Automatisierte Graphikbearbeitung von Wasserzeichen und Anbindung an bestehende Metadatenbank

Bachelor-Thesis

Julius-Maximilians-Universität Würzburg
Institut für deutsche Philologie
Studienfach: Digital Humanities

vorgelegt von:
Markus Bald

Matrikelnummer:
1808222

Würzburg, 2015

Inhaltsverzeichnis

1. Einleitung 3
2. Konkrete Zielsetzung 5
3. Grundlagen 6
   3.1 Faksimiles und Vorlagengraphiken 6
   3.1 Existierende Metadatenbank 7
   3.2 Bestehende Anwendung mit HTML5-Canvas und WebGL 9
4. Automatisierte Graphikbearbeitung von Wasserzeichen 11
   4.1 Möglichkeiten zur automatisierten Herausfilterung 11
   4.2 Schwierigkeiten bei der Umsetzung 15
5. Anbindung an eine existierende Metadatenbank 17
   5.1 Einbindung der Faksimiles 17
   5.2 Einbindung der Vorlagengraphiken 20
   5.3 Metadatenübersicht 23
   5.4 Probleme bei der Umsetzung 25
6. Zusammenfassung und Ausblick 27
7. Literaturverzeichnis 30

1. Einleitung

In einer Villa, die im Zuge von Ausgrabungen in Herculaneum im Jahre 1752 aufgedeckt wurde, gelang die Entdeckung geschwärzter Rollen, welche Klumpen aus Kohle ähnlich sahen und deshalb zunächst auch mit solchen verwechselt wurden. Erst nachdem viele von ihnen in Verkennung ihres Wertes bereits zerstört worden waren, wurde bei den restlichen erkannt, dass es sich hierbei um Papyrus-Schriftrollen handelte, die anschließend sorgfältig konserviert wurden.[1] Als die Rollen nach verschiedenen Versuchen erstmalig erfolgreich geöffnet werden konnten, wurden Faksimiles durch Abzeichnungen angefertigt. Meist wurden hierbei auch Wasserzeichen hinzugefügt. Seit ihrer Einführung im dreizehnten Jahrhundert sind Wasserzeichen als Methode verwendet worden, um die Herkunft festzulegen, Markenzeichen und Orte zu identifizieren, und die Größe und beabsichtigte Funktion von Papieren zu bestimmen.[2] Im Zuge der Digitalisierung stellte die Universität von Oxford die über viele Jahre hinweg entstandenen Faksimiles in einer indexierten Bilderdatenbank online zur Verfügung.

An eine Möglichkeit zur digitalen Weiterverarbeitung der Abzeichnungen mit dem Zugang über das Internet soll diese Arbeit anknüpfen. Sie wird konkret die Frage behandeln, wie mit Hilfe von automatisierter Graphikbearbeitung im Web-Bereich die Wasserzeichen der Faksimiles herausgefiltert und mit entsprechenden Vorlagen überlagert werden können. Hierzu soll das Programm an eine existierende Metadatenbank angebunden werden. Besonders diese Anbindung an eine Metadatenbank stellt dabei die Bedeutung eines spezifisch auf die Graphikbearbeitung von Wasserzeichen ausgerichteten Programmes heraus, denn mit Hilfe von professionellen Bildbearbeitungsprogrammen können die Abzeichnungen mit einer Vielzahl von Werkzeugen um einiges umfangreicher und durch Stapelverarbeitung auch automatisiert bearbeitet werden. Die bestehende Datenbank ermöglicht als Koordinationsmöglichkeit von Daten ein präzises Laden der Faksimiles und Vorlagengraphiken. Im Rahmen einer vorausgegangenen Projektarbeit habe ich die Datenbank mittels Konvertierung von Microsoft Access in MySQL in eine Weboberfläche überführt.

[1] Vgl. Walter Scott: Fragmenta herculanensia; a descriptive catalogue of the Oxford copies of the Herculanean rolls together with the texts of several papyri accompanied by facsimiles, Oxford 1885, S. 1.
[2] Vgl. Duffy, Christina: The Discovery of a Watermark on the St Cuthbert Gospel using Colour Space Analysis, online: http://www.bl.uk/eblj/2014articles/pdf/ebljarticle22014.pdf. (02.11.2015)

Im Folgenden soll zunächst das Vorgehen in einer konkreten Zielsetzung zusammengefasst werden. Anschließend werden die Grundlagen eines spezifisch auf die Herausfilterung von Wasserzeichen ausgerichteten Graphikbearbeitungsprogrammes geschildert. Sie bestehen neben den bereits thematisierten Abzeichnungen, Vorlagengraphiken und der Metadatenbank auch aus der Überschneidung der beiden HTML5-Erweiterungen Canvas und WebGL in einer vorhandenen Anwendung. Auf dieser Basis aufbauend werden danach sowohl die Möglichkeiten, als auch die Umsetzung einer automatisierten Graphikbearbeitung verdeutlicht. Komplementär dazu werden auch die Schwierigkeiten aufgezeigt. Auf diese Art und Weise werde ich anschließend auch bei der Anbindung der Metadatenbank verfahren. Dabei sollen die Möglichkeit der Einbindung einer bestehenden Vorlage wie auch das gegebenenfalls erforderliche Erstellen einer neuen Vorlage präzisiert werden, bevor die Arbeit mit einer Zusammenfassung und einem diskutierenden Ausblick darauf schließt, welche Umsetzungen weiterführend anwendbar wären.

2. Konkrete Zielsetzung

Für eine Übersicht über das Vorgehen soll an dieser Stelle nun zunächst eine konkrete Zielsetzung der Arbeit formuliert werden. Wie bereits in der Einleitung dargelegt, soll sie einerseits eine Umsetzung zur automatisierten Graphikbearbeitung von Wasserzeichen im Web-Bereich beschreiben. Dazu werden die clientseitige Programmiersprache JavaScript und dessen Bibliothek jQuery verwendet. Als Grundlagen werden hierbei die Graphiken der Abzeichnungen sowie eine Anwendung eingebunden, welche die HTML5-Erweiterungen Canvas und WebGL miteinander kombiniert. Diese werden im Kapitel 3.3 noch konkreter beschrieben. In der bestehenden Anwendung wurden bereits Funktionen zu Graphikeffekten implementiert, die anschließend zu einer automatisierten Funktion verknüpft werden sollen. Die relevanten Filter und die Auswahl für eine automatisierte Graphikbearbeitung werden im Kapitel 4.1 veranschaulicht.

Andererseits soll diese Arbeit jedoch auch aufzeigen, wie eine Anbindung an eine existierende Metadatenbank realisiert werden kann, da diese von großer Bedeutung für die Einbindung von Graphiken ist. Als Grundlagen werden hier dementsprechend wiederum die Bilder der Abzeichnungen, aber auch die Vorlagengraphiken der Wasserzeichen und die existierende Metadatenbank benötigt, auf welche in Kapitel 3.2 konkret Bezug genommen wird. Mithilfe der Anbindung sollen die entsprechenden Vorlagen der Faksimiles automatisch zur Verfügung gestellt und über der Abzeichnung positioniert werden können. Sollten keine Vorlagen gefunden werden, soll dem Nutzer allerdings auch eine Möglichkeit geboten werden, diese auf dem bearbeiteten Bild zu zeichnen und sein Ergebnis als neue Vorlage abzuspeichern. Außerdem soll eine Übersicht über weitere Metadaten angezeigt werden können.

3. Grundlagen

Wie in der Zielsetzung erwähnt, werden nun die Grundlagen, welche der Arbeit zugrunde liegen, verdeutlicht. Zunächst wird dazu der Fokus auf die Graphiken der Faksimiles und Vorlagen gelegt. Wie sind die Abzeichnungen in der Oxforder Bilderdatenbank indexiert und wie sind die Dateinamen aufgebaut? Dies ist eine wichtige Erkenntnis, um die Graphiken mithilfe der ebenfalls grundlegenden Metadatenbank gezielt in das Programm zu laden. Hier müssen Übereinstimmungen zwischen den Einträgen der Datenbank und den Dateinamen gefunden oder hergestellt werden. Schließlich wird auch eine bereits vorhandene Anwendung thematisiert, welche die HTML5-Erweiterungen Canvas und WebGL miteinander kombiniert. Dabei sollen sowohl die Anwendung, als auch die Erweiterungen vorgestellt werden.

3.1 Faksimiles und Vorlagengraphiken

Wie bereits im einleitenden Kapitel erwähnt, hat die Universität von Oxford eine indexierte Bilderdatenbank der Faksimiles zur Verfügung gestellt, welche von den Papyri erstellt wurden. Relevant ist hier die Untergliederung in die Felder Autoren, Titel, Datumsangaben, Genres, Papyrusnummern, abgekürzt auch „PHerc-Nr.“, und Bände. Denn für die präzise Einbindung von Graphiken ist die Auswahl eines eindeutigen Schlüssels wichtig, damit jedes Bild gezielt ausgewählt und geladen werden kann. Die hier verwendeten PHerc-Nummern mit ihren entsprechenden Fragmentbezeichnungen und die Bände inklusive ihrer jeweiligen Blattnummern sind für diesen Zweck optimal, denn diese Indizes enthalten bis auf geringfügige Ausnahmen, wenn bei den Titelblättern mehrfach keine Fragmentbezeichnung notiert wurde, ausschließlich unverwechselbare Einträge.

Von großer Bedeutung sind zudem die Dateinamen, denn mit diesen werden die Graphiken später in das Programm eingebunden. Dabei wurde in der Oxforder Datenbank durchgehend eine Bezeichnung wie die beispielsweise folgende gewählt: „herc.v001.0004.a.01.hires“. Mit „v001“ wird der erste von sieben Bänden benannt, die darauffolgende Zahl nach dem trennenden Punkt bis zum folgenden Punkt ist die Blattnummer. Eine PHerc-Nummer oder Fragmentbezeichnung ist im Dateinamen zwar nicht

zu erkennen, allerdings kann diese über einen Vergleich mit den Bänden und Blattnummern, die in der existierenden Metadatenbank notiert sind, ebenfalls zugeordnet werden. Diese Möglichkeit wird im folgenden Unterkapitel 3.2 konkretisiert.

Die insgesamt 198 Vorlagengraphiken wurden in zwei unterschiedliche Fassungen gegliedert. Dabei existiert eine ältere neben einer neueren Version, die jedoch beide für die Einbindung der Vorlagen von Bedeutung sind, denn die neueren Graphiken, bei denen die Bezeichnung gegenüber der älteren Fassung vereinheitlicht und bereinigt wurde, besitzt noch nicht den vollständigen Umfang der älteren Bilder. Zudem gilt zu beachten, dass die Fassungen unterschiedliche Konventionen besitzen: Bei den neueren ist der Typ, welcher der Abkürzung „WZ“ und einem Unterstrich folgt, von der Folgenummer durch einen weiteren Unterstrich abgesetzt, bei den älteren ist der Typ mit der Folgenummer direkt verknüpft worden.

Von Relevanz ist hier auch der Dateityp, denn das bei den Vorlagen verwendete JPEG-Format unterstützt keinen Alpha-Farbkanal. Dieser wird jedoch zwingend zur Darstellung eines transparenten Hintergrundes benötigt, der bei der Überlagerung des bearbeiteten Faksimilebildes wichtig ist, um die Vorlage mit dem bearbeiteten Bild zu vergleichen. Diese Problematik wird noch ausführlicher bei den Schwierigkeiten der Einbindung von Graphiken in Kapitel 5.3 behandelt.

3.2 Existierende Metadatenbank

Im vorherigen Abschnitt wurde bereits deutlich, dass die existierende Metadatenbank eine wichtige Grundlage bei der Einbindung der unterschiedlichen Graphiken ist, denn mit ihrer Hilfe können die Bilder, die stets über ihre Dateinamen identifiziert werden, mit weiteren Indizes verknüpft werden, sofern diese eindeutig sind. Bei den Faksimiles ist dies beispielsweise möglich, da die Dateinamen die eindeutigen Blattnummern enthalten, die sie abbilden, denn in der existierenden Metadatenbank sind die Blattnummern ebenfalls eingetragen und inhaltlich mit entsprechenden weiteren Daten verknüpft. Dies sind unter anderem die Bandbezeichnung, welche jedoch auch bereits im Dateinamen enthalten ist, die Blattart, welche aber bei den Faksimiles nur aus Abzeichnungen besteht, die PHerc-Nummer, welche ebenfalls zur Auswahl der Graphiken verwendet werden kann, da sie in Kombination mit einer Fragmentbezeichnung meist eindeutig identifizierbar ist, aber auch weitere wichtige Daten, die als Metadatenübersicht angezeigt

werden können, wie den Zeitraum der Bearbeitung, den Namen des zuständigen Angestellten oder den Autor und Titel der referenzierenden Edition.

Leider nicht aufgeführt sind die Bezeichnungen der Wasserzeichen, welche die Einbindung entsprechender Vorlagen ermöglichen. Hierfür bestünde die Möglichkeit, die Tabelle zu erweitern. Da eine solche jedoch bereits in einem Microsoft-Excel-Format existiert, lässt sich diese mit Excel zunächst in eine Microsoft-Access-Tabelle überführen und anschließend mit einem Open-Source-Konvertierungsprogramm namens Bullzip von Access nach MySQL konvertieren. Die dabei entstehende Datei beinhaltet eine Reihe von MySQL-Anweisungen, die in einem Server-Datenbankprogramm wie phpMyAdmin ausgeführt werden können, um eine neue Tabelle anzulegen, die der ursprünglichen Excel-Tabelle entspricht. Ist auf dem Server eine Tabelle eingerichtet, kann diese in einer Web-Anwendung mithilfe einer Programmiersprache wie PHP abgefragt werden. Dieses Vorgehen wird in Kapitel 5.1 konkret beleuchtet.

Die neue Tabelle enthält nun neben den Wasserzeichen-Nummern auch Angaben zum Band und zur Blattnummer. Somit muss sie nicht über einen Primärschlüssel mit den bereits bestehenden Metadaten verbunden werden, da diese Daten in den Dateinamen der Abzeichnungen enthalten sind. Daher kann eine Faksimilegraphik mithilfe der neuen Tabelle nun mit entsprechenden Vorlagenbildern verknüpft werden. Das Hinzufügen einer automatisch mitzählenden ID ermöglicht außerdem den eindeutigen Zugriff auf einzelne Zeilen, um diese gegebenenfalls zu bearbeiten. Die Tabelle, die bereits bestand, wird trotz der neu erstellten Tabelle weiterhin für die Umsetzung benötigt, um dem Nutzer die Möglichkeiten zum Einladen von Faksimiles anzubieten, denn in der neuen Tabelle sind nicht alle Faksimiles notiert. Zudem fehlen für die eindeutige Benennung der PHerc-Nummern, die in der neu erstellten Tabelle ebenfalls aufgeführt werden, die Fragmentbezeichnungen.

Bei vergleichender Betrachtung der Wasserzeichen-Nummern mit den Dateinamen der Vorlagen fällt nun auf, dass eine Abzeichnung, wie beispielsweise diejenige mit der Bandbezeichnung „MS. Gr. class. c. 6“ und der Blattnummer 1315, mehrere Vorlagen besitzen kann. Daher entspricht die Wasserzeichen-Nummer nicht immer den Vorlagen-Dateinamen, die jeweils eindeutig sind. Im Falle mehrerer Möglichkeiten wurde an die Wasserzeichen-Nummer eine weitere Nummer angehängt, die auf eine Blattnummer verweist. Um alle Vorlagen einbinden zu können, müssen daher alle Dateinamen in einer weiteren Tabelle aufgeführt werden. In dieser werden zwei Felder gebildet. Eine automatisch mitzählende ID ermöglicht wiederum den Zugriff auf das jeweilige Feld, falls

dieses bearbeitet werden muss. Die Dateinamen werden dann in das zweite Feld eingefügt. Damit sind nun alle Metadaten notiert, um das Programm nach der Zielsetzung zu erstellen.

3.3 Bestehende Anwendung mit HTML5-Canvas und WebGL

Nachdem die Faksimile- und Vorlagengraphiken bereits theoretisch in das Programm eingebunden werden können, sollen die Bilder der Abzeichnungen auch bearbeitet werden können. Die Grundlage hierfür bietet ein HTML5-Element mit dem Namen Canvas. Denn dieses bietet eine Reihe von Funktionen, um mithilfe von JavaScript clientseitig beispielsweise Graphen, Spielgraphiken, geometrische Objekte oder Bilder zu erstellen.[3] Letztere können allerdings auch über eine Quelle eingefügt und angepasst werden. Die Objekte, Methoden und Rechte zum Zeichnen und Bearbeiten von Graphiken werden von einem „2D Kontext“ zur Verfügung gestellt, der die Canvas-Oberfläche abbildet.[4] Mit diesem Kontext wird somit beispielsweise auch das Zeichnen neuer Vorlagen möglich, sollte eine der jeweiligen Abzeichnung entsprechende Vorlage noch nicht vorhanden sein.

Eine Erweiterung des HTML5-Canvas-Elementes stellt WebGL dar. Das wird in der Einleitung aus der Übersicht zu den Spezifikationen deutlich, die die Organisation Khronos, welche WebGL entwickelt hat, präsentiert:

> „WebGL™ is an immediate mode 3D rendering API designed for the web. It is derived from OpenGL® ES 2.0, and provides similar rendering functionality, but in an HTML context. WebGL is designed as a rendering context for the HTML Canvas element. The HTML Canvas provides a destination for programmatic rendering in web pages, and allows for performing that rendering using different rendering APIs. The only such interface described as part of the Canvas specification is the 2D canvas rendering context, CanvasRenderingContext2D. This document describes another such interface, WebGLRenderingContext, which presents the WebGL API. “[5]

[3] http://www.w3.org/wiki/HTML/Elements/canvas. (02.11.2015)
[4] http://www.w3.org/TR/2dcontext. (02.11.2015)
[5] https://www.khronos.org/registry/webgl/specs/1.0/#1. (02.11.2015)

Wie hier beschrieben wird, lässt sich WebGL auf das HTML5-Canvas-Element anwenden, indem ein anderer Kontext verwendet wird, der 3D-Darstellungen ermöglicht.

Eine Open-Source-Bibliothek für Graphikeffekte mit WebGL namens „glfx.js“ wurde im Jahr 2011 vom Entwickler Evan Wallace entwickelt. Diese wendete er in einem Bildbearbeitungsprogramm namens „WebGL-Filter“ an, welches er innerhalb von 24 Stunden für einen Wettbewerb kreierte. Hier wird deutlich, dass sich der WebGL-Kontext auch auf 2D-Graphiken anwenden lässt. Das Programm hat keine Lizenzbestimmungen, die eine Weiterbearbeitung verbieten oder einschränken. Darüber hinaus besitzt die Anwendung bereits einige Möglichkeiten, die für die Herausfilterung von Wasserzeichen geeignet sind. Hierauf wird das Kapitel 4.1 detailliert eingehen. Aus dem Programmcode geht allerdings hervor, dass Funktionen aufgrund der zeitlich eingeschränkten Entwicklung nicht realisiert wurden, denn zwei HTML-Div-Elemente, durch die Buttons dargestellt werden sollten, wurden durch Kommentare ausgeklammert und ihr Class-Attribut mit „button disabled“ bezeichnet. Hier hatte der Entwickler dem inneren HTML zwischen dem öffnenden und schließenden Tag zufolge wohl die Idee, eine Möglichkeit einzubinden, Schritte bei der Bearbeitung rückgängig zu machen oder wiederherzustellen. Eine solche Umsetzung ließe sich beispielsweise mit der Bearbeitung über verschiedene Ebenen realisieren, die nach jedem Bearbeitungsschritt erstellt werden und beliebig ein- und ausgeblendet werden könnten. Darauf wird das Kapitel 6 noch konkreter eingehen.

Obwohl die Umsetzung des Effektprogramms von Evan Wallace in sehr kurzer Zeit entwickelt wurde, bietet sie durch die bereits vorhandenen Anpassungsgelegenheiten eine geeignete Grundlage, die noch nicht optimal ausgereift ist, jedoch genügend Potenzial besitzt, um sie für individuelle Zwecke weiterzuentwickeln.

4. Automatisierte Graphikbearbeitung von Wasserzeichen

Nachdem alle grundlegenden Voraussetzungen beschrieben wurden, soll sich dieses Kapitel darauf aufbauend der automatisierten Graphikbearbeitung widmen. In das Bildeffektprogramm von Evan Wallace, welches der Umsetzung als Grundlage dienen soll, sind bereits einige Anpassungsmöglichkeiten integriert, die sich für die Herausfilterung von Wasserzeichen eignen. Dieses Kapitel soll dementsprechend neben der Implementierung auch auf die bestehenden Filter und ihre Kombinationsmöglichkeiten eingehen, um auf dieser Basis eine optimierte Automatisierung zu generieren. Dabei wird eine Auswahl von Filtern, der Reihenfolge ihres Aufrufs und der Einstellung ihrer Parameter getroffen. Abschließend wird dieses Kapitel auch aufzeigen, welche Probleme sich bei der Umsetzung einer automatischen Filterung ergeben können.

4.1 Möglichkeiten zur automatisierten Herausfilterung

In der Summe bietet das Programm „WebGL-Filter“ von Evan Wallace 17 verschiedene Graphikeffekte. Seine Bildeffektbibliothek glfx.js umfasst darüber hinaus noch weitere Filterfunktionen. Um nun die für eine Herausfilterung von Wasserzeichen relevanten Anpassungsmöglichkeiten herauszufinden, bietet die Anwendung einerseits die Möglichkeit, ein Faksimilebild vom lokalen System aus hochzuladen und die Werkzeuge zu testen. Andererseits wird aus den Kategorien der Seitenleiste „Blur“ (verwischen), „Warp“ (verzerren) und „Fun“ (Spaß) bereits deutlich, dass diese Filter für die Herausfilterung von Wasserzeichen ungeeignet sind, denn dessen Strukturen sollen geschärft und verdeutlicht werden. Die dafür relevanten Einstellungsmöglichkeiten sind in der ersten Kategorie „Adjust“ (Anpassen) zu finden. Hier können sowohl Helligkeit und Kontrast, als auch Farbton und Sättigung, Gradationskurven und Scharfzeichnung definiert werden. „Denoise“, ein Unschärfefilter, wird hingegen wie die weiteren Anpassungsmöglichkeiten aus der Bildeffektbibliothek nicht benötigt. Die vier zuvor erwähnten, relevanten Filter, insbesondere die Gradationskurven und der Scharfzeichner, sind im Hinblick auf den beabsichtigten Zweck am effektivsten und können ein zielführendes Ergebnis generieren.

Der Entwickler Evan Wallace geht in einer Dokumentation noch präziser auf die Wirkungsweise seiner Filterfunktionen ein. Er beschreibt, dass das Programm den ausgewählten Helligkeitswert zur Farbe hinzuaddiert, während die Kontrastierung multiplikativ erfolgt.[6] Bei den Gradationskurven gibt es zwei Ausprägungen. Wird der Kurvenfunktion nur ein Argument übergeben, wie dies in seiner Version des Programms der Fall ist, dann lässt sich damit die Belichtung der Graphik regulieren. Werden der Funktion drei Argumente übergeben, jeweils einen für den roten, grünen und blauen Farbkanal, dann lassen sich die Farbkanäle anpassen. Der Farbton wird rotierend und die Sättigung multiplikativ bestimmt. Er beschreibt die Rotation des RGB-Farbraums als einen drehbaren Würfel, der als Achsen rote, grüne und blaue Farbwerte besitzt. Wird dieser um die von schwarz bis weiß reichende Linie gedreht, welche ihn durchzieht, ändert sich der Farbton, welcher anschließend mit der Sättigung multipliziert wird. Die Scharfzeichner-Maske verstärkt hohe Frequenzen im Bild, um die Schärfe zu erhöhen.[7] Vereinfacht gesagt sind die hohen Frequenzen für Kanten und harte Übergänge verantwortlich, während die niedrigeren Frequenzen die eher weiteren Verläufe übernehmen.[8] Wieso eine Graphik in Frequenzen unterteilt ist, wird der folgende Absatz erläutern. Beim Scharfzeichner werden Durchschnittswerte von benachbarten Pixeln in einem definierbaren Radius berechnet, welche anschließend dazu verwendet werden, um die Differenz von Pixelwerten zu diesem Durchschnitt mit einem vom Nutzer einstellbaren Stärkeparameter zu erhöhen.[9]

Entscheidende Faktoren bei der Graphikbearbeitung zur Herausfilterung von Wasserzeichen sind demzufolge Kontraste, denn durch die Multiplikationen werden die Unterschiede zwischen den Farbwerten um ein Vielfaches deutlicher. Aber auch die Belichtung, welche in diesem Fall durch Gradationskurven beeinflusst werden kann, generiert ein zielführendes Ergebnis. Ein vereinfachtes Beispiel hierfür ist ein Geldschein, welcher ebenso wie die Faksimiles ein Wasserzeichen besitzt. Wird dieser gegen die Sonne gehalten, wird das Wasserzeichen durch das hindurchscheinende Licht und seine Transparenz sichtbar. Ein weiterer Aspekt sind Graphikfrequenzen, denn bei der Digitalisierung wird eine kontinuierliche, reale Graphik in eine Vielzahl von Pixeln gerastert. Dieser Vorgang lässt sich auch als Diskretisierung bezeichnen, da das kontinuierliche Signal in ein diskretes überführt wird. Jedem der Pixel wird nun genau ein Farbwert zugeordnet.

---

[6] Vgl. http://evanw.github.io/glfx.js/docs. (02.11.2015)
[7] Vgl. ebd. (02.11.2015)
[8] Külzer, Jakob: Filter im Frequenzraum, online: http://www.jakusys.de/t3/fileadmin/files/documents/kuelzer_filter_im_frequenzraum.pdf. (02.11.2015)
[9] Vgl. http://evanw.github.io/glfx.js/docs. (02.11.2015)

Da eine Farbgraphik nur eine begrenzte Anzahl an Farben kodiert, wird der lineare Farbverlauf, wie er in der Realität besteht, auf endlich viele Farbstufen reduziert. Dieser Vorgang wird auch als Quantisierung bezeichnet. Um den Farbwert eines Pixels zu bestimmen, erfolgt zunächst eine Abtastung des realen Bildes. Dabei werden die Bildfunktionswerte des realen Bildes in einer definierten Frequenz an bestimmten Stellen gemessen.[10]

In ihrem Aufsatz „The Discovery of a Watermark on the St Cuthbert Gospel using Colour Space Analysis" beschreibt die Autorin Christina Duffy, dass mithilfe eines Bandpassfilters, welchen das Bildbearbeitungsprogramm ImageJ anbietet, Strukturen von Wasserzeichen stark verdeutlicht werden können.[11] Der Bandpassfilter lässt nur ein bestimmtes Frequenzband zu. Unterhalb und oberhalb einer bestimmten Grenzfrequenz werden die Frequenzbereiche gesperrt beziehungsweise deutlich abgeschwächt.[12] Dazu wird eine „Fast Fourier Transformation" (FFT) angewendet, ein mathematisches Verfahren, mithilfe derer die Frequenzen berechnet werden können. Durch den Effekt des Bandpassfilters lässt sich unterstreichen, dass ausschließlich durch die Manipulation periodischer Schwingungen ein Wasserzeichen bereits effektiv verdeutlicht werden kann.

Eine optimale Reihenfolge eines automatisierten Filters lässt sich hieraus jedoch nicht ableiten, denn die verschiedenen Filtereffekte stehen in keiner direkten Abhängigkeit zueinander. Allerdings lässt sich durch die Wirkungskraft der Gradationskurven und des Scharfzeichners feststellen, dass diese Werkzeuge Bestandteile eines automatisierenden Filters sein sollten. Die Kombination beider Anpassungen kann ein Wasserzeichen bereits in dem Maße verdeutlichen, sodass die übrigen Filter weiterhin optional als Mittel der Nachbearbeitung verbleiben können. Eine weitere Möglichkeit zur Automatisierung bietet eine WebGL-Funktion namens „colorMask", bei der bestimmte Farbkanäle einzeln ein- oder ausgeblendet werden können. Dies wirkt sich auf die Kontrastierung der Graphik aus. Ist beispielsweise der rote Farbkanal ausgeblendet und sind der grüne und blaue aktiviert, zeigt sich ein deutlicher sichtbares Wasserzeichen als bei der ausschließlichen Einblendung des blauen Farbkanals, da dieser zu dunkel ist.

Die Filter sind objektorientiert implementiert. Sie besitzen einen Namen, eine Funktion, die ihre Einstellungsart, beispielsweise einen Schieberegler, definiert, eine Funktion für den Effekt beim Einstellen der Parameter der Einstellungsart, sowie eine weitere Funk-

[10] http://computergrafik.informatiker-wissen.de/digitales-bild.html. (02.11.2015)
[11] http://www.bl.uk/eblj/2014articles/pdf/ebljarticle22014.pdf. (02.11.2015)
[12] Arnold, Daniel: Digitale Filter, online: http://userpages.uni-koblenz.de/~physik/informatik/DSV/Filter.pdf. (02.11.2015)

tion zum Zurücksetzen der Parameter. Die verschiedenen Einstellungsarten sind Objekte, die dem Filter-Objekt untergeordnet sind, das bedeutet, dass sie wiederum Eigenschaften besitzen. Bei Schiebereglern beispielsweise wird ebenfalls ein Name festgelegt, auf den sich die Funktion für den Effekt bezieht, eine Bezeichnung, die dem Nutzer angezeigt wird, einen minimalen und maximalen Wert, bis zu welchen der Schieberegler bewegt werden kann, einen Wert zur Initialisierung und eine Schrittangabe, wie groß die Wertänderung bei Bewegen des Reglers sein soll. Die Werte werden bei jeder Anpassung an die Effektfunktion übergeben. Diese überträgt die Anpassung anschließend in Echtzeit auf das Bild.

Eine automatische Filterung kann nun erzeugt werden, indem ein neues Filterobjekt angelegt wird. Der Name kann beliebig gewählt werden, der Funktion, die die Einstellungsart definiert, werden zwei Schieberegler und Gradationskurven hinzugefügt. Die Schieberegler bilden dabei die Anpassungsmöglichkeiten des Scharfzeichners nach, der bereits implementiert ist, der Schärferegler erhält jedoch als initialisierenden Wert einen höheren Parameter. Um die Werte der Kurven zu definieren, muss eine andere Stelle im Programmcode geändert werden, die die Kurvenpunkte beim Laden zurücksetzt. Alle Einstellungsobjekte besitzen eine ID. Über eine If-Abfrage auf diese ID kann explizit dieser Gradationskurve ein weiterer Punkt hinzugefügt werden. Da eine Belichtung in diesem Fall nicht wie im Beispiel des Geldscheins durch ein transparentes Bild hindurch erfolgt, werden für die x-Achse stattdessen hohe Werte gegen 1, beispielsweise 0.75 gewählt, für die y-Achse niedrige Werte, beispielsweise 0.35. Die Gradationskurven sind eigene HTML5-Canvas-Elemente, auf denen Punkte gesetzt werden können, durch die dann automatisch eine Linie gezogen wird. Entsprechend der Punkte überträgt die Funktion im Filterobjekt diese Punkte anschließend auf die WebGL-Programmierung, welche die Pixelberechnungen vornimmt. Die Funktion des automatischen Filters ist zusammengesetzt und lautet „canvas.draw(texture).unsharpMask(this.radius, this.strength).curves(this.points).update();“ Die Parameter „this.radius“, „this.strength“ und „this.points“ sind die ausgewählten Punkte in den jeweiligen Schieberegler, die die Namen „radius“, „strength“ und „points“ tragen. Die Scharfzeichner-Funktion „unsharpMask“ überträgt die Werte des Radius- und Schärfe-Schiebereglers an den WebGL-Code, die Kurven-Funktion die Punkte, bei Veränderung eines Parameters werden stets beide Funktionen berechnet.

Abbildung 4.1: Links das Wasserzeichen des originalen Faksimiles mit der Bandzahl 6 und der Blattnummer 1315, rechts das Ergebnis einer automatisierenden Bearbeitung von Scharfzeichnung und Gradationskurven, nach Quelle: http://163.1.169.40/gsdl/collect/PHerc/index/assoc/HASH01d6/4ec78843.dir/herc.v006.1315.a.01.hires.jpg

4.2 Schwierigkeiten bei der Umsetzung

Wie sich gezeigt hat, lässt sich eine voreingestellte Graphikbearbeitung am Beispiel der bereits vorgegebenen Filter erzeugen. Eine Schwierigkeit stellt jedoch die Möglichkeit dar, den Effekt statt durch einen Klick auf den Filtertitel direkt beim Laden auf das Bild zu übertragen, da viele grundlegende Funktionen ebenfalls beim Laden initialisiert werden. Dadurch ergibt sich die Problematik, dass beim Laden des Bildschirms ein Effekt noch nicht geladen werden kann, weil andere Funktionen, die dafür benötigt werden, noch nicht geladen wurden.

Eine Möglichkeit, dies zu umgehen, ist eine „mouseover“-Funktion für das „body“-Element, welche bei der ersten Mausbewegung über den Bildschirm ausgeführt wird. Nach der Ausführung kann diese wieder deaktiviert werden. Jedoch kann die Problematik auch besser gelöst werden: Denn ist der automatisierende Filter beispielsweise das erste Filterelement, müsste dieses nur automatisch geöffnet werden. Die Selektion ist in diesem

Fall mit dem JavaScript-Befehl „document.getElementsByClassName("title")[0].parentNode“ möglich. Hier wird die erste Klasse „title“ angesprochen und dessen Elternknoten, das Filterelement mitsamt den Schiebereglern und Gradationskurven, selektiert. Das Element kann anschließend in einer vom Entwickler bereitgestellten Funktion zum Laden des Bildes ausgewählt und geöffnet werden. Dabei werden die Einstellungsmöglichkeiten und Parameter, die im Filterobjekt definiert sind, initialisiert.

Durch diese Vorgehensweise lässt sich ein Filter direkt beim Laden auf die jeweilige Graphik anwenden. Allerdings ist der Ladevorgang durch die Ausführung des Filters verlangsamt, der Nutzer kann mit dieser Lösung allerdings zum einen sehen, welche Effekte auf das Bild angewendet wurden, zum anderen kann er die automatisierte Bearbeitung anpassen und somit gegebenenfalls Feineinstellungen vornehmen. Das Problem der zeitlichen Verzögerung wird durch die kombinierte Berechnung des Schieberegler- und Gradationskurveneffekts des automatisierenden Filters hervorgerufen und überträgt sich daher auch auf eine mögliche Bearbeitung der automatisierten Parameter. Daher sollen nur einige besonders wirkungsvolle Filter an dieser Stelle angewendet werden. Die anderen Filter können wie bereits erwähnt als Mittel der Nachbearbeitung verbleiben, oder, wenn der automatische Filter nicht übernommen und geschlossen wird, als Möglichkeit einer individuellen Bearbeitung der Graphik.

Ein weiteres Problem ist die Möglichkeit, das bearbeitete Bild zu speichern. Der Entwickler Evan Wallace hat dies zwar in seiner Implementierung bedacht, allerdings funktioniert die intendierte Übertragung des Canvas-Bildes in einen neuen Browser-Tab nicht optimal. Hier wird nur eine graue Fläche angezeigt. Eine Lösung ist die Funktion update(), welche eigentlich den sichtbaren Inhalt des Canvas mit einem Filterresultat überschreiben soll. Hierdurch wird jedoch auch das Speichern des bearbeiteten Canvas möglich.

5. Anbindung an eine existierende Metadatenbank

Nachdem eine Möglichkeit der automatisierten Graphikbearbeitung hergestellt werden konnte, sollen die Faksimiles und ihre entsprechenden Vorlagengraphiken nun mithilfe einer Anbindung an die grundlegende Metadatenbank eingebunden werden können. Werden keine entsprechenden Vorlagen in der Datenbank gefunden, soll der Nutzer zudem individuelle Vorlagen erstellen können. Eine je nach Verfügbarkeit von Einträgen automatisch generierte Metadatenübersicht soll dem Nutzer außerdem weitere Informationen zur jeweiligen Abzeichnung präsentieren. Abschließend werden wie im vorherigen Kapitel 4.2 Probleme bei der Umsetzung und mögliche Lösungsansätze diskutiert.

5.1 Einbindung der Faksimiles

Zunächst sollen die Einträge der Metadatenbank dazu verwendet werden, um alle Faksimiles in das automatisierte Bildbearbeitungsprogramm, welches aus dem vorherigen Kapitel hervorging, einzubinden. Für eine präzise Auswahl der Graphiken müssen die Schlüssel, nach denen der Nutzer ein Bild selektiert, stets eindeutig sein. Dafür haben sich einerseits die PHerc-Nummer, allerdings nur in Kombination mit der jeweiligen Fragmentbezeichnung, und andererseits die Bandbezeichnung, verknüpft mit der Blattnummer, angeboten. Die Blattnummer ist hier bereits ein eindeutiger Schlüssel, der Vollständigkeit halber soll aber die zugehörige Bandbezeichnung mit abgefragt werden. Der andere Index, bestehend aus der PHerc-Nummer und der Fragmentbezeichnung, ist hingegen nicht immer eindeutig, da bei den Titelseiten keine Fragmentbezeichnung notiert wird. Dieses Problem wird im abschließenden Unterkapitel 5.4 zu den Schwierigkeiten noch einmal aufgegriffen.

Der Entwickler Evan Wallace hat bereits einen Dialog mit dem Nutzer zum Einladen von Graphiken implementiert. Entweder können diese aus einer Bildgalerie oder vom lokalen System ausgewählt werden. Würde diese Lösung übernommen werden, wäre der Umfang an Faksimiles für eine Galerie zu groß, zudem wären die Abzeichnungen als Miniaturansichten nicht unterscheidbar. Darüber hinaus wäre bei einem vollständigen Inventar an Abzeichnungen auch eine Auswahl vom lokalen System unsinnig. Im Folgenden sollen daher zwei neue Dialoge, jeweils eine für das Laden über die Bände und eine für

das Laden über PHerc-Nummern entwickelt werden. In die Vorgehensweise wird weiterhin der Button einbezogen, den der Entwickler bereits für den Aufruf seines Dialogs implementiert hat, allerdings wird seine Funktion entsprechend abgeändert.

Für die Dialoge mit dem Nutzer wird eine jQuery-Bibliothek namens “Sweetalert” zum Einsatz kommen, die bereits in die Benutzeroberfläche der Metadatenbank integriert wurde. Diese Dialogform ist eine optische Verbesserung gegenüber der herkömmlichen JavaScript-Alert-Funktion, bietet darüber hinaus jedoch auch Möglichkeiten zur Interaktion mit dem Nutzer. Unter anderem kann hier ein HTML-Formulareingabefeld genutzt werden, um ein Faksimile auszuwählen. Für eine präzise Eingabe und Hilfe soll dieses Eingabefeld Vorschläge anbieten, sobald der Nutzer das Feld anklickt oder damit beginnt, etwas einzutippen. Damit Vorschläge aus beiden eindeutigen Schlüsseln keine Verwirrung erzeugen, werden zwei unterschiedliche Dialoge entwickelt, deren Vorschlagsliste je nach Auswahl eines Schlüssels, entweder der Bandbezeichnung und Blattnummer oder der PHerc-Nummer und Fragmentbezeichnung, wechselt.

Ein Ansatz hierfür ist ein Dropdown-Menü mit beiden Schlüsseln als Listenelemente. Beim Klick auf ein Element soll dieses ausgewählt und der jeweilige Dialog geöffnet werden. Der Button zum Laden wird aus der Originalversion des Entwicklers übernommen, die bestehende Funktion zum Öffnen des Dialogs im JavaScript beziehungsweise jQuery wird nun abgeändert. Eine neue Anweisung für einen Klick auf den Button soll dem Nutzer nun das Auswahlmenü anbieten, dessen Listenelemente dann mit jeweils einer weiteren Funktion verbunden sind, welche beim Klick darauf den entsprechenden Dialog öffnet.

Die Dialogfenster fordern den Nutzer anschließend zur Eingabe des jeweiligen Schlüssels, entweder eines Bandes oder einer PHerc-Nummer auf. Sobald der Nutzer auf das Eingabefeld klickt oder damit beginnt, eine Eingabe vorzunehmen, sollen ihm Vorschläge angeboten werden, aus denen er einen Eintrag wählen kann, um seine Eingabe zu vervollständigen. Die Vorschläge können in HTML-Datalist-Elementen gespeichert werden. Diesen werden ID-Attribute zugewiesen, die anschließend als List-Attribut für das Input-Element, welches das Eingabefeld generiert, eingetragen und somit mit diesem verknüpft werden können. Für die Vorschläge wird nun mit Hilfe von PHP eine Datenbankabfrage ausgeführt. Wichtig hierfür ist, dass im Code zuvor eine Verbindung zur Datenbank hergestellt wird. Eine Abfrage kann in diesem Fall hartkodiert werden, da sich an den Tabellen- und Feldnamen keine Änderungen ergeben. Die Abfrage zur Ermittlung aller Bände inklusive Blattnummern sieht folgendermaßen aus:

„SELECT g_blattfrg_blätter_Bandbezeichnung, g_blattfrg_blätter_Blattnr FROM g_blattfrg_blätter WHERE (g_blattfrg_blätter_Bandbezeichnung LIKE '%MS. Gr. class. c.%') ORDER BY g_blattfrg_blätter_Blattnr“

Selektiert werden alle Einträge der Felder Bandbezeichnung und Blattnummer in der Tabelle g_blattfrg_blätter, bei denen die Bandbezeichnung „MS. Gr. class. c.“ enthält. „%“ bedeutet, dass davor oder danach weitere Buchstaben oder Zahlen stehen können, in diesem Fall folgt noch eine Bandnummer. „ORDER BY g_blattfrg_blätter_Blattnr“ sortiert die Einträge aufsteigend, sodass sie in der Vorschlagsliste anschließend in dieser Reihenfolge angezeigt werden. Jede Zeile wird anschließend kombiniert mit einem Option-Element zu einem Vorschlag mit dem entsprechenden Inhalt deklariert.

In einer ähnlichen Vorgehensweise werden die Einträge der PHerc-Nummern und Fragmentbezeichnungen in eine Vorschlagsliste eingefügt. Die Abfrage sieht folgendermaßen aus:

„SELECT g_blattfrg_blätter_Bandbezeichnung, g_blattfrg_blätter_Blattnr, g_edfrg_aktion_blattfrg_PHerc_Nr, g_edfrg_aktion_blattfrg_FrgNr_in_RefEdition FROM g_blattfrg_blätter, g_edfrg_aktion_blattfrg WHERE g_blattfrg_blätter_ID = ID_g_blattfrg_blätter AND (g_blattfrg_blätter_Bandbezeichnung LIKE '%MS. Gr. class. c.%') ORDER BY g_edfrg_aktion_blattfrg_PHerc_Nr“

Hier werden wiederum die Felder Bandbezeichnung und Blattnummer benötigt, denn die Bandbezeichnung muss weiterhin „MS. Gr. class. c.“ mit einer darauffolgenden Bandzahl sein. Über einen Vergleich der ID-Nummern werden die Einträge anschließend aus mehreren, verknüpften Tabellen ausgegeben. Die Einträge werden zunächst nach der PHerc-Nummer sortiert. Die Sortierung bei der Datenbankabfrage genügt hier allerdings nicht, denn nun sind zwar die PHerc-Nummern sortiert, jedoch nicht gemeinsam mit ihren entsprechenden Fragmentbezeichnungen. Alle Einträge mit der Nummer 19 beispielsweise würden demnach sortiert, jedoch mit einer wilden Reihenfolge der Fragmentbezeichnungen angezeigt werden. Deshalb werden die Einträge als kombinierte Strings zunächst in einem Array gespeichert. Auf dieses kann nun die Funktion „natsort“ ausgeführt werden, die in PHP vordefiniert ist und alle Elemente nach ihrer natürlichen Reihenfolge sortiert. Da die Feldeinträge im Array verbunden sind, folgen auf die PHerc-Nummern in aufsteigender Reihenfolge nun auch entsprechend ihre Fragmentbezeichnungen in ebendieser Reihenfolge. Danach können alle Einträge des Arrays in Option-Elementen als Vorschläge in die Liste eingetragen werden. Ein weiteres, wichtiges Datalist-Element enthält alle Einträge der ersten und zweiten Vorschlagsliste, hier werden

die Felder allerdings durch ein Semikolon getrennt. Diese ist für die weitere Verarbeitung und die Ausgabe der Metadatenübersicht von Bedeutung.

Einen Schritt weitergedacht stellt sich die Frage, wie die Auswahl über den Eingabewert des Nutzers mit dem entsprechenden Dateinamen in Verbindung gebracht werden kann. Hierfür bieten sich „reguläre Ausdrücke“ an, die bestimmte Vorkommen von Zeichen in einem String oder einer Zahl erkennen. Gegebenenfalls lassen sich diese auch ersetzen. Ist eine Eingabe zu einer PHerc-Nummer und Fragmentbezeichnung seitens des Nutzers erfolgt, wird dieser Wert in einem ersten Schritt zunächst mit allen Werten der Vorschlagsliste mit allen Metadaten abgeglichen. Dazu wird dem Eingabewert zuvor mit dem Befehl „inputValue = inputValue.replace(" ", ";");“ ein Semikolon hinzugefügt. Wird ein Eintrag gefunden, der den Eingabewert enthält, wird dieser als neuer Wert der Eingabe definiert. Somit ist bei beiden Dialogformen ein Eingabewert mit Bandzahl und Blattnummer vorhanden, denn diese sind stets im Dateinamen enthalten. Als regulärer Ausdruck wird mit „var bandzahl = inputValue.match("[1-7]{1}");“ daher im Eingabewert nach einer einzigen Zahl von eins bis sieben gesucht. Das erste Vorkommen dieser Zahl wird anschließend in einer Variablen Bandzahl gespeichert. Bei der Verarbeitung des Eingabewerts des Dialogs zum Laden über eine PHerc-Nummer werden die Bandzahl und das darauffolgende Semikolon aufgelöst. Die danach folgende Zahl ist die Seitenzahl. Diese ist beim Laden über den Band der übrig gebliebene Eingabewert, beim Laden über die PHerc-Nummer die nächste Zahl, die zwischen einer und vier Ziffern besitzt. Beide werden in der Variable „seitenzahl“ abgespeichert. Somit kann aus den beiden Variablen nun bei beiden Dialogen der Dateiname der Abzeichnung zusammengesetzt werden.

5.2 Einbindung der Vorlagengraphiken

Um die Vorlagengraphiken einbinden zu können, wurde im Kapitel zu den Grundlagen bereits deutlich, dass eine Erweiterung der ursprünglichen Metadatenbank notwendig war, denn zunächst waren keine Einträge zu den Wasserzeichen vorhanden, die mit den Dateinamen abgeglichen hätten werden können. Die Dialogform, welche der Entwickler Evan Wallace bereits zum Laden von Bildern implementiert hatte, fand bei den Faksimiles keine Anwendung. Bei den Vorlagengraphiken sieht dies allerdings genau umgekehrt aus, denn hier würden dem Nutzer in einer Galerie nur die entsprechenden Graphiken angeboten werden, die dem Faksimile über die Datenbank zugeordnet werden können.

Dies wäre hier daher ein optimaler Lösungsansatz. Auch die Möglichkeit, Vorlagen über das lokale System einzuladen, würde Sinn ergeben, da nicht für jede Abzeichnung Vorlagen eingetragen sind. Sollte der Nutzer individuelle Vorlagen erstellt haben, kann er diese gegebenenfalls über sein System in das Programm laden. Insgesamt ist der Entwurf des Entwicklers hier wie geschaffen für die weitere Vorgehensweise.

Dabei soll ein neues Filterelement erstellt werden, welches sich ebenfalls auf- und einklappen lassen soll, jedoch keine Schieberegler und Gradationskurven enthält, sondern die Tabellenstruktur der Schieberegler nutzt, um eine Galerie zu entwickeln. Falls keine Vorlagen notiert sind, sollen neue Vorlagen gezeichnet werden können. Darunter sollen in diesem Fall über einen Button auch individuelle Graphiken hochgeladen werden können.

Als erster Schritt wird in HTML ein neues Img-Element erstellt, in welches die Wasserzeichen-Graphik eingebunden werden soll. Der Das Src-Attribut, welches den Dateipfad enthält, bleibt zunächst leer. Danach werden in einem Datalist-Element alle Einträge der Felder „Band“, „Seitenzahl“ und „WZ_Nr“ aus der neu erstellten Tabelle, die von der Excel-Übersicht übernommen wurde, durch Semikolons getrennt gespeichert. Die Verwendung eines Datalist-Elements hat hier den Hintergrund, dass die Daten allgemein als Liste gespeichert sind. Das Element erhält ein Attribut „placeholder“, welches beim Laden den Pfad des Bildes erhält. Ein weiteres Datalist-Element enthält nach Abfrage der neuen Tabelle, welche alle Dateinamen der Vorlagen enthält, alle Namen der Vorlagengraphiken.

Anschließend wird ein Filterelement erstellt, das jedoch keine Schieberegler oder Gradationskurven enthalten soll. Das bedeutet, dass die Initialisierungsfunktion, welche im Objekt auf den Filternamen folgt, leer ist. In der darauffolgenden Effektfunktion wird zunächst eine bestehende Galerie gelöscht. Dies ist für den Fall von Bedeutung, dass eine andere Abzeichnung geladen wird. Beim Laden der Faksimiles wird wie erwähnt der Pfadname als ein Attribut „placeholder“ im Img-Element gespeichert. Dies hat den Hintergrund, dass Variablen ausschließlich lokal innerhalb einer Funktion verfügbar sind, vorausgesetzt, sie sind nicht global definiert. Daher kann die Filterfunktion nicht direkt auf den Pfad der Abzeichnung zugreifen. Stattdessen wird der dieser dem Attribut entnommen. Mithilfe eines regulären Ausdrucks wird nun zunächst der Pfad zum Dateinamen entfernt, sodass der Name verbleibt. Wie beim Dialog zum Laden einer Abzeichnung werden Band und Seitenzahl bestimmt.

Im Folgenden werden der Band und die Seitenzahl mit den Einträgen aus dem ersten Datalist-Element verglichen. Ergibt sich hierbei ein Treffer, ist in dem Eintrag, welcher

dem Feld „WZ_Nr“ entstammt, das entsprechende Wasserzeichen des Faksimiles enthalten. Dieses wird nun ein weiteres Mal mit den Vorlagennamen aus dem zweiten Datalist-Element abgeglichen, die den Namen des Wasserzeichens enthalten. Die Treffer werden in einem Array gespeichert und in eine Tabellenstruktur übertragen. Wird eine Vorlage in der Tabelle angeklickt, wird dessen Pfad in das Quellenattribut des Img-Elements eingetragen. Wichtig bei der Darstellung mit CSS ist, dass dieses zuvor nicht sichtbar war und nun eine absolute Positionierung mit einem höheren Z-Index als die Faksimilegraphik einnimmt. Um das Element zu bewegen und zu zoomen, stellt jQuery die Funktionen „draggable“ und „resizable“ zur Verfügung, die dem Nutzer erlauben, Elemente mit der Maus zu verschieben beziehungsweise zu vergrößern oder zu verkleinern.

Für den Fall, dass keine entsprechenden Vorlagen vorhanden sind, soll der Nutzer neue Vorlagen selbst anfertigen können. Mit einer If-Abfrage lässt sich testen, ob das Array, welches die entsprechenden Vorlagen speichern soll, leer ist. Ist dies der Fall, wird zunächst, um über die Faksimilegraphik zu zeichnen, ein neues, transparentes Canvas-Element erstellt, welches über dem Bild absolut positioniert ist. In eine Tabelle können Werkzeuge wie Zeichnen, Radieren, Entfernen der Zeichnung oder Bewegen erstellt werden. Die Koordinaten, nach denen gezeichnet wird, orientieren sich an Seitenabständen. Hier können sich Probleme ergeben, auf die das Unterkapitel zu den Schwierigkeiten bei der Umsetzung noch näher eingehen wird. Mit verschiedenen Methoden können nun Einstellungen vorgenommen werden. „ctx.strokeStyle = "#000";“ zum Beispiel definiert die Farbe zum Zeichnen, „ctx.lineWidth = 5;“ die Strichstärke, mit der auf dem Kontext „ctx“ gezeichnet werden soll. Wird das Zeichnen-Werkzeug ausgewählt, sorgt die Funktion „ctx.stroke();“ dafür, dass bei Klick auf den Kontext eine Zeichnung angefertigt werden kann. Beim Radieren wird die Operation umgekehrt, sodass die Farbe wieder mit dem ursprünglichen Bildinhalt übermalt werden kann.

Eine Möglichkeit zum Hochladen einer Vorlagengraphik bietet ein Input-Element vom Typ „file“. Über den Button können unter anderem Graphiken hochgeladen werden, die Quelle der ausgewählten Datei wird anschließend auf das Src-Attribut des Img-Elements übertragen, um die Vorlage anzuzeigen.

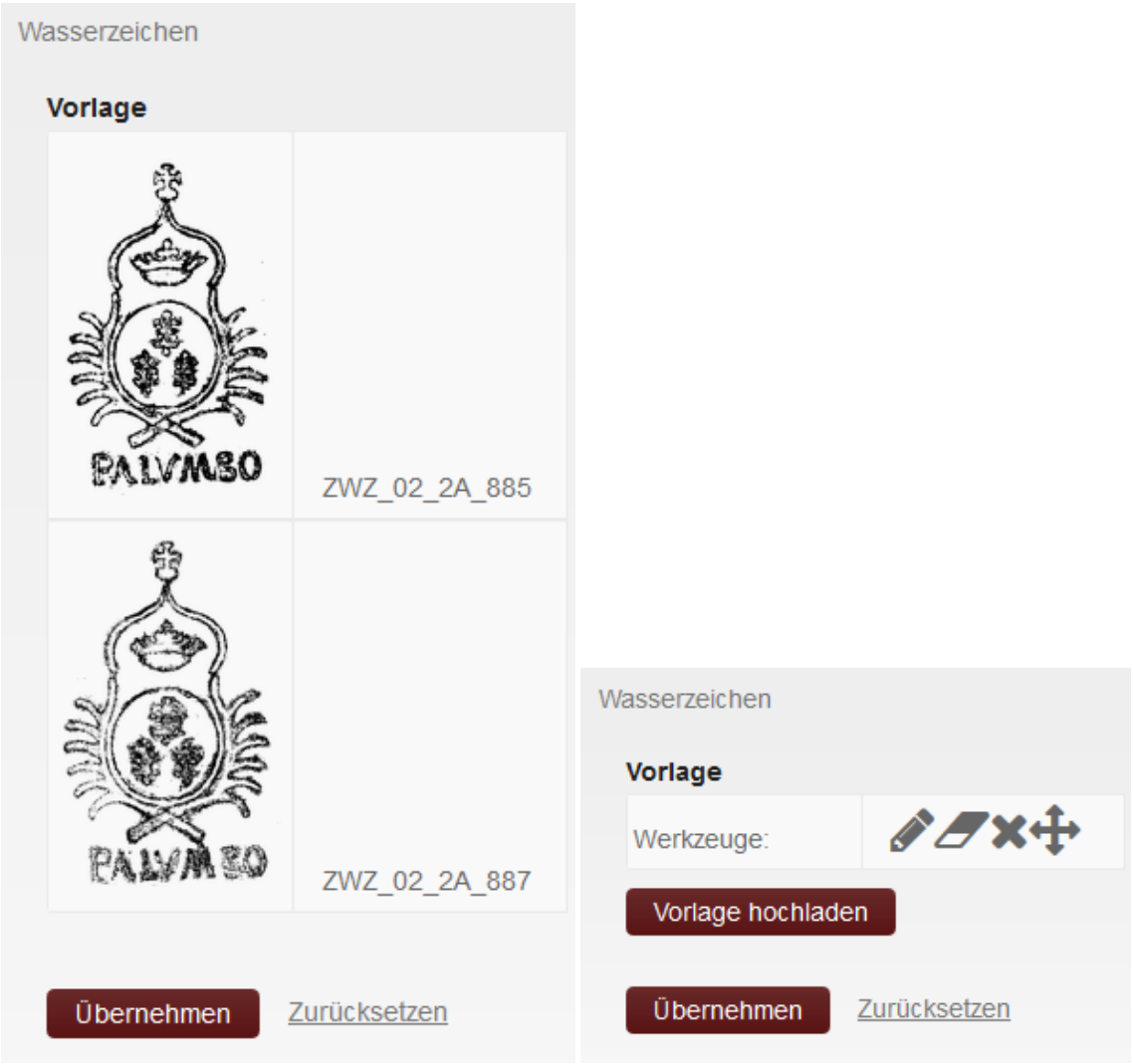


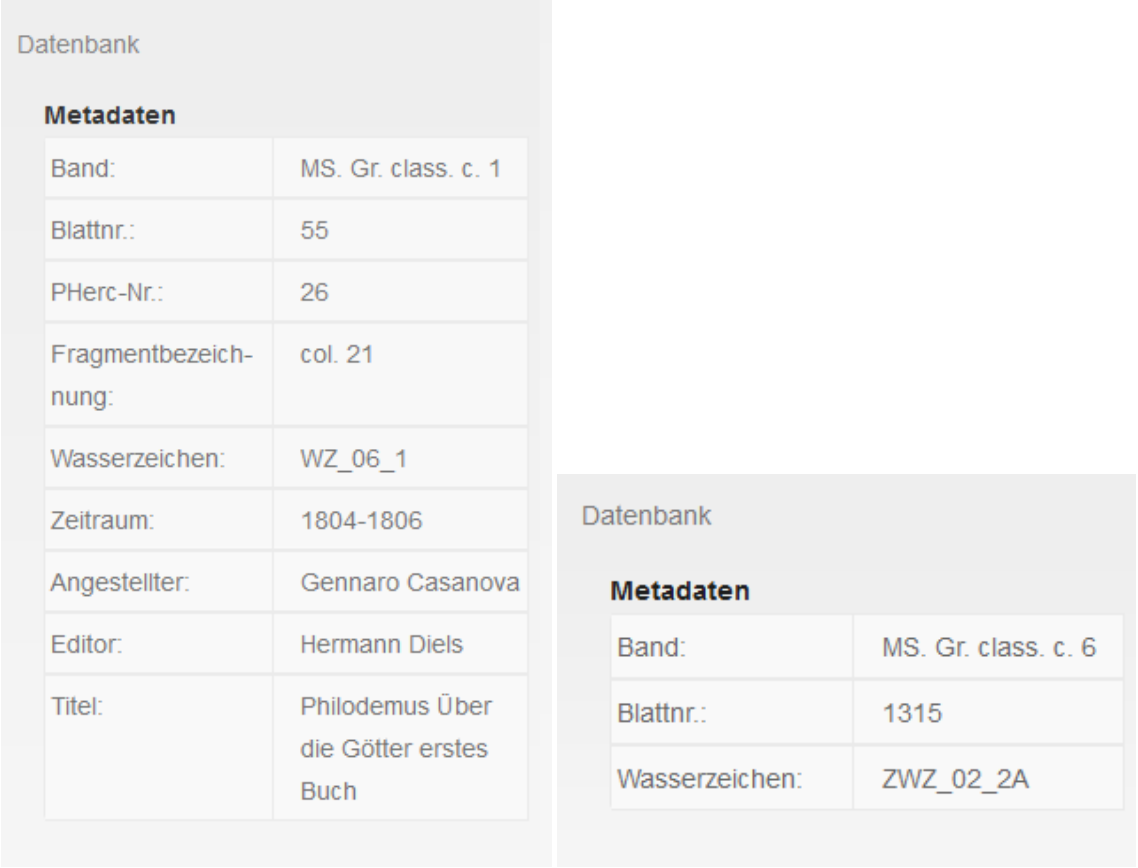
Abbildung 5.1: Links die entsprechenden Vorlagen des Faksimiles aus dem Band 6 mit der Blattnummer 1315 als Galerie, rechts die Möglichkeit zum Zeichnen und Hochladen von Vorlagen beim Faksimile aus dem Band 1 mit der Blattnummer 104, da hier in der Datenbank keine entsprechende Vorlage notiert ist.

5.3 Metadatenübersicht

Auch zur Erstellung einer Metadatenübersicht wird wieder ein Datalist-Element benötigt, allerdings wurde dieses bereits für das Laden von Faksimiles über eine PHerc-Nummer angelegt. Hier können nun noch weitere Metadaten abgefragt werden, die in der Datenbank gespeichert sind. Relevante Daten sind beispielsweise der Zeitraum, der Angestelltenname, der Editor und der Titel der referenzierenden Edition. Die Einträge der Felder werden hier wieder durch Semikolons getrennt. Danach wird ein weiteres Filterelement

nach dem bereits bekannten Muster angelegt. Der Wasserzeichenname wird wie bereits bei den Vorlagen erkannt. Aus dem Dateipfad werden durch reguläre Ausdrücke erneut die Band- und Seitenzahl gefiltert. Anschließend werden die Einträge des Datalist-Elements, welches die Wasserzeichennamen enthält, dahingehend untersucht, ob sie die jeweilige Band- und Seitenzahl des Dateipfads der Abzeichnung enthalten. Der nachfolgende Wert im entsprechenden Eintrag enthält anschließend den entsprechenden Namen des Wasserzeichens.

Nun wird auch in dem Datalist-Element mit den Metadaten nach der aus dem Dateipfad der Abzeichnung ermittelten Band- und Seitenzahl abgeglichen. Ergibt sich hier ein Treffer, wird dieser Eintrag mit dem Befehl „split“ entlang seiner Semikolons aufgeteilt und die dazwischenstehenden Metadaten in ein Array übertragen. Danach können die einzelnen Elemente des Arrays in einer entsprechenden Variablen gespeichert und in einer Tabellenstruktur ausgegeben werden. Mit einer If-Abfrage kann die Verfügbarkeit geprüft werden.



Datenbank

Metadaten

Band:	MS. Gr. class. c. 1
Blattnr.:	55
PHerc-Nr.:	26
Fragmentbezeichnung:	col. 21
Wasserzeichen:	WZ_06_1
Zeitraum:	1804-1806
Angestellter:	Gennaro Casanova
Editor:	Hermann Diels
Titel:	Philodemus Über die Götter erstes Buch

Datenbank

Metadaten

Band:	MS. Gr. class. c. 6
Blattnr.:	1315
Wasserzeichen:	ZWZ_02_2A

Abbildung 5.2: Links die variable Metadatenübersicht des Faksimiles aus dem Band 1 mit der Blattnummer 55, rechts die der Abzeichnung aus dem Band 6 mit der Blattnummer 1315. Die Einträge werden je nach Verfügbarkeit in der Datenbank notiert.

5.4 Probleme bei der Umsetzung

Eine Schwierigkeit bei der Anbindung an die entsprechende Metadatenbank ist, dass JavaScript keine Datenbankabfragen unterstützt. Stattdessen können jedoch PHP-Skripte Einträge abfragen. Als Ansatz können die Daten über Elemente, welche die Daten in einer Liste speichern, mit JavaScript abgefragt werden. Alternativ können mit AJAX auch Daten über ein externes PHP-Skript von JavaScript abgefragt und zurückgegeben werden, ohne dass die Seite neu geladen werden muss.

Werden wie in diesem Kapitel beschrieben Filterelemente gebildet, in denen eigentlich keine wirklichen Filter implementiert sind, stellt sich die Frage, wie diese angeglichen werden können, denn, wenn beispielsweise ein Filter geschlossen wird, ohne den Graphikeffekt zu übernehmen, bleiben die Veränderungen weiterhin bestehen. Eine Vorlage zum Beispiel behält ihre Pfadangabe, während der Scharfzeichner, wenn sein Effekt nicht übernommen wird, durch eine Funktion zurückgesetzt wird, sobald das Filterelement geschlossen wird. Diese Eventualitäten müssen im Programmcode entsprechend bedacht werden, indem diese Funktion erweitert wird. Schließt ein Filterelement ohne Klick auf den Button „Übernehmen“, müssen alle Einstellungen, die vorgenommen wurden, auf den vorherigen Stand zurückgesetzt werden.

Eine zusätzliche Schwierigkeit ist, dass die Wasserzeichen der Faksimiles häufig gespiegelt sind. Hier können allerdings über ein weiteres Filterelement Buttons mit CSS-Anweisungen eingebunden werden, durch die die Graphik einfach gespiegelt werden kann. In Abbildung 3.1 wurde dies rechts beispielsweise mithilfe des zusätzlichen Filterelements angewendet. Vergrößerungen und Verkleinerungen des Bildes mit dem Mausrad können über Funktionen definiert werden. Die jQuery-Funktion „draggable“ ermöglicht zudem das Verschieben des Bildes mit der Maus. Aus Vergrößerungen, Verkleinerungen oder Verschiebungen der Graphik resultieren allerdings Probleme beim Zeichnen. Die Stelle, an der gezeichnet wird, befindet sich anschließend nicht mehr dort, wo sich die Maus befindet. Der Grund hierfür ist eine Berechnung der Koordinaten, welche sich bei Veränderung der Position oder Größe des Canvas nicht verändert.

Eine weitere Problematik stellt die Verwendung der Vorschlagsliste mit dem Dialog der Bibliothek „SweetAlert“ dar, denn das List-Attribut des Eingabefeldes, in welchem die ID des Datalist-Elements angegeben wird, dessen Vorschlagsliste verwendet wird, soll sich je nach Auswahl des Nutzers ändern. Dies ist allerdings nicht direkt der Fall, wenn der Nutzer den Dialog zum Laden über die PHerc-Nummer anwählt. Erst im zweiten Versuch

wird die entsprechende Vorschlagsliste verwendet. Das extern eingebundene Skript, in dem das Eingabefeld definiert ist, wird hier nicht bereits im ersten Anlauf geändert.

Ein Ansatz für die Problematik, dass die Titelseiten beim PHerc-Schlüssel keine Fragmentbezeichnung enthalten, ist, eine automatische Nummerierung hinzuzufügen. Dies kann über If-Abfragen bei der Datenbankabfrage erreicht werden. Allerdings gilt dabei zu beachten, dass die Fragmentbezeichnungen zuvor unsortiert sind, dies bedeutet beispielsweise, dass der als erste Titelseite deklarierte Vorschlag in Wahrheit auf eine andere Titelseite verweisen kann.

Auch das Dateiformat der Vorlagen ist für die Überlagerung der bearbeiteten Graphik nicht optimal. Denn das JPEG-Format enthält keinen Alpha-Kanal, wodurch der Hintergrund der Vorlage in keinem Fall transparent sein kann. Hier muss zuvor eine Konvertierung von JPEG in ein PNG- oder GIF-Format erfolgen. Mit der Stapelverarbeitung von Adobe Photoshop beispielsweise lassen sich die 198 Vorlagen schnell bearbeiten, denn der Hintergrund der Vorlagen ist zumeist weiß. In einigen Ausnahmefällen entspricht die Vorlage einer zugeschnittenen Darstellung der Abzeichnung, aus welcher sie gewonnen wurde. In diesen Fällen ist eine Nachbearbeitung notwendig.

6. Zusammenfassung und Ausblick

An dieser Stelle folgt eine abschließende Revision der Erkenntnisse mit einem Ausblick auf weitere Möglichkeiten der Graphikbearbeitung von Wasserzeichen und der Anbindung an eine Metadatenbank.

Das Resultat dieser Arbeit ruht auf drei grundlegenden Säulen. Die zu bearbeitenden Faksimiles und die Vorlagengraphiken, welche erstere überlagern sollen, stellen eine von ihnen da. Bei den Abzeichnungen sind hier zur präzisen Auswahl der Bilder eindeutige Schlüssel wichtig. Beachtet werden müssen auch die Dateinamen, über welche eine Graphik schließlich mit ihrem Pfad eingebunden wird. Bei den Vorlagen ist von Bedeutung, dass diese einen Alpha-Kanal zur Darstellung von Transparenz besitzen. Die zweite Säule, die zur Einbindung der Graphiken und einer Metadatenübersicht tragend ist, ist die Datenbank. Diese bildet das Fundament zur Koordination der Daten für das Laden der Abzeichnungen über die PHerc-Nummer und die Fragmentbezeichnung, welche nicht im Dateinamen enthalten sind. Auch können die Faksimiles mit ihren entsprechenden Vorlagen ausschließlich verknüpft und die Metadatenübersicht nur deshalb erstellt werden, weil die Informationen, welche der Dateiname der Abzeichnungen enthält, mit weiteren Informationen aus der Datenbank verknüpft werden kann. Eine dritte Säule ist das Programm für Graphikeffekte des Entwicklers Evan Wallace, welches das HTML5-Canvas-Element mit dem 3D-Kontext WebGL kombiniert. Durch einige der Filter kann eine wirkungsvolle automatisierende Graphikbearbeitung zur Herausfilterung von Wasserzeichen ermöglicht werden.

Diese Umsetzung erfordert eine genaue Kenntnis über die Abläufe des Programmcodes. Wichtig ist in diesem Zusammenhang das Wissen darüber, wie die bestehenden Filter implementiert wurden und welchen Effekt sie ausüben. Für die Herausfilterung von Wasserzeichen zeigen sich verstärkte Kontraste, in Verbindung mit bestimmten Farbkanälen, eine Belichtung beziehungsweise Verdunkelung der Graphik und die Beeinflussung von Frequenzen am wirkungsvollsten. Eine bestimmte Reihenfolge muss dabei nicht beachtet werden.

Für die Einbindung der Graphiken ist relevant, dass mit JavaScript keine Datenbankabfragen möglich sind. Allerdings können die Einträge in einer Listenstruktur gespeichert und dann mit JavaScript abgerufen werden. Die Listen können zudem als Vorschlagslisten bei Eingabefeldern verwendet werden. Als Dialogformen bietet sich bei den Abzeichnungen eine solche Auswahlliste an, bei den Vorlagen kann eine Galerie erstellt werden.

Zudem ist hier die Möglichkeit des Ladens eigener Vorlagen vorteilhaft, wenn keine der Abzeichnung entsprechenden Vorlagen vorhanden sind. Alternativ kann auf einem überlagernden, transparenten Canvas-Element mit Kontext-Methoden auch eine neue Vorlage gezeichnet werden.

Die Überlagerung von Canvas-Elementen kann auch angewendet werden, um eine Graphik auf verschiedenen Ebenen zu bearbeiten, die sich einzeln aus- und einblenden lassen. Wird ein Filter übernehmen, könnte ein neues Element erzeugt werden, welches das vorherige überlagert. Hier wäre wichtig, dass der Kontext des vorherigen Canvas kopiert und auf die neue Ebene übertragen wird. Durch Ein- und Ausblendungen der Ebenen könnten somit alle Bearbeitungsschritte wiederhergestellt oder rückgängig gemacht werden, wie der Entwickler dies zunächst auch vorhatte, denn, wie in Kapitel 2.3 beschrieben, hat er zwei entsprechende Buttons eingefügt, deren Funktion er jedoch wohl aufgrund der zeitlichen Beschränkung seines Projekts nicht mehr realisieren konnte. Daher hat er die Buttons als Kommentare markiert, um sie im Programm nicht erscheinen zu lassen.

Auch die Implementierung eines Bandpass-Frequenzfilters, wie ihn Christina Duffy für ihrem Aufsatz „The Discovery of a Watermark on the St Cuthbert Gospel using Colour Space Analysis" verwendet hat, ließe sich noch verwirklichen. Wie beschrieben, werden dabei bestimmte Bereiche von hohen und niedrigen Frequenzen ausgeschlossen, um den Effekt zu erzeugen. Da ImageJ eine Anwendung ist, deren Code in Java öffentlich zugänglich ist, ließe sich dieses Vorhaben in JavaScript übertragen. Allerdings ist seine Ausführung um einiges langsamer, da hier komplexe Transformationen vorgenommen werden.

Mit der Einbindung eines Zauberstab-Werkzeugs könnten möglicherweise außerdem bearbeitete Wasserzeichen markiert werden. Dies gelingt bei den Abzeichnungen bereits durch das Anagnosis-Projekt von Kallimachos, bei dem Texte automatisiert erkannt werden. Die deutlicheren Strukturen der bearbeiteten Wasserzeichen können hier möglicherweise ebenfalls erkannt werden. Kallimachos führt Geisteswissenschaftler, Informatiker und Bibliothekare zu einem regional basierten Digital Humanities-Zentrum mit besonderem Funktionsprofil zusammen, das institutionell an der Universitätsbibliothek Würzburg angesiedelt ist.[13] Ziel von Anagnosis ist der Brückenschlag zwischen papyrologischen Bilddatenbanken und der internationalen Volltextdatenbank für literarische Papyri Digital Corpus of Literary Papyri (aufbauend auf papyri.info).[14]

[13] http://www.kallimachos.de/project/doku.php. (02.11.2015)
[14] http://www.kallimachos.de/project/doku.php/kallimachos:anagnosis:start. (02.11.2015)

Auch mit der Gestaltungssprache CSS sind mittlerweile einige Graphikbearbeitungen möglich, jedoch können damit keine Berechnungen an Pixeln, wie bei den Frequenzfiltern vorgenommen werden. Daher ist die Umsetzung mit Canvas und WebGL um einiges mächtiger. Die konkrete Zielsetzung konnte erfüllt werden.

7. Literaturverzeichnis

Sekundärliteratur

Arnold, Daniel: Digitale Filter, online: http://userpages.uni-koblenz.de/~physik/informatik/DSV/Filter.pdf.

Duffy, Christina: The Discovery of a Watermark on the St Cuthbert Gospel using Colour Space Analysis, online: http://www.bl.uk/eblj/2014articles/pdf/ebljarticle22014.pdf.

http://computergrafik.informatiker-wissen.de/digitales-bild.html.

http://evanw.github.io/glfx.js/docs.

http://www.kallimachos.de/project/doku.php.

http://www.kallimachos.de/project/doku.php/kallimachos:anagnosis:start.

http://www.w3.org/TR/2dcontext.

http://www.w3.org/wiki/HTML/Elements/canvas.

https://www.khronos.org/registry/webgl/specs/1.0/#1.

Külzer, Jakob: Filter im Frequenzraum, online: http://www.jakusys.de/t3/fileadmin/files/documents/kuelzer_filter_im_frequenzraum.pdf.

Walter Scott: Fragmenta herculanensia; a descriptive catalogue of the Oxford copies of the Herculanean rolls together with the texts of several papyri accompanied by facsimiles, Oxford 1885.