FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Automatische Zusammenfassung von Bildern

Kompakte Anordnung von Bildgruppen in einem regulären Raster

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Michael Krakhofer
Matrikelnummer 1126297

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Johanna Schmidt MSc

Wien, 13. Oktober 2015

Michael Krakhofer	Eduard Gröller

Erklärung zur Verfassung der Arbeit

Michael Krakhofer
2100 Korneuburg, Stockerauerstraße 75/4

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. Oktober 2015

Michael Krakhofer

Danksagung

Besonderer Dank gilt meiner Betreuerin Johanna Schmidt MSc, die mir das Thema dieser Arbeit anvertraut hat und mir bei deren Umsetzung mit Antworten und Ratschlägen zur Seite stand.
Weiters möchte ich meiner Familie und meinem Arbeitgeber danken, die mich in der Entscheidung für das Studium bekräftigt und mich auf meinem Weg unterstützt haben.

Kurzfassung

Das Ziel dieser Bachelorarbeit war es, verschiedene Ansätze zu suchen und zu implementieren, mit denen folgendes Problem gelöst werden kann:
Eine Menge von Bildern ist in verschiedene Gruppen eingeteilt, wobei jedes Bild genau einer Gruppe zugehörig ist. Die Zugehörigkeit der Bilder wird vom Benutzer vorgegeben. Die Bilder sollen anschließend in einem Raster, dessen Größe vom Benutzer frei wählbar ist, so angeordnet werden, dass die Gruppen klar als Cluster erkennbar sind. Dazu muss im Ergebnisbild gewährleistet sein, dass jedes Bild mindestens einen Nachbarn der selben Gruppe bezüglich der Manhatten Distanz aufweist. Zudem sollen Lücken und Cluster, die in mehrere Teile getrennt werden, vermieden werden. Es sollen mehrere Algorithmen gefunden und untersucht werden, mit denen sich diese Aufgabenstellung automatisch lösen lässt.
Zwar beschäftigen sich viele Publikationen mit der Aufgabenstellung, automatisch Gruppen aus einer Menge von Bildern zu finden, mit denen sie sich zusammenfassen lässt, doch für den Fall, dass die Gruppen bereits vorgegeben sind, wurde noch kein Weg publiziert, mit dem sich diese möglichst kompakt in einem Ergebnisbild anordnen lassen. Diese Arbeit soll verschiedene Ansätze bieten, auch dieses Problem zu lösen.

Abstract

The aim of this bachelor thesis was to find different approaches to solve the following problem:
A set of images is divided into different groups in a way that every image belongs to exactly one group. The user defines which image belongs to which group. The images should be positioned into a grid of a user-defined size, so that they can be recognised as clusters easily. To do so, it must be guaranteed, that every image has at least one neighbour image regarding the manhatten distance which belongs to the same group. Also there must not be any gaps, or clusters which split into several parts. Multiple algorithms which are able to solve this task are to be found and analysed.
There are many approaches whose goal is to create an image summarisation out of a set of images. But these approaches place the images in a way that the groups are shaped by the algorithm itself. Yet there is no method published, which lets the user define the involved groups and their associated set of images. This thesis meets this requirements and offers different solutions for this task.

Contents

List of Figures

List of Tables

CHAPTER 1

Einleitung

Diese Bachelorarbeit beschäftigt sich mit der automatischen Anordnung von Bildern in einem regulären Raster, wobei eine zuvor definierte Gruppenzugehörigkeit der Bilder ausschlaggebend für die Anordnung ist. Einleitend werden im folgenden Kapitel das Ausgangsproblem und die Rahmenbedingungen beschrieben, die Anforderungen definiert und die verwendeten Technologien angeführt.

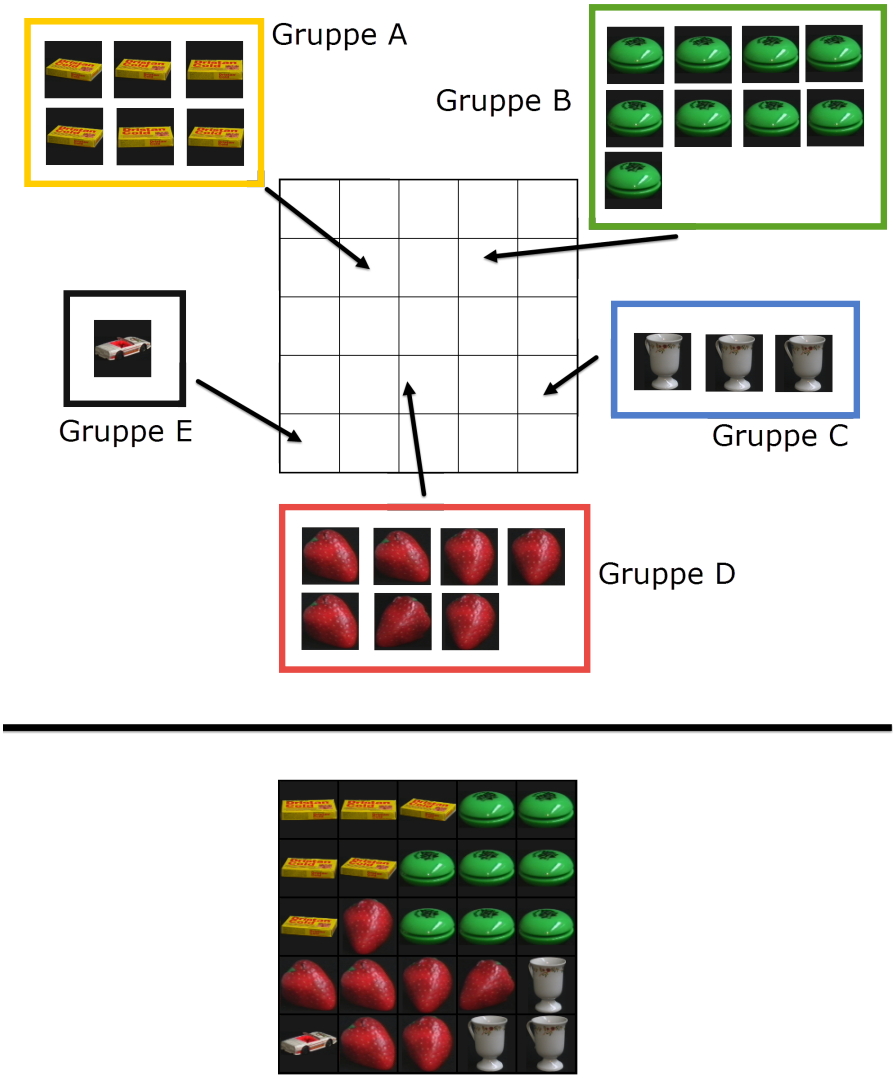
1.1 Beschreibung des Ausgangsproblems

Ausgangspunkt für die Problemstellung ist eine Menge von Bildern, die in Gruppen eingeteilt sind. Jedes Bild ist genau einer Gruppe zugeordnet, umgekehrt kann jeder Gruppe eine beliebig große Menge an Bildern zugeordnet sein. Diese Bilder sollen automatisch in einem regulären Raster auf eine Art und Weise angeordnet werden, dass sie klar als zusammengehöriger Cluster erkennbar sind. Aus dieser Anordnung entsteht ein Ergebnisbild, in dem alle Bilder untergebracht sind.
Ein Beispiel für die Problemstellung wird in Abbildung 1.1 illustriert. Im oberen Bereich der Abbildung erkennt man 5 Gruppen, denen eine unterschiedliche Anzahl an Bildern zugehörig ist, sowie einen regulären Raster mit insgesamt 25 freien Zellen. Im unteren Bereich wird eine mögliche, gültige Lösung gezeigt, in der die Einzelbilder unter Berücksichtigung ihrer Gruppenzugehörigkeit angeordnet sind.

1.2 Rahmenbedingungen und definierte Anforderungen

Zunächst musste für die Lösung dieser Aufgabe definiert werden, wann eine Lösung gültig ist und welche Anforderungen an die Anwendung, die das Problem lösen soll, gestellt werden. Im Rahmen dieser Arbeit wurden folgende Anforderungen an die Gültigkeit einer Lösung sowie an die Anwendung selbst definiert:

Figure 1.1: Skizzierte Aufgabenstellung



- Für eine gültige Zuordnung muss ein Bild mindestens einen direkten Nachbarn der selben Gruppe bezüglich der Manhatten Distanz aufweisen können, ausgenommen, die entsprechende Gruppe besitzt nur dieses eine Bild.

- Daraus folgt, dass rein diagonale Verbindungen nicht zulässig sind.

- Weiters dürfen im Ergebnisbild keine Lücken entstehen. Freie, also Zellen des Rasters ohne zugeordneten Bild, dürfen nur am Rand des Ergebnisbildes existieren.

Anders ausgedrückt müssen die freien Zellen selbst als ein zusammengehöriger Cluster erkennbar sein.

- Die Anzahl der Bilder und die Einteilung der Bilder in Gruppen dürfen frei vom Benutzer gewählt werden.
- Die Höhe und Breite des verwendeten Rasters dürfen frei vom Benutzer gewählt werden, wobei es auch erlaubt ist, dass mehr freie Zellen als Bilder, die angeordnet werden sollen, existieren.
- Der Raster muss mindestens so viele freie Zellen bieten, wie Bilder, die angeordnet werden sollen.
- Grundsätzlich kann davon ausgegangen werden, dass der Benutzer ausschließlich Bilder des selben Seitenverhältnisses auswählt. Sollte das nicht der Fall sein dürfen diese so skaliert werden, dass sie die Flächen der freien Zellen im Raster füllen.
- Die Anwendung, die das Problem löst, soll möglichst leicht bedienbar sein.
- Es soll eine Möglichkeit implementiert werden, das Ergebnisbild zu exportieren.

1.3 Verwendete Technologien

Die Technologien, die verwendet werden um die Aufgabenstellung dieser Arbeit zu lösen, durften frei gewählt werden. Die Verwendung beliebiger Frameworks und Libraries war erlaubt. Im Folgenden werden die verwendeten Technologien kurz beschrieben und deren Einsatz begründet.

1.3.1 .NET Framework

Als Entwicklungsumgebung wurde Visual Studio 2012 gewählt, mit der die Anwendung als .NET Applikation mittels Programmiersprache C# implementiert wurde. Das .NET Framework bietet eine Laufzeitumgebung, die es erlaubt, verwaltete Applikationen auszuführen. Es besteht aus der *Common Language Runtime* welche unter anderem Aufgaben zur Speicherverwaltung übernimmt. Zudem steht dem Entwickler eine Library von Klassen zur Verfügung. Ein großer Vorteil des Frameworks besteht darin, dass .NET Anwendungen überall dort ausgeführt werden können, wo die Laufzeitumgebung des Frameworks installiert ist. Die Wahl des .NET Frameworks wird damit begründet, dass sich, anders als bei älteren Frameworks wie MFC (*Microsoft Foundation Classes*) ein Garbagecollector um nicht mehr benötigte Objekte kümmert und so Memory Leaks im Vorhinein ausgeschlossen werden können. Weiters stellt Visual Studio bei Verwendung des .NET Frameworks einen komfortablen GUI-Editor zur Verfügung, mit dem schnell Userinterfaces erstellt werden können. Letztendlich war auch die Tatsache, dass .NET von zahlreichen externen Frameworks und Libraries unterstützt wird, ausschlaggebend für die Wahl.

1.3.2 OpenCV

Open Computer Vision (OpenCV) ist eine Programmbibliothek, welche Klassen und Methoden für Aufgaben der Bildverarbeitung und Visual Computing bietet. Der Grund für die Verwendung von OpenCV besteht darin, dass das Framework unter anderem einen Canvas bietet, auf dem schnell und unkompliziert gezeichnet werden kann.

1.3.3 EMGU

Um OpenCV im Rahmen des .NET Frameworks nutzen zu können wird ein Wrapper benötigt, da OpenCV native implementiert ist. EMGU ist ein plattformunabhängiger .NET Wrapper, der es .NET Applikationen erlaubt, auf Methoden von OpenCV zuzugreifen.

1.4 Ziele der Arbeit

Neben der Implementierung der Anwendung, welche in der Lage ist, die eingangs beschriebene Problemstellung zu lösen, wird ihm Rahmen dieser Bachelorarbeit verstärkt auf die entwickelten Algorithmen eingegangen. Ziel ist es, die verwendeten Methoden hinsichtlich ihrer qualitativen Unterschiede zu untersuchen. Dazu wird die Vorgangsweise jeder Methode sowohl textuell als auch mit Hilfe bildlicher Darstellungen und Beispielen erläutert. Im Rahmen dieser Arbeit wurden drei Ansätze gefunden, welche sich grundlegend anhand ihrer Vorgehensweise unterscheiden, was auch in den Ergebnissen sichtbar ist.

CHAPTER 2

Verwandte Arbeiten

Grundsätzlich wurde in der Forschung bereits Aufwand betrieben, um Informationen vieler Bilder möglichst gut in einem Gesamtbild zusammenfassen zu können. Allerdings gehen bestehende Lösungen von einem Distanzmaß anhand der Ähnlichkeit der Bilder untereinander aus. Anhand dieses Distanzmaßes werden die Bilder angeordnet, wodurch implizit Gruppen definiert werden, zu denen die jeweiligen Bilder gehören. Aufgrund dieser Tatsache wird im Rahmen dieser Arbeit nach einer Lösung gesucht, wie man eine Anordnung ausgehend von bereits definierten Gruppen vornehmen könnte.

Die konkrete Aufgabenstellung lässt sich weiter verallgemeinern. Ein ähnliches Problem ist das Packen von Rechtecken in einem größeren Rechteck, wobei man allerdings als Ausgangsbasis meist eine Menge von Rechtecken unterschiedlicher Größe hat und versucht, diese möglichst lückenlos anzuordnen. Weiters wurde noch Forschung mit Polyominos betrieben. Dabei untersucht man, wie man Gruppen von gleich großen Quadraten anordnen kann, sodass diese Teile möglichst gut in einen größeren, rechteckigen Bereich passen.

Zudem gibt es noch das Phänomen der Space Filling Curves, welche zumindest auf abstrakter Ebene Analogien zur Aufgabenstellung dieser Arbeit aufweisen. Space Filling Curves sind Kurven in der Mathematik, die man durch Rekursion theoretisch immer weiter verdichten kann, sodass man letztendlich in der Lage ist, eine zweidimensionale Fläche mit einer Linie füllen zu können. Letztendlich wurde eine Lösung dieser Arbeit durch eine dieser Space Filling Curves inspiriert.

Zuletzt werden in diesem Kapitel noch verschiedene Anwendungen zur Erstellung von Bildcollagen, sowie mehrere Arbeiten, die sich mit der Visualisierung von Sets oder mit der Anordnung von Bildern befassen, vorgestellt.

2.1 ImageHive Project

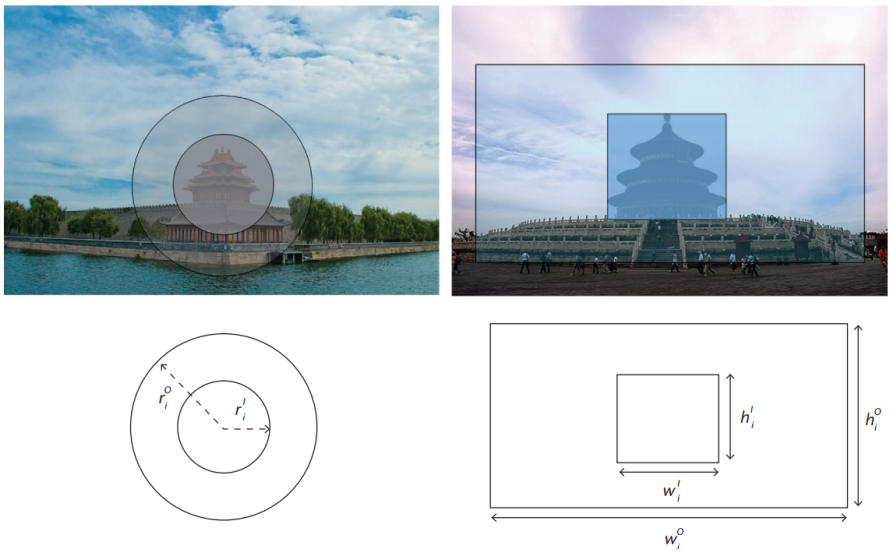
ImageHive [LTX11] ist eine Anwendung, die dazu in der Lage ist, eine Menge von Bildern zu analysieren und sie anhand ihrer Ähnlichkeit zueinander in einem großen Bild

anordnet. Dabei ist es zulässig, dass sich unwichtige Bereiche der Bilder überlappen, solange die wichtigen Informationen sichtbar bleiben. Gleichzeitig erlaubt die Anwendung dem Nutzer, in Echtzeit die Anordnung manuell zu beeinflussen, um Informationen, die kommuniziert werden sollen, besser abbilden zu können.

Figure 2.1: Ein Ergebnisbild von *ImageHive* [LTX11]

Bevor die Anordnung der Bilder stattfinden kann, muss *ImageHive* feststellen, welche Teile der Bilder wichtig und welche weniger wichtig sind, und welche Teile überhaupt keine relevanten Informationen aufweisen. Für diese Einteilung in Regionen gibt es Ansätze, durch die radiale oder rechteckige Regionen entstehen, wie sie in Abbildung 2.2 zu sehen sind.

Figure 2.2: *ImageHive*: Unterteilung der Bilder in Regionen [LTX11]

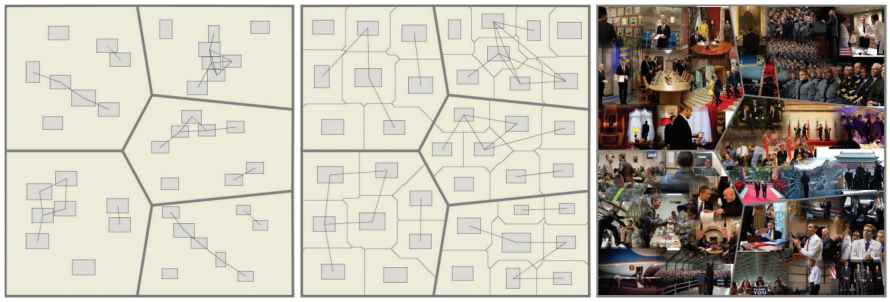


Die globale Platzierung der Bilder ergibt sich durch ein Ähnlichkeitsmaß der Bilder zueinander. Anhand eines *Graph Layout Models* werden die Bilder zunächst grob ange-

ordnet, wobei jeder Knotenpunkt ein Bild und jede Kante des Graphen die Ähnlichkeit zwischen zwei Bildern repräsentiert.

Anhand der Anordnung des *Graph Layout Models* lassen sich nun Cluster bilden. Nach der Bildung der Cluster lassen sich nun die Bilder innerhalb eines Clusters mittels *Online Voronoi Tesselation* so lokal platzieren, dass sich ihre Regionen, mit hohem Informationsgehalt nicht überlappen, und gleichzeitig möglichst viel Platz im zweidimensionalen Raum in Anspruch genommen wird.

Figure 2.3: *ImageHive*: Graph Layout Model [LTX11]



Der größte Unterschied zwischen der Aufgabe, welche *ImageHive* löst, und der Aufgabenstellung dieser Arbeit besteht darin, dass wir bereits von einer Menge an Gruppen mit entsprechend zugeordneten Bildern ausgehen. *ImageHive* bildet die Gruppen selbst anhand der globalen Position im Graph Layout Model. Zudem ist es beim *ImageHive* Project zulässig, dass sich unwichtige Bereiche der Bilder überlappen. Die Einteilung der Bilder in wichtige und weniger wichtige Regionen entfällt in unserer Aufgabenstellung. Wir wollen alle Informationen der Bilder zusammenfassend visualisieren.

2.2 IsoMatch

IsoMatch [OFF15] ist ebenfalls eine Anwendung, welche in der Lage ist, eine Menge von Bildern in einem größeren Bild anzuordnen. Im Gegensatz zu *ImageHive* überlappen sich hier die Bilder nicht. Zudem geschieht die Anordnung auch nicht mittels *Voronoi Tesselation*, sondern anhand eines regulären Rasters, der befüllt werden soll.

Im ersten Schritt wird die Dimensionalität der Daten - also der Eingabebilder - mittels *IsoMap* reduziert. Dabei muss festgelegt werden, welches Ähnlichkeitsmaß für die Anordnung ausschlaggebend sein soll. Beispielsweise kann man sich dabei an den Mittelwert der RGB-Werte orientieren. Die Positionen, an denen die Bilder zu liegen kommen können, werden vorgegeben, wobei jeder Position für einen bestimmten Wert des Distanzmaßes steht. Im Anschluss werden die tatsächlichen Bilder anhand ihrer Merkmale in dieser Struktur platziert. Danach wird ein Graph gebildet, der Kanten zwischen den Ausgabebildern der *IsoMap* und den Merkmalen der Bilder generiert, wodurch die Zuordnung der Bilder zu den entsprechenden Positionen entsteht. Erweiterungen erlauben beispielsweise auch die Anordnung der Bilder auf einer Kugel.

Figure 2.4: *IsoMatch*: Nachdem der Graph gebildet wurde (links) wird die Zuordnung in der finalen Struktur ermittelt (rechts) [OFF15]

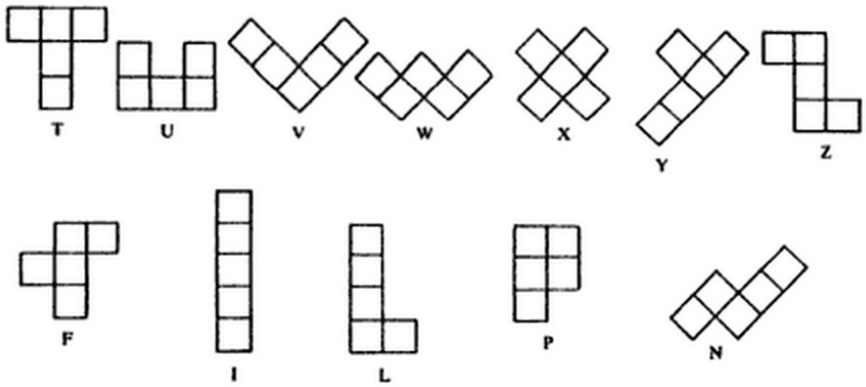
2.3 Heuristiken für das *Rectangle Packing Problem*

Das *Rectangle Packing Problem* [SS95] beschäftig sich mit der Frage, wie mehrere Rechtecke, die nicht notwendigerweise die selbe Größe aufweisen müssen, in einem größeren Rechteck angeordnet werden können, sodass der Platz optimal ausgenutzt wird. Aufgrund der Zugehörigkeit dieses Problems zur Klasse der NP-schweren Probleme existieren bis heute neben der exakten Lösung auch viele heuristische Ansätze.

2.4 Polyominos

Polyominos [Gol94] sind Strukturen, die sich durch Anordnung gleich großer Quadrate ergeben. Je nachdem, aus wie vielen Quadraten ein Polyomino zusammengesetzt wird, spezifiziert man den Namen des Polyominos. Beispielsweise spricht man von Polyominos, die aus vier Quadraten bestehen, wie es unter anderem beim bekannten Spiel Tetris der Fall ist, von Tetrominos. Auch Dominos sind namensgebend für das gleichnamige, bekannte Brettspiel. Alle möglichen Kombinationen für Pentominos werden im Abbildung 2.5 gezeigt.

Figure 2.5: Alle 12 möglichen Pentominos [Gol94]



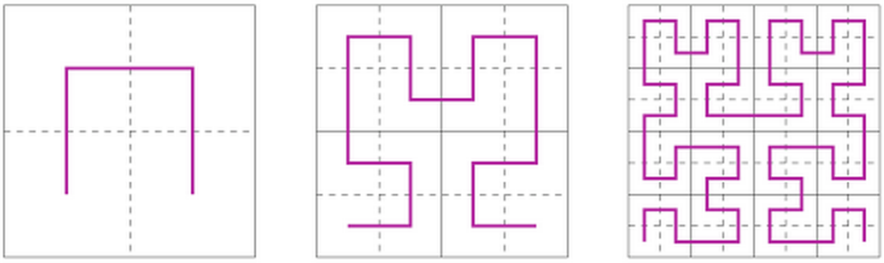
Neben der Anzahl der Formen, die sich für eine Klasse von Polyominos ergeben können, wurde auch bereits erforscht, welche Strukturen besonders gut für die wiederholte Nutzung geeignet sind, um beispielsweise rechteckige Strukturen herausbilden zu können [Kla69]. In weiterer Folge wurde auch untersucht, wie man Polyominos nutzen könnte, um vorgegebene Rechtecke füllen zu können. Allgemein hat man sich auch mit der Frage beschäftigt, unter welchen Bedingungen es möglich ist, Strukturen, die aus Zellen bestehen, in Polyominos unterteilen zu können [CL90].

Letztendlich können die Erkenntnisse aus diesem Forschungsgebiet aber nicht auf unsere Aufgabenstellung angewandt werden, da es dem Nutzer überlassen ist, welche Ausmaße der Raster annimmt und wieviele Cluster und wieviele zugehörige Bilder existieren. Im Allgemeinen wird beim Packen von Polyominos davon ausgegangen, dass man nur Polyominos einer Klasse, also immer gleich viele Quadrate pro Polyomino, zur Verfügung hat.

2.5 Space Filling Curves

Space Filling Curves [Bad13] sind Linienzüge in der Mathematik, die durch rekursives Anwenden von Ersetzungsregeln entstehen. Beispielsweise ist es möglich, einen Linienzug zu konstruieren, der zwei Punkte A und B miteinander verbindet aber bei unendlicher Anwendung der Ersetzungsregeln unendlich lang wird. Aus dieser Überlegung heraus hat man herausgefunden, dass es theoretisch auch möglich ist, eine Kurve zu konstruieren, die so dicht angeordnet ist, dass sie eine Fläche ausfüllt. Beispiele dafür sind die *Hilbert-Kurve* (siehe Abbildung 2.6) oder die *Peano-Kurve*.

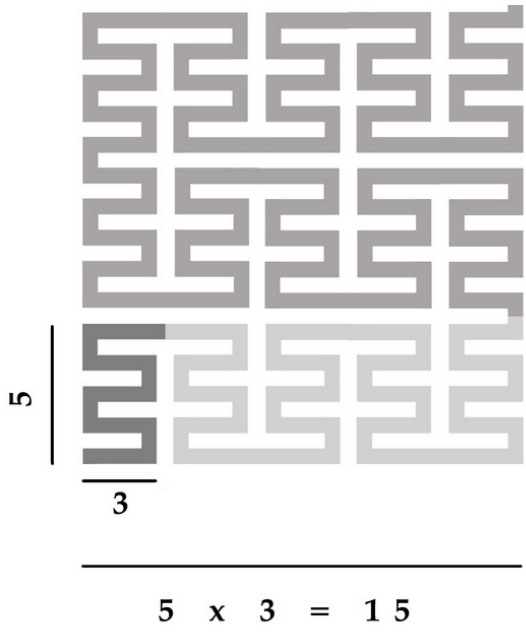
Figure 2.6: Die ersten 3 Iterationen zur Erzeugung der *Hilbert Kurve* [Bad13]



Im Wesentlichen ist bei diesen Kurven interessant, dass sie meistens dafür gedacht sind, Quadrate oder Rechtecke zu füllen. Je nachdem, wie dicht die Kurve bereits konstruiert wurde, durchläuft sie einen regulären Raster. Ein Nachteil der meisten Kurven besteht darin, dass die Seitenverhältnisse der Rechtecke, die gefüllt werden können, nicht beliebig sein können, was allerdings in unserem Rahmen gefordert ist.

Eine Space Filling Curve, die dazu in der Lage ist, beliebige Rechtecke zu füllen, ist das *Murray Polygon* [Col], eine Verallgemeinerung der *Peano Kurve*. Diese Kurve wurde als Vorbild für einen Lösungsansatz im Rahmen dieser Arbeit herangezogen.

Figure 2.7: Eine Darstellung des *Murray Polygons* [Col]



2.6 Bildcollagen

2.6.1 AutoCollage

AutoCollage [CRL06] ist eine Anwendung, mit der es möglich ist, aus einer Auwahl von Bildern eine Collage zu erstellen, welche die Bilder zusammenfasst. Dabei werden Regionen der Bilder, die weniger von Interesse sind, genutzt, um durch Blending fließende Übergänge zu erzeugen, wodurch das Ergebnis optisch ansprechender wirken soll. Um wichtige von unwichtgen Regionen unterscheiden zu können wird der Begriff *Collage Energy* eingeführt.

2.6.2 Picture Collage

Picture Collage [TLS09] befasst sich mit dem Problem, wie man eine Collage aus einer Gruppe von Bildern erstellen kann, wobei der Gehalt an sichtbarer Information maximiert werden soll. Unwichtige Bereiche dürfen sich überlappen. Dabei erzeugt *Picture Collage* zunächst eine 1-D Collage die im Anschluss mittels der *Monte-Carlo Markov-Ketten* Methode optimiert wird. Der Nutzer kann das Ergebnis mittels Drag-and-Drop selbst beeinflussen.

Figure 2.8: Ein Ergebnisbild von *AutoCollage* [CRL06]

2.6.3 Puzzle-like Collage

Üblicherweise gehen Anwendungen, die automatisch Collagen aus einer Auswahl von Bildern erstellen, von einer Menge rechteckiger Bilder aus. Diese werden im Anschluss häufig überlappend und möglicherweise mit fließenden Übergängen im Ergebnisbild platziert. Der Nachteil hinter dieser Vorgehensweise besteht darin, dass uninteressante Informationen im Ergebnisbild nicht vermieden werden. Die Anwendung *Puzzle-like Collage* [SGZM10] nutzt einen Ansatz, der die wichtigen Regionen der Bilder 'ausschneidet' und diese Im Ergebnisbild positioniert, so wie es beispielsweise Künstler mit Schere und Papier tun. Dadurch werden unwichtige Regionen der Bilder vermieden.

2.6.4 Image collection summarization via dictionary learning for sparse representation

Yang, Shen, Peng und Fan formulieren in ihrer Arbeit [CYF13] das Problem der *Image Summarisation* in ein *Dictionary Learing Problem* um. Oft sind Zusammenfassungen von Einzelbildern notwendig, um einen schnellen Überblick über verschiedene Kategorien vermitteln zu können. Deshalb befasst sich ihre Arbeit damit, Bilder zu ermitteln, die inhaltlich möglichst die selben Informationen beinhalten, wie einzelne Ausgangsbilder. Ein Beispiel für diese Herangehensweise ist in Abbildung 2.11 zu sehen.

Figure 2.9: Ein Ergebnisbild von *Picture Collage* [TLS09]

Figure 2.10: Ein Ergebnisbild von *Puzzle-like Collage* [SGZM10]

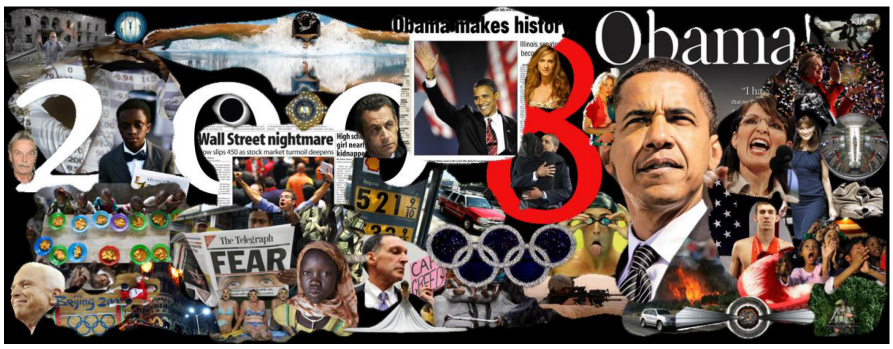


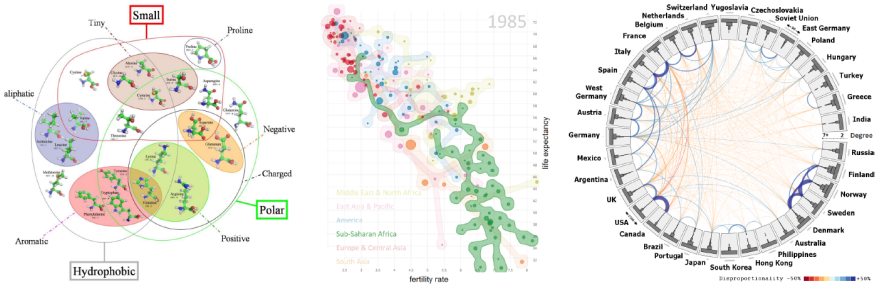
Figure 2.11: Das Bild rechts enthält die relevanten Informationen aus den beiden vorhergehenden Bildern [CYF13]

2.7 Set-Visualisierungen

2.7.1 Visualizing Sets and Set-typed Data: State-of-the-Art and Future Challenges

Diese Arbeit [BAR14] setzt sich mit aktuellen Methoden auseinander, die angewandt werden können, um aus einer Sammlung von Daten Gruppen und Beziehungen ableiten und visualisieren zu können. Außerdem werden die verschiedenen Techniken miteinander verglichen und Ratschläge gegeben, welche Techniken für welche Arten von Problemstellungen gewählt werden sollten.

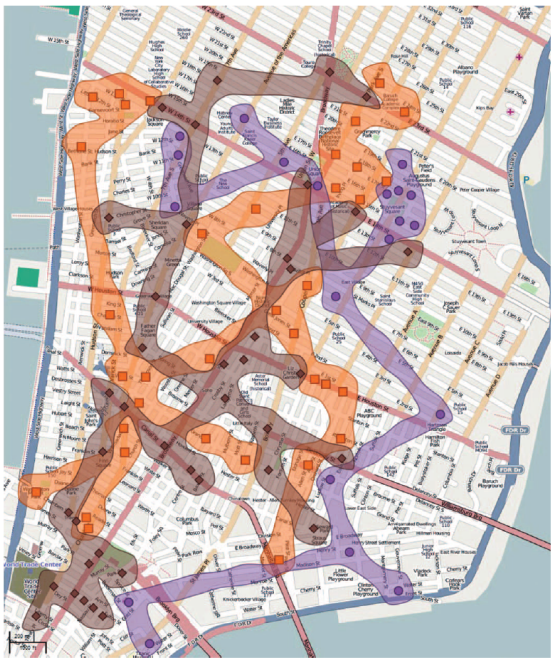
Figure 2.12: 3 verschiedene Möglichkeiten um Sets zu visualisieren [BAR14]



2.7.2 *Bubble Sets*: Revealing Set Relations with Isocontours over Existing Visualizations

Collins, Penn und Carpendale stellen in ihrer Arbeit [CCC09] fest, dass es oft schwierig ist, mehrere Beziehungen zwischen Daten in einer Visualisierung darstellen zu können. Deswegen beschreiben sie, wie man mit *Bubble Sets* bestehende Visualisierungen erweitern kann, um weitere Beziehungen hervorheben zu können, ohne das primäre Layout der Visualisierung zu stören.

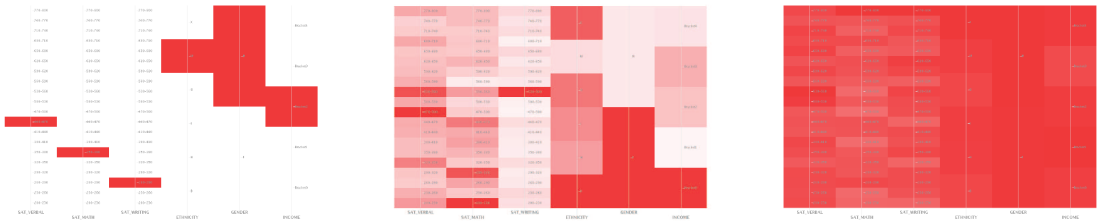
Figure 2.13: *Bubble Sets*: Die Sets werden über dem primären Layout der geografischen Lage visualsiert [CCC09]



2.7.3 Visualization of diversity in large multivariate data sets

Das Ziel von Visualisierungen besteht meistens darin, dem Betrachter einen Sachverhalt oder Zusammenhänge optisch zu veranschaulichen. Pham, Hess, Ju, Zhang und Metoyer stellen in ihrer Arbeit [TPM10] eine Methode vor, mit der es möglich ist, die Diversität großer Datensätze mit mehreren Merkmalen darzustellen. Dafür präsentieren sie die Visualisierungsmethode der *Diversity Maps*, mit der Nutzer die Diversität der Daten feststellen können.

Figure 2.14: Visualisierung der Diversität großer, multivariater Datensätze [TPM10]

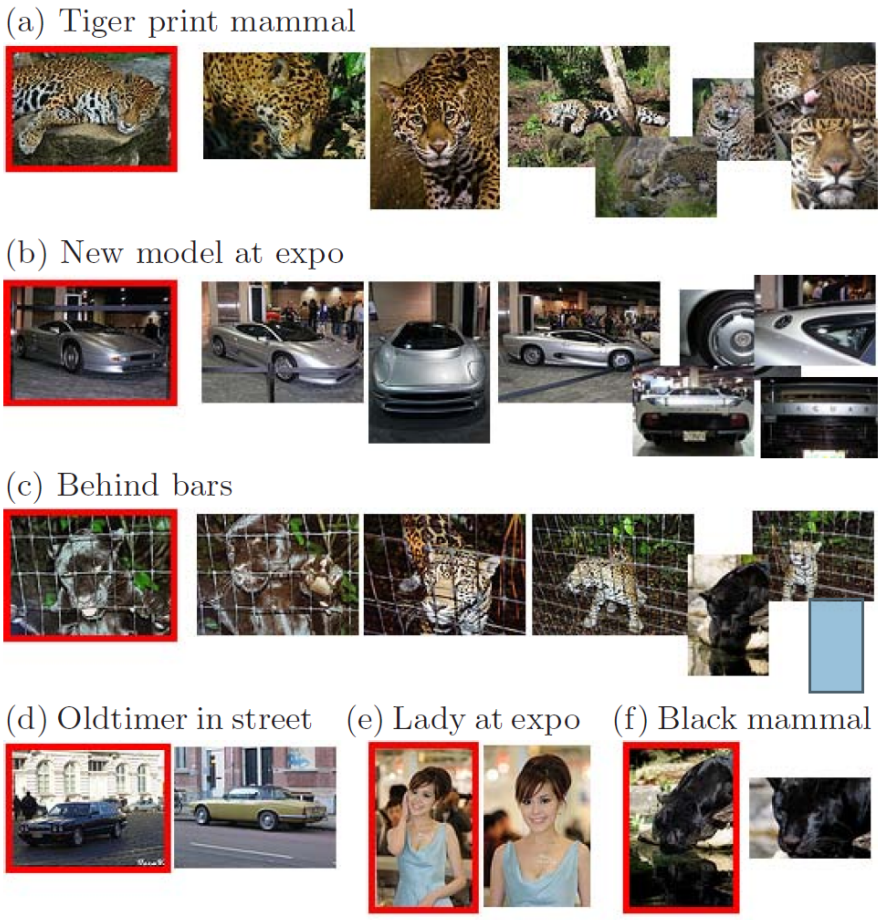


(a) Very low diversity (b) Medium diversity (c) Very high diversity

2.7.4 Visual Diversification of Image Search Results

Visual Diversification of Image Search Results [SvZG09] beschäftigt sich mit der Problemstellung, wie man eine Menge von Bildern, welche von einer Suchanfrage zurückgegeben werden, auf eine Art und Weise in Cluster unterteilen kann, dass möglichst unterschiedliche Cluster enstehen. In der Arbeit wird als Beispiel der Suchbegriff 'Jaguar' gegeben. Nutzer erhalten neben Bildern von Raubkatzen beispielsweise auch Bilder eines neuen Automodells oder Bilder anderer Kategorien, wie in Abbildung 2.15 zu sehen ist. Ziel ist es, aus den Ergebnisbildern möglichst jene auszuwählen, die repräsentativ für die verschiedenen Cluster sind.

Figure 2.15: Ergebnisse für die Suche nach 'Jaguar'. Rot umrahmte Bilder sind repräsentativ für die jeweiligen Cluster. [SvZG09]



2.8 Anordnung von Bildern

2.8.1 Kernelized Sorting

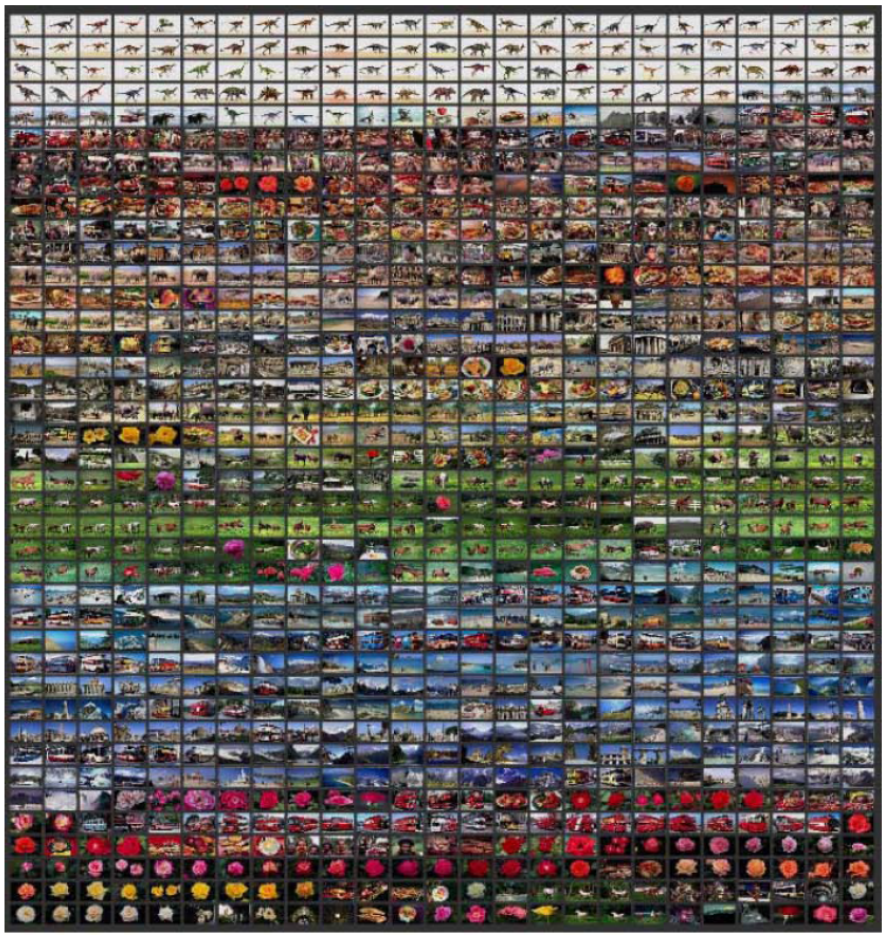
Kernelized Sorting [NQT10] ist ein Ansatz, mit eine paarweise Zuordnung verschiedener Objekte möglich ist, ohne ein globales Distanzmaß zu definieren. Um dieses Ziel zu erreichen nutzt dieser Ansatz das *Hilbert Schmidt Unabhängigkeits Kriterium*.

Figure 2.16: Bilder die anhand *Kernelized Sorting* in einem 2D-Raster platziert wurden [NQT10]

2.8.2 Similarity-Based Visualization for Image Browsing Revisited

Schoeffmann und Ahlström untersuchen in ihrer Arbeit [SA11] ob Nutzer bestimmte Bilder in einem Raster-Layout schneller finden können wenn diese Anhand ihrer Farbe sortiert sind. Dazu entwickelten sie einen Algorithmus, der Bilder innerhalb eines Rasters nach Farbe sortiert und führten im Anschluss eine Studie mithilfe verschiedener Testpersonen durch. Abbildung 2.17 zeigt eine solche sortierte Menge von Bildern. Schoeffmann und Ahlström waren in der Lage zu zeigen, dass Testpersonen ihre Suche schneller in sortierten Darstellungen durchführen konnten.

Figure 2.17: Eine Auswahl an Bildern, die sortiert wurden [SA11]



2.8.3 Interactive by-example design of artistic packing layouts

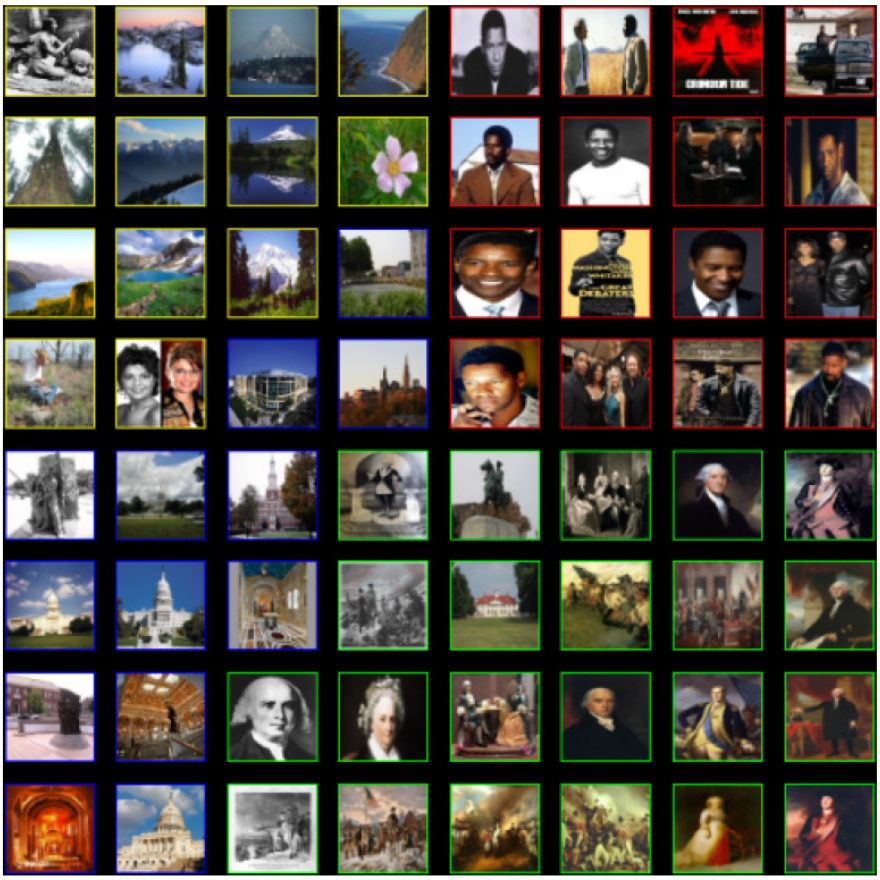
Reinert, Ritschel und Seidel stellen in dieser Arbeit [BRS13] ein System vor, mit dem es Nutzern ermöglicht wird, eine Menge von Objekten anhand verschiedener Merkmale räumlich zu sortieren. Dabei platziert der Nutzer ausgewählte Elemente die beispielsweise aufgrund ihrer Farbe oder Größe Extremwerte aufweisen. Diese können anschließend von den Nutzern räumlich angeordnet werden. Alle anderen Objekte positionieren sich anhand ihrer Merkmale anschließend selbst, sodass die Objekte räumlich anhand iherer Merkmale sortiert sind. Gleichzeitig wird durch eine *Voronoi Tesselation* ein einheitlicher Abstand zwischen den Objekten eingehalten. Abbildung 2.18 zeigt, wie die Referenzobjekte mit Stecknadel-Symbolen markiert sind.

Figure 2.18: Eine unsortierte Auswahl an Objekten (links) und zwei sortierte Ansichten. Die Stecknadeln geben die Referenzobjekte an, anhand denen die Sortierung durchgeführt wird. [BRS13]

2.8.4 Organizing Visual Data in Structured Layout by Maximizing Similarity-Proximity Correlation

Diese Arbeit [GSE13] befasst sich mit der Anordnung von Daten innerhalb eines Layouts, sodass die Nähe der Daten zueinander ihre Ähnlichkeit zueinander ausdrückt. Neben der 2-dimensionalen Anordnung wäre auch eine Erweiterung auf 3-dimensionale Daten denkbar, allerdings müsste man in diesem Fall das Problem der Überdeckung lösen. Um die Ähnlichkeit der Daten zu finden wurde der Begriff der *Max Correlation Map* definiert. Abbildung 2.19 zeigt ein Ergebnisbild, welches Bilder des Suchbegriffes 'Washington' bilden.

Figure 2.19: Verschiedene Bildkategorien des Suchbegriffes 'Washington', die anhand ihrer Ähnlichkeit zueinander in einem Layout platziert wurden. [GSE13]



CHAPTER 3

Methodologie

Im folgenden Kapitel wird zunächst näher auf die Gestaltung der Benutzeroberfläche sowie auf die Bedienung der Anwendung eingegangen. Danach werden drei Ansätze, die im Rahmen dieser Arbeit ausgearbeitet wurden, beschrieben. Dabei werden die angewandten Algorithmen Schritt für Schritt erläutert und mit Abbildungen illustriert. Zudem werden die Ergebnisse der Ansätze hinsichtlich ihrer qualitativen Merkmale untersucht.

3.1 Benutzeroberfläche

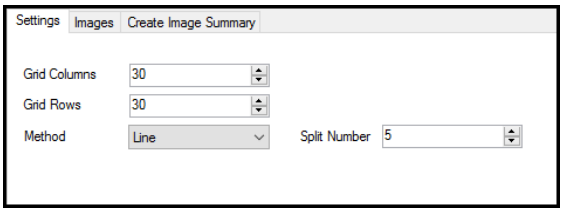
Ein Ziel dieser Arbeit bestand darin, dem Nutzer eine Benutzeroberfläche zur Verfügung zu stellen, die leicht verwendbar ist und lediglich grundlegende Kenntnisse bezüglich des Zwecks der Anwendung voraussetzt. Technisch wurde die Benutzeroberfläche mithilfe des Editors für .NET Anwendungen realisiert, der von Visual Studio zur Verfügung gestellt wird. Sämtliche Benutzersteuerelemente, die verwendet wurden, sind Bestandteil des .NET Frameworks. Grundsätzlich teilt sich die Benutzeroberfläche in drei Bereiche auf, zwischen denen mittels Karteireiter gewechselt werden kann. Für diese Designentscheidung gibt es mehrere Gründe. Einerseits ist somit gewährleistet, dass eine möglichst große Fläche für die Vorschau des Endergebnisses verfügbar ist. Die Auswahl der Bilder und die Einstellungen werden ohnehin nur vor dem Start der Lösungsansätze benötigt und sind daher in der Ergebnisansicht nicht relevant. Ein weiterer Vorteil besteht darin, dass der Raster beispielsweise nach Änderungen der Rastergröße erst dann neu gezeichnet wird, wenn er benötigt wird, nämlich genau dann, wenn der Nutzer zur Ergebnisansicht wechselt.

3.1.1 Einstellungen

Der Nutzer kann die Anzahl der Zeilen und Spalten des Rasters mittels entsprechenden Comboboxen wählen. Zudem hat er/sie die Wahl, sich für einen Lösungsansatz, der verwendet werden soll, zu entscheiden. Für den Fall, dass die Methode des Linien

Algorithmus (siehe Kapitel 3.4) gewählt ist, erscheint eine weitere Combobox, mit der die Anzahl der vertikalen Schnitte des Rasters beeinflusst werden kann. Ansonsten wird diese Einstellungsmöglichkeit verborgen.

Figure 3.1: Ansicht der Einstellungen



3.1.2 Auswahl der Gruppen und Bilder

Mithilfe dieser Ansicht hat der Benutzer die Möglichkeit, die Gruppen und die dazugehörigen Bilder festzulegen. Diese Ansicht unterteilt sich in zwei Bereiche. Links werden die Gruppen definiert, während rechts die zugehörigen Bilder hinzugefügt werden können. Die Funktionen der Buttons sind wie folgt definiert:

Create Cluster
Ein Klick auf diesen Button führt dazu, dass eine neue Gruppe angelegt und in der Listbox links angezeigt wird. Der Name für die Gruppe wird automatisch vergeben und setzt sich aus dem Wort 'Cluster' sowie einer angehängten Nummer zusammen, die automatisch inkrementiert wird. Der Grund für diese Vorgehensweise besteht darin, dass damit gewährleistet ist, dass die Gruppen eindeutige Namen aufweisen, mit denen sie identifiziert werden können.

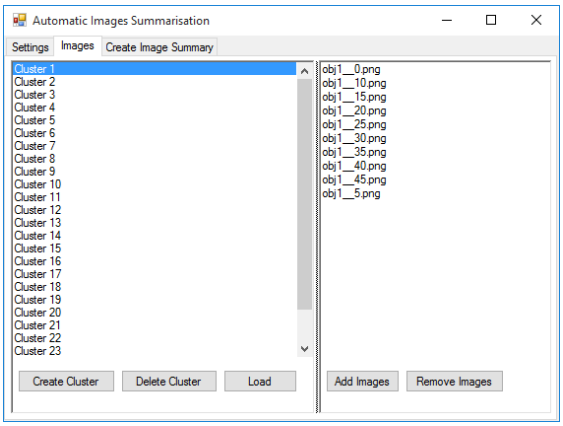
Delete Cluster
Wird diese Schaltfläche betätigt, so wird jene Gruppe entfernt, die aktuell in der linken Listbox ausgewählt ist. Sollten sich in dieser Gruppe bereits Bilder befinden, so werden diese ebenfalls entfernt.

Add Images
Wenn diese Schaltfläche betätigt wird, so öffnet sich ein Dialog, in dem der Nutzer beliebig viele Bilder auswählen kann. Bestätigt der Nutzer die Eingabe, so werden diese Bilder derjenigen Gruppe zugeordnet, die aktuell ausgewählt ist. Für den Fall, dass noch keine Gruppe existiert, geschieht nach der Betätigung des Buttons nichts.

Remove Images
Wird diese Schaltfläche betätigt, so werden sämtliche ausgewählten Bilder aus der entsprechenden Gruppe entfernt.

Load
Diese Funktion wurde implementiert, damit der Nutzer beim Aufkommen großer Datenmengen nicht gezwungen ist, jede Gruppe einzeln zu generieren und die Bilder für jede Gruppe entsprechend hinzuzufügen. Wird diese Schaltfläche betätigt, so öffnet sich ein Dialog für die Auswahl eines Ordners. Wird ein Ordner ausgewählt, so wird für jeden Unterordner, der sich darin befindet, automatisch eine Gruppe angelegt. Weiters wird jeder dieser Unterordner nach Bildern durchsucht. Jedes gefundene Bild wird sofort zur entsprechenden Gruppe hinzugefügt.

Figure 3.2: Ansicht für die Wahl der Gruppen und Bilder



3.1.3 Rasteransicht

In der Rasteransicht wird das OpenCV Framework eingesetzt, um den Raster zu zeichnen. Die Höhe und Breite der einzelnen Zellen des Rasters werden aus den Einstellungen zur Anzahl der Spalten und Zeilen und der verfügbaren Höhe und Breite der Zeichenfläche errechnet. Um die Fehlersuche während der Entwicklung zu erleichtern wurde zwischen den Zellen ein Abstand eingerechnet, wodurch die schwarzen Gitterlinien des Rasters entstehen. Diese fallen allerdings beim Export des Ergebnisbildes weg. Wird die Größe der Zeichenfläche verändert, so wird der Raster neu gezeichnet. Auf der rechten Seite der Ansicht befinden sich drei Buttons, sowie ein Textfeld, das dazu genutzt werden kann, um dem Nutzer zusätzliche Informationen zur Verfügung zu stellen, beispielsweise Informationen zur Anzahl der verwendeten Gruppen und die Anzahl ihrer zugeordneten Bilder. Die Funktionen der Buttons sind wie folgt definiert:

Create Image Summary
Sobald diese Schaltfläche betätigt wird, startet der vom Nutzer gewählte Algorithmus mit der Platzierung der Bilder im Raster. Um dem Nutzer Feedback über den Fortschritt der Anwendung zu geben, wird der Raster nach jedem relevanten Zwischenschritt neu gezeichnet.

Export Image
Nach Betätigung dieser Schaltfläche öffnet sich ein Dialog, in dem der Nutzer einen Speicherort für das Ergebnisbild angeben muss. Weiters hat man die Möglichkeit, die Höhe und Breite jedes einzelnen Bildes des Ergebnisbildes in Pixel anzugeben. Die Gesamtmaße des Ergebnisbildes werden unter den entsprechenden Auswahlfeldern angezeigt. Nachdem die Schaltfläche 'Export' betätigt wird, wird das Ergebnisbild mit den gewählten Maßen generiert und am angegebenen Speicherort abgelegt.

Debug
Die Schaltfläche Debug startet einen Testlauf, der besonders Hilfreich für die Tests und Fehlersuche der verschiedenen Ansätze war. Dabei wird, ausgehend von der Größe des Rasters, eine zufällige Anzahl an Gruppen mit jeweils zufällig vielen zugehörigen Elementen erzeugt, sodass der aktuelle Raster vollständig gefüllt werden kann.

Figure 3.3: Der Export Dialog

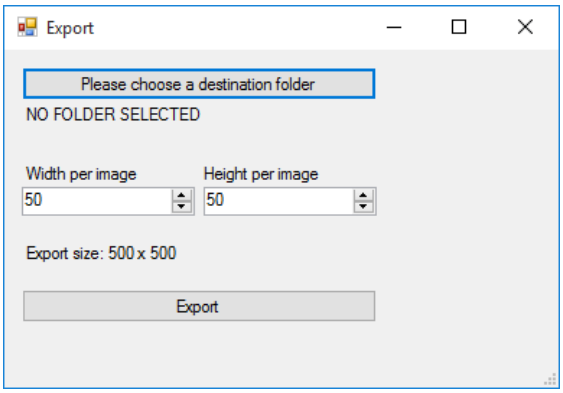
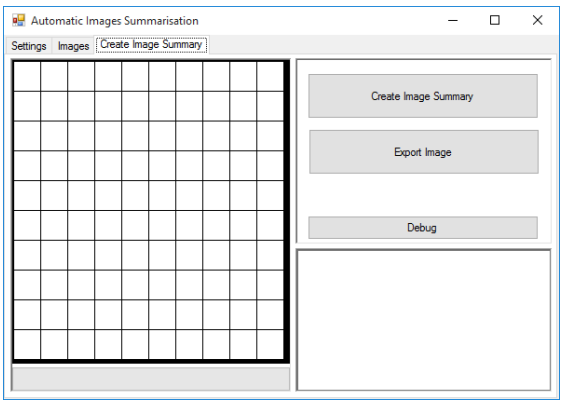


Figure 3.4: Die Rasteransicht der Anwendung



3.2 Umgang mit Bildern zur Laufzeit

Während der Implementierung der Anwendung hat sich gezeigt, dass es nicht praktikabel ist, alle Bilder des Nutzers auf einmal zu laden, um mit ihnen direkt die Füllung des Rasters durchzunehmen. Probleme zeigten sich hinsichtlich des Speicheraufwandes, die sich in Ausnahmefehler bezüglich des verfügbaren Speichers der Anwendung äußerten. Zudem führt das direkte Arbeiten mit den Bildern zu erheblichen Verschlechterungen der Laufzeit, weil jede Datenbewegung der Zellenelemente gleichzeitig zu einer entsprechenden Bewegung der Bilddaten führt. Weiters dauert auch das Zeichnen des Rasters erheblich länger, wenn für jeden Zeichenvorgang jedes Bild in ein Bildobjekt umgewandelt werden muss, mit dem EMGU, der Wrapper von OpenCV, arbeiten kann. Zudem müssen die Bilder auf die passende Größe skaliert werden, bevor sie gezeichnet werden.

Aufgrund dieser Nachteile wurde dazu übergegangen, nicht direkt mit den Bilddaten zu arbeiten, sondern jeder Gruppe eine zufällige Farbe zuzuweisen und zunächst während der Abarbeitung nur diese Farben in den Zellen zu rendern. In den Elementen, die platziert werden, ist lediglich der Speicherpfad zum entsprechenden Bild abgelegt. Beim Rendern des Ergebnisbildes sowie beim Export wird auf diesen Pfad zugegriffen und jedes Bild nacheinander temporär geladen, skaliert und gerendert.

3.3 Ansatz 1: Verformbare Cluster

3.3.1 Beschreibung des Ansatzes

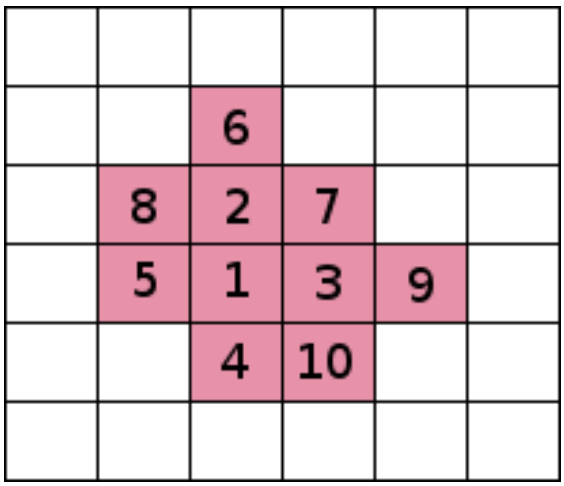
Die grundlegende Idee hinter diesem Ansatz war, dass man in einem Vorverarbeitungsschritt die Bilder der Gruppen bereits außerhalb des Rasters zusammenhängend anordnet. Diese Cluster sollen anschließend in den Raster fallen, wobei sie sich den Gegebenheiten möglichst gut anpassen sollen, sodass sie kompakt zu liegen kommen. Weiters sollen Lücken im Vorhinein vermieden werden. Dieser Ansatz geht davon aus, dass die Cluster bis zu einem gewissen Grad verformbar sind, ohne dass ihre Topologie in mehrere Teile zerfällt.

3.3.2 Algorithmus

Schritt 1: Bildung der Cluster
Im ersten Schritt werden die Bilder jeder Gruppe innerhalb eines lokalen Koordinatensystems und außerhalb der Koordinaten des Rasters zu einem zusammenhängenden Cluster angeordnet. Ausgehend von der ersten Zelle, die platziert wird, werden zunächst nacheinander der obere, rechte, untere und linke Nachbarplatz mit dem nächsten Bild der Gruppe besetzt. Danach werden die Nachbarn des zweiten Bildes, das platziert wurde, auf die selbe Art und Weise befüllt, bis schließlich alle Bilder platziert wurden. Durch diese Vorgehensweise ergeben sich rautenförmige Strukturen. In Abbildung 3.5 ist beispielsweise der Bildungsvorgang des Clusters einer Gruppe mit zehn Bildern demonstriert. Die Nummern in den Zellen geben die Reihenfolge an, in der die Bilder platziert werden. Bild 1 wird zuerst auf den Koordinaten [0 0] platziert. Danach folgen

die direkten Nachbarn des Bildes, bis schließlich die Nachbarn des Bildes 2 besetzt werden.

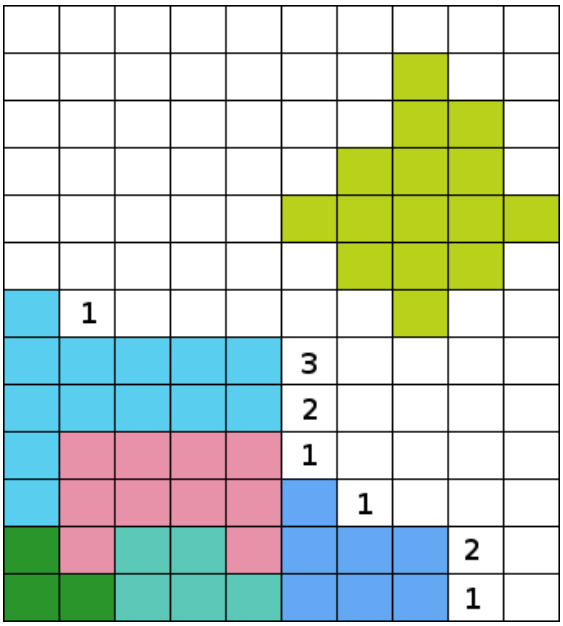
Figure 3.5: Bildung eines Clusters



Schritt 2: Platzierung der Cluster über dem Raster
Die Cluster, die sich ergeben, werden nach der Anzahl an Bildern, die sie beinhalten, sortiert, sodass zunächst kleinere Cluster abgearbeitet und im Raster platziert werden. Der Grund dafür besteht darin, dass sich bei den Tests gezeigt hat, dass kleinere Cluster am Ende des Algorithmus unter Umständen schwerer platziert werden können. Allerdings hängt die Schwierigkeit direkt mit der Methode der Randbehandlung zusammen, auf die noch näher eingegangen wird. Generell wird der Cluster, der abgearbeitet wird, zunächst so über dem Raster platziert, dass sich die unterste Zelle des Clusters eine Zeile über dem tatsächlichen Raster befindet. Was die horizontale Platzierung betrifft, so wird anhand einer Heuristik nach einer günstigen Startposition gesucht. Wie sich bei den Tests gezeigt hat, hat diese Startposition großen Einfluss auf die Qualität des Endergebnisses. Das liegt insbesondere daran, dass eine ungünstige Startposition dazu führen kann, dass im weiteren Verlauf der Platzierung der Cluster Lücken entstehen, welche in weiterer Folge ausgefüllt werden müssen. Das nachträgliche Ausfüllen von Lücken führt allerdings zu einer Verschlechterung hinsichtlich der Kompaktheit der Cluster. Dabei wird so vorgegangen, dass der Raster, in dem möglicherweise schon Bilder platziert sind, genauer betrachtet wird. Der Algorithmus sucht nach der Stelle, an der die größte Klippe existiert. Eine Klippenposition ist definiert als eine x-Position einer Spalte, in der eine Menge an freien Zellen existiert, die entweder um eine Position links oder rechts versetzt bereits mit Bildern befüllt ist. Je mehr freie Zellen in einer solchen Klippe existieren, desto höher ist sie.

Die höchste Klippe definiert den Startpunkt des nächsten Clusters, der an diese Position verschoben wird. Sollte die Bounding Box des Clusters beim Verschieben über

Figure 3.6: Positionierung des Clusters anhand der höchsten Klippe

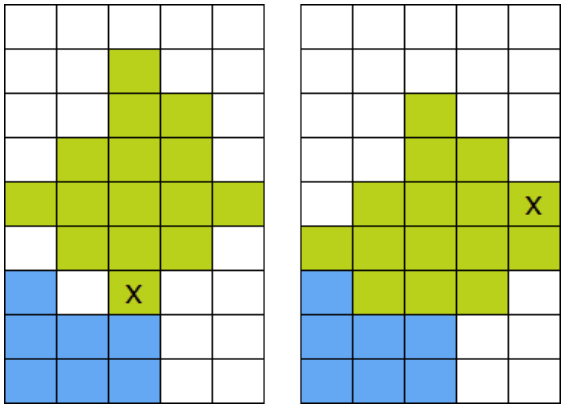


den Rand des Rasters ragen, so wird dieser soweit zurückverschoben, dass der Cluster wieder innerhalb des Rasters liegt. Ein Beispiel für die Suche nach der höchsten Klippe ist in Abbildung 3.6 illustriert. In dieser Situation erkennt der Algorithmus insgesamt vier mögliche Klippenpositionen, wobei die Klippe mit der Höhe 3 am höchsten ausfällt. Die horizontale Position des nächsten Clusters, der abgearbeitet wird, wird also so gewählt, dass er beim im nächsten Schritt so in den Raster fällt, dass er lückenlos mit dieser Klippe abschließt. Für den Fall, dass im Raster keine Klippe existiert, so wie es beispielsweise bei der Abarbeitung des ersten Clusters der Fall ist, wird die Startposition möglichst weit links im Raster gewählt.

Schritt 3: "Fallen lassen" des Clusters
Nachdem eine günstige Startposition für den aktuellen Cluster gefunden wurde, wird dieser sukzessive "fallen gelassen". Dabei werden alle Bilder des Clusters Zelle für Zelle nach unten bewegt. Dieser Vorgang ist erst dann beendet, wenn jedes der untersten Bilder des Clusters keine freie Rasterzelle mehr unter sich aufweist. Sollten manche Bilder bereits am Boden des Rasters angekommen sein, so werden die Bilder der gleichen Spalte nach unten bewegt. Gleichzeitig wird das unterste Bild dieser Spalte aus dem Raster entfernt und zum oberen Nachbarn des obersten Bildes des Clusters, welches die tiefste Position aufweist. Durch diese Vorgehensweise wird der obere Rand des Clusters, der zu Beginn noch rautenförmig war, immer weiter einem Rechteck angeglichen. Abbildung 3.7 zeigt diesen Vorgang des Verschiebens während des Falls. Der grüne Cluster befindet sich zunächst an einer Position, in der eine Verschiebung nach unten dazu führen würde, dass sich das mit 'x' markierte Bild mit dem blauen Cluster überlappen würde. Deshalb

wird dieses Bild bei der nächsten Verschiebung nach unten aus dem Raster entfernt und an eine der tiefsten Stellen am oberen Rand des Clusters gesetzt. Für die nächste Verschiebung nach unten geschieht das gleiche für die drei grünen Zellen, die nun am blauen Cluster angrenzen.

Figure 3.7: Verschiebung während des Fallvorgangs



Nachdem der Cluster im Raster zu liegen gekommen ist, ergibt sich im Allgemeinen ein oberer Rand, der noch weiter ausgeglichen werden kann. Dabei wird so vorgegangen, dass das rechte obere Bild des Clusters entfernt wird, und in die unterste Zelle möglichst weit links gelangt, in der der Cluster noch nicht vollständig befüllt ist. Dieser Vorgang wird wiederholt, bis der Cluster vollständig ausgeglichen ist. Danach werden die Bilder des oberen Randes, so weit es geht, nach links verschoben.

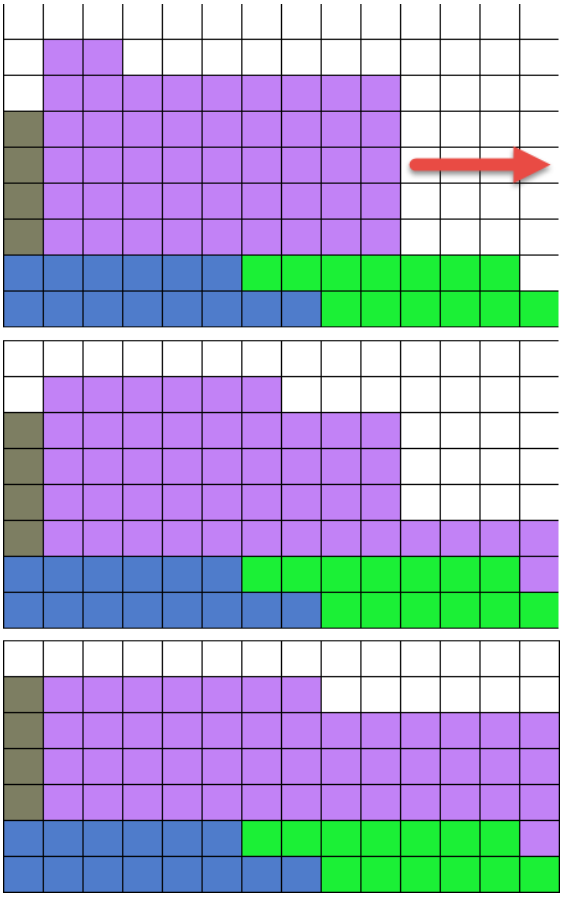
Schritt 4: Füllen von potentiellen Lücken
Um die Qualität des Endergebnisses zu verbessern versucht der Algorithmus nun, potentielle Lücken auszufüllen, indem der Cluster sich so verformt, dass diese Lücken gefüllt werden. Dabei muss zunächst festgelegt werden, wie eine solche Lücke erkannt wird.

Ausgehend vom rechten und linken Rand des Clusters wird die minimale Entfernung zur nächsten besetzten Rasterzelle, beziehungsweise zum Ende des Rasters ermittelt. Danach wird die Bounding Box des Clusters untersucht und die Höhe des Clusters ermittelt. Sollte die oberste Zeile des Clusters nicht vollständig gefüllt sein, so wird diese Zeile nicht zur Höhe hinzugerechnet. Sollte der Abstand zur nächsten besetzten Zelle geringer als diese Höhe sein, so kann davon ausgegangen werden, dass der Cluster genügend Masse besitzt, um die Lücke zu füllen, ohne dass die Kompaktheit des Cluster zu stark verschlechtert wird.

Daraufhin wird das Fundament für den Füllvorgang gelegt. Dabei werden sukzessive Bilder vom oberen-rechten Rand des Clusters entnommen und in Richtung der Lücke am Boden ausgelegt. Dieser Vorgang wiederholt sich, solange das Fundament nicht nach oben wachsen muss. Sobald das Fundament gelegt worden ist, befindet sich der Cluster in einem Zustand, in dem er wieder geebnet werden kann, woraufhin sich der aus Schritt 3 bekannte Vorgang wiederholt.

Ein Beispiel für den Vorgang des Füllens der Lücke wird in Abbildung 3.8 näher illustriert. Zunächst wird der Abstand des violetten Clusters zum Rand des Rasters ermittelt. Dieser Abstand ist geringer als die Höhe des Rasters, weshalb der Füllvorgang gestartet wird. Im nächsten Schritt ist zu sehen, wie das Fundament gelegt wird. Bilder werden vom oberen Rand des Clusters entnommen und im Anschluss am Boden platziert. Nachdem das Fundament gelegt wurde ist zu sehen, dass der Cluster ausgeglichen werden kann. Nach dem Vorgang des Ausgleichens wurde die Lücke erfolgreich gefüllt.

Figure 3.8: Füllen einer Lücke



Schritt 5: Prüfen auf eine günstigere Startposition
Die Vorgehensweise des Ansatzes geht davon aus, dass genügend Platz vorhanden ist, um den Cluster zu platzieren. Gegen Ende wird dieser Platz allerdings immer geringer, bis er schließlich nicht mehr ausreicht und eine andere Vorgehensweise gewählt werden muss, um den Vorgang abzuschließen. Diese Phase wird in weiterer Folge als Randbehandlung

bezeichnet. Um diese Randbehandlung so weit wie möglich hinauszuzögern, wird nun der Abstand vom Cluster zum oberen Rand des Rasters gemessen. Sollte dieser Abstand geringer sein, als die halbe Höhe der Bounding Box des Clusters, so kann es sinnvoll sein, die Startposition nicht anhand der höchsten Klippe, sondern anhand der tiefsten freien Stelle des Rasters zu wählen. Sollte das der Fall sein, so wird der Cluster zunächst aus dem Raster entfernt. Danach wird die tiefste freie Zelle im Raster ermittelt. Für den Fall, dass es mehrere Stellen der selben Tiefe gibt, wird diejenige gewählt, die sich am weitesten links befindet. Die Startposition wird daraufhin so gewählt, dass die Spalte des Clusters, die sich am weitesten Links befindet, diese Stelle auffüllt. Sollte der Cluster sich an dieser Position außerhalb des Rasters befinden, so wird dieser soweit verschoben, dass er sich im Raster befindet. Im Anschluss wiederholen sich Schritt 3 und Schritt 4. Danach wird wieder ermittelt, ob sich der Cluster zu nahe am oberen Rand des Rasters befindet. Sollte das der Fall sein wird der Cluster entfernt und die Randbehandlung gestartet und auch alle folgenden Cluster mit dieser Randbehandlung platziert. Die Randbehandlung entspricht der Vorgehensweise, die im Ansatz 2 erläutert wird.

3.3.3 Untersuchung der Ergebnisse

Die Beschreibung der Qualität der Ergebnisse muss in zwei Bereiche gegliedert werden, da ab einem gewissen Zeitpunkt die Randbehandlung zu tragen kommt. Die Qualität der Randbehandlung wird näher im dritten Lösungsansatz beschrieben. Was die allgemeine Vorgehensweise betrifft lassen sich die Ergebnisse wie folgt zusammenfassen.

Durch die Eigenschaft, dass die Cluster Unebenheiten selbstständig ausgleichen, ergeben sich im Allgemeinen Strukturen, die eine klare, geradlinige Grenzlinie aufweisen. Dadurch, dass die Cluster sich an den Untergrund anpassen und versuchen, Lücken zu füllen, verschlechtert sich zwar ihre Kompaktheit, da ihre Höhe verringert und ihre Breite erhöht wird, allerdings wird dadurch das Gesamtbild verbessert.

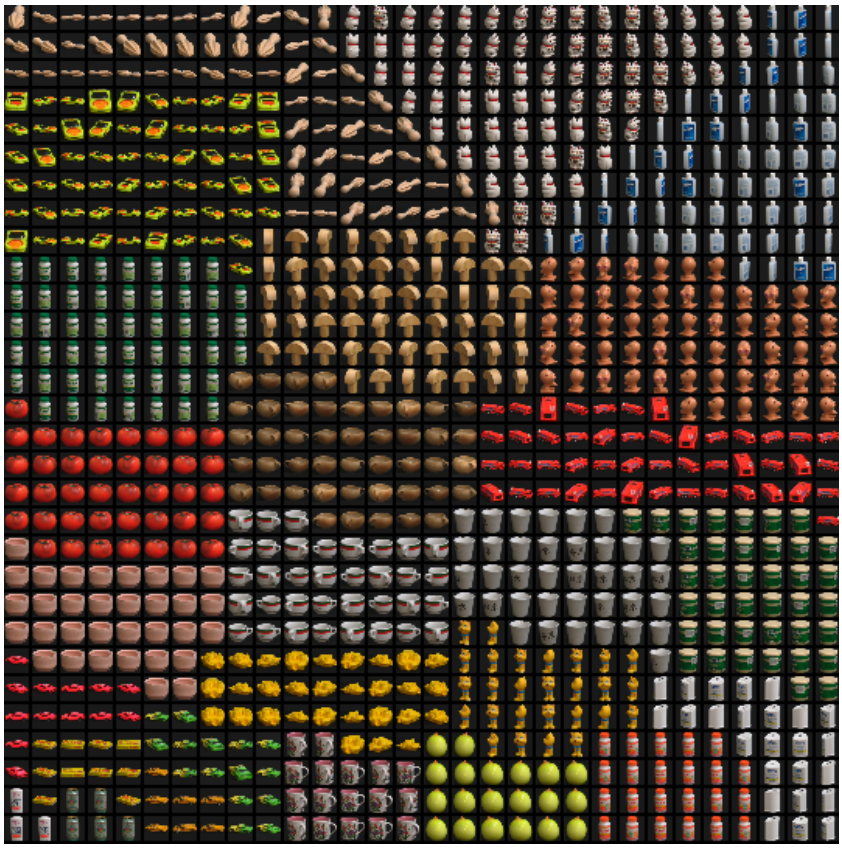
Weiters fällt noch auf, dass durch die aufsteigende Sortierung der Cluster nach ihrer Anzahl der Elemente ein Gesamtbild entsteht, in dem sich die kleinen Gruppen weiter unten befinden. Auch die Randbehandlung ist erkennbar. Beispielsweise ist in Abbildung 3.9 deutlich sichtbar, dass die obersten drei Cluster mittels Randbehandlung platziert wurden, da ihre Strukturen diagonale Grenzen aufweisen, die in den üblichen Strukturen des Ansatzes nicht vorkommen.

3.4 Ansatz 2: Linien Algorithmus

3.4.1 Beschreibung des Ansatzes

Die Grundidee hinter dieser Lösung besteht darin, dass eine zusammenhängende Linie durch den Raster gezogen werden soll, womit gewährleistet ist, dass auf jeden Fall immer eine gültige Lösung erzeugt werden kann. Der naive Ansatz würde darin bestehen, dass man ausgehend vom Eckpunkt links unten im Raster eine Linie bis zum rechten Rand zieht. Anschließend wird eine Zelle darüber befüllt und der Vorgang für die entgegengesetzte Richtung wiederholt. Diese Schritte werden solange wiederholt, bis

Figure 3.9: Beispiel für ein Ergebnis des ersten Ansatzes

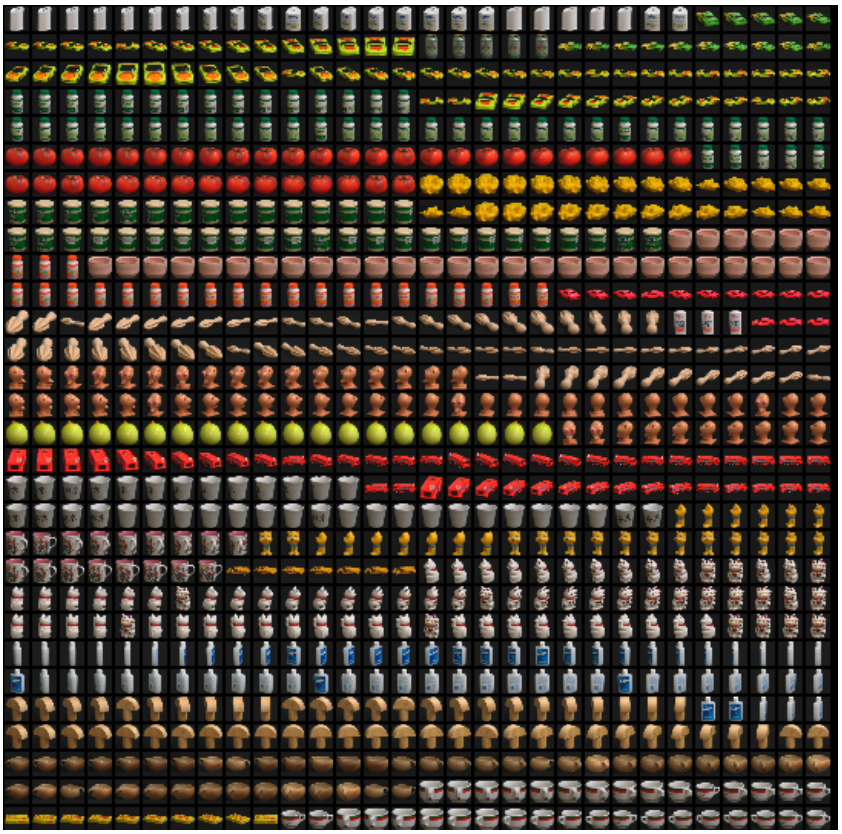


alle Bilder abgearbeitet worden sind. Durch diesen Ansatz entsteht zwar eine gültige Lösung, allerdings werden diese kaum als kompakte Cluster wahrgenommen, da sie im schlechtesten Fall lediglich eine Höhe von einer Zelle aufweisen. Selbst wenn sich die Höhe aufgrund der Anzahl der Bilder über mehrere Zeilen erstreckt bleibt immer noch die Tatsache, dass ein positionierter Cluster die gesamte Breite des Rasters in Anspruch nimmt. Ein Beispiel für ein Ergebnis, das ein solcher Linienzug liefert, ist in Abbildung 3.10 zu sehen.

Um die Ergebnisse zu verbessern wird der Raster zunächst in mehrere Regionen aufgeteilt, mit der Voraussetzung, dass jede dieser Regionen mit einer Linie zusammenhängend durchlaufen werden kann, und dass zwischen den abgearbeiteten Regionen eine Verbindungsstelle am Übergang existiert, wodurch die Gültigkeit der Lösung gewährleistet ist. Es stellt sich heraus, dass es nicht nur möglich ist, immer eine solche Aufteilung des Rasters zu finden, sondern auch, dass man die Anzahl der Regionen wählen kann. Durch die Wahl dieser Anzahl kann das Ergebnis vom Benutzer weiter beeinflusst werden.

Die Gestalt des Linienzuges, der genutzt wird, ähnelt sehr stark der des Murray Polygons (siehe Kapitel 2.5).

Figure 3.10: Beispiel für ein Ergebnis des naiven Ansatzes



3.4.2 Algorithmus

Schritt 1: Unterteilen des Rasters in Regionen
Damit die Regionen, in die der Raster vertikal unterteilt wird, zusammenhängend mit einem Linienzug durchlaufen werden können, muss der Raster eine ungerade Anzahl an Zeilen aufweisen. Andernfalls ist es nur möglich den Raster in zwei horizontale Regionen zu unterteilen. Mit dieser Vorgabe ist aber die Bedingung verletzt, dass der Nutzer dazu in der Lage sein soll, die Größe des Rasters frei zu wählen. Allerdings ist es möglich, den Raster bei einer geraden Anzahl an Zeilen nochmals horizontal zu unterteilen, wodurch zwei getrennte Regionen mit ungerader Anzahl an Zeilen entstehen (siehe Abbildungen 3.11c und 3.11d). Im Anschluss kann der Raster beliebig oft vertikal unterteilt werden.

Die horizontale Unterteilung wird vom Algorithmus so gewählt, dass der Raster möglichst mittig unterteilt wird. Was die vertikalen Unterteilungen angeht kann die Anzahl der Schnitte vom Benutzer gewählt werden. Der Algorithmus unterteilt den Rastern anschließend so, dass jede Region, so weit es möglich ist, die selbe Breite aufweist.

Schritt 2: Initialisieren des Algorithmus
Nach der Unterteilung des Rasters in Regionen muss die Startposition für den Linienzug

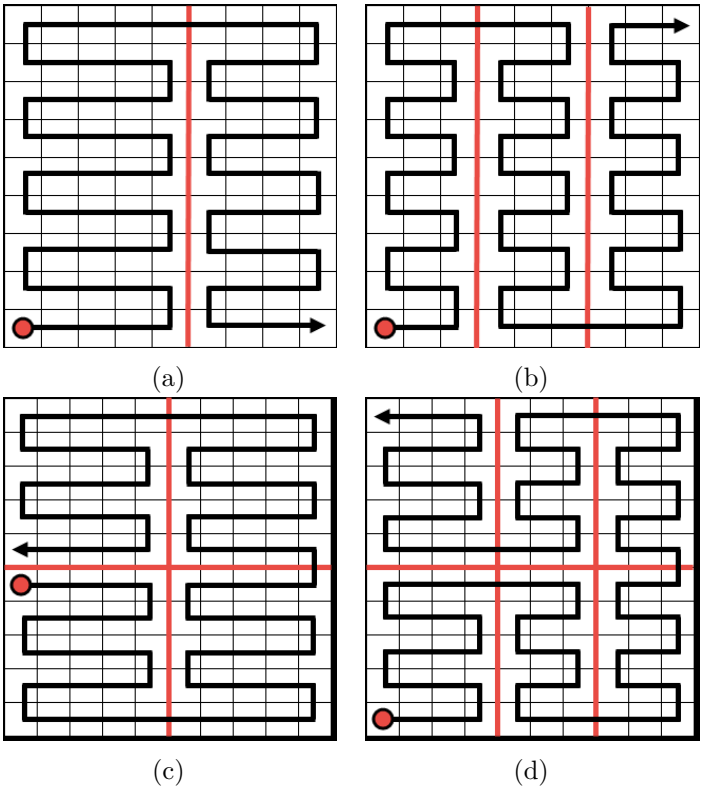


Figure 3.11: Mögliche Rasterteilungen und Startpositionen

ermittelt werden. Dazu müssen lediglich vier Fälle unterschieden werden, wobei nur die Anzahl Zeilen und die Anzahl der vertikalen Regionen, die vom Nutzer gewählt wurde, ausschlaggebend sind.

- Anzahl der Zeilen ungerade und Anzahl der horizontalen Regionen gerade (siehe Abbildung 3.11a)
- Anzahl der Zeilen ungerade und Anzahl der horizontalen Regionen ungerade (siehe Abbildung 3.11b)
- Anzahl der Zeilen gerade und Anzahl der horizontalen Regionen gerade (siehe Abbildung 3.11c)
- Anzahl der Zeilen gerade und Anzahl der horizontalen Regionen ungerade (siehe Abbildung 3.11d)

Schritt 3: Verteilen der Bilder auf dem Linienzug
Der Algorithmus befüllt nacheinander, Region für Region, die Zellen im Raster. Dabei wird bei der Region links bzw. links unten begonnen und fortlaufend die Region rechts davon als nächstes bearbeitet. Falls der Raster auch horizontal unterteilt wurde, wird

nach der Region ganz rechts die Region darüber und anschließend alle Regionen in der linken Richtung abgearbeitet.

Prinzipiell ist die Vorgehensweise für jede Region die selbe. Liegt der Eintrittspunkt des Linienzugs am oberen Rand der Region, so wird die primäre Richtung des Füllalgorithmus nach unten gesetzt, andernfalls nach oben. Falls der Eintrittspunk am linken Rand der Region liegt wird die Sekundärrichtung nach rechts gesetzt, ansonsten nach links.

Zunächst wird der Linienzug in Sekundärrichtung gezogen, bis der Rand der Region erreicht ist. Im Anschluss wird die nächste Zelle um eine Position in Richtung der Primärrichtung gefüllt, die Sekundärrichtung in entgegengesetzte Richtung gesetzt und fortgefahren. Dieser Vorgang wiederholt sich, bis schließlich die letzte Zeile befüllt wurde. Danach wird ein weiterer Schritt in Richtung der Sekundärrichtung gemacht, wodurch der Übergang von der abgearbeiteten Region in die nächste Region entsteht. Für den Fall, dass der Raster auch horizontal unterteilt wurde, werden die obere und untere Region am rechten Rand des Rasters als eine Region betrachtet.

3.4.3 Untersuchung der Ergebnisse

Durch die Regionsgrenzen ergeben sich Strukturen mit klar definierten Grenzen, die leicht erkennbar sind und im Allgemeinen als kompakt und zusammenhängend wahrgenommen werden können. Ein Nachteil der Methode wird sichtbar, wenn kurz nach der Abarbeitung einer Region nur noch wenige Bilder der Gruppe angeordnet werden müssen. Dadurch ist die Anordnung der Bilder für einen Bereich eher linienförmig. Ein Vorteil besteht darin, dass der Nutzer die Anzahl der vertikalen Regionen wählen kann. Somit kann beispielsweise für den Fall, dass viele Cluster höher als breit sind, die Anzahl der vertikalen Regionen verringert werden, sodass nach der Neuberechnung breitere Strukturen entstehen. Zudem bietet dieser Ansatz die schnellste Lösung der in dieser Arbeit vorgestellten Methoden, da durch die Füllung anhand eines Linienzugs lineare Laufzeit erreicht werden kann.

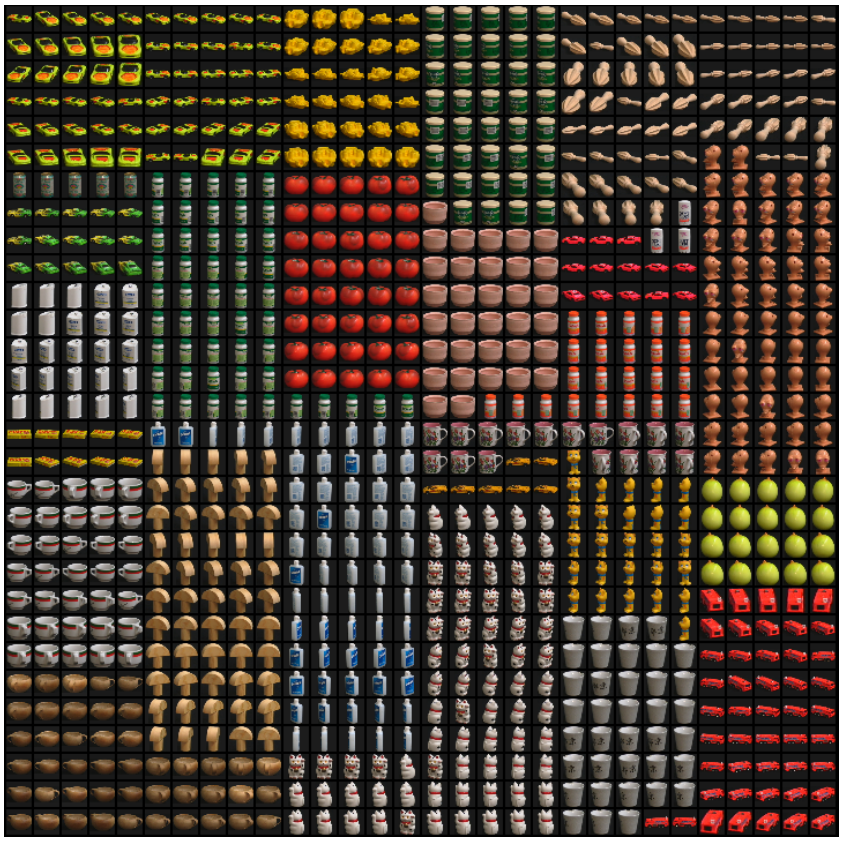
Abbildung 3.12 zeigt ein Ergebnisbild wobei der Nutzer 6 vertikale Regionen gewählt hat und der Raster aufgrund der geraden Anzahl an Zeilen zusätzlich in 2 horizontale Regionen geteilt wurde. Die entstandenen Regionsgrenzen lassen sich beim bloßen Betrachten des Ergebnisses erahnen.

3.5 Ansatz 3: Rating Algorithmus

3.5.1 Beschreibung des Ansatzes

Der dritte Ansatz, der in dieser Arbeit beschrieben wird, basiert auf der Idee, dass jede freie Zelle des Rasters bewertet wird, bevor das erste Bild einer Gruppe platziert wird. Nachdem die Startposition des ersten Bildes ermittelt wurde, werden alle folgenden Bilder dieser Gruppe nacheinander so angeordnet, dass der Cluster im Raster gleichmäßig anwächst.

Figure 3.12: Beispiel für ein Ergebnis des zweiten Ansatzes



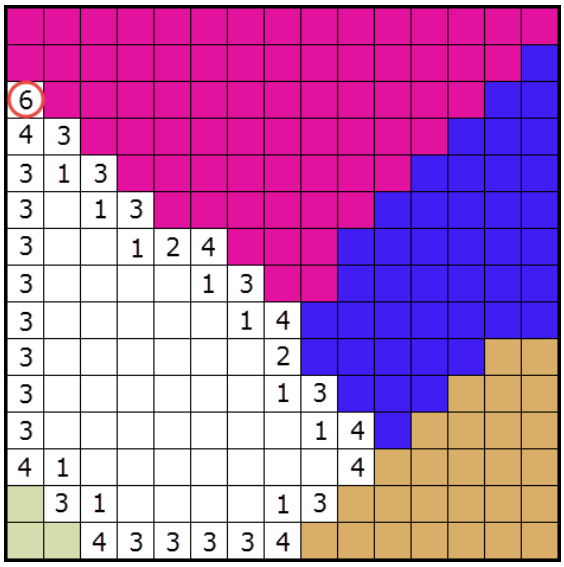
3.5.2 Algorithmus

Schritt 1: Bewerten des Rasters
Für die Bewertung wird jede freie Zelle des Rasters betrachtet. Nachdem alle freien Zellen mit dem Wert 0 initialisiert wurden, werden alle 8 benachbarten Zellen jeder freien Zelle betrachtet. Für jede dieser benachbarten Zellen, die nicht frei sind, wird der Wert der aktuellen Zelle um 1 erhöht. Was Nachbarn betrifft, die über den Rand des Rasters ragen, so werden diese ebenfalls gezählt.

In Abbildung 3.13 ist ein Beispiel für die Bewertung des Rasters gegeben. Lediglich die Zellen am Rand der 'freien' Region sind von Bedeutung. Wie man in der Abbildung erkennt, werden auch Felder am Rand des Rasters gewertet. In diesem Beispiel wird letztendlich die Zelle mit dem Wert 6 gewählt, da dies dem höchsten errechneten Wert entspricht. Für den Fall, dass mehrere Zellen den Höchstwert aufweisen, wird eine beliebige Zelle gewählt.

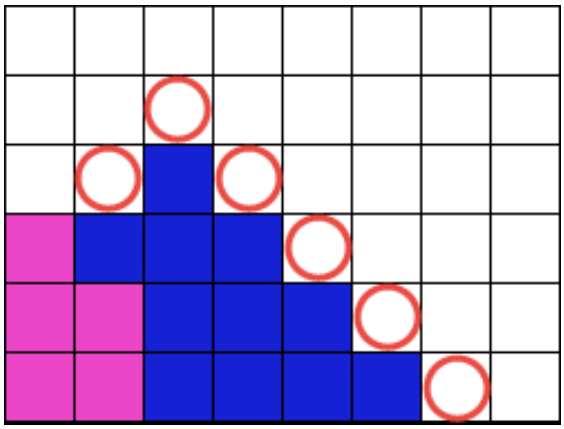
Im Anschluss wird jedes Bild der Gruppe nacheinander abgearbeitet und platziert. Um die Bilder platzieren zu können, ist es notwendig, entsprechende mögliche Zielpositionen zu ermitteln. Dieser Vorgang wird im nächsten Schritt näher erläutert.

Figure 3.13: Beispiel für eine Bewertung des Rasters



Schritt 2: Erfassen der nächsten Zielpositionen
Im nächsten Schritt hängt die Zielposition davon ab, ob es sich bei der aktuellen Zelle um die erste Zelle der Gruppe handelt. Sollte das der Fall sein, so wird das erste Bild der Gruppe auf einer beliebigen freien Zelle platziert, die im Raster die höchste Bewertung aufweist (siehe Schritt 1). Andernfalls werden alle gültigen freien Nachbarzellen der aktuellen Gruppe erfasst. Diese Menge an Randpositionen wird der Reihe nach mit den nächsten Zellen der Gruppe befüllt. Sollten alle dieser Randpositionen befüllt worden sein, so werden erneut alle gültigen Randpositionen erfasst. Sind alle Zellen der aktuellen Gruppe positioniert, so fährt der Algorithmus wieder bei Schritt 1 fort.

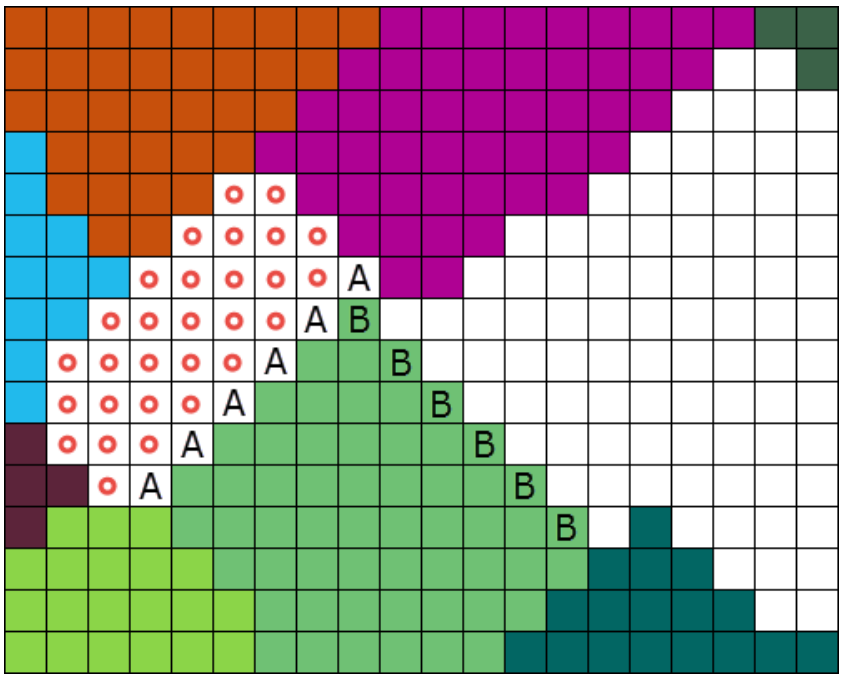
Figure 3.14: Erfassen von möglichen Zielpositionen



In Abbildung 3.14 befindet sich beispielsweise der blaue Cluster mitten im Vorgang der Platzierung. Alle letzten möglichen Zielpositionen, die ermittelt wurden, sind bereits mit Bildern befüllt. Deshalb werden neue Zielpositionen am Rand des Clusters ermittelt. Diese sind in der Abbildung mit roten Kreisen gekennzeichnet.

Schritt 3: Korrigieren von ungünstigen Platzierungen
Durch die Art der Positionierung kann es durchaus vorkommen, dass die Region der freien Zellen im Raster in zwei Teile geteilt wird. Falls das passiert, muss die Gruppe, die aktuell abgearbeitet wird, korrigiert werden, was wie folgt geschieht. Die Zellen der zwei Regionen an freien Zellen werden erfasst und die kleinere Region als Zielregion definiert, die es zu befüllen gilt. Ähnlich wie in Schritt 2 werden wieder Randmengen erfasst, wobei diesmal 2 Mengen betrachtet werden. Eine Menge A beinhaltet die Menge aller leeren Randzellen, die in der kleineren freien Region liegen. Die zweite Menge B enthält die besetzten Randzellen, die an der größeren Region angrenzen. Im Anschluss werden nacheinander Zellen aus Menge B entfernt und auf eine Position der Menge A platziert. Falls eine der Mengen leer ist, wird sie neu erfasst. Dieser Vorgang setzt sich solange fort, bis alle Zellen der kleineren Region einmal befüllt wurden. Falls die kleinere Region größer war als der Cluster selbst, so kann es durchaus vorkommen, dass sämtliche Zellen des Clusters gewandert sind. Bevor mit der Platzierung der übrigen Bilder fortgefahren wird, wird nochmals überprüft, ob die freie Region in zwei Teile zerfallen ist. Falls dies der Fall ist wiederholt sich Schritt 3.

Figure 3.15: Beispiel für eine Korrektur



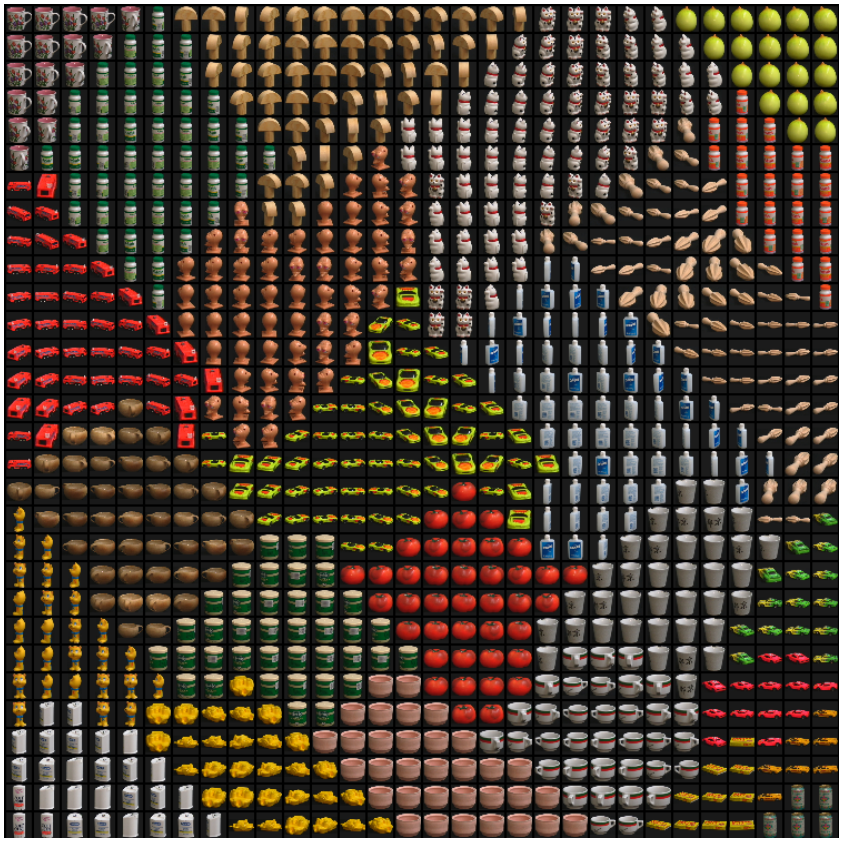
Um den Vorgang näher zu erläutern wird in Abbildung 3.15 ein konkretes Beispiel

gegeben. In diesem Fall wird aktuell der grüne Cluster abgearbeitet. Dieser grenzt an dem violetten Cluster an, sodass die freie Region in zwei Teile unterteilt wurde, wobei die kleinere Region links im Bild zu sehen ist. Es gilt nun, den Cluster so zu verschieben, dass sämtliche Zellen, die mit einem roten Kreis markiert sind, zumindest einmal aufgefüllt wurden. Die Zellen mit der Markierung A sind die Zellen, die in den nächsten Durchläufen der Korrektur befüllt werden. Die Zellen mit der Markierung B sind die Zellen, die für das Auffüllen aus dem Raster entfernt und anschließend auf eine Position der Zellen A gesetzt werden.

3.5.3 Untersuchung der Ergebnisse

Generell liefert dieser Ansatz Ergebnisse, die man als kompakt wahrnimmt. Aufgrund der Art, wie die Zellen anhand der freien Randzellen befüllt werden, ergeben sich Strukturen, die sich als Fragmente von Rauten beschreiben lassen. Durch die Korrekturen kann sich deren Gestalt allerdings auch stark abändern. Dadurch, dass die Grenzen häufiger diagonal verlaufen, entsteht je nach Auflösung des Rasters ein treppenförmiger Eindruck der Grenzen. Ob diese Tatsache als störend empfunden und die Wahrnehmung der Gruppen als Cluster davon negativ beeinflusst wird, müsste genauer evaluiert werden. Insgesamt sind die Strukturen dadurch allerdings dynamischer und möglicherweise interessanter für den Betrachter.

Figure 3.16: Beispiel für ein Ergebnis des dritten Ansatzes

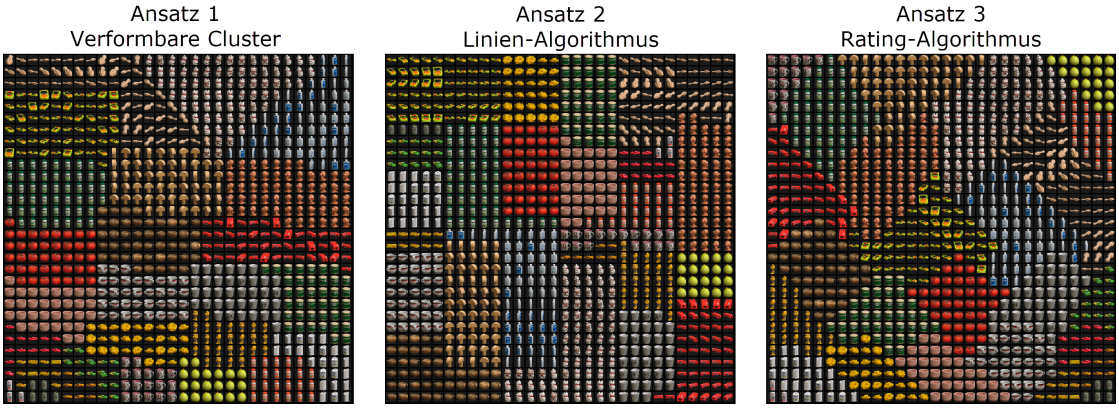


CHAPTER 4

Zusammenfassung und Vergleich der Ergebnisse

Die drei Ansätze, die in dieser Arbeit vorgestellt wurden, haben gezeigt, dass das Problem, Bilder zusammenhängend in einem Raster unterzubringen, auf mehreren verschiedenen Wegen behandelt werden kann. Alle Ansätze liefern gültige Ergebnisse, allerdings zeigt sich, dass die Qualität der Ergebnisse stark davon abhängt, wie an das Problem prinzipiell herangegangen wird. Abbildung 4.1 zeigt die Ergebnisse der drei Ansätze die mit den selben Ausgangsdaten ermittelt wurden.

Figure 4.1: Ergebnisse der verschiedenen Ansätze



Der qualitative Vergleich der Ansätze gestaltet sich aus mehreren Gründen schwierig. Einerseits muss definiert werden, welche Merkmale der Ergebnisse erfasst und verglichen werden. Beispielsweise könnte man für diesen Zweck das durchschnittliche Seitenverhältnis der Bounding Boxes der platzierten Gruppen heranziehen. Auch die Betrachtung des

Table 4.1: Messergebnisse der Laufzeiten der Ansätze und des Speicherbedarfs der Anwendung

n	**Ansatz 1 / ms**	**Ansatz 2 / ms**	**Ansatz 3 / ms**	**Speicherbedarf / MB**
100	66	3	57	10 – 50
200	128	2	165	10 – 50
300	192	3	352	10 – 50
400	242	2	691	10 – 50
500	264	3	1150	10 – 50
600	340	5	1753	10 – 50
700	341	4	2776	10 – 50
800	412	4	4031	10 – 50
900	339	6	4974	10 – 50
1000	412	5	7480	10 – 50
2000	1590	4	52626	10 – 50
3000	1692	4	180953	10 – 50
4000	1987	5	418719	10 – 50
5000	2713	8	817574	10 – 50

durchschnittlichen Verhältnisses zwischen Fläche und Umfang der Gruppen wäre ein sinnvolles Maß für den Vergleich der Ergebnisse. Ein weiteres Merkmal, das für den Vergleich interessant ist, wäre eine genauere Betrachtung der Laufzeit der Algorithmen. Das Problem beim Vergleich besteht darin, dass die Ergebnisse im Wesentlichen von Anzahl der Gruppen und deren zugehörigen Bildern abhängt, sowie von den Maßen des Rasters, die der Nutzer angibt. Trotzdem wurde die Laufzeit des Algorithmus für alle drei Ansätze für verschieden große Mengen an Ausgangsbildern festgehalten (siehe Tabelle 4.1). Dabei wurde lediglich die Zeit erfasst, welche benötigt wurde, um eine gültige Lösung zu ermitteln. Das Rendern der Zwischenergebnisse wurde für die Erfassung der Laufzeit deaktiviert, um die Ergebnisse nicht zu verfälschen. Der Rechner, auf dem die Messung erfolgt ist, verfügt über eine 2.40 GHz Quad-Core CPU (Intel Core i5-4258U) sowie über 8GB Arbeitsspeicher.

Die Ergebnisse zeigen, dass Ansatz 2, also der Linien-Algorithmus, aufgrund seiner linearen Herangehensweise eindeutig am schnellsten terminiert. Für Ansatz 3, also für den Rating-Algorithmus, lässt sich hingegen der stärkste Zuwachs an Laufzeit feststellen. Dies lässt sich dadurch erklären, dass nach jeder platzierten Zelle geprüft werden muss, ob die Region der freien Zellen in zwei Teile zerfallen ist. Was den Speicherbedarf der Anwendung betrifft, ließen sich keine wesentlichen Unterschiede zwischen den verschiedenen Herangehensweisen oder der Anzahl der Ausgangsbilder feststellen. Der Grund dafür liegt darin, dass die Bilder lediglich als String in der Anwendung verarbeitet werden und nur beim Rendern nacheinander temporär geladen werden. Ebenso wurden in den Ansätzen keine aufwändigen Datenstrukturen verwendet, die einen signifikanten Einfluss auf den Speicherbedarf nehmen könnten.

Ebenso wurden exemplarisch die Verhältnisse zwischen Fläche und Umfang der einzelnen Cluster der Ergebnisbilder aller drei Ansätze betrachtet. Abbildung 4.2 zeigt eine tabellarische Ansicht der Cluster mit entsprechender Anzahl an Elementen, Anzahl an Randbildern sowie deren Verhältnisse und eine Mittelung der Verhältnisse pro Ansatz.

Figure 4.2: Verhältnisse zwischen Anzahl der Elemente und Anzahl der Randelemente pro Cluster und Ansatz

		Ansatz 1		Ansatz 2		Ansatz 3	
Cluster	n	Randbilder	n / Randbilder	Randbilder	n / Randbilder	Randbilder	n / Randbilder
1	10	10	1,00	10	1,00	10	1,00
2	20	16	1,25	14	1,43	17	1,18
3	30	22	1,36	18	1,67	24	1,25
4	40	23	1,74	22	1,82	29	1,38
5	50	26	1,92	29	1,72	32	1,56
6	60	29	2,07	28	2,14	43	1,40
7	5	5	1,00	5	1,00	5	1,00
8	15	15	1,00	12	1,25	14	1,07
9	25	22	1,14	16	1,56	21	1,19
10	35	22	1,59	20	1,75	25	1,40
11	45	26	1,73	28	1,61	30	1,50
12	55	28	1,96	28	1,96	37	1,49
13	65	37	1,76	35	1,86	43	1,51
14	75	44	1,70	40	1,88	44	1,70
15	7	7	1,00	7	1,00	7	1,00
16	17	14	1,21	17	1,00	16	1,06
17	27	20	1,35	18	1,50	23	1,17
18	37	22	1,68	22	1,68	27	1,37
19	47	31	1,52	26	1,81	35	1,34
20	57	30	1,90	30	1,90	37	1,54
21	67	47	1,43	34	1,97	47	1,43
22	3	3	1,00	3	1,00	3	1,00
23	13	13	1,00	12	1,08	13	1,00
24	23	16	1,44	17	1,35	20	1,15
25	33	23	1,43	21	1,57	26	1,27
26	39	24	1,63	22	1,77	30	1,30
Summe:	900	Mittelwert:	1,45	Mittelwert:	1,55	Mittelwert:	1,28

Die Daten entstammen den Ergebnisbildern wie sie in Abbildung 4.1 zu sehen sind. Als Randbilder wurden sämtliche Bilder erfasst, die an ein Bild einer anderen Gruppe angrenzen, wobei auch diagonal benachbarte Bilder gezählt wurden. Laut diesen Ergebnissen liefert Ansatz 2 geringfügig bessere Ergebnisse als Ansatz 1, während die Cluster aus Ansatz 3 am wenigsten kompakt ausfallen. Grund dafür sind die eher diagonalen Grenzen der Cluster aus Ansatz 3, die in einer höheren Zahl an Randbildern resultieren. Wie aber bereits erwähnt hängen Vergleiche dieser Art stark von der Ausgangsmenge

der Bilder ab. Zudem hat der Nutzer durch die Wahl der Anzahl an Unterteilungen, die der Algorithmus in Ansatz 2 vornimmt, Einfluss auf das Endergebnis (siehe 3.4.2 - Algorithmus des zweiten Ansatzes).

Ein weiteres Qualitätsmerkmal, das betrachtet werden kann, ist die Art, wie die Algorithmen mit dem Fall umgehen, wenn weniger Bilder zur Verfügung gestellt werden, als freie Zellen im Raster existieren. Zwar terminieren alle Ansätze in diesem Fall, allerdings wurden in der Zielsetzung dieser Arbeit keine Bedingungen gestellt, wie die Algorithmen mit diesem Fall umgehen sollen. Somit bleiben je nach Ansatz unterschiedliche Bereiche des Rasters leer, wenn die Bilder platziert wurden. Der erste Ansatz terminiert in diesem Fall mit einer freien Region, die am oberen Rand des Rasters angesiedelt ist, und im Allgemeinen keine glatten Grenzen aufweist. Dagegen bleiben beim zweiten Ansatz für gewöhnlich die Zellen im linken oberen Bereich frei, wobei diese freie Region annähernd ein Rechteck bildet. Beim dritten Ansatz lässt es sich schwer vorhersagen, welche freie Region entsteht, allerdings weist diese Region eine ähnliche Struktur auf, wie die Strukturen der platzierten Cluster. Die drei Ansätze teilen sich lediglich die Tatsache, dass die freie Region, so wie die platzierten Cluster, eindeutig als zusammenhängende Gruppe erkennbar ist.

CHAPTER 5

Conclusio und Ausblick

Aufgrund der Natur der Aufgabenstellung, welche viel Freiraum für individuelle Lösungen bietet, war bereits zu Beginn zu erwarten, dass nicht die eine 'richtige' Lösung gefunden wurde, sofern eine solche für diese Aufgabenstellung existiert. Erst wenn klare Anforderungen an die Lösung sowie ein genau definiertes Maß für die Messung der Qualität der Ergebnisse beschrieben ist, lässt sich die Qualität der Ergebnisse so evaluieren, dass man ermitteln kann, ob eine optimale Lösung gefunden wurde oder nicht. Da aber lediglich gefordert wurde, dass die Cluster mit freiem Auge klar erkennbar sein müssen, kann man diese Aufgabe im Rahmen dieser Arbeit als gelöst betrachten. Welcher Ansatz in der Praxis gewählt werden sollte hängt im Wesentlichen von den Ausgangsmengen, den Maßen des Rasters sowie dem subjektiven Empfinden des Nutzers zur Bewertung der Kompaktheit der Ergebnisse ab. Während ein Nutzer eher rechteckige Strukturen bevorzugt, könnten andere Menschen eher kreisförmige Strukturen als kompakter wahrnehmen.

Was die in dieser Arbeit bearbeitete Zielsetzung betrifft, so sind noch einige Ansätze denkbar, mit denen das Problem gelöst werden kann, wobei sich die Ergebnisse - je nach Herangehensweise - unterschiedlich charakterisieren lassen können. Auch weiterführende Arbeiten an den bereits beschriebenen Ansätzen sind denkbar. Einerseits wäre es stellenweise möglich, die Algorithmen hinsichtlich ihrer Laufzeit weiter zu verbessern. Beispielsweise ist es im dritten Ansatz nicht notwendig, jede freie Zelle des Rasters zu bewerten. Andererseits könnte man sich noch mit dem Fall weiter auseinandersetzen, wenn mehrere Zellen im Raster zur Verfügung stehen, als Bilder, die im Raster platziert werden sollen. Es wäre beispielsweise wünschenswert, wenn die freien Zellen im Raster ausschließlich am Rand angesiedelt sind, unter Umständen so, dass das Gesamtergebnis der befüllten Zellen insgesamt nahezu als Rechteck beschrieben werden kann.

Außerdem stellt sich auch die Frage, ob es möglich ist, einen Algorithmus zu entwerfen, der den gesamten Kontext aller positionierten Cluster innerhalb des Rasters erfassen kann. Die hier vorgestellten Lösungen arbeiten Cluster für Cluster ab, wobei im Wesentlichen lediglich die freie Region im Raster von Relevanz ist. Vergleicht man diese Vorgehensweise

aber mit der eines Menschen, so lässt sich beobachten, dass Menschen durchaus in der Lage sind, durch bloßes Betrachten der Lösung bessere Positionierungen zu finden und gegebenenfalls Zellen austauschen, um die Cluster noch kompakter zu gestalten.

Letztendlich können weitere Verbesserungen hinsichtlich der Usability der Anwendung selbst vorgenommen werden. Wünschenswert wäre beispielsweise eine Drag and Drop Funktion, um Bilder und Gruppen durch Ziehen in die Anwendung erstellen zu können. Weiters ist es für die Ansicht der Gruppen und Bilder denkbar, eine Vorschau für die Bilder selbst zu implementieren, da derzeit nur der Dateiname angezeigt wird. Der Grund dafür liegt darin, dass möglicherweise nicht genügend Speicher zur Verfügung steht, um eine hohe Menge and Bilddaten verwalten zu können.

Bibliography

[Bad13] M. Bader. *Space-Filling Curves: An Introduction With Applications in Scientific Computing*. Springer, 2013.

[BAR14] W. Aigner H.Hauser S. Miksch B. Alsallakh, L. Micallef and P. Rodgers. Visualizing sets and set-typed data: State-of-the-art and future challenges. *Eurographics Conference on Visualization*, 2014.

[BRS13] T. Ritschel B. Reinert and H.P. Seidel. Interactive by-example design of artistic packing layouts. *ACM Transactions on Graphics*, 32, 2013.

[CCC09] G. Penn C. Collins and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15:1009–1016, 2009.

[CL90] J. H. Conway and J. C. Lagarias. Tiling with polyominoes and combinatorial group theory. *Journal of Combinatorial Theory, Series A*, 53:183–208, 1990.

[Col] A. J. Cole. The murray polygon. `http://alb.host.cs.st-andrews.ac.uk/cole/poly.html`. Accessed: 2015-08-27.

[CRL06] Y. Hamadi C. Rother, L. Bordeaux and A. Blake Lagarias. Autocollage. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2006*, 25:847–852, 2006.

[CYF13] J. Peng C. Yang, J. Shen and J. Fan. Puzzle-like collage. *Pattern Recognition*, 46:948–961, 2013.

[Gol94] S. W. Golomb. *Polynominoes*. Princeton University Press, 2 edition, 1994.

[GSE13] M. Gong G. Strong, R. Jensen and A.C. Elster. *Organizing Visual Data in Structured Layout by Maximizing Similarity-Proximity Correlation*. Springer, 2013.

[Kla69] D. A. Klarner. Packing a rectangle with congruent n-ominoes. *Journal of Combinatorial Theory*, 7:107–115, 1969.

[LTX11] S. Liu L. Tan, Y. Song and L. Xie. Imagehive: Interactive content-aware image summarization. *Computer Graphics and Applications, IEEE*, 32:46–55, 2011.

[NQT10] L. Song N. Quadrianto, A.J. Smola and T. Tuytelaars. Kernelized sorting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1809–1821, 2010.

[OFF15] M. Halber E. Sizikova O. Fried, S. DiVerdi and A. Finkelstein. Isomatch: Creating informative grid layouts. *EUROGRAPHICS*, 34, 2015.

[SA11] K. Schoeffmann and D. Ahlstrom. Similarity-based visualization for image browsing revisited. *Proceedings of the IEEE International Symposium on Multimedia*, pages 422–427, 2011.

[SGZM10] A. Tal S. Goferman and L. Zelnik-Manor. Puzzle-like collage. *Computer Graphics Forum*, 29:459–468, 2010.

[SS95] G. Scheithauer and U. Sommerweiß. Heuristics for the rectangle packing problem. 1995.

[SvZG09] X. Olivares S. van Zwol, R. van Leuken and L. Garcia. Visual diversification of image search results. *Proceedings of the 18th International Conference on World Wide Web*, pages 341–350, 2009.

[TLS09] J. Sun N. Zheng X. Tang T. Liu, J. Wang and H. Shum. Picture collage. *IEEE Transactions on Multimedia*, 11:1225–1239, 2009.

[TPM10] C. Ju E. Zhang T. Pham, R. Hess and R. Metoyer. Visualization of diversity in large multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics*, 16:1053–1062, 2010.