Seminar Computergrafik

WS 2009/10

Prof. Dr. W. Kurth,

Reinhard Hemmerling

PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing

---

Eingereicht von: Stefan Saalfeld

MatrNr: 20978954

Geb.-Datum: 07.05.1986

Inhaltsverzeichnis

Kurzfassung

Das Paper beschreibt einen neuen Algorithmus, der von interaktiven Bildbearbeitungstools genutzt werden kann, um verschiedene Funktionalitäten zu ermöglichen. Der Algorithmus nutzt eine Zufallssuche, um die Nachbarschaftssuche (engl.: Nearest Neighbour Search, kurz NNS) zum Bestimmen von Patches zu realisieren. Bisher konnte mit dem aufwändigen Ansatz der NNS keine Interaktivität erreicht werden. Der Algorithmus aus diesem Paper schafft jedoch Interaktivität durch den veränderten Ansatz mittels Zufallssuche, mit welchem er etwa 20- bis 100-mal so schnell ist wie vergleichbare Algorithmen. Die Eigenschaften des Algorithmus werden theoretisch durchgesprochen und sind an praktischen Anwendungen getestet, um die Qualität und die Performance zu zeigen. Er kann als Basis für viele Bildbearbeitungstools genutzt werden, wie zum Beispiel Inhaltssensible Bildskalierung und Deformation, Vervollständigung und Rekonstruktion von Bildbereichen und Neuanordnung der Bilddetails. Zusätzlich können Einschränkungen (Randbedingungen) festgelegt werden, um zusätzlich zur Interaktivität auch noch eine intuitive Bedienung zu gewährleisten.

Bild 1: Bildmanipulation an einem Beispiel

1 Einleitung

Durch digitale Bilder und den Möglichkeiten die Computer schaffen, haben Bildbearbeitungstools und Bildbearbeitungsmethoden an Bedeutung gewonnen. Sie erlauben Bilder auf verschiedene Arten zu manipulieren, zum Beispiel können Bilder skaliert werden ohne dass dabei der Inhalt verzerrt wird. Andere Algorithmen ermöglichen das Auffüllen von Fehlstellen mit passenden Strukturen, die das Bild gut ergänzen. Weiterhin existieren Neuanordnungsalgorithmen mit denen Artefakte verschoben und ganze Bilder neu zusammengesetzt werden können.

Alle diese Verfahren benötigen Benutzerinteraktionen, um gute Ergebnisse zu erzielen. Zunächst geben einige dieser Tools dem Benutzer die Möglichkeit Einschränkungen zu setzen, um bestimmte Bereiche anders zu behandeln, zum Beispiel um bestimmte Regionen von zu starker Veränderung zu schützen. Eine weitere Einschränkung kann genutzt werden, um dem Algorithmus zum Auffüllen von Fehlstellen einen Bereich vorzugeben, in dem dieser nach passenden Strukturen suchen soll. Diese Benutzeraktionen ermöglichen es dem Anwender seine Zielvorstellungen dem Programm mitzuteilen, welches ohne diese Zusatzinformationen das gewünschte Ergebnis mit geringerer Wahrscheinlichkeit erreichen würde. Außerdem hat der Anwender nicht immer ein konkretes Ziel vor Augen, in dem Fall kann er sich durch „trial and error“ mit dem Tool vertraut machen und zwischen den verschiedenen Ergebnissen wählen.

Diese Rolle der Interaktivität fordert zwei Eigenschaften von Bildbearbeitungstools. Es sollte erstens flexibel gestaltet sein und viele verschiedene Bearbeitungsmöglichkeiten anbieten, um die Ideen des Anwenders realisieren zu können. Zweitens sollte es schnell sichtbare Ergebnisse liefern, auf die der Benutzer nicht warten muss, um „trial and error“ zu ermöglichen. Jedoch besitzen die meisten guten Bildbearbeitungstools nur eine der beiden Eigenschaften, entweder sind diese flexibel und langsam, oder sie sind schnell, aber dann nur auf einige wenige Bearbeitungsmöglichkeiten beschränkt.

Ein Algorithmus der viele Bearbeitungsmöglichkeiten realisieren kann wird in dem Paper angesprochen, diesen nennt man „non-parametric patch sampling“ (NPPS). Dieser sucht

wiederholt nach ähnlichen Patches des Bildes A in Bild B. Dazu wird eine Abstandsfunktion L_p definiert, die den Patchabstand beschreibt. Diese Abbildung wird „Nearest-Neighbour Field“ (NNF) genannt. Um das NNF naiv zu bestimmen ist der Rechenaufwand O(mM²), für Bilder mit M Pixel und Patches mit m Pixel. Es gibt Ansätze die das Berechnen des NNF beschleunigen, dennoch bleibt dieser Teil sehr aufwändig und dadurch sind viele Bildbearbeitungstools langsam. Außerdem haben diese Beschleunigungsstrukturen die auf Bäumen basieren einen Speicherbedarf von O(M) mit relativ großen Konstanten. Dieses schränkt ihre Einsatzfähigkeit bei der Bildbearbeitung für größere Bilder ein.

Um das NNF effizient zu berechnen, werden für den Algorithmus der in dem Paper vorgestellt wird, folgende Betrachtungen angestellt:

- Zunächst ist zu beobachten, dass zu einem großen Patch von m Pixel wenige Übereinstimmungen zu finden sind. Andere Algorithmen nutzen Baumstrukturen (zum Beispiel der KD-Tree, der in O(mM log M) sucht) und Dimensionsreduktion, um die NNS zu beschleunigen. Der Algorithmus der in dem Paper vorgestellt wird erreicht eine höhere Geschwindigkeit mit weniger Speicherbedarf.
- Als zweiten Punkt wird die natürliche Struktur des Bildes betrachtet. Es wird keine unabhängige Suche einzelner Pixel durchgeführt, die die Ähnlichkeit von Umgebungspixeln unbeachtet lässt. Damit kann im Gegensatz zu anderen Algorithmen auch ein Performancegewinn erreicht werden.
- Als letzten Punkt wird das Gesetz der großen Zahlen genutzt. Ein Zufallspixel aus dem Bild wird sehr unwahrscheinlich ein gutes Ergebnis für den Patch liefern. Jedoch bei sehr vielen Zufallspixeln wird sehr unwahrscheinlich kein gutes Ergebnis vorliegen und somit wird ein guter Patch in den meisten Anwendungsfällen gefunden.

Mit diesen drei Betrachtungen wird nun ein Zufallsalgorithmus vorgestellt, der das NNF näherungsweise bestimmt und iterativ verbessert. Er startet indem er zufällige Abstände wählt, also ein initiales zufälliges NNF erstellt, oder er startet mit Vorinformationen. Danach folgen iterativ zwei weitere Schritte: die Ausbreitung, bei der die Umgebung des NNF nach Übereinstimmungen untersucht wird, und die Zufallssuche, die jede Iteration in einem kleineren Bereich nach guten Patches sucht. Der Algorithmus wird in Kapitel 3 theoretisch und empirisch durchgesprochen. Dabei wird gezeigt, dass er gut konvergiert, schneller ist als

bisherige Algorithmen und nur wenig mehr Speicher braucht, als für das Bild selbst benötigt wird.

2 Arbeitsumfeld

Patch basierte Methoden sind beliebt bei der Bild- und Videobearbeitung und werden von vielen Bildbearbeitungsprogrammen genutzt, um verschiedene Tools bereitzustellen. Einige von den Bearbeitungsmöglichkeiten, die solche Tools ermöglichen, werden im folgenden Text beschrieben.

2.1 Texturen-Darstellung und –Vervollständigung

Es gibt einfache, nicht-parametrisierte Textursynthese Methode die gute Ergebnisse liefern, indem sie stichprobenartig Patches zur Bildvervollständigung wählen. Weitere Fortschritte für bessere Strukturerhaltung können mit der Modifizierung der Such- und Stichprobenstrategie erreicht werden. Solche Algorithmen weisen Inkonsistenzen auf, wenn sie große Lücken mit komplexen Strukturen füllen sollen. Dieses Problem kann gelöst werden, indem man es als globale Optimierung ansieht und damit globale Konsistenz erzeugt. Dabei kann eine iterative NNS genutzt werden, um dem besten Patch für alle Pixel der Lücke zu finden. Jedoch ist dies sehr Rechenaufwendig und somit langsam. Diese und weitere Ansätze erreichen gute Ergebnisse bei der Bildbearbeitung, jedoch mit der Einschränkung, dass sie relativ langsam sind und nur kleine Bilder bearbeitet werden können.

2.2 Methoden zur Nachbarschaftssuche

Die hohe Qualität der Patch-Optimierungsmethoden benötigt viele Iterationen, die großen Rechenaufwand bedeuten. In dem Paper wird dieses Problem dadurch gelöst, indem ein Ausbreitungsschritt vorgenommen wird, welcher die Nachbarschaft der Patches kontrolliert. Zusammen mit der Zufallssuche wird damit die NNS beschleunigt.

2.3 Intuitive und interaktive Bearbeitung

Ein Bildbearbeitungsprogramm sollte eine intuitive Steuerung besitzen, damit der Anwender leicht und gezielt zu seinen Ergebnissen kommt. Dies wird durch vielfältige, übersichtliche

und skalierbare Bearbeitungstools realisiert, welche viele verschiedene Probleme lösen. Zum Beispiel sollten in einem Bild Bereiche markiert werden können, die zur Auffüllung von Lücken vom Programm genutzt werden sollen. Bestimmte Bereiche des Bildes sollten auch von Veränderungen geschützt werden können, oder aber Strukturen, wie Linien oder Zusammenhänge sollten erhalten bleiben, wenn sie dementsprechend markiert sind. Dabei muss das Programm schnell sichtbare Ergebnisse liefern, damit der Anwender nicht zu lange warten muss, um zu sehen, ob ihm das Resultat gefällt. Das „trial and error“ Prinzip sollte dem Anwender die Möglichkeit geben, schnell mit wenigen Änderungen verschiedenen Ergebnisse zu erzeugen und damit viel Erfahrungen mit dem Umgang der einzelnen Tools des Programms zu gewinnen.

2.4 Inhaltssensible Bildskalierung

Eine andere Möglichkeit Bilder zu manipulieren ist die inhaltssensible Bildskalierung. Dabei wird das Bild skaliert und die Struktur bestimmter Objekte sollen dabei erhalten bleiben und nicht verzerrt werden, siehe Bild 2.

Bild 2: Beispiel der inhaltssensiblen Bildskalierung

2.5 Neuanordnung von Bildinhalten

Neuanordnung von Bildinhalten bedeutet, dass die Bildinhalte verschoben, kopiert oder neu angeordnet werden, um zum Beispiel eine Person ins Zentrum des Bildes zu verschieben oder aber ein Objekt zu vervielfältigen.

3 Nachbarschaftssuche mittels Approximate Nearest Neighbour Algorithm (ANNA)

ANNA stellt den Kern der Arbeit dar. Mit diesem wird erreicht, dass gute Patches berechnet werden können.

Ein NNF wird definiert als Funktion $f : A \mapsto \mathbb{R}^2$. Diese beschreibt die Abstände aller möglichen Patch Koordinaten (Mittelpunkt des Patches) eines Bildes. Als Beispiel sei eine Patch Koordinate a von Bild A und der zugehörende nächste Nachbar b aus Bild B gegeben. Dann ist f(a) gleich b-a. Die Werte von f stellen die Abstände dar (und werden in einem Array gespeichert).

Dieses Kapitel beschreibt einen Zufallsalgorithmus zum Bestimmen des approximierten NNF. Dieser Schritt kommt von der Überlegung, dass in der Umgebung der möglichen Patches gute Resultate erzielt werden können.

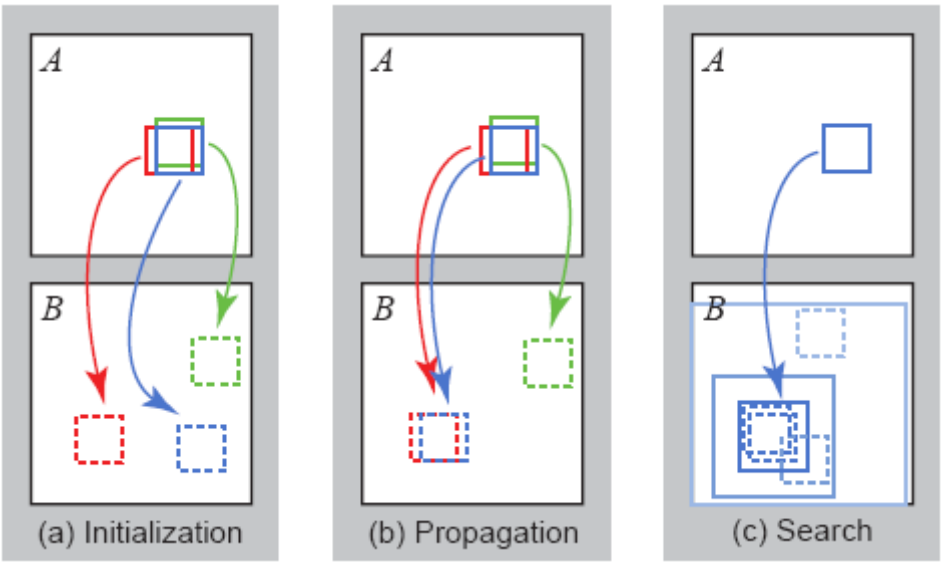

Bild 3: Phasen des Algorithmus

Der Algorithmus besteht aus drei Schritten, wie in Bild 3 dargestellt ist. Im ersten Schritt wird das NNF initialisiert mit Zufallsabständen oder mit Vorinformationen, die das Bildbearbeitungstool dem Benutzer ermöglicht. Danach folgt eine iterative Abfolge der beiden weiteren Schritte. Die Ausbreitung, die angrenzende Pixel betrachtet und die Zufallssuche, um in der kleineren Umgebung neue Abstände zu finden.

3.1 Initialisierung

Initialisiert wird das NNF durch Zufallswerte oder durch Vorinformationen. Bei der Initialisierung mit Zufallswerten werden alle Abstände im Bild B als möglich angenommen. Eine schrittweise Verfeinerung der Initialisierung kann die Qualität des Algorithmus verbessern. Dabei können aber lokale Minima auftreten, in denen sich der Algorithmus verlaufen kann. Um dies zu vermeiden wird vorher eine Zufallsinitialisierung durchgeführt und damit werden schlechte Patches ausgeschlossen.

3.2 Iteration über die Ausbreitung und Zufallssuche

Nach der Initialisierung wird das NNF iterativ verbessert. Jede Iteration des Algorithmus vergleicht die Abstände für die Ausbreitung P_j des Patches und führt eine Zufallssuche S_j durch. Damit laufen die Iterationen schematisch wie folgt ab: $P_1, S_1, P_2, S_2, ..., P_n, S_n$.

Die Ausbreitung wird durchgeführt, indem die Abstandsfunktion $f(x, y)$ an den bekannten Abständen $f(x-1, y)$ und $f(x, y-1)$ verbessert wird. Dazu wird angenommen, dass die Patch-Abstände in der Umgebung ähnlich sind. Wenn zum Beispiel an der Stelle (x-1,y) eine gute Übereinstimmung für die Stelle (x,y) ist, kann dieser für den Patch genutzt werden. Weiterhin werden alle Stellen der gegebenen Abstände verglichen und derjenige als Patch gewählt, der das beste Ergebnis aufweist. Als Resultat bekommt man einen guten Patch aus der untersuchten Umgebung R für die Stelle (x,y).

Um neue Abstände zu bekommen wird eine Zufallssuche genutzt, die in einer Umgebung zufällige Abstände ausgibt. Dabei ist die Umgebung beschränkt durch die Funktion $u_i = v_0 + \omega\alpha^i R_i$, mit $v_0 = f(x, y)$, also der Stelle die bearbeitet wird. Die Anzahl der Abstände ist i, also die möglichen Patches. α ist das Verhältnis der Suchfenstergrößen und ω ist die maximale Bildgröße. Weiterhin muss beachtet werden, dass der Suchbereich innerhalb des Bildes, in dem gesucht wird, liegen muss.

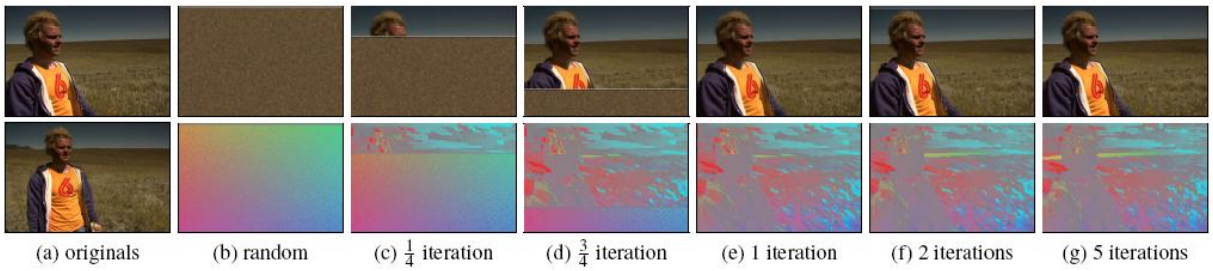

Bild 4: Veranschaulichung der Konvergenz des Algorithmus in 5 Iterationen an einem Beispiel. Das obere Bild ist durch Patches aus dem unteren Bild rekonstruiert. Die unteren Bilder b bis g stellen den Einfluss als Sättigung und den Betrachtungswinkel als Farbton dar.

Normalerweise werden verschiedene Abbruchkriterien für verschiedene Probleme genutzt. Jedoch hat sich in der Anwendung des Algorithmus herausgestellt, dass dieser auch sehr gut mit einer festen Iterationszahl auskommt. Dabei wird eine gute Konvergenz in etwa fünf Iterationen erreicht wie in Bild 4 zu sehen.

3.3 Analyse des Algorithmus

Die Berechnung des NNF konvergiert mit hoher Wahrscheinlichkeit in wenigen Iterationen. Der Algorithmus wurde analysiert für die exakte Berechnung des NNF und danach für die Berechnung von näherungsweisen Ergebnissen, die für normale Bildbearbeitungsprogramme ausreicht.

Dabei wurden verschiedene Testfälle betrachtet, unter anderem auch das Bild 5A, aus welchem das Bild 5B rekonstruiert werden sollte. Hierbei ist das Problem, das der weiße Hintergrund kaum Informationen von Übergängen enthält, was die Rekonstruktion erschwert. Es hat sich herausgestellt, dass selbst bei Testfällen die nicht optimal für den Algorithmus sind gute Ergebnisse erzielt werden. Dabei wird mit hoher Wahrscheinlichkeit eine gute Konvergenz nach fünf Iterationen erreicht, bei Bildern mit einer Auflösung von bis zu $M = 2000^2$ und Patches der Größe $m = 20^2$.

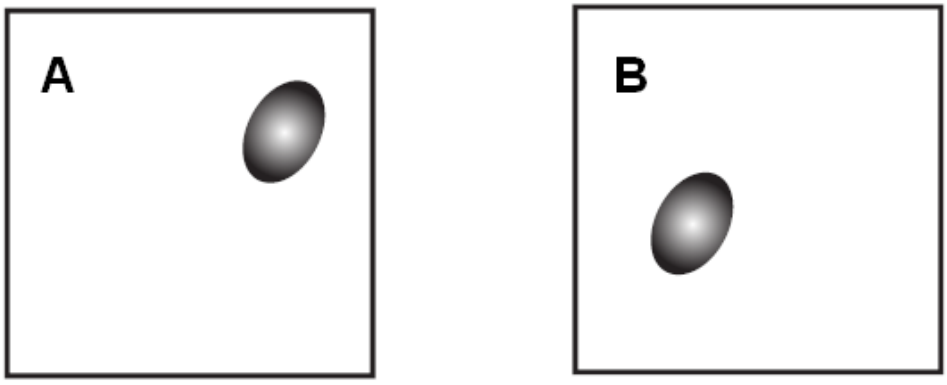

Bild 5: Testfall mit großem weißanteil

	Time [s]		Memory [MB]	
Megapixels	Ours	*k*d-tree	Ours	*k*d-tree
0.1	0.68	15.2	1.7	33.9
0.2	1.54	37.2	3.4	68.9
0.35	2.65	87.7	5.6	118.3

Tabelle 1: Geschwindigkeit und Speicherbedarf im Vergleich von dem vorgestellten mit einem normalerweise genutzten Algorithmus, angewendet auf das Bild Figure4

Wenn nun der Algorithmus an repräsentative Bilder (Natur, Gebäude, Personen, etc.) angewendet wird, dann zeigen sich sogar noch bessere Ergebnisse. In Tabelle 1 ist ein Vergleich des Algorithmus bei typischer Patchgröße von 7x7 Pixel dargestellt. Hier wird deutlich dass eine 20- bis 100-fache Beschleunigung erreicht wird und 20-mal weniger Speicher benötigt wird. Dieses Resultat ist repräsentativ für alle durchgeführten Bildbearbeitungen und dabei wurden nur fünf Iterationen benötigt.

4 Bildbearbeitungstools

In diesem Kapitel werden einige Tools angesprochen, die durch den Algorithmus ermöglicht werden und welche Vorteile gegenüber anderen Realisierungen bestehen.

Die Bildrekonstruktion in fehlenden Bereichen ermöglicht es Lücken in Bildern zu füllen und dabei Zusammenhänge zu erzeugen, so dass die Ausgabe ein gutes Resultat aufweist. Dazu können Randbedingungen gesetzt werden, um Strukturen zu rekonstruieren, wie zum Beispiel Linien die mit zwei Enden außerhalb der Lücke verbunden werden sollen. Es können aber auch Bereiche vorgegeben werden, die zum Auffüllen der Lücke genutzt werden sollen. Das Einführen der Randbedingungen schafft somit eine sehr gute und intuitive Bedienung. Gesteuert wird dies alles mit ziehen von Linien und markieren von Bereichen.

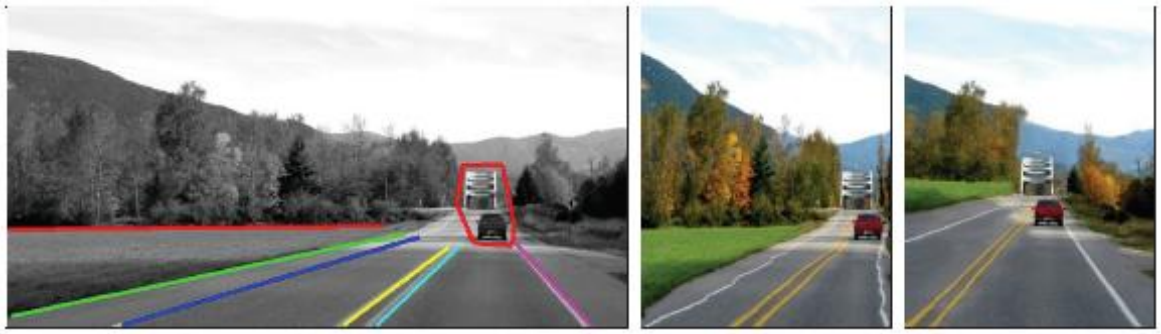

Bild 6: Skalierung der Breite eines Bildes, mit Verzerrung und ohne Verzerrung von Linien

Weiterhin wird mit ähnlichen Randbedingungen die Neuanordnung einzelner Bildobjekte realisiert, dazu werden diese umrahmt und per „drag and drop“ verschoben. Dieses kann auch bei der inhaltssensiblen Bildskalierung und der Deformation genutzt werden, um Bereiche festzulegen die nicht verändert werden sollen. Dabei können auch wieder Strukturen markiert werden, die erhalten bleiben sollen. Eine wichtige Rolle sind dabei Linien, denn wenn eine Linie verzerrt wird stört das sofort den Betrachter des Bildes. Wenn sich Linien jedoch nur in ihrer Länge oder ein wenig in der Breite verändern ist noch ein gutes Resultat vorhanden (siehe Bild 6).

5 Ergebnis und Ausblick

In der Arbeit wird eine neue NNS vorgestellt mit dem inhaltssensible Skalierung, Vervollständigung und Neuanordnung von Elementen an Bildern vorgenommen werden kann. Dieser wird verglichen auf die Qualität und die Performance im Gegensatz zu anderen Methoden mit denen gleiche Bildmanipulationen vorgenommen werden können.

Bild 7: Eingabebild, Bild mit Randbedingungen, Ausgabebild

Die Möglichkeiten der Bildmanipulationen sieht man im Bild 1, in dem ein alter griechischer Tempel restauriert und die Anzahl und Länge der Säulen verändert wird. Neben dem verändern von Gebäuden, können auch Personen in Bilder verändert werden (siehe Bild 7). Auch Pflanzen können so manipuliert werden, dass das Ausgabebild wie eine Originalaufnahme erscheint (siehe Bild 8).

Bild 8: Vergrößerung eines Baumes in der Nähe und des dahinterliegenden, entfernten Waldes.

In einigen Fällen liefert der Algorithmus schlechte Ergebnisse, zu erhaltende Strukturen werden zerstört, schlechte Patches werden gefunden und ungewollte Artefakte entstehen. Diese Probleme stören den Anwender aber wenig, weil der Algorithmus effizient ist und ein Schritt dadurch schnell ausgeführt und rückgängig gemacht werden kann. In einem erneuten Versuch kann der Algorithmus mit neuen Randbedingungen ausgeführt werden und somit wird ein anderes Ergebnis erzielt.

Um den Algorithmus zu beschleunigen, kann dieser als GPU-Implementierung realisiert werden. Zusätzlich könnte er genutzt werden, um die Manipulationen auch an Videos vorzunehmen. Eine andere Idee ist den Algorithmus in CAD Programmen zu nutzen, um zum Beispiel Objekte neu anzuordnen oder Lücken zu füllen.

Literaturverzeichnis

Connelly Barnes, Eli Shechtmann, Adam Finkelstein, Dan B Goldman. 2009. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. ACM Transactions on Graphics (Proc. SIGGRAPH)

Fang, H., Hart, J. C. 2007. Detail preserving shape deformation in image editing. ACM Transactions on Graphics 26, 3, 12.

Rubinstein, M., Shamir, A., Avidan, S. 2008. Improved seam carving for video retargeting. ACM SIGGRAPH 27, 3.

Princeton, Publications. URL http://www.cs.princeton.edu/gfx/pubs/ Barnes_2009_PAR/index.php. – Okt 2009