorgt dafür, dass die Farbwahl ausgelöst und an den Roboter weitergeleitet wird. Sinn dieser Methode ist, dass der Benutzer die Distanz in Millimeter angeben kann, nachdem der Pinsel voraussichtlich zu viel Farbe verloren hat und keinen sauberen Strich mehr auftragen kann. In einem anthropomorpheren System würde dies durch eine visuelle Sensorik erfasst werden, ähnlich einem Menschen, der mit seinen Augen erkennt, wann sein Werkzeug neue Farbe benötigt.



Zur Kommunikation über XML wird der Computer als Server konfiguriert. Diesem wird eine statische IP in IP-Reichweite der statischen IP der Robotersteuerung (i.d.R. 172.31.1.148) zugewiesen. Der Roboter erreicht hier den Computer über den Port 59152. IP und Port werden im Hauptprogramm des Computers konfiguriert, um einen Socket zu erstellen, der den Datenaustausch zulässt. Mitunter muss die Firewall des Servers deaktiviert werden, um eine Kommunikation zu ermöglichen. Der Server gibt der Steuerung Daten, wie beispielsweise den Koordinaten, im XML Format. Antworten geschehen in der Regel im gleichen Format und können genutzt werden, um Asynchronität zu vermeiden.

Abbildung 12, links

Schema des Hauptprogramms der Robotersteuerung. Um den Aufbau so flexibel wie möglich zu halten, haben Leinwand, Wasser-, Tuch- und Farbbehälter ein eigenes Koordinatensystem, eine sogenannte "Base". Verändert sich die Platzierung dieser Elemente oder werden Maluntergründe mit unterschiedlichen Dimensionen genutzt, müssen drei beliebige Punkte angefahren werden, um die Base neu zu definieren. Ein Aufstellen der Installation und das Wählen der Behälter ist somit flexibel möglich.

Abbildung 13

Ausschnitt eines voll maschinellen und generativen Kunstwerkes. Acryl auf Leinwand, 40 x 40 cm.

Abbildung 14

Acryl auf Leinwand, 70 x 70 cm.

Abbildung 15

Le Rêve, Picasso (1932). Neu interpretiert von einem Roboter. Acryl auf Leinwand, 40 x 40 cm.

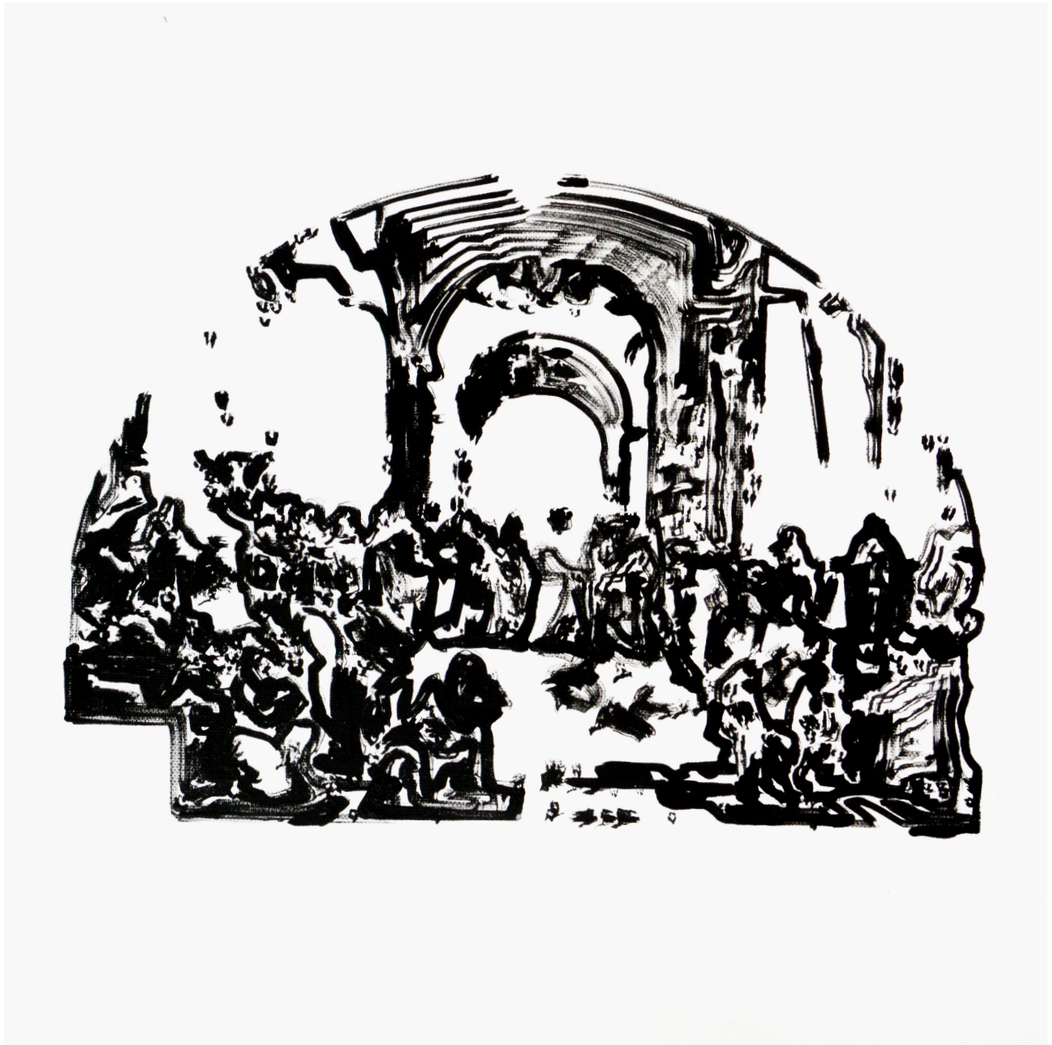


Abbildung 16

Die Schule von Athen, Raffael (1510-1511). Neu interpretiert von einem Roboter. Acryl auf Leinwand, 30 x 30 cm.

6. Gestalterische Reflexion

Im Hinblick auf die Qualität der Bilder wird schnell deutlich, dass das flächig-abstrakte nur schwer mit den gegebenen motorischen Werkzeugen dargestellt werden kann. Das Erreichen einer gleichmäßigen Fläche, wie viele abstrakte Werke des GAN's vorgeben, gelingt mit Pinseln kaum. Für die Zukunft wäre das Nutzen eines Messers oder einer Farbrolle interessant, um die synthetischen Anlehnungen an Barnett Newman korrekt auf Leinwand zu bringen. Die Abwesenheit jeglicher Kalkulation von Materialeigenschaften des Pinsels und der Farbe sorgt für einen dynamischen, aber schwer zu berechnenden Strich. Die Abhängigkeit zum GAN-generierten Input hemmt einen ästhetischen Zufall und ergibt eher einen Wettbewerb mit dem virtuellen Original. Dass die durch Farbverläufe dominierten Werke schwer durch Farbreduktion und komplexe Strichdynamik an Ästhetik gewinnen, ist nicht überraschend. Die generierten Porträts, die selbst im gesamten RGB-Farbraum nur schwer zu erkennende Organe beinhalten, können ebenfalls kaum vom Roboter abgebildet werden. Es bildet sich also ein schwieriges Spiel aus der Unfähigkeit flächig zu malen und der Schwierigkeit komplexe Motive realistisch zu generieren. Diese Probleme lassen sich durchaus bekämpfen. Dabei gibt es einige Parameter und Strukturen die so verändert werden könnten, dass die Ergebnisse an Wertigkeit gewinnen. So würde das Verkürzen der Strichlänge in einem impressionistischen Stil resultieren und das Selektieren von geeignetem Input würde eine Garantie mit sich bringen, ein Bild mit höherer Ästhetik zu erhalten. Dies reicht jedoch nicht aus, um das Streben nach einer maschinellen Autorenschaft zu beseitigen. So relevant die ästhetische Wertung auch ist, diese Arbeit unterliegt nicht primär der Metrik der Schönheit. Das Erstellen von Roboterkunst mit angemessener optischer Qualität ist ein bereits abgedecktes Feld. Es ist das Experimentieren und Suchen nach einem Weg der Maschine eine Urheberschaft zuzuweisen, die diese Arbeit motiviert. Es gilt zu beachten, dass diese Maschine nicht im Vergleich mit einem erwachsenen Menschen steht. Die geringe Lernzeit und die verhältnismäßig einfache Logik, auf der die Maschine beruht, macht eine Konkurrenz zu einem Kleinkind realistischer. Ohne Frage ist es stets attraktiv die Programme und ihre Resultate in Richtung Ästhetik zu lenken. Doch würde man diese Philosophie in den Superlativ stellen, so wäre die Lösung ein einfacher Drucker, der den Input akkurat auf Papier bringt, ohne fundamental anders als ein Industrieroboter zu arbeiten. Erst die Frage nach nicht-menschlicher Agentenschaft und Kreativität macht diese Arbeit wirklich relevant. Für die Zukunft ist es dementsprechend sinnvoll, einen Greifarm für universelle Werkzeugwahl zu nutzen, Sensorik und reaktionäre Systeme zu installieren und insbesondere sämtliche Arten und Qualitäten von generativer und menschlicher Kunst malen zu lassen. Obwohl diese Arbeit ihren Zeitplan eingehalten hat, leidet sie darunter, dass das Experimentieren und progressive Anpassen eines komplexen Systems nicht ausreichend stattgefunden hat. Dennoch kann die Arbeit als Erfolg angesehen werden. Insbesondere die Entwicklung eines neuartigen Algorithmus zum Malen eines Bildes hat einen signifikanten Wert. So wurde ein Weg gefunden, beschrieben und performant implementiert, wie ein beliebiges Bild schnell und zuverlässig in Koordinaten konvertiert werden kann, die ein beliebiger Roboter kartesisch anfahren kann. Zusätzlich wurde ein hohes Maß an Konfigurationsmöglichkeiten eingebettet. So berücksichtigt der Algorithmus Pinselgröße, maximale Strichlänge, vordefinierte Farbpaletten und Platzierung der Leinwand, Wasser-, Tuch- und Farbbehälter. Sowohl der Thinning-Algorithmus, als auch seine Implementierung haben das Potenzial diverse Tätigkeiten, von experimental bis industriell, zu erfüllen. Dies übersteigt einerseits den ursprünglichen Anspruch an dieses Werk und bietet anderseits ein ideales Grundgerüst für zukünftige Experimente mit erweiterter Ausstattung. Zuletzt ist es die außerordentliche Vielfalt und Komplexität an Kompetenzen, welche im Verlaufe des Projekts gewonnen wurden, die diese Arbeit außerordentlich wertvoll für mich macht. Von der Statikberechnung und dem Konstruieren eines normgerechneten Fundaments, der intensiven Programmierung eines Roboters, der wissenschaftlichen Vertiefung in sämtliche Ansätze der generativen Machine-Learning-Algorithmik, bis zum Identifizieren von geeigneten Farbquantisierungsmethoden: nach Erfüllung dieser Themen ist es durchaus angebracht anzuzweifeln, dass Farbauftrag und Inputwahl längerfristig die Ausschöpfung des vollen Potenzials verhindern würden. Das Bewegen in einem wenig erforschten Feld impliziert ein ineffizientes Suchen nach wertvollen Antworten. Wird dieser Ineffizienz mit ausreichend Zeit entgegnet, ist es plausibel, dass die Qualität erreicht wird, die sich das ästhetisch höchst kritische Wesen Mensch wünscht.

7. KI als Ersatz menschlicher Künstler

Die Industrie wird einen bemerkbaren Umschwung durch maschinelles Lernen erleben. Computer haben bereits in ihrer 70-jährigen Erfolgsgeschichte existente Märkte verändert, aufgehoben oder neu erschaffen. Genauso wie Mikrochips hat auch KI das Potential, ein eigenes Zeitalter auszurufen. Zu belegen ist dies mit der grundlegend anderen Art und Weise, wie Computer vor probalistischen Modellen wie im Machine Learning arbeiteten. Die Fähigkeit moderner Geräte, in Millisekunden scheinbar komplexe mathematische Operationen zu berechnen, schneller als es irgendein ein Mensch könnte, und die augenscheinlich enorme Menge an Prozessorleistung ist nicht selten täuschend. So kann schnell angenommen werden, Computer würden fundamental anders funktionieren. Insbesondere, weil sie Aufgaben in der Mathematik in mit Menschen unvergleichlichen Zeiträumen erledigen, aber in anderen Feldern wie Kommunikation, Verständnis der generellen Umwelt oder kontextueller Informationsbewertung außerordentlich weit hinter der Leistung des menschlichen Gehirns liegen. Ein Smartphone benötigt Bruchteile einer Sekunde, um beispielsweise 7924*914,29 zu berechnen, doch kein Supercomputer der Welt kann eine sinnvolle Unterhaltung mit einem Menschen führen – eine Fähigkeit, die Kinder im Grundschulalter längst gemeistert haben. Dies lässt schnell vermuten, hier würden zwei komplett unterschiedliche Strukturen agieren.

Die scheinbar prinzipielle Differenz zwischen den Aufgaben von Computern und Menschen hat dennoch oft mit der unterschätzten Komplexität der Intuition zu tun. Die Anzahl an Variablen und Konditionen, die in einer sinnvollen Unterhaltung mit einem Menschen agieren, reichen weit über traditionelle Software hinaus. Minimale Veränderungen in Gestik, Stimme oder Augenfokus wird vom Gesprächspartner in Millisekunden gedeutet. Die Art und Weise, welche Schlussfolgerungen aus welchen Aussagen getroffen werden, hilft uns ohne jegliche Anstrengung, eine unerkannt komplexe Analyse vom Charakter unseres Gesprächspartners zu erstellen. Dabei ist nicht nur der Berechnungsaufwand intuitiver Handlungen der scheidende Faktor zwischen Mensch und Maschine. Vielmehr fällt es dem Menschen schwer, eine Anleitung für sein Handeln zu formulieren. Während

logische Operationen wie in der Mathematik einem klaren und verhältnismäßig winzigen Reglement unterliegen, sind Menschen selbst kaum in der Lage, intuitives Verhalten bewusst zu erfassen und zu beschreiben. Einem Agenten beizubringen sich mit Menschen zu unterhalten oder ein Gesicht zu malen, fällt Computern oft deswegen schwer, weil der Programmierer selbst keine klare Anleitung für diese Handlungen hat. Verständlich wird dies beim Fahrradfahren. Diese Disziplin besticht durch das Korrigieren und Manipulieren von Gewicht und Momentum im Millisekundenbereich. Wer glaubt, einer Maschine oder einem Menschen aber das Fahrradfahren mit der Erklärung von Regeln beizubringen, wird schnell daran scheitern, die Regeln, die einen selbst auf dem Sattel halten, überhaupt zu erkennen. Machine Learning Algorithmen verzichten auf die Notwendigkeit, deterministische Regeln festgelegt zu haben und arbeiten stattdessen mit probalistischen Lernsystemen. Computer beschränken sich dadurch nicht mehr ausschließlich auf Aufgaben, dessen Regeln von Menschen einfach formuliert werden können, sondern erlernen eigene, komplexe und versteckte Regeln. Der schwindende Kontrast zwischen den Aufgabenfeldern von Menschen und Maschinen ist also nicht (nur) durch einen Unterschied von Funktionsprinzipien zwischen Gehirn und Computer zu erklären. Stattdessen zählt die Schwierigkeit, komplexe und intuitive Aufgaben zu beschreiben, zu den Gründen für die scheinbar prinzipielle Differenz.

Das Erschließen von Aufgabenfeldern, die vorher für Maschinen kaum zugänglich waren, wirft bei einigen Künstlern die Frage auf, ob KI ihre Arbeit ersetzt (Pfeiffer 2018). Auch der Autor dieser Arbeit wurde im Projektverlauf mit dieser Frage konfrontiert.

Diskutiert man allerdings die Möglichkeit, menschliche Künstler durch Maschinen zu ersetzen, deckt man zugleich wichtige Limitierungen von Maschinen, die Kunst erstellen, auf. Die enorme Subjektivität von Kunst macht diese Disziplin zu einer der genuin-humansten Felder des menschlichen Lebens. Kaum eine Fähigkeit verweigert Computern die Übernahme so wie künstlerische Kreativität. Grund dafür ist neben der Subjektivität die Tatsache, dass Kunst nicht nur einen generativen, sondern auch einen bewertenden Charakter hat. Pablo Picasso's angebliches Argument, „Computer sind nutzlos. Sie können nur Antworten geben.", ist eine Anspielung auf die Unfähigkeit, ihre Handlungen zu hinterfragen und dadurch neue Perspektiven zu geben. Diese Art von Bewertung des eigenen Handelns ist Grundlage für Innovation und gilt als fundamentaler Bestandteil

der modernen Kunst. Während bis zur Zeit der Fotografie die Fähigkeit, Dinge realistisch darzustellen, Maß der Qualität war, hat Kunst diese Identität durch den Fotoapparat als Massenprodukt verloren. Aufgabe der neuen Kunst war es, dorthin zu gehen, wo keiner zuvor war, indem Existentes gemischt, zerstört oder kombiniert wurde, um neue Sichtweisen zu kommunizieren. Auch die Anweisung des Programmierers müssten von einem autonomen Programm hinterfragt werden. Zusätzlich ist die künstlerische Interpretation eines Menschen mit einem enormen Datenaufwand verbunden. Objekte, Lichtwirkung, Farben, aber vor allem auch Gefühle, Symbole, politische Ereignisse, Meinungen, Unterhaltungen, Geschichten und Erlebnisse werden in der Kunst verarbeitet. Aufgrund des oft geringen Aufwands von beispielsweise minimalistischer Kunst kann argumentiert werden, dass der größte Wert moderner Kunst in der Verarbeitung dieser Erlebnisse liegt.

> "A 'program' which could produce brilliant music would have to wander around the world on its own, fighting its way through the maze of life and feeling every moment of it. It would have to understand the joy and loneliness of a chilly night wind, the longing for a cherished hand, the inaccessibility of a distant town, the heartbreak and regeneration after a human death. It would have to have known resignation and world-weariness, grief and despair"
>
> (Hofstadter, 1999)

Dieser Datenaufwand übersteigt bei Weitem die Komplexität von aktuellen KI-Applikationen. Die Notwendigkeit eines Bewusstseins durch die Hinterfragung von Input macht die Kunsterstellung zu einer sehr menschlichen Fähigkeit und damit potenziell zu einem Indikator für Intelligenz. Neben der notwendigen Datenmenge und Interpretationseigenständigkeit machen es weitere Faktoren Maschinen schwer bis unmöglich, die Arbeit des menschlichen Künstlers zu ersetzen. Kunst und ihre Qualität werden vor allem von ihren Betrachtern gewertet. Eine objektive Bewertung in der Kunst ist nur begrenzt sinnvoll. Kunst als schaffende Tätigkeit agiert stets unter der Prämisse, allein das Werk vom Betrachter bewerten zu lassen. Kreativität beziehungsweise ihre Attribute als Fundament der Kunst werden von Colton et al. als sekundäre Qualität angesehen. Dabei definiert der Autor dieser Qualitäten, John Locke, die primäre Qualität als beobachterunabhängige, intrinsische Eigenschaft und die sekundäre Qualität als wahrnehmungsabhängig und damit fest mit einem beliebigen Beobachter verbunden. Kreativität und im erweiterten Sinne auch Kunst als sekundäre Qualität beruht also auf Beobachtung, einer Wertung und daraus reagierender Handlung. Ein Beobachter ist die Maschine jedoch nicht. Sie besitzt weder die sensorische Hardware noch die Software, um im gesamten Kreationsprozess eine reaktive und reflektierte Rolle einzunehmen.
Obwohl es nicht selten Diskrepanzen in der Wertung von Kunst zwischen Experten und Laien gibt, ist es gerechtfertigt, die subjektive Bewertung eines beliebigen Betrachters zu respektieren, spätestens, wenn ihm Kontextinformationen des Werkes mitgeteilt werden. Ein Kunstwerk, das nur von einstudierten Personen bewertet werden darf, kann kaum als gute Kunst bezeichnet werden. Damit hat der Mensch als Betrachter eine nicht zu ersetzende Macht über die Maschinen. Selbst wenn Computer den menschlichen Künstlern ihre Arbeit wegnehmen würden, wäre dieser Prozess nur im Einvernehmen mit den Menschen möglich. Denn diese selbst entscheiden, was für sie (gute) Kunst ist und damit, wer ein Künstler für sie ist. Ein Mensch, der in seinem eigenen Werk Kunst sieht, hat damit seine Rolle als Künstler bereits sichergestellt. Selbst wenn Maschinen erlernen, Kunst nach eigenen Kriterien zu bewerten, so wäre dies nur ein Zusatz, aber kein Ersatz für die subjektive und oft unfreiwillige Empfindung eines Menschen, der ein Kunstwerk betrachtet.

„Art is a human enterprise, defined by experience, of both artist and audience.“

(Locke und Nidditch 1975)

8. Fazit

Der Begriff künstliche Intelligenz leidet traditionell an einem Missbrauch, der meist zwecks Aufmerksamkeitsgenerierung benutzt wird – häufig unbewusst durch fehlende Expertise, aber nicht selten bewusst, um ein Produkt zu bewerben oder Investoren anzulocken. McCormack et al. argumentieren, dass das Missverstehen von heutiger Künstlicher Intelligenz in der durch Komplexität hervorgerufenen Intransparenz liegt und weiter durch anthropomorphe Schnittstellen wie Chatbots oder menschenähnliche Avatare angetrieben wird. In GAN's, wie sie in dieser Arbeit verwendet wurden, sehen sie nicht ausreichend Autonomie, Authentizität, Autorenschaft und Intention. Sie kritisieren auch die replikative Natur der Kunstgenerierung durch GAN's.

> "Human artists do not learn to create art exclusively from prior examples. They can be inspired by experience of nature, sounds, relationships, discussions and feelings that a GAN is never exposed to and cannot cognitively process as humans do."
>
> (McCormack et al. 2019)

Die hier vorgestellte Maschine ist kein sozialer Agent. Sie erfährt nicht mehr als ihre wenigen Gigabyte an Input und zeigt weder die Motivation noch die Grundlage, sich expressiv zu verhalten. Die Banalität des Wirkungsprinzips vieler Machine Learning Algorithmen, das Finden einer statistischen Distribution, ist für McCormack ein Zeichen stark limitierter Autonomie und auch der hier vorgestellten Maschine fehlt es offensichtlich an dieser Eigenständigkeit. Wäre der Autor dieser Arbeit nicht hier, gäbe es die gezeigten Gemälde nicht. Nun kann argumentiert werden, dass das gleiche für jeden Menschen und seine Eltern gilt, ohne die der besagte

Mensch nicht auf der Welt wäre. Doch der Roboter und insbesondere der Computer waren auch vor dieser Arbeit voll funktionsfähige Geräte, die häufig in Betrieb waren. Doch weder der Roboter noch der Computer hatten die Muße, spontan ein Bild zu malen, als sie am Strom angeschlossen wurden.

Gleichzeit schlagen aber Mohammad Majid al-Rifaie und Mark Bishop (al-Rifaie und Bishop 2015), analog zu John Searle's Konzept von starker und schwacher KI (Searle 1980), eine Einteilung zwischen schwacher und starker Kreativität vor. Starke Kreativität erfordert dabei laut ihnen ein direktes Verständnis von dem, was der Künstler abbildet oder abbilden will. Schwache Kreativität verzichtet auf diese Anforderung und zielt eher auf eine Simulation der menschlichen Kreativität. Als Gegenargument zur imitierenden Lernweise, beispielsweise von GAN's, die nur aus existenten Kunstwerken lernen, weisen al-Rifaie und Bishop auf ein Taubenexperiment hin (Watanabe 2010). In diesem werden die Vögel auf das Erkennen von guten Bildern, also Werke mit allgemein anerkannten ästhetischen Eigenschaften, trainiert und können zuletzt erfolgreich zwischen völlig neuen guten und schlechten Werken unterscheiden. Ohne Zweifel hinterfragt dies die These, Menschen würden in ihrem Kunstempfinden und daraus resultierender Kunsterschaffung völlig unabhängig von existenten Werken agieren. Oberflächlich gesehen haben GAN's bereits ihre Fähigkeit, digitale Darstellung von Kunst mit der Qualität menschlicher Künstler zu erstellen, gezeigt. In einer Studie wurden 85% der Werke, die von einem auf abstrakten Expressionismus trainiertes GAN erstellt wurden, als menschlich erzeugt fehlinterpretiert (Elgammal et al. 2017). Prädikate wie „intentional" oder „kommunikativ" wurden dabei den maschinellen Werken von menschlichen Kritikern gegeben. Obgleich eine starke Abhängigkeit zum menschlichen Autor, insbesondere in der deterministischen Programmierung der Robotersteuerung, herrscht, gilt es dennoch zu beachten, dass dieser Diskurs in einer Sphäre von uneindeutigen Metriken und hoher Subjektivität stattfindet. Allen drei Grundpfeilern, Intelligenz, Kreativität und Kunst, fehlt es an universell akzeptierten Definitionen. So wird noch heute in wissenschaftlichen Kreisen diskutiert, was „Intelligenz" exakt ist, wie sie zu messen ist und ob sie als Alleinstellungsmerkmal des Menschen taugt. Der Kreativitätsbegriff wird von vielen Wissenschaftlern direkt umgangen (Plucker und Makel 2010) oder gilt als kontrovers (Tang and Werner, 2017; Hernández-Romero, 2017) und über die Definition von „Kunst" schreibt Kunstkritiker Keith Stuart: „Here is a good way to tell if a critic is having a moment of madness: they will attempt to define art." (Stuart 2014).

Die Diskussion um künstliche Intelligenz, Kreativität und Kunst gleicht im Moment noch dem Öffnen von Pandoras Büchse. Nur wenige Fragen und der Diskurs kollabiert in diverse Wissensdomänen wie Informatik, Neurobiologie, Quantenphysik, Mathematik, Robotik oder Kunstwissenschaft. Wie soll über Kunst, Kreativität und Intelligenz diskutiert werden, wenn für alle drei Begriffe eine nennenswert akzeptierte Definition fehlt? Die einhergehende Explosion an erweiterten Fragen und Themenfelder resultiert häufig in grundlegenden Debatten der Philosophie, insbesondere der Metaphysik. Ob das Gehirn vollständig simuliert werden kann, ist eine Frage der Deterministik und damit nach dem freien Willen und der Kausalität in Quantenmechanik und relativistischer Physik. Das Konzept des Bewusstseins dominiert die Neurowissenschaft und die Implikationen im Erschaffen eines neuen, intelligenten Wesens provoziert die Diskussion um die Existenz eines Gottes. Das Einnehmen einer absoluten Meinungsposition in dieser Debatte bleibt dementsprechend riskant und gegeben falls verfrüht.

Dass in dieser Arbeit ein intelligenter Agent autonom authentische Kunst erschaffen hat, ist sehr einfach anzuzweifeln. Doch auch darin liegt ein Wert. So ist es die Schwierigkeit, diesen Zweifel mit signifikanter Objektivität zu belegen, die ein wichtiges Licht auf unsere gravierende Unwissenheit über das menschliche Wesen und die Welt, in der es sich bewegt, wirft.

Eidesstattliche Erklärung

Hiermit versichere ich, dass diese Arbeit selbstständig von mir erstellt und verfasst wurde. Ich versichere, dass die Arbeit für keine anderweitigen Prüfungszwecke vorgelegt wurde und ausschließlich die angegebenen Quellen oder Hilfsmittel verwendet wurden. Zuletzt versichere ich hiermit, dass wörtliche Zitate als solche gekennzeichnet wurden.

Ort, Datum, Unterschrift

8. Literaturverzeichnis

al-Rifaie, Mohammad Majid; Bishop, Mark (2015): Weak and Strong Computational Creativity. In: Tarek R. Besold, Marco Schorlemmer und Alan Smaill (Hg.): Computational Creativity Research: Towards Creative Machines, Bd. 7. Paris: Atlantis Press (Atlantis Thinking Machines), S. 37–49.

Anonym (2018): Is artificial intelligence set to become art's next medium? AI artwork sells for $432,500 — nearly 45 times its high estimate — as Christie's becomes the first auction house to offer a work of art created by an algorithm. Christie's. Online verfügbar unter https://www.christies.com/features/A-collaboration-between-two-artists-one-human-one-a-machine-9332-1.aspx, zuletzt geprüft am 30.06.2019.

Arjovsky, Martin; Bottou, Léon (2017): Towards Principled Methods for Training Generative Adversarial Networks. Online verfügbar unter http://arxiv.org/pdf/1701.04862v1.

Arjovsky, Martin; Chintala, Soumith; Bottou, Léon (2017): Wasserstein GAN. Online verfügbar unter http://arxiv.org/pdf/1701.07875v3.

Armanious, Karim; Jiang, Chenming; Fischer, Marc; Küstner, Thomas; Nikolaou, Konstantin; Gatidis, Sergios; Yang, Bin (2018): MedGAN: Medical Image Translation using GANs. Online verfügbar unter http://arxiv.org/pdf/1806.06397v2.

Barrat, Robbie (2017): art-DCGAN. Modified version of Soumith Chintala's torch implementation of DCGAN with a focus on generating artworks. Online verfügbar unter https://github.com/robbiebarrat/art-DCGAN, zuletzt aktualisiert am 09.01.2019, zuletzt geprüft am 24.06.2019.

Bhagavatula, Partha; Claudianos, Charles; Ibbotson, Michael; Srinivasan, Mandyam (2009): Edge detection in landing budgerigars (Melopsittacus undulatus). In: PloS one 4 (10), e7301. DOI: 10.1371/journal.pone.0007301.

Burgess, Christopher P.; Higgins, Irina; Pal, Arka; Matthey, Loic; Watters, Nick; Desjardins, Guillaume; Lerchner, Alexander (2018): Understanding disentangling in $β $-VAE. Online verfügbar unter http://arxiv.org/pdf/1804.03599v1.

Dinh, Laurent; Krueger, David; Bengio, Yoshua (2014): NICE: Non-linear Independent Components Estimation. Online verfügbar unter http://arxiv.org/pdf/1410.8516v6.

Elgammal, Ahmed; Liu, Bingchen; Elhoseiny, Mohamed; Mazzone, Marian (2017): CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms. Online verfügbar unter http://arxiv.org/pdf/1706.07068v1.

Fautrel, Pierre; Caselles-Dupré, Hugo; Vernier, Gauthier (2019): Obvious - creation process. Unter Mitarbeit von Alexandre Guillemin. https://obvious-art.com. Online verfügbar unter https://web.archive.org/web/20180815173733/https://obvious-art.com/index.html, zuletzt geprüft am 26.06.2019.

Frid-Adar, Maayan; Diamant, Idit; Klang, Eyal; Amitai, Michal; Goldberger, Jacob; Greenspan, Hayit (2018): GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification. In: Neurocomputing 321, S. 321–331. DOI: 10.1016/j.neucom.2018.09.013.

Gatys, Leon A.; Ecker, Alexander S.; Bethge, Matthias (2015): A Neural Algorithm of Artistic Style. Online verfügbar unter http://arxiv.org/pdf/1508.06576v2.

Goodfellow, Ian J.; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil et al. (2014): Generative Adversarial Networks. Online verfügbar unter http://arxiv.org/pdf/1406.2661v1.

Gülzow, Jörg; Grayver, Liat; Deussen, Oliver (2018): Self-Improving Robotic Brushstroke Replication. In: Arts 7 (4), S. 84. DOI: 10.3390/arts7040084.

Hiasa, Yuta; Otake, Yoshito; Takao, Masaki; Matsuoka, Takumi; Takashima, Kazuma; Prince, Jerry L. et al. (2018): Cross-modality image synthesis from unpaired data using CycleGAN: Effects of gradient consistency loss and training data size. Online verfügbar unter http://arxiv.org/pdf/1803.06629v3.

Hofstadter, Douglas (1999): Gödel, Escher, Bach: An Eternal Golden Braid. Online verfügbar unter https://books.google.de/books/about/Gödel_Escher_Bach.html?id=8cgtAAAAYAAJ, zuletzt geprüft am 04.07.2019.

Huang, Zhewei; Heng, Wen; Zhou, Shuchang (2019): Learning to Paint with Model-based Deep Reinforcement Learning. Online verfügbar unter http://arxiv.org/pdf/1903.04411v2.

Iqbal, Talha; Ali, Hazrat (2018): Generative Adversarial Network for Medical Images (MI-GAN). In: J Med Syst 42 (11), S. 11. DOI: 10.1007/s10916-018-1072-9.

Jin, Yanghua; Zhang, Jiakai; Li, Minjun; Tian, Yingtao; Zhu, Huachun; Fang, Zhihao (2017): Towards the Automatic Anime Characters Creation with Generative Adversarial Networks. Online verfügbar unter http://arxiv.org/pdf/1708.05509v1.

Karras, Tero; Aila, Timo; Laine, Samuli; Lehtinen, Jaakko (2017): Progressive Growing of GANs for Improved Quality, Stability, and Variation. Online verfügbar unter http://arxiv.org/pdf/1710.10196v3.

Karras, Tero; Laine, Samuli; Aila, Timo (2018): A Style-Based Generator Architecture for Generative Adversarial Networks. Online verfügbar unter http://arxiv.org/pdf/1812.04948v3.

Kazemifar, Samaneh; McGuire, Sarah; Timmerman, Robert; Wardak, Zabi; Nguyen, Dan; Park, Yang et al. (2019): MRI-only brain radiotherapy: assessing the dosimetric accuracy of synthetic CT images generated using a deep learning approach. In: Radiotherapy and Oncology 136, S. 56–63. DOI: 10.1016/j.radonc.2019.03.026.

Kheradpisheh, Saeed Reza; Ghodrati, Masoud; Ganjtabesh, Mohammad; Masquelier, Timothée (2016): Deep Networks Can Resemble Human Feed-forward Vision in Invariant Object Recognition. In: Scientific reports 6, S. 32672. DOI: 10.1038/srep32672.

Kingma, Diederik P.; Dhariwal, Prafulla (2018): Glow: Generative Flow with Invertible 1x1 Convolutions. Online verfügbar unter http://arxiv.org/pdf/1807.03039v2.

Kingma, Diederik P.; Welling, Max (2013): Auto-Encoding Variational Bayes. Online verfügbar unter http://arxiv.org/pdf/1312.6114v10.

Ledig, Christian; Theis, Lucas; Huszar, Ferenc; Caballero, Jose; Cunningham, Andrew; Acosta, Alejandro et al. (2016): Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. Online verfügbar unter http://arxiv.org/pdf/1609.04802v5.

Liu, Ziwei; Luo, Ping; Wang, Xiaogang; Tang, Xiaoou (2016): Large-scale CelebFaces Attributes (CelebA) Dataset. Online verfügbar unter http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html, zuletzt aktualisiert am 29.07.2016, zuletzt geprüft am 25.06.2019.

Lloyd, S. (1982): Least squares quantization in PCM. In: IEEE Trans. Inform. Theory 28 (2), S. 129–137. DOI: 10.1109/TIT.1982.1056489.

Locke, John; Nidditch, Peter H. (Hg.) (1975): An essay concerning human understanding. (12. Dr.). Oxford: Clarendon Press.

Maaløe, Lars; Fraccaro, Marco; Liévin, Valentin; Winther, Ole (2019): BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling. Online verfügbar unter http://arxiv.org/pdf/1902.02102v1.

Mahendran, Aravindh; Vedaldi, Andrea (2014): Understanding Deep Image Representations by Inverting Them. Online verfügbar unter http://arxiv.org/pdf/1412.0035v1.

McCormack, Jon; Gifford, Toby; Hutchings, Patrick (2019): Autonomy, Authenticity, Authorship and Intention in Computer Generated Art. Online verfügbar unter https://books.google.de/books?id=vpiRDwAAQBAJ, zuletzt geprüft am 04.07.2019.

Mescheder, Lars; Geiger, Andreas; Nowozin, Sebastian: Which Training Methods for GANs do actually Converge? Online verfügbar unter http://arxiv.org/pdf/1801.04406v4.

Micikevicius, Paulius; Narang, Sharan; Alben, Jonah; Diamos, Gregory; Elsen, Erich; Garcia, David et al. (2017): Mixed Precision Training. Online verfügbar unter http://arxiv.org/pdf/1710.03740v3.

Perrett, David I.; Burt, D.Michael; Penton-Voak, Ian S.; Lee, Kieran J.; Rowland, Duncan A.; Edwards, Rachel (1999): Symmetry and Human Facial Attractiveness. In: Evolution and Human Behavior 20 (5), S. 295–307. DOI: 10.1016/S1090-5138(99)00014-8.

Pfeiffer, Andreas (2018): Creativity & Technology In The Age Of AI. Unter Mitarbeit von Angélique Dailcroix, Thomas Jouenne, Meredith Keeve, Scott Citron, Julia Zieger und Tina Touli. Adobe. Online verfügbar unter https://www.slideshare.net/adobe/creativity-technology-in-the-age-of-ai-119007383, zuletzt geprüft am 27.06.2018.

Plucker, Jonathan A.; Makel, Matthew C. (2010): Assessment of Creativity. In: James C. Kaufman und Robert J. Sternberg (Hg.): The Cambridge Handbook of Creativity. Cambridge: Cambridge University Press, S. 48–73.

Radford, Alec; Metz, Luke; Chintala, Soumith (2015): Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Online verfügbar unter http://arxiv.org/pdf/1511.06434v2.

Reed, Scott; van den Oord, Aäron; Kalchbrenner, Nal; Colmenarejo, Sergio Gómez; Wang, Ziyu; Belov, Dan; Freitas, Nando de (2017): Parallel Multiscale Autoregressive Density Estimation. Online verfügbar unter http://arxiv.org/pdf/1703.03664v1.

Safer, Morley (1993): September 19, 1993: Morley Safer's infamous 1993 art story. Online verfügbar unter https://www.youtube.com/watch?v=PjXKsZZ_pWc, zuletzt geprüft am 27.06.2019.

Salimans, Tim; Karpathy, Andrej; Chen, Xi; Kingma, Diederik P. (2017): PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. Online verfügbar unter http://arxiv.org/pdf/1701.05517v1.

Searle, John R. (1980): Minds, brains, and programs. In: Behav Brain Sci 3 (3), S. 417–424. DOI: 10.1017/S0140525X00005756.

Stuart, Keith (2014): Video games and art: why does the media get it so wrong? Another critic has taken another sideways glance – but the medium is strong enough to resist these withering ovations. The Guardian. Online verfügbar unter https://www.theguardian.com/technology/gamesblog/2014/jan/08/video-games-art-and-the-shock-of-the-new, zuletzt geprüft am 27.06.209.

Suzuki, Satoshi; Abe, Keiichi (1985): Topological structural analysis of digitized binary images by border following. In: Computer Vision, Graphics, and Image Processing 30 (1), S. 32–46. DOI: 10.1016/0734-189X(85)90016-7.

Thanh-Tung, Hoang; Tran, Truyen (2018): On catastrophic forgetting in Generative Adversarial Networks. Online verfügbar unter http://arxiv.org/pdf/1807.04015v4.

Thanh-Tung, Hoang; Tran, Truyen; Venkatesh, Svetha (2019): Improving Generalization and Stability of Generative Adversarial Networks. Online verfügbar unter http://arxiv.org/pdf/1902.03984v1.

Tian, Yuchen (2017): zi2zi: Master Chinese Calligraphy with Conditional Adversarial Networks. Online verfügbar unter https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html, zuletzt geprüft am 24.06.2019.

van den Oord, Aaron; Kalchbrenner, Nal; Kavukcuoglu, Koray (2016): Pixel Recurrent Neural Networks. Online verfügbar unter http://arxiv.org/pdf/1601.06759v3.

Vincent, James (2018): How three French students used borrowed code to put the first AI portrait in Christie's. Online verfügbar unter https://www.theverge.com/2018/10/23/18013190/ai-art-portrait-auction-christies-belamy-obvious-robbie-barrat-gans, zuletzt geprüft am 24.06.2019.

Watanabe, Shigeru (2010): Pigeons can discriminate "good" and "bad" paintings by children. In: Animal cognition 13 (1), S. 75–85. DOI: 10.1007/s10071-009-0246-8.

Yu, Simiao; Dong, Hao; Yang, Guang; Slabaugh, Greg; Dragotti, Pier Luigi; Ye, Xujiong et al. (2017): Deep De-Aliasing for Fast Compressive Sensing MRI. Online verfügbar unter http://arxiv.org/pdf/1705.07137v1.

Zaidel, Dahlia W.; Hessamian, Marjan (2010): Asymmetry and Symmetry in the Beauty of Human Faces. In: Symmetry 2 (1), S. 136–149. DOI: 10.3390/sym2010136.

Zhang, T. Y.; Suen, C. Y. (1984): A Fast Parallel Algorithm for Thinning Digital Patterns. In: Communications of the ACM 27 (3), S. 236–239. Online verfügbar unter http://agcggs680.pbworks.com/f/Zhan-Suen_algorithm.pdf, zuletzt geprüft am 27.06.2019.

Anhang

Der nachfolgende Code ist nicht für den sicheren Betrieb geeignet.

Server Python Programm

```
import numpy as np
from numpy import linalg as LA
import cv2
import sys
import math
import socket
import time


### CLI Arguments

genreType = sys.argv[1]
inputFile = sys.argv[2]
outputFile = sys.argv[3]
a = []
b = []
c = []

### Variables

colorAmount = 12
colorDifferenceMax = 500

## Define the color values of the paint here. You can change them to whatever you think repre-
sents the choosen images in 12 colors the best, or which paint you are using.

# Portrait Color Palette. Read numbers from left to right, top to bottom. Values are in BGR.
Naming is very roughly oriented at this https://en.wikipedia.org/wiki/List_of_colors:_A-F

deepChestnut_1 = [40, 75, 185]
bronze_2 = [74, 146, 216]
dutchWhite_3 = [135, 200, 231]
bone_4 = [205, 223, 222]
battleshipGrey_5 = [118, 133, 135]
ashGray_6 = [154, 171, 174]
frenchBeige_7 = [77, 113, 149]
babyPowder_8 = [250, 252, 252]
darkBrown_9 = [32, 43, 63]
blueSapphire_10 = [105, 100, 25]
coyoteBrown_11 = [61, 80, 101]
blackChocolate_12 = [10, 12, 20]

# Abstract Color Palette. Read numbers from left to right, top to bottom. Values are in BGR.
Naming is very roughly oriented at this https://en.wikipedia.org/wiki/List_of_colors:_A-F

cultured_1 = [249, 250, 251]
almond_2 = [221, 217, 210]
cadmiumRed_3 = [13, 38, 228]
citrine_4 = [28, 200, 240]
bdazzledBlue_5 = [148, 83, 11]
blueGray_6 = [140, 125, 80]
copperRed_7 = [82, 121, 209]
burlywood_8 = [152, 192, 220]
blackShadows_9 = [147, 160, 175]
burnishedBrown_10 = [103, 113, 131]
darkLiverHorses_11 = [54, 61, 77]
black_12 = [19, 18, 19]

if genreType == 'portrait':
	colorPaint = np.array([deepChestnut_1, bronze_2, dutchWhite_3, bone_4, battleship-
Grey_5, ashGray_6, frenchBeige_7, babyPowder_8, darkBrown_9, blueSapphire_10, coyoteBrown_11,
blackChocolate_12])
```



```
elif genreType == 'abstract':

	colorPaint = np.array([cultured_1, almond_2, cadmiumRed_3, citrine_4, bdazzledBlue_5,
blueGray_6, copperRed_7, burlywood_8, blackShadows_9, burnishedBrown_10, darkLiverHorses_11,
black_12])
else:
	sys.exit("Error, please specify Genre Type. Choose \"portrait\" or \"abstract\"")

### CLI info

print("Converting", genreType, "image to", np.shape(colorPaint)[0], "colors.")

### K-means Algorithm to reduce color amount to K

## Read Image file

img = cv2.imread(inputFile)

Y = img.reshape((-1, 3))
Y = np.float32(Y)

## Apply k-means

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
K = np.shape(colorPaint)[0] # nClusters, equivalent to the amount of paint colors used
ret,label,center=cv2.kmeans(Y,K,None,criteria,10,cv2.KMEANS_RANDOM_CENTERS)

center = np.uint8(center)
res = center[label.flatten()]
res2 = res.reshape((img.shape))

## Check image for closest color from the predefined palette
print(center)

for i in range(K):
	for j in range(K):
		#d = math.sqrt((((center[i][2])-(colorPaint[j][2])))**2 + (((center[i][1])-(co-
lorPaint[j][1])))**2 + (((center[i][0])-(colorPaint[j][0])))**2) # Calculate euclidean dis-
tance
		d = math.sqrt((((center[i][2])-(colorPaint[j][2]))*0.3)**2 + (((center[i]
[1])-(colorPaint[j][1]))*0.59)**2 + (((center[i][0])-(colorPaint[j][0]))*0.11)**2) # Calcu-
late euclidean distance with weights. May improve color matching. Eyes perceive some colors
better then others. Uncomment this line and comment out the above one, to apply.
		a.insert(j, int(d))
	print(a)
	if a[np.argmin(a)] < colorDifferenceMax:
		b.insert(i, np.argmin(a))
		a.clear()
		c.clear()
	else:
		b.insert(i, "null")
		a.clear()

print("Index of closest colors", b)


## Match color from k-means reduced input image with color palette
for i in range(K):
	print("center:", center[i])
	print("colorPaint", colorPaint[b][i])
	res2[np.where((res2== [center[i]]).all(axis=2))] = [colorPaint[b][i]]

### Convert to CNC path

## Brush variables
brushRadius = 10 # Brush radius in mm
canvasSize = [400, 400] # Canvas width and height in mm
resizeFactor = img.shape[0]/canvasSize[0]
brushPixel = brushRadius*resizeFactor
```

```
## Remove noise form mask with erosion followed by dilation (optional)

#kernel = np.ones((3,3),np.uint8) # Fine tune kernel for less or more denoising
#mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)

## Mask colors

kernelBrush = np.ones((int(brushPixel)-3,int(brushPixel)-3),np.uint8) # This is the erosion
kernel. Increase the weights that are subtracted from the values, to reduce missed spots
size = np.size(img)
done = False
strokeLength = 1000 # Maximum brush stroke length in mm
strokeLengthPixel = int(strokeLength*resizeFactor)

contourPath = []
contourParent = []
colorPath = []
path = []
nextPoint = ''
changeColor = True

contourExists = False
connectionOpen = True


### Configure socket here

host = "172.31.1.100"
port = 59152

## Bind with client

mySocket = socket.socket()
mySocket.bind((host,port))
mySocket.listen(1)
conn, addr = mySocket.accept()
print ("Connection from: " + str(addr))

## Define communication

def sendActions(message):
	message = message.encode()
	conn.send(message)
	print("Message sent to Client:", message)
	answer = conn.recv(33)
	answer = answer.decode()
	print("Client answered with:", answer)

### Loop through colors to find color clusters, create contours for them and erode them for
innermore contours

while True:
	for i in range(K):
		mask = cv2.inRange(res2, colorPaint[i], colorPaint[i]) # Apply mask
		colorSelected = colorPaint[i]
		while(not done):
			mask = cv2.GaussianBlur(mask,(9,9),0) # Blur mask to reduce edges and
			therefore CNC points. Must be an odd number and positive.
			ret, thresh = cv2.threshold(mask, 127, 255, 0)
			im2, contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.
			CHAIN_APPROX_SIMPLE)
			cv2.drawContours(res2, contours, -1, (0,255,0), 1)
			mask = cv2.erode(mask,kernelBrush,iterations = 1) # Shrink mask
			zeros = size - cv2.countNonZero(mask)
			if len(contours) > 0:
				sendActions('<Ext><Status>2</Status><Color>'+str(i+1)+'</Color></
				Ext>')
				currentStrokeLength = 0
			for l in range(len(contours)):
				newPath = False
```



```
				for m in range(len(contours[l])-1):
					contourExists = True
					currentX = contours[l][m][0][0]
					currentY = contours[l][m][0][1]

					if m < len(contours[l]):
						dist = np.linalg.norm(contours[l][m][0]-contours[l
						][m+1][0]) # Calculate stroke length
						currentStrokeLength = currentStrokeLength+dist

					if currentStrokeLength > strokeLength:
						sendActions('<Ext><Status>2</Status><Co
						lor>'+str(i+1)+'</Color></Ext>')
						currentStrokeLength = 0

					if currentX <= canvasSize[0] and currentY <= canvasSi
					ze[1]:
						sendActions('<Ext><Status>1</Status><Points><xyzabc
						X="'+str(currentX)+'" Y="'+str(currentY)+'" Z="0"
						A="0" B="0" C="0"/></Points></Ext>')
					else:
						sys.exit("Fatal: Points are being generated, that
						are larger than the given canvas size. Proceeding
						may cause serious damage and injury to the manipu
						lator and it's sourroundings.")
				print("newPath")
				sendActions('<Ext><Status>3</Status><Points><xyzabc X="'+str(cur
				rentX)+'" Y="'+str(currentY)+'" Z="0" A="0" B="0" C="0" /></
				Points></Ext>')
				newPath = True

		if zeros==size:
			done = True
			contourExists == False

done = False
	pcDone = True
conn.close()
cv2.imshow("res2", res2)
cv2.imshow("mask", mask)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

KRC Hauptprogramm

Unterprogramme wie water( ), drying( ) und l1c1( ) (Farbaufnahme)

werden hier nicht beschrieben, da sie individuell zu teachen sind.

Base[3] und Tool[2] müssen zuvor vermessen werden.

```
&ACCESS RVP
&REL 47
DEF ekiTransfer6old( )
;FOLD Declaration
   DECL EKI_STATUS RET
   DECL FRAME nextPoint
   DECL INT status
   DECL INT currentColor
   DECL BOOL pathStart
   $FLAG[1]=FALSE



   ;FOLD INI;%{PE}
      ;FOLD BASISTECH INI
         GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
         INTERRUPT ON 3
         BAS (#INITMOV,0 )
      ;ENDFOLD (BASISTECH INI)
      ;FOLD USER INI
         ;Make your modifications here
      ;ENDFOLD (USER INI)
      ;ENDFOLD (INI)
      ;FOLD Initialize sample data
         nextPoint = {X 200,Y 600,Z 600}
         status = 0
         currentColor = 1000
         pathStart = TRUE

      ;ENDFOLD (Initialize sample data)
      ;ENDFOLD (Declaration)


   ;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:,
5:100, 7:DEFAULT
      $BWDSTART = FALSE
      PDAT_ACT=PDEFAULT
      FDAT_ACT=FHOME
      BAS (#PTP_PARAMS,10 )
      $H_POS=XHOME
      PTP  XHOME
   ;ENDFOLD


;FOLD PTP P8 Vel=10 % PDAT4 Tool[2]:pinselC Base[3]:canvas1;%{PE}%R 8.3.48,%MKUKATPBA-
SIS,%CMOVE,%VPTP,%P 1:PTP, 2:P8, 3:, 5:10, 7:PDAT4
$BWDSTART=FALSE
PDAT_ACT=PPDAT4
FDAT_ACT=FP8
BAS(#PTP_PARAMS,10)
PTP XP8
;ENDFOLD
RET=EKI_Init("xmlTest")
RET=EKI_Open("xmlTest")
```



```
LOOP
WAIT FOR $FLAG[993] == TRUE
RET=EKI_GetInt("xmlTest","Ext/Status",status)
$FLAG[993] = FALSE

WAIT FOR $FLAG[997] == TRUE

SWITCH status

CASE POINT
RET=EKI_GetFrame("xmlTest","Ext/Points/xyzabc",nextPoint)
if pathStart == TRUE THEN

   ;FOLD PTP hoverNextPoint Vel=20 % PDAT8 Tool[2]:pinselC Base[3]:canvas1;%{PE}%R
8.3.48,%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P4, 3:, 5:100, 7:PDAT4
   $BWDSTART=FALSE
   PDAT_ACT=PPDAT2
   FDAT_ACT=FP4
   BAS(#PTP_PARAMS,20)
   PTP nextPoint : {X 0,Y 0,Z -100,A 0,B 0,C 0}
   ;ENDFOLD

   ;FOLD PTP nextPoint Vel=5 % PDAT4 Tool[2]:pinselC Base[3]:canvas1;%{PE}%R 8.3.48,%MKUKAT-
PBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P4, 3:, 5:100, 7:PDAT4
   $BWDSTART=FALSE
   PDAT_ACT=PPDAT4
   FDAT_ACT=FP4
   BAS(#PTP_PARAMS,5)
   LIN nextPoint
   ;ENDFOLD
   pathStart = FALSE

ENDIF
pathStart = FALSE
;FOLD PTP P4 Vel=100 % PDAT4 Tool[2]:pinselC Base[3]:canvas1;%{PE}%R 8.3.48,%MKUKATPBA-
SIS,%CMOVE,%VPTP,%P 1:PTP, 2:P4, 3:, 5:100, 7:PDAT4
$BWDSTART=FALSE
PDAT_ACT=PPDAT4
FDAT_ACT=FP4
BAS(#PTP_PARAMS,70)
LIN nextPoint
;ENDFOLD


CASE RENEWCOLOR
RET=EKI_GetInt("xmlTest","Ext/Color",currentColor)
pathStart = TRUE
;RET=EKI_GetFrame("xmlTest","Ext/Points/xyzabc",nextPoint)
;FOLD PTP hoverPoint Vel=20 % PDAT4 Tool[2]:pinselC Base[3]:canvas1;%{PE}%R 8.3.48,%MKUKAT-
PBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P4, 3:, 5:100, 7:PDAT4
$BWDSTART=FALSE
PDAT_ACT=PPDAT4
FDAT_ACT=FP4
BAS(#PTP_PARAMS,20)
PTP $POS_ACT : {X 0,Y 0,Z -100,A 0,B 0,C 0}
;ENDFOLD
colorPick(currentColor)

CASE NEWPATH
RET=EKI_GetFrame("xmlTest","Ext/Points/xyzabc",nextPoint)
pathStart = TRUE
;FOLD PTP hoverPoint Vel=20 % PDAT4 Tool[2]:pinselC Base[3]:canvas1;%{PE}%R 8.3.48,%MKUKAT-
PBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P4, 3:, 5:100, 7:PDAT4
$BWDSTART=FALSE
PDAT_ACT=PPDAT4
FDAT_ACT=FP4
BAS(#PTP_PARAMS,20)
LIN $POS_ACT : {X 0,Y 0,Z -100,A 0,B 0,C 0}
;ENDFOLD
```

```
ENDSWITCH

RET=EKI_ClearBuffer("xmlTest","Ext")
RET=EKI_SetBool("xmlTest","Robot/Status",true)
RET=EKI_Send("xmlTest","Robot")



ENDLOOP

RET=EKI_Clear("xmlTest")
RET=EKI_Close("xmlTest")

   ;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:HOME, 3:,
5:100, 7:DEFAULT
      $BWDSTART = FALSE
      PDAT_ACT=PDEFAULT
      FDAT_ACT=FHOME
      BAS (#PTP_PARAMS,10 )
      $H_POS=XHOME
      PTP  XHOME
   ;ENDFOLD
END

DEF colorPick(COLOR:IN)
INT COLOR
colorHome()
water()
drying()
colorHome()
SWITCH COLOR
   CASE 1
      l1c1()
   CASE 2
      l1c2()
   CASE 3
      l1c3()
   CASE 4
      l2c1()
   CASE 5
      l2c2()
   CASE 6
      l2c3()
   CASE 7
      l3c1()
   CASE 8
      l3c2()
   CASE 9
      l3c3()
   CASE 10
      l4c1()
   CASE 11
      l4c2()
   CASE 12
      l4c3()
ENDSWITCH
colorHome()

END
```


KRC Hauptprogramm – globale Variablen

```
&ACCESS RVP
&REL 47
DEFDAT  ekiTransfer6old
;FOLD EXTERNAL DECLARATIONS;%{PE}%MKUKATPBASIS,%CEXT,%VCOMMON,%P
;FOLD BASISTECH EXT;%{PE}%MKUKATPBASIS,%CEXT,%VEXT,%P
EXT  BAS (BAS_COMMAND  :IN,REAL  :IN )
;ENDFOLD (BASISTECH EXT)
;FOLD USER EXT;%{E}%MKUKATPUSER,%CEXT,%VEXT,%P
;Make your modifications here
DECL BOOL collisionocc=FALSE
DECL CONST INT POINT=1
DECL CONST INT RENEWCOLOR=2
DECL CONST INT NEWPATH=3

;ENDFOLD (USER EXT)
;ENDFOLD (EXTERNAL DECLARATIONS)
DECL BASIS_SUGG_T LAST_BASIS={POINT1[] "P10                    ",POINT2[] "P10
",CP_PARAMS[] "CPDAT0                  ",PTP_PARAMS[] "PDAT7                  ",CONT[] "
",CP_VEL[] "2.0                   ",PTP_VEL[] "20                   ",SYNC_PA-
RAMS[] "SYNCDAT                ",SPL_NAME[] "S0                     ",A_PARAMS[] "ADAT0
"}
DECL E6POS XP1={X 272.500458,Y 64.1022873,Z -139.427612,A 88.7173233,B 6.68055201,C
-6.75402117,S 2,T 11,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP1={TOOL_NO 2,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL PDAT PPDAT2={VEL 100.000,ACC 30.0000,APO_DIST 100.000,GEAR_JERK 10.0000,EXAX_IGN 0}
DECL PDAT PPDAT3={VEL 100.000,ACC 50.0000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL E6POS XP2={X 272.500458,Y 64.1022873,Z -139.427612,A 88.7173233,B 6.68055201,C
-6.75402117,S 2,T 11,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP2={TOOL_NO 2,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL E6POS XP3={X 323.515778,Y 71.6573486,Z -150.640778,A 88.7295532,B 6.69334459,C
-15.0158014,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP3={TOOL_NO 2,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL PDAT PPDAT4={VEL 100.000,ACC 50.0000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL E6POS XP4={X 323.515778,Y 71.6573486,Z -150.640778,A 88.7295532,B 6.69334459,C
-15.0158014,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP4={TOOL_NO 2,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL E6POS XP5={X 1347.12073,Y 37.7801933,Z 1062.70044,A -107.690994,B 42.5165787,C
-105.567268,S 2,T 3,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP5={TOOL_NO 2,BASE_NO 5,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL PDAT PPDAT5={VEL 100.000,ACC 100.000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL E6POS XP6={X 1347.12073,Y 37.7801933,Z 1062.70044,A -107.690994,B 42.5165787,C
-105.567268,S 2,T 3,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP6={TOOL_NO 2,BASE_NO 5,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL E6POS XP7={X 1347.12073,Y 37.7801933,Z 1062.70044,A -107.690994,B 42.5165787,C
-105.567268,S 2,T 3,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP7={TOOL_NO 2,BASE_NO 5,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL E6POS XP8={X 109.677643,Y -6.88640451,Z -63.7406158,A 46.4953041,B 9.04451,C
-5.17782068,S 2,T 3,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP8={TOOL_NO 2,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL PDAT PPDAT6={VEL 100.000,ACC 50.0000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL E6POS XP9={X 323.515778,Y 71.6573486,Z -150.640778,A 88.7295532,B 6.69334459,C
-15.0158014,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP9={TOOL_NO 2,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
DECL PDAT PPDAT7={VEL 100.000,ACC 50.0000,APO_DIST 100.000,GEAR_JERK 50.0000,EXAX_IGN 0}
DECL E6POS XP10={X 323.515778,Y 71.6573486,Z -150.640778,A 88.7295532,B 6.69334459,C
-15.0158014,S 2,T 10,E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP10={TOOL_NO 2,BASE_NO 3,IPO_FRAME #BASE,POINT2[] " ",TQ_STATE FALSE}
ENDDAT
```

KRC Ethernet Konfiguration

```
<ETHERNETKRL>
	<CONFIGURATION>
		<EXTERNAL>
			<IP>172.31.1.100</IP>
			<PORT>59152</PORT>
			<TYPE>Server</TYPE>
		</EXTERNAL>
		<INTERNAL>
			<ENVIRONMENT>Program</ENVIRONMENT>
			<BUFFERING Mode="FIFO" Limit="512"/>
			<BUFFSIZE Limit="13029"/>
			<TIMEOUT Connect="60000"/>
			<ALIVE Set_Flag="1" Ping="10"/>
			<IP>172.31.1.147</IP>
			<PORT>54600</PORT>
			<PROTOCOL>TCP</PROTOCOL>
			<MESSAGES Display="error" Logging="disabled" />
		</INTERNAL>
	</CONFIGURATION>
	<RECEIVE>
		<XML>
			<ELEMENT Type="FRAME" Tag="Ext/Points/xyzabc"/>
			<ELEMENT Type="INT" Tag="Ext/Color"/>
			<ELEMENT Type="INT" Tag="Ext/Status" Set_Flag="993"/>
			<ELEMENT Tag="Ext" Set_Flag="997"/>
		</XML>
	</RECEIVE>
	<SEND>
		<XML>
			<ELEMENT Tag="Robot/Status"/>
		</XML>
	</SEND>
</ETHERNETKRL>
```