Digital ist besser?*

Möglichkeiten der automatisierten Aufbereitung und Bewertung von Fileablagen mit Python am Beispiel einer digitalen Fotosammlung

Stephan Lenartz

Landesarchiv Baden-Württemberg

Dialog Digital
Landesarchiv Baden-Württemberg
Band 1

Herausgegeben vom
Landesarchiv Baden-Württemberg

Bei der vorliegenden Publikation handelt es sich um die für die Veröffentlichung überarbeitete und aktualisierte Fassung der Masterthesis von Stephan Lenartz im Studiengang Konservierung Neuer Medien und Digitaler Information an der Staatlichen Akademie der Bildenden Künste Stuttgart (Erstprüfer: Prof. Johannes Gfeller, Zweitprüfer: Prof. Dr. Christian Keitel).

Satz: BookDesigns, Heidesee

Landesarchiv
Baden-Württemberg

urn:nbn:de:101:1-2020052506

Danksagung des Autors

Herzlichst gedankt sei Dr. Kai Naumann vom Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Ludwigsburg, für die fachliche Betreuung der vorliegenden Arbeit. Seine zahlreichen E-Mails sowie Auskünfte vor Ort und am Telefon waren mir eine große Hilfe.

Für ihre Unterstützung des Projekts und die kritische Begleitung der Workflow-Entwicklung gilt mein ausdrücklicher Dank außerdem Dr. Franz-Josef Ziwes und Sibylle Brühl vom Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen.

Inhalt

Abstract und Kurzfassung

Memory institutions are faced with ongoing changes that originate in an extensive technological transformation process. For many years now such institutions have focused on challenges that arise in the need for digital long-term preservation and user access solutions. However, it is only gradually that the discussion targets concrete ways of processing unstructured data. The complexity and extent of born-digital archives necessitates new strategies and tools for tackling archival core tasks. After a brief sketch of approaches to the appraisal of photographs and archive‘s experiences in processing file collections so far established, this thesis focuses on a digital photographic collection of the amateur photographer Gunter Aipperspach, which was archived at Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen. The workflow description introduces the use of sixteen scripts, specifically written in the Python programming language, that enabled widely automated processing of about 220.000 digital photographs and thus demonstrates a row of Python’s potential benefits for digital archivists.

Auch in Gedächtnisinstitutionen hat die Digitalisierung längst Einzug gehalten. Seit Jahren widmet man sich hier den Herausforderungen, die sich bei der notwendigen Schaffung neuer Strukturen zur Langzeitarchivierung und Bereitstellung digitaler Informationen stellen. Erst allmählich rückt dabei die ganz konkrete Auseinandersetzung mit der Überlieferung unstrukturierter Fileablagen in den Vordergrund. Komplexität und Umfang solcher Dateisammlungen erfordern bei der Bewerkstelligung archivarischer Kernaufgaben zum Teil neue Strategien und Werkzeuge. Nach der Schilderung grundlegender Aspekte der Fotobewertung und Ausführungen zum Stand der Archivierung von Fileablagen, stellt die vorliegende Arbeit den digitalen Nachlass des Amateurfotografen Gunter Aipperspach, der am Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen archiviert wurde, in den Mittelpunkt. Eine Workflow-Beschreibung führt die Anwendung von sechzehn Skripten vor, die in der Programmiersprache Python geschrieben und mithilfe derer die rund 220.000 Fotografien bis hin zum Ingest weitgehend automatisiert bewertet und aufbereitet wurden. Exemplarisch wird so der potentielle Nutzen von Python für die digitale archivarische Arbeit aufgezeigt.

1. (Digitale) Fotografien im Archiv. *Eine Annäherung*

Der ungarische Maler, Fotograf und Bauhaus-Lehrer László Moholy-Nagy bemerkte einmal, nicht der „Schrift-, sondern der Photographie-Unkundige wird der Analphabet der Zukunft sein“[1]. Kein Jahrhundert später hat die Fotografie am vorläufigen Ende einer Kette technischer Innovationen mit der digitalen Wende ihre wohl offensichtlichste Transformation zum Massenmedium vollzogen und so haben das Foto und das Fotografieren eine Allgegenwärtigkeit erlangt, die für Moholy-Nagy im ersten Drittel des vergangenen Jahrhunderts noch kaum zu erahnen war. Seine Vision hat sich längst erfüllt. Auf besonders eindrückliche Weise offenbart sie sich heute, wenn vielen von uns der Umgang mit der Smartphone-Kamera mindestens so selbstverständlich erscheint wie das Schreiben, das im Übrigen ja nicht mehr mit dem von Moholy-Nagy implizierten Stift, sondern im Alltag häufig ebenfalls mit und auf dem Smartphone stattfindet. Gleichzeitig sind wir als Rezipierende eine ständige – vielfach auch kritisierte, aber wohl nicht mehr abebbende – Bilderflut in Print- und Onlinemedien gewohnt, die Texte oftmals in den Hintergrund rückt.

Während aber der lesende Analphabet per se als contradictio in adiecto – als Widerspruch in sich – erscheint, besteht auf Seiten der Fotografie ein Ungleichgewicht zwischen der Fähigkeit des Fotografierens und jener einer angemessenen Rezeption, das vielleicht erst auf den zweiten Blick zutage tritt. Zum Erfolg des Mediums beigetragen hat wohl auch, dass wir Bilder kognitiv viel leichter verarbeiten als Texte. Bilder implizieren eine scheinbare Eindeutigkeit, was leicht vergessen lässt, dass Fotografien wie auch Texte immer in einem bestimmten Kontext stehen, der die Zulässigkeit einer eindimensionalen Wahrnehmung verbietet. Auch kann ein und dasselbe Foto auf ganz unterschiedliche Weisen „gelesen“ werden, je nachdem worauf das persönliche Interesse Blick und Aufmerksamkeit richtet. Bilder im Allgemeinen und Fotografien im Besonderen sind demnach viel weniger eindeutig und eindimensional als sie uns intuitiv erscheinen mögen und so erhebt das bloße Fotografieren-Können uns auch noch längst nicht über den Status eines fotografieunkundigen Analphabeten.[2] Wie der Textrezeption liegt der Rezeption von Fotografien eine Komplexität zugrunde, deren Bewältigung für einen kundigen Umgang erlernt werden muss. Nur so erschließt sich uns ihr wahrer Gehalt.

Als eine logische Folge des Verbreitungs- und Bedeutungszuwachses, den das Medium auf so eindrückliche Weise erfuhr, hat auch in den Geistes-, Sozial-, Kultur- und Naturwissenschaften eine Hinwendung zu Bildquellen als Gegenstand stattgefunden.[3] Nicht zuletzt rückt die Fotografie so auch zwangsläufig in den Fokus von Gedächtnisinstitutionen, denen die verantwortungsvolle Aufgabe zukommt, diejenigen Bildquellen zu erhalten, die zukünftig Forschern und interessierten Nutzern zur Rekonstruktion unserer Gegenwart zur Verfügung stehen werden. Besonders die Archive haben sich inzwischen verstärkt der Dokumentation

1 Zitiert nach *Benjamin*, Gesammelte Schriften, S. 151.
2 Natürlich erfordert die Rezeption anspruchsvoller Texte ebenfalls mehr als grundlegende Lese- und Schreibkomptenzen.
3 In Anlehnung an den „linguistic turn“ wird diese Hinwendung zum Bild als „iconic turn“ bezeichnet. Einen Überblick über die Entwicklungen und deren Stellenwert in den einzelnen Disziplinen gibt: *Günze und Mersch*, Bild, besonders S. 373–465. Auch wenn Günzel und Mersch dies nicht explizit berücksichtigen, widmet sich auch die lange textbezogene Geschichtswissenschaft inzwischen verstärkt dem Quellenwert von Bilddokumenten. Siehe z. B. den Einführungsband: Paul (Hg.), Visual History; Zum richtigen Umgang mit Fotografien in der Landeskunde z. B. auch: *Reininghaus*, Fotografien, S. 21–42.

gesamtgesellschaftlicher Prozesse verschrieben.[4] Hier, wo traditionell vor allem schriftliche Zeugnisse von kultureller und rechtlich-administrativer Bedeutung ausgewählt, erschlossen und langfristig aufbewahrt werden, tritt die Analogie des fotografischen Analphabeten besonders plastisch hervor. Ein im Moholy-Nagy'schen Sinne der Fotografie unkundiger Archivar oder eine ebensolche Archivarin wird ein Foto ebenso wenig bewerten und damit über seine Archivwürdigkeit entscheiden können, wie ein Analphabet, der sich schriftlichen Akten und Urkunden gegenübersieht.[5]

Zudem erscheint die Digitalisierung der Fotografie geradezu als symptomatisch für die weitgreifende Digitalisierung der Gesellschaft und so drängt sich uns – vielmehr noch als sein fotografisches Pendant – der „digitale Analphabet" ins Bewusstsein, von dem nicht selten in mahnender und warnender Absicht zu lesen und zu hören ist.[6] Zweifellos standen und stehen auch den Archiven mit der notwendigen Hinwendung zu einer digitalen Überlieferungsbildung große Herausforderungen ins Haus – vermutlich ist der Begriff der Revolution nicht ganz unangebracht – und so verwundert es kaum, dass die Literatur zur Entwicklung neuer Konzepte und Lösungen kaum mehr zu überblicken ist.

Auch über eine fachgerechte Bewertung wurde in der deutschen Archivlandschaft ausgiebig und teils kontrovers diskutiert. Inzwischen herrscht über die zentralen Prinzipien, Methoden und Ziele weitgehend Konsens, wenn auch hier mitunter die Auswirkungen der digitalen Archivierung zu diskutieren bleiben.[7] Für den Bereich der Archivaliengattung Fotografie hingegen ist diesbezüglich erst in jüngerer Zeit eine intensivere Auseinandersetzung zu beobachten, obschon entsprechende Erfahrungen bereits seit längerem existieren.[8] Bemerkenswert erscheint dabei, dass wiederum die Digitalisierung durch Ablösung der analogen Fotografie als Katalysator dafür verantwortlich zeichnet, dass sich Archive wachsenden fotografischen Beständen gegenübersehen. Hinzu treten massenhaft digitale Fotografien, deren Archivierung parallel dazu bereits ebenfalls zu meistern ist.

Die Bewertungs-Entscheidung ist und bleibt eine archivarische Kernaufgabe. Nicht alles kann und soll dauerhaft archiviert werden. Gerade in Zeiten der gesteigerten digitalen Produktion bedarf es also einer begründeten Selektion.[9] Die besondere Verantwortung ergibt sich aus dem Wissen, dass eine einmal gefällte Kassations-Entscheidung unumkehrbar und die zur Vernichtung freigegebene Information unwiederbringlich verloren ist. Das gilt für Unterlagen jeglicher Art genauso wie für Fotografien, seien sie analoger oder inzwischen eben immer häufiger auch digitaler Natur. Dass allein die zu bearbeitende Informationsmenge eine Herausforderung oder gar ein geradezu unüberwindbares Hindernis für eine

4 Zu dieser Entwicklung und der damit verbundenen Terminologie, hier vor dem Hintergrund der Hochschularchive zusammengefasst, siehe: *Kühnel*, Dokumentationsprofil.
5 Selbiges gilt natürlich auch für die Erschließung von Fotografien.
6 Eine Google-Suche erzielt 124.000 Treffer.
7 Die Liste entsprechender Veröffentlichungen ist lang. Als Überblick über die Kontroverse der 90er Jahre empfiehlt sich: *Uhl,* Geschichte der Bewertungsdiskussion; Ausführlicheres hierzu bei: *Robert Kretzschmar*, Die „neue archivische Bewertungsdiskussion"; zur Rolle der Bewertung im digitalen Archiv: *Kretzschmar*, Alles neu zu durchdenken?; *Tiemann* (Hg.), Bewertung und Übernahme elektronischer Unterlagen; Türck, Veränderungen von Bewertungsgrundsätzen bei der Übernahme digitaler Unterlagen?.
8 Vgl. *Metz*, Weniger ist oft mehr, hier S. 4. Man beachte auch gleich den programmatischen Beitragstitel, der die Notwendigkeit einer Bewertung prägnant zusammenfasst.
9 Dies gilt auch für elektronischen Unterlagen. Siehe hierzu beispielsweise auch: *Neukom*, Können wir endlich alles behalten?

archivische Bearbeitung – von der Bewertung über die Erschließung bis zur Bereitstellung – sein kann, ist nicht neu.[10] Doch allein, wenn man sich am Beispiel der Fotografie die Möglichkeiten der digitalen Speicherung vergegenwärtigt und man bedenkt, dass dies eine enorme Steigerung der Informationsmenge zur Folge hat, wird deutlich, dass dieses Problem in digitalen Zeiten noch eklatanter werden wird. Die Herausforderungen der digitalen Archivierung sind vielfältig und komplex, aber allein die zu bearbeitende Informationsmenge kann schnell zu einem kritischen Moment werden. Die vorliegende Arbeit bewegt sich im Spannungsfeld der als Notwendigkeit erachteten Bewertung und der mit herkömmlichen Methoden und Konzepten schlichtweg nicht zu bewältigenden Problematik, die sich aus der überlieferten Informationsmenge ergibt.

Glücklicherweise stellt uns die Informationstechnologie nicht nur vor neue Herausforderungen, sondern liefert neue Lösungsmöglichkeiten gleich mit. Auf die theoretische Darstellung des archivischen Umgangs mit Fotografien und digitalen Dateisammlungen folgt daher eine praxisbezogene Darstellung, wie die archivischen Herausforderungen der Aufbereitung einer digitalen Sammlung von rund 200.000 Fotografien des Amateurfotografen Gunter Aipperspach mithilfe weitgehend plattformunabhängiger Hilfsmittel bewerkstelligt werden kann. Dabei wurde von Beginn an darauf Wert gelegt, dass die zum Einsatz kommende Software zur Automatisierung der Teil-Prozesse möglichst plattformunabhängig adaptierbar und auf ähnliche Fälle anwendbar ist. Unabhängig von der konkreten Sammlung soll so auch veranschaulicht werden, welche Möglichkeiten uns die Computerisierung der Arbeit bietet – zumindest dann, wenn wir bereit sind, den Status des digitalen Analphabetismus abzulegen.

10 Vgl. hierzu: *Greene* und *Meissner*, More Product, Less Process.

2. Zur archivischen Bewertung fotografischer Sammlungen

2.1. Erfahrungen mit der Bewertung von Fotografien

Die vorliegende Arbeit versteht sich als Beitrag zur computergestützten Aufbereitung und Bewertung digitaler Dateisammlungen, die auf der exemplarischen Bearbeitung eines digitalen Fotonachlasses aufbaut. Wenngleich sich die bisher erschienenen Beiträge zur Bewertung von archivischen Fotobeständen vornehmlich auf analoge Bestände beziehen, sind hieraus zur Auslotung der Möglichkeiten und Grenzen der automatisierten Bewertung digitaler Foto-Sammlungen grundsätzliche Ansätze zu entnehmen. Die nachfolgenden Ausführungen destillieren also eine unter diesem spezifischen Gesichtspunkt stehende Auswahl an Erkenntnissen aus der bisherigen Forschungsdiskussion, ohne aber auf eine umfassende Literaturübersicht zur archivischen (Foto-) Bewertung abzuzielen. Vielfach werden bei der Thematisierung herkömmlicher Fotosammlungen auch Bestandsbildungs- oder Erhaltungsmaßnahmen berührt, die für die vorliegende Arbeit untergeordnete oder keine Relevanz haben. Beispielsweise entfällt im Fall digitaler Fotosammlungen der Schritt zur Identifizierung des fotografischen Materials. In Ermangelung eines Trägermediums ist somit auch der physische Erhaltungszustand kein relevantes Kriterium. Schäden an Negativ- oder Positivmaterialien auf Glas-, Film- und Papierträgern können hingegen die Notwendigkeit kostenintensiver konservatorischer Maßnahmen nach sich ziehen. Aufwände dieser Art sind vor allem bei älteren analogen Fotobeständen zu analysieren und die damit kurz-, mittel- oder auch langfristig prognostizierten Kosten entsprechend abzuwägen und in die Bewertungsentscheidung miteinzubeziehen.[11] Die Bestandserhaltungsmaßnahmen digitaler Fotografien sind vergleichsweise unkritisch.

Auf dem Feld der Fotobewertung besonders hervorgetan haben sich in den vergangenen Jahren Nora Mathys und Axel Metz.[12] Den lange vorherrschenden zurückhaltenden Umgang mit der Bewertung von Bildarchiven begründet Metz damit, „dass die Aufbewahrung von Fotos aufgrund ihres vergleichsweise geringen Volumens deutlich weniger Raumprobleme aufwirft als die Aktenarchivierung. Gerade das Thema Raumnot aber war immer wieder ein entscheidendes Movens für neue Bewertungsansätze“[13]. Interessanterweise kehrt sich das Verhältnis zwischen dem Platzbedarf von Text und Bild bei der digitalen Speicherung hingegen um, weswegen Metz auch mit häufigeren Bewertungsvorgängen bei Digitalfotos rechnet.[14] An dieser Stelle kann vorweggenommen werden: In der Tat ist die schiere Anzahl an Einzelbildern ein

11 Allgemeine Empfehlungen zur Archivierung von Fotografien, mit jedoch nur sehr knappen Bemerkungen zur Bewertung, bietet z. B. folgende Handreichung: Memoriav (Hg.), Die Erhaltung von Fotografien. Eine recht aktuelle Einführung in den archivischen Umgang mit Fotografien und entsprechende weiterführende Literaturhinweise liefert außerdem folgende Masterarbeit: *Feldt*, Herausforderungen im Umgang mit digitalen und analogen Fotografien im kommunalen Archivwesen. An dieser Stelle sei zudem angemerkt, dass die Bewertungs-Thematik in der englischsprachigen Literatur schon um einiges früher als in der Deutschen ihren Niederschlag gefunden hat. Als besonders wegweisend ist hier zu nennen: *Leary*, The Archival appraisal of photographs. Eine Ausnahme für den deutschen Bereich bildet außerdem ein auf Film- und Fernsehproduktionen (d. h. AV-Medien) bezugnehmender Beitrag aus der DDR, der aufgrund der thematischen Nähe auch für die Bewertung von Fotos zumindest allgemeine Kriterien umreißt: *Kahlenberg* und *Schmitt*, Zur archivischen Bewertung von Film- und Fernsehproduktionen.

12 *Mathys*, Welche Fotografien sind erhaltenswert?; *Mathys*, Das visuelle Erbe. Neben dem genannten Titel (siehe Anm. 8) von Axel Metz außerdem: *Metz*, Nicht jedes Bild sagt mehr als tausend Worte; *Metz*, Die archivische Bewertung von Fotobeständen. Zuvor waren nur einzelne Beiträge zur Thematik erschienen, darunter: *Wiegand*, Das „archivische Foto“.

13 *Metz*, Weniger ist oft mehr, S. 4.

14 Vgl. ebenda, S. 11.

ganz entscheidender Beweggrund für die Bewertung der in der vorliegenden Arbeit thematisierten Fotosammlung, denn zum einen fallen bei der digitalen Speicherung zukünftig für jede einzelne Datei laufende Kosten an.[15] Aber mehr noch: Angesichts der Sammlungsgröße von mehr als 200.000 Fotos, erscheint eine intellektuelle Bewertung in einem angemessenen Kosten-/Nutzenverhältnis nicht leistbar und erfordert geradezu die Entwicklung eines automatisierten Ansatzes. So soll auch einer der Gründe ausgeschaltet werden, der bei analogen Fotosammlungen häufig für einen Verzicht einer Bewertung und Erschließung führt, womit die Fotografien von vorneherein von einer potenziellen Nutzung ausgeschlossen werden.[16] Insofern erscheint das Prinzip der teilweisen digitalen Automatisierung von Aufbereitungs- und Bewertungsprozessen – also eine technische Unterstützung des Archivars bei wichtigen Kernaufgaben seiner Arbeit – als ein nicht geringzuschätzender Vorteil gegenüber der Aufbereitung analoger Bestände, bei der diese Möglichkeit fehlt.[17]

Nach der Analyse von Metz lohnt sich eine Bewertung schon ab Kassationsraten von 16,7 %, wobei er bei seiner Rechnung explizit nur die potenziell entstehenden Verzeichnungskosten – sowohl Einzel- als auch Sammelerschließung eingeschlossen – berücksichtig.[18] Zieht man Verpackungs- und Lagerungs- beziehungsweise Speicherkosten hinzu, liegt die Amortisierungsrate entsprechend noch niedriger, wobei Metz aufgrund eigener Erfahrungen von realistischen Kassationsraten zwischen 50 und 85 % ausgeht.[19] Aus finanzieller Sicht erscheint eine Bewertung demnach ratsam und somit kann unabhängig von weiteren Faktoren bereits ein kaum zu missachtendes Argument für ihre Durchführung ins Feld geführt werden.

Weiterhin sprechen auch genuin archivische Zielsetzungen für eine Bewertung. Die Erschließung sowie langfristige Lagerung oder Speicherung von weniger Verzeichnungseinheiten bedeutet in erster Linie eine Informationsverdichtung, die unmittelbar dem Interesse zukünftiger Nutzer entgegenkommt, gezielt und mit möglichst wenig Rechercheaufwand an die gewünschten Informationen zu gelangen.[20] Metz geht davon aus, dass digitale Fotografien genauso viel Erschließungsaufwand bedeuten wie analoge.[21] Dass dies nur bedingt der Fall ist, wird auch anhand der Erschließung der Sammlung Aipperspach zu sehen sein – allerdings wird hier auch keine Einzelbilderschließung vorgenommen.

15 Siehe hierzu auch: *Wolf*, Stadt im Bild 2.0.

16 Dass eine Nicht-Bearbeitung von Beständen ganz allgemein ein großes Problem für Archive darstellt, ist weithin bekannt. Siehe auch noch einmal Anm. 10.

17 Einen Sonderfall bilden gewissermaßen digitalisierte Bestände, auf die in der vorliegenden Arbeit nicht weiter Bezug genommen werden soll, die aber im Archivwesen angesichts großangelegter Digitalisierungsprojekte einen wichtigen Platz einnehmen. Hierzu sei beispielhaft verwiesen auf: *Pfenninger*, Bildarchiv digital; *Caraffa*, „Wenden!“.

18 Vgl. *Metz*, Nicht jedes Bild sagt mehr als tausend Worte, S. 6–9.

19 Vgl. ebenda, S. 33.

20 Dadurch steigt nicht nur die Zufriedenheit des Nutzers, auch haben Archivmitarbeiter weniger Anfragen zu bearbeiten und somit mehr freie Kapazitäten für andere Arbeiten zur Verfügung. Vgl. ebenda, S. 9.

21 Vgl. ebenda, S. 8. Zum Vergleich zieht Metz die Möglichkeit einer flachen Erschließung elektronischer Akten heran. Diese seien im Volltext nach der gewünschten Information durchsuchbar, Bilder hingegen bedürften per se der Bereitstellung zusätzlicher Informationen. Grundsätzlich ist dies richtig, aber die bereits gegenwärtigen und erst recht zukünftigen Möglichkeiten zur automatischen Gewinnung entsprechender Metadaten, man denke an Motiverkennungs-Algorithmen, sind nicht zu unterschätzen. Nur am Rande sei bemerkt, dass sich dies grundsätzlich sehr weit und radikal denken lässt: So analysiert der auf dem Feld der Medienarchäologie forschende Medienwissenschaftler Wolfgang Ernst die Möglichkeiten der digitalen Speicherung vor dem Hintergrund medientheoretischer Fragestellungen und plädiert gar für eine gänzlich textbefreite Organisation digitaler Bildarchive: *Ernst*, „Jenseits der Verschlagwortung?“

Grundsätzlich sind Bewertungsverfahren so anzulegen, dass die angesetzten Kriterien transparent sind. Wenn es also darum geht, Bewertungsentscheidungen von vorneherein nachvollziehbar zu gestalten, sind möglichst objektivierbare Kriterien hilfreich. Wenn auch inzwischen einige Arbeiten zur Thematik vorliegen, scheitern allgemeingültige Postulate vorerst an der medienspezifischen Komplexität von Bildquellen und der Existenz verschiedener Überlieferungsformen, weshalb vermehrt auf spezifische Bestände Bezug genommen wird.[22] Insgesamt fehlt es jedoch noch immer an einer „breite[n] Palette an Varianten“[23], welche die unterschiedlichen Anforderungen verschiedener Sammlungen abzubilden vermag. Ob eine übergreifende Bewertungstheorie darüber hinaus ein grundsätzliches Desiderat darstellt, sei an dieser Stelle zur Disposition gestellt.[24] Das Ideal einer gänzlich objektiven, also einer von subjektiven Entscheidungen des ausführenden Archivars freien Bewertung, ist jedenfalls ohnehin nicht zu erreichen. Unabhängig von jedem theoretischen Fundament und konkreten Kriterien bleibt die zu erzielende Bewertungsqualität immer in besonders starker Abhängigkeit an die ausführende Person, deren Wissen und Verantwortungsbewusstsein geknüpft, was jedoch nicht als Argument gegen eine Bewertung gewendet werden sollte. Auch das Wissen darum, dass jedes Werturteil einer zeitlichen Relativität unterworfen ist, von der sich die bewertende Person nicht lösen kann, sollte nicht als ein Solches interpretiert werden.[25]

Um diese Umstände so gut es eben geht abzufedern, liegt es im Interesse der Archive, fotografische Überlieferungen beispielsweise in die Entwicklung von Bewertungsmodellen miteinzubeziehen, die gewonnenen Erkenntnisse jedenfalls zu bündeln und in spezifische Bewertungsverfahren münden zu lassen. Die vermehrt erscheinenden Publikationen sind Hinweis darauf, dass diese Verantwortung zunehmend ernst- und die damit verbundenen Herausforderungen angenommen werden.

2.2 Skizzierung zentraler Kriterien der Bewertung

Fotografische Überlieferungen begegnen uns im Archiv meist in einer der drei folgenden Varianten. 1. Fotos als Bestandteil von Akten. 2. Geschlossene Übernahmen vorstrukturierter Sammlungen, beispielsweise in Form von (Fotografen-)Nachlässen. 3. Fotobestände, die im Archiv angelegt und meist thematisch sortiert wurden. Während die Fotografien im ersten Fall in der Regel der Aktenbewertung unterliegen, sind für die beiden weiteren Fälle spezifische Bewertungskriterien anzulegen. Der in der vorliegenden Arbeit bearbeitete Bestand ist der zweiten Kategorie zuzuordnen, weshalb sich die nachfolgenden Ausführungen auf diese

22 Vgl. LWL-Archivamt für Westfalen, Schaust Du noch oder archivierst Du schon?, S. 102; *Sagurna*, Bewertungsfragen.
23 *Leimgruber* u. a., Das Fotoerbe sichern, S. 152.
24 Dahingehend vertritt Michael Pfeiffer einen auf systemtheoretischen und konstruktivistischen Überlegungen gestützten Sammlungsansatz, der die Anschlussfähigkeit der aufgenommenen Information ins Zentrum stellt. Als grundlegendes Problem macht er die überwiegend inhaltsbezogene Bewertung von Fotografien aus, die im Kern dem archivischen Provenienzprinzip widerspräche, also eher dem Pertinenzprinzip verhaftet sei und eine allgemein operationalisierbare Vorgehensweise erschwere. Siehe: *Pfeiffer*, Visuelle Überlieferungsbildung. Zur Fixierung auf den scheinbaren Gegensatz der beiden Prinzipien sei hier lediglich kritisch angemerkt, dass sich Pertinenz- und Provenienzprinzip viel weniger als unvereinbare Gegensätze gegenüberstehen, als dass sie sich gegenseitig ergänzen. Gleichwohl bleibt richtig, dass allgemeine Empfehlungen für inhaltliche Bewertungskriterien kaum ausgesprochen werden können. Siehe hierzu auch: *Pilger*, Grundsätze, Methoden und Strategien der Überlieferungsbildung in Archiven.
25 Vgl. hierzu *Pfrunder*, Aufwerten, umwerten, abwerten, S. 31–33.

Kategorie beschränken.[26] Die drei Basis-Optionen der Bewertung sind in etwa vergleichbar mit der archivischen Bewertung von gleichförmigen (Massen-)Unterlagen: Vollarchivierung, Teilarchivierung, Totalkassation.[27] Da eine vorstrukturierte Sammlung in der Regel seriellen Charakter hat und nach thematischen, örtlichen oder zeitlichen Gesichtspunkten unterteilt ist, empfiehlt sich allgemein ein Top-Down-Vorgehen, sodass nacheinander die gesamte Sammlung, einzelne Einheiten und dann erst Einzelbilder in den Blick genommen werden. Anders als im Fall von Akten, wo in der Regel keine Einzelblattbewertung durchgeführt wird, ist eine Bewertung von Fotografien auf Einzelbildebene typisch.[28]

Durch Erstbegutachtung ist auf institutioneller Ebene zunächst die grundsätzliche Entscheidung darüber zu treffen, ob die Sammlung zum Sammlungsprofil des Archivs passt und wie sie sich zu weiteren (Teil-) Beständen verhält. Im Sinne einer Überlieferungsbildung im Verbund wäre weiterhin zu überprüfen, ob die Überlieferung an der richtigen Stelle geschieht. Unabhängig von den gewöhnlichen Sperrfristen sind bei der Archivierung von Fotografien immer auch urheberrechtliche Fragen zu klären.[29] Demnach sind im Vorfeld der Übernahme Regelungen zu treffen, die für Rechtssicherheit sorgen. Dies schließt hier also insbesondere die Übertragung entsprechender Nutzungsrechte ein.[30]

Es wird nun angenommen, dass eine Teilarchivierung vorgenommen werden soll. Ziel der weiteren Bewertung ist also die zielgerichtete Verdichtung der Sammlung. Mittels Stichproben – auch als „Sampling“ oder „Sample-Bildung“ bezeichnet – kann die Sammlung ausgedünnt werden. Zentral erscheint dabei, dass einzelne Auswahlkriterien sorgfältig gegeneinander abgewogen werden. Ist die Sammlung in kleinere Einheiten unterteilt, kann das Ausdünnen auf unterschiedlichen Ebenen geschehen. Es ist sowohl eine Kassation ganzer Einheiten als auch eine Bewertung auf Einzelbildebene denkbar.

Welche Kriterien sollten nun handlungsweisend sein und wie verhalten sich die beiden Möglichkeiten der Ausdünnung zueinander? Bei der archivischen Bewertung im Allgemeinen erhält die Beurteilung des Unikatcharakters einer Information besondere Aufmerksamkeit. Daher wird auch für Fotos die Redundanz als eines der führenden Kassationskriterien angeführt.[31] Dass dieses Prinzip, allzumal im Falle digitaler Sammlungen, nicht unumstößlich ist, wird später noch zu sehen sein. Ohnedies tritt hier jedoch bereits die Unmöglichkeit allgemeingültiger Aussagen hervor, was auf die oben angedeutete Problematik zurückgeführt werden kann, die sich aus den Gegensätzen einer inhalts- und einer evidenzbasierten Bewertung ergibt.[32] Grundsätzlich ist festzuhalten, dass, anders als bei Einzelfotos, Evidenzwerte bei geschlossenen Sammlungen, also auch Nachlässen, eine bedeutende Rolle spielen, da sich hierin Arbeitsweise des Fotografen ausdrückt.

26 Die Einteilung folgt *Metz*, Weniger ist oft mehr, S. 4. Der Autor bezieht sich in seinen Arbeiten in erster Linie auf den dritten Fall.
27 Vgl. *Kretzschmar*, Aussonderung und Bewertung von sogenannten Massenakten, S. 108.
28 Vgl. *Charbonneau*, The Selection of Photographs, S. 132.
29 Siehe *Pilger*, Ein neues Positionspapier des VdA-Arbeitskreises „Archivische Bewertung“ zur Überlieferungsbildung im Verbund.
30 Ausdrücklich und besonders davon betroffen sind auch digitale Fotos, da die Langzeitarchivierung digitaler Informationsobjekte bereits automatisch das Anfertigen einer Kopie bedeutet. Vgl. nestor-Arbeitsgruppe Standards für Metadaten (Hg.), Wege ins Archiv, S. 16–18.
31 Metz verwendet den Oberbegriff „Redundanz“ für Dubletten und sehr ähnliche Bilder mit gleichbleibender Aussage, die er als „Quasi-Dubletten“ bezeichnet. Vgl. *Metz*, Nicht jedes Bild sagt mehr als tausend Worte, S. 28. Zum Umgang mit Redundanz vgl. z. B. auch: *Wolf*, Stadt im Bild 2.0, S. 67; *Noble*, Considerations for Evaluationg Local History Photographs, S. 20.
32 Diese Einteilung unternahm erstmals Theodore R. Schellenberg. Vgl. *Schellenberg*, The Appraisal of Modern Public Records.

Daher plädiert Axel Metz auch für eine Bewertung auf der Ebene ganzer Einheiten oder auf Ebene der gesamten Überlieferung, wohingegen die Bewertung einzelner Bilder „in solchen Fällen zumeist“[33] entfalle. Die Bedeutung der Evidenz verbietet demnach zunächst auch, technisch unzulängliche – beispielsweise unscharfe, über- und unterbelichtete – Aufnahmen oder eben Redundanzen zu entfernen, da diese ebenfalls Ausdruck der Arbeitsweise des Fotografen sind. Wie im Fall des in der vorliegenden Arbeit bearbeiteten Nachlasses kann dennoch eine Einzelbildbewertung notwendig erscheinen, sodass sich aus einer Gewichtung aller Kriterien gegeneinander letztlich eine Kombination beider Verfahren ergibt. Es bleibt dann aber darauf zu achten, dass die charakteristischen Aspekte der einzelnen Einheiten für sich genommen auch nach der Einzelbildbewertung erhalten bleiben.[34]

Vereinbarkeit mit dem Sammlungsprofil, rechtliche Aspekte, Redundanz, technische Mängel – also die bisher neben dem Evidenzkriterium bereits erwähnten Kriterien – sind noch recht problemlos objektivierbar. Selbiges gilt für ein weiteres zentrales Kriterium, nämlich das der begleitenden Kontextinformation.[35] Die grundlegenden Informationen zum Fotografen, Aufnahmedatum oder Motiv sollten vorhanden sein, zusätzliche dokumentarische Information erhöhen den Wert einer Sammlung, wohingegen ihr Fehlen die jeweiligen Fotografien in den meisten Fällen wertlos erscheinen lässt, sodass diese kassiert werden können. Auch dies hatte entscheidende Auswirkungen auf den in der vorliegenden Arbeit beschriebenen Bewertungsprozess.

Schwieriger, weil nicht so leicht objektivierbar, sind andere Kriterien. Beispielsweise der Informationsgehalt und der fotohistorische Wert einer Fotografie, eng damit zusammenhängend außerdem die zu antizipierenden Benutzerinteressen und insbesondere auch ästhetische Kriterien.[36] Je nach Sammlung ist außerdem eine spezifische Ausdifferenzierung inhaltlicher Kriterien sinnvoll. Was die Bewertungsentscheidungen also so komplex erscheinen lässt, ist die Notwendigkeit der abwägenden Verknüpfung gattungsspezifischer, inhaltlicher und klassischen archivarischen Bewertungskriterien, die mit dem Provenienzprinzip zusammenhängen.[37]

33 Vgl. *Metz*, Nicht jedes Bild sagt mehr als tausend Worte, S. 5.
34 Vgl. *Mathys*, Welche Fotografien sind erhaltenswert?, S. 37f.
35 Metz subsumiert diese Kriterien daher auch unter Oberbegriff „harte Kriterien“. Vgl. *Metz*, Nicht jedes Bild sagt mehr als tausend Worte, S. 13–20.
36 Vgl. ebenda, S. 20–23; *Mathys*, Das visuelle Erbe, S. 101.
37 So schon: *Wiegand*, Das „archivische Foto“, S. 41. Statt der weiteren Auflistung spezifischer Kriterien, die ohne konkrete Bezüge zu einem Bestand kaum an Anschaulichkeit gewinnen, sei abschließend auf den Kriterienkatalog von Nora Mathys verwiesen, den diese im Rahmen ihrer Bewertung des Ringier Bildarchivs entworfen hat: *Mathys*, Das visuelle Erbe, S. 95 und 99ff.

3. Der archivische Umgang mit Fileablagen

3.1 Grundtendenzen der Archivierung elektronischer Daten[38]

Es wurde eingangs bereits darauf hingewiesen, welch umwälzenden Einfluss der Einzug der Digitalisierung auf die archivische Arbeit insgesamt hatte.[39] International wurde um die Jahrtausendwende verstärkt mit der Entwicklung von Konzepten und Standards zur digitalen Langzeitarchivierung begonnen. Zum Teil recht zögerlich wurden diese auch in Deutschland aufgegriffen, um in erster Linie der in den Archivgesetzen bereits damals seit fast zwanzig Jahren verankerten Verpflichtung nachzukommen, elektronische Unterlagen übernehmen und langfristig – d. h. auf unbegrenzte Dauer – speichern zu müssen.[40]

Wenn auch mit der technischen Konzeptionierung und anschließenden Realisierung digitaler Archive sich den Möglichkeiten nach nicht grundsätzlich darauf beschränkt wurde, konzentriert sich die archivische Praxis kommunaler und staatlicher Archive bis zum heutigen Tag doch überwiegend auf die geregelte Übernahme elektronischer Unterlagen, also derjenigen Daten, die in den unterschiedlichen Provenienzstellen, beispielsweise in Fachverfahren und im Zuge der nach und nach realisierten Einführung der E-Akte auch in Form elektronischer Aktenführung entstehen.[41] Insgesamt unterscheiden sich die Daten freilich in Art und Zusammensetzung je nach abgebender Provenienzstelle, wobei eine zunehmende Hinwendung zu umfangreicheren und komplexeren Daten zu erkennen ist, die besondere Anforderungen an die Übernahme stellen.[42]

Im Zuge wachsender praktischer Erfahrungswerte schlug sich der Umgang mit elektronischen Unterlagen zunehmend auch in archivfachlichen Diskussionen wie der Bewertungsdiskussion nieder.[43] Bis auf Weiteres lässt sich Letztere zusammenfassen mit: „Die grundsätzlichen Methoden und Prinzipien der Bewertung gelten für eine digitale Überlieferung genauso wie für die herkömmliche Papierüberlieferung, denn ausschlaggebend ist der Inhalt, nicht die Form.“[44] Gleichzeitig gelten diesbezüglich jedoch eine Reihe von Einschränkungen, die sich aus einigen Besonderheiten elektronischer Daten und ihrer Erhaltungsbedingungen ergeben, auf die hier nicht weiter eingegangen werden kann. Hier sei lediglich exemplarisch auf die Bedeutung signifikanter

38 Ausdrücklich ausgenommen von der Betrachtung werden hier von analogen Fotobeständen angefertigte Digitalisate (vgl. auch Anm. 17).

39 Vgl. diesbezüglich auch die Artikel zum veränderten archivarischen Berufsbild in Archivar 63 (2010/4). Darunter: *Kretzschmar*, Aktuelle Entwicklungstendenzen des archivarischen Berufsbilds.

40 Vereint wurden die Konzepte erstmals im Jahre 2011 unter dem Dach des Kompetenznetzwerks nestor. Siehe hierzu die entsprechende Publikation (hier in der aktualisierten Fassung aus dem Jahr 2012): nestor-Arbeitsgruppe Digitale Bestandserhaltung (Hg.), Leitfaden zur digitalen Bestandserhaltung; hierzu auch: *Keitel*, Der nestor-Leitfaden zur Digitalen Bestandserhaltung und seine Folgen für die Archive. Ohne die Bedeutung der weiteren Konzepte schmälern zu wollen, sei an dieser Stelle weiterhin nur auf die deutsche Übersetzung des vielleicht einflussreichsten, des OAIS-Referenzmodells (ISO 14721), verwiesen: nestor-Arbeitsgruppe OAIS-Übersetzung / Terminologie (Hg.), Referenzmodell für ein Offenes Archiv-Informations-System; wegweisend beispielsweise die Entwicklung des digitalen Archivs „DIMAG“ am Landesarchiv Baden-Württemberg: *Keitel*, *Lang* und *Naumann*, Konzeption und Aufbau eines digitalen Archivs.

41 Mit Unterlagen sind allgemein alle digitalen Objektive gemeint, die Gegenstand einer Bewertung sein können.

42 Zur Bewertung von Datenbanken aus Fachanwendungen siehe exemplarisch: *Koch* u. a., Bewertungsautomat statt Autopsie.

43 So bereits in Ansätzen im Positionspapier des VdA-Arbeitskreises Archivische Bewertung aus dem Jahr 2005: *Kretzschmar*, Positionen des Arbeitskreises Archivische Bewertung im VdA zur archivischen Überlieferungsbildung; vgl. dazu das aktuelle Positionspapier zur Bewertung elektronischer Fachverfahren: VdA-Arbeitskreis „Archivische Bewertung“ (Hg.), Bewertung elektronischer Fachverfahren; außerdem: *Tiemann*, Bewertung und Übernahme elektronischer Unterlagen.

44 *Zahnhausen*, Überlieferungsbildung von analog zu digital, S. 17.

Eigenschaften, die in Abhängigkeit antizipierter Nutzungsziele bestimmt und in jeder Repräsentation eines digitalen Informationsobjekts – somit also beispielsweise im Rahmen von Formatmigrationen – erhalten werden müssen. Ihre Bestimmung wird daher bereits zu einem integralen Bestandteil der inhaltlichen Bewertung. Die im Vergleich zur herkömmlichen Überlieferung nicht zuletzt hierdurch veränderte Art der Überlieferungsbildung zwingt zu gewissen Anpassungen der Bewertungsstrategien, die weiterhin zu diskutieren sind.[45]

Grundsätzlich fällt die Breite der Erfahrungswerte in einzelnen Archiven auf dem Gebiet der digitalen Archivierung noch recht unterschiedlich aus.[46] Dies gilt umso mehr, je weiter man den Blick von der Kern-Überlieferung digitaler Unterlagen im Sinne strukturierter Abgaben von kommunalen und staatlichen Behörden abwendet. Übernahmen von Webseiten, E-Mails und unstrukturierten Dateisammlungen bilden mehrheitlich noch Ausnahmen; zum Teil werden auch solche Daten aber bereits seit längerer Zeit erfolgreich archiviert.[47] Besonders die Archivierung von Dateisammlungen findet erst in jüngster Zeit stärkere Aufmerksamkeit. Dieses Phänomen soll im Folgenden etwas genauer betrachtet werden, womit sich auch dem Kerngegenstand der vorliegenden Arbeit angenähert wird.

3.2 Aufbereitung und Bewertung unstrukturierter Fileablagen

3.2.1 Fileablagen im Archivkontext

Unter einer unstrukturierten Fileablage oder Dateisammlung wird eine Menge genuin digitaler Daten nicht näher bestimmten Typs verstanden, die in einer ebenfalls nicht näher definierten Ordnung in einem Verzeichnissystem abgelegt sind und dem Archiv in ihrer Gesamtheit, beispielsweise auf einem tragbaren Wechsel-Datenträger, angeboten werden. Solche Ablagen entstehen abseits der geregelten Aktenführung auch in Behörden, vor allem aber sind sie typisch für nicht-amtliche Überlieferungen, beispielsweise für digitale Nachlässe.[48]
Für die amtliche Überlieferung gilt: Die in nicht ohne Weiteres nachvollziehbaren Prozessen und Kontexten entstandenen Daten wurden bezüglich ihrer Aussagekraft und Authentizität lange als zweifelhaft eingestuft und ihre Archivfähigkeit daher von Grund auf in Frage gestellt. Mittlerweile aber „haben diese Anforderungen an die Archivfähigkeit elektronischer Unterlagen zum Teil einer gewissen Ernüchterung Platz

45 Neben dem in Anm. 44 genannten Positionspapier zur Bewertung elektronischer Fachverfahren und dem Nestor Leitfaden zur digitalen Bestandserhaltung (Anm. 40) siehe auch *Schmidt*, Signifikante Eigenschaften und ihre Bedeutung für die Bewertung elektronischer Unterlagen; grundlegend zum Konzept der „Signifikanten Eigenschaften“ und deren Bedeutung: *Keitel*, Benutzerinteressen annehmen und signifikante Eigenschaften festlegen. Weitere Faktoren, die je nach Art der digitalen Information Einfluss auf eine geänderte Bewertung haben, finden sich bei: *Türck*, Veränderungen von Bewertungsgrundsätzen bei der Übernahme digitaler Unterlagen?, S. 16–29; Ein sehr guter Überblick über Stand und Probleme der Bewertung findet sich auch bei: *Bischoff*, Bewertung elektronischer Unterlagen und die Auswirkungen archivarischer Eingriffe auf die Typologie zukünftiger Quellen; vgl. außerdem: *Harvey*, DCC Digital Curation Manual.

46 Dies liegt nicht zuletzt an der diversen rechtlichen Situation (E-Government). Eine Regel dürfte aber zudem sein: Je größer das Archiv, desto größer die finanziellen und personellen Kapazitäten auch zur Umsetzung der digitalen Langzeitarchivierung. Daraus folgt vor allem für kleinere und mittlere Archive die Bedeutung von Verbundlösungen. Zu den damit wiederum verbundenen Problemen siehe beispielsweise: *Fischer*, Gemeinsame Lösungen für ein gemeinsames Problem.

47 Einblick in die Entwicklung, die hier nicht vollständig dargestellt werden kann, geben beispielsweise die online verfügbaren Unterlagen der seit 1997 jährlich stattfindenden Tagungen des Ak AUdS: Arbeitskreis „Archivierung von Unterlagen aus digitalen Systemen“. https://www.sg.ch/kultur/staatsarchiv/auds.html (aufgerufen am 13.10.2019).

48 Als Nachlassgeber kommen Privatpersonen oder Personen des öffentlichen Interesses in Frage. Diese Formen der Überlieferung hat beispielsweise auch schon Eingang gefunden in: Überlieferungsprofil „Nichtstaatliches Archivgut“, S. 6.

gemacht“[49]. Der praktischen Hinwendung zur langfristigen Archivierung solcher Daten stehen damit keine grundsätzlichen archivtheoretischen Vorbehalte mehr entgegen. In erster Linie wurde hier dem Umstand Rechnung getragen, dass trotz aller rechtlichen Vorschriften keine vollkommene Vereinheitlichung der bürokratischen Arbeitsorganisation erreicht wird, was bisher auch nicht von den im Rahmen der Digitalisierung eingeleiteten Bemühungen um eine intensivierte Zusammenarbeit von Archiven und Behörden aufgefangen wurde. Daher ist damit zu rechnen, dass Archive in Zukunft, trotz der Bemühungen um die geregelte Aktenführung, in größerem Umfang mit Dateisammlungen aus behördlichen oder institutionellen Kontexten konfrontiert sein werden. Ein gern angeführtes Beispiel sind schon jetzt E-Mails, die keinen Eingang in die reguläre papierne oder elektronische Aktenführung genommen haben; deren Ablagestruktur ähnelt in diesem Fall der Struktur anderer Fileablagen.[50] Für nicht-amtliche Überlieferungen ist eine unstrukturierte Überlieferung wie erwähnt ohnehin die Regel, sodass Archive auch hier mit entsprechenden Übernahmen rechnen müssen.

3.2.2 Aufbereitung und Bewertung von Fileablagen

Fileablagen bereiten Probleme, für die sich kaum einheitliche theoretische oder technische Lösungen entwickeln lassen. Dies liegt vor allem in folgenden Charakteristika begründet:[51]

Größe: Unübersichtliche Anzahl an Einzel-Dateien.
Nebeneinander archivwürdiger und nicht-archivwürdiger Dateien.
Keine standardisierte Ablage-Struktur (meist thematisch organisiert).
Hohe Komplexität aufgrund unterschiedlicher, z. T. unbekannter Dateitypen.

Fasst man die Faktoren nochmals zusammen, sind vor allem Größe und individuelle Struktur ausschlaggebend dafür, dass sich Aufbereitung- und Bewertungsprozesse genuin digitaler Dateisammlungen sehr aufwendig gestalten. Unter Aufbereitung- und Bewertungsprozessen werden hier diejenigen Arbeitsschritte verstanden, die bis zum Ingest in das digitale Archiv abzuarbeiten sind. Damit ist in erster Linie die Bewertung gemeint, jedoch rücken im Umgang mit unstrukturierten Daten auch Fragen der Bildung intellektueller Einheiten und der Erschließung in den Mittelpunkt des Vorgehens. Auf der technischen Ebene handelt es sich in erster Linie um das Vergleichen, Verschieben, Kopieren, Löschen, Bearbeiten und Validieren von Information. Strategien etablierter Vorgehensweisen lassen sich nicht ohne Weiteres auf unstrukturierte Fileablagen übertragen, da beispielsweise bereits die Abgrenzung einzelner intellektueller Einheiten Schwierigkeiten bereitet.[52] Ein zentrales Problem der existierenden Strategien besteht in der mangelnden Skalierbarkeit auf die spezifischen

49 *Bischoff*, Bewertung elektronischer Unterlagen und die Auswirkungen archivarischer Eingriffe auf die Typologie zukünftiger Quellen, S. 48.
50 Zur vorausgreifenden Zusammenarbeit vgl. beispielsweise: *Hering*, Ohnmächtig vor Bits and Bytes?, S. 93ff; zur Bedeutung einer aus archivischen Gesichtspunkten optimierten Ablagestruktur von Dateien in File-Systemen siehe: *Schludi*, Zwischen Records Management und digitaler Archivierung.
51 Diese Auflistung dient nur der groben Charakterisierung und ließe sich weiter ausdifferenzieren. Die einzelnen Merkmale fallen je nach Sammlung unterschiedlich ins Gewicht und können sich auch nur teilweise als relevant erweisen.
52 Der Begriff „Intellektuelle Einheit“ entstammt dem PREMIS-Datenmodell und meint hier die logische Einheit einer zusammenhängenden Menge von Inhalten. Vgl. PREMIS Data Dictionary for Preservation Metadata., S. 5f.

Anforderungen konkreter Dateisammlungen.[53] Anders als für Fachanwendungen lassen sich aufgrund der individuellen Ausformungen auch nur bis zu einem gewissen Grad allgemein anwendbare Software-Lösungen entwickeln, da ein sehr breites Funktionsspektrum abgedeckt werden muss.[54] Fileablagen gelten daher noch immer als „Albtraum heutiger Archivare".[55]

Im Kontext der Fotobewertung ist bereits angeklungen, dass in der vorliegenden Arbeit die archivische Bewertung zur Informationsverdichtung als notwendig erachtet wird. Dies gilt ausdrücklich auch für Fileablagen. Zum Teil wird eine Bewertung aufgrund der genannten Schwierigkeiten oder mit dem Verweis auf eine vermeintliche Verminderung der Authentizität abgelehnt. Auch immer weiter sinkende Speicherkosten werden als Argument gegen eine Selektion angeführt, wobei unberücksichtigt bleibt, dass auch die Menge an zu speichernder Information immer weiter ansteigt. Eine Vollarchivierung dürfte in den seltensten Fällen die beste Lösung sein.[56] Ziel sollte daher vielmehr eine softwaregestützte Vorgehensweise sein, die den Archivar entlastet, indem Bewertungs- und weitere Aufbereitungsprozesse so weit wie möglich verkürzt und wo möglich automatisiert werden. Sämtliche Schritte müssen jedoch intellektuell nachvollziehbar bleiben. Mit dem Wissen um die Anwendung geeigneter Programme und adaptierbarer Vorgehensweisen lässt sich – und darum geht es eigentlich – grundsätzlich viel Zeit einsparen. Der Ansatz eines software-gestützten Vorgehens wird beispielsweise am britischen Nationalarchiv verfolgt. Von dort stammt die bisher in dieser Form einzigartige Untersuchung zum Einsatz von eDiscovery-Software.[57] Darunter wird jene Software verstanden, die im Rahmen der Datenaufbereitung und Bewertung zum gezielten Aufspüren und Weiterverarbeiten relevanter Informationen genutzt werden kann. In erster Linie ist hier an textbasierte Suchen zu denken, gleichwohl der Einsatz geeigneter Software auch darüber hinaus, so mitunter durch die Implementierung bildbasierter Suchmechanismen, denkbar ist. Wie dies aussehen kann, wird später noch am konkreten Beispiel zu sehen sein.

Dort, wo von der ausführenden Person nicht mehr alle Prozesse intellektuell ausgeführt werden, beispielsweise im Rahmen (teil-)automatisierter Bewertungsprozesse, besteht jedoch immer das Risiko eines ungewollten Datenverlusts. So kommt auch das britische Nationalarchiv vorläufig zu dem Schluss: „No one product or provider can offer a fully automated process with a guarantee of 100 % accuracy."[58] Als grundsätzliche Eignungsvoraussetzung gilt daher auch, dass die Ergebnisse einer Automatisierung intellektuell korrigierbar sein sollten. So kann das Risiko des ungewollten Datenverlusts zumindest abgemildert werden. Der Einsatz von Automatisierungsprozessen bleibt aber in jedem Fall einem Risikomanagement unterworfen. Letztlich läuft dieses auf die Abwägung des Verlustrisikos gegen die Finanzierbarkeit einer mutmaßlich zuverlässigeren intellektuellen Vorgehensweise hinaus.[59]

53 Vgl. beispielhaft: *Harvey*, DCC Digital Curation Manual, S. 11.
54 Sehr treffend erscheint daher auch in Abgrenzung zur vordefinierten, strukturierten Ablage die Bezeichnung „kreative digitale Ablage", wie sie auf der 2016 in München stattgefundenen Konferenz „Kreative digitale Ablagen und die Archive" geprägt wurde. Siehe hierzu: *Naumann* und *Puchta* (Hg.), Kreative digitale Ablagen und die Archive.
55 *Taylor*, Eine hydraartige Matrjoschka, S. 7.
56 Vgl. auch die verschiedenen Bewertungsansätze bei: *Gilliland*, Archival appraisal: practising on shifting sands, besonders S. 35–37.
57 Hierin werden Ziele und Herausforderungen eines software-gestützten Vorgehens zusammengefasst: The National Archives, The application of technology-assisted review to born-digital records transfer.
58 Ebenda, S. 26.
59 Vgl. ebenda; Vgl. ebenfalls *Taylor*, A Hydra-like Russian Doll.

Das Argument des unbeabsichtigten Datenverlusts kann sicherlich auch gegen ein auf den ersten Blick womöglich befremdlich erscheinendes Vorgehen eingewandt werden: Zufallsselektion statt Bewertung.[60] Die Autoren Neumayer und Rauber vertreten die bewusst provokativ formulierte Ansicht, dass die Zufallsauswahl bei einer digitalen „collection of sufficien size"[61] nicht nur einfacher und kostensparender sei, sondern auch zu weniger subjektiven und damit insgesamt besseren, authentischeren Ergebnissen führe als eine klassische intellektuelle Bewertung. Die Frage nach einer ausreichenden Sammlungsgröße ist vielleicht nicht leicht zu beantworten und ob das Ausschalten der subjektiv bewertenden Person mehr wiegt als ein durch den Zufall bedingter Verlust wichtiger Information ist im Einzelfall zu entscheiden. Die Autoren haben zumindest aber genau dann eine Reihe an Argumenten auf ihrer Seite, wenn eine intellektuelle Bewertung aufgrund der Sammlungsgröße schlichtweg nicht in Frage kommt. Durch eine Zufallsauswahl nach dem Muster „nehme jedes n-te Element" besteht im Gegensatz zu herkömmlichen Bewertungsstrategien auch kein Skalierungsproblem. Einschränkend ist jedoch anzuführen, dass die Homogenität einer Sammlung ganz wesentlich darüber entscheidet, wie zuverlässig ein zufallsbasiertes Sampling ausgeführt werden kann. Die Sammlung bedarf zur Bestimmung eines geeigneten Vorgehens in jedem Fall einer entsprechenden Analyse.[62] Allein zwei Faktoren bestimmen jedoch letztlich darüber, wie viele Elemente übersprungen werden sollen: Sammlungsgröße und Speicherkapazität des digitalen Archivs.

Eine Zufallsauswahl ist technisch leicht umsetzbar und bedarf im Gegensatz zur intellektuellen Bewertung nur sehr geringer Ressourcen. Und schließlich ist auch das Argument der Objektivität nicht ganz von der Hand zu weisen. Zudem ist das Verfahren als einziges statistisch valide nachvollziehbar und auswertbar. Vor allem aber gewinnt es im Umgang mit digitalen Sammlungen genau dann an Berechtigung, wenn es in Kombination mit herkömmlichen Bewertungsstrategien eingesetzt wird.[63] So kann die Zufallsauswahl in einem ersten Schritt zur Reduzierung der Sammlungsgröße genutzt werden. Das Ergebnis dieser Reduktion kann in einem zweiten Schritt, sei es durch intellektuelle oder weitere automatisiert realisierte Bewertungsprozesse, verfeinert und im Zweifelsfall korrigiert werden. Wie dies im Konkreten aussehen kann, zeigt der Workflow in der vorliegenden Arbeit. Mit Terry Cook bleibt abschließend festzuhalten: „Sampling is a powerful tool […] but should be used sparingly and only when all the conditions for statistical validity can be met and all other appraisal options have been considered first."[64]

3.2.3 Der aktuelle Stand: Praktische Erfahrungen im Umgang mit Fileablagen

Wie bereits angedeutet, gibt es bisher nur wenige konkrete Erfahrungen im Umgang mit unstrukturierten Fileablagen, auf die zurückgegriffen werden könnte. Zumindest ist die Liste entsprechender Publikationen sehr überschaubar. Dies gilt im Übrigen nicht nur für den deutschen, sondern auch den europäischen und

60 Siehe *Neumayer* und *Rauber*, Why Appraisal is Not 'Utterly' Useless and Why It's Not the Way to Go Either.
61 Ebenda, S. 2.
62 Zum Sampling auf Zufallsbasis nach dem Muster „jedes n-te Element" siehe auch die Ausführungen zum „systematic random sampling" bei: *Cook*, „Many are called, but few are chosen", S. 38. Cook nennt auch zwei weitere Sampling-Ansätze, auf die hier jedoch nicht näher eingegangen werden soll.
63 Im Sinne eines derart kombinierten Vorgehens werden auch die Schlussfolgerung der Autoren verstanden. Vgl. *Neumayer* und *Rauber*, Why Appraisal is Not 'Utterly' Useless and Why It's Not the Way to Go Either, S. 2.
64 *Cook*, "Many are called, but few are chosen", S. 39.

internationalen Archiv-Sektor.[65] Daher wird hier eine Zusammenschau der einschlägigen internationalen und nationalen Erfahrungen gegeben.

Ein früher Beitrag von Mumma, Dingwall und Bigelow vom Stadtarchiv Vancouver erschien bereits 2011. Die Autoren waren an der Archivierung der vom Organisationskomitee der Olympischen und Paralympischen Winterspiele 2010 produzierten Unterlagen beteiligt. Dabei handelte es sich um analoge Unterlagen, aber vor allem ein 25 TB umfassendes Datenkonvolut, das auf verschiedenen Systemen und Datenträgern produziert und übernommen wurde.[66] Die Autoren entwarfen einen an INTERPARES 3 anknüpfenden dreigeteilten Bewertungsansatz, der besonders den Evidenzwert der Unterlagen berücksichtigt. Unter anderem aufgrund der immensen Größe der Sammlung sowie der Verwendung des Archivsystems Archivematica ist der Beitrag für den praktischen Ansatz der vorliegenden Arbeit insgesamt aber wenig bedeutend. Ohnehin geht der Beitrag in seiner Kürze kaum auf konkrete Lösungen ein, gibt jedoch einen vergleichsweise allgemeinen Überblick über Herausforderungen, die sich auch bei der Bearbeitung kleinerer Fileablagen stellen und fasst einzelne relevante Schlussfolgerungen zusammen. So nennen die Autoren beispielsweise allgemeingültige Faktoren, die beim Sampling zu berücksichtigen sind: 1. Konsistenz der Ablagestruktur. 2. Homogenität der Dateitypen. 3. Größe der Sammlung. Im Übrigen äußern sie abschließend zudem etwas konsterniert, dass sie trotz einer über sechs Jahre währenden Zusammenarbeit mit dem Organisationskomitee nur minimalen Einfluss auf dessen Aktenführung erzielen konnten.

Auch die von Uglean Jackson und McKinley veröffentlichte Fallstudie zur Archivierung einer 50.000 Dateien umfassenden AV-Sammlung mit einer Größe von 2,5 Terabyte bleibt relativ oberflächlich. Zwar wird auf die Verwendung von Tools wie BagIt und FITS (File Information Tool Set) oder die Möglichkeit einer Excel-basierten Analyse der Verzeichnisstrukturen und Dateiformate auf Basis eines Tools namens Karen's Directory Printer hingewiesen. In erster Linie betonen die Autoren jedoch den hohen Aufwand und die Notwendigkeit einer schrittweisen Vorgehensweise sowie skalierbarer Strategien, weshalb die Studie aus heutiger Sicht nur wenige praxisrelevante Hinweise gibt.[67]

Ungefähr zur gleichen Zeit veröffentlichte Michael Shallcross[68] erste Materialien zur Bearbeitung digitaler Unterlagen („Born-Digital Materials"), die bereits wesentlich konkreter auf die Bearbeitung von Dateisammlungen Bezug nehmen. Anhand dieser und einzelnen nachfolgenden Veröffentlichungen lässt sich außerdem das Bestreben einer sukzessiven Automatisierung des Umgangs mit digitalen Unterlagen nachverfolgen. Wie zu sehen sein wird, erweist sich aber auch die Publikation zur manuellen Bearbeitung noch immer als hilf- und einflussreich.[69] Zum einen entwirft Shallcross hier einen Workflow, der auf das Prinzip „More Product, Less Process"[70] zurückgreift und in erster Linie auf eine tool-basierte Analyse der Dateisammlung

65 Bibliotheken sind hier einzuschließen, da sie seit Beginn der elektronischen Archivierung eine wichtige Rolle spielen.
66 *Mumma*, *Dingwall* und *Bigelow*, A First Look at the Acquisition and Appraisal of the 2010 Olympic and Paralympic Winter Games Fonds.
67 *Uglean Jackson* und *McKinley*, It's How Many Terabytes?! Die Autoren waren zum Zeitpunkt der Veröffentlichung an der University of California Irvine und der California Digital Library beschäftigt.
68 Archival Assistant an der Bentley Historical Library (Michigan, USA).
69 *Shallcross*, Bentley Historical Library Guidelines for the Manual Processing of Born-Digital Materials.
70 Vgl. Anm. 10; Vgl. auch den folgenden ebenfalls 2012 erschienen Beitrag, der zeigt, wie sehr im Anfang begriffen die Suche nach Ansätzen zu dieser Zeit war: *Phillips*, More Product, Less Process for Born-Digital Collections.

abzielt, bei der weniger die einzelne Datei als eine Batch-Behandlung vieler Dateien im Vordergrund steht. Der Workflow deckt den gesamten Ingest-Prozess ab und stellt beispielsweise auch die Erstellung von Logfiles zur Metadatengenerierung heraus. Rezipiert wurden aber insbesondere auch die verwendeten Tools wie TreeSize Professional oder IrfanView und auch die vorliegende Arbeit integriert diese in den Workflow. An der Bentley Historical Library sah man diese Toolsammlung jedoch lediglich als kurzfristige Lösung an. Daher wurden 20 dieser Werkzeuge in einen automatisierten Prozess namens „AutoPro“, eine Sammlung von 33 Windows-Shell-Skripten, integriert.[71] Damit konnte der ursprüngliche manuelle Workflow von 40 Schritten auf nurmehr 11 Schritte reduziert werden. Logfiles zu verschiedenen Prozessen, beispielsweise Ordner- und Dateiumbenennungen, wurden in CSV-Dateien abgelegt. Für die vorliegende Arbeit stellt der skriptgesteuerte Ansatz eine zentrale Inspirationsquelle dar, obschon nicht in dieser Form auf die Implementierung verschiedener Tools abgezielt wird. 2014 startete man an der Bentley Historical Library zusammen mit der Hauptbibliothek der Universität Michigan ein Projekt zur “ArchivesSpace – Archivematica – DSpace Workflow Integration”, womit man sich von der losen Skriptsammlung hin zu einer unter einem Dach vereinten Lösung zuwandte, was dann zur Entwicklung des „Appraisal and Arrangement tab“ für Archivematica führte. Die Ergebnisse nutzen somit in erster Linie Anwendern dieser Archivlösung.[72]

An der britischen Wellcome Library wird gegenwärtig an der Entwicklung skalierbarer Workflows zur Aufbereitung digitaler und hybrider Sammlungen gearbeitet. Victoria Sloyan hat in diesem Rahmen zwei Festplatten aus dem Nachlass zweier Wissenschaftler bewertet und ihre Ergebnisse vergleichend nebeneinandergestellt.[73] Sowohl die Bewertung als auch den Umgang mit sensiblen Daten („sensitivity review“) stellt Sloyan als zentrale Aufgaben heraus, wobei beide Prozesse Hand in Hand gehen. Bemerkenswert an der entwickelten Lösung ist die Kombination verschiedener Bewertungsansätze, die von einer Macro-Bewertung größerer Teile bis hin zu einer granularen Bewertung auf der Ebene einzelner Dateien reicht, dabei sowohl evidenz- als auch informationsbezogene Kriterien in den Blick nimmt und durch den Einsatz von DROID auch technische Bewertungsmethoden – beispielsweise zur Auswahl bestimmter Dateiformate und zur Duplikatsuche – integriert. Nicht zuletzt findet ein randomisiertes Sampling auf Basis eines einfachen Online-Zufallszahlengenerators Anwendung. Zwar seien nicht alle Probleme gelöst worden, jedoch habe sich der Workflow als skalierbar und effizient erwiesen; trotz der signifikant unterschiedlichen Strukturierung und Homogenität der beiden Nachlässe.

Wenn auch der Workflow auf technischer Ebene nicht über die Verwendung von DROID hinausgeht und kaum Ansätze zur Automatisierung verfolgt werden, zeigt Sloyan, wie mit einer Kombination unterschiedlicher Strategien eine Bewertung unstrukturierter digitaler Sammlungen möglich ist, ohne mit der bisherigen Bewertungspraxis zu brechen.

71 *Shallcross* und *Deromedi*, Automated Digital Processing at the Bentley Historical Library.
72 *Shallcross*, Appraising Digital Archives with Archivematica; zum Appraisal and Arrangement Tab siehe: *Shallcross*, The End is Just a New Beginning!
73 *Sloyan*, Born-digital archives at the Wellcome Library. Die Datenträger hatten einen Umfang von 4.559 Dateien (5,6 GB) und 16.378 Dateien (12,7 GB).

In Deutschland ist eine Hinwendung zu der Thematik erst in jüngster Zeit zu beobachten, wenn auch das Thema beispielsweise auf dem Südwestdeutschen Archivtag 2012 bereits berührt wurde. Vor allem durch den Beitrag von Kemper und Naumann, der unter anderem die Bedeutung von Verzeichnisnamen zur Metadatengenerierung hervorhebt und die effiziente Verarbeitung dieser Daten mit MS Access sowie den Export von XML- oder CSV-Dateien zum anschließenden Transport dieser Daten nahelegt.[74] In erster Linie ging es jedoch noch um die erste Annäherung an ein für die meisten kleineren und mittleren Archive neues Themenfeld, das hier sehr sprechend als „neues Handwerk" begriffen wurde.

Ein großes Projekt startete ebenfalls bereits 2012 am Deutschen Literaturarchiv Marbach, wo man sich an Erschließung des 1.2 TB großen und sehr komplexen digitalen Nachlass des Literatur- und Medienwissenschaftlers Friedrich Kittler machte. Ergebnis dieser Arbeiten war ein Tool namens „Indexer", das sämtliche Daten indiziert – einschließlich diverser Metadaten und eines Volltextindexes – und diese Informationen über ein Web Frontend für die bewertende Analyse zur Verfügung stellt. Die Entwicklung dieses Tools scheint insgesamt weit vorangeschritten zu sein. So war offenbar war auch eine Implementierung in die BitCurator-Suite angedacht, jedoch liegen hierzu wie zur Nachnutzbarkeit im Allgemeinen keine weiteren Informationen vor.[75]

Eine breitere Auseinandersetzung mit Dateisammlungen fand erst 2016 auf der Tagung „Kreative digitale Ablagen und die Archive" statt.[76] Zum einen fanden sich hier Beiträge zum Entwicklungsstatus spezieller Querschnittsanwendungen wie Bytebarn, Package Handler und PIT.[77] Die Beiträge von Miegel, Schmidt, Schieber (Landesarchiv Hessen) sowie Knobloch (Landesarchiv Baden-Württemberg) und Birn (Kreisarchiv Reutlingen) lieferten darüber hinaus aber auch konkrete Anregungen zum Einsatz von Software wie Total Commander oder Treesize Professional – deren Verwendung bereits von Shallcross beschrieben wurden – und weiteren Tools sowie kommandozeilenbasierten Werkzeugen. Teilweise wurden diese in erste Workflow-Entwürfe integriert. Auf eine konkrete Dateisammlung Bezug nahm hier der Beitrag von Susanne Belovari.[78] Anhand der 677 GB (65057 Dateien) großen Dateisammlung einer Schule entwickelte Belovari einen einfachen MPLP-Workflow, der explizit für klassische Archivare und Archivarinnen auch ohne größere Einarbeitungszeit ausführbar sein sollte.[79] Die unkomplizierte Handhabung der eingesetzten Werkzeuge stand dabei im Mittelpunkt, ebenso die Beschränkung auf einige wenige Tools. Besonders TreeSize Professional stellte sich in der vergleichenden Handhabung als geeignet heraus. Belovari nutzte das Tool unter anderem zur Beseitigung von Duplikaten (bei einem Duplikatanteil von 70 %) und reduzierte die Sammlung innerhalb von vier Tagen auf 74,2 GB. Anschließend wurde die Sammlung in das digitale Archivsystem des Landesarchivs Baden-Württemberg DIMAG eingespielt.

74 *Kemper* und *Naumann*, Selbermachen! Vgl. hier weiterhin die bereits zitierten Beiträge von *Schludi* und *Wolf* (Anm. 15 und Anm. 50). Der Ansatz der Metadatengenerierung aus der Ordnerstruktur wurde bereits vom Paradigm-Project vorgeschlagen und findet auch bei Taylor Berücksichtigung. Vgl. Paradigm, Arranging and cataloguing digital and hybrid archives.
75 *Enge* und *Kramski*, „Arme Nachlassverwalter…".
76 Auf den 2017 erschienenen Tagungsband wurde bereits verwiesen. Vgl. Anm. 54.
77 Vgl. die Beiträge von *Huth*, *Naumann* und *Straßenburg* in: Ebenda.
78 Die Autorin vom Universitätsarchiv der Universität Illinois war 2016 drei Wochen am Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Ludwigsburg.
79 Vgl. Anm. 10 u. Anm. 70.

Abschließend ist hier noch mal auf das Projekt von Isabel Taylor (Generallandessarchiv Karlsruhe) hinzuweisen.[80] Ihre Bewertung einer Fileablage aus einem Schulamt (16 GB, 57.000 Dateien) verfolgte zunächst mit einer Mischung aus evidenz- und informationswertbezogener Analyse im Prinzip einen ähnlichen Ansatz wie der gleichzeitig durchgeführte Bewertungsansatz der Wellcome Library, jedoch verbot die hohe Komplexität (z. B. 15 Unterordner) der Sammlung ein vergleichbares Vorgehen, weshalb Taylor schließlich über acht Monate hinweg eine Einzeldateibewertung vornahm. Auch Taylor setzte erfolgreich verschiedene Tools ein, kommt aber schließlich zum Ergebnis, dass das Vorgehen zu aufwendig gewesen sei und meldet aufgrund nicht-dokumentierter Löschungen auch Bedenken bezüglich der Authentizität der verbliebenen Daten an. Gleichzeitig aber zweifelt Taylor an einer Anwendbarkeit des von der Wellcome Library entwickelten Workflows auf komplexere Dateisammlungen. Zur Bewertung komplexerer Fileablagen hält die Autorin hingegen die Einbeziehung neuer Technologien wie eDiscovery-Software für einen erfolgversprechenderen Ansatz.

Insgesamt zeigen die Beiträge nicht nur eine zunehmende praktische Hinwendung zur Bewertung digitaler Sammlungen, sondern zum Teil auch die Anwendung ähnlicher oder gleicher Werkzeuge. So scheint sich der Einsatz von Tools wie TreeSize Professional zu bewähren und durch den Erfahrungsaustausch allmählich durchzusetzen. Gleichzeitig wird – man vergleiche die Beiträge Sloyans und Taylors oder dem MPLP-Ansatz Belovaris mit der Entwicklung des „Indexers“ am DLA Marbach – deutlich, mit welch unterschiedlichen Sammlungen und individuellen Problemen einerseits umgegangen werden muss, andererseits angesichts der verfügbaren technischen Expertise und des zu leistenden Aufwands umgegangen werden kann. Bis auf Weiteres sind daher insgesamt keine Patentrezepte zu erwarten. Es scheint aber insgesamt an automatisierbaren Prozessen zu mangeln, die eine Verringerung des Aufwands bedeuten würden, wobei sich hierin das zentrale Dilemma spiegelt. Je komplexer die Sammlung, je notwendiger im Sinne der Nutzbarkeit also auch eine Bewertung, desto aufwendiger die Aufbereitung und desto mehr Zeit würde folglich eine Automatisierung einsparen. Umgekehrt lassen sich Automatisierungsansätze leichter auf weniger komplex, homogene Sammlungen anwenden, wie dies auch für den Einsatz eines Zufalls-Samplings gilt. Der nun im Folgenden präsentierte Workflow versucht eine Synthese und Weiterentwicklung der bisherigen Erfahrungen, gleichwohl auch hier auf die individuellen Anforderungen reagiert werden musste.

80 Vgl. Anm. 55 und Anm. 59.

4. Aufbereitung und Bewertung eines digitalen Fotonachlasses

4.1 Zur Fotosammlung Aipperspach[81]

Der Sigmaringer Gunter Aipperspach begann in den 1970er Jahren als Amateurfilmer, bevor er in den 80er Jahren das Fotografieren aufnahm. Bis heute hält er in dokumentarischer Absicht die Gegenwart seiner Umgebung bildlich fest. Dabei bilden Sigmaringen und seine Teilorte, das dortige Alltagsgeschehen, Ereignisse, Brauchtum und religiöse Feste einen Schwerpunkt, jedoch finden sich unter seinen Aufnahmen auch Bildserien einzelner umliegender Gemeinden und von Urlaubsreisen. Nach einer zweijährigen Übergangsphase arbeitet Aipperspach seit 2005 ausschließlich digital. 2007 begann er mit der Erweiterung seines Aufnahmespektrums um Luftaufnahmen aus dem Kleinflugzeug. Bereits 2011 wurde die Übernahme der Fotosammlung Aipperspach in das Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen vereinbart. Neben 59 Schmalfilmen des Formats Super 8 übernahm das Archiv fünf Fotoboxen mit ca. 2000 Diapositiven aus den Jahren 1986 bis 1995, die Aipperspach außerdem zuvor digitalisiert und mit Kontextinformationen (Titel, Karten- und Zeitungsausschnitte) versehen hatte.

Gegenstand der vorliegenden Arbeit ist die Sammlung von rund 200.000 Digitalfotos im JPEG-Format, die Aipperspach dem Staatsarchiv Sigmaringen im September 2016 auf einer externen Festplatte übergab. Die thematisch in einzelne Unterordner gegliederte Sammlung ist eine chronologische Reihung seiner Fotografien aus den Jahren 2003 bis 2016, die Aipperspach um von ihm als privat empfundene Aufnahmen reduziert hat. Neben der Dokumentation des Lebens in Sigmaringen finden sich Reise-Bilder aus Baden-Württemberg, weiteren Teilen Deutschlands und dem überwiegend europäischen Ausland (Großbritannien, Israel, Niederlande, Polen, Tschechien). Wie schon die Digitalisate der Dia-Sammlung sind die Serien mit Texterläuterungen und Zeitungsausschnitten angereichert.

Der archivische Wert der Sammlung Aipperspach besteht für das Staatsarchiv Sigmaringen in der visuellen Dokumentation von Landschaft, Besiedelung, Stadtansichten, markanten Gebäuden und Alltagsgeschehen, wodurch die schriftliche Überlieferung im Staatsarchiv ergänzt wird. Zudem liegt keine Parallelüberlieferung einer vergleichbaren fotografischen Sammlung vor, was den Wert der Aufnahmen noch einmal erhöht. Über die zeitliche Spanne der Aufnahmen hinweg lassen sich Änderungsprozesse nachvollziehen. Die begleitende Dokumentation ermöglicht die Identifizierung einzelner Motivgruppen und enthält zusätzliche Informationen. Daher ist die Sammlung dauerhaft zu erhalten, allerdings bedarf es hierzu einer sinnvollen Reduzierung. Die Arbeitsweise des Fotografen soll dabei nachvollziehbar bleiben. Charakteristisch hierfür sind insbesondere die Bilder mit Texteinblendungen (im Folgenden als Tafeln bezeichnet) und die beigefügten Zeitungsausschnitte. Zum Teil finden sich auch Fotos mit Texteinschreibungen.

81 Die Ausführungen beruhen auf Informationen von Sybille Brühl vom Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen.

Die erforderlichen Verwertungsrechte liegen dem Staatsarchiv vor. Aufnahmen fremder Fotografen sind davon unberührt und sollen daher kassiert werden.

4.2 Ausgangslage und Zielsetzung

Die Fotosammlung Aipperspach wurde zur automatisierten Bewertung in Kopie auf einer externen Festplatte entgegengenommen. Im Gespräch mit Kai Naumann vom Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Ludwigsburg und Sybille Brühl vom Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen, die die Sammlung von Herrn Aipperspach übernommen und bereits einer Vorbewertung unterzogen hatte, wurden die Ziele des weiteren Vorgehens besprochen. Anlass war die notwendige Reduzierung der Sammlungsgröße, die aufgrund des immensen Aufwands nicht allein durch ein intellektuelles Bewertungsverfahren erzielt werden konnte. Als Richtwert wurde die Reduktion des Umfangs auf etwa 10 % der ursprünglichen Sammlungsgröße von rund 200.000 Fotos vereinbart, da das Archiv nicht die vollständige Sammlung übernehmen könne. Mit über 20.000 Bildern würde so jedoch immer noch ein mehr als zehnmal so großer Bestand angelegt werden als im Fall der 2000 Diapositive von Aipperspach, die bereits im Jahr 2011 übernommen wurden. Da von Seiten des Archivs keine Erfahrungen im Umgang mit einer derart großen digitalen Fotosammlung vorlagen, sollte ein Workflow entwickelt werden, der die nötigen Schritte zur Aufbereitung und Reduzierung der Sammlung unter Berücksichtigung der vorgegebenen Bewertungskriterien so weit wie möglich automatisieren würde. Am Ende dieses Prozesses sollte der Ingest der Primärdaten in das Digitale Archivsystem DIMAG sowie der Erschließungsmetadaten in das Archivische Informationssystem Scope-Archiv stehen.

Die grundsätzlichen Bewertungskriterien sind bereits der obigen Beschreibung zu entnehmen. Gemäß dem Sammlungsprofil des Staatsarchivs Sigmaringen sollten Fotos von Urlaubsreisen und sonstigen Ausflügen, die Aipperspach außerhalb Baden-Württembergs aufgenommen hatte, grundsätzlich nicht übernommen werden. Als Ausnahme wurde die in der holländischen Gemeinde Boxmeer gemachten Aufnahmen vereinbart. Ebenso sollten Aufnahmen dritter Fotografen, für die keine rechtlichen Vereinbarungen vorlagen, kassiert werden. Um die Sammlung darüber hinaus auszudünnen, sollte ein Samplingverfahren erarbeitet werden. Bereits hier wurde die Entwicklung eines Zufallsfilters ins Auge gefasst, der sowohl auf Einzelbildebene als auch auf Ordnerebene zur Ausdünnung der Sammlung zum Einsatz kommen könnte. Zudem wurde die Löschung von Duplikaten, Quasi-Duplikaten und unscharfer Fotos als mögliche Operationen bestimmt.

Die Organisation der Sammlung wurde von Herrn Aipperspach ohne spezielle Fotoverwaltungs-Software über den Windows-Explorer vorgenommen. Wesentliches Strukturmerkmal der Ablage waren chronologisch angelegte Ordner in denen sich in der Regel die Bilddateien befanden. Mit Serie – in der Folge auch als Ereignis bezeichnet – ist immer der Inhalt dieser Hauptordner gemeint (F:\Aipp\Ereignis\).[82] In einigen Fällen

82 Alle Ordner dieser Ebene folgen dem gleichen Bezeichnungsmuster. Ein Beispiel: F:\Aipp\37 Rathausplatzkonzert in Sigmaringen mit d. Stadtkapelle SIG am 12.6.05.

gliederte sich die Ordnerstruktur in weitere Unterverzeichnisse auf. Innerhalb der einzelnen Verzeichnisse folgten die Bilddateien einer fortlaufenden Nummerierung. Eine Unterteilung der einzelnen Serien realisierte Aipperspach durch eingeschobene Tafeln: Bilder mit einfarbigem Hintergrund, in die er erläuternde Texte zu den nachfolgenden Fotos eingesetzt hatte. Ebenso befanden sich zwischen den Fotos Karten- und Zeitungsauschnitte, die zusätzliche Kontextinformationen lieferten. Besonders auch bei den ersten Bildern eines jeden Verzeichnisses handelte es sich um Tafeln, Karten- oder Zeitungssauschnitte. Aufgrund des Informationswerts dieser Bilder, aber auch weil sich in dieser Ablage die Arbeitsweise Aipperspach spiegelte, sollten diese nach Möglichkeit erhalten werden.

Die Aufbereitung der Sammlung sollte über die Bewertung hinaus die SIP[83]-Formierung umfassen und auch die Aufbereitung der beim Ingest per Lookup-Datei bereitzustellenden Metadaten einschließen. Demnach sollte die Sammlung in Teilen umstrukturiert und die Ordnerbezeichnungen einer Redaktion unterzogen werden. Zum weiteren Vorgehen wurden keine konkreten Vorgaben oder Einschränkungen vereinbart. Der Einsatz eines bestimmten Betriebssystems – MS Windows, macOS oder einer Linux-Distribution – sowie bestimmter Tools wurde nicht näher eingegrenzt. Jedoch wurde die Realisierung der Kernfunktionalitäten über eine relationale Datenbank – beispielsweise MS Access oder MariaDB – erörtert, mit der sich die notwendigen Informationen und Bewertungsentscheidungen organisieren ließen. Einige Aufgaben wären so mithilfe geeigneter Tools, andere durch gezielte Abfragen in der Datenbank zu realisieren gewesen und die Ergebnisse hätten ebenfalls in der Datenbank hinterlegt werden können, um am Ende per Kommandozeile das Kopieren der als archivwürdig bewerteten Dateien in ein separates Verzeichnis vorzunehmen.

Gemeinsam wurde abschließend eine Testsuite zusammengestellt. Hierzu wurden zehn Ordner exemplarisch ausgewählt, die charakteristische Merkmale – wie z.B. besondere Größe, mehrere Unterordner, zwischen den einzelnen Fotos verteilte Tafeln – aufwiesen. Anhand dieser Testsuite sollte der Workflow entwickelt und erprobt werden.[84]

4.3 Erste Analyse-Schritte

Nach der Einarbeitung in den archivischen Umgang mit Foto- und Dateisammlungen sowie der Klärung der Ausgangslage und der genauen Zielsetzung wurde entschieden, die Aufbereitung der Fotosammlung unter Microsoft Windows vorzunehmen, wenn auch aufgrund rein persönlicher Präferenzen zunächst eine Unix-Umgebung vorgezogen wurde. Damit sollte der verbreiteten Verwendung von Windows-Bürorechnern, auf denen der Workflow umsetzbar sein sollte, entsprochen werden. Zudem erschien so auch der Rückgriff auf bereits bekannte und bewährte Tools wie Total Commander, TreeSize Pro und IrfanView möglich.[85] Durch die anfänglich im Zentrum des Aufbereitungsprozesses stehende SQL-Datenbank wäre eine teilweise Übertragung des Workflows auf andere Systeme allerdings prinzipiell möglich. Dieser Punkt mag nebensächlich

83 Die Abkürzung ist im Sinne des OAIS-Referenzmodells zu verstehen: SIP: Submission Information Package. Dort auch in Abgrenzung zum AIP: Archival Information Package. Die Bezeichnungen werden in der vorliegenden Arbeit als identische Einheiten verstanden, da die zunächst bei der Aufbereitung gebildeten SIPs beim späteren Ingest in das digitale Archiv zu AIPs formiert werden.

84 Im Übrigen zeigte sich auch Herr Aipperspach einverstanden mit der Vorgehensweise.

85 Die Verwendung der Tools wurde bereits in der Literatur beschrieben, weshalb hier nicht näher darauf eingegangen werden soll. Zu IrfanView und TreeSize siehe: *Shallcross*, Bentley Historical Library Guidelines for the Manual Processing of Born-Digital Materials, S. 9–12. Zur Verwendung von Total Commander und TreeSize siehe die Folien von Marco *Birn*. In: *Naumann* und *Puchta* (Hg.), Kreative digitale Ablagen und die Archive.

erscheinen, wurde jedoch trotz der Entscheidung für ein Windows System weiterhin berücksichtigt, da besonders im Falle von Nachlässen damit zu rechnen ist, dass die Nachlasser entgegen der weiten Verbreitung von Windows mit Linux- oder Mac-Systemen gearbeitet haben. Somit kann es sich durchaus als Vorteil erweisen, wenn die Aufbereitung solcher Daten ebenfalls auf solchen Systemen durchführbar ist.[86]

Der Einrichtung des Bearbeitungsrechners folgten der Reihe nach grundsätzliche Analyse-Operationen, von denen einige – Virenprüfung, Bestandsaufnahme mit ID oder Prüfsummen, Anlegen einer Kopie – mit jeder Dateisammlung unmittelbar nach der Übernahme durchgeführt werden sollten. Da sich hier inzwischen ähnliche Vorgehensweisen etabliert haben dürften, sei das Vorgehen lediglich kurz skizziert. Zunächst wurde die Sammlung auf Viren überprüft und mithilfe von TreeSize Pro eine Bestandsaufnahme sämtlicher Verzeichnisse und Total Commander eine Bestandsaufnahme sämtlicher Dateipfade inklusive MD5-Prüfsummen erstellt. Die Bestandssaufnahmen wurden als CSV-Datei abgelegt und in die eingerichtete SQL-Datenbank[87] eingespielt, die nach wie vor als Grundbaustein des weiteren Vorgehens angesehen wurde. Zudem wurden die Bilddaten auf IPTC-Metadaten überprüft, solche waren jedoch keine hinterlegt. Mit Exiftool wurden die technischen Metadaten sämtlicher Bilddateien extrahiert.

Dateitypen		
Dateiendung	**Größe**	**Dateien**
Grafik-Dateien	449,2 GB	210.190
„.jpg“	449,1 GB	210.189
„.tif“	27,3 MB	1
Datenbankdateien	1,5 GB	1.486
„.db“	1,5 GB	1.486
Sonstige Dateien	1,9 MB	16
„(keine)“	1,4 MB	4
„.mxc2“	12,7 KB	2
„.1“	329,6 KB	2
„.2“	51,2 KB	2
„.3“	8,6 KB	2
„.4“	25,4 KB	2
„.5“	42,7 KB	1
„.6“	2,9 KB	1

86 So beispielsweise im Fall des Nachlasses von Friedrich Kittler, der mit Gentoo-Linux arbeitete. Vgl. *Enge* und *Kramski*, „Arme Nachlassverwalter…“.
87 Verwendet wurde MariaDB. Wenn im Folgenden SQL genannt wird, ist diese Datenbank gemeint.

Dateitypen		
Dateiendung	**Größe**	**Dateien**
Video-Dateien	950,6 MB	11
„.mov“	542,9 MB	10
„.wmv“	407,8 MB	1
Office Dateien und Dokumente	86,9 MB	7
„.doc“	73,6 MB	4
„.pdf“	13,3 MB	3
Konfigurationsdateien	277 Bytes	3
„.ini“	277 Bytes	3
Text Dateien	4,8 KB	1
„.log“	4,8 KB	1

Abb. 1: Dateizusammensetzung der Sammlung Aipperspach.

Nach diesen Vorbereitungen wurden Verzeichnisstruktur und Dateizusammensetzung genauer analysiert. Hierzu erwies sich ebenfalls TreeSize Pro als hilfreiches Werkzeug. Die Analyse gab folgende Eck-Daten, die genaue Dateizusammensetzung ist dem Anhang zu entnehmen (vgl. Abb. 1):

Datenvolumen:	451 Gigabyte
Verzeichnisse:	1664
Unterverzeichnisse:	495 in 81 Verzeichnissen
Max. Verzeichnistiefe:	3
Dateien:	211714
Dupletten (MD5):	46945
Max. Pfadlänge:	257 Zeichen

Abschließend wurde mit Total Commander eine Arbeitskopie der Sammlung angelegt – das Tool eignet sich gegenüber dem einfachen Kopieren im Windows-Explorer besonders durch die parallele Verifizierung des Kopiervorgangs, womit sichergestellt wird, dass die Daten unverändert bleiben. Ebenso erfolgte das Anlegen der Testsuite aus den zehn zuvor bereits bestimmten Ordnern.

4.4 Schlussfolgerungen für die weitere Vorgehensweise

Die Sammlung Aipperspach wies – auch im Vergleich zu den bisher in der Literatur thematisierten Nachlässen (siehe 3.2.3) – ein hohes Maß an Systematik und Homogenität auf. Ausgehend vom Hauptverzeichnis der Sammlung (E:\Aipp\) gliederte sich die Verzeichnisstruktur in maximal drei Unterverzeichnisse. Aufgrund der eindeutigen Ordnerbezeichnungen lag zudem eine klare thematische und chronologische Zuordnung der Bilddateien vor. Zieht man die Systemdateien, wie z. B. DB-, INI- oder LOG-Dateien, ab, blieben außerdem nur wenige bewertungswürdige Dateiformate übrig. Namentlich waren dies JPG, TIF, MOV, WMV, DOC und PDF. Da eine Fotosammlung archiviert werden sollte, wurde außerdem entschieden, sich bei der automatischen Bewertung auf die Bilddateien zu konzentrieren.[88] Die siebzehn übrigen Dateien sollten (vorläufig) kassiert werden.[89] Die vergleichsweise hohe Systematik der Sammlung darf jedoch nicht darüber hinwegtäuschen, dass aufgrund der großen Homogenität letztlich von den zunächst 211.714 Dateien ganze 210.189 archivfähige und potenziell archivwürdige Dateien übrigblieben und der Ausschluss der restlichen Dateien lediglich eine Reduktion des Datenvolumens um knapp 2 GB bedeutete. Das JPG-Format wird aufgrund der eingesetzten Kompression grundsätzlich zwar nicht als Archivformat empfohlen, jedoch wird von einer Migration in der Regel abgesehen, sofern die Daten zum Zeitpunkt der Anbietung bereits in diesem Format vorliegen, da die Formatspezifikationen offen standardisiert sind und das Format außerdem sehr weit verbreitet ist.[90] Im Rahmen der Aufbereitung waren demnach auch keine spezifischen Bestandserhaltungsmaßnahmen in Betracht zu ziehen, da das digitale Archiv des Landesarchivs Baden-Württemberg für den Umgang mit dem Format vorbereitet ist und somit unbeabsichtigte Manipulationen auf der technischen Ebene ausgeschlossen werden können.[91]

Durch die Überprüfung der Pfadlängen wurde sichergestellt, dass keine Pfade mit Überlänge (auf Windows-Systemen Pfade > 259 Zeichen) auftraten. Ein Kopieren der Daten war demnach unproblematisch. Jedoch galt es, angesichts einer geplanten Redaktion der Ordnerbezeichnungen, die Pfadlängen im Auge zu behalten, um im Fall der knapp unterhalb der Maximallänge liegenden Pfade nicht im Nachhinein problematische Längen zu erzeugen, die zum „Übersehen" betroffener Dateien durch eingesetzte Werkzeuge oder im schlechtesten Fall zu unbemerktem Datenverlust durch unvollständige Kopiervorgänge hätte führen können.

Die mit TreeSize durchgeführte Duplikatsuche ist prüfsummenbasiert, findet also hundertprozentige Übereinstimmungen. Oftmals wird die Entfernung von Duplikaten als eines der wichtigen Elemente innerhalb des Bewertungsvorgangs von Dateisammlungen bezeichnet. Dies ist aufgrund der häufig hohen Duplikatraten und der auf alle Dateitypen anwendbaren prüfsummenbasierten Vorgehensweise zunächst nicht weiter verwunderlich, da beides zusammengenommen ein effizientes Vorgehen verspricht, mit dem sich effektiv Speicherplatz sparen lässt. Zudem gehen, so lange lediglich der reine Informationswert

88 Und damit im Wesentlichen auf JPG-Dateien. Die einzige TIF-Datei wurde später durch den Zufallsselektor kassiert.
89 Somit bliebe eine spätere intellektuelle Bewertung noch möglich. Von der Realisierung einer „vorläufigen Kassation" wird bei der Beschreibung des Workflows noch die Rede sein.
90 Vgl. die Empfehlungen der KOST: Koordinierungsstelle für die dauerhafte Archivierung elektronischer Unterlagen, Katalog archivischer Dateiformate: JPEG. Zur Verwendung des Formats siehe ebenso: *Naumann* und *Schmidt*, Chancen und Risiken des Einsatzes verlustbehafteter Bildkompression in der digitalen Archivierung.
91 Zu erhalten sind bei den vorliegenden Bildern in jedem Fall die nach Nestor-Leitfaden bestimmbaren Signifikanten Eigenschaften: Integrität, Metadaten, Form, Größe/Auflösung, Ausrichtung, Farbigkeit und Farbanordnung. Vgl. nestor-Arbeitsgruppe Digitale Bestandserhaltung (Hg.), Leitfaden zur digitalen Bestandserhaltung, S. 37–42.

der Archivobjekte berücksichtigt wird, durch die Duplikatentfernung keinerlei Informationen verloren. Auch bei der Sammlung Aipperspach war der Duplikatanteil mit immerhin rund 22 % signifikant, aber verglichen mit häufig verzeichneten Raten von 50–70 % doch relativ gering. Bei der Durchsicht der Duplikate wurde ersichtlich, dass es sich häufig um Tafeln, Karten- und Zeitungsausschnitte handelte, die Aipperspach zu verschiedenen Bildserien hinzusortiert hatte. Durch gezieltes Entfernen solcher Duplikate wären demzufolge sowohl Verluste bezüglich des Evidenz- als auch des Informationswertes auf der Ebene einzelner Serien in Kauf zu nehmen gewesen, ein händisches Selektieren der übrigen Duplikate wurde indessen als zu aufwendig erachtet.

Aufgrund zweier Prämissen wurde daher auf eine Duplikatentfernung auf Einzelbildebene verzichtet. Einerseits würde die Zufallsselektion zu einer Reduzierung des Duplikatanteils sorgen, andererseits wurde der Umgang mit Duplikaten in erster Linie als Aufgabe der Speicherschicht des Digitalen Archivs betrachtet. Ähnlich wie bereits bei modernen RAID-Systemen sollte es hier ermöglicht werden, die redundante Ablage von Dateien zu vermeiden und das Dateimanagement entsprechend zu organisieren, um den Speicherbedarf zu verringern. Die prüfsummengestützte Duplikatsuche mit TreeSize Pro wurde daher lediglich zur Auffindung identischer Verzeichnisse hinzugezogen, womit die redundante Archivierung gleicher Serien verhindert werden sollte. Solche Dopplungen waren bei der Durchsicht bereits aufgefallen.

Der Umgang mit Quasidubletten, auf den bei der Analyse der Sammlung anfangs ebenfalls einige Zeit verwendet wurde, gestaltete sich aufgrund der Verwendung ähnlicher Bilder zur Kontextualisierung einzelner Serien oder Bildgruppen nicht weniger schwierig. Insbesondere der Einsatz der Software dupeGuru wurde dahingehend erprobt.[92] Das Programm erlaubt die Suche nach ähnlichen Bildern innerhalb eines stufenlos wählbaren Bereichs von 0 bis 100 % Übereinstimmung. Ziel der Überlegungen war es zunächst, innerhalb einzelner Serien ähnliche Bilder zu finden und einige dieser Bilder zu kassieren. Auf die Sammlung angewendet, befanden sich unter den Ergebnissen in erster Linie Tafeln und Kartenausschnitte, die sich nur durch wenige Details unterschieden; vor allem aber bei den Tafeln war offensichtlich, dass diese nicht kassiert werden sollten. Eine intellektuelle Auswahl derjenigen Quasi-Dubletten, die für eine Löschung in Frage kam, war andererseits angesichts der Masse – wie schon bei den Dubletten – nicht effizient ausführbar. Ebenso wenig bedeutete aber das schrittweise Anwenden der Suche auf einzelne Serien eine Effizienzsteigerung zur Durchsicht der einzelnen Serien im Explorer. Somit wurde auch das inhaltsbasierte Kriterium der Ähnlichkeit auf Einzelbildebene letztlich verworfen. Da sich ähnliche Bilder innerhalb einer Serie hintereinander befanden, weil sich die Szenerie und der Standpunkt des Fotografen zwischen den Aufnahmen kaum verändert hatte, würde aber auch hier der Zufallsselektor für eine Reduktion sorgen, da eine bestimmte Anzahl an Bildern übersprungen werden sollte. Der Einsatz von dupeGuru dürfte jedoch immer dann eine wesentliche Erleichterung mit sich bringen, wenn ähnliche Bilder innerhalb einer Sammlung in jedem Fall kassiert werden sollen und sich nicht, wie hier, gerade unter den ähnlichen Bildern auch zahlreiche Bilder befinden, die aufgrund ihres Informationsgehalts archiviert werden sollen.

92 Das für alle gängigen Betriebssysteme verfügbare kostenlose Tool dient der inhaltsbasierten Ähnlichkeitssuche (Informatiker sprechen von „unscharfer Suche“ oder „fuzzy-Suche). Siehe: dupeGuru – finds duplicate files.

Mit der Beobachtung, dass innerhalb einzelner Serien bestimmte Informationen nicht verloren gehen sollten, obschon sie in anderen Serien ebenfalls vorhanden waren, wurde auch die Frage der Fokus-Setzung für die SIP- und in der Folge auch AIP-Formierung berührt. Die Analyse mit TreeSize Pro ergab, dass 81 der 1664 Verzeichnisse ihrerseits Unterverzeichnisse – also weitere Serien – enthielten. Angesichts der in Relation zur Verzeichnisanzahl geringen Anzahl von Unterverzeichnissen erschien es jedoch legitim, die SIPs auf Ereignisebene zu formieren, und die Unterverzeichnisse aufzulösen. Zu klären blieb, in welcher Weise dies realisiert werden sollte.

Insgesamt ließ sich aus dieser Analyse ableiten, dass sich mit den zunächst naheliegenden Mitteln (Duplikatsuche über Prüfsummen, Ähnlichkeitssuche) keine effiziente Bewertung ohne bedeutenden Verlust an Kontextinformation erzielen lies. Die auf den ersten Blick so klare Strukturierung der Sammlung wies also doch Fallstricke auf, denen zu begegnet werden musste. Ein Verzicht auf die Duplikatentfernung wurde schließlich auch dadurch bekräftigt, dass sich abzeichnete, dass die Zufallsselektion ein wesentlicher Bestandteil des Workflows werden müsse, um das Ziel einer Reduktion um rund 90 % mit der gewünschten Effizienz zu erreichen. Auf Einzelbildebene würde sich – quasi als Nebenprodukt – auch die Zahl der Duplikate verringern. Andererseits war auch für das randomisierte Sampling ein unerwünschter Informationsverlust einzukalkulieren, weshalb dieses Problem zunächst ungelöst blieb. Unter Berücksichtigung dieser Ergebnisse bedurfte es der weiteren Analyse auszuführender Bearbeitungsschritte, wobei eine Identifikation der automatisierbaren Prozesse und eine daraus resultierende Abgrenzung zu überwiegend händisch auszuführenden Vorarbeiten sinnvoll erschien. Daraus sollte sich schließlich auch eine grundsätzliche Entscheidung darüber ableiten lassen, mit welchen Werkzeugen eine Automatisierung umsetzbar erschien und ob sich das Vorgehen idealerweise vereinheitlichen ließe oder aber ein Workflow aus der schrittweisen Verwendung unterschiedlicher Tools zu stricken wäre.

4.4.1 Automatisierbare Prozesse

Zu den automatisiert auszuführenden Prozessen wurde von Beginn an das Zufallssampling gezählt. Ebenso erschien – obwohl sich die Dubletten- und viel mehr noch die Quasi-Dublettenerkennung als nicht zielführend erwiesen hatten – weiterhin der Einsatz von Bilderkennungs-Verfahren vielversprechend. Denn Folgendes sollte nicht völlig unberücksichtigt bleiben: „Neue Chancen der bild- und tonbasierten Bild- und Tonsuche ergeben sich just in dem Moment, wenn eine Photographie im Archiv gar keine Photographie mehr ist, sondern ein Datenformat in Architekturen binär kodierter Datenverarbeitung.“[93]

Zwischenzeitlich wurden die anfangs in der SQL-Datenbank abgelegten Exif-Metadaten dahingehend untersucht, ob sich einige dieser Daten möglicherweise zur Identifikation bestimmter Fotos nutzen ließen. Jedoch erwiesen sich keine der Daten als derart charakteristisch. Da fehlende Schärfe als Kassationskriterium eingestuft wurde, kam darüber hinaus die Ermittlung mit hoher Wahrscheinlichkeit unscharfer Bilder über die Faustregel „Verschlusszeit <= 1 / Brennweite“ in Betracht.[94] Dies wäre auch mit den vorhandenen Daten über einfache SQL-

93 *Ernst*, Wohldefinierte (Bild-)Archive – und was sie nicht sind, S. 11.
94 Der Faustformel zufolge ist bei längeren Verschlusszeiten mit Unschärfe durch unruhige Kamerahaltung (Verwackeln) zu rechnen. Faktoren wie die Verwendung eines Blitzes, Stativs, Bildstabilisators oder eine besonders ruhige Hand des Fotografen werden dabei natürlich nicht berücksichtigt. Vgl. z. B. das auf dieser Formel basierende Lightroom-Plugin: *Branca*, Lightroom plugin.

Abfragen zu realisieren gewesen. Dass die verfügbaren Daten darüber hinaus jedoch keine Grundlage für weitere Bildanalysen boten, wurde als weiteres Argument für den Rückgriff auf echte bildbasierte Suchmechanismen aufgefasst, weshalb sich auf die Suche nach einer auf diesem Gebiet mächtigeren Lösung gemacht wurde. Auch insgesamt erschien die Datenbank-Lösung aufgrund der strikten Trennung der Datenhaltung innerhalb der Datenbank und der Dateiverwaltung der Bilddateien über zusätzliche Werkzeuge oder den Windows-Explorer als Grundpfeiler des zu entwickelnden Workflows letztlich zu umständlich, obwohl die Datenhaltung in Tabellen aufgrund der flexiblen Auswertbarkeit und der Weiterverarbeitung durchaus sinnvoll erschien.[95]

Zudem erwiesen sich nicht nur dupeGuru sondern auch weitere Tools und Programme wie beispielsweise Adobe Photoshop Elements – der integrierte Organizer unterstützt ebenfalls die inhaltsbasierte Suche ähnlicher Bilder – als wenig hilfreich. Zwei Faktoren, die zum Teil auf die Größe der Sammlung Aipperspach zurückgehen, sprachen letztlich immer gegen deren Verwendung: 1. Die unübersichtliche Darstellung der umfangreichen Suchergebnisse, was eine notwendige intellektuelle Bewertung des Ergebnisses und die Selektion archivwürdiger / kassabler Bilder in der Programmoberfläche erschwerte. 2. Die mangelnde Exportfunktion von Metadaten über Suchergebnisse, die – sei es als Log oder zur Weiterverarbeitung in der Datenbank – nützlich gewesen wären. Aus diesen beiden Punkten resultierte 3. Ein insgesamt kaum effizientes und damit im Ergebnis unbefriedigendes Handling.

Abgesehen von der Zufallsselektion und der bildbasierten Bewertung sollten auch die Aufbereitungsprozesse – Umstrukturierung der Sammlung, Redaktion der Ordnernamen, Ablage der Erschließungsmetadaten – weitgehend automatisiert werden. Während im Bereich Bilderkennung theoretisch noch der Rückgriff auf service-basierte Anwendungen denkbar gewesen wäre, war für die Aufbereitungsprozesse von Beginn an eine skriptbasierte Realisierung über CMD-Skripte angedacht. Da nun aber die Datenbanklösung mangels auswertbarer Daten, die durch Tools hätten geliefert werden sollen, zunehmend in den Hintergrund rückte, zeichnete sich verstärkt ein dritter Weg ab, der sowohl die bildbasierten Automatisierungsprozesse als auch die notwendigen Operationen innerhalb des Dateisystems in sich vereinen würde: Eine skriptbasierte Lösung, die nicht auf CMD-Skripten sondern auf der Programmiersprache Python aufbaute, womit auf einen sehr viel größeren Werkzeugkasten zurückgegriffen werden konnte.

4.4.2 Notwendige Vorarbeiten

Mit der Identifikation der automatisierbaren Arbeitsschritte und der Festlegung auf Python als zentrales Werkzeug ließ sich das weitere Vorgehen in zwei große Bereiche aufteilen, die von hier an parallel bearbeitet wurden. Zum einen wurde mit der Entwicklung der Skripte begonnen, wobei sich zunächst auf die allgemeinen Aufbereitungsprozesse konzentriert wurde. Anschließend konnten Lösungen für die bildbasierten Arbeitsschritte entwickelt werden, die eine tiefere Einarbeitung in die Programmierung erforderten. Durch

95 Daher sei hier betont, dass sich durch gezielte Datenbank-Abfragen grundsätzlich schnell Informationen generieren lassen, die bei der Aufbereitung von Dateisammlungen nützlich sein können. Der Rückgriff auf SQL oder Access sollte in diesem Rahmen weiterhin in Erwägung gezogen werden. Beispiele dafür, wie weit eine elaborierte Anwendung führen kann, finden sich auch bei: *Bayer*, Bytebarn; *Enge* und *Kramski*, „Arme Nachlassverwalter…“.

stetige Tests mithilfe der angelegten Testsuite wurden die einzelnen geschriebenen Skripte schrittweise so weit optimiert, dass sich ein geschlossener Workflow beschreiben lies, der nach einem einmaligen großen Testdurchlauf und sich anschließenden Optimierungsarbeiten erfolgreich durchgeführt wurde. Diese Arbeiten beanspruchten während des gesamten Vorgehens die meiste Zeit.

Außerdem waren folgende Punkte weiterhin zu durchdenken. Die Identifikation völlig übereinstimmender Ordner fügte sich in die ebenfalls noch ausstehende Suche nach Verzeichnissen, die nicht überliefert werden sollten, weil sie Bilder enthielten, für die keine Rechte vorlagen oder diese nicht zum Sammlungsprofil des Archivs passten. Ebenso waren Ordner zu ermitteln, die aufgrund ihres archivischen Werts vollständig übernommen werden sollten. Weiterhin war ebenfalls noch zu entscheiden, auf welcher Ebene die SIPs formiert werden sollten. Da diese Schritte zum Teil auf die intellektuelle archivarische Bewertung angewiesen waren, fand diesbezüglich bis zur Ausführung des finalen Workflows eine stetige Verständigung mit Sybille Brühl und Franz-Josef Ziwes sowie Kai Naumann (alle Landesarchiv Baden-Württemberg) statt. Hinsichtlich der SIP-Formierung wurde schließlich entschieden, dass sämtliche Verzeichnisse mit sieben oder mehr Unterverzeichnissen vollständig aufgelöst und zu eigenen SIPs formiert werden sollten, wohingegen Ordner mit weniger Unterverzeichnissen zu einem einzigen SIP zusammenzufassen wären. Die Zahl der Unterverzeichnisse wurden in der Detailansicht von TreeSize Pro ermittelt (vgl. Abb. 2). Da es sich um nur sechzehn Verzeichnisse mit mehr als sieben Unterverzeichnissen handelte, wurde entschieden, das Umstellen händisch vorzunehmen und nicht weiter zu automatisieren, das Vorgehen aber an geeigneter Stelle in den Workflow zu integrieren. Die restlichen Umstrukturierungen sollten automatisiert stattfinden.

Diagramm | Details | Dateitypen | Benutzer | Dateialter | Top 100 Dateien | Historie

Absoluter Pfad	Verzeichnisse	Dateien
E:\Aipp\2500. GARTENSCHAU 2013 in SIGMARINGEN vom 11.5. - 15.9.2013\	95	16.098
E:\Aipp\844d. Auf den Spuren Christi im Hl.Land Israel v.12.-21.3.2008\	38	1.264
E:\Aipp\1979 Dorfansichten und Bilder von Unterschmeien und Oberschmeien\	20	1.029
E:\Aipp\2340 Dorfansichten und Bilder von Unterschmeien und Oberschmeien\	20	1.029
E:\Aipp\1337 Die Stadthalle Sigmaringen vor dem Um-u.Erweiterungsbau, Bilder vom 14.-19.12.2009 Gleich wie Nr.1399b Ext.Festpl.Samsung 2010-2011\	17	670
E:\Aipp\1133a Bilder von BERLIN, SCHWERIN und SCHLESWIG-HOLSTEIN 29.4.-2.5.2009 -Gesamtfassung, unterteilt in Ordner-\	16	1.466
E:\Aipp\2069a Bilder von der Stadthalle Sigmaringen vor dem Umbau 2009\	16	681
E:\Aipp\2130. FIDELISFESTE IN SIGMARINGEN seit 2006\	16	3.527
E:\Aipp\2657. Bildervorführung beim Seniorennachmittag am 6.1.2014 durch Gunter Aipperspach\	15	594
E:\Aipp\3051. FASNET 2015 IN SIGMARINGEN -15 Ordner-\	15	2.022
E:\Aipp\3179. Bilder vom BLÜTENZAUBER SIGMARINGEN 12.6.-9.8.2015\	15	2.065
E:\Aipp\2120 Bilder vom Hauptbahnhof Hbf Stuttgart vor der Umwandlung vom Kopf- zum Durchgangsbahnhof Stuttgart 21 S 21\	12	269
E:\Aipp\3050. FASNET MIT BRÄUTELN IN UNTERSCHMEIEN IM FEBRUAR 2015 -12 Ordner-\	12	1.932
E:\Aipp\2698 FASNET IN SIGMARINGEN 2014 Bilder vom 23.2. - 4.3.2014\	11	2.271
E:\Aipp\784 Winter-und Weihnachtsbilder von Sigmaringen 2008-2010\	10	177
E:\Aipp\1044b.Winter-u.Weihnachtsbilder in Sigmaringen 2008-2010\	10	178
E:\Aipp\1336 Winter-und Weihnachtsbilder von Sigmaringen 2008-2010\	10	177
E:\Aipp\2389 50 Jahre Firma Kaut Holzverarbeitung Sigm.-Unterschmeien-Jubiläumsveranstaltung 9.3.2013\	10	472
E:\Aipp\3110. -SIGMARINGEN IM WANDEL-Häuser Leopoldplatz1,Karlstr.8, 10 u.12 werden abgerissen u. -L1-Leopodplatz 1 wird aufgebaut 2011-2014\	10	1.609
E:\Aipp\3347a. Winter-u.Weihnachtsbilder in Sigmaringen 2008-2010\	10	178
E:\Aipp\2191 Reise nach POLEN-Danzig und Masuren zum Kennenlernen- vom 12.-20.7.2012 -etwas gekürzt-\	9	1.520
E:\Aipp\2520 WALLFAHRT-PILGERREISE nach ITALIEN vom 30.8. - 6.9.2013\	9	1.852
E:\Aipp\1610 50 Jahre SV U-O-Schmeien Bilderzusammenstellung 2001-2010\	8	623
E:\Aipp\2130a. Bilder von den STEHEMPFÄNGEN AUF SCHLOSS SIGMARINGEN anl. der FIDELISFESTE in Sigmaringen 2006 - 2012\	8	526
E:\Aipp\1642a 50-jähr.Jubiläum SV Unter-Oberschmeien, alle Bilder vom Zeitabschnitt 2001-2009 der Fotoausstellung am 10.10.2010\	7	624
E:\Aipp\3180. Abbau Sigmaringer Blütenzauber 2015 ab 10. bis 14.8.2015\	5	251

Abb. 2: Ursprüngliche Verteilung von mehr als sieben Unterverzeichnissen (Ausschnitt aus TreeSize Professional).

4.5 Wieso Python?

Bei der Entwicklung von MPLP-Ansätzen zur Aufbereitung und Bewertung von Dateisammlungen steht nicht zuletzt der Wunsch nach einer möglichst einfachen, ohne größere Einstiegshürden umsetzbaren Lösung neben dem Wunsch nach der größtmöglichen Automatisierung.[96] Diesem verständlichen Wunsch wohnt jedoch ein Widerspruch inne, der mit anwachsender Komplexität der zu bewältigenden Problematik immer schwerer wiegen wird. Einfach einzusetzende Werkzeuge bieten immer auch einen nur beschränkten Funktionsumfang, was bei zunehmender Komplexität der Problemstellung mutmaßlich zur Notwendigkeit der immer kleingliedrigeren Aneinanderreihung von Spezial-Tools führen wird, solange es nicht – sofern überhaupt möglich – eine professionelle Fachanwendung entwickelt wurde, die keine Anforderungen und Wünsche mehr unberücksichtigt lässt. Durch die Verkettung von Tools steigt aber wiederum die Anwendungs-Komplexität. Die Erprobung von erst einmal geeignet erscheinenden Tools wie dupeGuru hat im Rahmen der vorliegenden Arbeit zudem gezeigt, wo angesichts der Dateianzahl die Anwendung solcher Tools auch bei Dateisammlungen mit vergleichsweise geringer Komplexität an ihre Grenzen stößt – wenn nämlich die Informationsfülle auch durch den Werkzeugeinsatz nicht auf eine für den Menschen überschaubare Menge reduziert wird.

Im Zuge des favorisierten Datenmanagements über SQL, sollten die zur Aufbereitung nutzbaren Informationen mit einem allgemein anwendbaren Werkzeug – ein solches ist die Datenbank – organisiert werden. Wenn auch die Anwendung einer Datenbanksprache wie SQL zweifellos eine höhere Hürde darstellt als die Verwendung von Werkzeugen mit grafischen Oberflächen, sollte dieser Ansatz jedoch keineswegs als Widerspruch zur MPLP-Philosophie aufgefasst werden. Viel mehr bietet die Anwendung von SQL grundsätzlich einen flexiblen Lösungsansatz, der unabhängig von sammlungsspezifischen Eigenschaften anwendbar ist. Erscheint der Erwerb von Grundkenntnissen in einer Datenbanksprache zur Bearbeitung einer einzigen Dateisammlung vielleicht noch recht hoch, so wird das Überwinden dieser Hürde zwar nicht leichter aber bedeutend reizvoller, wenn man den sich mit jedem weiteren Anwendungsfall potenzierenden Nutzen bedenkt.

Da nun die Datenbanklösung im vorliegenden Fall nicht wie gewünscht weiterführte, erschien die Hinwendung zu Python als logische Folge, denn die soeben für SQL beschriebenen Zusammenhänge gelten auch hier. Zwar ist die Einstiegshürde zunächst zugegebenermaßen noch mal etwas höher als im Fall von SQL, doch eröffnet sich durch den Einsatz von Python auch eine ungleich höhere Vielzahl an Anwendungsmöglichkeiten. Selbst dann, wenn die zur Verfügung stehenden Werkzeuge sich als untauglich erweisen, bleibt mit Python sinngemäß der Griff zum digitalen Schweizer Taschenmesser. Python ist eine dynamische Programmiersprache, die sich zum Schreiben von kleinen Programmen besonders eignet, da diese mit jedem üblichen Texteditor geschrieben und bearbeitet werden und als Skripte schnell ausgeführt werden können. Python selbst und damit auch die in Python geschriebenen Skripte lassen sich unter Windows-, Linux- und macOS-Umgebungen ausführen. Die Syntax der Sprache wurde speziell mit dem Ziel der Einfachheit

96 So auch: *Belovari*, Rasche und einfache Bearbeitung von Dateisammlungen.

entwickelt und gilt weithin als leicht erlernbar. Durch die modulare Erweiterbarkeit ist Python jedoch gleichzeitig sehr mächtig. Es findet aufgrund des einfachen Einstiegs vielfach Anwendung in wissenschaftlichen Bereichen, eignet sich damit aber auch hervorragend für die Anwendung im Archiv.[97]

Aufgegeben wird mit der Hinwendung zu Python zweifellos ein ohne Einarbeitungszeit gleich übernehmbares, weil intuitives Vorgehen, wie es die meisten Archivare und Archivarinnen vom Umgang mit kleineren Tools und Programmen gewohnt sein dürften. Dieser Nachteil wird aber durch eine Reihe von Vorteilen aufgewogen. Über die konkrete Aufbereitung der Sammlung Aipperspach hinaus, zeigt der beschriebene Workflow exemplarisch, wie mit vergleichsweise wenig Aufwand Aufgaben abgearbeitet werden können, die ohne Skripteinsatz ein Vielfaches an Arbeitszeit in Anspruch nehmen würden – ganz im Sinne von: More Product, Less (human) Processing. Als Proof of Concept sollte vielleicht auch erwogen werden, dass sich die zur Bearbeitung der Sammlung Aipperspach notwendigen Kenntnisse weitgehend erst im Rahmen der Bearbeitung selbst angeeignet wurden.

Der zentrale Vorzug des skriptgesteuerten Vorgehens liegt darin, dass die automatisierten Aufbereitungs- und Bewertungsschritte soweit vereinheitlicht wurden, dass sie alle der gleichen strengen Logik folgen, die sich aus der Steuerung über CSV-Dateien ergibt – vom Verschieben bestimmter Dateien über die Zufallsauswahl bis hin zur bildbasierten Suche. Dies schließt auch das automatische Anlegen von CSV-Logfiles ein, durch die jeder einzelne Bearbeitungsschritt dokumentiert wird. Bei allen Veränderungen wird ein Logfile in der Form „Alter Zustand;Neuer Zustand“ abgelegt. Die Formatierung der CSV-Dateien folgt einem einheitlichen, weil selbst definierten, Schema. Somit besteht, anders als bei der Verwendung von Tools und Programmen, die vollständige Kontrolle über die Erzeugung von Metadaten. Das Datenmanagement über Tabellen gleicht letztlich dem der Datenbank-Lösung. Daher ist es auch jederzeit möglich, diese Daten in eine Datenbank zu importieren – beispielsweise um bestimmte Daten zusammenzuführen oder den Datensatz nach bestimmten Informationen zu durchsuchen.[98] Auch bei einem auf diese Art kombinierten Vorgehen besteht weiterhin nicht nur die weitgehende Unabhängigkeit von anderen Werkzeugen sondern auch von Betriebssystemen.

97 Einen guten Einstieg mit Informationen zur Installation und Anwendungsbeispielen für Python im Archiv gibt: *Wiedeman*, Python for Archivists. Ein dem in der vorliegenden Arbeit in Teilen ähnlicher Ansatz, der z. B. auch ein Zufallssampling einschließt, findet sich bei: *England* und *Hanson*, Automating Digital Archival Processing at Johns Hopkins University; vgl. weiterhin folgenden kurzen Erfahrungsbericht: *Berish*, Learning Python as a Processing Archivist.

98 Dies wurde im Rahmen des Workflows nebenbei erprobt, dazu Beginn der Workflow-Entwicklung eine solche Datenbank angelegt wurde. Das Vorgehen fand letztlich jedoch keinen Eingang in den produktiven Workflow und soll daher in der vorliegenden Arbeit nicht weiter thematisiert werden.

5. Workflow-Beschreibung

5.1 Allgemeine Erläuterungen zur Skript- und Workflowsteuerung

Der im Folgenden geschilderte Workflow beschreibt die automatisierte Aufbereitung und Bewertung der Sammlung Aipperspach. Die hier erläuterten Prozesse finden auch insofern automatisiert statt, als dass während des Durchlaufens der gesamten Schrittabfolge keine weiteren intellektuellen Bewertungsentscheidungen getroffen werden müssen.[99] Der Workflow ist in sich geschlossen und bildet sozusagen die mittlere Schicht eines Bewertungs-Sandwiches. Diejenigen Bewertungsentscheidungen, die vorab intellektuell zu treffen waren, weil diese in die Automatisierung eingehen mussten, wurden bereits geschildert. Er setzt mit der geloggten Löschung der vorab zur Kassation freigegebenen Verzeichnisse ein. Bis hierin sind keinerlei verändernde Eingriffe in die Dateisammlung erfolgt. Gleichwohl ermöglicht der Workflow eine intellektuelle Nachbewertung und damit auch Korrekturen des Bewertungsergebnisses, da keine Daten endgültig gelöscht werden – mit Ausnahme der zuvor bereits intellektuell als nicht archivwürdig bewerteten Verzeichnisse im ersten Arbeitsschritt des Workflows.

In sich geschlossen erscheint der Workflow insbesondere auch, weil alle vorgenommenen Veränderungen nun durch CSV-Dateien geloggt werden, d.h. es findet kein Löschen, Verschieben oder Umbenennen von Verzeichnissen oder Dateien statt, das hinterher nicht anhand eines solchen Logfiles nachvollziehbar wäre. Dies wird als Grundprinzip des Workflows angesehen. Geschlossenheit bedeutet hingegen nicht, dass der Workflow an einem Stück ausgeführt werden muss. Bis auf den abschließenden Ingest und die bildbasierten Prozesse, die insgesamt einige Stunden Rechenzeit beanspruchen, lässt sich der Workflow jedoch in wenigen Stunden durchlaufen.

Während des Workflows kommen insgesamt sechzehn verschiedene Skripte zum Einsatz, die allesamt dem Anhang zu entnehmen sind.[100] Aus der Skript-Anordnung allein ergibt sich noch keine logische Abfolge. Ebenso wenig lassen sich alle Skripte exakt einem Bearbeitungsschritt zuordnen, da einige Skripte – beispielsweise diejenigen zur Erstellung von Verzeichnis- und Dateilisten – mehrfach zum Einsatz kommen. Alle Skripte wurden zwar für den konkreten Anwendungsfall geschrieben, gleichzeitig ist aber keines der Skripte derart spezifisch, dass es nur auf die Sammlung Aipperspach angewendet werden könnte, sodass Teile des Workflows bei Bedarf auf ähnliche Anwendungsfälle adaptiert werden können. Zwar wurden die Skripte nur unter Windows konzeptioniert und getestet, grundsätzlich sind sie jedoch auch in Linux- und

99 Mit Ausnahme der intellektuellen Nachbewertung kassabler Fotografien, die durch die automatisierte Suche nach unscharfen Fotografien gefunden werden.

100 Das Skript „bilder_imagehash.py“ fand keinen Eingang in den endgültigen Workflow. Es findet sich dennoch im Anhang, da es dem interessierten Leser als Beispiel für die kombinierte Anwendung von Python und einer SQL-Datenbank dienen kann. Die vom Skript zu jeder Bilddatei ausgegebenen Hash-Werte sollte in der Datenbank verglichen werden und über „hamming distances“ ähnliche Bilder gefunden werden. Für den Einsatz der Tafel-Suche erwiesen sich die Image Hashes jedoch als nicht genau genug.

Macumgebungen ohne größere Modifikationen lauffähig.[101] Zum Aufruf wird Python 3.x benötigt. Neben der Standardbibliothek erfordern einige Skripte die Installation folgender Bibliotheken: natsort, pytesseract, pillow, numpy, opencv.[102]

Die Skripte wurden mit dem Ziel einer möglichst flüssigen Integrationsfähigkeit in die einzelnen Teilworkflows geschrieben. Die Konfigurationen finden zum Teil im Skripttext, zum Teil durch Nutzereingaben statt. Es hätte grundsätzlich auch auf Nutzer-Eingaben verzichtet werden können, jedoch ermöglichen diese im Einzelnen eine vereinfachte Bedienung gegenüber der Vorabkonfiguration im Skripttext. So können Input-Dateien (CSV-Dateien) einfach per Drag-and-Drop im Ausführungsfenster abgelegt werden und die händische Eingabe langer Pfade entfällt vollständig. Auf die Übergabe von Kommandozeilenparametern bei Skriptaufruf wurde verzichtet, grundsätzlich werden variable Parameter per Eingabedialog oder im Skript-Text angegeben.[103] Die Kommentierung des Codes beschränkt sich im Anhang weitgehend auf die Beschreibung der Aufgabe einzelner Funktionen und Prozeduren sowie einzelner Schlüsselstellen und eine kurze einleitende Beschreibung.

Die Skripte können in drei Kategorien eingeteilt werden – und zwar je nach Art der „Objekte“ auf die sie angewendet werden, was auch bei der Namensgebung berücksichtigt wurde: 1. Ordner – 2. Dateien – 3. Bilddateien.[104] Mit Ausnahme eines Skripts laufen alle Skripte unabhängig von anderen Tools oder Programmen ab. Das Skript „bilder_oeffnen.py“ übergibt jedoch eine Liste mit Bildpfaden zum Öffnen an IrfanView, dessen Installation dementsprechend vorausgesetzt wird.

Die Ausführung des Workflows gestaltet sich am einfachsten, wenn alle Skripte in einem Verzeichnis liegen (vgl. Abb. 3). Das erste Aufrufen eines Skripts legt den Ordner „Logs“ an, in dem nun nacheinander alle Skripte ihre Ausgabe in Form von CSV-Dateien ablegen. Sämtliche Logs enden mit einem Zeitstempel in der Form „DD-MM-YYYY_HH-MM-SS“ (vgl. Abb. 4). Einige Skripte verlangen einem Input-File – mit Ausnahme des Skripts „bilder_oeffnen.py“ (TXT-Datei) ist dies eine CSV-Datei. Sämtliche Input-Files des Workflows sollten hier abgelegt werden (vgl. Abb. 5). Nach Ausführen der Datei „start“ wird eine Eingabeaufforderung gestartet, über die per Drag-and-Drop das gewünschte Skript aufgerufen werden kann (vgl. Abb. 6). Die korrekten Einstellungen sollten vor jedem Start in einem Texteditor überprüft werden.

Neben Python und IrfanView wurden der Texteditor Notepad ++ sowie Microsoft Word und Excel sowie testweise deren LibreOffice-Äquivalente verwendet. Der Einsatz dieser Programme wird an den entsprechenden Stellen noch näher zu erläutern sein. Unterstützend kommen Total Commander und TreeSize Professional zum Einsatz.

101 Eine Einschränkung entsteht bei einigen Skripten durch die Überprüfung der korrekten Pfadeingabe. Hier wird Windows-typisch an dritter Stelle ein Pfad-Trenner erwartet (F:\pfad), wohingegen auf Unix-System der Separator an erster Stelle stünde (\home\pfad). Vgl. z. B. S. 94, Python-Skripte I, Z. 39. Die unterschiedliche Pfadzusammensetzung ist bei systemübergreifender Verwendung also zu berücksichtigen.

102 Die Pakete lassen sich in der Regel über den Python-eigenen Paketmanager „pip“ herunterladen und installieren. Auf Windows-Systemen empfiehlt sich der Download der drei letztgenannten Pakete (als „unoffical releases“) von: Python Extension Packages for Windows by Christoph Gohlke.

103 Vgl. S. 109, Python-Skripte XII, Z. 32 und 34.

104 Die Dateitypen lassen sich im Code definieren. Vgl. beispielsweise S. 108, Python-Skripte XI, Z. 66.

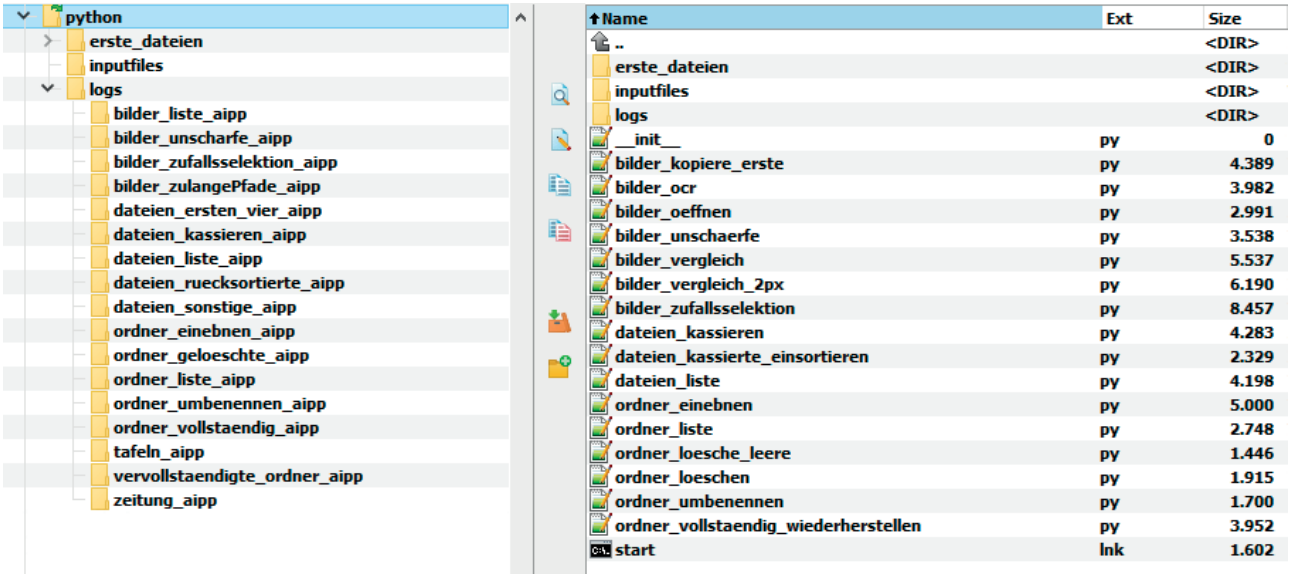

Abb. 3: Skript-Verzeichnis mit allen Skripten, inkl. Logverzeichnis und Input-Files (Ausschnitt aus Total Commander).

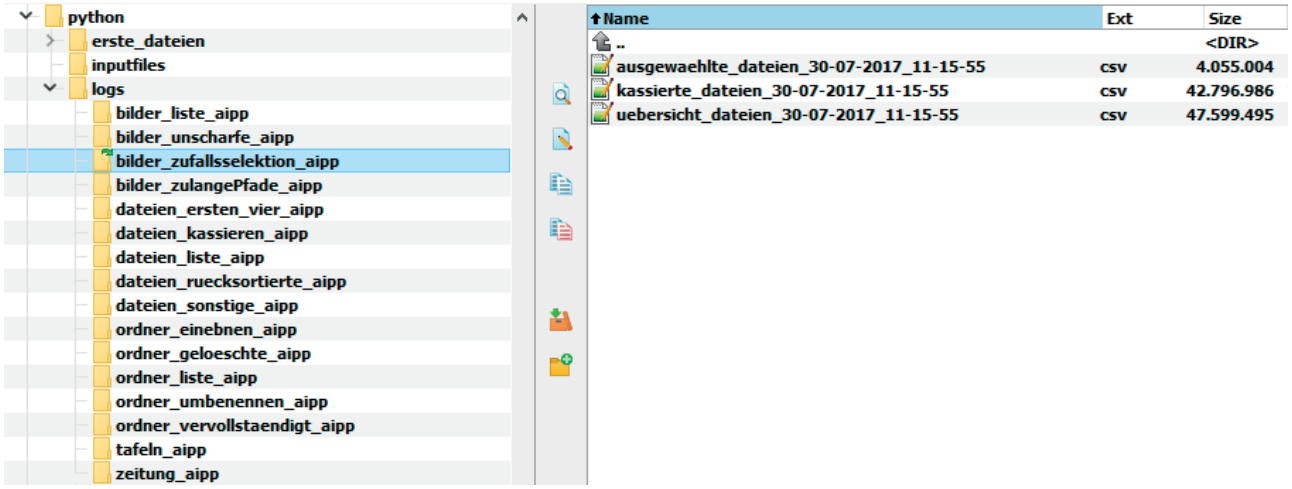

Abb. 4: Anlage von Log-Dateien durch das Skript „ordner_einebnen.py“ (Ausschnitt aus Total Commander).

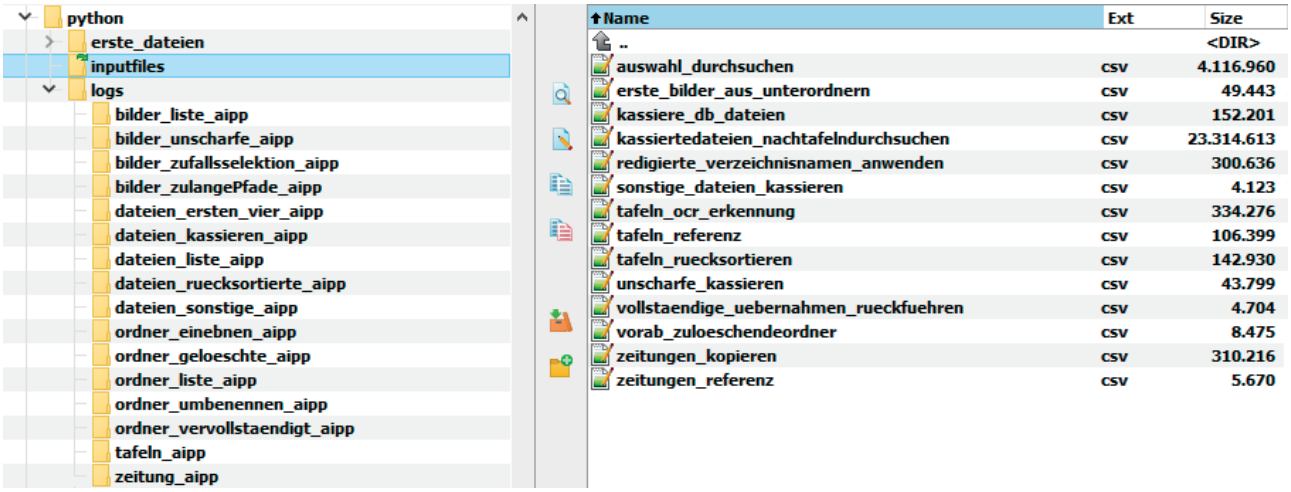

Abb. 5: Input-Files zur Steuerung einzelner Skripte (Ausschnitt aus Total Commander).

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. Alle Rechte vorbehalten.

C:\skripte>python dateien_liste.py

Pfad des Stammverzeichnisses? (z.B. F:\Aipp): F:\Aipp

Welche Dateitypen sollen erfasst werden? "B" für Bilder / "S" für Sonstige: b

MD5-Prüfsummen erstellen? "J" für Ja / "N" für Nein: j

ede8c1f5f4605ac4e999190bc33f75b0        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(101).JPG
ebe1236ff899150eb0d3ba1a26126705        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(102).JPG
7afcf09fd7345733c469ee8011e52e78        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(103).JPG
2aaf33f32c30bc347039b74f1898d9d5        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(104).JPG
1175f6504c6fae829d34f335b5e037a1        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(105).JPG
26ad2c0f70a5d293845564116ca861a4        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(106).jpg
b2d4601863ff2ee8702b80a6d2b6fcfb        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(107).jpg
a3cc9d0f96f582593110c394230e684c        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(108).JPG
396b473578b9dd8e40b10ba180bb0439        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(109).JPG
1c61d45517b224aa8246648e9e48c057        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(110).JPG
c1ed8d89288dc72012b94a54224e6764        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(111).JPG
c7293da3bd0ed6b72305c2600c984f86        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(112).JPG
018506be45c753bc87902ee52e4ec05b        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(113).JPG
7365d2e58a56f97f07d403707df8fcbb        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(114).JPG
782952bb139c591c3436bb0ec360fcff        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(115).JPG
14589c1877daf52f687ba27b7de6d372        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(116).JPG
e306a067040c2b0a23660315db148074        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(117).JPG
fa9ecb25fdd4c669e068635ac45c24c7        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(118).JPG
503dc72bcefb576f4f705824678078e1        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(119).JPG
5aedb4e1013789014f241b16422c0394        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(120).JPG
02b1a6ec3eedbdd63c98a8aff0302f5d        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(121).JPG
1e9a234ee4c422207252a61ddb507937        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(122).JPG
6ca878c798cd69b0ea23277d9f9b6161        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(123).JPG
1511e87c5493e9cdf7a590dd8fb96254        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(124).JPG
d49e7deb89d1dd9c171958d93610951c        F:\Aipp\1002 Volkstrauertag in Oberschmeien 16.11.2008\25. Volkstrauertag in OSchmeien 16.11.98\(125).JPG
```

Abb. 6: Bildschirm-Ausgabe des Skripts „dateien_liste.py“ (Ausschnitt aus Windows Eingabeaufforderung).

5.2. Löschen nicht archivwürdiger Verzeichnisse

Aufgabenbereich:	Bewertung
Eingesetzte Skripte:	ordner_liste.py, ordner_loeschen.py
Inputfiles:	vorabzuloeschende_ordner.csv
Tools:	TreeSize Pro, Notepad++

Zunächst sollten alle Verzeichnisse gelöscht werden, die intellektuell – jedoch durch die Duplikatsuche in TreeSize Pro und einige Handgriffe in Notepad++ unterstützt – ausgewählt wurden. Ziel war die Entfernung von Bildern, die entweder nicht von Herrn Aipperspach oder außerhalb Baden-Württembergs aufgenommen wurden sowie die Entfernung von Redundanzen auf Verzeichnisebene. Es wurden hier keine Einzelbilder in den Blick genommen.

Durch die konsequente Vergabe von Verzeichnisnamen (vgl. hierzu noch einmal S. 33, Abb. 2) durch Herrn Aipperspach, war es möglich, die Auswahl auf Basis dieser Bezeichnungen vorzunehmen. Da jedoch nicht jedes der 1664 Verzeichnisse einzeln überprüft werden sollte, wurde mit dem Skript „ordner_liste.py“ eine Verzeichnisliste erstellt – das Skript gibt eine CSV-Datei mit der Spalte „Verzeichnisse“ aus, in der alle Verzeichnispfade aufgelistet werden. Die Verzeichnisliste war in Notepad++ durch „Suchen und Ersetzen“ und Regulärer Ausdrücke leicht in eine Wortliste zu zerlegen. Mithilfe des Notepad-Plugins TextFX konnte diese Liste von 19.668 Wörtern auf 3.336 reduziert werden, indem eine alphabetische Sortierung mit dem Zusatz „Sort outputs only UNIQUE lines“ erstellt wurde. Das Ergebnis war also eine alphabetische Liste aller von Aipperspach in den Ordnerbezeichnungen verwandten Wörter. Diese Liste konnte nun bequem absteigend nach Schlüsselbegriffen durchsucht werden, wobei hier wiederum lediglich die Substantive zu berücksichtigen waren. Neben Ortsbezeichnungen wie „London“ fielen auch Wörter wie „Kurzfassung“ und „Bildervorführung“ ins Auge, womit sich so auch bereits Hinweise auf Redundanzen finden ließen. Gleichzeitig ließ sich so auf die Existenz von Bildern vom „Südkurier“ schließen.

Die gefundenen Schlüsselwörter wurden anschließend in Notepad++ als Suchbegriffe auf die ursprüngliche Verzeichnisliste angewendet. Der Editor ermöglicht das Anzeigen sämtlicher Treffer als eigenständige Liste („Find All In Current Document“), sodass sich unkompliziert alle entsprechenden Verzeichnispfade kopieren und in ein neues Dokument einfügen ließen. Nach Abarbeiten aller Schlüsselbegriffe entstand so eine Verzeichnisliste mit vollständig zu kassierenden Ordnern.

Zum Teil waren bei der intellektuellen Durchsicht einzelner Verzeichnisse bereits zu löschende Verzeichnisse aufgefallen. Auch diese Verzeichnisse wurden in die erstellte Liste aufgenommen. Ebenso wurde das Ergebnis der Duplikatsuche in TreeSize Pro nach Verzeichnissen abgesucht, in denen sich auffällig viele Dateien doppelten. So konnte die Liste nochmals ergänzt werden. Die endgültige Aufzählung umfasste 79 Pfade.

Die Liste wurde als „vorabzuloeschende_ordner.csv“ im Input-Verzeichnis abgelegt und bei Aufruf des Skripts „ordner_loeschen.py“ übergeben. Die Löschung jedes einzelnen Verzeichnisses wurde durch das Skript in einer Log-Datei fixiert – dies ist auch an der Übereinstimmung von Input- und Logfile nachvollziehbar. Zusätzlich ist das Löschen in der Bildschirmausgabe zu verfolgen. Da die Löschung auf intellektuellen Kriterien fußte, wurden die Verzeichnisse mithilfe des Skripts vom Datenträger gelöscht. Bei den folgenden Operationen innerhalb des Workflows wird dies hingegen anders gehandhabt.[105] Gegenüber einem händischen Vorgehen – Suche und Löschen im Windows Explorer – bleibt das Vorgehen durch die Logfiles nachvollziehbar, auch wird die Möglichkeit ungewollter Löschungen durch die sukzessive Listenerstellung und den automatisierten Löschvorgang vermindert. Der eigentliche Löschvorgang war nach der sorgfältigen Vorbereitung eine Sache von Sekunden. Die Sammlung wurde durch diesen Schritt auf 187.257 Dateien mit einem Datenvolumen 404 GB reduziert.

5.3 Auflösen bestimmter Unterverzeichnisse

Aufgabenbereich:	SIP-Formierung
Tools:	TreeSize Professional, Total Commander

Nun folgte das händische Umstrukturieren derjenigen Verzeichnisse mit sieben oder mehr Unterverzeichnissen. Der Schritt wurde an dieser Stelle im Workflow platziert, da bei einer früheren Auflösung auch solche Verzeichnisse hätten umstrukturiert werden müssen, die erst im vorherigen Schritt gelöscht wurden – das Vorgehen in dieser Reihenfolge war demnach ökonomischer, wenn auch die automatisierte Skript-Kette hierdurch unterbrochen wurde, was formal etwas unelegant erscheinen mag, aber in der Praxis keinerlei Schwierigkeiten bedeutete.

Nachdem die aufzulösenden Verzeichnisse in der Vorbereitung mit TreeSize Pro identifiziert wurden, erfolgte die Umstrukturierung in Total Commander, wobei noch einmal zwei unterschiedliche Verfahrensweisen unterschieden wurden, die sich aus dem Strukturierungsprinzip Aipperspachs ergaben. Einige Unterverzeichnisse folgten der Nummerierung, die Aipperspach für die Ordnernamen auf der Hauptebene – F:\Aipp\Ereignis\ – verwendet hatte. Bei der Benennung anderer Unterverzeichnisse griff Aipperspach hingegen auf eine eigenständige Nummerierung zurück oder verzichtete gänzlich auf vorangestellte Ziffern. Diejenigen Verzeichnisse, die der Hauptnummerierung folgten, wurden einfach auf die Hauptebene verschoben und so in die bestehende Reihung eingegliedert.

In vierzehn Verzeichnissen mussten die Unterverzeichnisse jedoch umbenannt werden. Auch dies wurde in Total Commander ausgeführt. Das integrierte Multi-Rename Tool gewährleistet in solchen Fällen ein

105 Nochmals sei an dieser Stelle darauf hingewiesen, dass der Workflow an einer verifizierten Kopie vollzogen wurde, sodass auch durch diesen Löschvorgang keine Daten unwiderruflich verloren gehen konnten. Auf die unterschiedliche Vorgehensweise wird im Weiteren noch eingegangen.

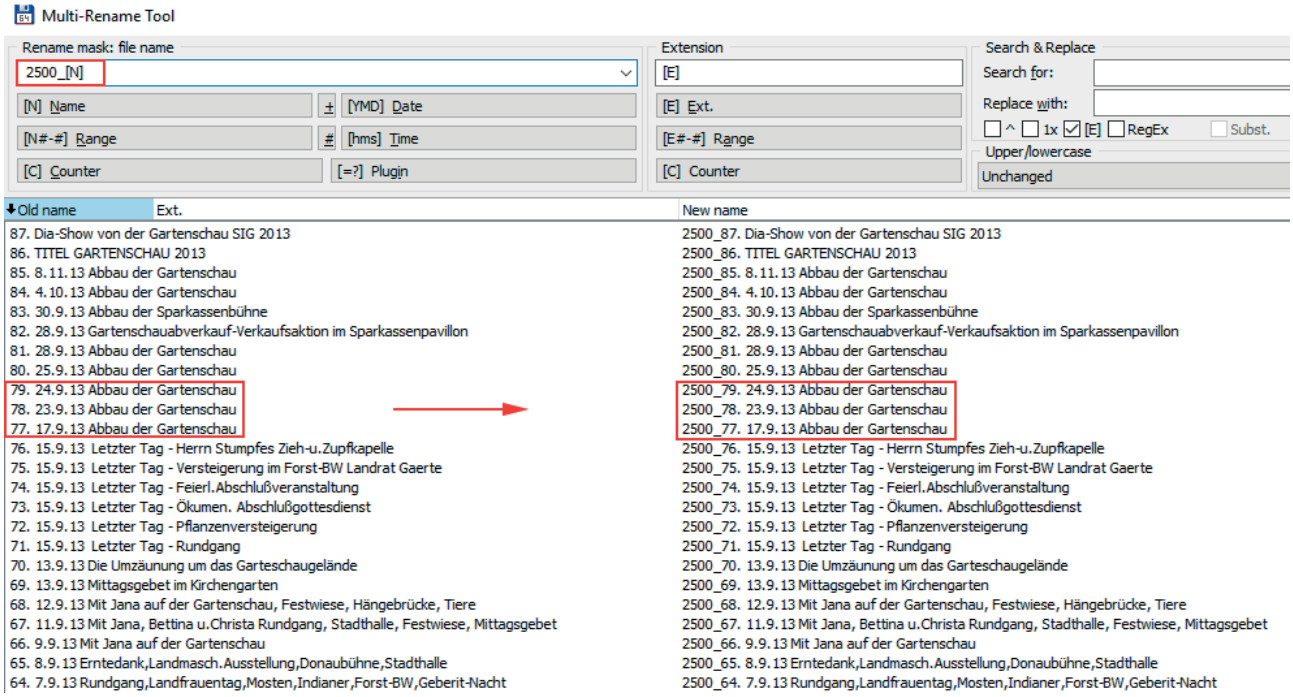

Abb. 7: Mehrfach-Umbenennen von Verzeichnissen via Multi-Rename Tool in Total Commander.

komfortables Vorgehen.[106] Durch Voranstellen der Ordnernummer des Hauptordners blieb der Ursprung dieser Ordner auch nach dem anschließenden Verschieben auf die Hauptebene ersichtlich. Alle leeren Hauptordner, die ihrerseits die Unterordner, nicht aber Dateien enthalten hatten, wurden gelöscht.

5.4 Redigieren der Ordnernamen

Aufgabenbereich:	SIP-Formierung, Erschließung
Eingesetzte Skripte:	ordner_liste.py, ordner_umbenennen.py
Inputfiles:	redigierte_verzeichnisnamen_anwenden.csv
Tools:	TreeSize Pro, Notepad++, Microsoft Word, Microsoft Excel

Die zum Teil recht langen Ordnerbezeichnungen boten zuverlässige Informationen zu Anlässen, Aufnahmeorten und tagesgenaue Aufnahmedaten, wiesen aber einige formale Unregelmäßigkeiten auf, die beseitigt werden sollten, um rechtschreibkonforme Bezeichnungen zu übernehmen, die auch als Erschließungsmetadaten Verwendung finden konnten. Zu korrigieren waren fehlerhaft – entweder mehrfach oder an der falschen Stelle – gesetzte sowie fehlende Leer- und Satzzeichen, mangelhafte Groß-/Kleinschreibung und

106 Im Beispiel in Abb. 7 wurden 95 Verzeichnisse mit dem Präfix „2500“ versehen und anschließend durch Verschieben zu Hauptordnern gemacht, wohingegen der vormalige Hauptordner E:\Aipp\2500. GARTENSCHAU 2013 in SIGMARINGEN vom 11.5. - 15.9.2013\ gelöscht wurde (Vgl. auch S. 33, Abb. 2).

sonstige Rechtschreibfehler. Zudem sollten häufig verwendete Abkürzungen aufgelöst werden (vgl. nochmals S. 33, Abb. 2).

Zunächst wurde mit dem Skript „ordner_liste.py“ eine Auflistung aller nach den ersten beiden Bearbeitungsschritten verbliebenen Verzeichnisse angelegt. Die Rechtschreibkorrekturen wurden mithilfe der Rechtschreibüberprüfung in Microsoft Word ausgeführt, die auch die Abkürzungen als fehlerhafte Schreibweisen erfasste.[107] Hierzu musste die Verzeichnisliste lediglich in ein neues Word-Dokument kopiert werden. Parallel wurde händisch die fehlerhafte Groß-/Kleinschreibung korrigiert – sehr häufig lag eine durchgehende Großschreibung vor.

Anschließend wurden die korrigierten Pfade wiederum in ein neues Dokument in Notepad++ eingefügt, wo, teils mittels Regulären Ausdrücken oder durch einfaches „Suchen und Ersetzen“, die fehlerhafte Interpunktion korrigiert und die mehrfache Leerzeichen entfernt wurden. Auch diese Schritte wären zwar in Word auszuführen gewesen, gestalteten sich aber aufgrund des vertrauteren Umgangs mit Regulären Ausdrücken in Notepad++ auf diesem Weg effizienter.

Die unredigierte Verzeichnisliste wurde nun mit Microsoft Excel geöffnet, um eine zusätzliche Spalte mit den korrigierten Verzeichnispfaden anzulegen.[108] In Notepad++ wurde diese Liste in das gewünschte CSV-Format gebracht und als „redigierte_verzeichnisnamen_anwenden.csv“ im Input-Verzeichnis abgelegt. Diese Datei wurde schließlich dem Skript „ordner_umbenennen.py“ übergeben, das die vollautomatische Umbenennung auf Basis des Abgleichs der beiden Spalten im Input-File vornahm. So wurden schließlich 941 Verzeichnisse korrigiert. Zur Überprüfung, ob die Umbenennung einzelner Ordner nicht zu Pfadüberlängen geführt hatte, kam erneut TreeSize Pro zum Einsatz. Tatsächlich traten in einem Verzeichnis Dateipfade mit einer Länge zwischen 265 und 267 Zeichen auf – der Verzeichnisname wurde um die notwendige Anzahl von Zeichen gekürzt.[109]

Das Redigieren der Verzeichnisnamen nahm noch immer einige Zeit in Anspruch, wurde jedoch durch den Einsatz der Rechtschreibkorrektur und einiger „Suchen und Ersetzen“-Operationen deutlich verkürzt.[110] Durch den Skript-Einsatz ging das Umbenennen schnell von der Hand; jede Änderung bleibt anhand des Logfiles nachvollziehbar.

107 Analog dazu kann auch mit LibreOffice Writer verfahren werden.
108 Aufgrund der komfortableren Arbeit mit CSV-Dateien ist der Einsatz von LibreOffice Calc gegenüber Excel vorzuziehen. Der nachfolgende Schritt entfällt bei der Anwendung von Calc – da Zeichenkodierung, Text- und Feldtrenner frei gewählt werden können, müssen diesbezüglich keine nachträglichen Anpassungen in Notepad++ vorgenommen werden.
109 Über die Zeichenlängen der jeweils in einer Zeile geführten Pfade wäre dies in Notepad++ bereits vorab zu erkennen gewesen, da jedoch nicht damit gerechnet wurde, dass das Problem auftreten würde, wurde sich auf die Überprüfung nach Umbenennung beschränkt.
110 Einzig die durchgehende Großschreibung wurde nicht automatisiert. Der Einsatz der TextFx-Funktion „Proper Case“ (in Notepad++) beseitigte zwar jede durchgehende Großschreibung, sorgte aber insgesamt auch nicht für eine fehlerfreie Schreibung, weshalb letztlich auf die Anwendung verzichtet und das Problem händisch angegangen wurde. In anderen Fällen ist die Funktion womöglich aber hilfreich.

5.5 Erstellen eines Referenz-Clusters zur späteren Tafelerkennung

Aufgabenbereich:	Bewertung
Eingesetzte Skripte:	bilder_kopiere_erste.py
Inputfiles:	–
Tools:	–

Bereits mehrfach wurde auf die Bedeutung der ersten Dateien innerhalb eines jeden Verzeichnisses verwiesen, bei denen es sich in der Regel um Kontextinformationen handelte. Da in einem der weiteren Schritte (5.10) Tafeln und Zeitungsauschnitte – die sich zum Teil auch über einzelne Verzeichnisse verteilten, also nicht nur am Anfang eines Verzeichnisses lagen – durch bildbasierte Suchen gefunden werden sollten, bedurfte es vor dem Einebnen der Ordnerstruktur eines Zwischenschritts. Anhand einer stichprobenartigen Durchsicht der Verzeichnisse wurde entschieden, die ersten vier Dateien eines jeden Verzeichnisses zu berücksichtigen (relevant auch für die Schritte 5.6 und 5.7). In der Regel handelte es sich zwar nur bei den ersten beiden Bildern um Tafeln und Zeitungen, teils auch Karten, um jedoch eine möglichst vollständige Auswahl – gleichzeitig aber nicht übermäßig viele False-Positives, die eine nachträgliche intellektuelle Selektion aufwändiger hätten werden lassen – zu erhalten, wurde die Auswahl auf vier Dateien erhöht.

Mit Ausführung des Skripts „bilder_kopiere_erste.py“ wurde nun ein Verzeichnis „erste_dateien“ im Skript-Verzeichnis angelegt (vgl. S. 33, Abb. 2), in das automatisch die ersten vier Bilddateien eines jeden Verzeichnisses kopiert wurden. Da noch keine Ordner – bis auf diejenigen die vollständig gelöscht wurden – in ihrer Struktur verändert wurden, war dies vor Ausführung des nächsten Schritts noch möglich. Später diente eine Auswahl dieser Bilder – 2.752 Bilder wurden kopiert – dann als Referenz für die bildbasierte Suche.

5.6 Verzeichnisstruktur einebnen

Aufgabenbereich:	SIP-Formierung
Eingesetzte Skripte:	ordner_einebnen.py
Inputfiles:	–
Tools:	–

Um die SIPs auf Ordnerebene zu formieren, sollte nun die Verzeichnisstruktur abgeflacht werden. Nachdem bereits händisch alle Verzeichnisse mit sieben oder mehr Unterverzeichnissen aufgelöst worden waren, würde dieser Schritt sämtliche Unterverzeichnisse auflösen, sodass lediglich die Hauptordner auf Ereignisebene (F:\Aipp\Ereignis) übrigbleiben würden.

Für diesen Schritt war lediglich das Skript „ordner_einebenen.py“ auszuführen. Das Skript arbeitet sich ausgehend vom angegebenen Stammpfad (hier F:\Aipp) durch die einzelnen Verzeichnisse und löscht sukzessive

alle in den Hauptordnern (Also unterhalb der Ebene F:\Aipp\Ereignis) vorhandenen Unterordner, nachdem die darin enthaltenen Dateien in den Hauptordner verschoben wurden. Für das Verschieben der Dateien war aus zweierlei Gründen ein Umbenennungsprinzip zu ersinnen und anzuwenden: 1. Namenskonflikte mussten vermieden werden. 2. Sollte der Ursprung der Dateien erkennbar bleiben.

Das Skript verschiebt die Dateien daher nicht nur, sondern nimmt eine Umbenennung nach dem Muster „Bezeichnung des aufgelösten Ordners“ + „_“ + „Dateiname“ + „Dateiendung“ vor. Sofern mehrere Ordner gleichzeitig aufgelöst werden – die verschobenen Dateien also beispielsweise aus F:\Aipp\Ereignis\Unterordner1\Unterordner2 stammen und in F:\Aipp\Ereignis\ verschoben werden und „Unterordner1“ sowie „Unterordner2“ anschließend gelöscht werden – wird das Präfix aus den Bezeichnungen der aufgelösten Ordner zusammengesetzt: „Bezeichnung des aufgelösten Ordners1“ + „_“ „Bezeichnung des aufgelösten Ordners2“ + „_“ + „Dateiname“ + „Dateiendung“. Durch die Verwendung von „_“ als Trennzeichen, das im Prinzip in der gesamten Pfadzusammensetzung lediglich den Pfadtrenner „/“ ersetzt, bleiben sämtliche Pfadlängen unverändert, sodass Komplikationen ausgeschlossen sind. Neben sämtlichen Lösch-, Verschiebe- und Umbenennungsoperationen loggt das Skript auch die ersten vier Bilddateien eines jeden aufgelösten Verzeichnisses in einer CSV-Datei.

Mit Ausführung von „ordner_einebnen.py“ wurden sämtliche Operationen innerhalb einer Minute abgearbeitet und 88 Ordner aufgelöst.

5.7 Zufallsselektion

Aufgabenbereich:	Bewertung
Eingesetzte Skripte:	dateien_zufallsselektion.py
Inputfiles:	erste_bilder_aus_unterordnern.csv
Tools:	Notepad++

Der folgende Schritt bildete in mehrfacher Hinsicht den Kern des Workflows. Zum einen wurde hier eine vollautomatische Reduktion der Sammlung vorgenommen – es wurden erstmals Bilder ausgewählt und kassiert. Dies geschah durch ein abgewandeltes „systematic random sampling“[111], das auf dem Prinzip basiert: „Behalte eine festzulegende Anzahl Bilddateien vom Anfang eines jeden Verzeichnisses sowie jede weitere x-te Bilddatei, kassiere alle anderen Bilddateien“. Darüber hinaus bedingte die Anwendung des Skripts auch die Vorgehensweise der nachfolgenden Schritte, da die nicht-ausgewählten Dateien nicht gelöscht wurden. Stattdessen wurde im Sammlungsordner ein neues Unterverzeichnis „kassierte_dateien“ (hier F:\Aipp\kassierte_dateien) angelegt, in das die komplette Verzeichnisstruktur gespiegelt und die zu kassierenden Dateien verschoben wurden; die zu archivierenden Dateien verblieben in den ursprünglichen Verzeichnissen. Das Skript erlaubt zudem durch den Input von CSV-Dateien die Angabe solcher Dateien, die in jedem Fall kassiert oder aber archiviert werden sollen.

111 Vgl. Anm. 62.

Im vorangegangenen Schritt wurden einige Verzeichnisse aufgelöst und die neuen Pfade der ersten vier Dateien aus einem jeden aufgelösten Verzeichnis geloggt. Dieses Logging war nötig, da diese Dateien nach dem Verschieben nicht mehr am Anfang eines Verzeichnisses lagen, sondern im Hauptordner hinter denjenigen Dateien, die sich dort bereits befunden hatten und es einer Möglichkeit bedurfte, diese Dateien vom Zufallsselektor auszunehmen. Da aus jedem ursprünglichen Verzeichnis die ersten vier Dateien übernommen werden sollten, wurde zunächst der Zufallsselektor so programmiert, dass die ersten vier Bilddateien und weiterhin jedes siebte Bild eines jeden Verzeichnisses ausgewählt werden sollten. Beide Parameter sind im Skript frei konfigurierbar. Zusätzlich wurde die im vorausgegangenen Schritt angelegte Logdatei „erste_bilder_aus_unterordnern.csv“ nun als Inputfile an das Skript übergeben, sodass diese Bilder von der Zufallsauswahl unberührt blieben und in den ursprünglichen Verzeichnissen verblieben.

Mit der Entscheidung, dass jedes weitere siebte Bild übernommen werden sollte, wurde entscheidend über die Größe der als archivwürdig bewerteten Sammlung bestimmt. Angesichts des ursprünglich gesetzten Ziels einer Reduktion des Datenvolumens auf ungefähr 10 % hätte die Zufallsauswahl schärfer angesetzt werden können, jedoch wurde ein gewisser Kompromiss eingegangen, sodass auch aus Verzeichnissen mit wenigen Bildern nicht nur die ersten vier Bilder ausgewählt wurden und zudem nach Möglichkeit aus einzelnen kleineren Teilserien mindestens ein Bild erhalten blieb. Nach Ausführen des Skripts blieben noch 17 % der Dateien in ihren ursprünglichen Verzeichnissen – im Folgenden als Auswahl bezeichnet, beispielsweise F:\Aipp\Ereignis\bildx.jpg –, 83 % wurden vorläufig „kassiert“. Das als „Kassieren“ bezeichnete Verschieben erfolgte nach dem Muster F:\Aipp\Ereignis\bildy.jpg nach F:\Aipp\kassierte_dateien\Ereignis\bildy.jpg.

5.8 Kassieren nicht archivwürdiger Dateiformate

Aufgabenbereich:	Bewertung
Eingesetzte Skripte:	dateien_liste.py, dateien_kassieren.py
Inputfiles:	sontige_dateien_kassieren.csv
Tools:	Notepad++

Da durch die Zufallsauswahl lediglich Bilddateien berücksichtigt wurden, befanden sich nach wie vor Dateien in der Auswahl, die vorab bereits aufgrund ihres Dateityps als nicht archivwürdig bewertet wurden (vgl. nochmals S. 85, Tabelle 1). Um welche Dateien es sich handelte, war bereits aus der anfänglichen Analyse mit TreeSize Pro bekannt. Das weitere Vorgehen soll jedoch demonstrieren, wie Dateitypen-bezogene Operationen auch ohne zusätzliches Tool ausgeführt werden können. Um eine Übersicht über die Dateien zu erhalten, wurde mit dem Skript „dateien_liste.py“ eine Aufzählung aller in der Auswahl liegenden Nicht-Bilddateien erstellt. Da im Testlauf festgestellt wurde, dass DB-Dateien während der Arbeit vom System teilweise neu angelegt werden, sollte deren Löschung an das Ende des Workflows verschoben. Dafür wurden diese schlichtweg mithilfe von Notepad++ aus dem Input-File entfernt. Die restlichen Dateien sollten jedoch gleich kassiert werden. Daher wurde das erstellte Input-File mit den Dateipfaden der verbliebenen 38 sonstigen Dateien dem Skript „dateien_kassieren.py“

übergeben. Die Dateien wurden also vorläufig kassiert, indem die entsprechenden Ordner in „kassierte_dateien“ verschoben wurden.[112] Somit befanden sich – abgesehen von den „unsichtbaren“ DB-Dateien – nur noch JPG-Dateien in der Auswahl.

5.9 Kassieren unscharfer Fotos

Aufgabenbereich:	Bewertung
Eingesetzte Skripte:	bilder_unschaerfe_laplace.py, bilder_oeffnen.py, dateien_kassieren.py
Inputfiles:	auswahl_durchsuchen.csv, unscharfe_kassieren.csv
Tools:	Notepad++, IrfanView

Die im Weiteren erfolgten bildbasierten Schritte (5.9–5.11) dienten der kriteriengestützten Korrektur der vollzogenen Zufallsselektion. Zunächst sollten aus der getroffenen Auswahl unscharfe Bilder entfernt werden. Das Kriterium der technischen Unzulänglichkeit zur Bewertung von Fotos wurde bereits erwähnt, ist aber aus mehrfacher Hinsicht nicht unproblematisch.

Die Durchsicht der Sammlung Aipperspach hatte ergeben, dass sich keine falsch belichteten Aufnahmen in der Sammlung befanden. Dies war nicht weiter verwunderlich, da Aipperspach bereits eine Auswahl seiner Fotos zur Übergabe an das Archiv angelegt und in diesem Rahmen offenbar stark über- oder unterbelichtete Bilder aussortiert hatte. Als Kriterium der technischen Unzulänglichkeit blieb damit die Unschärfe. Bei der Durchsicht der Sammlung waren einige Fotos mit unzureichender Fokussierung gesichtet worden. Grundsätzlich war dies auch darauf zurückzuführen, dass es sich um Amateuraufnahmen handelte, die insgesamt nicht an den Qualitätsmerkmalen professioneller Fotografien gemessen werden dürfen, was wiederum erklärt, warum Aipperspach selbst diese Aufnahmen nicht aussortierte. Diese Charakteristik sollte durch ein Entfernen unscharfer Bilder einerseits nicht verwischt werden. Andererseits fiel hier jedoch auch eine entscheidende Einschränkung der (teil-)automatisiert vorgenommenen Schärfebewertung weg, die insbesondere bei professionellen oder künstlerischen Fotografien zum Tragen kommt, bei denen Unschärfe ein bewusst eingesetztes Stilmittel sein kann, was dann zum Problem wird, wenn die eingesetzte Software nicht zwischen bewusst eingesetzter und versehentlicher Unschärfe zu unterscheiden vermag. Der künstlerische Einsatz von Unschärfe konnte im vorliegenden Fall ausgeschlossen werden, gleichwohl führte die Verwendung einer weit geöffneten Blende zum Teil auch hier für Freistellungseffekte, wenn beispielsweise die fotografierte Person im Vordergrund scharf, der Hintergrund hingegen unscharf abgebildet wurde – solche Aufnahmen waren jedoch selten. Zur Entfernung unscharfer Bilder wurde „bilder_unschaerfe_laplace.py“ geschrieben und angewendet, wobei entschieden wurde, nur mehr sehr unscharf erscheinende Bilder zu kassieren.

112 Beispielsweise: E:\Aipp\21 Vatertagstreffen der Musikkapelle in Oberschmeien an Christi Himmelfahrt 5.5.05\(216).mov nach E:\Aipp\kassierte_dateien\21 Vatertagstreffen der Musikkapelle in Oberschmeien an Christi Himmelfahrt 5.5.05\(216).mov.

Das Skript nutzt Teile von OpenCV – eine für Python und andere Programmiersprachen frei verfügbare Bibliothek zur Bildverarbeitung (Computer Vision).[113] Aus einer Liste mit Bildpfaden wird Bild für Bild geöffnet und zunächst in Graustufen umgewandelt. Um die weiteren Berechnungszeit zu verkürzen, wird das Bild außerdem auf 25 % der Originalgröße skaliert. Diese Änderungen am Bild finden nur an der geöffneten Kopie, nicht aber am Original statt. Die Dateien bleiben also während der gesamten Verarbeitung unverändert. Dies gilt für alle Skripte. Anschließend wird auf eine in OpenCV enthaltene Funktion namens „Laplacian" zurückgegriffen, bei der es sich um eine Implementierung des diskreten Laplace-Operators handelt.[114] Dieser wird – neben anderen Operatoren – in der Bildverarbeitung zur Kantendetektion genutzt und findet sich in gängigen Bildverarbeitungsprogrammen als Laplace-Filter, der ein Bild mit dunklen Flächen und hellen Kanten ausgibt. Für ein besseres Verständnis seien die Zusammenhänge kurz skizziert. Die Laplacian-Funktion berechnet im Graustufenbild anhand einer vorgegebenen Matrix (Faltungsmaske) die Intensitätsunterschiede (Helligkeit) benachbarter Pixel. Dort wo im Ergebnis besonders hohe Abweichungen auftreten, finden sich im Bild Kanten. Besonders leicht lässt sich das Prinzip an einer – vorliegenden oder gedachten – Schwarz-Weiß-Fotografie nachvollziehen: Die Flächen eines abgebildeten Objekts erscheinen homogen und weisen weiche Helligkeitsübergänge auf, härtere Übergänge hingegen finden sich an Kanten. Was sagen aber Kanten innerhalb eines Bildes über die Bildschärfe aus? Hier wird sich einer einfachen Annahme bedient: Je mehr Kanten innerhalb eines Bildes, desto schärfer das Bild. Besser vorstellen lässt sich die umgekehrte Korrelation: Je unschärfer das Bild, desto mehr verschwimmen auch die Kanten. Mathematisch wird dieser Zusammenhang auswertbar, indem die Varianz der Ergebnismatrix berechnet wird: Je höher die Varianz, desto mehr Kanten, desto schärfer das Bild.[115] Um Bilder mit einer gewissen Unschärfe zu finden, bedarf es lediglich der Festlegung eines geeigneten Grenzwerts.[116]

Als Inputfile diente „auswahl_durchsuchen.csv", eine Liste der Bilddateien innerhalb der Auswahl, also allen Bildern, die nicht vorläufig kassiert wurden, denn nur diese Bilder sollten nochmals gefiltert werden. Eine solche Liste wurde in Schritt 5.7 ausgegeben.[117] Nach Öffnen von „bilder_unschaerfe.py" wurde unter Angabe dieses Inputfiles nach sehr unscharfen Bildern gesucht. Die als CSV-Datei ausgegebene Ergebnisliste mit 411 Pfaden hätte theoretisch sofort an das Skript „dateien_kassiern.py" übergegeben werden können. Jedoch ermöglicht „bilder_unschaerfe.py" durch zusätzliche Ausgabe einer TXT-Datei ein unkompliziertes Überprüfen des Such-Ergebnisses mittels IrfanView. Die TXT-Datei wurde an das Skript „bilder_oeffnen.py" übergeben, woraufhin sich die Thumbnail-Ansicht von IrfanView mit den gefundenen Bildern öffnete.[118] So konnte das Ergebnis schnell intellektuell begutachtet werden. Durch Entfernen von Bildern aus der Thumbnail-Ansicht und anschließendes Speichern der Bildpfade – IrfanView ermöglicht das bequeme Exportie-

113 Vgl. OpenCV website.
114 Nähere Informationen zur Funktion „Laplacian" in OpenCV finden sich hier: OpenCV 2.4.13.3 documentation. Image Filtering: Laplacian.
115 Technisch beschrieben findet sich dies in folgendem Artikel: *Pech-Pacheco* u. a., Diatom autofocusing in brightfield microscopy: a comparative study, S. 315f. Die Realisierung in Python-Code folgt den Ausführungen bei: *Rosebrock*, Blur detection with OpenCV – PyImageSearch.
116 Vgl. S. 112, Python-Skripte XIII, Z. 77.
117 Vgl. in Abb. 2 die Datei „uebersicht_dateien_30-07-2017_11-15-55.csv". Eine solche Auflistung wäre aber auch mit dem Skript „dateien_liste.py" schnell neu anzulegen gewesen.
118 Gegenüber dem direkten Übergeben der Liste per Aufruf von IrfanView in der Eingabeaufforderung, bietet das Skript den Vorteil der Pfadlängenüberprüfung. Darauf wird im nächsten Schritt noch eingegangen.

ren einer Pfadliste aller sich in der Thumbnail-Ansicht befindenden Bilder als TXT-Datei – konnte mittels Notepad++ schließlich eine 348 Pfade umfassende Liste als „unscharfe_kassieren.csv“ im Input-Verzeichnis abgelegt werden. Ein Aufruf dieser Datei mittels „dateien_kassieren.py“ verschob diese Dateien wiederum in das entsprechende Unterverzeichnis im Ordner „kassierte_dateien“. Das gesamte Vorgehen nahm, abgesehen von der benötigten Rechenzeit, lediglich einige Minuten in Anspruch, wobei die meiste Zeit auf die intellektuelle Nachselektion der zu kassierenden Bilder fiel.

5.10 Tafeln und Zeitungen zurückführen

Aufgabenbereich:	Bewertung, Erschließung (5.10.2)
Eingesetzte Skripte:	dateien_liste.py, bilder_oeffnen.py, bilder_vergleich.py, bilder_vergleich_2px.py, dateien_kassierte_einsortieren.py, bilder_ocr.py
Inputfiles:	tafeln_referenz.csv, kassiertedateien_nachtafelndurchsuchen.csv, auswahl_durchsuchen.csv, tafeln_ruecksortieren.csv, tafeln_ocr_erkennung.csv, zeitungen_refrenz.csv, zeitungen_kopieren.csv
Tools:	Notepad++, IrfanView, Windows Explorer

5.10.1 Tafeln zurückführen

Durch die Zufallsauswahl wurden Tafeln kassiert, die jedoch in jedem Fall archiviert werden sollten. Daher fand nun das Skript „bilder_vergleich.py“ Anwendung, das ebenfalls die Computer-Vision-Bibliothek OpenCV einbezieht.

Die grundlegende Idee hinter dem Skript ist die, dass sich mit den Tafeln unter den in Schritt 5.5 herauskopierten ersten vier Bilddateien, alle übrigen Tafeln finden lassen sollten, da diese auf den gleichen Hintergrundbildern basierten. Das Skript öffnet nacheinander Bilder einer Referenzauswahl, verkleinert diese stark und speichert von den verbleibenden Pixeln eines jeden Bildes sämtliche RGB-Werte – pro Pixel also je einen Wert für Rot, Gelb und Blau. Anschließend wird die nach diesen Referenzwerten zu Durchsuchende Bildauswahl durchlaufen, wobei nun die RGB-Werte eines jeden Bildes mit den gespeicherten Werten aller Referenzbilder mit übereinstimmenden Abmessungen abgeglichen werden. Liegt eine Übereinstimmung innerhalb des festgelegten Grenzwertes vor, erfolgt eine Bildschirmausgabe und der Bildpfad wird in einer CSV-Datei gespeichert, die am Ende des Prozesses einer Auflistung aller gefunden Bilder enthält.

Zunächst wurde mithilfe der Testsuite ein geeigneter Grenzwert zur Ermittlung der Tafeln ermittelt, sodass alle Tafeln, gleichzeitig aber möglichst wenige False-Positives gefunden wurden. Analog zur Vorgehensweise im vorherigen Schritt wurden mithilfe der Thumbnail-Ansicht in IrfanView aus den herauskopierten Dateien eine Tafelauswahl erstellt, die mit Notepad++ als Referenzliste „tafeln_referenz.csv“ abgelegt

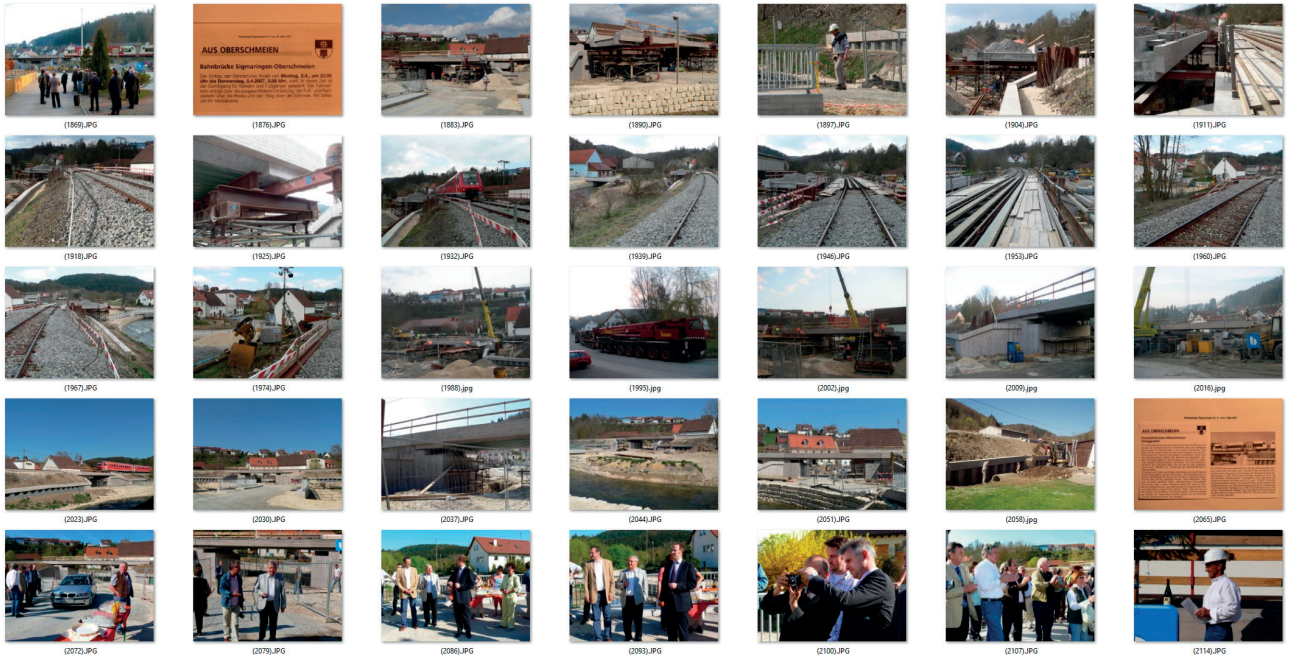

Abb. 8: Ausschnitt eines bewerteten Verzeichnisses vor der automatisierten Tafel-Rückführung.
Fotos: Gunter Aipperspach. Die Verwertungsrechte liegen beim Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen.

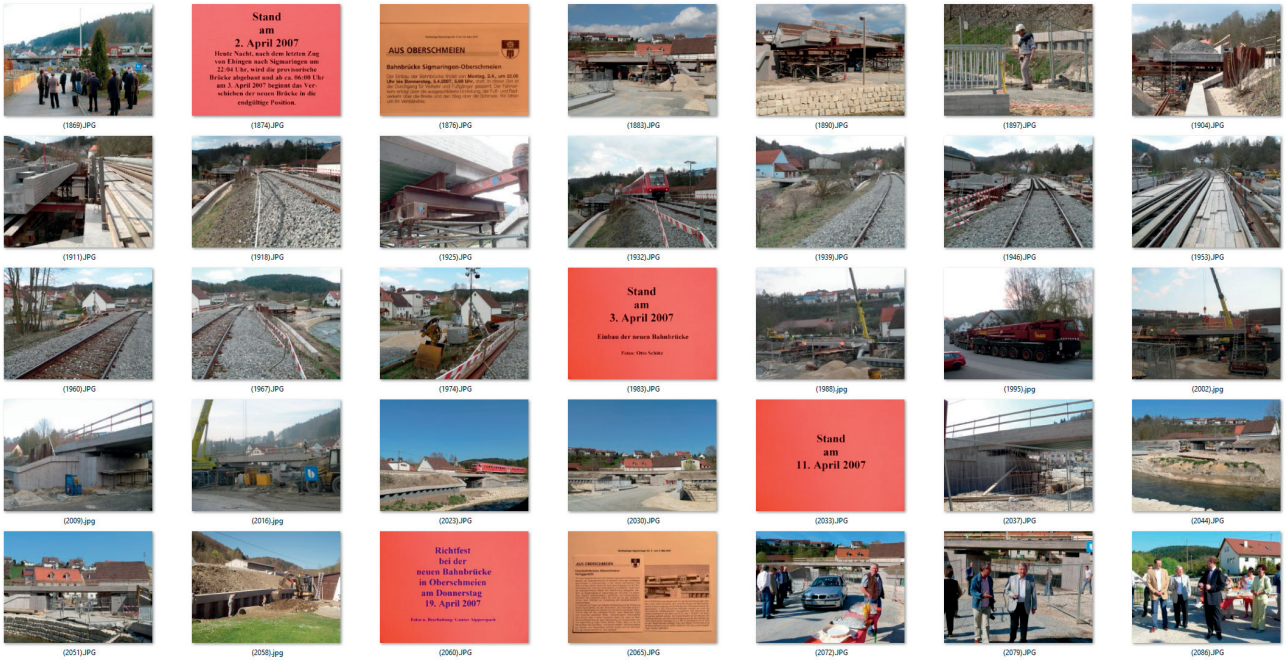

Abb. 9: Ausschnitt des in Abb. 8 gezeigten bewerteten Verzeichnisses nach der automatisierten Tafel-Rückführung.
Fotos: Gunter Aipperspach. Die Verwertungsrechte liegen beim Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen.

wurde. Mit „dateien_liste.py“ wurde anschließend eine Auflistung aller Bilder in „kassierte_dateien“ erstellt und als „kassierte_dateien_nachtafelndurchsuchen.csv“ im Input-Verzeichnis abgelegt. Nun konnten mittels „bilder_vergleich.py“ die vorläufig kassierten Dateien (155022 Bilder) nach Tafeln durchsucht werden. Das Ergebnis der Suche wurde wiederum begutachtet, um False-Positives zu beseitigen.

Mit Anlegen des Ordners „kassierte_dateien“ waren jedoch Pfadlängen von mehr als 259 Zeichen entstanden. Für den Workflow war dies bis hierhin nicht von Bedeutung, da die Skripte auch mit langen Pfaden zurechtkommen.[119] Anders hingegen IrfanView: Das Programm übergeht beim Öffnen der Pfade mit Überlänge, ohne dies dem Benutzer anzuzeigen, sodass die betroffenen Bilder womöglich unbemerkt nicht in der Thumbnail-Ansicht auftauchen. Dies registriert jedoch das Skript „bilder_oeffnen.py“ und gibt die von IrfanView nicht geöffneten Dateipfade auf dem Bildschirm und in einer CSV-Datei aus, sodass die Fotos über den Windows Explorer inspiziert werden können. Die nicht-betroffenen Dateien werden hingegen am bequemsten wie im vorangegangenen Schritt in IrfanView begutachtet und ausgewählt.

Tatsächlich waren vier Verzeichnisse im Suchergebnis von dem Problem betroffen, die dort gefundenen Bilder wurden überprüft und die Tafel-Pfade zu der in IrfanView vorgenommenen Nach-Selektion hinzugefügt. Schließlich konnte die 967 Einträge umfassende Auflistung kassierter Tafeln „tafeln_ruecksortieren.csv“ an das Skript „dateien_kassierte_einsortieren.py“ übergeben werden. Dieses Skript sortierte die Dateien in die entsprechenden Verzeichnisse innerhalb der Auswahl ein und stellte somit die von Aipperspach durch die Tafeln realisierte Struktur innerhalb der Bildserien wieder her (vgl. Abb. 8 und 9).

5.10.2 Text-Erkennung (OCR) der Tafeltexte

Zunächst war geplant, die Tafel-Texte für die Erschließung der Sammlung zu verwenden, ohne dass dies hinterher realisiert wurde. Die Extraktion wurde jedoch in den Workflow integriert und durchgeführt. Noch bevor die Tafeln zurücksortiert werden konnten, war daher auch die bestehende Auswahl mithilfe der angelegten Tafel-Referenz zu durchsuchen. Somit wurden auch die durch den Zufallsselektor ausgewählten Tafeln erfasst. Hierzu wurde analog zum soeben beschriebenen Vorgehen verfahren. Anschließend wurde aus der Liste der vorläufig kassierten, später zurückgeführten Tafeln und der Tafeln innerhalb der Auswahl eine Gesamtliste der Pfade aller zu archivierenden Tafeln erstellt und als „tafeln_ocr_erkennung.csv“ im Input-Verzeichnis abgelegt.[120]

Das Skript „bilder_ocr.py“ nutzt neben OpenCV die mittlerweile von Google entwickelte Texterkennungssoftware Tesseract, die unter anderem für Windows, macOS und Linux frei verfügbar ist und sich mittels Texterkennungsdaten zur Zeichenerkennung für verschiedene Sprachen ausstatten lässt. Die

119 Im Rahmen der Zufallsauswahl hätten die Dateien z. B. auch nach F:\kass\Ereignis\ verschoben werden können, um dies zu verhindern. Das Skript wäre nur entsprechend zu konfigurieren gewesen. Jedoch sollte der Ansatz eines einzigen Arbeitsverzeichnisses, nämlich F:\Aipp\, nicht aufgegeben werden – mit der einzigen im Folgenden beschriebenen Konsequenz.

120 Natürlich hätte auch nach dem Rückführen der vorläufig kassierten Tafeln eine Suche innerhalb der Auswahl ausgeführt werden können, jedoch sparte das hier beschriebene Vorgehen Zeit, da die rund 1.000 zurückgeführten Tafeln so nicht ein weiteres Mal analysiert werden mussten.

Abb. 10: Das Bild wird automatisch aufbereitet. Anschließend erfolgt die Texterkennung durch Tesseract.
Bild: Gunter Aipperspach. Die Verwertungsrechte liegen beim Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Sigmaringen.

Software kommt ohne grafische Oberfläche aus, kann aber beispielsweise mit ImageMagick oder Python angesprochen werden.[121] Zur Aufbereitung der Tafeln für die Texterkennungssoftware werden bei Skriptaufruf die per Input-File gelieferten Bildpfade nacheinander geöffnet. Das jeweilige Bild wird zunächst in Graustufen umgewandelt und verkleinert. Ferner wird durch Histogrammäqualisation eine Kontrastverstärkung des Schriftbilds erzielt. Zudem wird das Bild leicht aufgehellt. Eine abschließende Rauschunterdrückung dient ebenfalls der Verbesserung der Erkennungsrate. Erst dann wird die derart aufbereitete Kopie des Bildes zur Texterkennung an Tesseract übergeben und der Text extrahiert (vgl. Abb. 10). Der Reihe nach wird mit allen Bildern derart verfahren, bis die gesamte Liste abgearbeitet ist.

Die exakten Einstellungen zur Bildaufbereitung sind abhängig vom Ausgangsbild und wurden speziell auf die Tafeln abgestimmt. Nach Rückführung der vorläufig kassierten Tafeln wurde das Skript unter Angabe des Input-Files „tafeln_ocr_erkennung.csv“ ausgeführt. Innerhalb von zweieinhalb Stunden extrahierte das Skript 2612 Tafeltexte und legte eine CSV-Datei in der Form ‘„Dateipfad“;“Text“’ an, die gemäß der ursprünglichen Planung als Alternative zur Erschließung über die Ordnernamen hätte dienen können.

5.10.3 Zeitungen zurückführen

Genau wie die Tafeln waren auch die Zeitungsausschnitte vorab als erhaltenswert eingestuft worden. Daher mussten ebenfalls die bei der Zufallsselektion vorläufig kassierten Ausschnitte gesammelt werden. Grundsätzlich wurde hier genauso verfahren wie bei der Rückführung der Tafeln, jedoch mit zwei bedeutenden Unterschieden. Jedoch war vorab entschieden worden, die Zeitungsausschnitte aufgrund der unterschiedlichen Rechtslage getrennt von der Fotosammlung Aipperspach zu handhaben. Daher galt es, diese nicht wie die Tafeln in die Auswahl zurückzuführen, sondern hieraus eigenständige SIPs zu formieren. Außerdem kam nicht das Skript „bilder_vergleich.py“ sondern „bilder_vergleich_2px.py“ zum Einsatz.

121 Der offizielle Entwicklungsstand findet sich auf GitHub: Tesseract Open Source OCR Engine; eine Windows-Version stellt die UB Mannheim zur Verfügung: Tesseract at UB Mannheim on GitHub.

Die Bezeichnung des letzteren Skripts soll bereits auf die Funktionsweise und den wesentlichsten Unterschied zum bereits angewendeten Vergleichs-Skript hinweisen. Der Abgleich zweier Bilder findet hier nicht anhand des gesamten Bildinhalts statt, sondern lediglich anhand zweier Pixel. Nun ist anhand zweier Bildpunkte zwar überhaupt nicht auf eine inhaltliche Ähnlichkeit zweier Bilder zu schließen, jedoch war dies hier auch nicht erforderlich. Viel mehr wird sich hier zunutze gemacht, dass die abfotografieren Zeitungsausschnitte in den Randbereichen sehr ähnliche Papiertöne – weißgraue oder bei fehlendem Weißabgleich aufgrund des verwendeten Kunstlichts gelbliche Töne – aufwiesen (vgl. die beiden Zeitungsausschnitte auf S. 50, Abb. 8). Es werden je ein Pixel des oberen und unteren Bildrands abgeglichen. Entgegen der Verwendung eines einzigen Pixels werden so bereits zahlreiche False-Positives vermieden; beispielsweise bei einem Abgleich eines das Zeitungspapier repräsentierenden weißen Referenz-Pixels mit dem Pixel eines wolkenverhangenen Himmels, der sich nur in der oberen Bildhälfte wiederfindet. Gleichwohl wird eine gewisse Zahl an False-Positives kaum vermeidbar sein. Zu einem gewissen Grad sind solche bei diesem Vorgehen auch von vornherein zu erwarten, da der Farbwertabgleich unter Berücksichtigung einer gewissen Toleranz stattfindet, um auch leichte Farbtonabweichungen gegenüber den Referenzbildern zuzulassen. Der Abgleich funktioniert aufgrund der Beschränkung auf zwei Pixel zudem völlig unabhängig vom Inhalt des jeweiligen Zeitungsausschnitts – es spielt also keine Rolle, ob dieser sich lediglich aus Text oder einer Kombination aus Text und einer Abbildung zusammensetzt. Textausschnitte können sich also optisch stark voneinander unterscheiden, weshalb ähnlichkeitsbasierte Suchen an ihre Grenzen stoßen, daher hatte beispielsweise auch dupeGuru in anfänglichen Tests keine brauchbaren Ergebnisse geliefert. Das Vorgehen folgt letztlich dem Prinzip der Beschränkung auf das kleinstmögliche übereinstimmende Merkmal. Dies ermöglicht die vergleichsweise einfache Realisierung – für eine bedeutend genauere Suche wären wesentlich komplexere Algorithmen vonnöten. Nicht zuletzt wird in bedeutendem Maß Rechenzeit gespart, da nur wenige Daten pro Bild verarbeitet werden müssen. Der Kompromiss besteht in der Inkaufnahme einer signifikanten Anzahl von False-Positives, der aber angesichts der vergleichsweise einfachen Nachselektion im Fall der Sammlung Aipperspach nicht sonderlich ins Gewicht fiel.

Nachdem die Arbeit an den Tafeln abgeschlossen war, wurde wiederum aus den herauskopierten ersten Dateien die benötigten Bilder zur Referenzbildung selektiert, um „zeitungen_refrenz.csv“ zu erstellen. Diese wurde an „bilder_vergleich_2px.py“ übergeben, womit die Suche nach Zeitungsausschnitten unter den verbliebenen 154.055 vorläufig kassierten Bilddateien gestartet wurde. Skriptgesteuerte Suche und anschließende Selektion ermöglichten schließlich die Ermittlung von 2.044 Zeitungsausschnitten aus rund 6.000 Treffern. Die Selektion in IrfanView gestaltete sich nicht zuletzt deswegen sehr leicht, weil in der Regel ganze Teilserien ähnlicher Bilder hintereinander zu False-Positives geführt hatten, die schnell erkannt, selektiert und auf einen Schlag aus der Thumbnail-Ansicht entfernt werden konnten. Da die Zeitungsausschnitte nicht in die jeweiligen Ordner in der Auswahl zurückgeführt werden sollten, wurde „dateien_kassierte_einsortieren.py“ so modifiziert, dass die Ausschnitte unter Aufrechterhaltung der Hauptordner-Struktur in ein eigenes Verzeichnis kopiert wurden, das im Weiteren unabhängig von der Fotosammlung behandelt wurde.[122] Auch dieser Arbeitsschritt konnte mithilfe der Skripte insgesamt in wenigen Stunden absolviert werden.

122 Es fand als ein Verschieben der Zeitungsausschnitte von F:\Aipp\kassierte_dateien\Ereignis\ausschnitt.jpg nach F:\Aipp\zeitungen\Ereignis\ausschnitt.jpg statt.

5.11 Vollständige Übernahmen realisieren

Aufgabenbereich:	Bewertung (vorab)
Eingesetzte Skripte:	ordner_vollstaendig_wiederherstellen.py
Inputfiles:	vollstaendige_uebernahmen_rueckfuehren.csv
Tools:	Notepad++

Vorab wurden 41 vollständig zu übernehmenden Ordner bestimmt, die einige besonders bedeutende Bildserien, darunter beispielsweise einige Serien mit Luftaufnahmen, enthielten.[123] Diese Ordner wurden von der bisherigen Aufbereitung und Bewertung bewusst nicht ausgenommen, da dies eine aufwändigere Programmierung der Berücksichtigung von Ausnahmen bedeutet hätte. Aus den Ordnern wurden also durch Zufallsselektion und möglicherweise durch das automatisierte Entfernen unscharfer Fotos einige Dateien vorläufig kassiert.

Es wurde daher ein Skript geschrieben, das alle Bilddateien eines Verzeichnisses zurück in den entsprechenden Ordner innerhalb der Auswahl verschieben sollte. Im Prinzip arbeitet das Skript genau wie „dateien_kassierte_einsortieren.py“, benötigt jedoch anders als dieses keine explizite Auflistung aller zurückzuführenden Dateien, was das Vorgehen an dieser Stelle des Workflows wesentlich erleichterte.

So bedurfte es lediglich einer Liste der vollständig zu übernehmenden Ordner, um mithilfe dieses Inputfiles alle kassierten Bilddateien zurück in die entsprechenden Verzeichnisse in der Auswahl zu verschieben und die vollständig zu übernehmenden Ordner wiederherzustellen.

5.12 Leere Ordner und DB-Dateien aussortieren

Aufgabenbereich:	Bewertung
Eingesetzte Skripte:	ordner_loesche_leere.py, dateien_liste.py, dateien_kassieren.py
Inputfiles:	kassiere_db_dateien.csv
Tools:	–

Bisher unberücksichtigt geblieben waren auch leere Verzeichnisse innerhalb der Sammlung. Im Rahmen der finalen Aufbereitung vor dem Ingest sollten diese nun entfernt werden. Hierzu wurde das Skript „ordner_loesche_leere.py“ aufgerufen. Dieses überprüft alle Verzeichnisse unterhalb eines angegebenen Stammpfades (hier: F:\Aipp) und löscht diese sofern sie leer sind, also keine Dateien enthalten. Da leere Verzeichnisse somit auch keinerlei Information enthalten, die vor dem Ingest im Rahmen einer Nachbewertung womöglich doch noch als erhaltenswert erscheinen könnte, werden die Ordner durch das Skript nicht in „kassierte_dateien“ verschoben, sondern genau wie die vorab als kassabel bewerteten Verzeichnisse gelöscht – das Vorgehen wird wie gewohnt geloggt.

123 Diese Auswahl nahm Frau Brühl am Staatsarchiv Sigmaringen vor.

Ebenso wurden nun die in Schritt 5.8 nicht kassierten DB-Dateien aus der Auswahl entfernt, indem zunächst mit dem Skript „dateien_liste.py“ eine entsprechende Auflistung erstellt wurde, die dann mit Aufruf von „dateien_kassieren.py“ unmittelbar verarbeitet werden konnte. Bei einem erneuten Durchlaufen des Workflows wären die genannten Schritte – Löschen leerer Verzeichnisse und Aussortieren sämtlicher nicht archivwürdiger Dateiformate – sinnvollerweise auf die finale Bereinigung zu vereinen.[124]

5.13 Intellektuelle Begutachtung

Der Workflow sieht grundsätzlich eine intellektuelle Nachbewertung vor. Mit der Aufteilung in eine Auswahl und den davon getrennten Bereich der vorläufig kassierten Dateien wird die Möglichkeit geschaffen, nachträglich Korrekturen an der Auswahl vorzunehmen. Da im Rahmen der automatischen Bewertung keine Bilder gelöscht werden, bleiben alle vorgenommenen Änderungen jederzeit mit den zur Verfügung stehenden Skripten – aber natürlich auch händisch – reversibel, ohne dabei auf eine zweite vollständige Arbeitskopie der Sammlung zurückgreifen zu müssen. Vor allem beim Umgang mit Bildern ermöglicht die vorgenommene Aufteilung so auch den unmittelbaren Vergleich der jeweils ausgewählten und der kassierten Dateien eines Verzeichnisses, beispielsweise über die Vorschauansichten in zwei parallel geöffneten Windows Explorer-Fenstern. In Abhängigkeit der zur Verfügung stehenden Ressourcen ist ein Verzicht, eine eng begrenzte oder auch umfassendere Nachbewertung vorstellbar.

Zur Begutachtung des automatisierten Bewertungsergebnisses wurde die Sammlung schließlich per SFTP-Transfer auf einen Server des Landesarchivs Baden-Württemberg übertragen. Nachdem entschieden wurde, dass vor dem Ingest keine Änderungen an der bewerteten Auswahl vorgenommen werden sollten, konnte dieser am Landesarchiv Baden-Württemberg, Abt. Staatsarchiv Ludwigsburg vorgenommen werden.

5.14 Ergebnisse der automatisierten Aufbereitung

Nachdem die Workflow-Konzeptionierung und sämtliche Tests abgeschlossen waren, konnte die Sammlung Aipperspach binnen weniger Tage von 211.714 Dateien in 1.664 Verzeichnissen und einem Datenvolumen von 451 Gigabyte auf nur mehr 34.136 JPG-Dateien in immer noch 1.333 Verzeichnissen und eine Gesamtgröße von 71 Gigabyte reduziert werden. Zusätzlich wurden 2.041 Zeitungsausschnitte separiert.

124 Erst während des Ingests stellte sich heraus, dass auch hier noch eine dezidierte Filterung nach Dateitypen vorgenommen werden kann. Somit hätte das Löschen dieser Dateitypen freilich auch ganz entfallen können. Das Löschen leerer Verzeichnisse wäre dann theoretisch auch an den Beginn zu stellen. Unabhängig davon, zeigt das hier beschriebene Vorgehen aber, wie – zum Beispiel bei der Bearbeitung komplexerer Sammlungen aus Gründen der gesteigerten Übersichtlichkeit – bestimmte Dateitypen ohne den Einsatz systemgebundener Werkzeuge systematisch verschoben oder auch gelöscht werden können.

Ohne eine explizite Deduplizierung vorzunehmen, wurde auch der Anteil der Duplikate von 22 auf rund 10 % herabgesetzt und somit mehr als halbiert – bei weitgehender Erhaltung der vorhandenen Kontextinformationen. Weitgehend vor allem auch deshalb, weil in der zur Verfügung stehenden Zeit keine Lösung für solche Fotografien entwickelt werden konnte, in die Aipperspach kleinere erläuternde Inschriften zu Personen oder Orten eingeschrieben hatte und die daher ebenfalls nach Möglichkeit erhalten werden sollten. Zwar wurde an dem Problem gearbeitet, denn eine entsprechend genaue Motiv- bzw. Texterkennung wäre prinzipiell auch mit Python zu realisieren, allerdings konnte dies zeitbedingt an einem gewissen Punkt nicht weitergeführt werden.

Auch der Anteil ähnlicher Bilder, vor allem Quasi-Dubletten, wurde durch die Zufallsselektion signifikant verkleinert, da sich ähnliche Bilder innerhalb der einzelnen Serien naturgemäß hintereinander befanden und bei der Auswahl jeweils sechs Bilder übersprungen wurden. Freilich fand bei dieser Art der Selektion keine qualitative Beurteilung statt, welches Bild einer solchen Teilserie denn nun das Erhaltenswerteste sei. Ebenso wenig wurde berücksichtigt, dass die Überlieferung mehrerer ähnlicher Ansichten ein und derselben Szene in mancherlei Hinsicht auch sinnvoll erscheinen kann. An solchen Punkten treten die Nachteile einer zufallsbasierten Auswahl zutage. Entsprechend wurde bei der Begutachtung hinterher festgestellt, dass eine intellektuelle Bewertung zum Teil eine andere Auswahl bedeutet hätte. So wären beispielsweise unterschiedliche Perspektiven eines Gebäudes gewählt worden, wo durch die Zufallsauswahl lediglich eine Ansicht erhalten geblieben war. Im Falle eines lokalgeschichtlich bedeutenden Familiengrabeswurde festgestellt, dass der Zufallsgenerator ebendiese Bilder aus dem Friedhofsrundgang entfernt, weniger bedeutende Grabaufnahmen hingegen ausgewählt hatte. In solchen Fällen bleibt eine intellektuelle Bewertung erforderlich.

Daher sei hier noch einmal der Sandwich-Charakter des entwickelten Vorgehens hervorgehoben. Die Programmierung des Zufallsgenerators erlaubt es, vorab als erhaltenswert bewertete Bilder von der Selektion auszunehmen. Und auch hinterher ist ein Rückführen von Dateien möglich. Die Automatisierung ersetzt nicht die intellektuelle Bewertung, sie sollte sie aber in dem Maß ergänzen, das angesichts der Größe der Fileablage notwendig ist, um eine Bearbeitung überhaupt vornehmen zu können. Auch im Sinne der zu bewahrenden Authentizität digitaler Sammlungen sollte außerdem berücksichtigte werden, dass sämtliche zum Ergebnis der automatisierten Aufbereitung beigetragenen Prozesse – die Zufallsselektion eingeschlossen – transparent sind und nachvollziehbar bleiben. Dies gewährleisten die Skripttexte, wie auch der während der Bearbeitung erstellte dokumentarische Aufbereitungsbericht, vor allem aber das stringente Logging sämtlicher Auswahl- und Änderungsoperationen.

6. Ingest nach DIMAG und ScopeArchiv

Die aufbereitete Fotosammlung und die Sammlung der Zeitungsausschnitte wurden abschließend in das Digitale Archivsystem des Landesarchivs Baden-Württemberg DIMAG eingestellt. Dabei wurden gemäß der OAIS-konformen Archivierung die beiden Aufgabenbereiche Archival Storage und Data Management berührt. Zum einen mussten aus den aufbereiteten SIPs für das Digitale Archivsystem – Archival Storage – AIPs formiert werden. Zudem waren Erschließungsmetadaten für das Archivinformationssystem ScopeArchiv – Teil des Data Managements – zu erzeugen und anschließend mit dem ScopeArchiv Übernahmeassistenten in dieses zu transferieren.

Der Ingest wurde mit dem DIMAG-IngestTool ausgeführt. Es wird hier das Vorgehen für die Fotosammlung beschrieben, die Einstellung der Schriftstücke erfolgte weitgehend analog hierzu. Die Paketierung der AIPs erfolgte anhand der vorgegebenen, zum Teil im Rahmen der Aufbereitung zu diesem Zweck modifizierten, Ordnerstruktur. Der Fokus wurde dementsprechend auf die Ebene der Hauptordner (F:\Aipp\Ereignis) gelegt. Am Ende des Workflows wurden die vorhandenen DB-Dateien zwar gelöscht, sicherheitshalber wurden aber

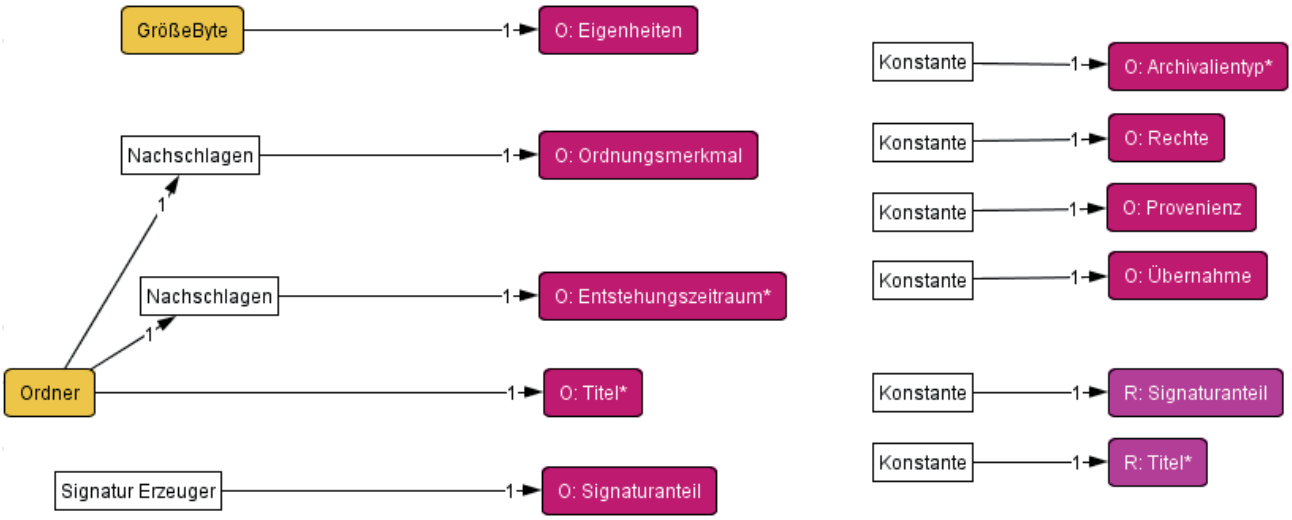

Abb. 11: Mapping in IngestTool für DIMAG und das Archivinformationssystem ScopeArchiv (vgl. Tabelle 2).

auch in IngestTool noch einmal DB-Dateien herausgefiltert, um definitiv auszuschließen, dass solche Dateien Eingang in die formierten AIPs finden konnten. Denn möglicherweise waren bei der Begutachtung nach dem SFTP-Transfer erneut solche Dateien angelegt worden.
Im zweiten Schritt erfolgte das Mapping.[125] Zur Generierung von „Dateimetadaten pro AIP“ wurden zwei Quellelemente angelegt. Als Quelle für die Metadatenerzeugung sollten in erster Linie die im Rahmen des Workflows aufbereiteten Ordnernamen dienen, weshalb das Element „Ordner“ angelegt wurde. Da in ScopeArchiv nicht

125 Vgl. Abb. 11. Bei den magentafarbenen Elementen steht „O“ für „Object“ = Intellectual entity; in den violetten Feldern steht „R“ für „Representation“. Für die Ebene F = „File“ fand kein Mapping statt.

unmittelbar die Anzahl an Dateien pro AIP als Metadatum hinterlegt werden konnten, wurde zusätzlich das Quellelement „GrößeByte“ erstellt. Mit der so erfassten Größe des AIPs und der ermittelten durchschnittlichen Dateigröße von 2.040.000 Byte war es daher vor der Übertragung an ScopeArchiv immerhin möglich, einen Näherungswert für die Dateianzahl zu berechnen und dem Feld „Umfang“ zuzuordnen, sodass sich künftig der Archivar oder Nutzer über das Findmittel eine Vorstellung von der Bildanzahl der jeweiligen Einheit verschaffen kann. Das Quellelement „Ordner“ lieferte unmittelbar die Titeldaten für „O: Titel“. Über Lookup-Tabellen wurden zudem die Erschließungsfelder „O: Ordnungsmerkmal“ und „O: Entstehungszeitraum bedient. Zur Erzeugung der Lookup-Tabellen mussten lediglich mithilfe Regulärer Ausdrücke die Ordnungsnummern sowie die Datumsangaben von den Ordnerbezeichnungen abgeschnitten und in zwei Tabellen jeweils in separate Spalten neben die Ordnerbezeichnungen gesetzt werden. Die Datumsangaben wurden auf diesem Weg zudem in die einheitliche Form MM.DD.YYYY gebracht. Mittels des in IngestTool implementierten Signaturerzeugers wurde eine fortlaufende Signatur erzeugt, die zusammen mit der Bestandssignatur eine eindeutige Bestellsignatur bildet. Die restlichen Felder wurden durch Konstanten bedient. Diese und die genaue Zuordnung der Mapping-Läufe für ScopeArchiv und DIMAG können Abbildung 12 entnommen werden.

Quelle	Mapping-Element (Abb. 11)	Ziel in ScopeArchiv	Ziel in DIMAG
Quellelement „GrößeByte“ (Quotient aus Ordnergröße in Bytes und durchschnittlicher Dateigröße	O: Eigenheiten	Umfang	–
Aus Ordnernamen erzeugte Nachschlagetabelle	O: Ordnungsmerkmal	Vorsignatur	–
Aus Ordnernamen erzeugte Nachschlagetabelle	O: Entstehungszeitraum	Entstehungszeitraum	Entstehungszeitraum
Quellelement „Ordnername“	O: Titel	Titel	Titel
Automatisch hochgezählte Ganzzahl plus Präfix für Bestandssignatur	O: Signaturanteil	Bestellsignatur	Signaturanteil
Konstante „Foto“	O: Archivalientyp	Erschließungsformular	Archivalientyp
Konstante „Die Verwertungsrechte an den Fotos von Gunter Aipperspach wurden dem Landesarchiv Baden-Württemberg übertragen. Der Rechtsstatus eventuell vorhandener Zeitungsausschnitte und Fotos dritter Personen ist ungeklärt.“	O: Rechte	Rechteinhaber	Rechte
Konstante „Gunter Aippers-pach“	O: Provenienz	Autor/Fotograf/Künstler	Provenienz

Quelle	Mapping-Element (Abb. 11)	Ziel in ScopeArchiv	Ziel in DIMAG
Konstante „Im Jahr 2017 übernommen von Sibylle Brühl.“	O: Übernahme	–	Übernahme
Konstante „R 1“	R: Signaturanteil	–	R-Signaturanteil
Konstante „Zugang (JPEG)“	R: Titel	–	R-Titel

Abb. 12: Zuordnung des Ingest-Mappings nach DIMAG und ScopeArchiv (vgl. Abb. 11).

Die digitale Sammlung Aipperspach wird nun – neben N 1/101 T 1, dem zuvor bereits angelegten Teilbestand aus Diapositiven und Super8-Filmen Aipperspachs – als N1/101 T 2 beim Staatsarchiv Sigmaringen geführt. Die Abbildungen 13 und 14 zeigen exemplarisch das Ergebnis der hier beschriebenen Erschließung in ScopeArchiv sowie die Verortung des Bestands in DIMAG.

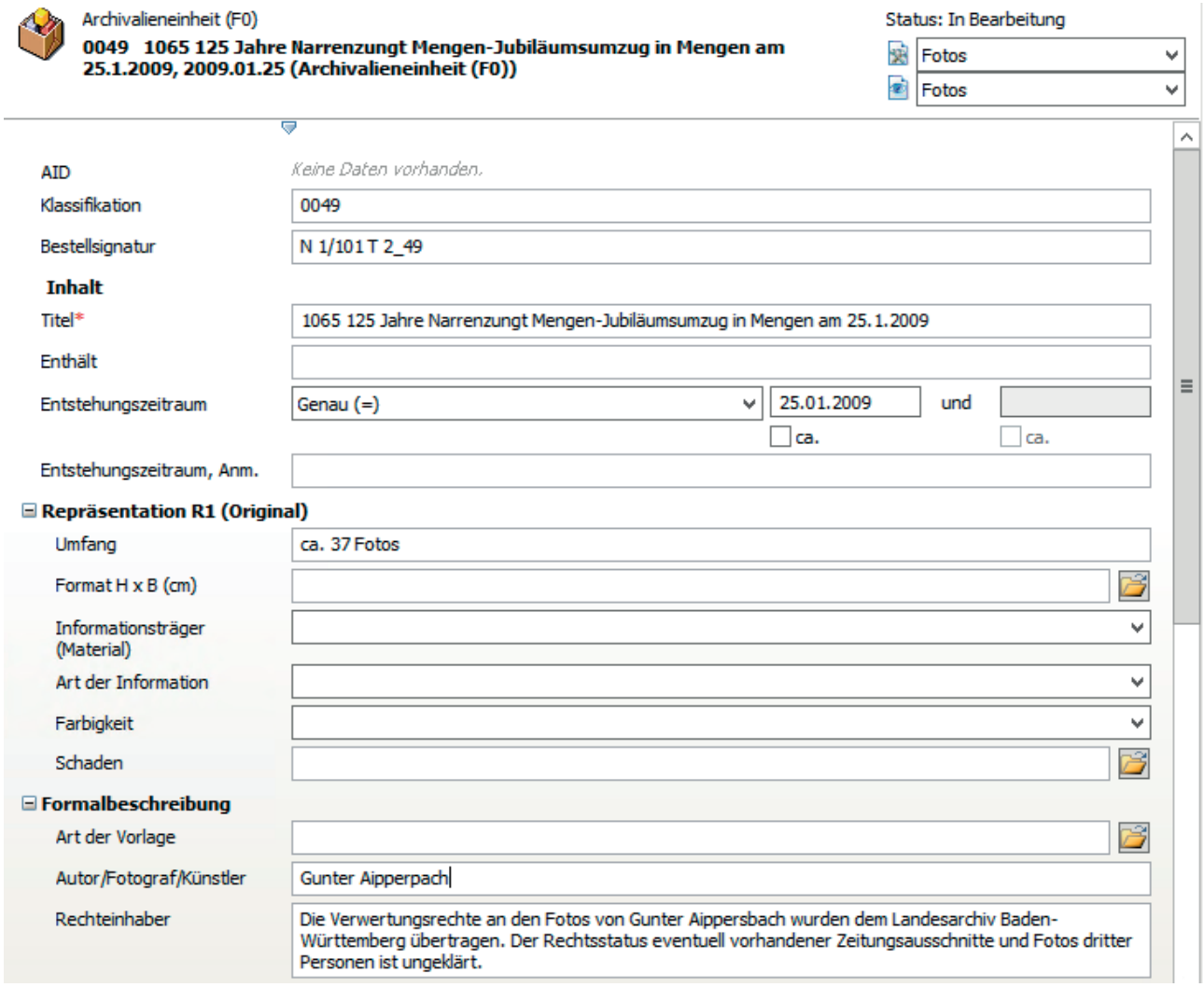

Abb. 13: Ein Beispiel für die fertige Erschließung in ScopeArchiv.

Tektonik:
- D: Digitales Archiv
  - A: Archivalien
    - StAS: Staatsarchiv Sigmaringen (Klassifikation = 6)
      - : Nachlässe, Partei-, Vereins- und...
        - : Nachlässe
          - : Fotoarchiv Gunter Aipperspach (geb.1940)
            - : Fotoarchiv Gunter Aipperspach (geb.1940):...
              - : Fotos

Suche aID:
go
Modus wechseln

Signatur ?	Titel ?	Eigenschaften ?
N 1/101 T 2_1	100 Einschulung der Grundschüler in Laiz (aus Unterschmeien Robin Stauß,...	
N 1/101 T 2_2	1002 Volkstrauertag in Oberschmeien 16.11.2008	
N 1/101 T 2_3	1003 Volkstrauertag auf d. Hedinger Friedhof Sigmaringen mit Bürgermeister Dr....	
N 1/101 T 2_4	1004 Umgestaltung Stadteingang West Sigmaringen beim Cafe Schön 9.6. - 21.11...	
N 1/101 T 2_5	1005a Die Josefskapelle in Sigmaringen im Schnee 22.11.2008	
N 1/101 T 2_6	1005d Harald Kaut beim Schneeräumen in Unterschmeien-Äckerle 22.11.2008	
N 1/101 T 2_7	1006 Bilder von Sigmaringen im Schnee 22.-23.11.2008	
N 1/101 T 2_8	1008 Vorbereitungen in der Innenstadt Sigmaringen auf die Weihnachtszeit 25.-28...	
N 1/101 T 2_9	1016 VfB Stuttgart-FC Schalke 04 2-0 in Stuttgart 30.11.2008	
N 1/101 T 2_10	1018 Aufnahme der neuen Ministranten in St. Johann Sigmaringen mit Vikar...	
N 1/101 T 2_11	1019 Gruppenbilder der Ministranten von St. Johann Sigmaringen 7.12.2008	
N 1/101 T 2_12	1021 Winterliches Unterschmeien nach starkem Schneefall 10.12.2008	
N 1/101 T 2_13	1022 Winterliches Sigmaringen nach starkem Schneefall 11.12.2008	
N 1/101 T 2_14	1024a Weihnachtlich beleuchtetes Sigmaringen in der Innenstadt 12.12.2008	
N 1/101 T 2_15	1025b Sigmaringer Adventskalender 2008 Öffnen d.12.Fensters d.GS Laiz -kleine...	
N 1/101 T 2_16	1025c Sigmaringer Adventskalender-alle Fenster offen- 7.1.2009	
N 1/101 T 2_17	1027 Sigmaringen on Ice Offizielle Eröffnung durch Bürgermeister Dr. Rapp...	
N 1/101 T 2_18	1027b Sigmaringen on Ice 17.12.2008	
N 1/101 T 2_19	1027c Sigmaringen on Ice 22.12.2008	
N 1/101 T 2_20	1027d Sigmaringen on Ice 26.12.2008 Fam. Aipperspach, Stauß, Frick	
N 1/101 T 2_21	1027e Sigmaringen on Ice 3.1.2009 Fam. Frick	

Abb. 14: Tektonik und einzelne Bestelleinheiten in DIMAG.

7. Schlussbemerkungen

Der am Ende der Bearbeitungszeit erfolgreich ausgeführte Workflow zur Aufbereitung der digitalen Fotosammlung Gunter Aipperspach ging aus der Gelegenheit hervor, über mehrere Wochen hinweg, schrittweise verschiedene Bearbeitungswege erschließen und diese gleich in der Praxis erproben zu können. Einige der eingeschlagenen Wege erwiesen sich nicht als zielführend, andere konnten in der zur Verfügung stehenden Zeit wiederum nicht zu Ende gegangen werden. Nur die erfolgreich beschrittenen Wege haben letztlich Eingang in den Workflow und damit auch in die vorliegende Arbeit gefunden. Mit der Hinwendung zu Python wurde ein Weg gewählt, der von vorneherein gleich in mehrfacher Hinsicht reizvoll erschien und ein großes Maß an Unabhängigkeit und Flexibilität erwarten ließ.

Keinesfalls soll der Eindruck erweckt werden, mit dem hier gewählten Lösungsansatz sei eine Art Königsweg beschritten worden. Neben der eigentlichen Arbeit an den Skripten erfordert die durchgängig textbasierte Arbeitsweise mit verschiedenen parallel angewendeten Skripten und CSV-Dokumenten ihrerseits ebenfalls ein gewisses Maß an Routine, um die durch den Skripteinsatz potenziell möglichen Effizienzgewinne bestmöglich auszuschöpfen. Vor allem aber konnten die Möglichkeiten der Computer Vision, also jener Techniken, die für die Arbeit mit Bildern zum Einsatz kommen, nur angedeutet werden. Insbesondere bleibt diesbezüglich auf die potenziellen Möglichkeiten im Bereich Motiverkennung zu verweisen. Mithilfe von maschinellem Lernen ließen sich weitaus genauere Suchmechanismen entwickeln, als dies hier realisiert werden konnte. Konsequent weitergedacht wäre eine ausgereifte Motiverkennung dann nicht nur für die Bewertung, sondern auch für eine automatisierte Erschließung der Sammlung auf Einzelbildebene einsetzbar. Bereits seit einiger Zeit bieten sich in diesem Bereich schon einige Service-Dienste an, die hinsichtlich ihrer Vertrauenswürdigkeit und der gebotenen Funktionalitäten einmal genauer zu begutachten wären.[126] Im Idealfall ließe sich das randomisierte Samplingverfahren, das hier mit Mitteln der Bilderkennung lediglich um die bewusste Selektion / Kassation einzelner Bildgruppen ergänzt wurde, mit dem Einsatz intelligenterer maschineller Verfahren gänzlich ersetzen. Die Ermittlung unscharfer Fotografien konnte einerseits zwar erfolgreich umgesetzt werden, letztlich ließ sich jedoch auf Basis dieses Kriteriums keine signifikante Zahl an Fotografien kassieren. Doch auch an dieser Stelle zeigt das Vorgehen, wie mittels einer selbst programmierten Automatisierung große Dateimengen effizient auf bestimmte Merkmale hin analysiert werden können.

Die im Rahmen der Arbeit eingesetzten Skripte sind insgesamt nicht als fertige Suite zu verstehen. Viel mehr steht mit ihnen eine Sammlung – sicherlich an vielen Stellen optimierbarer und ausbaufähiger – Werkzeuge zur Verfügung. Natürlich hätten nicht alle Bearbeitungsschritte im Einzelnen in Python ausgeführt werden müssen, wie es hier getan wurde, auch um die systemunabhängige Anwendbarkeit und Nachnutzbarkeit der Skripte im Sinne des exemplarischen Vorgehens durchgehend zu wahren. Es sei hier nur auf die Kommandozeileninterpreter cmd.exe oder PowerShell verwiesen, mit denen sich in Windows einzelne Arbeitsschritte auf ähnliche Weise umsetzen ließen. Hier gilt es weiterhin, in der täglichen Praxis herauszufinden, mit welchen Mitteln einzelne programmierbare Aufgaben am effizientesten umgesetzt werden können. Unabhängig davon manifestierte sich im Laufe der Arbeit jedoch die Erkenntnis, dass sich das für den Einsatz

126 Verwiesen sei daher hier noch auf: Hindle, Automated image analysis with IIIF.

einfacher Skripte notwendige Rüstzeug mit überschaubarem Aufwand aneignen lässt. Auch wird es mit großer Sicherheit bei der Bearbeitung von neuen Dateisammlungen immer wieder neu anwendbar und erweiterbar sein. Zusammengenommen vermag dies vielleicht Anreiz für den archivischen Einsatz ebensolcher digitalen Werkzeuge zu sein, wenn sich vonseiten der Archivare und Archivarinnen an diese nicht herangewagt wird, weil sie grundlegende Programmierkenntnisse erfordern.

Die gesamte Bearbeitung der Fotosammlung hat zudem gezeigt, wie im Umgang mit Dateisammlungen die archivarischen Aufgabenbereiche der Bewertung und Erschließung zusammenrücken. Auch hat sich hier die Kombination verschiedener Bewertungsansätze bewährt, wie sie beispielsweise von Victoria Sloyan beschrieben wurde. Für die erfolgreiche Aufbereitung digitaler Sammlungen erscheint also insgesamt eine Kombination verschiedener Ansätze, aber vor allem auch technischer Hilfsmittel sinnvoll. Das Einbeziehen von Python könnte dabei in vielen Fällen eine nützliche Option sein.

Anhang

Literaturverzeichnis

Arbeitskreis „Archivierung von Unterlagen aus digitalen Systemen“. https://www.sg.ch/kultur/staatsarchiv/Spezialthemen-/auds.html (aufgerufen am 14.04.2020).

Bayer, Peter R.: Bytebarn – Dateisammlungen unter Dach und Fach. In: Sächsisches Archivblatt H. 2 (2016) S. 15–17. https://publikationen.sachsen.de/bdb/artikel/27188/documents/38406 (aufgerufen am 13.10.2019).

Benjamin, Walter: Gesammelte Schriften III, Kritiken und Rezensionen. Hg. von Hella *Tiedemann-Bartels* (Suhrkamp-Taschenbuch Wissenschaft 933). Frankfurt am Main 2006.

Belovari, Susanne: Rasche und einfache Bearbeitung von Dateisammlungen: ein MPLP-Ansatz. In: Kai *Naumann* und Michael *Puchta* (Hg.): Kreative digitale Ablagen und die Archive. Ergebnisse eines Workshops des KLA-Ausschusses Digitale Archive am 22./23. November 2016 in der Generaldirektion der Staatlichen Archive Bayerns (Sonderveröffentlichungen der Staatlichen Archive Bayerns 13). München 2017. S. 17–29. https://www.bundesarchiv.de/DE/Content/Downloads/KLA/tagungsdokumentation-kreative-digitale-ablagen.pdf (aufgerufen am 13.10.2019).

Berish, Amy: Learning Python as a Processing Archivist: A Reflection. Blogpost 2016. http://blog.rockarch.org/?p=1483 (aufgerufen am 13.10.2019).

Bischoff, Frank M.: Bewertung elektronischer Unterlagen und die Auswirkungen archivarischer Eingriffe auf die Typologie zukünftiger Quellen. In: Archivar 67 (2014/1) S. 40–52. http://www.archive.nrw.de/archivar/hefte/2014/ausgabe1/Archivar_Internet_2014_1_neu.pdf (aufgerufen am 13.10.2019).

Branca, Fabrizio: Lightroom plugin: Detect blurry images. Blogpost 2011. http://fbrnc.net/blog/2011/07/lightroom-plugin-detect-blurry-images (aufgerufen am 13.10.2019).

Caraffa, Constanza: „Wenden!“ Fotografien in Archiven im Zeitalter ihrer Digitalisierbarkeit: ein material turn. In: Rundbrief Fotografie 18 (2011/1) S. 8–15.

Charbonneau, Normand: The Selection of Photographs. In: Archivaria 59 (2005) S. 119–138. http://www.archivaria.ca/index.php/archivaria/article/download/12504/13628 (aufgerufen am 13.10.2019).

Cook, Terry: “Many are called, but few are chosen”: Appraisal Guidelines for Sampling and Selecting Case Files. In: Archivaria 32 (1991) S. 25–50. http://archivaria.ca/index.php/archivaria/article/viewFile/11759/12709 (aufgerufen am 13.10.2019).

dupeGuru – finds duplicate files. https://dupeguru.voltaicideas.net/ (aufgerufen am 13.10.2019).

Enge, Jürgen und Heinz Werner *Kramski*: „Arme Nachlassverwalter…" Herausforderungen, Erkenntnisse und Lösungsansätze bei der Aufbereitung komplexer digitaler Dateisammlungen. In: Jörg *Filthaut* (Hg.): Von der Übernahme zur Benutzung. Aktuelle Entwicklungen in der digitalen Archivierung. 18. Tagung des Arbeitskreises „Archivierung von Unterlagen aus digitalen Systemen" am 11. und 12. März 2014 in Weimar (Schriften des Thüringischen Hauptstaatsarchivs Weimar 6). Weimar 2015. S. 53–62. https://www.researchgate.net/publication/273203413_Arme_Nachlassverwalter_Herausforderungen_Erkenntnisse_und_Losungsansatze_bei_der_Aufbereitung_komplexer_digitaler_Datensammlunge (aufgerufen am 13.10.2019).

England, Elizabeth und Eric *Hanson*: Automating Digital Archival Processing at Johns Hopkins University. Blogpost 2017. https://blogs.loc.gov/thesignal/2017/05/ (aufgerufen am 13.10.2017).

Ernst, Wolfgang: „Jenseits der Verschlagwortung? Plädoyer für ein nicht-textbasiertes Bildgedächtnis". Vortrag im Museum des Westfälischen Kunstvereins Münster aus Anlaß einer Ausstellung mit Werken von Olaf Nicolai. Münster 2007. https://www.musikundmedien.hu-berlin.de/de/medienwissenschaft/medientheorien/texte-zur-medienarchaeologie/bildnic-net.pdf (aufgerufen am 13.10.2019).

Ernst, Wolfgang: Wohldefinierte (Bild-)Archive – und was sie nicht sind. Vortrag auf der Tagung Depot und Plattform: Bildarchive im post-fotografischen Zeitalter der Deutschen Gesellschaft für Photographie. Köln 2009. https://www.musikundmedien.hu-berlin.de/de/medienwissenschaft/medientheorien/texte-zur-medienarchaeologie/phot-arc-koeln-kurz.pdf (aufgerufen am 13.10.2017).

Feldt, Christine: Herausforderungen im Umgang mit digitalen und analogen Fotografien im kommunalen Archivwesen. Masterarbeit im Weiterbildungs-Masterstudiengang Archivwissenschaften an der Fachhochschule Potsdam. 2014. https://opus4.kobv.de/opus4-fhpotsdam/files/911/Masterarbeit_C.FELD.pdf (aufgerufen am 13.10.2019).

Fischer, Ulrich: Gemeinsame Lösungen für ein gemeinsames Problem. Verbundlösungen für die elektronische Langzeitarchivierung in Deutschland. In: Archivpflege in Westfallen-Lippe 80 (2014) S. 20–24. https://f.hypotheses.org/wp-content/blogs.dir/1762/files/2014/04/archivpflege-heft-80-01-2014-fuer-webseite.pdf (aufgerufen am 13.10.2019).

Gilliland, Anne J.: Archival appraisal: practising on shifting sands. In: C. *Brown* (Hg.): Archives and recordkeeping. Theory into practice. London 2014. S. 31–62.

Greene, Mark und Dennis *Meissner*: More Product, Less Process: Revamping Traditional Archival Processing. In: The American Archivist 68 (2005) S. 208–263. http://www.archivists.org/prof-education/pre-readings/IMPLP/AA68.2.MeissnerGreene.pdf (aufgerufen am 13.10.2019).

Günzel, Stephan und Dieter *Mersch*: Bild. Ein interdisziplinäres Handbuch. Stuttgart 2014.

Harvey, Ross: DCC Digital Curation Manual. Instalment on Appraisal and Selection. http://www.dcc.ac.uk/resources/curation-reference-manual/completed-chapters/appraisal-and-selection (aufgerufen am 13.10.2019).

Hering, Rainer: Ohnmächtig vor Bits and Bytes? Archivische Aufgaben im Zeitalter der Digitalisierung. In: Rainer *Hering* und Dietmar *Schenk* (Hg.): Wie mächtig sind Archive? Perspektiven der Archivwissenschaft (Veröffentlichungen des Landesarchivs Schleswig-Holstein 104). Hamburg 2013. S. 83–97. http://hup.sub.uni-hamburg.de/volltexte/2013/133/pdf/HamburgUP_LASH104_HeringSchenk_Archive.pdf (aufgerufen am 13.10.2019).

Hindle, Adrian: Automated image analysis with IIIF. Blogpost 2017. https://blog.cogapp.com/automated-image-analysis-with-iiif-6594ff5b2b32 (aufgerufen am 13.10.2019).

Kahlenberg, Friedrich P. und Heiner *Schmitt*: Zur archivischen Bewertung von Film- und Fernsehproduktionen. Ein Diskussionsbeitrag. In: Der Archivar 34 (1981/2) Sp. 233–242.

Keitel, Christian: Benutzerinteressen annehmen und signifikante Eigenschaften festlegen. Einige neue Aufgaben für Archivare. In: Heiner *Schmitt* (Hg.): Archive im digitalen Zeitalter. Überlieferung – Erschließung – Präsentation (Tagungsdokumentationen zum Deutschen Archivtag 14). Fulda 2010. S. 29–42.

Keitel, Christian: Der nestor-Leitfaden zur Digitalen Bestandserhaltung und seine Folgen für die Archive. In: Christian *Keitel* und Kai *Naumann* (Hg.): Digitale Archivierung in der Praxis (Werkhefte der Staatlichen Archivverwaltung Baden-Württemberg Serie A Heft 24). Stuttgart 2013. S 267–277. https://www.landesarchiv-bw.de/web/59083 (aufgerufen am 13.10.2019).

Keitel, Christian, Rolf *Lang* und Kai *Naumann*: Konzeption und Aufbau eines digitalen Archivs: Von der Skizze zum Prototypen. In: Katharina *Ernst* (Hg.): Erfahrungen mit der Übernahme digitaler Daten. Bewertung, Übernahme, Aufbereitung, Speicherung, Datenmanagement. 11. Tagung des Arbeitskreises „Archivierung von Unterlagen aus digitalen Systemen“ vom 20./21. März 2007 (Veröffentlichungen des Archivs der Stadt Stuttgart 99). Stuttgart 2007. S. 36–41. https://www.landesarchiv-bw.de/sixcms/media.php/120/42600/aufsatz_labw_aufbau.pdf (aufgerufen am 13.10.2019).

Kemper, Joachim und Kai *Naumann*: Selbermachen! Praktische Tipps zur Archivierung digitaler Unterlagen, Digitalisierung und Öffentlichkeitsarbeit im Netz. In: Kai *Naumann* und Peter *Müller* (Hg.): Das neue Handwerk – Digitales Arbeiten in kleinen und mittleren Archiven. Stuttgart 2013. S. 85–92. https://www.landesarchiv-bw.de/sixcms/media.php/120/59735/LETZTE_Das%20Handwerk%20Inh%2096S.pdf (aufgerufen am 13.10.2019).

Koch, Elke u. a.: Bewertungsautomat statt Autopsie: Sind jetzt zehntausend Akten in zehn Sekunden bewertet? In: Archivar 70 (2017/2) S. 173–177. http://www.archive.nrw.de/archivar/hefte/2017/Ausgabe-2/Archivar-2_2017.pdf (aufgerufen am 10.09.2017).
Koordinierungsstelle für die dauerhafte Archivierung elektronischer Unterlagen: Katalog archivischer Dateiformate: JPEG. http://kost-ceco.ch/wiki/whelp/KaD/index.php?ld=http://kost-ceco.ch/wiki/whelp/KaD/pages/JPEG.html (aufgerufen am 13.10.2019).

Kretzschmar, Robert: Aussonderung und Bewertung von sogenannten Massenakten Erfahrungen der staatlichen Archivverwaltung Baden-Württemberg. In: Robert *Kretzschmar* (Hg.): Historische Überlieferung aus Verwaltungsunterlagen (Werkhefte der Staatlichen Archivverwaltung Baden-Württemberg 7). Stuttgart 1997. S. 103–118. https://www.landesarchiv-bw.de/sixcms/media.php/120/58039/A%207%20Kretzschmar%20Aussonderung%20Massenakten.pdf (aufgerufen am 13.10.2019).

Kretzschmar, Robert: Die „neue archivische Bewertungsdiskussion" und ihre Fußnoten. Zur Standortbestimmung einer fast zehnjährigen Kontroverse. In: Archivalische Zeitschrift 82 (1999) S. 7–40. http://www.landesarchiv-bw.de/sixcms/media.php/120/56979/Kretzschmar_Archivalische_Zeitschrift_82.pdf (aufgerufen am 13.10.2019).

Kretzschmar, Robert: Positionen des Arbeitskreises Archivische Bewertung im VdA – Verband deutscher Archivarinnen und Archivare zur archivischen Überlieferungsbildung. In: Archivar 58 (2005/2) S. 88–94.

Kretzschmar, Robert: Aktuelle Entwicklungstendenzen des archivarischen Berufsbilds. In: Archivar 63 (2010/4) S. 356–360. http://www.archive.nrw.de/archivar/hefte/2010/ausgabe4/ARCHIVAR_04-10_internet.pdf (aufgerufen am 13.10.2019).

Kretzschmar, Robert: Alles neu zu durchdenken? Archivische Bewertung im digitalen Zeitalter. In: Archivpflege in Westfalen-Lippe 80 (2014) S. 9–15. https://www.lwl.org/waa-download/archivpflege/heft80/Heft_80_2014.pdf (aufgerufen am 13.10.2019).

Kühnel, Karsten: Ein Dokumentationsprofil für ein Universitätsarchiv – Teil 1: Grundlegung.
https://archive20.hypotheses.org/693#_ftnref9 (aufgerufen am 13.10.2019).

Leary, William H.: The Archival appraisal of photographs: a RAMP study with guidelines. Paris 1985. http://unesdoc.unesco.org/images/0006/000637/063749eo.pdf (aufgerufen am 13.10.2019).

Leimgruber, Walter u. a.: Das Fotoerbe sichern. In: Nora *Mathys*, Walter *Leimgrube* und Andrea *Voellmin* (Hg.): Über den Wert der Fotografie. Zu wissenschaftlichen Kriterien für die Bewahrung von Fotosammlungen. Baden 2013. S. 150–153.

LWL-Archivamt für Westfalen, Schaust Du noch oder archivierst Du schon? Fotos und Filme in Archiven. 69. Westfälischer Archivtag. Abstracts. Hamm 2017. https://www.lwl.org/waa-download/tagungen/WAT2017/AT2017_abstracts.pdf (aufgerufen am 13.10.2019).

Mathys, Nora: Welche Fotografien sind erhaltenswert? Ein Diskussionsbeitrag zur Bewertung von Fotografennachlässen. In: Archivar 60 (2007/1) S. 34–40. http://www.archive.nrw.de/archivar/hefte/2007/Archivar_2007-1.pdf (aufgerufen am 13.10.2019).

Mathys, Nora: Das visuelle Erbe. Ein Produkt des Zufalls und der Überlieferungsbildung? In: Nora *Mathys*, Walter *Leimgrube* und Andrea *Voellmin* (Hg.): Über den Wert der Fotografie. Zu wissenschaftlichen Kriterien für die Bewahrung von Fotosammlungen. Baden 2013. S. 91–103.

Memoriav (Hg.): Die Erhaltung von Fotografien. Empfehlungen. Bern 2007. http://memoriav.ch/wp-content/uploads/2014/07/empfehlungen_foto_de.pdf (aufgerufen am 13.10.2019).

Metz, Axel: Nicht jedes Bild sagt mehr als tausend Worte – Ein Beitrag zur Bewertung von Fotobeständen. Transferarbeit im Rahmen der Ausbildung zum höheren Archivdienst an der Archivschule Marburg 2007. https://www.landesarchiv-bw.de/sixcms/media.php/120/42632/Transferarbeit_Metz.pdf (aufgerufen am 13.10.2019).

Metz, Axel: Weniger ist oft mehr – Betrachtungen zur archivischen Bewertung von Fotobeständen. In: Archive in Thüringen. Audiovisuelle Medien in Archiven, Tagungsband. Weimar 2010. S. 4–12. http://zs.thulb.uni-jena.de/servlets/MCRFileNodeServlet/jportal_derivate_00200483/2010_SH_Archive_pdfa1b.pdf (aufgerufen am 13.10.2019).

Metz, Axel: Die archivische Bewertung von Fotobeständen – Ein Remedium gegen die Bilderflut. In: Archivpflege in Westfalen-Lippe 75 (2011) S. 28–32. https://www.lwl.org/waa-download/archivpflege/heft75/Heft_75_2011.pdf (aufgerufen am 13.10.2019).

Mumma, Courtney C., *Glenn* Dingwall und *Sue* Bigelow: A First Look at the Acquisition and Appraisal of the 2010 Olympic and Paralympic Winter Games Fonds: or, SELECT * FROM VANOC_Records AS Archives WHERE Value="true". In: Archivaria 72 (2011) S. 93–122.

Naumann, Kai und Michael *Puchta* (Hg.): Kreative digitale Ablagen und die Archive. Ergebnisse eines Workshops des KLA-Ausschusses Digitale Archive am 22./23. November 2016 in der Generaldirektion der Staatlichen Archive Bayerns (Sonderveröffentlichungen der Staatlichen Archive Bayerns 13). München 2017. https://www.bundesarchiv.de/DE/Content/Downloads/KLA/tagungsdokumentation-kreative-digitale-ablagen.pdf (aufgerufen am 13.10.2019).

Naumann, Kai und Christoph *Schmidt*: Chancen und Risiken des Einsatzes verlustbehafteter Bildkompression in der digitalen Archivierung. Vortragsfolien. Basel 2017. https://www.sg.ch/content/dam/sgch/kultur/staatsarchiv/auds-20181/dakumentationen-camps/08-AUDS%202017%20Chancen%20und%20Risiken_Publikationsversion.pdf (aufgerufen am 13.10.2019).

nestor-Arbeitsgruppe Digitale Bestandserhaltung (Hg.): Leitfaden zur digitalen Bestandserhaltung. Vorgehensmodell und Umsetzung. Version 2.0. (nestor-Materialien 15). Frankfurt am Main 2012. urn:nbn:de:0008-2012092400 (aufgerufen am 13.10.2019).

nestor-Arbeitsgruppe OAIS-Übersetzung / Terminologie (Hg.): Referenzmodell für ein Offenes Archiv-Informations-System. Deutsche Übersetzung 2.0 (nestor-Materialien 16). Frankfurt am Main 2013. urn:nbn:de:0008-2013082706 (aufgerufen am 13.10.2019).

nestor-Arbeitsgruppe Standards für Metadaten (Hg.): Wege ins Archiv – Ein Leitfaden für die Informationsübernahme in das digitale Langzeitarchiv (nestor-Materialien 10). Frankfurt am Main 2008. urn:nbn:de:0008-2008103009 (aufgerufen am 13.10.2019).

Neukom, Thomas: Können wir endlich alles behalten? Archivische Bewertung elektronischer Unterlagen. In: arbido 3 (2016) S. 12–14. http://arbido.ch/assets/files/arbido_2016_3_low_161127_132457.pdf (aufgerufen am 13.10.2019).

Neumayer, Robert und Andreas *Rauber*: Why Appraisal is Not 'Utterly' Useless and Why It's Not the Way to Go Either: A Provocative Position Paper. 2007. http://www.ifs.tuwien.ac.at/~neumayer/pubs/NEU07_appraisal.pdf (aufgerufen am 13.10.2019).

Noble, Richard: Considerations for Evaluationg Local History Photographs. In: Picturescope 31 (1984) S. 17–21.

OpenCV 2.4.13.3 documentation. Image Filtering: Laplacian. http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#laplacian (aufgerufen am 13.10.2019).

OpenCV website. http://opencv.org/ (aufgerufen am 13.10.2019).

Paradigm, Arranging and cataloguing digital and hybrid archives. Chapter 4: Appraisal and disposal. Appraising digital records: a worthwhile exercise? 2008. http://www.paradigm.ac.uk/workbook/appraisal/digital-appraisal.html (aufgerufen am 13.10.2019).

Paul, Gerhard (Hg.): Visual History. Ein Studienbuch. Göttingen 2006.

Pech-Pacheco, J. u. a.: Diatom autofocusing in brightfield microscopy: a comparative study. In: A. *Sanfeliu* u. a. (Hg.): Proceedings 15th International Conference on Pattern Recognition. Volume 3. Image, Speech, and Signal Processing. Los Alamitos 2000. S. 314–317. http://optica.csic.es/papers/icpr2k.pdf (aufgerufen am 13.10.2019).

Pfeiffer, Michel: Visuelle Überlieferungsbildung – Neue Sammlungs- und Bewertungsperspektiven oder nur alter Wein in neuen Schläuchen? In: Irene *Ziehe* und Ulrich *Hägele* (Hg.): Fotografie und Film im Archiv. Sammeln, Bewahren, Erforschen. Münster 2013. S. 129–140.

Pfenninger, Kathryn: Bildarchiv digital (Rundbrief Fotografie. Sonderheft 7). Esslingen 2001.

Pfrunder, Peter: Aufwerten, umwerten, abwerten. Vom Fotoarchiv zum kulturellen Gedächtnis. In: Nora *Mathys*, Walter *Leimgrube* und Andrea *Voellmin* (Hg.): Über den Wert der Fotografie. Zu wissenschaftlichen Kriterien für die Bewahrung von Fotosammlungen. Baden 2013. S. 30–41.

Phillips, Meg: More Product, Less Process for Born-Digital Collections: Reflections on CurateCamp Processing. Gastbeitrag. 2012. https://blogs.loc.gov/thesignal/2012/08/more-product-less-process-for-born-digital-collections-reflections-on-curatecamp-processing/ (aufgerufen am 13.10.2019).

Pilger, Andreas: Ein neues Positionspapier des VdA-Arbeitskreises „Archivische Bewertung" zur Überlieferungsbildung im Verbund. In: Archivar 65 (2012/1) S. 6–11. http://www.archive.nrw.de/archivar/hefte/2012/ausgabe1/Archivar_1_2012.pdf (aufgerufen am 13.10.2019).

Pilger, Andreas: Grundsätze, Methoden und Strategien der Überlieferungsbildung in Archiven. In: Robert *Kretzschmar* und Rainer *Hering* (Hg.): Zeitgeschichte, Archive und Geheimschutz. Beiträge einer Sektion auf dem 49. Deutschen Historikertag 2012 in Mainz. Stuttgart 2013. S. 40–49.

PREMIS Data Dictionary for Preservation Metadata. version 2.1. 2011. http://www.loc.gov/standards/premis/v2/premis-report-2-1.pdf (aufgerufen am 13.10.2019).

Python Extension Packages for Windows by Christoph Gohlke. http://www.lfd.uci.edu/~gohlke/pythonlibs/ (aufgerufen am 13.10.2019).
Reininghaus, Wilfried: Fotografien in der Landes- und Ortsgeschichte Westfalens. In: Westfaelische Forschungen 58 (2008) S. 21–42.

Rosebrock, Adrian: Blur detection with OpenCV – PyImageSearch. Blogpost 2015. http://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/ (aufgerufen am 13.10.2019).

Sagurna, Stephan: Bewertungsfragen. Der Nachlaß des Fotoamateurs Johannes Weber als regionalhistorische Quelle. In: Rundbrief Fotografie 20 (2013/1) S. 11–18.

Schellenberg, Theodore R.: The Appraisal of Modern Public Records. In: Bulletins of the National Archives 8 (1956) S. 223–278. https://www.archives.gov/research/alic/reference/archives-resources/appraisal-of-records.html (aufgerufen am 13.10.2019).

Schludi, Ulrich: Zwischen Records Management und digitaler Archivierung. Das Dateisystem als Basis von Schriftgutverwaltung und Überlieferungsbildung. In: Kai *Naumann* und Peter *Müller* (Hg.): Das neue Handwerk – Digitales Arbeiten in kleinen und mittleren Archiven. Stuttgart 2013. S. 20–38. https://www.landesarchiv-bw.de/sixcms/media.php/120/59735/LETZTE_Das%20Handwerk%20Inh%2096S.pdf (aufgerufen am 13.10.2019).

Schmidt, Christoph: Signifikante Eigenschaften und ihre Bedeutung für die Bewertung elektronischer Unterlagen. In: Katharina *Tiemann* (Hg.): Bewertung und Übernahme elektronischer Unterlagen – Business as usual? Beiträge des Expertenworkshops in Münster am 11. und 12. Juni 2013 (Texte und Untersuchungen zur Archivpflege 28). Münster 2013. S. 20–29.

Shallcross, Michael: Appraising Digital Archives with Archivematica. 2016. http://dx.doi.org/10.1109/BigData.2016.7840985 (aufgerufen am 13.10.2019).

Shallcross, Michael: Bentley Historical Library Guidelines for the Manual Processing of Born-Digital Materials. 2011. http://hdl.handle.net/2027.42/96439 (aufgerufen am 13.10.2017).

Shallcross, Michael: The End is Just a New Beginning! Blogpost 2016. http://archival-integration.blogspot.de/2016/11/the-end-is-just-new-beginning.html (aufgerufen am 13.10.2017).

Shallcross, Michael und Nancy *Deromedi*: Automated Digital Processing at the Bentley Historical Library. 2012. https://deepblue.lib.umich.edu/bitstream/handle/2027.42/95923/BHL_iPRESpaper_20120828.pdf (aufgerufen am 13.10.2019).

Sloyan, Victoria: Born-digital archives at the Wellcome Library. Appraisal and sensitivity review of two hard drives. In: Archives and Records 37 (2016/1) S. 20–36. http://dx.doi.org/10.1080/23257962.2016.1144504 (aufgerufen am 13.10.2019).

Taylor, Isabel: A Hydra-like Russian Doll: Appraising and Describing the Shared Drive of a Staatliches Schulamt. In: Jaarboeken Stichting Archiefpublicaties (2018), S. 150–159. https://kvan.courant.nu/issue/JB/2018-11-14/edition/null/page/77 (aufgerufen am 13.10.2019).

Taylor, Isabel: Eine hydraartige Matrjoschka: Wie wir die Fileablage eines staatlichen Schulamtes bewertet und erschlossen haben. Vortragsfolien. Potsdam 2016. https://www.sg.ch/content/dam/sgch/kultur/staatsarchiv/auds-2016/archivierung-von-unterlagen-mit-besonderen-strukturen/01_TAYLOR_Vortragsfolien%20(29.02.16).pdf (aufgerufen am 13.10.2019).

Tesseract at UB Mannheim on GitHub. https://github.com/UB-Mannheim/tesseract/wiki (aufgerufen am 13.10.2019).

Tesseract Open Source OCR Engine (main repository) on GitHub. https://github.com/tesseract-ocr/tesseract (aufgerufen am 13.10.2019).

The National Archives: The application of technology-assisted review to born-digital records transfer, Inquiries and beyond. Research report. http://www.nationalarchives.gov.uk/documents/technology-assisted-review-to-born-digital-records-transfer.pdf (aufgerufen am 13.10.2019).

Tiemann, Katharina (Hg.): Bewertung und Übernahme elektronischer Unterlagen – Business as usual? Beiträge des Expertenworkshops in Münster am 11. und 12. Juni 2013 (Texte und Untersuchungen zur Archivpflege 28). Münster 2013. https://www.lwl.org/waa-download/publikationen/TUA_28.pdf (aufgerufen am 13.10.2019).

Türck, Verena: Veränderungen von Bewertungsgrundsätzen bei der Übernahme digitaler Unterlagen? Untersuchung von Bewertungsentscheidungen anhand baden-württembergischer Beispiele. Transferarbeit im Rahmen der Laufbahnprüfung für den Höheren Archivdienst. Marburg 2014. https://www.landesarchiv-bw.de/sixcms/media.php/120/57173/Transferarbeit_VerenaTuerck_02.pdf (aufgerufen am 13.10.2019).

Überlieferungsprofil „Nichtstaatliches Archivgut". Erarbeitet im Rahmen der abteilungsübergreifenden Dienstbesprechung „Nichtstaatliches Archivgut" des Landesarchivs Nordrhein-Westfalen. Düsseldorf 2011. http://www.archive.nrw.de/lav/abteilungen/fachbereich_grundsaetze/BilderKartenLogosDateien/Ueberlieferungsbildung/__berlieferungsprofil_NSA.pdf (aufgerufen am 10.09.2017).

Uglean Jackson, Laura und Matthew *McKinley*: It's How Many Terabytes?! A Case Study on Managing Large Born Digital Audio-visual Acquisitions. In: IJDC 11 (2016/2) S. 64–75. http://dx.doi.org/10.2218/ijdc.v11i2.391 (aufgerufen am 13.10.2019).

Uhl, Bodo: Die Geschichte der Bewertungsdiskussion. Wann gab es neue Fragestellungen und warum? In: Andrea *Wettmann* (Hg.): Bilanz und Perspektiven archivischer Bewertung. Beiträge eines archivwissenschaftlichen Kolloquiums. Marburg 1994. S. 11–35.

VdA-Arbeitskreis „Archivische Bewertung" (Hg.): Bewertung elektronischer Fachverfahren. Diskussionspapier des VdA-Arbeitskreises „Archivische Bewertung" (Stand: 9. Dezember 2014). https://www.vda.archiv.net/fileadmin/user_upload/pdf/Arbeitskreise/Archivische_Bewertung/Bewertung_Fachverfahren_Positionen_StandDez2014.pdf (aufgerufen am 13.10.2019).

Wiedeman, Gregory: Python for Archivists: breaking down barriers between systems. https://practicaltechnologyforarchives.org/issue7_wiedeman/ (aufgerufen am 13.10.2019).

Wiegand, Peter: Das „archivische Foto" – Überlegungen zu seiner Bewertung. In: Rundbrief Fotografie 11 (2004/1) S. 19–24.

Wolf, Tanja: Stadt im Bild 2.0. Digitale Fotosammlungen in der Stadtverwaltung Worms. In: Kai *Naumann* und Peter *Müller* (Hg.): Das neue Handwerk. Digitales Arbeiten in kleinen und mittleren Archiven. Vorträge des 72. Südwestdeutschen Archivtags am 22. und 23. Juni 2012 in Bad Bergzabern. Stuttgart 2013. S. 66–72. https://www.landesarchiv-bw.de/sixcms/media.php/120/59735/LETZTE_Das%20Handwerk%20Inh%2096S.pdf (aufgerufen am 13.10.2019).

Zahnhausen, Vera: Überlieferungsbildung von analog zu digital – Erfahrungen bei der Übernahme von digitalem Archivgut. In: Katharina *Tiemann* (Hg.): Bewertung und Übernahme elektronischer Unterlagen – Business as usual? Beiträge des Expertenworkshops in Münster am 11. und 12. Juni 2013 (Texte und Untersuchungen zur Archivpflege 28). Münster 2013. S. 8–19.

Verzeichnis der für den Aufbereitungsprozess verwendeten Hard- und Software

I. Hardware

Computer
Macbook Pro 11,1 (2013); Intel Core i5 2.60 GHz, 8 GB RAM

II. Software

Betriebssystem
Windows 10 Pro v.10.0.15063

Office-Anwendungen und Texteditor
LibreOffice Writer u. Calc v.5.3.5.2
Microsoft Word und Excel 2016 v.1706
Notepad++ v.7.3.3

Programmierung
Python 3.6.1
Zusätzliche Python-Bibliotheken
natsort v.5.0.3
numpy v.1.13.0
opencv v.3.2.0
pillow v.4.1.1
pytesseract v.0.1.7

Sonstige Tools
IrfanView v.4.44
Tesseract v.4.0.0
Total Commander v.9.0a
TreeSize Professional. v.6.3.7

Python-Skripte

I. ordner_liste.py

```
"""Skript zur Erstellung von Verzeichnislisten. Alternative zu entsprechenden
Listen aus Treesize o.ä. mit definiertem Ausgabeformat zur
Weiterverarbeitung. Nach Starten des Skripts kann zwischen einer reinen
Ordnerliste und einer erweiterten Liste "Verzeichnispfad;Anzahl
Unterverzeichnisse; Anzahl Bilddateien; Anzahl sonstige Dateien" gewählt
werden. Ausgabe: CSV-Datei.
"""

import csv
from datetime import datetime
from natsort import natsorted
import os

logname = 'ordnerliste_aipp_original'  # Hier den Lognamen eintragen

os.makedirs('./logs/ordner_liste_aipp', exist_ok=True)

log_csv = csv.writer(open(
    f'./logs/ordner_liste_aipp/{logname}_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
    encoding='utf-8'), delimiter=';', lineterminator='\n', quoting=csv.QUOTE_ALL)

log_csv.writerow(['Verzeichnisse'] + ['Unterverzeichnisse'] + ['Bilddateien'] +
                 ['Sonstige Dateien'])


def main():
    stammpfad, antwort = eingabe()
    verzeichnisliste(stammpfad, antwort)
    print('\n' + 'Prozess abgeschlossen.')


def eingabe():
    # Eingabe des Stammpfads; Wahlmöglichkeit: Einfache Verzeichnisliste,
    # erweiterte Liste
    print()
```

```
37      while True:
38          stammpfad = input('Pfad des Stammverzeichnisses?'
39                            ' (z.B. F:\\testsuite_1): ')
40          if os.path.exists(stammpfad) and stammpfad[2] == os.path.sep:
41              break
42          elif not os.path.exists(stammpfad):
43              print('\nBitte einen existenten Pfad eingeben.\n')
44          else:
45              print('\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
46      print()
47      while True:
48          antwort = input('Erweiterte Verzeichnisliste? Bitte "J" für Ja,'
49                          ' "N" für Nein eingeben: ')
50          if antwort.lower() == 'j' or antwort.lower() == 'n':
51              break
52          else:
53              print('\nBitte "J" für Ja oder "N" für Nein eingeben.\n')
54      print()
55      return stammpfad, antwort
56
57
58  def verzeichnisliste(stammpfad, antwort):
59      # Gehe durch Verzeichnisstruktur und erstelle CSV-Datei
60      for verzeichnis, unterverzeichnisse, dateien in os.walk(stammpfad,
61                                                              topdown=True):
62          unterverzeichnisse = natsorted(unterverzeichnisse,
63                                         key=lambda y: y.lower())
64          dateipfade = []
65          if antwort.lower() == 'j':
66              if len(dateien) > 0:
67                  for name in dateien:
68                      bezeichnung, endung = os.path.splitext(name)
69                      typen = ['.jpg', '.jpeg', '.tif', '.tiff']
70                      if endung.lower() in typen:
71                          dateipfade.append(name)
72                          sonstige = (len(dateien) - len(dateipfade))
73                  log_csv.writerow([verzeichnis] + [len(unterverzeichnisse)] +
74                                   [len(dateipfade)] + [sonstige])
75
76              else:
```

```
77                  log_csv.writerow([verzeichnis] + [len(unterverzeichnisse)] +
78                                   ['0'] + ['0'])
79          else:
80              log_csv.writerow([verzeichnis])
81
82
83  if __name__ == '__main__':
84      main()
85
```

II. ordner_loeschen.py

```
1   """
2   Skript zum Löschen von Ordnern über CSV-Liste mit der Spalte "Ordner".
3   Verwendet zur Löschung von definitiv zu kassierenden Ordnern zu Beginn
4   des Workflows. Ein Verschieben dieser Ordner in "kassierte_Dateien" wird hier
5   ausnahmsweise bewusst vermieden, da diese Ordner vorab bereits bewertet wurden
6   und keine Dateien dieser Ordner durch weitere Skripte / Tools berücksichtigt
7   werden sollen, ohne dies explizit bei jedem weiteren Schritt berücksichtigen
8   zu müssen.
9   """
10
11  import csv
12  from datetime import datetime
13  import os
14  import re
15  import shutil
16
17  os.makedirs('./logs/geloeschte_ordner_aipp', exist_ok=True)
18
19  i = csv.writer(open(
20      f'./logs/geloeschte_ordner_aipp/vorab_geloeschte_ordner_'
21      f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
22      encoding='utf-8'), delimiter=';', lineterminator='\n')
23
24  i.writerow(['Geloeschte Ordner'])
25
26
27  # Eingabe der CSV-Datei
```

```
28  print()
29  while True:
30      zu_loeschende_ordner = input(
31          'CSV-Datei mit der Spalte "Ordner" und den Pfaden der zu loeschenden'
32          ' Ordner angeben. (z.B. C:\\ordner_loeschen_aipp.csv): ')
33      if os.path.exists(zu_loeschende_ordner) and zu_loeschende_ordner[2] == \
34              os.path.sep and re.search(r'^.+\.csv', zu_loeschende_ordner):
35          break
36      elif not os.path.exists(zu_loeschende_ordner):
37          print('\n' + 'Bitte eine existente Datei angeben.' + '\n')
38      else:
39          print('\n' + 'Bitte auf korrekte Schreibweise der Pfadangabe achten.'
40                + '\n')
41  print()
42
43  # Endgültiges Löschen der Verzeichnisse
44  with open(zu_loeschende_ordner,
45            encoding='utf-8') as quelldatei:
46      tabelle = csv.DictReader(quelldatei, delimiter=';', lineterminator='\n')
47      for zeile in tabelle:
48          pfad = zeile['Ordner']
49          if os.path.exists(pfad):
50              shutil.rmtree(pfad)
51              print('Ordner gelöscht: ' + pfad)
52              i.writerow([pfad])
53          else:
54              print('Ordner nicht vorhanden: ' + pfad)
55
56  print('\n' + 'Prozess abgeschlossen.')
57
```

III. ordner_loesche_leere.py

```
1   """
58  Skript zum Aufspüren und automatischen Löschen von leeren Verzeichnissen,
59  aus denen keine SIPS gebildet werden sollen
60  """
61
62  import csv
```

```
63  from datetime import datetime
64  import os
65
66  os.makedirs('./logs/geloeschte_loeschen_aipp', exist_ok=True)
67
68  log_csv = csv.writer(open(
69      f'./logs/geloeschte_ordner_aipp/geloeschte_leere_ordner_'
70      f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
71      encoding='utf-8'), delimiter=';', lineterminator='\n')
72
73  log_csv.writerow(['Geloeschte Ordner'])
74
75
76  # Eingabe des Stammverzeichnisses
77  print()
78  while True:
79      stammpfad = input(
80          'Pfad des Stammverzeichnisses, aus dem sämtliche leeren Verzeichnisse'
81          ' entfernt werden sollen? (z.B. F:\\Aipp): ')
82      if os.path.exists(stammpfad) and stammpfad[2] == os.path.sep:
83          break
84      elif not os.path.exists(stammpfad):
85          print('\nBitte einen existenten Pfad eingeben.\n')
86      else:
87          print(
88              '\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
89  print()
90
91  # Suche nach leeren Verzeichnissen, Löschen, Logging
92  for verzeichnis, unterverzeichnisse, dateien in os.walk(stammpfad,
93                                                          topdown=True):
94      if verzeichnis.count(os.path.sep) > 1:
95          ebene2 = verzeichnis.split(os.path.sep)[2]
96          if ebene2 != 'kassierte_dateien':
97              if len(os.listdir(verzeichnis)) == 0:
98                  os.rmdir(verzeichnis)
99                  print(verzeichnis)
100                 log_csv.writerow([verzeichnis])
101
102 print('\n' + 'Prozess abgeschlossen.')
103
```

IV. ordner_umbenennen.py

```
"""
Skript zur Umbenennung von Ordnern nach Redigieren der Ordnernamen.
Eingabe: CSV-Datei mit bestehenden und korrigierten Ordnernamen ("Alter Pfad",
"Neuer Pfad")
"""

import csv
from datetime import datetime
import os
import re
import shutil

os.makedirs('./logs/ordner_umbenennen_aipp', exist_ok=True)

log_csv = csv.writer(open(
    f'./logs/ordner_umbenennen_aipp/umbenannte_ordner_aipp_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
    encoding='utf-8'), delimiter=';', lineterminator='\n')

log_csv.writerow(['Alter Pfad'] + ['Neuer Pfad'])


# Eingabe einer CSV-Datei
print()
while True:
    pfade_csv = input('CSV-Datei mit den Spalten "Alter Ordnername" und '
                      '"Neuer Ordnername" angeben. (z.B. C:\\ordnernamen.csv): ')
    if os.path.exists(pfade_csv) and pfade_csv[2] == os.path.sep and \
            re.search(r'^.+\.csv', pfade_csv):
        break
    elif not os.path.exists(pfade_csv):
        print('\nBitte eine existente Datei angeben.\n')
    else:
        print(
            '\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
print()

with open('{}'.format(pfade_csv), encoding='utf-8') as liste:
```

```
    # Lesen der Ordnerpfade in CSV-Datei und Umbenennen der Verzeichnisse
    tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
    for zeile in tabelle:
        alter_ordnername = zeile['Alter Ordnername']
        neuer_ordnername = zeile['Neuer Ordnername']
        if os.path.exists(
                alter_ordnername) and alter_ordnername != neuer_ordnername:
            shutil.move(alter_ordnername, neuer_ordnername)
            print(
                f'Ordner "{alter_ordnername}" umbenannt in: "{neuer_ordnername}"')
            log_csv.writerow([alter_ordnername] + [neuer_ordnername])

print('Prozess abgeschlossen.')

```

V. ordner_einebnen.py

```
"""Das Skript ebnet die Verzeichnisstruktur auf Ebene 2 (Ereignisebene) ein.
Auf höhere Verzeichnisebenen verschobene Dateien werden umbenannt. Die Namen
der eliminierten Verzeichnisse dienen als Namens-Präfix. Zusätzlich werden
jeweils die ersten Bilddateien in den eingeebneten Verzeichnissen geloggt.
Diese können so, analog zu den ersten Dateien in den Hauptordnern, automatisch
bei der Zufallsselektion von der Kassation ausgenommen werden.
"""

import csv
from datetime import datetime
from natsort import natsorted
import os
import shutil

# Logge die Pfade der ersten X Bilddateien aus den eingeebneten Ordnern,
# d.h. derjenigen Dateien, die sicher übernommen werden sollen
anzahl = 4

os.makedirs('./logs/ordner_einebnen_aipp', exist_ok=True)

verschobene_dateien_log = csv.writer(open(
    f'./logs/ordner_einebnen_aipp/verschobene_dateien_'
```

```
23          f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
24          encoding='utf-8'), delimiter=';', lineterminator='\n',
25          quoting=csv.QUOTE_ALL)
26
27      verschobene_dateien_log.writerow(['Alter Pfad'] + ['Neuer Pfad'])
28
29      aufgeloeste_ordner_log = csv.writer(open(
30          f'./logs/ordner_einebnen_aipp/aufgeloeste_verzeichnisse_'
31          f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
32          encoding='utf-8'), delimiter=';', lineterminator='\n',
33          quoting=csv.QUOTE_ALL)
34
35      aufgeloeste_ordner_log.writerow(['Aufgelöste Verzeichnisse'])
36
37      k = csv.writer(open(
38          f'./logs/ordner_einebnen_aipp/erste_bilder_aus_unterordnern_'
39          f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
40          encoding='utf-8'), delimiter=';', lineterminator='\n',
41          quoting=csv.QUOTE_ALL)
42
43      k.writerow(['Pfade'])
44
45
46      def main():
47          stammpfad = eingabe()
48          ebne_vz_struktur_ein(stammpfad)
49          loesche_tiefere_verzeichnisse(stammpfad)
50          print('\n' + 'Prozess abgeschlossen.')
51
52
53      def eingabe():
54          # Eingabe des Stammverzeichnisses
55          print()
56          while True:
57              stammpfad = input('Pfad des Stammverzeichnisses? (F:\\Aipp): ')
58              if os.path.exists(stammpfad) and stammpfad[2] == os.path.sep:
59                  break
60              elif not os.path.exists(stammpfad):
61                  print('\nBitte einen existenten Pfad eingeben.\n')
62              else:
```

```
63                print(
64                    '\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
65        print()
66        return stammpfad
67
68
69    def ebne_vz_struktur_ein(stammpfad):
70        # Ebne Verzeichnissstruktur auf Ebene1 ein; relativ zum Stammpfad.
71        # Hänge Namenspräfix an verschobene Dateien.
72        # Logge außerdem die ersten Dateien in den aufgelösten Unterordnern.
73        for verzeichnis, unterverzeichnisse, dateien in os.walk(stammpfad,
74                                                                topdown=True):
75            dateien = natsorted(dateien, key=lambda y: y.lower())
76            dateien_zaehler = 0
77            anzahl_zaehler = 0
78            for dateiname in dateien:
79                if os.path.join(verzeichnis, dateiname).count(os.path.sep) > 3:
80                    alter_pfad = os.path.join(verzeichnis, dateiname)
81                    sammlung = os.path.sep.join(verzeichnis.split(os.path.sep)[:2])
82                    ereignis = verzeichnis.split(os.path.sep)[2]
83                    bezeichnung, endung = os.path.splitext(dateiname)
84                    # Beim Logging der ersten Dateien berücksichtigte Dateitypen
85                    typen = ['.jpg', '.jpeg', '.tif', '.tiff']
86                    vz_tiefe = os.path.join(verzeichnis).count(os.path.sep)
87                    aufzuloesender_ordner = \
88                        alter_pfad.split(os.path.sep)[vz_tiefe]
89                    praefix = \
90                        aufzuloesender_ordner[:len(aufzuloesender_ordner)]
91                    i = 1
92                    while i <= vz_tiefe:
93                        ebene_hoeher = alter_pfad.split(os.path.sep)[vz_tiefe - i]
94                        if ebene_hoeher == ereignis:
95                            break
96                        else:
97                            aufzuloesender_ordner_x = \
98                                alter_pfad.split(os.path.sep)[vz_tiefe - i]
99                            praefix_x = aufzuloesender_ordner_x[
100                                                :len(aufzuloesender_ordner_x)]
101                           praefix = '_'.join(praefix_x + praefix)
102                       i += 1
```

```
103                 neuer_pfad = os.path.join(sammlung, ereignis,
104                                           f'{praefix}_{dateiname}')
105                 print('Alter Pfad: ' + alter_pfad)
106                 print('Neuer Pfad: ' + neuer_pfad)
107                 shutil.move(alter_pfad, neuer_pfad)
108                 verschobene_dateien_log.writerow([alter_pfad] + [neuer_pfad])
109
110                 # Logge die ersten X (anzahl) Dateien
111                 if endung.lower() in typen:
112                     if anzahl_zaehler < anzahl:
113                         print('Ausgewählte Datei: ' + neuer_pfad)
114                         k.writerow([neuer_pfad])
115                         anzahl_zaehler += 1
116                 dateien_zaehler += 1
117
118
119 def loesche_tiefere_verzeichnisse(stammpfad):
120     # Loesche die tieferen Verzeichnisse, nachdem die Dateien
121     # herausgezogen wurden.
122     # topdown = False zur rekursiven Loeschung der leeren Verzeichnisse
123     for verzeichnis, unterverzeichnisse, dateien in os.walk(stammpfad,
124                                                             topdown=False):
125         if verzeichnis.count(os.path.sep) > 2:
126             # Löschung der aufgelösten Verzeichnisse endgültig, send2trash
127             # nachinstallieren, wenn Verschieben in "Papierkorb" gewünscht
128             os.rmdir(verzeichnis)
129             print('\n' + 'Verzeichnis geloescht: ' + verzeichnis)
130             aufgeloeste_ordner_log.writerow([verzeichnis])
131
132
133 if __name__ == '__main__':
134     main()
135
```

VI. ordner_vollstaendig_wiederherstellen.py

```
"""
Skript, um nach der automatisierten Bewertung, Ordner, die vollständig
archiviert werden sollen, wieder zu vervollständigen. Dateitypen, die nicht
archiviert werden sollen, sind in Z. 77 zu spezifizieren. Sie bleiben, falls
vorhanden, unter den kassierten Dateien.
"""

import csv
from datetime import datetime
import os
import re
import shutil


logname = 'vollstaendig_uebernommen_aipp'  # Bezeichnung der Logdatei

os.makedirs('./logs/ordner_vollstaendig_aipp', exist_ok=True)

log_csv = csv.writer(open(f'./logs/ordner_vollstaendig_aipp/{logname}_'
                          f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv',
                          'w', encoding='utf-8'), delimiter=';',
                     lineterminator='\n')

log_csv.writerow(['Alter Pfad'] + ['Neuer Pfad'])


def main():
    pfade_csv = eingabe()
    ordnerliste = erstelle_liste(pfade_csv)
    sortiere_dateien_zurueck(ordnerliste)


def eingabe():
    # Eingabe einer CSV-Datei mit Ordnerpfaden
    print()
    while True:
        pfade_csv = input(
            'CSV-Datei mit Spalte "Ordner" und Pfaden der vollständig zu'
```

```
39              'übernehmenden Ordner angeben. (z.B. C:\\Ordner_Uebernahme.csv): ')
40          if os.path.exists(pfade_csv) and pfade_csv[2] == os.path.sep and \
41                  re.search(r'^.+\.csv', pfade_csv):
42              break
43          elif not os.path.exists(pfade_csv):
44              print('\nBitte eine existente Datei angeben.\n')
45          else:
46              print('\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
47      print()
48      return pfade_csv
49
50
51  def erstelle_liste(pfade_csv):
52      # Lese Dateipfade aus CSV-Datei
53      with open('{}'.format(pfade_csv), encoding='utf-8') as liste:
54          ordnerliste = []
55          tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
56          for row in tabelle:
57              ordnerliste.append(row['Ordner'])
58      return ordnerliste
59
60
61  def sortiere_dateien_zurueck(ordnerliste):
62      # Sortiere Dateien von "F:\Sammlung\kassierte_dateien_\Ereignis"
63      # nach "F:\Sammlung\Ereignis"
64      for k in ordnerliste:
65          for verzeichnis, unterverzeichnisse, dateien in os.walk(k,
66                                                                  topdown=True):
67              datei_zaehler = 1
68              print('Starte mit: {}\n'.format(verzeichnis))
69              for name in dateien:
70                  sammlung = os.path.sep.join(k.split(os.path.sep)[:2])
71                  ereignis = k.split(os.path.sep)[3]
72                  dateipfad = os.path.join(k, name)
73                  neuer_pfad = os.path.join(sammlung, ereignis, name)
74                  bezeichnung, endung = os.path.splitext(name)
75                  # Dateitypen, die nicht archiviert werden sollen,
76                  # können hier ausgeschlossen werden
77                  typen = ['.db', '.ini']
78                  if endung.lower() not in typen:
```

```
79                      shutil.move(dateipfad, neuer_pfad)
80                      print(f'{datei_zaehler}. Datei verschoben von:'
81                            f' "{dateipfad}" nach: "{neuer_pfad}"')
82                      log_csv.writerow([dateipfad] + [neuer_pfad])
83                      datei_zaehler += 1
84              if len(os.listdir(verzeichnis)) == 0:
85                  os.rmdir(verzeichnis)
86                  print('\nVerzeichnis vollständig wiederhergestellt und aus '
87                        'Ordner "kassierte_dateien" entfernt.')
88              else:
89                  print('\nMindestens eine nicht als archivwürdig bewertete '
90                        'Datei in "kassierte_dateien" verblieben. Verzeichnis '
91                        'wurde daher nicht entfernt.')
92              print()
93      print('\nAusgewählte Ordner mit archivwürdigen Dateien vollstaendig '
94            'wiederhergestellt.')
95
96
97  if __name__ == '__main__':
98      main()
99
```

VII. dateien_liste.py

```
1   """Das Skript erstellt eine Dateipfadliste. Wahlweise mit oder ohne
2   MD5-Prüfsummen. Zudem kann gewählt werden, ob die in Z.96 definierten
3   Bildtypen oder alle restlichen Dateitypen aufgelistet werden sollen. Wahl
4   jeweils über die Eingabe nach Start des Skripts. Durch ändern von != nach ==
5   in Z.98 können wahlweise die ausgewählten Dateien ODER kassierte_dateien
6   gelistet werden.
7
8   Anwendungsbeispiele:
9
10  Reine Bilderliste von Auswahl zur Suche nach unscharfen Dateien.
11  Reine Bilderliste der kassierten Dateien, zum Auffinden von Tafeln mithilfe
12  von "bilder_vergleich.py"
13  """
14
15  import csv
```

```
16  from datetime import datetime
17  import hashlib
18  from natsort import natsorted
19  import os
20  import shutil
21  import tempfile
22
23  logname = 'erste_dateien'  # Hier bitte den Lognamen eintragen
24
25  os.makedirs('./logs/dateien_liste_aipp', exist_ok=True)
26
27  log_csv = csv.writer(open(
28      f'./logs/dateien_liste_aipp/{logname}_'
29      f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
30      encoding='utf-8'), delimiter=';', lineterminator='\n',
31      quoting=csv.QUOTE_ALL)
32
33  log_csv.writerow(['Bilder'] + ['MD5'])
34
35  log_txt = tempfile.NamedTemporaryFile(mode='w', delete=False, encoding='ANSI')
36
37  log_txt.write('Bilder\n')
38
39
40  def main():
41      stammpfad, antworten = eingabe()
42      erstelle_dateiliste_csv(stammpfad, antworten)
43      schreibe_txt(antworten[0])
44      print('\n' + 'Prozess abgeschlossen.')
45
46
47  def eingabe(antworten=None):
48      # Eingabe von Stammpfad, Wahl: Liste Bilder / Sonstige Dateien,
49      # Hashwerte Ja / Nein
50      if antworten is None:
51          antworten = []
52      print()
53      while True:
54          stammpfad = input('Pfad des Stammverzeichnisses? (z.B. F:\\Aipp): ')
55          if os.path.exists(stammpfad) and stammpfad[2] == os.path.sep:
```

```
56                  break
57              elif not os.path.exists(stammpfad):
58                  print('\nBitte einen existenten Pfad eingeben.\n')
59              else:
60                  print('\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
61          print()
62          while True:
63              a = input(
64                  'Welche Dateitypen sollen erfasst werden? "B" für Bilder / "S" '
65                  'für Sonstige: ')
66              if a.lower() == 'b' or a.lower() == 's':
67                  antworten.extend(a)  # antworten[0]
68                  break
69              else:
70                  print(
71                      '\nBitte "B" für Bilder oder "S" für Sonstige eingeben.\n')
72          print()
73          while True:
74              b = input('MD5-Prüfsummen erstellen? "J" für Ja / "N" für Nein: ')
75              if b.lower() == 'j' or b.lower() == 'n':
76                  antworten.extend(b)  # antworten[1]
77                  break
78              else:
79                  print('\nBitte "j" für Ja oder "n" für Nein eingeben.\n')
80          print()
81          return stammpfad, antworten
82
83
84  def erstelle_dateiliste_csv(stammpfad, antworten, ebene2=''):
85          # Erstelle Dateiliste
86          for verzeichnis, unterverzeichnisse, dateien in os.walk(stammpfad,
87                                                                  topdown=True):
88              dateien = natsorted(dateien,
89                                  key=lambda y: y.lower())  # Windows-Sortierung
90              for name in dateien:
91                  datei = os.path.join(verzeichnis, name)
92                  bezeichnung, endung = os.path.splitext(name)
93                  if os.path.join(verzeichnis, name).count(os.path.sep) > 3:
94                      ebene2 = (verzeichnis.split(os.path.sep)[2])
95                      # Bild-Dateitypen, die gelistet werden sollen
```

```
96                  typen = ['.jpg', '.jpeg', '.tif', '.tiff']
97              # "kassierte_dateien" einschließen?
98              if ebene2 != 'kassierte_dateien':
99                  if (antworten[0].lower() == 'b' and endung.lower() in typen) \
100                         or (antworten[0].lower() == 's' and endung.lower()
101                             not in typen):
102                     schreibe_csv(datei, antworten[1])
103
104
105 def get_md5(datei):
106     # Berechnung von MD5-Prüfsummen
107     dateihash = hashlib.md5()
108     dateihash.update(open(datei, 'rb').read())
109     md5 = dateihash.hexdigest()
110     return md5
111
112
113 def schreibe_csv(datei, antwort, md5=''):
114     # Ausgabe in CSV-Datei (und temp. TXT-File)
115     if antwort.lower() == 'j':
116         md5 = get_md5(datei)
117         print(md5 + '\t' + datei)
118     log_csv.writerow([datei] + [md5])
119     log_txt.write(datei + '\n')
120
121
122 def schreibe_txt(antwort):
123     # Bei Bilderlisten wird die zusätzliche TXT-Datei (ANSI) zur Übergabe an
124     # bilder_oeffnen.py dauerhaft gespeichert
125     log_txt.close()
126     logTempFile = log_txt.name
127     if antwort.lower() == 'b':
128         shutil.copy2(logTempFile,
129                      f'./logs/dateien_liste_aipp/{logname}_'
130                      f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.txt')
131     os.remove(logTempFile)
132
133
134 if __name__ == '__main__':
135     main()
136
```

VIII. dateien_kassieren.py

```
"""Das Skript kassiert eine Datei-Auswahl, d.h. die Dateien werden in das
entsprechende Verzeichnis im Ordner "kassierte_dateien" verschoben.
"""

import csv
from datetime import datetime
import os
import re
import shutil

logname = 'kassierte_aipp_unscharfe'  # Hier Namen der Logdatei angeben

os.makedirs('./logs/dateien_kassieren_aipp', exist_ok=True)

log = csv.writer(open(
    f'./logs/dateien_kassieren_aipp/{logname}_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
    encoding='utf-8'), delimiter=';', lineterminator='\n')

log.writerow(['Alter Pfad'] + ['Neuer Pfad'])


def main():
    dateipfade_csv = eingabe()
    ausnahmen = ausnahme()
    kassiere_dateien(dateipfade_csv, ausnahmen)
    print('\n' + 'Prozess abgeschlossen.')


def eingabe():
    # Eingabe von CSV-Datei mit zu kassierenden Dateipfaden
    print()
    while True:
        pfade_csv = input(
            'CSV-Datei mit der Spalte "Dateien" und den Pfaden der zu'
            ' kassierenden Dateien angeben.(z.B. C:\\unscharfe_bilder.csv): ')
        if os.path.exists(pfade_csv) and pfade_csv[2] == os.path.sep and \
                re.search(r'^.+\.csv', pfade_csv):
```

```
39              break
40          elif not os.path.exists(pfade_csv):
41              print('\nBitte eine existente Datei angeben.\n')
42          else:
43              print('\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
44      print()
45      return pfade_csv
46
47
48  def ausnahme(ausnahmen=None):
49      # Eingabe von CSV-Datei mit Ausnahmen
50      if ausnahmen is None:
51          ausnahmen = []
52      while True:
53          pfad_csv = input(
54              'Sollen bestimmte Dateien NICHT kassiert werden? CSV-Datei mit der'
55              ' Spalte "Pfade" angeben (z.B. C:\\erste_bilder_in_'
56              'unterordnern.csv). Sonst weiter mit "Enter": ')
57          if os.path.exists(pfad_csv) and pfad_csv[2] == os.path.sep and \
58                  re.search(r'^.+\.csv', pfad_csv):
59              break
60          elif pfad_csv and not os.path.exists(pfad_csv):
61              print('\nBitte eine existente Datei angeben.\n')
62          elif not pfad_csv:
63              break
64          else:
65              print(
66                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
67
68      if pfad_csv:
69          with open(pfad_csv, encoding='utf-8') as liste:
70              tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n',
71                                       quoting=csv.QUOTE_ALL)
72              for row in tabelle:
73                  ausnahmen.append(row['Pfade'])
74      print()
75      return ausnahmen
76
77
78  def kassiere_dateien(datei_liste, ausnahmen):
```

```
79      # Verschiebe die zu kassierenden Dateien von "F:\Sammlung\Ereignis" nach
80      # "F:\Sammlung\kassierte_dateien\Ereignis"
81      with open(f'{datei_liste}', encoding='utf-8') as liste:
82          tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
83          datei_zaehler = 1
84          for row in tabelle:
85              zu_kassierende_datei = row['Dateien']
86              sammlung = os.path.sep.join(
87                  zu_kassierende_datei.split(os.path.sep)[:2])
88              ereignis = zu_kassierende_datei.split(os.path.sep)[2]
89              dateiname = zu_kassierende_datei.split(os.path.sep)[3]
90              kassierte_datei = os.path.join(sammlung, 'kassierte_dateien')
91              ereignis_kd = os.path.join(kassierte_datei, ereignis)
92              neuer_pfad = os.path.join(
93                  sammlung, kassierte_datei, ereignis, dateiname)
94              if zu_kassierende_datei not in ausnahmen:
95                  if not os.path.exists(ereignis_kd):
96                      os.makedirs(ereignis_kd)
97                  if not os.path.exists(neuer_pfad):
98                      shutil.move(zu_kassierende_datei, neuer_pfad)
99                      print(f'{datei_zaehler}. Datei verschoben von: "' +
100                           zu_kassierende_datei + '" nach: "' + neuer_pfad + '"')
101                     log.writerow([zu_kassierende_datei] + [neuer_pfad])
102                 else:
103                     print(f'Datei mit gleichem Namen befindet sich bereits '
104                           f'unter den kassierten Dateien: {ereignis_kd}')
105                 datei_zaehler += 1
106
107
108 if __name__ == '__main__':
109     main()
110
```

IX. dateien_kassierte_einsortieren.py

```
1   """Das Skipt dient zum nachträglichen Rücksortieren von vorläufig
2   kassierten Dateien. Anwendung für Tafeln.
3   """
4
```

```
import csv
from datetime import datetime
import os
import re
import shutil

logname = 'ruecksortierte_tafeln'  # Hier Namen der Logdatei angeben

os.makedirs('./logs/dateien_ruecksortierte', exist_ok=True)

log = csv.writer(open(
    f'./logs/dateien_ruecksortierte/{logname}_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
    encoding='utf-8'), delimiter=';', lineterminator='\n')

log.writerow(['Alter Pfad'] + ['Neuer Pfad'])


# Eingabe CSV-Datei mit Pfaden der zu archivierenden Dateien
print()
while True:
    pfad_csv = input(
        'CSV-Datei mit Spalte "Dateien" und Pfaden der einzusortierenden'
        ' Dateien angeben. (z.B. C:\\kassierte_tafeln.csv): ')
    if os.path.exists(pfad_csv) and pfad_csv[2] == os.path.sep and \
            re.search(r'^.+\.csv', pfad_csv):
        break
    elif not os.path.exists(pfad_csv):
        print('\nBitte eine existente Datei angeben.\n')
    else:
        print('\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
print()

# Sortiere Dateien von "F:\Sammlung\kassierte_dateien\Ereignis" nach
# "F:\Sammlung\Ereignis"
with open('{}'.format(pfad_csv), encoding='utf-8') as liste:
    tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
    datei_zaehler = 1
    for row in tabelle:
        einzusortierende_datei = row['Dateien']
```

```
45            sammlung = \
46                os.path.sep.join(einzusortierende_datei.split(os.path.sep)[:2])
47            ereignis = einzusortierende_datei.split(os.path.sep)[3]
48            dateiname = einzusortierende_datei.split(os.path.sep)[4]
49            neuer_pfad = os.path.join(sammlung, ereignis, dateiname)
50            if os.path.exists(neuer_pfad):
51                print(f'{neuer_pfad}: Datei bereits vorhanden.')
52                break
53            else:
54                shutil.move(einzusortierende_datei, neuer_pfad)
55                print(f'{datei_zaehler}. Datei verschoben von: "' +
56                      einzusortierende_datei + '" nach: "' + neuer_pfad + '"')
57                log.writerow([einzusortierende_datei] + [neuer_pfad])
58                datei_zaehler += 1
59
60    print('\n' + 'Prozess abgeschlossen.')
61
```

X. bilder_oeffnen.py

```
1     """Das Skript übergibt an IrfanView eine Liste (.TXT) mit zu öffnenden
2     Bildern. Diese werden in der Thumbnailansicht geöffnet. Gültiger Pfad zur
3     IrfanView-Installation (.exe) muss in Z.22 angegeben werden!
4
5     Sollten Pfadlängen > 260 Zeichen enthalten sein, werden diese in IrfanView
6     nicht geöffnet. Eine Liste mit den entsprechenden Pfaden wird ausgegeben.
7     (CSV-Datei)
8
9     ACHTUNG!
10    IRFANVIEW NIMMT NUR ANSI-CODIERTE TEXTFILES.
11    PFADE MIT SONDERZEICHEN (UMLAUTE...) WERDEN SONST NICHT GEÖFFNET.
12    KEINE FEHLERMELDUNG!
13    """
14
15    import csv
16    import os
17    import re
18    import subprocess
19    from datetime import datetime
```

```
20
21  # Pfad zur lokalen Installation von IrfanView
22  irfanview = r'C:\Program Files\IrfanView\i_view64.exe'
23
24  # IrfanView-Parameter
25  args = ['/filelist=', '/thumbs']
26
27
28  def main():
29      bildpfade_txt = eingabe()
30      zu_lange_pfade = finde_ueberlaenge(bildpfade_txt)
31      logge_ueber_laenge(zu_lange_pfade)
32      print('\n' + 'Schließen Sie IrfanView um fortzufahren.')
33      oeffne_bilder(bildpfade_txt)
34
35
36  def eingabe():  # Eingabe der Textdatei mit Pfadliste
37      print()
38      while True:
39          bildpfade_txt = input('Bitte Textdatei (ANSI) mit Pfadliste angeben. '
40                                '(z.B. C:\\bilderliste.txt): ')
41          if os.path.exists(bildpfade_txt) and bildpfade_txt[2] == os.path.sep \
42                  and re.search(r'^.+\.txt', bildpfade_txt):
43              break
44          elif not os.path.exists(bildpfade_txt):
45              print('\nBitte Pfad zu existierender Datei eingeben.\n')
46          else:
47              print(
48                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten und '
49                  'Textdatei wählen.\n')
50      print()
51      return bildpfade_txt
52
53
54  def finde_ueberlaenge(pfade_txt, zu_lange_pfade=None):
55      # Überprüfung der Pfade und Ausgabe, wenn < 260 Zeichen.
56      if zu_lange_pfade is None:
57          zu_lange_pfade = []
58      with open(pfade_txt, encoding='ANSI') as quelldatei:
59          zaehler = 1
```

```
60          for line in quelldatei:
61              bildtypen = re.match(r'^.+\.bmp$|^.+\.jpg$|^.+\.jpeg$|^.+\.png$|'
62                                   r'^.+\.tif$|^.+\.tiff$', line, re.IGNORECASE)
63              if bildtypen is not None:
64                  if len(line) <= 260:
65                      print(f'{zaehler}. Datei öffnen: ' + line[:-1])
66                      zaehler += 1
67                  elif len(line) > 260:
68                      zu_lange_pfade.append(line)
69      return zu_lange_pfade
70
71
72  def logge_ueber_laenge(zu_lange_pfade):
73      # Ausgabe der Pfade > 260 Zeichen
74      if len(zu_lange_pfade) > 0:
75          os.makedirs('logs/bilder_zulangePfade_aipp', exist_ok=True)
76          log = csv.writer(
77              open('./logs/bilder_zulangePfade_aipp/bilder_zulangePfade_'
78                   + datetime.now().strftime('%d-%m-%Y_%H-%M-%S') + '.csv',
79                   'w', encoding='utf-8'), delimiter=';',
80              lineterminator='\n')
81          log.writerow(['Pfade'])
82          print('\nFolgende Pfade sind zu lang für ein direktes Öffnen in der'
83                ' Thumbnail-Ansicht.\n')
84          zaehler = 1
85          for pfad in zu_lange_pfade:
86              print(f'{zaehler}.: {pfad}')
87              log.writerow([pfad])
88              zaehler += 1
89
90
91  def oeffne_bilder(bilder_liste):
92      # Öffne übergebene Bilder in Irfan-View (Thumbnail-Ansicht)
93      subprocess.call([irfanview, args[0], bilder_liste, args[1]])
94
95
96  if __name__ == '__main__':
97      main()
98
```

XI. bilder_kopiere_erste.py

```
"""Kopiert die in Z.16 festgeleget Anzahl an ersten Bild-Dateien aus jedem
Ordner in ein eigenes Unterverzeichnis (relativ zum Ort der
Skriptausführung). Die Bildauswahl kann anschließend händisch ergänzt /
ausgedünnt werden und dient als Referenz für das Skript "bilder_vergleich.py".

Ausgabe:    TXT-Datei zum Öffnen mit "bilder_oeffnen.py".
            CSV zur Fütterung der Bildvergleich-Skripte.
"""

import csv
from datetime import datetime
from natsort import natsorted
import os
import shutil

anzahl = 2  # Hier Anzahl der zu kopierenden Dateien festlegen

os.makedirs('./logs/bilder_erste_aipp', exist_ok=True)

log_csv = csv.writer(open(
    f'./logs/bilder_erste_aipp/erste_bilder_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
    encoding='utf-8'), delimiter=';', lineterminator='\n')

log_csv.writerow(['Bilder'])

log_txt = open(
    f'./logs/bilder_erste_aipp/erste_bilder_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.txt', 'w', encoding='ANSI')

log_txt.write('Bilder')


def main():
    stammpfad = eingabe()
    kopiere_dateien(stammpfad)
    print('\n' + 'Prozess abgeschlossen.')

```

```
39
40  def eingabe():
41      # Eingabe des Stammverzeichnisses
42      print()
43      while True:
44          stammpfad = input('Pfad des Stammverzeichnisses? (z.B. F:\\Aipp): ')
45          if os.path.exists(stammpfad) and stammpfad[2] == os.path.sep:
46              break
47          elif not os.path.exists(stammpfad):
48              print('\nBitte einen existenten Pfad eingeben.\n')
49          else:
50              print(
51                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
52      print()
53      return stammpfad
54
55
56  def kopiere_dateien(stammpfad):
57      # Durchlaufen aller Verzeichnisse unterhalb des Stammverzeichnisses und
58      # Kopieren der jeweils ersten Dateien nach ".\erste_dateien"
59      for verzeichnis, unterverzeichnisse, dateien in os.walk(
60              stammpfad, topdown=True):
61          datei_zaehler = 0
62          kopierte_dateien_zaehler = 0
63          # Windows-Sortierung (alphabetic natural sort)
64          dateien = natsorted(dateien, key=lambda y: y.lower())
65          # Welche (Bild-)formate sollen berücksichtigt werden?
66          typen = ['.jpg', '.jpeg', '.tif', '.tiff']
67          print('Bearbeitetes Verzeichnis: ' + verzeichnis)
68          while datei_zaehler < len(dateien):
69              dateipfad = os.path.join(verzeichnis, dateien[datei_zaehler])
70              auswahl_ordner = os.path.join(os.getcwd(), 'erste_dateien')
71              if not os.path.exists(auswahl_ordner):
72                  os.makedirs(auswahl_ordner)
73              auswahl_datei = os.path.join(auswahl_ordner, dateien[datei_zaehler])
74              bezeichnung, endung = os.path.splitext(dateien[datei_zaehler])
75              if kopierte_dateien_zaehler < anzahl:
76                  if endung.lower() in typen:
77                      suffix = 0
78                      while os.path.exists(auswahl_datei):
```

```
79                          suffix += 1
80                          auswahl_datei = os.path.join(
81                              auswahl_ordner, f'{bezeichnung}_{suffix}{endung}')
82                      log_csv.writerow([auswahl_datei])
83                      log_txt.write(auswahl_datei + '\n')
84                      shutil.copy2(dateipfad, auswahl_datei)
85                      kopierte_dateien_zaehler += 1
86                      datei_zaehler += 1
87                  else:
88                      datei_zaehler += 1
89              else:
90                  break
91
92
93  if __name__ == '__main__':
94      main()
95
```

XII. bilder_zufallsselektion.py

```
1   """Skript zur Zufallsauswahl von Bilddateien.
2
3   Das Skript wählt eine in Z.32 zu definierende Anzahl an Dateien aus, die vom
4   Anfang eines jeden Verzeichnisses übernommen werden sollen. In Z.34 kann
5   angegebenen werden, wie viele Dateien pro Verzeichnis übersprungen werden
6   sollen, bis eine Xte Datei wiederum übernommen werden soll.
7
8   Dateitypen:
9   Die bei der Selektion berücksichtigten Dateitypen werden in Z.168 festgelegt.
10
11  Das Skript belässt die ausgewählten Dateien in ihren Verzeichnissen. Die
12  restlichen Dateien werden in ein neu angelegtes Verzeichnis
13  "kassierte_dateien" verschoben. Die ursprüngliche Verzeichnisstruktur bleibt
14  darin unverändert erhalten.
15
16  OPTIONAL:
17
18  Definieren von Ausnahmen über CSV-Dateien 1. Über das Einlesen einer
19  CSV-Datei können weitere Dateien bestimmt werden, die auf jeden Fall
```

```
20  übernommen werden. z.B. die ersten Dateien aus den eingeebneten Unterordnern.
21  2. Ebenso können Dateien unabhängig von der Zufallsauswahl kassiert werden.
22  Z.b. unscharfe Dateien, wenn nach diesen vorab bereits gesucht wurde.
23  """
24
25  import csv
26  from datetime import datetime
27  from natsort import natsorted
28  import os
29  import re
30  import shutil
31
32  anzahl = 4  # Die ersten X Dateien sollen ausgewählt werden
33
34  jede_xte_datei = 7  # Jede wievielte Datei soll außerdem gewählt werden?
35
36  os.makedirs('./logs/bilder_zufallsselektion_aipp', exist_ok=True)
37
38  log_auswahl = csv.writer(open(
39      f'./logs/bilder_zufallsselektion_aipp/ausgewaehlte_dateien_'
40      f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
41      encoding='utf-8'), delimiter=';', lineterminator='\n')
42
43  log_auswahl.writerow(['Pfade'])
44
45  log_kassiert = csv.writer(open(
46      f'./logs/bilder_zufallsselektion_aipp/kassierte_dateien_'
47      f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
48      encoding='utf-8'), delimiter=';', lineterminator='\n')
49
50  log_kassiert.writerow(['Alte Dateipfade'] + ['Neue Dateipfade'])
51
52  log_uebersicht = csv.writer(open(
53      f'./logs/bilder_zufallsselektion_aipp/uebersicht_dateien_'
54      f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
55      encoding='utf-8'), delimiter=';', lineterminator='\n')
56
57  log_uebersicht.writerow(['Datei'] + ['Ausgewählt'] + ['Kassiert']
58                          + ['Bereits kassiert'] + ['Pfad der kassierten Datei'])
59
```

```
60
61  def main():
62      stammpfad = eingabe_stammpfad()
63      selektierte_dateien = eingabe_selektion()
64      kassanda = eingabe_kassanda()
65      zufallsselektion(stammpfad, selektierte_dateien, kassanda)
66      print('\n' + 'Prozess abgeschlossen.')
67
68
69  def eingabe_stammpfad():
70      # Eingabe des Stammverzeichnisses
71      print()
72      while True:
73          stammpfad = input('Pfad des Stammverzeichnisses? (z.B. F:\\Aipp\\): ')
74          if os.path.exists(stammpfad) and stammpfad[2] == os.path.sep:
75              break
76          elif not os.path.exists(stammpfad):
77              print('\nBitte einen existenten Pfad eingeben.\n')
78          else:
79              print(
80                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
81      print()
82      if not stammpfad.endswith(os.path.sep):
83          stammpfad = stammpfad + os.path.sep
84      return stammpfad
85
86
87  def eingabe_selektion(archivwuerdige_dateien=None):
88      # Optionale Angabe einer CSV-Datei mit zu archivierenden Dateipfaden
89      if archivwuerdige_dateien is None:
90          archivwuerdige_dateien = []
91      while True:
92          pfad_csv = input(
93              'Kassiere folgende Dateien NICHT. CSV-Datei mit der Spalte "Pfade" '
94              'angeben (z.B. C:\\erste_bilder_in_unterordnern.csv). Sonst weiter'
95              ' mit "Enter": ')
96          if os.path.exists(pfad_csv) and pfad_csv[2] == os.path.sep and \
97                  re.search(r'^.+\.csv', pfad_csv):
98              break
99          elif pfad_csv and not os.path.exists(pfad_csv):
```

```
100             print('\nBitte eine existente Datei angeben.\n')
101         elif not pfad_csv:
102             break
103         else:
104             print(
105                 '\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
106
107     if pfad_csv:
108         with open(pfad_csv, encoding='utf-8') as liste:
109             tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n',
110                                      quoting=csv.QUOTE_ALL)
111             for row in tabelle:
112                 archivwuerdige_dateien.append(row['Pfade'])
113     print()
114     return archivwuerdige_dateien
115
116
117 def eingabe_kassanda(kassable_dateien=None):
118     # Optionale Angabe einer CSV-Datei mit zu kassierenden Dateipfaden
119     if kassable_dateien is None:
120         kassable_dateien = []
121     while True:
122         dateipfade_csv = input(
123             'Kassiere folgende Dateien UNBEDINGT. CSV-Datei mit der Spalte'
124             ' "Pfade" angeben (z.B. C:\\erste_bilder_in unterordnern.csv). '
125             'Sonst weiter mit "Enter": ')
126         if os.path.exists(dateipfade_csv) and dateipfade_csv[2] == os.path.sep \
127                 and re.search(r'^.+\.csv', dateipfade_csv):
128             break
129         elif dateipfade_csv and not os.path.exists(dateipfade_csv):
130             print('\nBitte eine existente Datei angeben.\n')
131         elif not dateipfade_csv:
132             break
133         else:
134             print(
135                 '\nBitte auf korrekte Schreibweise der Pfadangabe achten.\n')
136
137     if dateipfade_csv:
138         with open(dateipfade_csv, encoding='utf-8') as liste:
139             tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n',
```

```
140                                     quoting=csv.QUOTE_ALL)
141             for row in tabelle:
142                 kassable_dateien.append(row['Pfade'])
143     print()
144     return kassable_dateien
145
146
147 def zufallsselektion(stammpfad, vorab_selektierte_dateien, kassanda):
148     # Zufallsselektion, Dateibewegung, Logging
149     stammpfad_tiefe = stammpfad.count(os.path.sep)
150     for verzeichnis, unterverzeichnisse, dateien in os.walk(stammpfad,
151                                                             topdown=True):
152         print('Bearbeitetes Verzeichnis: ' + verzeichnis)
153         datei_zaehler = 0
154         auswahl_zaehler = 0
155         dateien = natsorted(dateien, key=lambda y: y.lower())
156         while datei_zaehler < len(dateien) and len(dateien) > 0:
157             dateipfad = os.path.join(verzeichnis, dateien[datei_zaehler])
158             ereignis = os.path.sep.join(verzeichnis.split(os.path.sep)[stammpfad_
    tiefe:])
159             if 'kassierte_dateien' in ereignis:
160                 break
161             kassierte_dateien = os.path.join(stammpfad, 'kassierte_dateien')
162             ereignis_kd = os.path.join(kassierte_dateien, ereignis)
163             if not os.path.exists(ereignis_kd):
164                 os.makedirs(ereignis_kd)
165             dateipfad_kd = os.path.join(ereignis_kd, dateien[datei_zaehler])
166             bezeichnung, endung = os.path.splitext(dateien[datei_zaehler])
167             # Welche Dateitypen sollen bei der Auswahl berücksichtigt werden?
168             typen = ['.jpg', '.jpeg', '.tif', '.tiff']
169             if endung.lower() in typen:
170                 if auswahl_zaehler < anzahl:
171                     print('Ausgewählte Datei: ' + dateipfad)
172                     log_auswahl.writerow([dateipfad])
173                     log_uebersicht.writerow([dateipfad] + ['x'] + [''] + ['']
174                                             + [''])
175                     auswahl_zaehler += 1
176                 elif datei_zaehler % jede_xte_datei == 0 or dateipfad in\
177                         vorab_selektierte_dateien and dateipfad not in kassanda:
178                     print('Ausgewählte Datei: ' + dateipfad)
```

```
179                         log_auswahl.writerow([dateipfad])
180                         log_uebersicht.writerow([dateipfad] + ['x'] + [''] + ['']
181                                                 + [''])
182                     else:
183                         if not os.path.exists(dateipfad_kd):
184                             shutil.move(dateipfad, dateipfad_kd)
185                             print('Kassierte Datei: ' + dateipfad)
186                             log_kassiert.writerow([dateipfad] + [dateipfad_kd])
187                             log_uebersicht.writerow(
188                                 [dateipfad] + [''] + ['x'] + [''] + [dateipfad_kd])
189                         else:
190                             print('Datei existiert bereits: {}'.format(dateipfad_kd))
191                             log_uebersicht.writerow(
192                                 [dateipfad] + [''] + [''] + ['x'] + [dateipfad_kd])
193                 datei_zaehler += 1
194
195
196     if __name__ == '__main__':
197         main()
```

XIII. bilder_unschaerfe.py

```
1   """Skript zur Erkennung von Unschärfe über die OpenCV-Funktion "Laplacian".
2   """
3
4   import csv
5   import cv2
6   from datetime import datetime
7   import numpy as np
8   import os
9   import re
10
11  logname = 'unscharfe_aipp'  # Hier Lognamen angeben
12
13  os.makedirs('./logs/bilder_unscharfe', exist_ok=True)
14
15  log_csv = csv.writer(open(
16      f'./logs/bilder_unscharfe/{logname}_'
17      f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
```

```
18          encoding='utf-8'), delimiter=';', lineterminator='\n')
19
20      log_csv.writerow(['Bilder'] + ['Hash'])
21
22      log_txt = open(
23          f'./logs/bilder_unscharfe/unscharfe_{logname}_'
24          f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.txt', 'w',
25          encoding='ANSI')
26
27      log_txt.write('Bilder\n')
28
29
30      def main():
31          bildpfade_csv = eingabe()
32          oeffne_bilder(bildpfade_csv)
33          print('\n' + 'Prozess abgeschlossen.')
34
35
36      def eingabe():
37          # Eingabe der CSV-Datei mit Bilderliste
38          print()
39          while True:
40              bilderliste = input('Bitte CSV-Datei mit Bilderliste angeben.'
41                                  ' (z.B. C:\\bilder_liste.csv): ')
42              if os.path.exists(bilderliste) and bilderliste[2] == os.path.sep and \
43                      re.search(r'^.+\.csv', bilderliste):
44                  break
45              elif not os.path.exists(bilderliste):
46                  print('\nBitte Pfad zu existierender Datei eingeben.\n')
47              else:
48                  print(
49                      '\nBitte auf korrekte Schreibweise der Pfadangabe achten und '
50                      'CSV-Datei wählen.\n')
51          print()
52          return bilderliste
53
54
55      def oeffne_bilder(pfade_csv):
56          # Öffne Bilderliste
57          with open(pfade_csv, encoding='utf-8') as liste:
```

```
58          tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
59          for row in tabelle:
60              bild_pfad = row['Bilder']
61              # print(bild_pfad)
62              berechne_laplace(bild_pfad)
63
64
65  def berechne_laplace(bildpfad):
66      # Berechne Schärfewert über Laplace-Funktion von OpenCV
67      stream = open(str(bildpfad), 'rb')
68      bytearray_bild = bytearray(stream.read())
69      array_bild = np.asarray(bytearray_bild, dtype=np.uint8)
70      bild = cv2.imdecode(array_bild, cv2.IMREAD_GRAYSCALE)
71      bild_klein = cv2.resize(bild, None, fx=0.5, fy=0.5,
72                              interpolation=cv2.INTER_AREA)
73      wert = cv2.Laplacian(bild_klein, cv2.CV_64F).var()
74      wert = int(wert)
75      # Grenzwert: Werte < 110 bedeuteten im Test sichtbare Unschärfe,
76      # ~ 65 starke Unschärfe.
77      if wert <= 110:
78          ausgabe(bildpfad, wert)
79
80
81  def ausgabe(bildpfad, wert):  # Ausgabe in CSV- und TXT-Dateien
82      print(f'Wert für Bild "{bildpfad}": {wert}')
83      log_csv.writerow([bildpfad] + [wert])
84      log_txt.write(bildpfad + f' {wert}\n')
85
86
87  if __name__ == '__main__':
88      main()
89
```

XIV. bilder_vergleich.py

```
1   """Skript zum Auffinden von ähnlichen Bildern.
2   Verwendung mit Referenzbildern: Tafeln
3   """
4
```

```
import csv
import cv2
from datetime import datetime
import numpy as np
import os
import re

logname = 'matches_auswahl_aipp'  # Hier Lognamen angeben

os.makedirs('./logs/tafeln_aipp', exist_ok=True)

log_txt = open(
    f'./logs/tafeln_aipp/{logname}_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.txt', 'w', encoding='ANSI')

log_txt.write('Bilder' + '\n')

log_csv = csv.writer(open(
    f'./logs/tafeln_aipp/{logname}_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
    encoding='utf-8'), delimiter=';', lineterminator='\n')

log_csv.writerow(['Bilder'])


def main():
    referenzbilder = eingabe_referenz()
    bildersammlung = eingabe_sammlung()
    startwerte = referenz(referenzbilder)
    vergleich(bildersammlung, startwerte)
    print('\nProzess abgeschlossen.')


def eingabe_referenz():
    # Eingabe einer CSV-Datei mit Pfaden zu Referenzbildern
    print()
    while True:
        referenzbilder = input('Bitte Liste mit Refernz-Bildern angeben.'
                               ' (z.B. C:\\tafelauswahl.csv): ')
        if os.path.exists(referenzbilder) and referenzbilder[2] == \
```

```
45                  os.path.sep and re.search(r'^.+\.csv', referenzbilder):
46              break
47          elif not os.path.exists(referenzbilder):
48              print('\nBitte Pfad zu existierender Datei eingeben.\n')
49          else:
50              print(
51                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten und'
52                  ' CSV-Datei wählen.\n')
53      print()
54      return referenzbilder
55
56
57  def eingabe_sammlung():
58      # Eingabe einer CSV-Datein mit Pfaden der zu durchsuchenden Verzeichnisse
59      while True:
60          bildersammlung = input(
61              'Bitte Liste der Bild-Sammlung, die durchsucht werden soll, '
62              'angeben. (z.B. C:\\bilderliste.csv): ')
63          if os.path.exists(bildersammlung) and bildersammlung[2] == \
64                  os.path.sep and re.search(r'^.+\.csv', bildersammlung):
65              break
66          elif not os.path.exists(bildersammlung):
67              print('\nBitte Pfad zu existierender Datei eingeben.\n')
68          else:
69              print(
70                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten und'
71                  ' CSV-Datei wählen.\n')
72      print()
73      return bildersammlung
74
75
76  def get_farbwerte(bildpfad):
77      # Liste Pixel-Werte für das zu öffnende Bild auf
78      stream = open(str(bildpfad), 'rb')
79      bild_bytes = bytearray(stream.read())
80      numpyarray = np.asarray(bild_bytes, dtype=np.uint8)
81      bild = cv2.imdecode(numpyarray, cv2.IMREAD_COLOR)
82      bild_klein = cv2.resize(bild, None, fx=0.05, fy=0.05,
83                              interpolation=cv2.INTER_AREA)
84      rgb = cv2.cvtColor(bild_klein, cv2.COLOR_BGR2RGB)
```

```
85          hoehe, breite = rgb.shape[:2]
86          for x in range(hoehe):
87              farbwerte = []
88              for y in range(breite):
89                  px = rgb[x, y]
90                  farbwerte = np.append(farbwerte, px)
91          return farbwerte
92
93
94      def referenz(referenzbilder, startwerte=None):
95          # Öffne Tabelle mit Pfaden zu Referenzbildern, speichere deren Farbwerte
96          if startwerte is None:
97              startwerte = []
98          print('\nStarte Sammeln der Referenzwerte.\n')
99          with open(referenzbilder, encoding='utf-8') as liste:
100             tabelle = csv.DictReader(
101                 liste, delimiter=';', lineterminator='\n', quoting=csv.QUOTE_ALL)
102             for row in tabelle:
103                 bildpfad = row['Bilder']
104                 farbwerte = get_farbwerte(bildpfad)
105                 if not np.all(startwerte == farbwerte):
106                     startwerte.append(farbwerte)
107                     print(f'{len(startwerte)}. Wert hinzugefügt.')
108                 else:
109                     print('Wert schon vorhanden.')
110         print()
111         return startwerte
112
113
114     def vergleich(sammlung, startwerte):
115         # Öffne Tabelle mit Pfaden zu Bildern in Sammlung, vergleiche Farbwerte mit
116         # Referenzwerten, speichere falls Übereinstimmung > Grenzwert
117         print('\nStarte Suche nach Übereinstimmungen.\n')
118         with open(sammlung, encoding='utf-8') as liste:
119             tabelle = csv.DictReader(
120                 liste, delimiter=';', lineterminator='\n', quoting=csv.QUOTE_ALL)
121             for row in tabelle:
122                 bildpfad = row['Bilder']
123                 if os.path.exists(bildpfad):
124                     alle = get_farbwerte(bildpfad)
```

```
125                 equal = []
126                 for i in range(len(startwerte)):
127                     if len(alle) == len(startwerte[i]):
128                         equal = np.append(equal, np.count_nonzero(
129                             alle == startwerte[i]))
130                     if sum(s > 300 for s in equal):  # Grenzwert für Ausgabe
131                         print(bildpfad)
132                         log_txt.write(bildpfad + '\n')
133                         log_csv.writerow([bildpfad])
134             else:
135                 print('Bildpfad existiert nicht.')
136
137
138 if __name__ == '__main__':
139     main()
140
```

XV. bilder_ocr.py

```
1   """Skript zur OCR-Texterkennung in Tafeln.
2   Ausgabe: CSV-Datei mit Pfadangaben und dem erkannten Text.
3   """
4
5   import csv
6   import cv2
7   from datetime import datetime
8   import numpy as np
9   import os
10  from PIL import Image
11  import pytesseract
12  import re
13
14  logname = 'tafeltexte_ocr'  # Hier Lognamen angeben
15
16  os.makedirs('./logs/tafeln_aipp', exist_ok=True)
17
18  i = csv.writer(open(
19      f'./logs/tafeln_aipp/'
20      f'{logname}_{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
```

```
        encoding='utf-8'), delimiter=';', lineterminator='\n',
        quoting=csv.QUOTE_ALL)

    i.writerow(['Dateipfad'] + ['Text'])


    def main():
        bildpfade_csv = eingabe()
        get_bildpfad(bildpfade_csv)
        print('\n' + 'Prozess abgeschlossen.')


    def eingabe():
        # Eingabe einer CSV-Datei mit Pfadliste
        print()
        while True:
            pfade_csv = input(
                'Bitte CSV-Datei mit Spalte "Bilder" angeben. '
                '(z.B. C:\\bilder_liste.csv): ')
            if os.path.exists(pfade_csv) and pfade_csv[2] == os.path.sep and \
                    re.search(r'^.+\.csv', pfade_csv):
                break
            elif not os.path.exists(pfade_csv):
                print('\nBitte Pfad zu existierender Datei eingeben.\n')
            else:
                print(
                    '\nBitte auf korrekte Schreibweise der Pfadangabe achten und'
                    ' CSV-Datei wählen.\n')
        print()
        return pfade_csv


    def get_bildpfad(pfade_csv):
        # Durchuche die CSV-Datei nach Dateipfaden.
        # Bei Bild: Übergabe an text_erkennen(bildpfad)
        with open(pfade_csv, encoding='utf-8') as liste:
            tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
            for row in tabelle:
                bildpfad = row['Bilder']
                # Angabe der Dateitypen (caseinsensitive)
```

```
61              if re.search(r'^.+\.jpg$|^.+\.jpeg$|^.+\.tif$|^.+\.tiff$', bildpfad,
62                           re.IGNORECASE):
63                  print(bildpfad)
64                  text_erkennen(bildpfad)
65
66
67  def text_erkennen(bildpfad):
68      # Öffne Bild, bereite Bild vor, erkenne Text, übergebe Text an Ausgabe
69      stream = open(str(bildpfad), 'rb')
70      bild_bytes = bytearray(stream.read())
71      numpyarray = np.asarray(bild_bytes, dtype=np.uint8)
72      bild = cv2.imdecode(numpyarray, cv2.IMREAD_GRAYSCALE)
73      # Bei Tafeln auf 0.4 skalieren
74      bild_klein = cv2.resize(bild, None, fx=0.4, fy=0.4,
75                              interpolation=cv2.INTER_AREA)
76      # histogram equalization
77      clahe = cv2.createCLAHE(clipLimit=3.0,
78                              tileGridSize=(8, 8))
79      cl1 = clahe.apply(bild_klein)
80      denoised = cv2.fastNlMeansDenoising(cl1)  # denoise
81      bild_neu = np.where((255 - denoised) < 50, 255, denoised + 50)  # aufhellen
82      bild_neu = Image.fromarray(bild_neu)
83      text = pytesseract.image_to_string(bild_neu, lang='deu')  # Text erkennen
84
85      ausgabe(bildpfad, text)
86
87
88  def ausgabe(bildpfad, text):
89      # Speichere erkannten Text
90      print('\n' + f'Bildpfad: {bildpfad}')
91      print('\n' + f'Text: \n\n{text}\n')
92      text = text.replace('\n', ' ')  # Text in einer Zeile ausgeben
93      i.writerow([bildpfad] + [text])
94
95
96  if __name__ == '__main__':
97      main()
```

XVI. bilder_vergleich_2px.py

```
"""Skript zum Abgleich zweier Pixel im Randbereich der jeweiligen Bilder.
Verwendet zum Scannen nach Zeitungsausschnitten.
"""

import csv
import cv2
from datetime import datetime
import numpy as np
import os
import re

logname = 'matches_zeitungungen_2px_aipp'  # Hier den Lognamen angeben.

os.makedirs('./logs/zeitungen_aipp', exist_ok=True)

a = open(
    f'./logs/zeitungen_aipp/{logname}_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.txt', 'w', encoding='ANSI')

a.write('Bilder' + '\n')

b = csv.writer(open(
    f'./logs/zeitungen_aipp/{logname}_'
    f'{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv', 'w',
    encoding='utf-8'), delimiter=';', lineterminator='\n')

b.writerow(['Bilder'])


def main():
    referenzbilder_csv = eingabe_referenz()
    bildersammlung_csv = eingabe_sammlung()
    startwerte = referenz(referenzbilder_csv)
    treffer = vergleich(bildersammlung_csv, startwerte)
    ausgabe(treffer)
    print('\nProzess abgeschlossen.')


```

```
39  def eingabe_referenz():
40      # Eingabe einer CSV-Datein mit Pfaden zu Referenzbildern
41      print()
42      while True:
43          referenzbilder_csv = input(
44              'Bitte Liste mit Refernz-Bildern angeben.'
45              ' (z.B. C:\\zeitungen.csv): ')
46          if os.path.exists(referenzbilder_csv) and referenzbilder_csv[2] == \
47                  os.path.sep and re.search(r'^.+\.csv', referenzbilder_csv):
48              break
49          elif not os.path.exists(referenzbilder_csv):
50              print('\nBitte Pfad zu existierender Datei eingeben.\n')
51          else:
52              print(
53                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten und'
54                  ' CSV-Datei wählen.\n')
55      return referenzbilder_csv
56
57
58  def eingabe_sammlung():
59      # Eingabe einer CSV-Datein mit Pfaden der zu durchsuchenden Verzeichnisse
60      while True:
61          bildersammlung_csv = input(
62              '\n' + 'Bitte Liste der Bild-Sammlung, die durchsucht werden soll, '
63                     'angeben. (z.B. C:\\kassierte_dateien_aipp.csv): ')
64          if os.path.exists(bildersammlung_csv) and bildersammlung_csv[2] == \
65                  os.path.sep and re.search(r'^.+\.csv', bildersammlung_csv):
66              break
67          elif not os.path.exists(bildersammlung_csv):
68              print('\nBitte Pfad zu existierender Datei eingeben.\n')
69          else:
70              print(
71                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten und'
72                  ' CSV-Datei wählen.\n')
73      print()
74      return bildersammlung_csv
75
76
77  def get_farbwerte(bildpfad):
78      # Liste Pixel-Werte zweier Pixel
```

```
79          stream = open(str(bildpfad), 'rb')
80          bild_bytes = bytearray(stream.read())
81          numpyarray = np.asarray(bild_bytes, dtype=np.uint8)
82          bild = cv2.imdecode(numpyarray, cv2.IMREAD_COLOR)
83          rgb = cv2.cvtColor(bild, cv2.COLOR_BGR2RGB)
84          hoehe, breite = bild.shape[:2]
85          px = rgb[hoehe - 10, breite - 10]  # px oben rechts
86          px = np.append(px, rgb[
87              hoehe - (hoehe - 10), breite - (breite - 10)])  # px unten links
88          return px
89
90
91      def referenz(referenzbilder, startwerte=None):
92          # Öffne Tabelle mit Pfaden zu Referenzbildern, speichere deren Pixel-Werte
93          if startwerte is None:
94              startwerte = []
95          print('\nStarte Sammeln der Referenzwerte.\n')
96          i = 1
97          with open(referenzbilder, encoding='utf-8') as liste:
98              tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
99              for zeile in tabelle:
100                 bildpfad = zeile['Bilder']
101                 px = get_farbwerte(bildpfad)
102                 print(f'{i}. Wert: {px}')
103                 i += 1
104                 # Füge Wert hinzu, wenn nicht schon vorhanden (inkl. Toleranz)
105                 for wert in range(len(startwerte)):
106                     if np.allclose(startwerte[wert], px, rtol=0.1):
107                         print('Wert schon vorhanden.\n')
108                         break
109                 else:
110                     startwerte.append(px)
111                     print('Wert hinzugefügt.\n')
112         print()
113         return startwerte
114
115
116     def vergleich(sammlung, startwerte, treffer=None):
117         # Öffne Tabelle mit Pfaden zu Bildern in Sammlung, vergleiche Farbwerte mit
118         # Referenzwerten
```

```
119     if treffer is None:
120         treffer = []
121     print('\nStarte Suche nach Übereinstimmungen.\n')
122     with open(sammlung, encoding='utf-8') as liste:
123         tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
124         for row in tabelle:
125             bildpfad = row['Bilder']
126             if os.path.exists(bildpfad):
127                 px = get_farbwerte(bildpfad)
128                 # Überprüfung auf Gleichheit (inkl. Toleranz)
129                 for wert in range(len(startwerte)):
130                     if np.allclose(startwerte[wert], px, rtol=0.1):
131                         if bildpfad not in treffer:
132                             treffer.append(bildpfad)
133                             print(bildpfad)
134             else:
135                 print('Bildpfad existiert nicht.')
136     return treffer
137
138
139 # Speichern der Treffer in CSV- und TXT-Dateien
140 def ausgabe(treffer):
141     for bildpfad in treffer:
142         b.writerow([bildpfad])
143         a.write(bildpfad + '\n')
144
145
146 if __name__ == '__main__':
147     main()
148
```

XVII. bilder_imagehash.py

```
"""Skript zur Generierung von Image-Hash-Werten. Sollte ursprünglich die
Suche nach Tafeln unterstützen. Die ausgegebene CSV-Datei kann in einer
SQL-Datbenbank nach entsprechenden Werten durchsucht werden. Die Hashwerte
ermöglichen jedoch lediglich die Suche nach Duplikaten, für Near-Duplicates
sind sie nicht genau genug. Das Skript fand daher im Workflow keine
Berücksichtigung.
"""

import csv
from datetime import datetime
import imagehash
import os
from PIL import Image
import re

# Auswahl: 'a','d' oder 'p' für average-, difference- oder perceptual-hash.
h_alg = 'd'

os.makedirs('./logs/bilder_hashes', exist_ok=True)

log_csv = csv.writer(open(
    f'./logs/bilder_hashes/'
    f'{h_alg}_imagehashes_{datetime.now().strftime("%d-%m-%Y_%H-%M-%S")}.csv',
    'w', encoding='utf-8'), delimiter=';', lineterminator='\n')

log_csv.writerow(['Dateipfad'] + ['Hash'])


def main():
    dateipfade_csv = eingabe()
    berechne_hashwerte(dateipfade_csv)
    print('\nProzess abgeschlossen.')


def eingabe():
    # Eingabe einer CSV-Datei mit Bildpfaden
    print()
    while True:
```

```
39          pfade_csv = input('Bitte CSV-Datei mit Bilderliste angeben. '
40                            '(z.B. C:\\bilder_liste.csv): ')
41          if os.path.exists(pfade_csv) and pfade_csv[2] == os.path.sep and \
42                  re.search(r'^.+\.csv', pfade_csv):
43              break
44          elif not os.path.exists(pfade_csv):
45              print('\nBitte Pfad zu existierender Datei eingeben.\n')
46          else:
47              print(
48                  '\nBitte auf korrekte Schreibweise der Pfadangabe achten und'
49                  ' CSV-Datei wählen.\n')
50      print()
51      return pfade_csv
52
53
54  # Berechne Hash-Werte, Ausgabe in CSV-Log
55  def berechne_hashwerte(pfade_csv):
56      with open(pfade_csv, encoding='utf-8') as liste:
57          tabelle = csv.DictReader(liste, delimiter=';', lineterminator='\n')
58          for zeile in tabelle:
59              bildpfad = zeile['Bilder']
60              print(bildpfad)
61              bild = Image.open(bildpfad)
62              if h_alg == 'a':
63                  # 8x8 = 64 / 4 = 16 hex digits = 64 bit
64                  hash_wert = (imagehash.average_hash(bild, hash_size=8))
65              elif h_alg == 'd':
66                  hash_wert = (imagehash.dhash(bild, hash_size=8))
67              elif h_alg == 'p':
68                  hash_wert = (imagehash.phash(bild, hash_size=8))
69              # print(j)
70              log_csv.writerow([bildpfad] + [hash_wert])
71
72
73  if __name__ == '__main__':
74      main()
75
```