

# Master-Thesis

## Anwendung bioinformatischer Methoden zur Datensuche in der IT-Forensik

Eingereicht am: 6. Juli 2019  
von: Bernd Beuermann,  
geboren am 13. Oktober 1973 in Herne;  
Studiengang IT-Sicherheit und Forensik  
Matrikelnummer: 202846

Betreuer: Prof. Dr.-Ing. Matthias Kreuseler  
weitere Gutachter: Prof. Dr.-Ing. Antje Raab-Düsterhöft

## **Vorwort und Danksagung**

Informationstechnik hat sich in den letzten Jahren zu einem festen Bestandteil einer jeden modernen Gesellschaft entwickelt. Gerade die Bereiche, deren Beeinträchtigung die Funktionsfähigkeit einer Gesellschaft stark beeinflussen kann, sind häufig von Informationstechnik abhängig. Hierzu gehören u. a. die Energieversorgung oder der medizinische Sektor. In der Folge werden auch immer mehr kriminelle Handlungen unter Nutzung der Informationstechnik begangen. Um dieser Entwicklung seitens der Strafverfolgung entgegenzutreten zu können, erscheint ein interdisziplinärer Ansatz in der Informatik erfolversprechend. Hierbei fand die Bioinformatik bisher kaum Beachtung.

Ich bin der Hochschule Wismar/WINGS zu besonderem Dank verpflichtet, die mir diese bioinformatischen Untersuchungen im Rahmen des Studiengangs IT-Sicherheit und Forensik ermöglicht hat.

## Aufgabenstellung

Aufgabe dieser Master-Thesis ist es, die forensische Datensuche mit Hilfe bioinformatischer Methoden durchzuführen und deren Eignung hierfür zu bewerten. Hierzu sind aus dem Portfolio der bioinformatischen Methoden möglichst geeignete auszuwählen. Die Anwendung dieser erfolgt mit Software, die die genutzten Algorithmen in Form von anwendungsbereiten Befehlen zur Verfügung stellen. Das Programm *Matlab* mit seiner *Bioinformatics Toolbox* ist für diesen Bereich ein bekanntes und anerkanntes Hilfsmittel. Für die Bewertung der verwendeten bioinformatischen Methoden werden die Wahrscheinlichkeiten für falsch-positive und falsch-negative Ergebnisse berechnet. Diese basieren auf Vergleichen von Dateien (Templates) mit forensisch relevantem Material eines Falles (Images). Die für die Untersuchungen erforderlichen Templates - im Schwerpunkt Bilddaten - entstammen frei verfügbar dem Internet, die Images werden speziell für die Untersuchungen erzeugt. Um den Herausforderungen in der IT-Forensik gerecht zu werden, sind unterschiedliche Arten von Images erforderlich. Auf der Grundlage von Images mit unveränderten Dateien und intaktem Filesystem kann die grundsätzliche Eignung der eingesetzten Methoden bewertet werden. Von besonderer Bedeutung sind veränderte Daten - z. B. durch das teilweise Überschreiben von Daten (Fragmente) oder die Datei-/Bildbearbeitung (Ähnlichkeit von Dateien) - mit keiner oder nur noch anteiliger Übereinstimmung mit dem Original. Nach der Untersuchung der grundsätzlichen Eignung folgen somit die Suche nach Fragmenten und die Suche nach ähnlichen Daten mit Hilfe der Bioinformatik. Eine Bewertung der Eignung im Sinne der Aufgabenstellung ist erst durch eine vergleichende Betrachtung mit etablierten Verfahren der IT-Forensik möglich. Die Wahl fällt unter Berücksichtigung der Verfügbarkeit für wissenschaftliche Untersuchungen (u. a. Studentenlizenz, kostenlose Lizenz für das Anfertigen von Abschlussarbeiten) und der Verbreitung in der IT-Forensik auf *The Sleuth Kit/Autopsy* (einschließlich *Scalpel*)<sup>1</sup>, *X-Ways* und *EnCase* zur Abdeckung der Forschungsbereiche „Filesystem“, „Hashing“ und „File Carving“ sowie *sdhash* für das „Similarity Hashing“. Die Ergebnisse sollen erste Hinweise geben und eine Ausgangsbasis für weitere, tiefergehende Untersuchungen darstellen, um möglicherweise künftig das Methodenportfolio der IT-Forensik um bioinformatische zu erweitern.

---

<sup>1</sup> Im weiteren Text werden zur Vereinfachung nur die Begriffe *Autopsy* und *Scalpel* verwendet.

## Kurzreferat

Die moderne Gesellschaft kann ohne den Einsatz von Informationstechnik nicht mehr existieren. Hierzu gehört leider auch der kriminelle Technikeinsatz. Die Beweisführung in der IT-Forensik gelingt nur über den Nachweis belastenden Materials - in der Regel Daten bzw. Dateien. Die Analyse erfolgt mit Hilfe spezieller Forensiksoftware, wobei die Vollständigkeit der Dateien (Metadaten und Inhalt) oder ein intaktes Filesystem für belastbare Analyseergebnisse grundsätzlich vorausgesetzt werden müssen. Für das Auffinden von erzeugten Dateifragmenten, die eine forensische Analyse deutlich erschweren bis unmöglich machen können, existieren keine Standardverfahren. Man findet allerdings eine gleichartige Herausforderung bei der Suche nach (veränderten) Gensequenzen in der Bioinformatik. Die Gensequenzen setzen sich aus Basen zusammen und die bioinformatische Vorgehensweise in Form einer Berechnung von Basenhäufigkeiten zum Auffinden relevanter Regionen und das Alignieren zweier Gensequenzen als ein mögliches Maß für deren Übereinstimmung (Alignmentscores) könnten auch in der IT-Forensik erfolgreich eingesetzt werden; dieser Sequenzvergleich in der Bioinformatik ist mit der forensischen Suche nach Templates bzw. Mustern in einem Image vergleichbar und bildet den Untersuchungsgegenstand der vorliegenden Arbeit.

Im ersten bioinformatischen Untersuchungsabschnitt wird das Auffinden von unveränderten Dateien in Images durch den Vergleich von Basenhäufigkeiten - Einzelbasen (1-Tupel) bis zu Basenquadrupel (4-Tupel) - untersucht. Auf dieser Basis sind die Identifizierung und Lokalisierung von Dateien möglich. Die Wahrscheinlichkeiten für falsch-negative (*FNR*) und falsch-positive Ergebnisse (*FPR*) sind von der Tupellänge abhängig, auch die Größe des Untersuchungsfensters, mit dem das Image und ggf. Template „abgefahren“ werden, beeinflusst die Analyse. Die Berechnung globaler und lokaler Alignmentscores im zweiten Untersuchungsabschnitt kann die *FNR* und *FPR* zusätzlich reduzieren. Im dritten Abschnitt wird das Vorgehen bei unveränderten Dateien auf Dateifragmente übertragen. Die Fragmentsuche ist erfolgreich durchführbar, wobei die Größe des Untersuchungsfensters in Abhängigkeit von der Fragmentgröße einen entscheidenden Faktor darstellt - eine exakte Größenübereinstimmung zwischen Untersuchungsfenster und Fragment ist jedoch nicht



erforderlich. Im vierten Untersuchungsabschnitt wird die Suche nach ähnlichen Daten, u. a. mit Hilfe der Erzeugung von Konsensussequenzen durch Multiple Alignments, betrachtet. Abschließend werden die bioinformatischen Ergebnisse mit etablierten Verfahren verglichen.

Bei der bioinformatischen Untersuchung unveränderter Dateien mit Hilfe von Basenhäufigkeiten gelingt der Ausschluss von im Image nicht vorhandenem Material mit einer  $FNR \leq 6\%$  (Basenquadrupel), unter zusätzlicher Anwendung von Alignments kann diese  $FNR$  auf  $0\%$  sinken. Im Vergleich hierzu liegen die Werte bei  $0\%$  für *Autopsy*, *X-Ways*, *EnCase* (Ergebnisse der Arbeit) und bei ca.  $5\%$  für *sddhash* (vgl. [35]). Der Einsatz bioinformatischer Methoden kann bei der Fragmentsuche vorteilhaft sein, da mit Hilfe etablierter Verfahren teilweise weniger oder im Extremfall sogar keine Fragmente auffindbar sind. Die Suche nach ähnlichen Dateien - vor allem unterschiedliche Versionen einer Datei, z. B. durch die Bildbearbeitung mit Filtern - ist grundsätzlich möglich, bedarf jedoch weiterer Untersuchungen zur Anwendung bioinformatischer Methoden. *Sddhash* als einziges zur Ähnlichkeitssuche eingesetztes etabliertes Verfahren konnte nicht überzeugen.

Bei intakten Filesystemen und unveränderten Daten sind die hier eingesetzten bioinformatischen Methoden hinsichtlich ihrer Zeit- und Platzkomplexität den etablierten Forensikverfahren deutlich unterlegen. Insgesamt bietet die Bioinformatik interessante Ansätze, die unabhängig vom Datenkontext - Filesystem und Datei-Metadaten - eine Imageanalyse ermöglichen. Auf Grund der vielfältigen Parametrisierungsmöglichkeiten bioinformatischer Methoden, z. B. Wahl des Alphabets und der Bewertungs-/ Substitutionsmatrix, sind weitere Untersuchungen erforderlich, um deren Nutzen für die IT-Forensik über die hier vorgestellten Ergebnisse hinaus zu beschreiben.

**Abstract**

Modern society can no longer exist without the use of information technology. Unfortunately, this includes the criminal use of technology. The argumentation in the IT forensics succeeds only on the evidence of incriminating material - usually data or files. The analysis is carried out with the help of special forensic software, whereby the completeness of the files (metadata and content) or an intact file system for reliable analysis results must be assumed in principle. There are no standard methods for finding generated file fragments, which can make forensic analysis significantly more difficult or even impossible. However, there is a similar challenge in the search for (altered) gene sequences in bioinformatics. The gene sequences are composed of bases and the bioinformatic approach in the form of a calculation of base frequencies to find relevant regions and the aligning of two gene sequences as a possible measure of their agreement (alignment scores) could be used successfully in IT forensics - this sequence comparison in bioinformatics is comparable with the forensic search for templates or patterns in an image and forms the object of investigation of the present work.

In the first bioinformatic investigation section, the detection of unchanged files in images is examined by comparing base frequencies - single bases (1-tuple) to base quadruples (4-tuples). On this basis, the identification and localization of files are possible. The probabilities for false-negative (*FNR*) and false-positive results (*FPR*) depend on the tuple length. The size of the window, with which the image and possibly the template are traversed, also influences the analysis. The calculation of global and local alignment scores in the second section can further reduce the *FNR* and *FPR*. In the third section the procedure for unchanged files is transferred to file fragments. The fragment search can be carried out successfully, whereby the size of the examination window depending on the fragment size is a decisive factor - an exact size match between the examination window and the fragment is not necessary. In the fourth part of the investigation the search for similar data is considered, amongst other things with the help of the generation of consensus sequences by multiple alignments. Finally, the bioinformatic results are compared with established methods.

In the bioinformatic examination of unchanged files with the help of base frequencies the exclusion of material not present in the image succeeds with an  $FNR \leq 6\%$  (base

quadruple), with the additional application of alignments this *FNR* can decrease to 0 %. In comparison, the values are 0 % for *Autopsy*, *X-Ways*, *EnCase* (results of the work) and about 5 % for *sdhash* (see [35]). The use of bioinformatic methods can be advantageous in the search for fragments, because with the aid of established methods in some cases fewer or in extreme cases even no fragments can be found. The search for similar files - especially different versions of a file, for example by image editing with filters - is possible in principle, but requires further investigation on the application of bioinformatic methods. *Sdhash* as the only established method for similarity search could not convince. In intact file systems and unchanged data, the bioinformatic methods used here are clearly inferior to established forensic methods in terms of their time and space complexity. Overall, bioinformatics offers interesting approaches, that independent of the data context - file system and file metadata - enable an image analysis. Due to the variety of parameterization options of bioinformatic methods, for example the choice of the alphabet and rating/substitution matrix, further research is needed to describe their benefits to IT forensics beyond the results presented here.

## Inhaltsverzeichnis

1	Einleitung .....	10
2	Problemstellung .....	13
2.1	Allgemeine Problematik.....	13
2.2	Spezielle Problematik - Bezug zur Bioinformatik .....	14
3	Ziel der Arbeit.....	18
4	Aktuelle Vorgehensweisen zur forensischen Datensuche.....	19
4.1	Filesystem .....	19
4.2	Hashing.....	20
4.3	File Carving.....	21
4.4	Hex-Editor.....	22
5	Bioinformatische Grundlagen (Alignments).....	23
6	Methodisches Vorgehen in der Arbeit.....	30
6.1	Vorüberlegungen .....	30
6.2	Grundlegende Vorgehensweise beim Vergleich von Image und Template.....	30
6.2.1	Transkription der Bitsequenz einer Datei in eine Basensequenz .....	33
6.2.2	Basenhäufigkeiten und orientierende Suche.....	34
6.2.3	Testung der orientierenden Suche in <i>Matlab</i> .....	43
6.2.4	Berechnung globaler und lokaler Alignments .....	46
6.3	Statistische Methoden zur Bewertung der Untersuchungsergebnisse .....	49
6.4	Überblick zur eingesetzten Software (Bioinformatik und etablierte Verfahren).....	52
6.5	Zusammenfassung des methodischen Vorgehens.....	52
7	Untersuchungsgegenstand „Bioinformatische Methoden“ .....	55
7.1	Orientierende Suche mit Hilfe von Basenhäufigkeiten - unveränderte Daten .....	55
7.1.1	Größe des Untersuchungsfensters ist gleich der Templatelänge.....	55
7.1.2	Auswirkungen der Fenstergröße auf das Untersuchungsergebnis .....	63
7.2	Alignments - unveränderte Daten .....	65
7.2.1	Globale Alignments.....	65
7.2.2	Lokale Alignments .....	70
7.2.3	Suche auf Basis globaler und lokaler Alignments .....	74
7.3	Untersuchung fragmentierter Daten.....	79
7.4	Untersuchung ähnlicher Daten (u. a. Multiple Alignments).....	84
7.4.1	Suche nach ähnlichen Dateien mit Hilfe von Basenhäufigkeiten .....	84
7.4.2	Suche nach ähnlichen Dateien auf Basis von Alignments.....	87
7.4.3	Suche nach ähnlichen Dateien - „Dateiversionen“ .....	89
7.4.4	Suche nach ähnlichen Dateien auf Basis von Alignments - „Dateiversionen“ .....	90
8	Vergleichende Betrachtung mit etablierten Verfahren .....	92
8.1	Vorüberlegungen, Voruntersuchungen .....	92
8.1.1	Filesystem .....	92
8.1.2	Hashing.....	92
8.1.3	File Carving.....	94
8.2	Untersuchung unveränderter Daten .....	97
8.3	Untersuchung fragmentierter Daten.....	99
8.4	Untersuchung ähnlicher Daten .....	101

---

9	Diskussion .....	103
9.1	Orientierende Suche mit Hilfe von Basenhäufigkeiten - unveränderte Daten .....	103
9.2	Alignments - unveränderte Daten .....	113
9.3	Untersuchung fragmentierter Daten .....	118
9.4	Untersuchung ähnlicher Daten .....	120
9.5	Vergleichende Betrachtung mit etablierten Verfahren .....	123
9.5.1	Unveränderte Daten .....	123
9.5.2	Fragmentierte Daten .....	123
9.5.3	Ähnliche Daten .....	127
9.6	Zusammenfassung in Bezug auf die Zielstellung .....	129
10	Zusammenfassung der Arbeit .....	132
11	Ausblick .....	135
	Literaturverzeichnis .....	137
	Bilderverzeichnis .....	140
	Tabellenverzeichnis .....	142
	Abkürzungsverzeichnis .....	144
	Anlagenverzeichnis und Anlagen .....	146
	Erklärung .....	192
	Thesen .....	193

## 1 Einleitung

Der Untersuchungsprozess in der IT-Forensik lässt sich aus Sicht des Bundesamtes für Sicherheit in der Informationstechnik (BSI) in unterschiedliche Abschnitte einteilen (vgl. [9], S. 24f und Bild 1). Beginnend mit der *Vorbereitung*, die einen strategischen und operationalen Anteil umfasst, folgt die *Datensammlung*. Hierbei ist eine Verfälschung der Daten unbedingt zu vermeiden. Die gesammelten Daten werden dann dem Abschnitt der *Untersuchung* zugeführt, der im Wesentlichen die Datenextraktion und Datenauswahl für die weiteren Prozessabschnitte umfasst. Es schließt sich die *Datenanalyse* mit der Zusammenführung der Ergebnisse (logisch und zeitlich) aus der Untersuchung und möglicherweise einer erneuten Durchführung vorangehender Prozessabschnitte an. Der letzte Abschnitt beinhaltet die Erstellung der Abschlussdokumentation, z. B. zur Vorlage und Präsentation in einem gerichtlichen Verfahren. Von jedem Prozessschritt aus kann es erforderlich sein, zu einem anderen, vorhergehenden Schritt zurückzukehren, z. B. offenbart die Untersuchung einen weiteren Datenbedarf (Datensammlung). Dieser Umstand ist im mittleren Bereich von Bild 1 dargestellt.

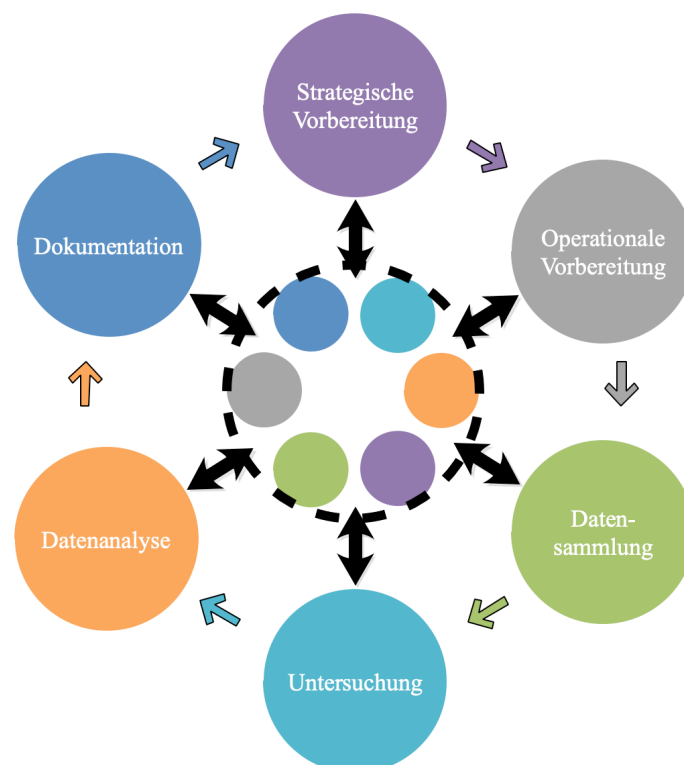


Bild 1: Forensischer Prozess, orientierend an den Vorgaben des BSI (vgl. [9], S. 24f)

Die Suche nach relevanten Daten im Abschnitt *Untersuchung* des forensischen Prozesses ist ein wichtiger Schritt. Aus den unbekannten, sichergestellten Daten werden die für den jeweiligen Fall relevanten Anteile extrahiert (Datensuche). Sowohl in flüchtigen als auch permanent gespeicherten Daten kann sich strafrechtlich bedeutendes Material befinden, z. B. verbotenes Bildmaterial gemäß § 184b Strafgesetzbuch (kinderpornographische Schriften). Die heute im privaten Bereich verwendeten Speichergrößen von bis zu mehreren Terabyte, mit zunehmender Tendenz, schließen eine nur manuell durchgeführte Datensuche und Datenanalyse in einer akzeptablen Zeit aus. Hinzu kommt, dass die im Jahr 2016 im Vergleich zu den beiden Vorjahren deutlich angestiegene Zahl der Cybercrime-Fälle (vgl. [11], [12], [13], [14]) einen hohen administrativen und damit personellen Aufwand bedingt, der kaum gedeckt werden kann (vgl. [2], [5], [18], [27], [45]). Vor dem Hintergrund wachsender Datenmengen und knapper personeller Ressourcen ist die Anwendung einer geeigneten IT-Unterstützung zur Datensuche und -analyse unabdingbar. Neben Softwareprodukten für spezielle Einsatzszenarien, z. B. Process Dumper<sup>2</sup> zur Erstellung von Speicherauszügen, gibt es IT-Forensiktools, die als Werkzeugsammlung einen breiten Anwendungsbereich von der Datensuche bis hin zur Datenanalyse und graphischen Aufbereitung abdecken. In diesem Zusammenhang häufig eingesetzte Softwareprodukte wie *Autopsy*, *X-Ways* oder *EnCase* bieten teilweise sehr gute Analyseergebnisse, setzen hierfür aber grundsätzlich eine intakte Dateistruktur voraus - u. a. Filesystem oder Datei-Header (vgl. [15], [17]). Durch eine fehlerhafte Dateistruktur, teilweise überschriebene Daten oder Datenmanipulationen können IT-forensische Untersuchungen stark behindert oder gar unmöglich gemacht werden (vgl. [47]). In diesem Zusammenhang kann davon ausgegangen werden, dass für die Strafverfolgungsbehörden das Auffinden strafrechtlich relevanten Materials mit dem Einsatz aktueller, professioneller IT-Werkzeuge künftig immer schwieriger wird. Somit könnte die Bedeutung von Verfahren zunehmen, die relevantes Material auch ohne bzw. ohne vollständige kontextbezogene Informationen wie die Metadaten in einem Dateihdr aufspüren können. In Foren wird teilweise gezielt nach Möglichkeiten gefragt, Festplatten oder Daten auf „Knopfdruck“ unbrauchbar bzw. nicht lesbar zu machen. Das Überschreiben der File Allocation Table bzw. Master File Table als

---

<sup>2</sup> Im Internet verfügbar unter: <https://trapkit.de/tools/pd/index.html> [letzter Zugriff: 5. Juli 2019].

wesentlicher Teil eines Filesystems bewirkt, dass ein Zugriff auf die Daten mit den Standardmitteln des Betriebssystems nicht mehr möglich ist. Auch das anteilige Überschreiben von Daten verhindert einen Dateizugriff. Sowohl Speicher- oder Festplattendefekte als auch bewusst durchgeführte Nutzeraktionen können Daten unbrauchbar machen. Im Falle der Sammlung strafrechtlichen Materials, u. a. pornographische Bilder, wird der Beschuldigte im Vorfeld polizeilicher Untersuchungen das Material sicherlich vernichten wollen - hierzu könnte er relevante Hardware physikalisch zerstören oder einen Löschalgorithmus starten (selbst erstelltes Skript). Auf Grund der in der Regel nicht angekündigten polizeilichen Maßnahmen, z. B. Hausdurchsuchung, dürfte eine physikalische Zerstörung schon aus Zeitgründen kaum erfolgreich sein. Auch eine vollständige Löschung der Daten ist wenig aussichtsreich. Es reicht das anteilige Löschen/Überschreiben aus, um eine Datei nur noch mit sehr großem Aufwand - wenn überhaupt - zu finden oder zu rekonstruieren. Im Rahmen der forensischen Datensuche sind auch nicht zugewiesene Speicherbereiche oder Slack Space mit Daten aus dem Hauptspeicher (vgl. [46], S. 226f) von Interesse.

Die oben genannten Aspekte der forensischen Datensuche werden im folgenden Abschnitt aufgenommen und die Motivation für den Einsatz bioinformatischer Algorithmen dargestellt. Es folgt die Definition der relevanten Fragestellungen, die mit Hilfe der Untersuchungen beantwortet werden sollen. In der Folge wird zunächst ein grober Überblick der etablierten Verfahren in der IT-Forensik zur Datensuche gegeben, um ein Verständnis hinsichtlich der Voraussetzungen für eine erfolgreiche Datensuche zu vermitteln und gleichzeitig Problembereiche aufzuzeigen, die eine Datensuche erschweren können. Es schließt sich ein Abschnitt zu den bioinformatischen Grundlagen an. Nach einer ausführlichen Darstellung der methodischen Vorgehensweise bildet das sich daran anschließende Kapitel mit den Untersuchungsergebnissen bei der Anwendung bioinformatischer Methoden den Schwerpunkt der Arbeit. Die Untersuchungen schließen mit einem Vergleich zu den Ergebnissen etablierter Verfahren ab. Beide Ansätze (Bioinformatik und etablierte IT-Forensiktools) werden diskutiert und in Bezug auf die Beantwortung der Fragestellungen bewertet.



## 2 Problemstellung

In den folgenden Abschnitten sollen die Herausforderungen bei der Datensuche in der IT-Forensik kurz skizziert werden. Auf dieser Basis wird der Vergleich zur Bioinformatik gezogen und damit gleichzeitig die Ausgangsbasis für die im Rahmen dieser Arbeit durchgeführten Untersuchungen gebildet.

### 2.1 Allgemeine Problematik

In der IT-Forensik werden sehr große Datenmengen gesammelt und mit Hilfe spezieller IT-Werkzeuge, u. a. Schreibschutzmodul und Software, in eine bitidentische Kopie (Image) zur weiteren Bearbeitung übertragen. Die Herausforderung besteht darin, in dem Bitmuster aus „0“ und „1“ alle bzw. alle für den jeweiligen Fall relevanten Dateien oder Daten zu identifizieren. Bei intaktem Filesystem oder vollständigen Dateihedern ist die Dateisuche mit Forensiktools wie *Autopsy*, *X-Ways* oder *EnCase* effektiv und effizient möglich (im Markt etablierte Verfahren). Zum Auffinden von existierenden oder gelöschten und nicht überschriebenen Dateien sind zunächst die Einträge in den Filesystem-Tabellen (z. B. Master File Table) mit den darin enthaltenen zugeordneten Speicherbereichen zu analysieren. Dateien in nicht zugewiesenen Bereichen können über eine Mustererkennung dateitypspezifischer Bytes ausfindig gemacht werden, sogenanntes File Carving. Um dies erfolgreich durchführen zu können, müssen diese Dateien grundsätzlich einen *Header* mit den typspezifischen Daten (Metadaten) und den zugehörigen *Footer* aufweisen. **Die Zuweisung von Speicherbereichen durch ein Dateisystem (Allozierung) und das Vorhandensein von Metadaten wird im Rahmen dieser Arbeit als Kontext verstanden.** Kontextfreie Daten können durch Löschvorgänge mit anschließenden Schreibaktivitäten entstehen. Zusätzlich sind Datenmanipulationen, die den strukturellen Aufbau einer Datei verändern, oder zufällige Datenveränderungen mit Integritätsverlust (vgl. [10], S. 490) nicht auszuschließen. Auch sind Datenveränderungen mit dem Erhalt der Dateistruktur - z. B. Speicherung der Originaldatei in einem anderen, proprietären Dateiformat - denkbar. Das Auffinden relevanten Materials in solchen Dateien mit intaktem, aber wenig bis gar nicht bekanntem Aufbau dürfte sehr schwierig sein. Ein Ansatz dieses Problem anzugehen, liegt in der Entwicklung geeigneter Hashing-Verfahren, die so „robust“ sind, dass auch unvollständige oder manipulierte Dateien erkannt werden (leicht veränderte oder ähnliche

Dateien liefern den gleichen Hashwert). Der Einsatz von sogenannten Fuzzy oder Byte-wise Hashing-Verfahren ist durchaus erfolgreich - als Beispiele seien *ssdeep*, *sdhash* oder *mvHash-B* genannt. Sie unterliegen jedoch Limitationen, das Erkennen semantisch identischer Dateien in unterschiedlichen Dateiformaten ist kaum möglich (vgl. [22]). Eine Hashfunktion soll idealerweise jede Änderung einer Eingabe auf eine andere Ausgabe abbilden, Kollisionen treten grundsätzlich nicht auf. Das Auftreten von Kollisionen kann in der IT-Forensik bei der Datensuche erwünscht sein, zum Aufdecken „ähnlicher“ Dateien. Hierbei bildet die Definition des Begriffs „Ähnlichkeit“ eine Herausforderung, ebenso die Implementierung einer in diesem Sinne geeigneten Hashfunktion.

Das Problem der Identifizierung relevanter kontextfreier (unbekannter) Daten durch den Abgleich mit bekannten Daten im Sinne einer Mustererkennung zu lösen, ist eine Kernaufgabe der Bioinformatik ([34], S. 38):

*„Der Vergleich von Sequenzen unbekannter Biomoleküle mit Sequenzen solcher Biomoleküle bekannter Funktion (und eventuell auch bekannter Struktur) ist ein zentrales Anliegen der Bioinformatik.“*

## 2.2 Spezielle Problematik - Bezug zur Bioinformatik

Der Aufbau einer Datei und die Repräsentation in einem Image sind in Bild 2 schematisch dargestellt.

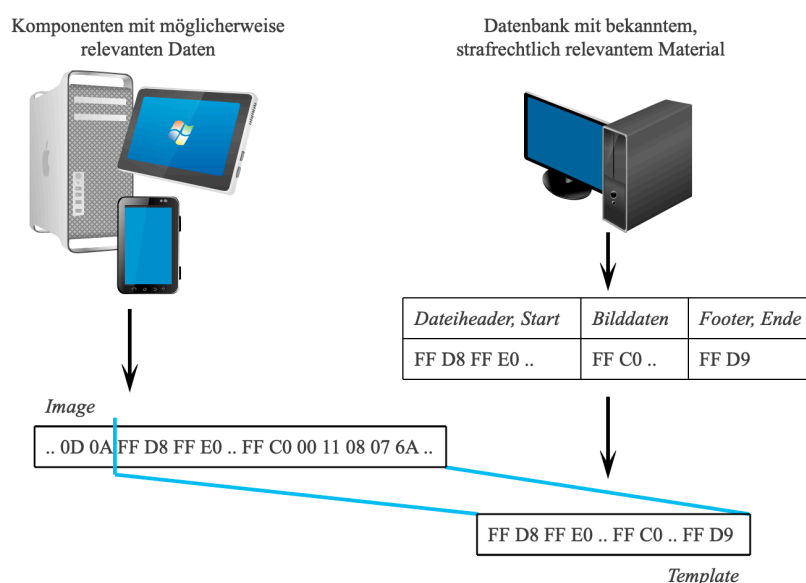


Bild 2: Zusammenhang zwischen Image- und Template-Bytemustern (IT-Forensik)

Dem Image steht ein Template (im Image zu identifizierendes Muster) gegenüber, wie es beispielsweise in Datenbanken der Strafverfolgung aus bereits bearbeiteten Fällen zu finden sein könnte. Dieser Sachverhalt ist für die Bioinformatik in Bild 3 skizziert.

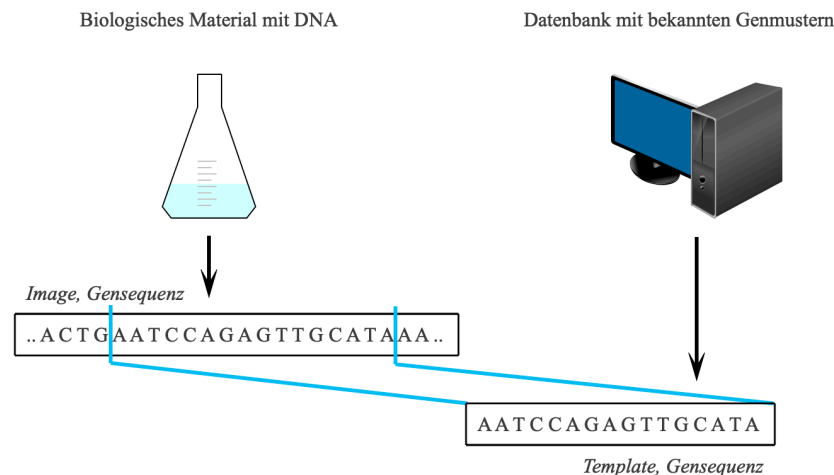


Bild 3: Zusammenhang zwischen Image- und Template-Sequenzmuster (Bioinformatik)

Das Auffinden von Sequenzmustern in der zu untersuchenden Gensequenz entspricht grundsätzlich der geschilderten Problematik in der IT-Forensik. Eine große unbekannte Datenmenge (Image) wird mit Einträgen einer Datenbank verglichen. In der Datenbank befindet sich strafrechtlich relevantes Material (Templates) - z. B. Texte, Bilder oder Tonaufnahmen. Die Image- und Template-Bytemuster sind mit Gensequenzen vergleichbar, die aus den Nukleinbasen Adenin (A), Thymin (T), Guanin (G) und Cytosin (C) als Bausteine der Desoxyribonukleinsäure (DNA) aufgebaut sind. Ziel ist es in beiden Fällen, eine Übereinstimmung bzw. Ähnlichkeit zwischen den Byte- bzw. Gensequenzen zu identifizieren. Der offensichtlich vergleichbaren Ausgangsproblematik bei der Daten-/Mustersuche in der IT-Forensik und Bioinformatik stehen auch Unterschiede gegenüber, die bei den folgenden Untersuchungen zu berücksichtigen sind. Eine grobe Übersicht zu den Gemeinsamkeiten und Unterschieden der IT-Forensik und Bioinformatik ist in Tabelle 1 zusammengefasst. Sie erhebt keinen Anspruch auf Vollständigkeit, zeigt aber auf, dass der wesentliche Unterschied auf verschiedene Alphabete zurückzuführen ist. Da evolutionäre Prozesse wie in der Biologie, z. B. Gen-/Musterveränderungen (Mutationen) in Verbindung mit einer bestimmbarer Wahrscheinlichkeit, für die Datenverarbeitung in der IT grundsätzlich keine Geltung besitzen, dürften die entsprechenden

bioinformatischen Methoden hier von untergeordneter Bedeutung sein. Das heißt, dass der Einsatz von Matrices mit Wahrscheinlichkeiten für Mutationen (u. a. Point Accepted Mutation [PAM] und Blocks Substitution Matrix [BLOSUM]) als wesentlicher Verfahrensbestandteil - wie bei den bioinformatischen Algorithmen FASTA und BLAST (vgl. [21], S. 376ff) - sowie beispielsweise die Betrachtung phylogenetischer Bäume (vgl. [4], S. 246ff) oder von Hidden-Markov-Modellen (vgl. [4], S. 216ff) in den Hintergrund treten.

Tabelle 1: Vergleich von IT-Forensik und Bioinformatik

Gemeinsamkeiten	Unterschiede
<ul style="list-style-type: none"> <li>eine große Datenmenge wird nach Mustern durchsucht</li> <li>es existieren spezielle Algorithmen oder Softwareprodukte zur Datensuche in Images/Suche in Gensequenzen</li> <li>es existieren Verfahren zur Untersuchung von Ähnlichkeiten</li> <li>große Datenmengen (Genom: ca. <math>3 \times 10^9</math> Basenpaare<sup>3</sup>, IT-Forensik: keine Größenbeschränkung bei Images und Templates)</li> </ul>	<ul style="list-style-type: none"> <li>Muster bestehen in der Bioinformatik aus den 4 Buchstaben A, C, G und T<sup>4</sup>  <math>\leftrightarrow</math> Muster in der IT-Forensik bestehen aus Bytes (bzw. einer Bitfolge) mit den „Buchstaben“ 0 und 1</li> <li>evolutionäre Prozesse spielen in der Bioinformatik eine große Rolle, hingegen gibt es sie in der IT-Forensik grundsätzlich nicht</li> </ul>

Zu den bioinformatischen Methoden, deren Verwendung hier geeignet erscheint, gehören die Sequenzanalyse auf Basis von Basenhäufigkeiten und die Berechnung von globalen und lokalen Alignments. Das Alignieren von Sequenzen beruht im Wesentlichen darauf, dass zwei Sequenzen durch das Einfügen und Verlängern von Lücken so verändert (ausgerichtet) werden, dass sie optimal übereinstimmen. Die Anzahl nicht übereinstimmender Sequenzanteile soll durch das Alignieren minimiert werden. Mit dem Alignieren erfolgt die Berechnung eines Alignmentsscores, der als Ausdruck der Übereinstimmung zweier Sequenzen gelten kann. Hinzu kommt die Anwendung Multipler Alignments, die die Eigenschaften mehrerer Sequenzen in einer sogenannten

<sup>3</sup> Bei einer Repräsentation jeder Base durch die ASCII-Zeichen „A“, „C“, „G“ und „T“ mit einem Speicherbedarf von jeweils 8 Bits (= 1 Byte) ergibt sich insgesamt ein Speicherbedarf von ca. 3 GB. Für weitere Informationen zum menschlichen Genom (Gensequenz als Gesamtheit der vererbaren Informationen) wird auf [28] verwiesen.

<sup>4</sup> Weitere spezifische Alphabete, z. B. für Aminosäuren mit 20 Einbuchstaben, sind nicht Gegenstand der Arbeit und werden nicht betrachtet (vgl. [31], S. 27ff).

Konsensussequenz abbilden können. Diese Konsensussequenz ihrerseits (Template) ist dann Bestandteil eines globalen oder lokalen Alignments als Abgleich zwischen Image und Template.

In diesem Zusammenhang scheint der Einsatz bioinformatischer Methoden in der IT-Forensik bisher insbesondere für das Aufdecken anormalen Verhaltens, z. B. im Sinne eines Intrusion Detection Systems, untersucht worden zu sein (vgl. [8], [24], S. 631ff und [30]). Hier steht vor allem die Analyse von Systemdateien, in denen die Aufrufe von System Calls des jeweiligen Betriebssystems abgelegt sind, im Vordergrund. Es werden Implementierungen von Alignmentalgorithmen verwendet, die die Dateiinhalte mit oder ohne Transkription in das Basenalphabet bzw. Aminosäuralphabet analysieren. Dateien mit einem anormalen Nutzerverhalten können so mit dem Inhalt „normaler“ Dateien verglichen (aligniert) werden. In Tabelle 2 ist ein orientierender Auszug aus [8] (Appendix B) für die Übersetzung von Dateiinhalten in eine Aminosäuresequenz dargestellt.

Tabelle 2: Transkription von System Calls in das Aminosäuren-Alphabet

<b>System Call Number</b>	<b>Amino Acid</b>	<b>FILE Function Name</b>
<b>3</b>	A (Alanin)	NtWriteFile
<b>5</b>	R (Arginin)	NtUnlockFile
<b>7</b>	C (Cystein)	NtSetInformationFile
<b>10</b>	D (Asparaginsäure)	NtReadFile
<b>11</b>	E (Glutaminsäure)	NtQueryVolumeInformationFile

### 3 Ziel der Arbeit

Vor dem Hintergrund des Einsatzes bioinformatischer Methoden bzw. Algorithmen zur Datensuche in der IT-Forensik ergibt sich als übergeordnete Fragestellung:

#### **Ist der Einsatz bioinformatischer Methoden zur Mustererkennung und damit Datensuche in der IT-Forensik geeignet?**

Für die Beantwortung dieser übergeordneten Fragestellung ergeben sich folgende untergeordnete Fragestellungen:

1. Kann die Analyse von Basenhäufigkeiten zur Suche nach (unveränderten) Dateien eingesetzt werden?
2. Kann die Analyse von Basenhäufigkeiten zur Suche nach Dateifragmenten eingesetzt werden?
3. Bedarf es für die Anwendung bioinformatischer Methoden ggf. einer Vorverarbeitung der Eingabedaten?
4. Können bioinformatische Methoden zur Erzeugung von globalen und lokalen Alignments erfolgreich im Rahmen der Datensuche eingesetzt werden?
5. Können Multiple Alignments zur Suche nach ähnlichen Dateien eingesetzt werden?
6. Sind die Ergebnisse beim Einsatz bioinformatischer Algorithmen mit denen etablierter Forensiktools, z. B. *Autopsy*, vergleichbar?

## 4 Aktuelle Vorgehensweisen zur forensischen Datensuche

Bevor auf die bioinformatischen Besonderheiten und das grundsätzliche Vorgehen in der Arbeit zur Beantwortung der oben aufgeführten Fragen eingegangen wird, erfolgt an dieser Stelle ein kurzer Überblick der Standard-Vorgehensweisen zur Datensuche in der IT-Forensik.

### 4.1 Filesystem

In der IT-Forensik wird man die Datensuche mit der Identifizierung eines Datei-/Filesystems und dessen Analyse beginnen. Die wesentlichen Aspekte der hier verwendeten Dateisysteme unter Windows sollen hinsichtlich ihrer Relevanz für die Untersuchungen kurz vorgestellt werden. Sehr verbreitet, insbesondere bei mobilen Datenträgern, ist das Dateisystem File Allocation Table (FAT), das in seiner Dateizuordnungstabelle für jedes Cluster<sup>5</sup> (Zuordnungseinheit) des Speichermediums einen Eintrag enthält - Cluster können belegt, frei oder beschädigt sein (vgl. [1], S. 117ff). Jedes Cluster enthält die Adresse der nächsten Zuordnungseinheit, die bei der Dateispeicherung nicht unbedingt zusammenhängend oder in einer bestimmten Reihenfolge vergeben werden muss. Die Cluster lassen sich in einen oder mehrere Sektoren unterteilen. Legt das Betriebssystem Dateien auf dem Speichermedium ab, so werden immer ganze Cluster zugewiesen (alloziert), auch wenn die Dateigröße nur einen Teil des letzten Clusters belegt (Bild 4).

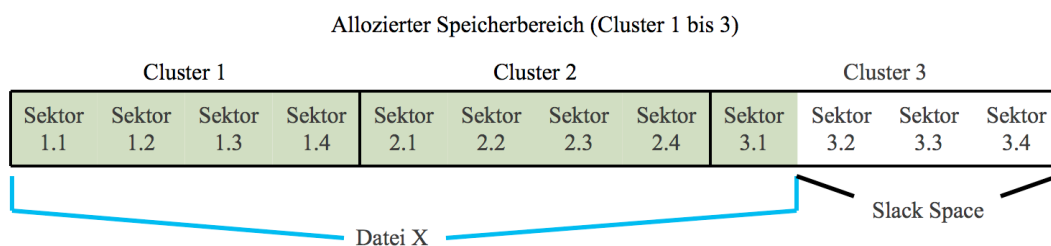


Bild 4: Speicherzuweisung, Cluster und Slack Space

<sup>5</sup> Häufig findet man den Begriff „Block“ als Synonym für „Cluster“. Der Begriff „Block“ steht hier auch für die Verwendung des Untersuchungsfensters, das die Dateien in ihrer Länge nicht vollständig umfasst, sondern entsprechend seiner Größe block- oder abschnittsweise abfährt.

Durch die Datei nicht genutzte Clusteranteile werden als Slack Space bezeichnet<sup>6</sup>, der nicht selten durch das Betriebssystem mit Hauptspeicherinhalt gefüllt wird (Bild 4; vgl. [19], S. 109ff). Im Gegensatz zur FAT steht in der Master File Table (MFT) des New Technology File Systems (NTFS) jeder Verzeichniseintrag für eine Datei und enthält u. a. die Datei-Metadaten und Informationen über die zu einer Datei gehörenden Cluster. In Abhängigkeit von der Größe können auch komplette Dateien in der MFT abgelegt werden. Die Untersuchung des Filesystems ist in den hier eingesetzten etablierten Verfahren *Autopsy*, *X-Ways* und *EnCase* implementiert.

## 4.2 Hashing

Die Grundidee beim Hashing besteht darin, dass beliebig lange Eingabedaten (Dateien) durch eine Hashing-Funktion auf eine Ausgabe fester Länge, in den meisten Fällen deutlich kleiner als die Länge der Eingabedaten, im Sinne eines spezifischen Dateiprofils abgebildet werden. Hierdurch lassen sich große Dateidatenbanken mit einem vergleichsweise geringen Speicheraufwand verwalten. Die in der IT standardmäßig eingesetzten kryptographischen Hashing-Funktionen stellen eine „Einbahnstraße“ dar; das heißt, von dem Ausgabewert kann idealerweise nicht auf die Eingabedaten geschlossen werden. Zusätzlich führt jede Bitänderung in der Eingabe zu einem anderen Ausgabewert, es treten idealerweise keine Kollisionen auf. Eine Übereinstimmung der Hashingwerte zwischen Image und Template ist jedoch nur möglich, wenn das Template unverändert im Image vorhanden ist und exakt die gesuchte Datei (Template) als Eingabe für die Hashing-Funktion genutzt wird. Von besonderem Interesse ist das Auffinden ähnlichen Materials, da das in einer Datenbank gespeicherte strafrechtlich relevante Material im Image in leicht veränderter Form vorhanden sein könnte, bedingt z. B. durch eine Nutzerbearbeitung oder die Speicherung in einem anderen Dateiformat als das Original. Die kryptographischen Hashing-Verfahren lassen Kollisionen grundsätzlich nicht zu, so dass die Datensuche erfolglos bliebe. Es sind also spezielle Hashing-Algorithmen erforderlich, die im Falle ähnlicher Dateien Kollisionen zulassen. Solche Algorithmen sind beispielsweise in *sdhash* [36] oder *ssdeep* (vgl. [7] und [26]) implementiert. Diese Verfahren können zur Datensuche in der IT-Forensik eingesetzt werden, nicht selten kommt es zu falsch-positiven Ergebnissen (vgl. [6]). Das Image wird

---

<sup>6</sup> Die weitere Unterteilung in RAM Slack, Drive Slack, File Slack oder MFT Slack erfolgt hier nicht.



in Blöcke eingeteilt, deren jeweiliger Hashwert mit dem Hashwert des Templates verglichen wird. Der errechnete Score ist Ausdruck der Ähnlichkeit zwischen den Hashwerten und gibt die Wahrscheinlichkeit für das Vorhandensein des Templates bzw. von Templatefragmenten im Image an. Für *sdfhash* ergibt sich beispielsweise folgende Einteilung (vgl. [36], S. 13):

- 21 - 100: sehr hohe Wahrscheinlichkeit; hohe Reliabilität; je größer der Score, desto geringer ist die Wahrscheinlichkeit für falsch-positive Ergebnisse
- 11 - 20: marginale Wahrscheinlichkeit
- 1 - 10: sehr geringe Wahrscheinlichkeit; häufig falsch-positive Ergebnisse
- 0: keine Übereinstimmung erkennbar

Das Hashing-Tool *ssdeep* scheint hier nicht geeignet, da bei der Ausgabe nur vollständige Dateien angegeben werden (vgl. [42]). Bei der Suche nach Dateifragmenten ist die Fragmentgröße als unbekannt vorauszusetzen. Damit würde jedes zu untersuchende Image weitere Verarbeitungsschritte bedingen, um die Lokalisation des Templates im Image einzugrenzen. Hingegen kann *sdfhash* die einzelnen Segmente mit ihren Scores ausgeben und ermöglicht so direkt eine Dateilokalisation (vgl. [36], S. 15f). Die Untersuchung von Hashwerten ist in den hier eingesetzten etablierten Verfahren *Autopsy*, *X-Ways* und *EnCase* implementiert. Die Berechnung und der Vergleich von block- bzw. segmentweise gehashten Dateien erfolgt für die folgenden Untersuchungen mit *X-Ways* bzw. *sdfhash*.

### 4.3 File Carving

Beim File Carving wird der Bytecode nach prominenten Stellen durchsucht. Jedes Dateiformat besitzt eine spezielle Byte-Signatur, die den Dateibeginn markiert. Es folgen die Metainformationen im Dateiheder, der eigentliche Dateiinhalt wird in der Regel durch Ende-Markierungen des *Headers* und der Datei (*Footer*) eingegrenzt. Für eine Datei im *jpg*-Format ergibt sich folgende Byte-Signatur (Auswahl; vgl. [40], S. 210ff):

- Start of Image: **FF D8**
- Metadaten: ... (verschiedene Segmente)
- Start of Scan: **FF DA** (Inhaltsdaten)
- End of File: **FF D9**

Fragmente einer Datei ohne Dateihheader (Metainformationen) können durch das File Carving grundsätzlich nicht gefunden werden. Erfolgt das File Carving rückwärts, also vom *Footer* aus, kann unter der Annahme, dass der nächste gefundene *Footer* das Ende der vorangehenden Datei und damit den Anfang der zu rekonstruierenden Datei markiert, möglicherweise relevantes Dateimaterial aufgefunden werden. Das File Carving ist in den hier eingesetzten etablierten Verfahren *Scalpel*, *X-Ways* und *EnCase* implementiert.

#### 4.4 Hex-Editor

Mit Hilfe eines Hex-Editors können Dateien in ihrer Byte-Struktur als Hexadezimalziffern dargestellt werden. Häufig stellen diese Softwaretools den Bytecode (linker Bereich in Bild 5) und die abgeleiteten ASCII-Zeichen (rechter Bereich in Bild 5) dar.

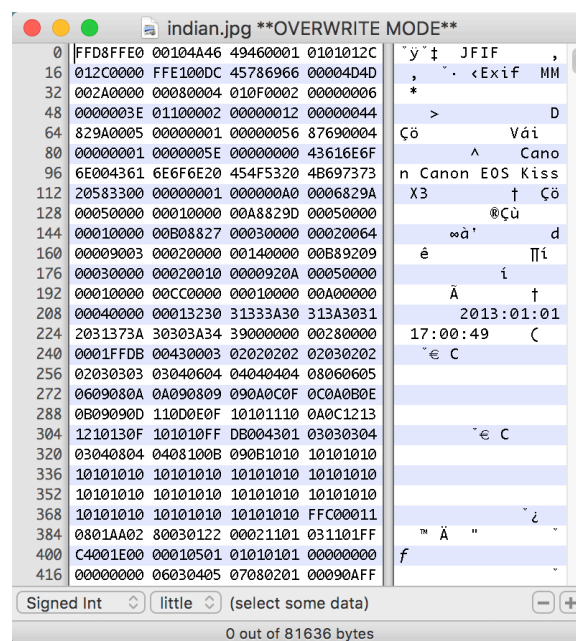


Bild 5: Auszug der Byte-Struktur von *indian.jpg* (*iHex*)

Sie ermöglichen die Suche nach Bytemustern, Suchbegriffen und die Manipulation der einzelnen Bytepositionen. In Bild 5 ist ein Auszug der Datei *indian.jpg* unter Verwendung von *iHex* zu sehen. Der Hex-Editor *iHex* kommt im Rahmen der Arbeit zum manuellen Überschreiben von Daten eines Images zur späteren Fragmentsuche zum Einsatz. Des Weiteren zum Befüllen von „freien“, mit Nullen gefüllten Speicherbereichen, z. B. Slack Space.

## 5 Bioinformatische Grundlagen (Alignments)

Nachdem die wesentlichen Aspekte etablierter Vorgehensweisen kurz dargestellt wurden, folgen in diesem Abschnitt bioinformatische Grundlagen zum Verständnis der folgenden Untersuchungen. An die Ausführungen in Kapitel 2 anknüpfend spielt für die Mustererkennung in der Bioinformatik die Ermittlung von **Häufigkeiten** eine wichtige Rolle. Ihre Verwendung zur Datensuche - wie in der vorliegenden Arbeit - folgt dem auch aus anderen Disziplinen bekannten Ansatz, dass Elemente (hier Basen bzw. Basentupel) gezählt werden. Die spezifische Anpassung der Zählweise für die folgenden Untersuchungen ist im folgenden Kapitel beschrieben und soll hier nicht weiter vertieft werden. In diesem Kapitel wird die Berechnung von **Alignments** näher beschrieben, da sie eine Kernaufgabe der Bioinformatik mit dem Vergleich von (mindestens) zwei Sequenzen in Bezug auf ihre Übereinstimmung oder Ähnlichkeit bildet. Ausdruck dieser Übereinstimmung ist der sogenannte Alignmentsscore. Die Sequenzen oder Sequenzbestandteile werden so übereinandergelegt und durch das Einfügen von Lücken verschoben, dass sich ein optimaler Score ergibt. Trotz dieser Maßnahmen muss nicht jedes Basenpaar der beiden Sequenzen die gleiche Base aufweisen. Zur Berechnung der Sequenzähnlichkeit kommen Scoring- oder Substitutionsmatrices zum Einsatz, um ungleiche Basenpaare geeignet in einen Alignment- oder Ähnlichkeitsscore einfließen zu lassen (vgl. [23], S. 163ff und [34], S. 38ff). Das Ergebnis eines Sequenzvergleiches ergibt sich beispielsweise wie folgt (hier sind keine Lücken vorhanden):

```

A A C G T G G A C G G C C T A T T A G C
|  | |   | | | |  |   |  | | | |
A C C G C A G A C G T C A G A A T A G C

```

Der Score für das oben aufgeführte Beispiel soll hier sechs betragen, 13 Übereinstimmungen und sieben nicht übereinstimmenden Basen ( $13 - 7 = 6$ ).

Eine informationstechnische Unterstützung bei der Berechnung von Alignments bietet das im Rahmen der Arbeit eingesetzte Softwareprodukt *Matlab*. Die entsprechende Eingabe - zur Verdeutlichung der erforderlichen Alignmentparameter - für das oben gezeigte Beispiel lautet:

```

Sequenz_1 = 'AACGTGGACGGCCTATTAGC';
Sequenz_2 = 'ACCGCAGACGTCAGAATAGC';
scoring_matrix = [1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1];
[score, alignment, startat] = nwalign(Sequenz_1, Sequenz_2, 'ScoringMatrix',
    scoring_matrix, 'gapopen', 1, 'extendgap', 1, 'alphabet', 'nt');

```

Die Scoring Matrix (scoring\_matrix) spannt eine vier mal vier große Tabelle auf, in der jeder Spalte und Zeile eine Base zugeordnet ist. In jede Zelle wird ein Wert eingetragen, der bei der entsprechenden Basenkombination für die Berechnung verwendet werden soll. So wird im oben aufgeführten Beispiel die Übereinstimmung von Basen mit 1 bewertet, die Nichtübereinstimmung mit -1 (die einzelnen Zeilen der Matrix sind mit einem Semikolon getrennt, vgl. auch Bild 6). Die Verwendung der Scoring Matrix ist in Bild 6 vertiefend erläutert. In der Matrix kann für jedes Basenpaar der in den Alignmentsscore einfließende Wert abgelesen werden. Der für das Eröffnen (Gap Open) und Erweitern einer Lücke (Extend Gap) angegebene Wert stellt eine Strafe (Penalty) dar und geht in die Berechnung entsprechend als negativer Wert ein. Auf der rechten Seite in Bild 6 sind zwei alignierte Sequenzen mit einem Score von zwei dargestellt. Eine genaue Beschreibung des Vorgehens beim Alignieren folgt weiter unten.

	<b>A</b>	<b>C</b>	<b>G</b>	<b>T</b>	Gap Open = 1, Extend Gap = 1								
<b>A</b>	<b>1</b>	-1	-1	-1	A	C	C	G	T	A	T	A	Sequenz 1
<b>C</b>	-1	<b>1</b>	-1	-1									
<b>G</b>	-1	-1	<b>1</b>	-1	A	G	C	A	T	A	-	A	Sequenz 2
<b>T</b>	-1	-1	-1	<b>1</b>	1	-1	1	-1	1	1	-1	1	= 2

Bild 6: Anwendung einer Scoring Matrix

Eine Anpassung der Scoring Matrix (z. B. scoring\_matrix = [1 -2 -1 -1; -1 1 -2 -1; -1 -1 -2; -1 -1 -1 1];) ergibt für das bereits genannte Beispiel mit dem ursprünglichen Score von sechs folgendes Alignment (jetzt ergibt sich ein Score von fünf bei 13 Übereinstimmungen und acht nicht übereinstimmenden Basen bzw. vorhandenen Lücken):

```

A A C G T G G A C G G C - C T A T T A G C
|   | |   | | | |   |   | |   | |
A C C G C A G A C G T C A G A A T - A G C

```

Eine Veränderung der Scoring Matrix beeinflusst also den Alignmentsscore. In der vorliegenden Arbeit wird die Scoring Matrix  $[1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1]$  eingesetzt; entscheidend ist die Verwendung einer Matrix, um den Vergleich der Untersuchungsergebnisse zu ermöglichen. Ebenso sind Lücken einheitlich zu bewerten (hier: Gap Open = 1 und Extend Gap = 1).

Bei den zuvor gezeigten Alignments handelt es sich um **globale Alignments**, die mit dem Algorithmus nach Needleman und Wunsch erstellt wurden (vgl. [31], S. 162ff, [33] und [34], S. 50ff). Hierbei erfolgt das Alignment in beiden Sequenzen ab der ersten Basenposition und versucht, die beiden vollständigen Sequenzen in Übereinstimmung zu bringen. Die Vorgehensweise bei der Ermittlung des optimalen globalen Alignments nach Needleman und Wunsch soll kurz beschrieben werden. Die beiden zu vergleichenden Sequenzen spannen, wie in Bild 7 zu sehen, eine Matrix auf. In die zweite Zeile bzw. Spalte trägt man mit Null beginnend den Wert (Penalty) für das Einfügen einer Lücke (Gap) ein. Beispielsweise wird das Verbleiben in einer Zeile für jede Spalte mit einem „Strafwert“ in Höhe von 1, also -1, bewertet. Ist die Matrix mit den Sequenzen und Gap-Werten initialisiert, beginnt man die einzelnen Zellen mit Werten für den Sequenzvergleich zu füllen. Für jede Zelle  $Z_{i,j}$  sind zunächst ausgehend von  $Z_{i-1,j-1}$  drei Werte zu berechnen:

1. Das Einfügen einer Lücke in die vertikale Sequenz (Verbleib in der gleichen Zeile, eine Spalte vorrücken)  $\rightarrow Z_{i,j-1} + \text{Gap-Wert}$ .
2. Das Einfügen einer Lücke in die horizontale Sequenz (Verbleib in der gleichen Spalte, eine Zeile nach unten)  $\rightarrow Z_{i-1,j} + \text{Gap-Wert}$ .
3. Match oder Mismatch der beiden Nukleinbasen in den Sequenzen (gemäß der Scoring Matrix)  $\rightarrow Z_{i-1,j-1} + \text{Score Match/Mismatch gemäß Scoring Matrix}$ .

Der Zellenwert  $Z_{i,j}^{global}$  ergibt sich als Maximum der drei zuvor errechneten Werte:

$$Z_{i,j}^{global} = \max \begin{cases} Z_{i,j-1} + \text{Gap-Wert} \\ Z_{i-1,j} + \text{Gap-Wert} \\ Z_{i-1,j-1} + \text{Score Match|Mismatch} \end{cases} \quad (5.1)$$

Für den ersten Tabellenwert in Bild 7 (Zeile 3, Spalte 3) ergeben sich somit folgende Werte: -1 für das Einfügen einer Lücke in die vertikale Sequenz, -1 für das Einfügen einer Lücke in die horizontale Sequenz und +1 für den Match zwischen „C“ und „C“. Das Maximum mit +1 wird eingetragen. Bewertet man das Einfügen und Erweitern von Lücken mit 1, den Mismatch mit -1 und den Match mit +1 ergibt sich für das Beispiel insgesamt die in Bild 7 befüllte Matrix.

		C	T	A	G
	0	-1	-2	-3	-4
C	-1	+1	0	-1	-2
G	-2	0	0	-1	0
A	-3	-1	-1	+1	0
T	-4	-2	0	0	0

Optimales globales Alignment:

C	T	A	G	
C	G	A	T	Score: 0

Bild 7: Berechnung globaler Alignments (das optimale Alignment erfordert keine Lücken)

Anschließend wird ein Backtracking durchgeführt, der Pfad von rechts unten (Alignmentscore) beginnend nach links oben eingezeichnet. Es wird diejenige der drei Zellen ausgewählt, deren Wert zum Maximum geführt hat. Manchmal sind mehrere Pfade möglich, die Auswahl obliegt der jeweiligen Implementierung des Algorithmus. In einem letzten Schritt werden beide Sequenzen entlang des Pfades von links oben nach rechts unten untereinander geschrieben. Horizontale bzw. vertikale Pfeile bedeuten das Einfügen einer Lücke in die vertikale (untere) bzw. horizontale (obere) Sequenz. Übereinstimmende Nukleinbasen werden zur besseren Übersicht zusätzlich durch einen Balken verbunden. In Bild 8 ist zusätzlich die Berechnung eines globalen Alignments mit und ohne das Einfügen von Lücken dargestellt, um das Auffinden des optimalen Alignments zu verdeutlichen. Das Alignment ohne Lücken entspricht damit nicht dem Algorithmus nach Needleman und Wunsch, es soll ausschließlich die Bedeutung von Lücken für das optimale Alignment verdeutlichen. Im unteren Bildteil ist nur der blaue Pfad als Alignment dargestellt. Mit den grauen Linien wird ein zusätzlicher, alternativer Pfad markiert. Das entsprechende Alignment mit dem gleichen Score ist in Bild 8 nicht aufgeführt.

		C	A	T	T
	0	-1	-2	-3	-4
C	-1	+1	0	-1	-2
G	-2	0	0	-1	-2
A	-3	-1	+1	0	-1
T	-4	-2	0	+2	+1

Globales Alignment ohne Lücken:  
(Beispiel, entspricht nicht dem Algorithmus nach Needleman und Wunsch)

C A T T

| |

C G A T

Score: 0

		C	A	T	T
	0	-1	-2	-3	-4
C	-1	+1	0	-1	-2
G	-2	0	0	-1	-2
A	-3	-1	+1	0	-1
T	-4	-2	0	+2	+1

Optimales globales Alignment:

C - A T T

| | |

C G A T -

Score: 1

Bild 8: Berechnung globaler Alignments (das optimale Alignment erfordert das Einfügen von Lücken)

Die Berechnung **lokaler Alignments** mit Hilfe des Algorithmus nach Smith und Waterman (vgl. [31], S. 177ff und [39]) erfolgt grundsätzlich nach dem gleichen Schema wie in Bild 8 für das globale Alignment gezeigt. Im Vordergrund stehen hier allerdings Ähnlichkeiten von Teilsequenzen. Lokale Alignments entsprechen zunächst dem Problem der längsten gemeinsamen Teilsequenz („Longest Common Subsequence“; vgl. [38], S. 111ff). Um zwischen den zu alignierenden Sequenzen die längste gemeinsame Teilsequenz zu finden, werden die Scoring-Werte und die Bewertung von Lücken angepasst:  $\text{scoring\_matrix} = [1\ 0\ 0\ 0; 0\ 1\ 0\ 0; 0\ 0\ 1\ 0; 0\ 0\ 0\ 1]$ , Gap Open = 0, Extend Gap = 0. Der Zellenwert  $Z_{i,j}^{local}$  ergibt sich wie folgt:

$$Z_{i,j}^{local} = \max \begin{cases} Z_{i,j-1} + \text{Gap-Wert} \\ Z_{i-1,j} + \text{Gap-Wert} \\ Z_{i-1,j-1} + \text{Score Match|Mismatch} \\ 0 \text{ sonst.} \end{cases} \quad (5.2)$$

Im Gegensatz zum Backtracking bei globalen Alignments beginnend ab der rechten unteren Zelle, wird bei lokalen Alignments das Maximum in der Matrix bestimmt und

davon ausgehend das Alignment ermittelt. Dieses Vorgehen ist in Bild 9 skizziert. Es können mehrere lokalen Alignments mit dem gleichen Maximalwert existieren.

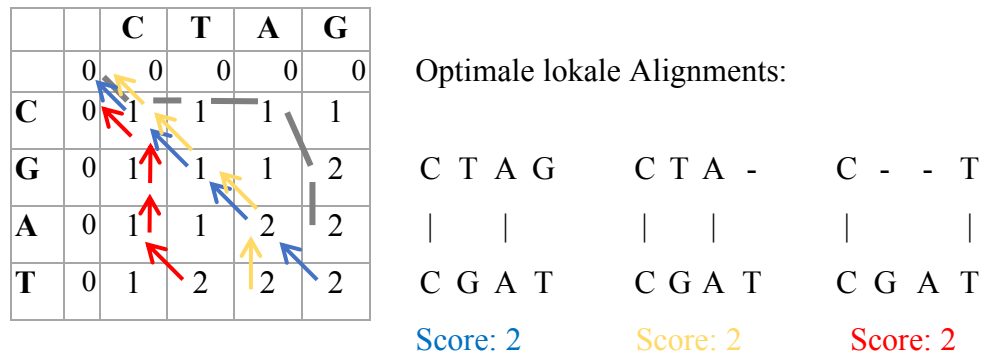


Bild 9: Berechnung von lokalen Alignments

In Bild 9 sind beispielhaft drei lokale Alignments dargestellt (der graue Pfad markiert beispielhaft ein weiteres lokales Alignment mit dem Start in der dritten Zeile und letzten Spalte); die Berechnung der Zellenwerte orientiert sich an dem Vorgehen bei globalen Alignments, wobei in der Matrix keine negativen Werte zu finden sind (der kleinste Wert ist null). In *Matlab* wird mit dem Befehl *showalignment()* das lokale Alignment für den roten Pfad visualisiert.

Abweichend von der oben dargestellten Scoring Matrix für lokale Alignments **kommen in der vorliegenden Arbeit für alle Alignments die Scoring Matrix [1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1] (Bild 6) und die Bewertung von Lücken mit eins zum Einsatz.**

Die bisherigen Verfahren zum Erstellen globaler bzw. lokaler Alignments vergleichen zwei Sequenzen hinsichtlich ihrer Ähnlichkeit. Ebenso ist es mit Hilfe des **Multiplen Alignments** möglich, mehr als zwei Sequenzen in eine Konsensussequenz zu überführen. Diese eine Sequenz soll die Ähnlichkeitsmerkmale mehrerer Sequenzen vereinen und kann als Template zum Abgleich mit einem Image verwendet werden. Für die Datensuche in der IT-Forensik bedeutet dies, dass möglicherweise der Konsensus ähnlicher Dateien/Bilder im Sinne einer semantischen Suche eingesetzt werden könnte. Für die Sequenzen „CATTGGATAG“, „CGATAATGCTAG“ und „CTAGTAGTAG“ ergibt sich mit dem *Matlab*-Befehl für Multiple Alignments *multialign()* das in Bild 10 gezeigte Alignment. Die Konsensussequenz kann eine größere Basenanzahl aufweisen als die längste zur Erzeugung des Multiplen Alignments verwendete Sequenz.



---

C	A	T	T	G	-	-	-	-	-	G	A	T	A	G	Sequenz 1
C	-	-	-	G	A	T	A	A	T	G	C	T	A	G	Sequenz 2
C	-	-	-	-	-	T	A	G	T	A	G	T	A	G	Sequenz 3

---

Konsensussequenz:

C A T T G A T A G T G C T A G

Bild 10: Beispiel für ein Multiples Alignment mit Konsensussequenz

Die Konsensussequenz wird aus dem Multiplen Alignment durch *seqconsensus()* in *Matlab* erzeugt. Das Scoring von Matches, Mismatches und Lücken findet analog zur Berechnung der Alignments statt:

`scoring_matrix = [1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1],`

Gap Open = 1, Extend Gap = 1).

## **6 Methodisches Vorgehen in der Arbeit**

Nach den Ausführungen in den beiden vorhergehenden Kapiteln zu den etablierten Vorgehensweisen und bioinformatischen Methoden in Form der Alignments wird in diesem Kapitel das grundlegende methodische Vorgehen in der Arbeit dargelegt. Weitergehende Informationen für spezifische Untersuchungen, z. B. die Ähnlichkeitssuche, und in diesem Zusammenhang erforderliche Abweichungen von der hier vorgestellten grundsätzlichen Vorgehensweise sind im jeweiligen Untersuchungsabschnitt erläutert.

### **6.1 Vorüberlegungen**

Für die durchzuführenden Untersuchungen werden Images und Templates benötigt. Die Images für die vorliegende Arbeit beinhalten Dateien, die teilweise auch als Template (zum Abgleich mit dem Image) verwendet werden. Zusätzlich setzt sich die Gesamtheit der Templates auch aus Dateien zusammen, die im Image nicht vorkommen, um Aussagen zu den für Tests bzw. Methoden üblichen Kennzahlen in Form von falsch-positiven und falsch-negativen Ergebnissen treffen zu können. Jeder Untersuchungsabschnitt umfasst idealerweise mehrere Images mit den zugehörigen Templates zur Erhöhung der Reliabilität und Validität der Untersuchungsergebnisse. In diesem Zusammenhang sind auch die Anzahl der Dateien zur Erzeugung der Images und die Anzahl der Templatedateien von Bedeutung; die Gewinnung einer großen Anzahl von Dateien ist durch den Einsatz von Softwaretools für den Download von Bilddateien aus dem Internet einfach durchführbar und für die Imageerstellung bzw. Templateauswahl, wie in der vorliegenden Arbeit, geeignet. Die bioinformatischen Untersuchungen finden mit Algorithmen statt, die durch spezielle Software bereits implementiert zur Verfügung gestellt werden. Der Einsatz dieser Algorithmen bedingt eine weitere Programmierung, da auf Grund der Datenmengen in Verbindung mit beschränkten Hardwareressourcen nur eine abschnittsweise Untersuchung der Daten möglich ist. Eine solche Programmierung wird im Rahmen der Arbeit erstellt.

### **6.2 Grundlegende Vorgehensweise beim Vergleich von Image und Template**

Ausgangspunkt der folgenden Untersuchungen bilden unter dem Betriebssystem Microsoft Windows 10 erstellte Partitionen auf einem USB-Stick. Diese werden mit dem

*MiniTool Partition Wizard* angelegt und anschließend mit Dateien in unterschiedlichen Formaten - überwiegend Bilddaten im *jpg*-Format (aus dem Internet in großer Anzahl leicht zu gewinnen) - belegt. Bei den **Dateien für die Images bzw. Templates** handelt es sich um durch den Autor der Arbeit selbst erstellte Bilder, Studienmaterial des Masterstudiengangs „IT-Sicherheit und Forensik“ der Hochschule Wismar/WINGS (z. B. *pdf*-Dokumente) und um Bilder von den Webseiten <https://pixabay.com/de/> und [www.pexels.com/de](http://www.pexels.com/de) [letzte Zugriffe: 5. Juli 2019]. Der Download der Bilddateien (z. B. die 10.000 Dateien für das Image F und 500 weitere Templatedateien)<sup>7</sup> erfolgt u. a. mit Hilfe der Software *PicaLoader*. Die Bilder von *pixabay* und *Pexels* dürfen kostenlos für kommerzielle und nichtkommerzielle Zwecke verwendet und bearbeitet werden, eine Namensnennung ist nicht erforderlich.<sup>8</sup>

Neben Images mit unveränderten Dateien werden für die Fragmentsuche Daten durch das Betriebssystem Microsoft Windows 10 zur Verfügung gestellte Funktionen gelöscht und mit anderen Daten (teilweise) überschrieben, um **Dateifragmente** unterschiedlicher Größe zu erzeugen. Anschließend werden von den Partitionen mit Hilfe der Software *AccessData® FTK® Imager* die Untersuchungsimages im RAW-Format (Rohdaten-Format) angelegt. Die genannte Erzeugung von Fragmenten vor der Imageerstellung (Löschvorgänge durch das Betriebssystem) wird durch Manipulationen - nachdem die Images erstellt wurden - ergänzt. Es erfolgt das „zufällige“ manuelle Überschreiben von Dateien mit dem Hex-Editor *iHex*. Darüber hinaus findet auch eine automatisierte Bearbeitung bereits erstellter Images in der Weise statt, dass Abschnitte fester Länge und in definierten Abständen mit Nullen überschrieben werden. Eine detaillierte Auflistung der verwendeten Hard- und Software ist in Anlage A, detaillierte Informationen zu den Images sind in Anlage B und dem entsprechenden Untersuchungsabschnitt zu finden. Die erstellten Images sind im Sinne des Ergebnisses einer Datensammlung beim potenziellen Täter zu verstehen, sie enthalten möglicherweise strafrechtlich relevantes Material. Das Auffinden der Imagelokalisationen mit den relevanten Daten beruht grundsätzlich auf dem Vergleich mit bereits bekannten Daten, sogenannten Templates.

---

<sup>7</sup> Die Kurzbezeichnungen der Images entsprechen der ursprünglichen Zuordnung des Laufwerksbuchstabens durch das Betriebssystem. Hierdurch soll die Gefahr von Verwechslungen minimiert werden. Ein Teil der Images und Templates diente im Vorfeld der Arbeit zur Vertiefung der forensischen Kenntnisse und vorbereitenden Maßnahmen.

<sup>8</sup> Weitere Angaben zu den Lizenzbedingungen können über die genannten Webseiten abgerufen werden.

**Im Rahmen dieser Arbeit werden Templates als bekannte Daten bzw. Dateien verstanden; sie dienen als Suchmuster für die Analyse der Images, die das relevante Material (die Templates) enthalten können.** Der Vergleich von Image und Template bzw. die Dateisuche umfasst die folgenden Punkte:

1. Vergleich von Image und Template - unveränderte Dateien im Image (Dateisuche).
2. Vergleich von Image und Template - teilweise überschriebene und manuell in das Image eingefügte Dateiinhalte (Fragmentsuche).
3. Suche nach ähnlichen Dateien - u. a. wird aus mehreren ähnlichen Templates eine sogenannte Konsensussequenz als neues Template erzeugt (Ähnlichkeitssuche).

Belastbare Untersuchungsergebnisse der Anwendung bioinformatischer Methoden zur Datensuche in der IT-Forensik im Rahmen von Studien sind aktuell nicht verfügbar, so dass die oben dargestellte Reihenfolge der Dateisuche für die vorliegende Arbeit entwickelt wurde und im Sinne einer Erhöhung des Schwierigkeitsgrades gesehen werden kann. Kaum ein etabliertes Verfahren deckt alle diese Aspekte mit implementierten Funktionalitäten ab. Die forensische Datensuche dürfte immer unter der Voraussetzung beginnen, dass die Daten unverändert vorhanden sind. Erst in einem weiteren Schritt geht man von veränderten bzw. manipulierten Daten aus. Stellen sich auch hierbei keine Sucherfolge ein, kann die Ähnlichkeitssuche zu positiven Ergebnissen führen. Vor dem Hintergrund der teilweise zeitlich sehr aufwendigen Analysen in Abhängigkeit von der eingesetzten Hard- und Software ist in der IT-Forensik ein solches stufenweises Vorgehen zu empfehlen. Auch bei dem Einsatz und der Bewertung neuer Methoden sollten die Untersuchungen mit unveränderten Daten beginnen, um auf dieser Basis die Analysen mit Fragmenten fortzusetzen. Die Suche nach ähnlichen Daten kann hiervon grundsätzlich unabhängig betrachtet werden, wobei die hier dargestellten bioinformatischen Methoden auch zur Ähnlichkeitssuche eingesetzt werden sollen und dies erst mit einer Bewährung bei der Suche nach unveränderten und fragmentierten Dateien sinnvoll erscheint. Eine erforderliche Programmierung zur abschnittsweisen Untersuchung der Images bei unveränderten Daten wird gleichsam bei der Ähnlichkeitssuche zur Anwendung kommen müssen. Blicke die Suche nach

unveränderten Daten erfolglos, dürfte dies auch bei anderen Suchvorgängen (Fragment- oder Ähnlichkeitssuche) mit der gleichen Softwareumgebung der Fall sein.

### 6.2.1 Transkription der Bitsequenz einer Datei in eine Basensequenz

Bevor bioinformatische Analysen durchgeführt werden können, ist das Bitmuster der Images und Templates in eine „Gensequenz“ (Basensequenz) zu übertragen. Im Rahmen der Arbeit kommen speziell für bioinformatische Untersuchungen vorhandene Softwaretools zum Einsatz (*Biopython* und *Matlab*, weitere Informationen sind in den folgenden Abschnitten zu finden). Die dort genutzten Algorithmen verwenden für die Berechnungen grundsätzlich festgelegte Alphabete, u. a. das Alphabet {A, C, G, T} für die Nukleinbasen Adenin, Cytosin, Guanin und Thymin. Sie sind wesentlicher Bestandteil der menschlichen DNA und bilden die Grundlage für die oben bereits erwähnte Mustererkennung. Mit Hilfe einer für die Arbeit erstellten *Python*-Klasse (Anlage E) wird eine Datei Byte für Byte eingelesen, das Bitmuster in ein Muster der Nukleinbasen übertragen und das Ergebnis in einer Datei im FASTA-Format gespeichert. Dieses Format ist ein spezielles Speicherformat für bioinformatische Informationen und ermöglicht eine direkte Verarbeitung durch bioinformatische Algorithmen. In Bild 11 ist ein solcher Auszug aus einer FASTA-Datei gezeigt. Die einzelnen Image- und Templatedateiinhalte werden also für die folgenden Untersuchungen in eine Basensequenz umgewandelt.

```

>indian.jpg Fasta-Sequenz
TTTTTCGATTTTGAACAAAAACAACAGGCACGCAGCCACGAAAAAACAAA
CAAAACAACAGTAAACAGTAAAAAATTTTGGACAAAATCTACACCCT
GACGGCCGCGAAAAAAACATCCATCAAAAAGGGAAAAAAGAAAAAGAA
AAAAACAACAAATTAAGAAAAAAGAAAAAAGAAAAAAGAAAAAAGAAA
AATTGAAACACAAAAAAGAAAAAAGAAAAAAGAAAAAAGAAAAAAGAAA
CAGAAGGCGGAAAAAACAAAAAAGAAAAAAGAAAAAAGAAAAAAGAAA ...

```

Bild 11: Auszug aus einer FASTA-Datei

Gemäß Bild 12 erfolgt dies in der Weise, dass jedes Dateibyte (8 Bits) in Gruppen zu je zwei Bits unterteilt und jeder Bitgruppe eine Base zugeordnet wird:

00 $\cong$ A ( <i>Adenin</i> )	01 $\cong$ C ( <i>Cytosin</i> )
10 $\cong$ G ( <i>Guanin</i> )	11 $\cong$ T ( <i>Thymin</i> )

Diese Festlegung wird für die folgenden Untersuchungen zugrunde gelegt. Eine andere Festlegung ist auch möglich, entscheidend ist eine einheitliche Umsetzung für die in die Berechnungen eingehenden Dateien (Images und Templates). Die Originaldateien erfahren durch die in Bild 12 gezeigte Vorverarbeitung eine Größenzunahme um den Faktor 4.

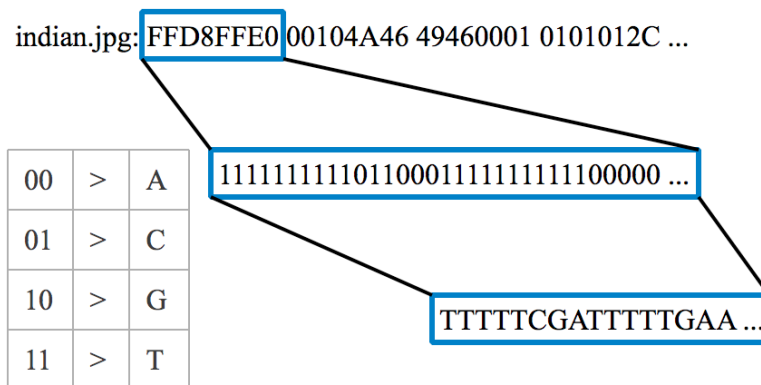


Bild 12: Übersetzung von Bytes in Nukleinbasen

### 6.2.2 Basenhäufigkeiten und orientierende Suche

Ein prominenter Ansatz bei der Mustererkennung in der Bioinformatik ist die Betrachtung von Basenhäufigkeiten<sup>9</sup>. Vor allem das Auftreten bestimmter kurzer Basensequenzen, sogenannter Codonen. Es handelt sich um Basen-3-Tupel, die eine Aminosäure kodieren. In der Bioinformatik spielen Codonen eine wichtige Rolle und spezielle Algorithmen (*codoncount()* in *Matlab*) zur Bestimmung der Codonen-Häufigkeiten ermitteln deren Vorkommen ab dem Beginn einer Sequenz in einer Schrittweite von drei (linke Seite in Bild 13). Für die Verarbeitung der Dateien im Rahmen dieser Arbeit wird diese Grundidee der Verwendung von Häufigkeiten - wie in Bild 13 auf der rechten Seite dargestellt (rote Umrandungen) - angepasst. Die Schrittweite beträgt hier nicht drei, sondern wird auf eins verkürzt und ermöglicht so die Erhöhung der Anzahl der in die Berechnungen eingehenden Basentupel. In *Matlab* kommt daher der Befehl *nmercount()* zum Einsatz. Je nach gewählter Tupellänge ermittelt *nmercount()*

<sup>9</sup> Der Begriff „Basenhäufigkeit“ umfasst für die folgenden Betrachtungen allgemein die Häufigkeiten von Basentupeln.

alle in der Basensequenz vorkommenden Muster, z. B. werden bei einer Tupellänge von zwei mit den Basentupeln bzw. Mustern {AA, AC, AG, AT, CA, CC, CG, CT, GA, GC, GG, GT, TA, TC, TG, TT} alle entsprechenden Muster in einer Basensequenz mit der Schrittweite eins identifiziert und gezählt.<sup>10</sup>



Bild 13: Codon-Häufigkeit versus Basentupel-Häufigkeit

Sowohl Images als auch Templates sind hinsichtlich ihrer Größe grundsätzlich keinen Beschränkungen unterworfen. Die Analyse einer Imagedatei in *Matlab* in Verbindung mit der vollständigen Übertragung der Datei in den Hauptspeicher ist auf Grund der eingesetzten Hardware nicht möglich oder aus zeitlichen Gründen nicht praktikabel. Aus

<sup>10</sup> Die Übertragung des Bitmusters einer Datei in das Basenalphabet und die anschließende Häufigkeitsermittlung kann auch ohne diese Transkription erfolgen. Eine Verwendung der in *Matlab* implementierten Funktion *nmercount()* ist so nicht möglich. Für die weitere Verarbeitung von Image und Template zur Berechnung von Alignmentsscores ist die Transkription zwingend erforderlich, so dass dieser Schritt bereits für die Häufigkeitsermittlung durchgeführt wird.

diesem Grunde erfolgt die Unterteilung des zu analysierenden Images in Untersuchungsabschnitte.<sup>11</sup> Die Größe dieser Abschnitte basiert auf der Länge des jeweiligen Untersuchungsfensters und entspricht im Falle der Suche nach unveränderten Dateien grundsätzlich der Länge des aktuell zu betrachtenden Templates. Eine Größe des Untersuchungsfensters kleiner als das aktuell untersuchte Template bedingt auch hier eine Unterteilung des Templates in Abschnitte, die mit jedem Imageabschnitt zu vergleichen sind - erforderlich kann dies bei der Suche nach Dateifragmenten sein. Diese für alle Untersuchungen der Arbeit grundlegende Vorgehensweise ist in Bild 14 dargestellt (Größe des Untersuchungsfensters gleich der Templatelänge).

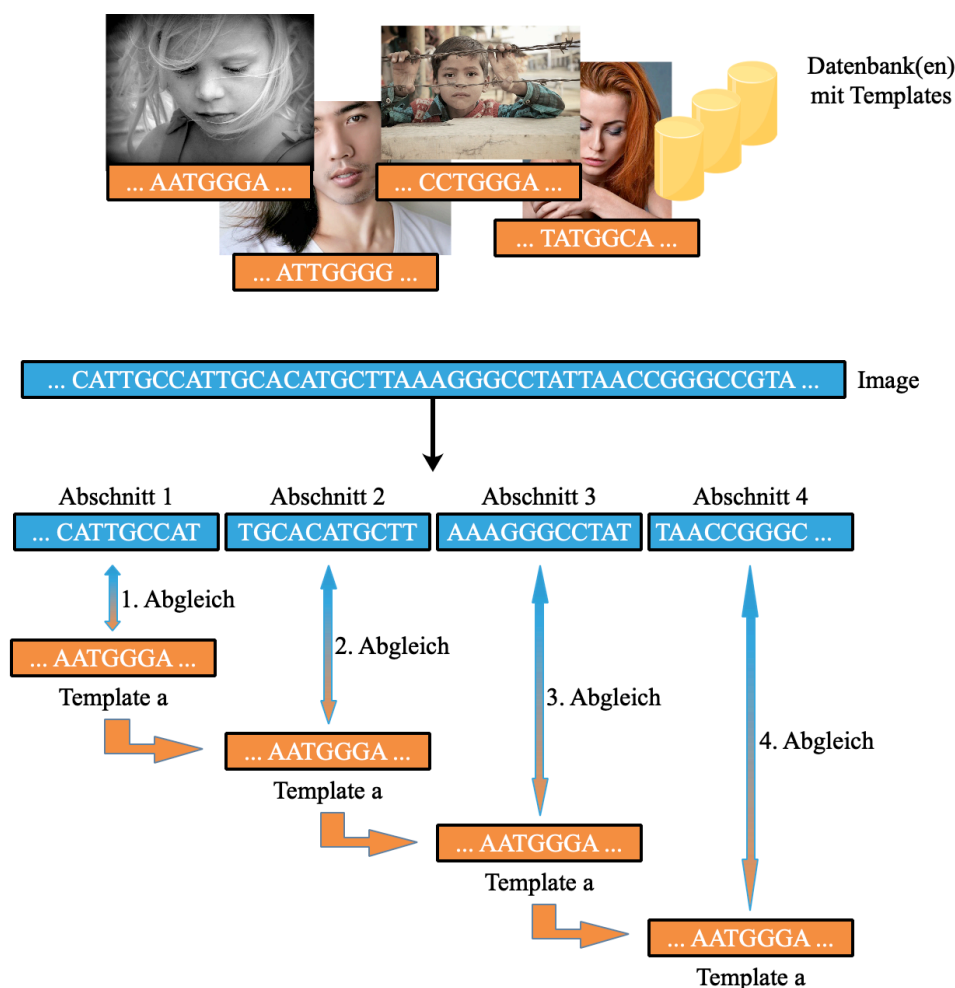


Bild 14: Abgleich zwischen Image und Template - orientierende Suche

<sup>11</sup> Eine solche Vorgehensweise - Unterteilung zu untersuchender Dateien in Blöcke bzw. Untersuchungsabschnitte - kommt auch in etablierten Verfahren wie *sdhash* oder *X-Ways* (Block Hashing) zum Einsatz (siehe hierzu auch Kapitel 4).



Die Realisierung der abschnittswisen Untersuchung erfolgt mit **eigens für die Arbeit in Matlab erstellten Algorithmen bzw. Skripten**. Es besteht hierbei die Möglichkeit, beliebig lange Basentupel für die Analyse zu definieren (Markierung ♦ in Anlage G), z. B. 2-Tupel: „AA“ ... „TT“ (insgesamt 16 Tupel) oder 4-Tupel: „AAAA“ ... „TTTT“ (insgesamt 256 Tupel). Bei der für die Arbeit eingesetzten Hardware (Anlage A) sind Tupellängen bis maximal vier praktikabel, darüber hinaus kommt es zu Systeminstabilitäten. Auch die Größe des Untersuchungsfensters ist frei wählbar. Sollte das Untersuchungsfenster beispielsweise kleiner als die Templatelänge gewählt werden (Eingabe  $t/2$  für die Hälfte der jeweiligen Templatelänge), wird jeder Imageabschnitt mit jedem Templateabschnitt verglichen.

Mit Hilfe der Basenhäufigkeiten im Image und Template soll eine orientierende Suche nach wahrscheinlichen Lokalisationen im Image realisiert werden (Bild 14). Hierzu werden die Basenhäufigkeiten im Template und an der aktuellen Position des Untersuchungsfensters im Image ermittelt (Tabelle 3). Der Vergleich der Basenhäufigkeiten wird nach folgender Formel durchgeführt:

$$\text{Differenzsumme}_{Image, Template}^{x,z} = \sum_{bt=1}^y |H_{Template_{bt}}^z - H_{Image_{bt}}^x| \quad (6.1)$$

mit  $bt = \text{aktuelles Basentupel}$ <sup>12</sup>,  $y = \text{Tupelanzahl}$ ,  $x = \text{Imageabschnitt}$  und  $z = \text{Templateabschnitt}$

Die Berechnung nach Formel 6.1 im Sinne eines Häufigkeitsabgleichs zwischen Image und Template entstammt keinen wissenschaftlichen Vorgaben, wird speziell für die folgenden Untersuchungen angewandt und ergibt sich daraus, dass eine Veränderung der Häufigkeiten durchaus zu vergleichbaren Häufigkeitssummen führen kann, so dass unterschiedliche Häufigkeitsprofile bezogen auf die einzelnen Basenhäufigkeiten nicht allein durch die Summenbildung differenzierbar sein müssen. Ein Beispiel hierfür seien die roten Werte in Tabelle 3, bei denen unveränderte Häufigkeitssummen mit einer erhöhten Differenzsumme einhergehen. Das bedeutet, dass die Differenzsumme nach Formel 6.1 geeignet erscheint, um Änderungen in Häufigkeitsprofilen aufzudecken. Hierbei wird die Gleichgewichtung der betrachteten Basentupel vorausgesetzt. Auch eine

---

<sup>12</sup> Die Basentupel werden lexikalisch geordnet und von Beginn an durchnummeriert. Das aktuelle Basentupel entspricht seiner Nummer in der Menge der Basentupel.

„einfache“ Differenzbildung (ohne Betrag) ist nicht ausreichend, entscheidend ist der in Bild 15 gezeigte Unterschied (schraffierter Bereich) zwischen zwei Häufigkeitsprofilen.

Tabelle 3: Beispiel für Basenhäufigkeiten im Image und Template

Basentupel	Häufigkeiten im Template $H_T$	Häufigkeiten im Image (Abschnitt x) $H_I^x$	Betrag der Differenz
AA	135.368 ( <b>150.368</b> )	157.895 ( <b>142.895</b> )	22.527 ( <b>7.473</b> )
AC	147.685	151.072	3.387
AG	142.343	148.693	6.350
AT	159.556	155.294	4.262
CA	144.465	150.315	5.850
CC	159.563	158.286	1.277
CG	155.781 ( <b>140.781</b> )	158.713 ( <b>173.713</b> )	2.932 ( <b>32.932</b> )
CT	168.805	160.194	8.611
GA	147.032	150.593	3.561
GC	154.070	153.061	1.009
GG	161.570	160.051	1.519
GT	165.950	163.949	2.001
TA	158.087	154.152	3.935
TC	167.297	165.088	2.209
TG	168.928	160.197	8.731
TT	167.584	156.531	11.053
$\Sigma$	<b>2.504.084</b>	<b>2.504.084</b>	<b>89.214 (104.160)</b>

Die Darstellung in Bild 15 ist gezielt als Plot und nicht Balkendiagramm gewählt worden, um die Unterschiede zwischen den Profilen besser hervorheben zu können.

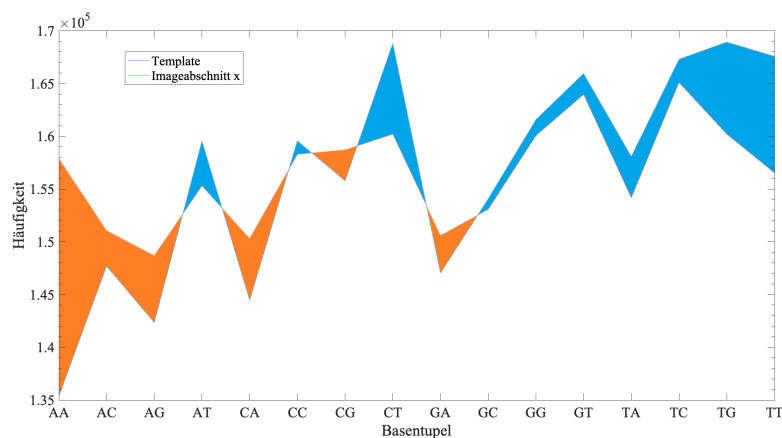


Bild 15: Unterschied zwischen den Häufigkeitsprofilen eines Images und Templates

Ausgehend von der Linie für das Image sind die Unterschiede durch Flächen oberhalb (blau) und unterhalb (orange) bedingt, dadurch wird die Betragsbildung in Formel 6.1 notwendig. Die berechneten Differenzsummen (*DiffSum*) sind durch unterschiedlich große Templates bedingt, ein direkter Vergleich ist so nicht möglich. In Abhängigkeit von der Templatelänge werden auch die Differenzsummen größer oder kleiner sein, erst die Normierung der einzelnen Ergebniswerte auf die jeweilige Templatelänge bzw. Größe des Untersuchungsfensters gleicht dies aus (normierte Differenzsummen). Der Imageabschnitt (Position des Untersuchungsfensters, siehe auch Bild 14) mit der kleinsten normierten Differenzsumme (*normDiffSum*) ist die wahrscheinlichste Lokalisation für das aktuell untersuchte Template. In die Ermittlung der Basenhäufigkeiten gehen die einzelnen Dateibestandteile (Metainformationen und Inhalt) gleichermaßen ein, die gesonderte Betrachtung erfolgt nicht.

Die folgenden Untersuchungen analysieren Images mit verschiedenen Templates, die in den Images vorhanden sind. Daraus werden die Mittelwerte (einfaches arithmetisches Mittel) der kleinsten normierten Differenzsummen mit den zugehörigen Standardabweichungen berechnet. Zusätzlich erfolgt die Analyse mit Templates, die nicht im Image vorhanden sind. Die auch hierfür errechneten Mittelwerte und Standardabweichungen werden dann mit den Mittelwerten und Standardabweichungen der im Image vorhandenen Templates mit Hilfe statistischer Tests verglichen, um signifikante Unterschiede aufzudecken.

Nachdem die Vorgehensweise bei der Bestimmung der Basenhäufigkeiten im Image und Template und deren Vergleich skizziert wurden, gilt es noch zu beachten, dass die Häufigkeiten beginnend ab der Position eins im Image bestimmt werden. Die Speicherung von Dateien beginnt jedoch abhängig vom vorhandenen Filesystem an anderer Stelle im Datenträger, da die ersten Sektoren mit den Angaben des Filesystems gefüllt sind. Auch beanspruchen die unterschiedlich großen Dateien jeweils spezifisch große Abschnitte (Cluster) auf dem Datenträger. **Bewusst herbeigeführte Veränderungen am Datenträger, um die Analyse durch IT-Forensiker zu erschweren oder unmöglich zu machen, können nicht ausgeschlossen werden, so dass eine Berücksichtigung des Filesystems und eine clusterorientierte Suche nach Dateien nicht vorgenommen werden.** Damit ist eine exakte Übereinstimmung der Position des Untersuchungsfensters mit der tatsächlichen Position des Templates - soweit

es im Image vorhanden ist - nicht zu erwarten, insbesondere bei Dateifragmenten ist auch mit einer Anpassung der Fenstergröße eine vollständige Überdeckung eher unwahrscheinlich. Daher wird das Image in mehreren Durchgängen mit einem gegenüber dem Imagebeginn verschobenen Untersuchungsfenster (Shift) erneut untersucht. Grundsätzlich kann von folgenden in Bild 16 gezeigten Fällen der Überdeckung ausgegangen werden. Der farbliche Balken soll das Image darstellen, das mit Datei 3 das aktuell zu untersuchende Template enthält und mit dem Untersuchungsfenster (\*) abgefahren wird. Die ursprünglichen Fensterpositionen sind mit  $WX_{alt}$  und die nach der Verschiebung ( $\rightarrow$ ) neue Position mit  $WX_{neu}$  gekennzeichnet. Im ersten Fall wird das Template im Image durch das Untersuchungsfenster nicht vollständig überdeckt. Eine Verschiebung des Fensterstarts im Image um die halbe Fensterlänge kann eine Verbesserung ( $W1_{neu}$ ) zur Folge haben. Im zweiten Fall ( $W2$ ) wird eine Verschiebung des Fensters zu keinem besseren Ergebnis führen.

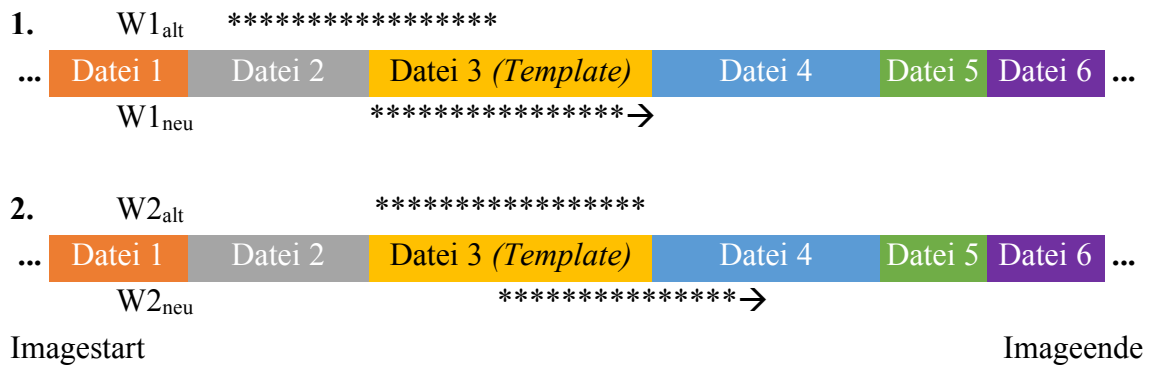


Bild 16: Untersuchungsfenster im Image

Das Untersuchungsfenster wird mit jedem Durchlauf  $i \in \mathbb{N}^{\geq 2}$  zur Analyse des Images um einen Shift nach Formel 6.2 verschoben:

$$Shift_i = fix(Shift_{i-1} + \frac{\text{Länge des Untersuchungsfensters}}{\text{Anzahl der Durchläufe } i}) \text{ mit } Shift_1 = 0^{13} \quad (6.2)$$

<sup>13</sup>  $Shift_1=0$  gilt, sofern vom Nutzer nicht zu Beginn der Suche bereits ein anderer Shift angegeben wird. Der Nutzer hat die Möglichkeit, folgende Parameter zu definieren: Größe des Untersuchungsfenster, Beginn im Image (Shift), Imagelänge, Schwellenwert für *normDiffSum*, Berechnung von Alignments und Wiederholung der Parameterdefinitionen für jeden Programmdurchlauf (Anlage G). Die *Matlab*-Funktion *fix()* rundet das Ergebnis auf den nächsten Integer-Wert nahe null.

Mit jeder neuen Analyse bzw. jedem neuen Durchlauf des Images wird die errechnete *normDiffSum* der Analyse ( $i - 1$ ) mit dem Wert des Durchlaufs  $i$  verglichen. Sollte folgende Bedingung gelten, wird die Imageanalyse abgebrochen:

$$\min(\text{normDiffSum}_i) \geq \min[\min(\text{normDiffSum}_1), \dots, \min(\text{normDiffSum}_{i-1})] \quad (6.3)$$

Die zur *normDiffSum* gespeicherte Imagelokalisation und ggf. Templatelokalisation (bei einem Untersuchungsfenster kleiner als die aktuelle Templatelänge) gehen in die folgende Alignmentberechnung ein. Die Imageanalyse ohne Shift mit einer Schrittweite von eins durchzuführen, ist bei der eingesetzten (beschränkten) Hardware und grundsätzlich in der Größe nicht beschränkten Images bzw. Templates unter zeitlichen Gesichtspunkten nicht praktikabel.

Das *Matlab*-Skript in Anlage G implementiert die oben beschriebene grundlegende Vorgehensweise, die hier als Pseudocode vorgestellt wird (Symbole kennzeichnen die entsprechende Codestelle in der Anlage):

```

array_image      = aktuell zu untersuchendes Image;
array_templates  = Templates für die Untersuchung;
window           = Größe des Untersuchungsfensters (grundsätzlich Länge des
                  aktuellen Templates);
Lade das aktuell zu untersuchende Image;
for Elemente in array_templates
  Lade ein Template;
  base_rate_template = Fahre mit window das Template ab und berechne die
  Basenhäufigkeiten; □
  while minimum noch nicht gefunden
    base_rate_image = Fahre mit window das Image ab und berechne die
    Basenhäufigkeiten (1. Durchlauf: Start bei Base 1); •
    sum_ = Summe(Betrag(base_rate_image - base_rate_template)); ★
    norm_ = sum_ / window;
    minimum_i = Ermittle und speichere die normDiffSum und den zugehörigen
    Image-/Templatebereich;
    Verschiebe den Start des Untersuchungsfensters nach Formel 6.2 (Shift); ∇
    Durchlaufe erneut das Image und finde ggf. minimum_{i+1} < minimum_i; wenn
    minimum_{i+1} ≥ minimum_i → Schleifenabbruch;
  global_score = globaler Score für den Imagebereich mit minimum; ❖
  local_score = lokaler Score für den Imagebereich mit minimum. ⌘

```

Die Anwendbarkeit der Bestimmung von Basenhäufigkeiten zur Diskriminierung von Dateien und deren Identifizierung in Images erfolgt im Schwerpunkt auf Grundlage der

drei für die Arbeit erzeugten Images L, V und F gemäß Tabelle 4. Die Images zur Fragmentsuche sind hiervon abgeleitet, Details sind in Anlage B zu finden. Das für die Ähnlichkeitssuche eingesetzte Image „Similar Files“ wird im entsprechenden Untersuchungsabschnitt eingeführt. Das Image L enthält 43 Dateien und wird mit 124 Templates untersucht (25 sind im Image enthalten), anschließend erfolgt eine Überprüfung anhand des zweiten Images V mit 528 Dateien (250 Templates, von denen 125 im Image enthalten sind). Über diese Angaben hinaus enthält jedes Image weitere Daten, z. B. in Form der *System Volume Information*, *Volume Boot*, *Unallocated Clusters* oder gelöschte Dateien.

Tabelle 4: Dateiformate für die Untersuchung von Basenhäufigkeiten (Images L / V / F)

<b>Dateiformat</b>	<b>Anzahl in den Images</b>	<b>Anzahl in den Templates</b>	<b>Bemerkung</b>
<i>jpg</i>	21 / 526 / 9.188	110 / 240 / 837	Bildformat
<i>png</i>	2 / 2 / 810	3 / 10 / 163	Bildformat
<i>ico</i>	- / - / -	1 / - / -	Bildformat
<i>xcf</i>	1 / - / -	1 / - / -	Bildformat
<i>pdf</i>	17 / - / -	7 / - / -	Dokumentenformat
<i>xlsx</i>	2 / - / -	- / - / -	Office-Format
<i>ppt</i>	- / - / -	2 / - / -	Office-Format
<b>Σ</b>	<b>43 / 528 / 9.998<sup>14</sup></b>	<b>124 / 250 / 1.000</b>	

Die Suche im dritten und größten Image F mit ca. 10.000 Dateien findet mit Hilfe von 1.000 Templates (500 davon im Image vorhanden) statt. Das Dateimaterial der drei Images umfasst die in Tabelle 4 aufgeführten Dateiformate. Die Analyse der oben genannten Images soll die Anwendbarkeit der in *Matlab* eingesetzten Skripte durch ihre Anzahl bestätigen, auf Grund ihrer unterschiedlichen Größen die Praktikabilität der Methoden aufzeigen und durch ihre verschiedenen Dateianzahlen belastbare Untersuchungsergebnisse liefern.

<sup>14</sup> Insgesamt handelt es sich um 10.000 Dateien, davon werden zwei Dateien mit der Endung *jpg* als Textdatei mit einer Größe von 0 Byte identifiziert (als Ursache können Übertragungsfehler/Kopierfehler nicht ausgeschlossen werden). Diesen Imagedateien ist keine Lokalisation zuzuordnen, so dass sie hier nicht mitgezählt werden.

### 6.2.3 Testung der orientierenden Suche in *Matlab*

Wo immer möglich, werden bereits implementierte Funktionen in *Matlab* genutzt. Darüber hinaus ist die Programmierung zur blockweisen Untersuchung von Image und ggf. Template erforderlich. Die zur Überprüfung der Korrektheit des oben dargestellten *Matlab*-Skriptes (Pseudocode) mit speziellen Images und Templates durchgeführten Tests sind in Tabelle 5 aufgeführt. Die Templates enthalten ausschließlich die Bitfolgen „00“, „01“, „10“ oder „11“ bei einer Dateilänge von 10 KB.

Tabelle 5: Testimages und Testtemplates (TestT) zur Überprüfung des *Matlab*-Skriptes

Testimage	TestT (00..00)	TestT (01..01)	TestT (10..10)	TestT (11..11)
00..01..00	erkannt	lokalisiert	-	-
00..10..00	erkannt	-	lokalisiert	-
00..11..00	erkannt	-	-	lokalisiert
01..00..01	lokalisiert	erkannt	-	-
01..10..01	-	erkannt	lokalisiert	-
01..11..01	-	erkannt	-	lokalisiert
10..00..10	lokalisiert	-	erkannt	-
10..01..10	-	lokalisiert	erkannt	-
10..11..10	-	-	erkannt	lokalisiert
11..00..11	lokalisiert	-	-	erkannt
11..01..11	-	lokalisiert	-	erkannt
11..10..11	-	-	lokalisiert	erkannt

Die 16 Testimages bestehen zunächst nur aus dem Bitmuster „00“, „01“, „10“ oder „11“ und sind jeweils 100 KB groß. Ab dem Byte 10.000 sind in die Testimages dann die bereits bei den Templates genannten Bitfolgen über eine Länge von 10 KB eingebettet (Bild 17).

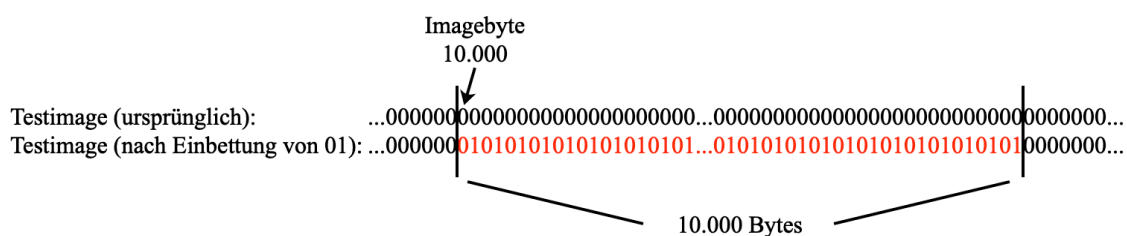


Bild 17: Erstellung von Testimages zur Skriptüberprüfung

Die Testungen erfolgten mit 1-, 2-, 3- und 4-Tupel und sind in Tabelle 5 nicht gesondert aufgeführt, da die Lokalisation des jeweiligen Templates in jedem Fall korrekt gefunden wurde. „Erkannt“ in Tabelle 5 zeigt die Erkennung und Lokalisation des Templates an. Der in Rede stehende Algorithmus ermittelt die (eine) wahrscheinlichste Imagelokalisation, weitere werden grundsätzlich nicht ausgegeben, wobei sich die „erkannten“ Templates mehrfach im Testimage befinden.

Zusätzlich zu dem bisher getesteten Algorithmus-Anteil der Analyse von Basenhäufigkeiten wird nun der gesamte Algorithmus betrachtet (Basenhäufigkeiten und Berechnung von Alignmentscores). Hierzu dient das Testimage „Test Algorithm“ mit einer einzelnen, sich wiederholenden Datei *5Oen8B7tBqvS5-43rB6lQ.jpg* (Dateigröße: 192.233 Byte). Die orientierende Suche wird mit den Templates *5Oen8B7tBqvS5-43rB6lQ.jpg* (vorhanden, links in Bild 18) und *indian.jpg* (nicht vorhanden, rechts in Bild 18) durchgeführt.

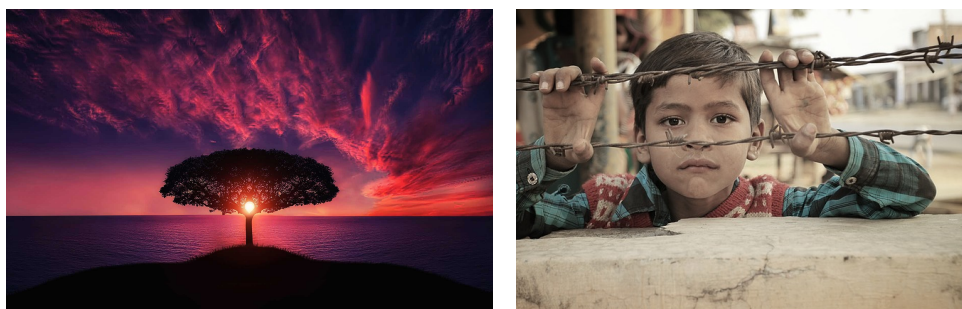


Bild 18: Templates für den Algorithmustest der orientierenden Suche

Es ergeben sich die in Tabelle 6 aufgeführten Werte für die kleinsten normierten Differenzsummen (*normDiffSum*), größten normierten globalen und normierten lokalen Alignmentscores. Die kursiven Werte beziehen sich auf das im Testimage nicht vorhandene Template *indian.jpg*. Das Untersuchungsfenster entspricht der Templatelänge (Dateigröße) in Basen. Der Shift beträgt anfangs null, damit beginnt die orientierende Suche am Beginn des Images. Des Weiteren wird ein Shift von 65.536 untersucht. Die ersten 32 Sektoren (0 bis 31) benötigt das Filesystem FAT, ab Sektor 32 beginnt die Dateispeicherung. Für die zu analysierende Sequenz des Images nach der FAT ergibt sich bei einer Sektor-/Blockgröße von 512 Bytes somit ein Shift von  $32 * 512 * 4 = 65.536$  Basen. Die Analyse des Images „Test Algorithm“ mit dem Forensiktool *EnCase* ergibt für das oben genannte vorhandene Template als kleinstes Startbyte im Image



281.600 (erste Lokalisation), in der entsprechenden Basensequenz also  $281.600 * 4 = 1.126.400$  Basen.

Tabelle 6: Ergebnisse der Untersuchung des Images „Test Algorithm“

	Differenzsumme (normiert)	Globaler Score (normiert)	Lokaler Score (normiert)
<b>Untersuchungsfenster = Templatelänge [Basen]; Shift = 0 Basen</b>			
<i>1-Tupel</i>	0,0014   0,0294	0,0950   0,0965	0,1061   0,1033
<i>2-Tupel</i>	0,0028   0,0762	0,0765   0,0966	0,0925   0,1033
<i>3-Tupel</i>	0,0031   0,1158	0,0767   0,0966	0,0928   0,1033
<i>4-Tupel</i>	0,0039   0,1656	0,0767   0,0966	0,0928   0,1033
<b>Untersuchungsfenster = Templatelänge [Basen]; Shift = 65.536 Basen</b>			
<i>1-Tupel</i>	0,0006   0,0218	0,5264   0,0998	0,8404   0,1057
<i>2-Tupel</i>	0,0008   0,0723	0,5264   0,0998	0,8404   0,1057
<i>3-Tupel</i>	0,0011   0,1133	0,5264   0,0998	0,8404   0,1057
<i>4-Tupel</i>	0,0016   0,1688	0,5264   0,0998	0,8404   0,1057
<b>Untersuchungsfenster = Templatelänge [Basen]; Shift = 1.126.400 Basen</b>			
<i>1-Tupel</i>	$1,3005 * 10^{-6}$   0,0358	0,9999   0,0730	1,0000   0,0868
<i>2-Tupel</i>	$1,3005 * 10^{-6}$   0,1385	0,9999   0,0869	1,0000   0,0949
<i>3-Tupel</i>	$1,3005 * 10^{-6}$   0,2532	0,9999   0,0895	1,0000   0,0944
<i>4-Tupel</i>	$1,3005 * 10^{-6}$   0,3654	0,9999   0,0895	1,0000   0,0944

An dieser Stelle soll eine kurze Bewertung der in Tabelle 6 dargestellten Ergebnisse erfolgen, da sie einen Einfluss auf die weiteren Untersuchungen haben können. **Bei einer vollständigen Überdeckung des Templates mit der entsprechenden Imagelokalisation - wie hier bei einem Shift = 1.126.400 Basen - wird eine kleinste normierte Differenzsumme in Höhe von null erwartet und für die ebenso auf die Länge des Untersuchungsfensters normierten globalen bzw. lokalen Alignmentsscores die maximalen Werte von eins.<sup>15</sup> Diese Erwartungen werden durch das Skript der orientierenden Suche erfüllt.**

<sup>15</sup> Es sei darauf hingewiesen, dass die mehrfache Speicherung einer Datei nur mit veränderten Dateinamen möglich ist. Damit unterscheiden sich die Bitmuster bzw. Basensequenzen diesbezüglich. Die bei einem Shift = 1.126.400 Basen ermittelte *normDiffSum* beruht auf der Datei *5Oen8B7tBqvS5-43rB6lQ - Kopie.jpg* und entspricht nicht dem ursprünglichen Dateinamen *5Oen8B7tBqvS5-43rB6lQ.jpg*, damit können die Abweichungen bei der *normDiffSum* von null und beim normierten globalen Score von eins begründet werden.

#### 6.2.4 Berechnung globaler und lokaler Alignments

Die mit der orientierenden Suche ermittelte wahrscheinlichste Imagelokalisation des aktuell untersuchten Templates geht in die Berechnung der Alignmentsscores ein, um den Übereinstimmungsgrad der dort lokalisierten Imagesequenz mit der Templatesequenz zu bestimmen. Die orientierende Suche liefert in jedem Falle eine kleinste normierte Differenzsumme und damit auch eine Imagelokalisation, unabhängig von dem tatsächlichen Vorhandensein des Templates im Image. Der globale oder lokale Alignmentsscore soll hier die Unterscheidung zwischen im Image vorhandenen und nicht vorhandenen Templates ermöglichen. Es handelt sich bei der Berechnung von Alignmentsscores also um eine Ergänzung der orientierenden Suche, die das Suchergebnis bestätigen oder ein falsch-positives Ergebnis aufzeigen hilft. Die Berechnung der globalen und lokalen Alignmentsscores wird mit *Biopython* und *Matlab* durchgeführt:

1. *Biopython*-Funktionen: *pairwise2.align.globalxx()*, *pairwise2.align.localxx()*
2. *Matlab*-Funktionen: *nwalign()*, *swalign()*

Es stellt sich bei der Verwendung von *Biopython* heraus, dass neben einem hohen Zeitbedarf für die Berechnungen insbesondere nur kleinere Untersuchungsblöcke im Vergleich zu *Matlab* mit der hier eingesetzten Hardware verarbeitet werden können. Die Platzkomplexität beträgt sowohl für den Needleman Wunsch- als auch Smith Waterman-Algorithmus grundsätzlich  $\mathcal{O}(n, m) = n * m$ . Für die Ermittlung eines Alignments sind mehrere Matrices notwendig, so dass sich in der Folge der Platzbedarf um einen Faktor  $k$  erhöht, hinzu kommt der Speicherbedarf für die Werte in der Matrix. Die Abstützung auf die genannten Algorithmen zur Berechnung von Alignments ist somit hinsichtlich ihrer praktischen Anwendung u. a. durch die Größe des zur Verfügung stehenden Hauptspeichers begrenzt. Weitere aktive Prozesse, hierunter auch die *Python*-Entwicklungsumgebung, reduzieren das Speicherangebot zusätzlich und eine einfache Anpassung der Untersuchungsblockgrößen auf die maximal verfügbare Größe des Hauptspeichers ist so nicht möglich; die Software *Matlab* bietet hier nach Tests im Rahmen der Arbeit mehr Spielraum. Hinzukommt, dass Implementierungen der gleichen Algorithmen in *Matlab* einen deutlich geringeren Zeitbedarf als in *Biopython* besitzen (empirisch ermittelt). Vor diesem Hintergrund erfolgen die Berechnungen der Alignments für die folgenden Untersuchungen ausschließlich in *Matlab*.

Die oben aufgeführten *Matlab*-Funktionen verfügen über folgende Parameter, mit deren Hilfe die Funktionsausführung und damit das Alignmentergebnis beeinflusst werden können (vgl. [38], S. 115):

***nwalign*(Sequenz\_1, Sequenz\_2, 'ScoringMatrix', scoring\_matrix, 'gapopen', 1, 'extendgap', 1, 'alphabet', 'nt') → globales Alignment:**

Sequenz_1	= Sequenz 1, z. B. das Image
Sequenz_2	= Sequenz 2, z. B. das Template
ScoringMatrix	= Berechnungswerte für einen sogenannten Basen-Match bzw. Basen-Mismatch
gapopen	= Berechnungswert für die Eröffnung einer Lücke („Strafwert“)
extendgap	= Berechnungswert für die Verlängerung einer Lücke („Strafwert“)
Alphabet	= das verwendete Alphabet - hier {A, C, G, T} für Nukleinbasen (nt); eine Alternative wäre die Verwendung des Alphabets für Aminosäuren (aa)

***swalign*(Sequenz\_1, Sequenz\_2, 'ScoringMatrix', scoring\_matrix, 'gapopen', 1, 'extendgap', 1, 'alphabet', 'nt') → lokales Alignment:**

siehe *nwalign*()

Für die Scoring-Matrix werden die Werte [1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1] gewählt, um Ähnlichkeitsscores zu berechnen (vgl. [38], S. 110ff). Die Funktionen *nwalign*() und *swalign*() können auf verschiedenen Alphabeten arbeiten, allerdings führt die Eingabe der Bits „0“ und „1“ zu einer Fehlermeldung - die beschriebene Transkription ist erforderlich.

Die für eine forensische Untersuchung eingesetzten Images und Template unterliegen keinen Größenbeschränkungen. Im Gegensatz hierzu wird die Untersuchungshardware immer über begrenzte Ressourcen, z. B. Arbeitsspeicher und Anzahl der zur Verfügung stehenden Prozessoren, verfügen. Damit ist es - wie oben bereits mehrfach erwähnt - erforderlich, die Daten mit an den vorhandenen Ressourcen angepassten Fenstergrößen blockweise zu untersuchen. In der Folge gilt es, die Frage zu beantworten:

Sind die Alignmentergebnisse von der Fenstergröße abhängig?

Zur Beantwortung der Frage werden 100 Templatedateien zur Suche im Image V unter dem Einsatz verschiedener Fenstergrößen - 16.000, 32.000, 48.000 und 64.000 - verwendet. Die Definition größerer Fenster kann bei der hier eingesetzten Hardware (Anlage A) zu Systeminstabilitäten führen. Die statistisch ermittelten  $p$ -Werte sind in Anlage C zu finden. Zwischen den Fenstergrößen existieren statistisch signifikante Unterschiede. Die Fenstergröße sollte so groß wie möglich gewählt werden, um idealerweise den gesamten Dateiinhalt in einem Schritt zu alignieren. Dies ist auf Grund der unterschiedlichen Dateigrößen und der begrenzten Hardwareressourcen nicht in jedem Falle möglich. In der Folge wird für die globalen und lokalen Alignmentberechnungen grundsätzlich eine Fenstergröße von 64.000 Basen verwendet, diese kann bei der Suche nach unveränderten Dateien in Abhängigkeit von der Templategröße oder bei der Suche nach Dateifragmenten in Abhängigkeit von der vermuteten Fragmentgröße auch weniger als 64.000 Basen betragen. Die Größe des Arbeitsspeichers lässt vereinzelt auch Fenstergrößen oberhalb von 64.000 Basen zu, allerdings wächst der Speicherbedarf bei der Berechnung Multipler Alignments mit der Sequenzanzahl und idealerweise werden aus zeitlichen Gründen möglichst viele Untersuchungen - z. B. die Analyse der Images L, V und F - parallel durchgeführt; damit **beträgt hier die Fenstergröße zur Berechnung von Alignments in allen Fällen grundsätzlich 64.000 Basen.**

Die weitere Analyse der mit der orientierenden Suche ermittelten wahrscheinlichsten Imagelokalisation erfolgt für eine Fenster- bzw. Templategröße über 64.000 Basen wie in Bild 19 gezeigt.

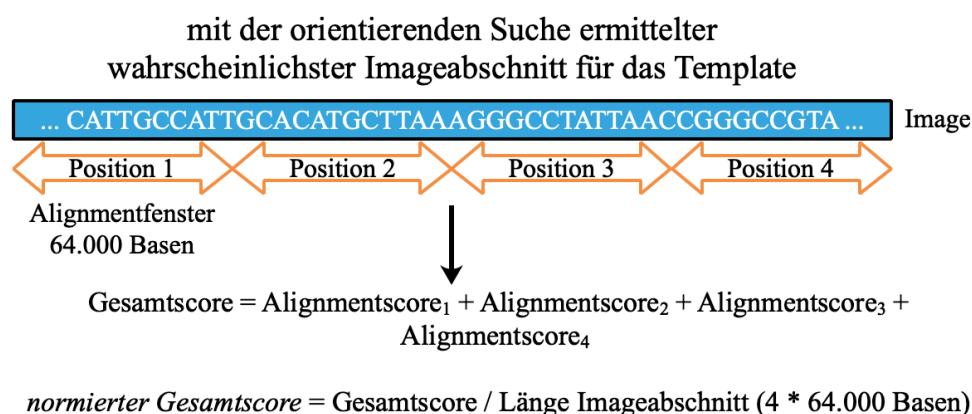


Bild 19: Alignments in Ergänzung zur orientierenden Suche

Der Gesamtscore als Maß der Übereinstimmung zwischen Imageabschnitt und Template ergibt sich als Summe der abschnittsweise errechneten Alignmentscores, die auf die Fenstergröße normiert wird (Summe der Alignmentscores / Fenstergröße [Basen]).

### 6.3 Statistische Methoden zur Bewertung der Untersuchungsergebnisse

Sowohl etablierte Verfahren als auch bioinformatische Methoden werden das in einem Image vorhandene (strafrechtlich) relevante Material unter allen Rahmenbedingungen (intaktes/nicht intaktes Filesystem, unveränderte/veränderte Dateien) nicht immer einwandfrei identifizieren. Die Bewertung eines Verfahrens bzw. einer Methode hinsichtlich der Aussagekraft kann durch verschiedene Kenngrößen - z. B. Sensitivität, Spezifität oder die Wahrscheinlichkeiten für falsch-positive und falsch-negative Ergebnisse - erfolgen. Kennzeichnend für jede Untersuchung seien hier die Wahrscheinlichkeiten für falsch-positive und falsch-negative Ergebnisse, deren Angabe auch in Studien zu etablierten Verfahren zu finden ist oder aus den Analyseergebnissen berechnet werden kann. Vergleichbar einem Testverfahren in der medizinischen Statistik bilden die Templates eine Stichprobe, deren Elemente sich durch die beiden Merkmale „im Image vorhanden“ und „im Image nicht vorhanden“ unterscheiden. Die statistischen Auswertungen basieren auf einer Vierfeldertafel (Tabelle 7; vgl. u. a. [44], S. 199ff und S. 259ff), deren Werte in die Berechnung der Wahrscheinlichkeiten für falsch-positive und falsch-negative Ergebnisse gemäß Formel 6.4 und 6.5 eingehen.<sup>16</sup>

Tabelle 7: Schema einer Vierfeldertafel

	<b>Templates im Image V<sup>+</sup></b>	<b>Templates <u>nicht</u> im Image V<sup>-</sup></b>
<b>Methode/Software X erkennt verdächtige Daten (E<sub>+</sub>)</b>	a {a'}	b
<b>Methode/Software X erkennt verdächtige Daten nicht (E<sub>-</sub>)</b>	c	d

*Wahrscheinlichkeit für falsch-positive Ergebnisse (False Positive Rate):*

$$FPR = P(E_+|V^-) = 1 - P(E_-|V^-) = 1 - \left(\frac{d}{d+b}\right) \quad (6.4)$$

<sup>16</sup> Informationen zu den *FPR* und *FNR* können auch im Internet abgerufen werden unter: [https://en.wikipedia.org/wiki/Precision\\_and\\_recall](https://en.wikipedia.org/wiki/Precision_and_recall) [letzter Zugriff: 5. Juli 2019].

*Wahrscheinlichkeit für falsch-negative Ergebnisse (False Negative Rate):*

$$FNR = P(E_-|V^+) = 1 - P(E_+|V^+) = 1 - \left(\frac{a}{a+c}\right) \quad (6.5)$$

Der Abgleich von Image und Template zum Auffinden unveränderter Dateien oder von Dateifragmenten entspricht durchaus auch den Herausforderungen von Identifikationssystemen in der Biometrie (vgl. [25], S. 11ff). Die Angaben von *FPR* und *FNR* basieren bei der orientierenden Suche auf einer Überlappung der mit Hilfe der *normDiffSum* ermittelten mit der tatsächlichen Imagelokalisation. Die mit bioinformatischen Methoden ermittelten Imagelokalisationen der Templates können von den tatsächlichen Lokalisationen abweichen, für vorhandene Templates ergibt sich eine Überlappung der ermittelten mit der tatsächlichen Lokalisation von größer als 0 % - idealerweise größer als 50 % und im optimalen Fall 100 %. Ergibt die Untersuchung für vorhandene Templates eine Überlappung größer als 0 % (Wert *a* in der Vierfeldertafel) und stimmt der Templatename mit dem Namen der Imagedatei überein (in den *Matlab*-Tabellen mit \*\*\* gekennzeichnet), so handelt es sich um ein korrekt positives Ergebnis ( $E_+|V^+$ ) und wird in geschweiften Klammern gesondert angegeben ( $\{a'\}$ ). Ein korrekt negatives Untersuchungsergebnis für im Image nicht vorhandene Templates zeichnet sich durch eine Überlappung in Höhe von 0 % aus ( $E_-|V^-$ ), die mit der orientierenden Suche ermittelte Lokalisation ist keiner tatsächlichen Imagelokalisation zuzuordnen. Eine Namensübereinstimmung ist bei den nicht vorhandenen Templates nicht von Bedeutung (auch nicht möglich). Die orientierende Suche liefert in jedem Falle eine *normDiffSum* und damit auch eine Imagelokalisation, dennoch ist eine fehlende Überlappung u. a. auf Grund der für das Filesystem reservierten Cluster und vorhandenem Slack Space möglich.<sup>17</sup> Für die in geschweiften Klammern gesondert angegebenen korrekten Ergebnisse ( $\{a'\}$  in Tabelle 7) folgen nach den Formeln 6.4 und 6.5 auch andere Werte für *FPR* und *FNR* (*a* wird durch  $a'$  und *b* durch  $[b + (a - a')]$  ersetzt), diese werden ebenfalls gesondert in geschweiften Klammern angegeben. Diese Vorgehensweise ist auf

---

<sup>17</sup> Zusätzlich kann bei der Analyse der Untersuchungsergebnisse z. B. die Bedingung einer Überlappung > 50 % für ein positives Untersuchungsergebnis formuliert werden. Die Anwendung des beschriebenen Shifts soll eine Überlappung von mindestens 50 % garantieren, so dass eine Überlappung ≤ 50 % kennzeichnend für ein negatives Ergebnis sein kann. Für die hier bekannten Images ist dies möglich, in der Realität kennt der IT-Forensiker die Images nicht. Der Schwellenwert für die Überlappung beträgt daher bei allen Untersuchungen 0 %.

Grund der hier bekannten Images möglich und dient der Bewertung der eingesetzten Methoden. Damit kann die Glaubwürdigkeit als Anforderung an die forensische Untersuchung abgeschätzt werden (vgl. [9], S. 22f). In einem weiteren Schritt kann das Festlegen eines Schwellenwertes für die berechneten *normDiffSum* der vorhandenen und nicht im Image vorhandenen Templates die Trennschärfe zwischen diesen beiden Templategruppen erhöhen. Die Zusammensetzung der Images für die vorliegende Arbeit ist bekannt, die Einteilung in vorhandene und nicht vorhandene Templates ist einfach möglich. Bei unbekannten Images müssen die Untersuchungsergebnisse mit einem „Ausgangswert“ verglichen werden, dieser kann der Schwellenwert sein. In diesem Sinne werden auch geeignete Schwellenwerte für die globalen und lokalen Alignmentsscores angegeben. Mit einer Verschiebung der Schwellenwerte wird man die beiden Kenngrößen *FPR* und *FNR* nicht gleichzeitig optimieren können. In Abhängigkeit vom angestrebten Ziel ist der Schwellenwert zu wählen (vgl. [25], S. 18ff). In diesen Fällen erfolgt die Beschreibung des Vorgehens in den jeweiligen Untersuchungsabschnitten. Die Berechnung der Kennzahlen *FPR* und *FNR* dient - wie oben bereits erwähnt - dem Vergleich der bioinformatischen Ergebnisse untereinander, soweit möglich dem Vergleich mit den Untersuchungsergebnissen der etablierten Verfahren im Rahmen dieser Arbeit und anderen Studienergebnissen.

Die statistische Analyse auf signifikante Unterschiede zwischen den untersuchten Gruppen der im Image vorhandenen und nicht vorhandenen Templates (*normDiffSum* und Alignmentsscores) beruht unter der Voraussetzung einer Normalverteilung mit annähernd gleicher Varianz auf der Anwendung des t-Tests für zwei unverbundene Stichproben (vgl. [44], S. 185ff). Sollte die Voraussetzung der Normalverteilung nicht gegeben sein, kommt der verteilungsfreie Rangsummentest nach Wilcoxon für zwei Stichproben (bzw. U-Test nach Mann und Whitney, vgl. [44], S. 190f) zum Einsatz. Das Signifikanzniveau beträgt für beide Tests 5 %. Die Überprüfung der Normalverteilung erfolgt mit Hilfe des Tests nach Lilliefors (*lillietest()* in *Matlab*). Extremwerte und Ausreißer gehen in die Berechnungen ein, die entsprechende Bereinigung der Untersuchungsergebnisse erfolgt nicht.

#### 6.4 Überblick zur eingesetzten Software (Bioinformatik und etablierte Verfahren)

Zur Datensuche in den Images unter Verwendung bioinformatischer Methoden werden *Biopython* (Vorverarbeitung in Form der Transkription und Modifikation von Images zur Fragmenterzeugung) und *Matlab* (Datensuche) eingesetzt und den Analyseergebnisse der etablierten Verfahren *Autopsy* (einschließlich *Scalpel* für das File Carving), *X-Ways*, *EnCase* und *sdhash* [36] gegenübergestellt.<sup>18</sup> *Biopython* ist eine in der Programmiersprache *Python* entwickelte Sammlung frei verfügbarer Werkzeuge für biologische Berechnungen (vgl. [16]). Die Software *Matlab* stellt umfangreiche Werkzeuge, darunter eine *Bioinformatics Toolbox*, für mathematische Berechnungen bereit. Unter den etablierten Verfahren generiert das Softwaretool *sdhash* terminalbasiert Hashwerte für einzelne Imageabschnitte, die mit den Hashwerten von Templates verglichen werden. Kennzeichnend für *sdhash* soll sein, dass auch unterschiedliche Dateiversionen erkannt werden können (vgl. [36], S. 12). Bei den anderen etablierten Verfahren - *Autopsy*, *X-Ways* und *EnCase* - handelt es sich um Software-Suiten mit einer grafischen Benutzeroberfläche. Sie alle verfügen über Funktionalitäten zur Analyse von Filesystemen, Erzeugung von Hashwerten und zum File Carving.

Im Rahmen dieser Arbeit in *Python* oder *Matlab* erstellte Programme (Skripte, Funktionen, Klassen) sind vollständig auf dem beigefügten Datenträger zu finden, dies gilt auch für die verwendeten Images, Templates und Berechnungsergebnisse (*Matlab*-Dateien \*.mat). Für das Verständnis wesentliche Teile sind in Anlage D bis Anlage H aufgenommen. Alle *Python*- und *Matlab*-Programme sind in Entwicklungsumgebungen - hier *Eclipse* und *Matlab* - ausführbar.

#### 6.5 Zusammenfassung des methodischen Vorgehens

Eine Zusammenfassung des grundsätzlichen Vorgehens in der Arbeit ist Bild 20 zu entnehmen. Die in eine Basensequenz übersetzten Dateien (Kapitel 6) werden in Untersuchungsblöcke unterteilt. Ein Image wird sowohl mit im Image vorhandenen als auch nicht vorhandenen Templates „verglichen“, um die Untersuchungsblöcke (Image

---

<sup>18</sup> Weitere Informationen zu den eingesetzten etablierten Verfahren finden sich in Anlage A und Abschnitt 8.1.



und Template) zu identifizieren, die die kleinste normierte Differenzsumme der Basenhäufigkeiten aufweisen (Kapitel 7).

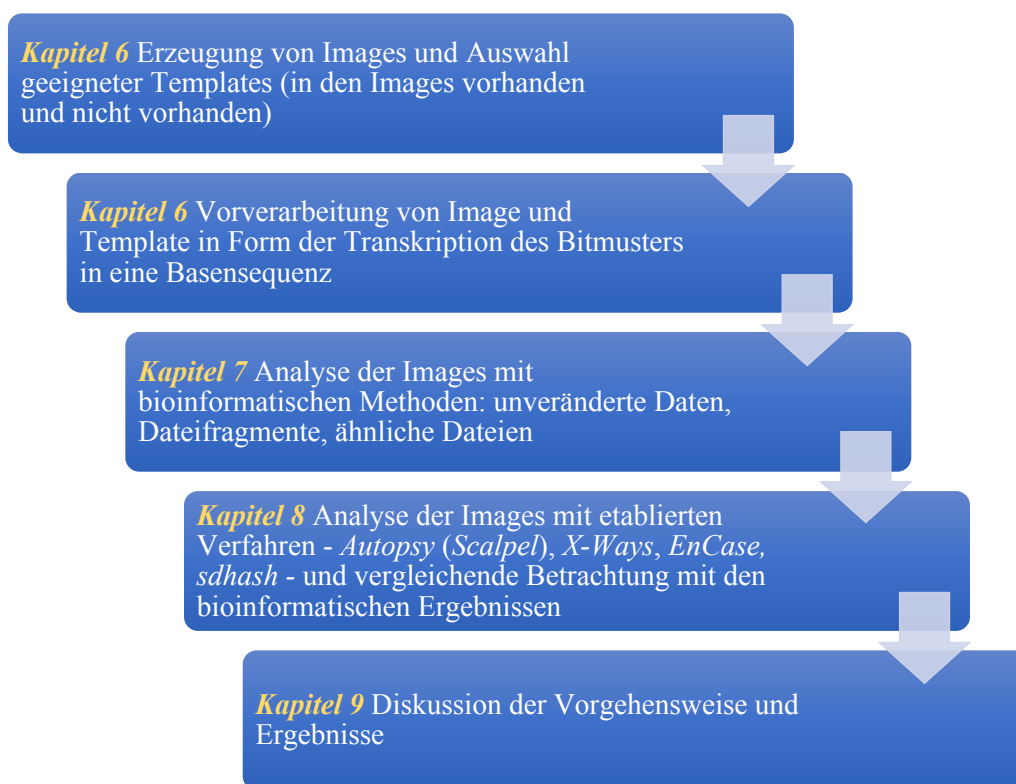


Bild 20: Vorgehensweise in der Arbeit

Ein in dieser Weise identifizierter Untersuchungsblock entspricht gleichzeitig der wahrscheinlichsten Lokalisation des Templates bzw. Templateblocks im Image. Für die im Image vorhandenen Templates berechneten kleinsten normierten Differenzsummen werden das einfache arithmetische Mittel und die zugehörige Standardabweichung errechnet, ebenso für die untersuchten nicht vorhandenen Templates. Der Image-Template-Vergleich erfolgt jeweils für die Basentupel eins bis vier. Die genannten Werte (Mittelwert und Standardabweichung in Abhängigkeit vom untersuchten Basentupel) werden grundsätzlich in jedem Untersuchungsabschnitt tabellarisch aufgeführt. Zusätzlich werden die beiden Templategruppen - im Image vorhanden und nicht vorhanden - statistisch hinsichtlich eines signifikanten Unterschiedes ( $p$ -Wert)<sup>19</sup> analysiert. Auch dieses Ergebnis findet sich in den Tabellen. Da die tatsächlichen

<sup>19</sup> Der  $p$ -Wert eines statistischen Tests beschreibt die Wahrscheinlichkeit für das Vorliegen der Nullhypothese (vgl. [44], S. 169ff). Die Nullhypothese hier besagt, dass sich die Untersuchungsgruppen nicht unterscheiden.

Imagelokalisationen der Dateien bekannt sind, ist in einem nächsten Schritt die Befüllung einer Vierfeldertafel (Tabelle 7) möglich. Die hier eingetragenen Werte sind die Grundlage für die nach den Formeln 6.4 und 6.5 aufgezeigten *FPR* und *FNR*. Die Einführung und Berücksichtigung von Schwellenwerten werden im jeweiligen Abschnitt gesondert ausgeführt. Zur Bestätigung oder Zurückweisung der ermittelten wahrscheinlichsten Imagelokalisation sollen die Berechnungen der globalen und lokalen Alignmentsscores dienen. Auch für die Alignmentsscores werden die oben genannten Kennzahlen (Mittelwerte, Standardabweichungen, *p*-Wert, *FPR* und *FNR*) angegeben. Ebenso erfolgt die Suche allein auf Basis von Alignmentberechnungen, Häufigkeitsprofile spielen keine Rolle. Das Image wird wiederum mit dem Untersuchungsfenster abschnittsweise abgefahren, wobei für jeden Abschnitt der normierte globale und lokale Alignmentsscore berechnet wird. Der Imageabschnitt mit dem maximalen normierten globalen bzw. lokalen Alignmentsscore entspricht dann der wahrscheinlichsten Imagelokalisation des untersuchten Templates. Die zuvor eingesetzten Images und Templates werden gleichermaßen mit den genannten etablierten Verfahren untersucht und wo immer möglich die Kennzahlen *FPR* und *FNR* errechnet (Kapitel 8). Diese Ergebnisse werden den bioinformatischen gegenübergestellt. Abschließend werden in Kapitel 9 die Vorgehensweise und Untersuchungsergebnisse diskutiert.

## 7 Untersuchungsgegenstand „Bioinformatische Methoden“

Dieses Kapitel bildet den Schwerpunkt der Arbeit und wendet die oben beschriebenen bioinformatischen Methoden auf die Datensuche an. Auf Basis der hier errechneten Untersuchungsergebnisse ist dann im nächsten Kapitel der Vergleich mit etablierten Verfahren möglich.

### 7.1 Orientierende Suche mit Hilfe von Basenhäufigkeiten - unveränderte Daten

#### 7.1.1 Größe des Untersuchungsfensters ist gleich der Templatelänge

In Bild 21 sind die Basenhäufigkeiten an der errechneten wahrscheinlichsten Imagelokalisation für ein nicht vorhandenes Template visualisiert, in Bild 22 entsprechend für ein vorhandenes Template.

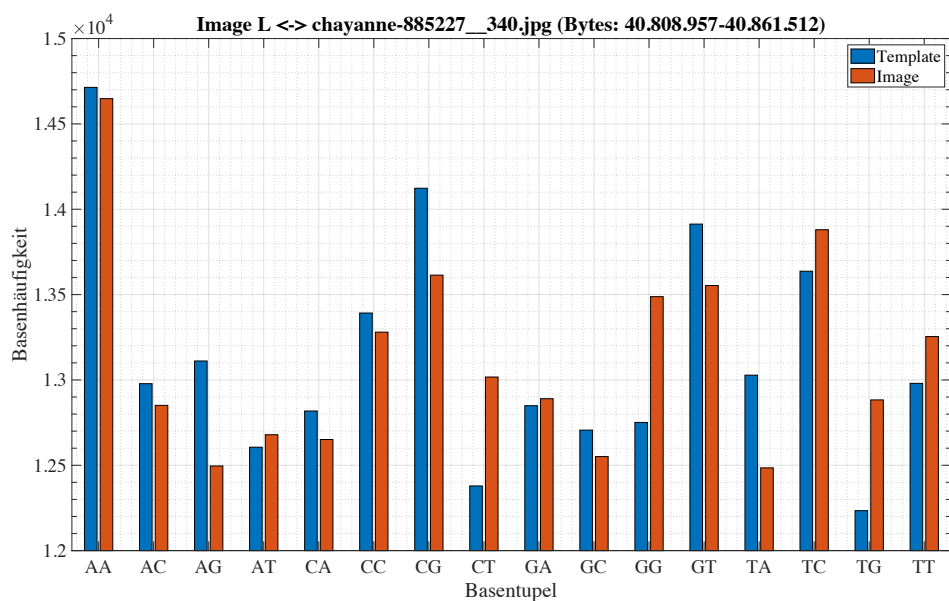


Bild 21: Beispiel für den Vergleich von Basenhäufigkeiten, Template nicht vorhanden (2-Tupel)

Die Summe der Differenzen zwischen den gruppierten Säulen in Bild 21 ist vergleichsweise größer als in Bild 22 und zeugt von einer geringeren Übereinstimmung.

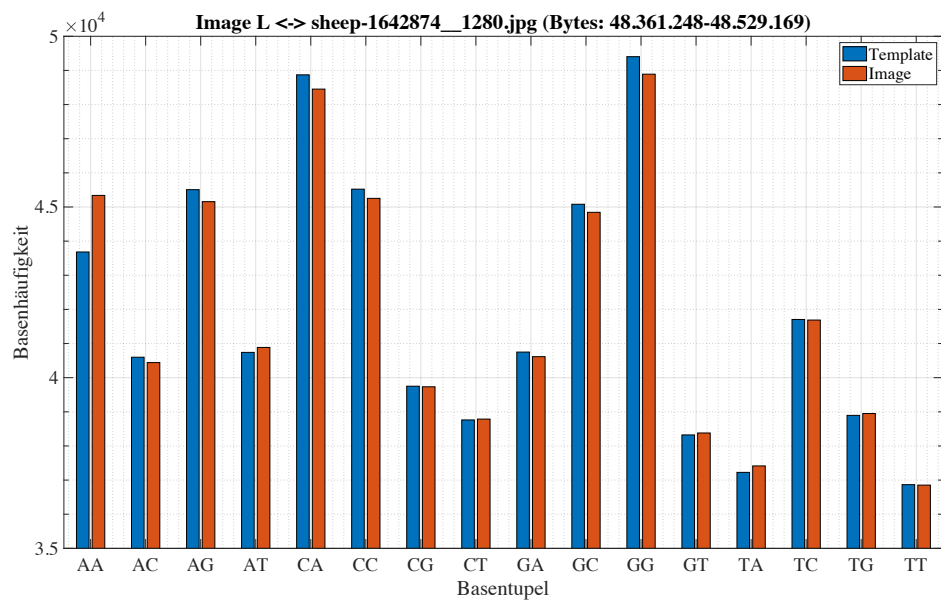


Bild 22: Beispiel für den Vergleich von Basenhäufigkeiten, Template vorhanden (2-Tupel)

In Tabelle 8 sind für die *normDiffSum* die Mittelwerte (mean) und in runden Klammern die Standardabweichungen (std) aufgeführt.

Tabelle 8: *NormDiffSum* der orientierenden Suche - unveränderte Dateien

<i>normDiffSum</i>			U-Test [ <i>p</i> -Wert]
Templates im Image V <sup>+</sup> Templates <u>nicht</u> im Image V <sup>-</sup>			
Image L	25 Dateien	99 Dateien	
1-Tupel	0,0039 (0,0036)	0,0038 (0,0154)	0,0010
2-Tupel	0,0131 (0,0056)	0,0247 (0,0379)	9,2359 * 10 <sup>-6</sup>
3-Tupel	0,0238 (0,0095)	0,0564 (0,0556)	5,9047 * 10 <sup>-12</sup>
4-Tupel	0,0359 (0,0155)	0,1010 (0,0752)	1,9965 * 10 <sup>-13</sup>
Image V	125 Dateien	125 Dateien	
1-Tupel	0,0022 (0,0025)	0,0036 (0,0085)	0,1665
2-Tupel	0,0150 (0,0076)	0,0224 (0,0166)	2,1332 * 10 <sup>-6</sup>
3-Tupel	0,0308 (0,0175)	0,0530 (0,0337)	3,1947 * 10 <sup>-12</sup>
4-Tupel	0,0448 (0,0275)	0,0946 (0,0538)	1,6702 * 10 <sup>-25</sup>
Image F	500 Dateien	500 Dateien	
1-Tupel	0,0017 (0,0018)	0,0022 (0,0024)	5,2107 * 10 <sup>-11</sup>
2-Tupel	0,0119 (0,0051)	0,0145 (0,0076)	3,7758 * 10 <sup>-9</sup>
3-Tupel	0,0247 (0,0099)	0,0311 (0,0124)	1,4971 * 10 <sup>-16</sup>
4-Tupel	0,0391 (0,0157)	0,0536 (0,0171)	1,6494 * 10 <sup>-36</sup>

In Tabelle 9 sind die Ergebnisse der orientierenden Suche als Vierfeldertafel angegeben. Zunächst werden keine Schwellenwerte berücksichtigt. Von den im Image L vorhandenen Templates werden ab der Untersuchung mit Basentupeln der Länge drei, bis auf ein Template, alle mit einer Überlappung der ermittelten mit der tatsächlichen Imagelokalisation von über 0 % erkannt. Für das eine nicht erkannte Template existiert keine Überlappung mit einer relevanten Imagelokalisation. Dies ist möglich, da Imagebereiche ohne Dateizuordnung vorhanden sind - der Speicherbereich für die FAT und nicht allozierte Speicherbereiche. Von den vorhandenen Templates sind insgesamt 23 Templates korrekt identifiziert, der Name der an der tatsächlichen Imagelokalisation vorhandenen Datei stimmt mit dem Templatenamen überein. Ein ähnliches Verhalten zeigt sich auch für das Image V. Die Untersuchung des größten Images F ergibt für die Tupellänge vier allerdings eine im Vergleich zu den anderen Images eher geringe Anzahl korrekt positiver Ergebnisse. Die positiven Testergebnisse ( $E_+$ ) umfassen auch eine entsprechende Anzahl falsch-positiver Resultate.

Tabelle 9: Vierfeldertafel *normDiffSum*, kein Schwellenwert - unveränderte Daten

Image	Templates im Image $V^+$			Templates <u>nicht</u> im Image $V^-$		
	<i>L</i>	<i>V</i>	<i>F</i>	<i>L</i>	<i>V</i>	<i>F</i>
<b><math>E_+</math></b> <i>1-Tupel</i>	15 {7}	102 {13}	452 {17}	8	74	471
<i>2-Tupel</i>	21 {19}	119 {64}	460 {100}	25	99	478
<i>3-Tupel</i>	24 {23}	124 {99}	475 {176}	32	104	472
<i>4-Tupel</i>	24 {23}	125 {120}	482 {286}	28	99	463
<b><math>E_-</math></b> <i>1-Tupel</i>	10	23	48	91	51	29
<i>2-Tupel</i>	4	6	40	74	26	22
<i>3-Tupel</i>	1	1	25	67	21	28
<i>4-Tupel</i>	1	0	18	71	26	37

Die sich aus Tabelle 9 ergebenden *FPR* und *FNR* sind in Tabelle 10 bis Tabelle 12 aufgeführt. Sie zeigen die geringsten Wahrscheinlichkeiten für falsch-negative Ergebnisse bei den Tupellängen drei und vier in allen Images.

Tabelle 10: *FPR* und *FNR* Image L, *normDiffSum*, kein Schwellenwert - unveränderte Daten

<b>Image L</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,0808 {0,1495}	0,4000 {0,5882}
<i>2-Tupel</i>	0,2525 {0,2673}	0,1600 {0,1739}
<i>3-Tupel</i>	0,3232 {0,3300}	0,0400 {0,0417}
<i>4-Tupel</i>	0,2828 {0,2900}	0,0400 {0,0417}

Tabelle 11: *FPR* und *FNR* Image V, *normDiffSum*, kein Schwellenwert - unveränderte Daten

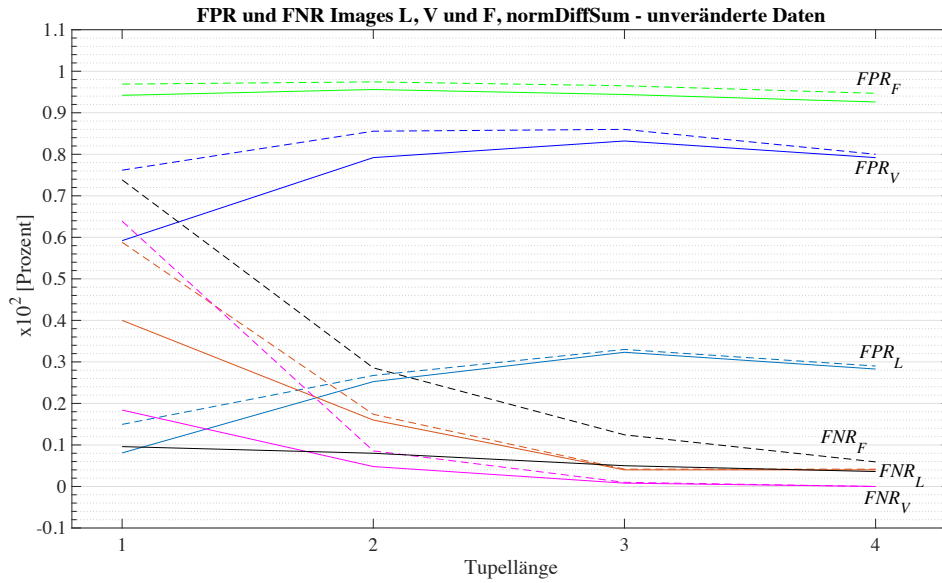
<b>Image V</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,5920 {0,7617}	0,1840 {0,6389}
<i>2-Tupel</i>	0,7920 {0,8556}	0,0480 {0,0857}
<i>3-Tupel</i>	0,8320 {0,8600}	0,0080 {0,0100}
<i>4-Tupel</i>	0,7920 {0,8000}	0,0000 {0,0000}

Tabelle 12: *FPR* und *FNR* Image F, *normDiffSum*, kein Schwellenwert - unveränderte Daten

<b>Image F</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,9420 {0,9690}	0,0960 {0,7385}
<i>2-Tupel</i>	0,9560 {0,9744}	0,0800 {0,2857}
<i>3-Tupel</i>	0,9440 {0,9650}	0,0500 {0,1244}
<i>4-Tupel</i>	0,9260 {0,9468}	0,0360 {0,0592}

Die Ergebnisse aus Tabelle 10 bis Tabelle 12 sind in Bild 23 visualisiert. Die Tabellenwerte in geschweiften Klammern sind in Bild 23 als gestrichelte Linien dargestellt. Diese Form der Darstellung mit der linearen Verbindung zwischen den Basentupeln wird zur besseren Übersicht gewählt.

Eine Auswertung ohne die Anwendung von Schwellenwerten erscheint fraglich. Sollten die Dateien im Image Größen aufweisen, die idealerweise keinen Slack Space zulassen und die orientierende Suche nach den für das Dateisystem reservierten Clustern beginnen, gibt es keine negativen Untersuchungsergebnisse. Jedem Template (auch den nicht vorhandenen) würde durch das hier angewandte Vorgehen eine Imagelokalisation mit einer Datei zugeordnet.

Bild 23: FPR und FNR für *normDiffSum*, kein Schwellenwert - unveränderte Daten

In der Folge wird für die *normDiffSum* der Schwellenwert  $ut$  zur Berechnung der falsch-positiven und falsch-negativen Ergebnisse für im Image (i. I.) vorhandene Templates zunächst wie folgt definiert:

$$\begin{aligned}
 ut_{Tupellänge} &= t_{Tupellänge}^{normDiffSum \text{ i.I.}} \\
 &= mean_{Tupellänge}^{normDiffSum \text{ i.I.}} + std_{Tupellänge}^{normDiffSum \text{ i.I.}}
 \end{aligned} \tag{7.1}$$

Der Schwellenwert  $ut$  schließt für die weiteren Untersuchungen als oberer Schwellenwert idealerweise alle im Image vorhandenen Templates ein. Es wird angenommen, dass der IT-Forensiker im Schwerpunkt relevantes Material auffinden möchte und zunächst nicht der (vollständige) Ausschluss relevanten Materials im Vordergrund steht. Dies scheint auf Grund der in Tabelle 8 angegebenen geringeren Standardabweichungen für die im Image vorhandenen Templates auch aussichtsreicher als die Definition eines Schwellenwertes für die nicht im Image vorhandenen Templates. Von dieser Vorstellung ausgehend, werden - getrennt für jede Tupellänge - folgende Schwellenwerte  $t_n$  zur Berechnung der FPR und FNR untersucht:

$$t_n \in [a, b] \tag{7.2}$$

$$\text{mit } a = \begin{cases} t_-, \text{ falls } std_{Tupellänge}^{normDiffSum \text{ i.I.}} < mean_{Tupellänge}^{normDiffSum \text{ i.I.}} \\ 0, \text{ sonst.} \end{cases} \tag{7.3}$$

$$\text{und} \quad t_- = \text{mean}_{Tupell\ddot{a}nge}^{normDiffSum \text{ i.I.}} - \text{std}_{Tupell\ddot{a}nge}^{normDiffSum \text{ i.I.}} \quad (7.4)$$

$$\text{und} \quad b = \text{mean}_{Tupell\ddot{a}nge}^{normDiffSum \text{ i.I.}} + \text{std}_{Tupell\ddot{a}nge}^{normDiffSum \text{ i.I.}} \quad (7.5)$$

Die einzelnen zu betrachtenden Schwellenwerte ergeben sich nach:

$$ut_n = a + n * \frac{\text{std}_{Tupell\ddot{a}nge}^{normDiffSum \text{ i.I.}}}{10}, ut_n \in t_n \text{ mit } n \in \mathbb{N}^{\geq 0} \quad (7.6)$$

Zum besseren Verständnis der Formeln 7.2 bis 7.6 ergeben sich als Beispiel für einen Mittelwert (mean) in Höhe von 0,0038 und einer Standardabweichung (std) von 0,0154 für  $a = 0$ , da  $\text{std} > \text{mean}$ ,  $b = 0,0038 + 0,0154 = 0,0192$  und  $ut_0 = 0$ ,  $ut_1 = 0,00154$ ,  $ut_2 = 0,00308$ ,  $ut_3 = 0,00462 \dots ut_{12} = 0,01848$ ,  $ut_{13} = 0,02002 \rightarrow ut_{13} = 0,0192$ .

Die Bestimmung des besten Schwellenwertes erfolgt nach den Kriterien:

1.  $ut_n$  mit den kleinsten  $FPR$  und  $FNR$ , ansonsten
2.  $ut_n$  mit der kleinsten Summe aus  $FPR$  und  $FNR$ . Bei mehreren gleichen Summen entscheidet die kleinste  $FNR$  über die Wahl des Schwellenwertes.

Anschließend wird der beste Schwellenwert  $ut_n$  mit  $t_-$ ,  $\text{mean}_{Tupell\ddot{a}nge}^{normDiffSum \text{ i.I.}}$  und  $b$  hinsichtlich deren Abweichung voneinander in Abhängigkeit von der Tupellänge untersucht. Der U-Test ergibt die größten statistischen Abweichungen für  $ut_n$  und  $t_-$  sowie  $b$ , die geringste Abweichung des besten Schwellenwertes  $ut_n$  ist bei  $\text{mean}_{Tupell\ddot{a}nge}^{normDiffSum \text{ i.I.}}$  gegeben. In allen Fällen liegen keine signifikanten Unterschiede vor. Für die folgenden Berechnungen unter Berücksichtigung eines oberen Schwellenwertes für  $normDiffSum$  wird

$$ut_{Tupell\ddot{a}nge} = \text{mean}_{Tupell\ddot{a}nge}^{normDiffSum \text{ i.I.}} \quad (7.7)$$

gewählt. Diese Vorgehensweise soll bei gleicher Größe des Untersuchungsfensters einen grundsätzlich nur von der Tupellänge abhängigen Schwellenwert definieren und bewusst keinen für die jeweilige Untersuchungsreihe spezifischen. Idealerweise stimmen die so ermittelten Schwellenwerte für alle untersuchten Images überein und können imageunabhängig auch für andere Untersuchungen genutzt werden - dies ist für die



Images L, V und F nur teilweise erfüllt. In Tabelle 13 sind die Anzahlen der mit der orientierenden Suche identifizierten Templates unter Berücksichtigung des Schwellenwertes nach Formel 7.7 als Vierfeldertafel aufgeführt.

Tabelle 13: Vierfeldertafel  $normDiffSum \leq ut$  - unveränderte Daten

Image	Templates im Image $V^+$			Templates <u>nicht</u> im Image $V^-$		
	$L$	$V$	$F$	$L$	$V$	$F$
<b>E<sub>+</sub></b> 1-Tupel	8 {3}	70 {8}	313 {10}	4	44	262
2-Tupel	13 {11}	61 {47}	254 {67}	3	29	202
3-Tupel	14 {14}	61 {58}	245 {126}	1	3	152
4-Tupel	16 {16}	62 {61}	221 {177}	1	0	43
<b>E<sub>-</sub></b> 1-Tupel	10	21	41	87	33	19
2-Tupel	1	2	28	10	6	15
3-Tupel	0	0	22	0	8	10
4-Tupel	0	0	3	0	0	7

Hieraus ergeben sich die in Tabelle 14 bis Tabelle 16 gezeigten Wahrscheinlichkeiten für falsch-positive und falsch-negative Ergebnisse. In den Tabellen aufgeführte „-“ entsprechen in *Matlab* dem Berechnungsergebnis *NaN* (Not a Number; Division durch null).

Tabelle 14: *FPR* und *FNR* Image L,  $normDiffSum \leq ut$  - unveränderte Daten

Image L	<i>FPR</i>	<i>FNR</i>
1-Tupel	0,0440 {0,0938}	0,5556 {0,7692}
2-Tupel	0,2308 {0,3333}	0,0714 {0,0833}
3-Tupel	1,0000 {1,0000}	0,0000 {0,0000}
4-Tupel	1,0000 {1,0000}	0,0000 {0,0000}

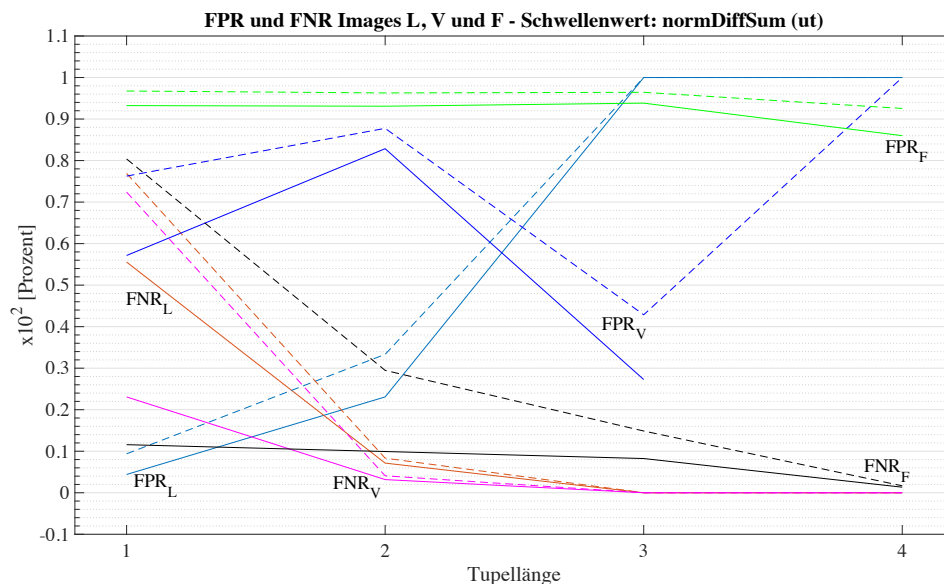
Tabelle 15: *FPR* und *FNR* Image V,  $normDiffSum \leq ut$  - unveränderte Daten

Image V	<i>FPR</i>	<i>FNR</i>
1-Tupel	0,5714 {0,7626}	0,2308 {0,7241}
2-Tupel	0,8286 {0,8776}	0,0317 {0,0408}
3-Tupel	0,2727 {0,4286}	0,0000 {0,0000}
4-Tupel	- {1,0000}	0,0000 {0,0000}

Tabelle 16: *FPR* und *FNR* Image F, *normDiffSum*  $\leq$  *ut* - unveränderte Daten

Image F	<i>FPR</i>	<i>FNR</i>
1-Tupel	0,9324 {0,9675}	0,1158 {0,8039}
2-Tupel	0,9309 {0,9629}	0,0993 {0,2947}
3-Tupel	0,9383 {0,9644}	0,0824 {0,1487}
4-Tupel	0,8600 {0,9255}	0,0134 {0,0167}

Die Ergebnisse aus Tabelle 14 bis Tabelle 16 sind in Bild 24 visualisiert. Die Tabellenwerte in geschweiften Klammern sind in Bild 24 als gestrichelte Linien dargestellt. Diese Form der Darstellung mit der linearen Verbindung zwischen den Tupellängen wird zur besseren Übersicht gewählt.

Bild 24: *FPR* und *FNR* für Schwellenwert *normDiffSum* (*ut*) - unveränderte Daten

Ergänzend soll darauf hingewiesen werden, dass die im Image L nicht vorhandene Datei *BSI\_Fingerabdruckerkennung\_Historie.pdf* mit der *normDiffSum* in Verbindung mit einer Übereinstimmung - der ermittelten mit der tatsächlichen Imagelokalisation - von 98 % der im Image vorhandenen Datei *BSI\_Fingerabdruckerkennung\_Historie\_eingebettet\_indian.pdf* zugeordnet wird. Beide Dateien enthalten den gleichen Text, unterscheiden sich jedoch durch eine Einbettung des Bildes *indian.jpg*. Vergleichbare Untersuchungsergebnisse existieren für die Dateien *9303\_p10\_cons\_en.pdf* (Imagedatei) und *9303\_p10\_cons\_en\_indianJPG\_2.pdf* (nicht vorhandenes Template) mit einer

Überdeckung von 96 %. Mit Hilfe des *Matlab*-Skriptes zur Suche nach ähnlichen Dateien (Anlage D) können alternative Lokalisationen, deren *normDiffSum* um einen bestimmten Prozentanteil vom Minimum nach oben abweichen, bestimmt werden. So kommt es beispielsweise unter einer zugelassenen Abweichung von 10 % bei 2-Tupel vor, dass die Templatedatei X eine 71%ige Überdeckung mit der Imagedatei Y und eine 84%ige Überdeckung mit seiner tatsächlichen Lokalisation im Image aufweist, wobei im ersten Fall eine kleinere normierte Differenzsumme ermittelt wird. In den oben genannten Fällen unterscheiden sich Image- und Templatedatei nur teilweise, sind aber nicht identisch. Auf Grund der unterschiedlichen Dateinamen ist eine Erfassung als korrektes positives Untersuchungsergebnis zunächst nicht möglich (siehe hierzu auch Abschnitt 6.3).

#### 7.1.2 Auswirkungen der Fenstergröße auf das Untersuchungsergebnis

Im Folgenden werden das Image V und die Templates mit zwei festen Fenstergrößen zur Berechnung der Basenhäufigkeiten abgefahren, um Auswirkungen auf das Untersuchungsergebnis aufzudecken. Bisher entspricht die Größe des Untersuchungsfensters der Größe des aktuell für die Untersuchung genutzten Templates. Orientierend an den Imagedatei- und Templategrößen werden die Fenstergrößen mit 64.000 und 128.000 Basen gewählt. Von den insgesamt 250 Templates zur Untersuchung des Images V befinden sich 125 Templates im Image. Die durchschnittliche Größe der Dateien des Images V beträgt ca. 68 KB, wobei knapp 87 % der Dateien kleiner als 100 KB sind und eine durchschnittliche Größe in Höhe von ca. 25 KB ( $\cong$  100.000 Basen) aufweisen. Die für das Image V verwendeten Templates besitzen eine durchschnittliche Dateigröße von ca. 50 KB, knapp 90 % aller Templates sind unter 100 KB groß, durchschnittlich umfassen sie ca. 32 KB ( $\cong$  128.000 Basen). Die Ergebnisse ohne die Berücksichtigung von Schwellenwerten sind in Tabelle 17 als Vierfeldertafel zusammengefasst.

Die tupelabhängigen *normDiffSum* sind nicht explizit aufgeführt und können der Ergebnisdatei auf dem Datenträger entnommen werden.

Tabelle 17: Vierfeldertafel für Image V, kein Schwellenwert, unterschiedliche Fenstergrößen

<b>TW<sup>20</sup></b>	Templates im Image V <sup>+</sup>		Templates <u>nicht</u> im Image V <sup>-</sup>	
	<b>64.000</b>	<b>128.000</b>	<b>64.000</b>	<b>128.000</b>
<b>E<sub>+</sub></b> <i>1-Tupel</i>	44 {6}	24 {2}	34	47
<i>2-Tupel</i>	48 {21}	31 {17}	55	61
<i>3-Tupel</i>	55 {41}	34 {29}	65	57
<i>4-Tupel</i>	59 {57}	39 {38}	62	55
<b>E<sub>-</sub></b> <i>1-Tupel</i>	81	101	91	78
<i>2-Tupel</i>	77	94	70	64
<i>3-Tupel</i>	70	91	60	68
<i>4-Tupel</i>	66	86	63	70

Die Ergebnisse der Untersuchungen unter Anwendung des Schwellenwertes für *normDiffSum* gemäß Formel 7.7 (für jede Tupellänge und Fenstergröße werden die Mittelwerte neu berechnet) sind in Tabelle 18 als Vierfeldertafel dokumentiert.

Tabelle 18: Vierfeldertafel für Image V, *normDiffSum* ≤ *ut*, unterschiedliche Fenstergrößen

<b>TW</b>	Templates im Image V <sup>+</sup>		Templates <u>nicht</u> im Image V <sup>-</sup>	
	<b>64.000</b>	<b>128.000</b>	<b>64.000</b>	<b>128.000</b>
<b>E<sub>+</sub></b> <i>1-Tupel</i>	42 {6}	22 {2}	34	43
<i>2-Tupel</i>	45 {21}	29 {17}	52	57
<i>3-Tupel</i>	49 {40}	33 {29}	57	53
<i>4-Tupel</i>	49 {49}	35 {34}	30	43
<b>E<sub>-</sub></b> <i>1-Tupel</i>	38	43	79	46
<i>2-Tupel</i>	35	31	58	35
<i>3-Tupel</i>	33	29	50	36
<i>4-Tupel</i>	31	27	44	37

Aus den oben genannten Vierfeldertafeln ergeben sich die in Tabelle 19 und Tabelle 20 gezeigten *FPR* und *FNR*.

<sup>20</sup> Template Window = Größe des Untersuchungsfensters.

Tabelle 19: *FPR* und *FNR* Image V, keine Schwellenwerte, unterschiedliche Fenstergrößen

	<i>FPR</i>		<i>FNR</i>	
<b>TW</b>	<b>64.000</b>	<b>128.000</b>	<b>64.000</b>	<b>128.000</b>
<i>1-Tupel</i>	0,2720 {0,4417}	0,3760 {0,4694}	0,6480 {0,9310}	0,8080 {0,9806}
<i>2-Tupel</i>	0,4400 {0,5395}	0,4880 {0,5396}	0,6160 {0,7857}	0,7520 {0,8469}
<i>3-Tupel</i>	0,5200 {0,5684}	0,4560 {0,4769}	0,5600 {0,6306}	0,7280 {0,7583}
<i>4-Tupel</i>	0,4960 {0,5039}	0,4400 {0,4444}	0,5280 {0,5366}	0,6880 {0,6936}

Tabelle 20: *FPR* und *FNR* Image V,  $normDiffSum \leq ut$ , unterschiedliche Fenstergrößen

	<i>FPR</i>		<i>FNR</i>	
<b>TW</b>	<b>64.000</b>	<b>128.000</b>	<b>64.000</b>	<b>128.000</b>
<i>1-Tupel</i>	0,3009 {0,4698}	0,4832 {0,5780}	0,4750 {0,8636}	0,6615 {0,9556}
<i>2-Tupel</i>	0,4727 {0,5672}	0,6196 {0,6635}	0,4375 {0,6250}	0,5167 {0,6458}
<i>3-Tupel</i>	0,5327 {0,5690}	0,5955 {0,6129}	0,4024 {0,4521}	0,4677 {0,5000}
<i>4-Tupel</i>	0,4054 {0,4054}	0,5375 {0,5432}	0,3875 {0,3875}	0,4355 {0,4426}

## 7.2 Alignments - unveränderte Daten

### 7.2.1 Globale Alignments

Nachdem für jedes Template mit Hilfe der orientierenden Suche die wahrscheinlichste Lokalisation im Image bestimmt ist, erfolgt in einem zweiten Schritt die Berechnung globaler Alignments. Der entsprechende *Matlab*-Code ist Anlage G zu entnehmen (❖). Anknüpfend an die Ausführungen zur Größe des Untersuchungsfensters in Abschnitt 6.2.4 wird die Berechnung der globalen Alignments mit einer Fenstergröße von 64.000 Basen durchgeführt. Die globalen Scores für das Image V sind beispielhaft auf Basis von Basen-4-Tupeln in Bild 25 dargestellt. Die auf die Dateilänge (Basenanzahl) normierten globalen Scores ergeben die in Tabelle 21 aufgeführten Mittelwerte (mean) und in runden Klammern genannten Standardabweichungen (std).

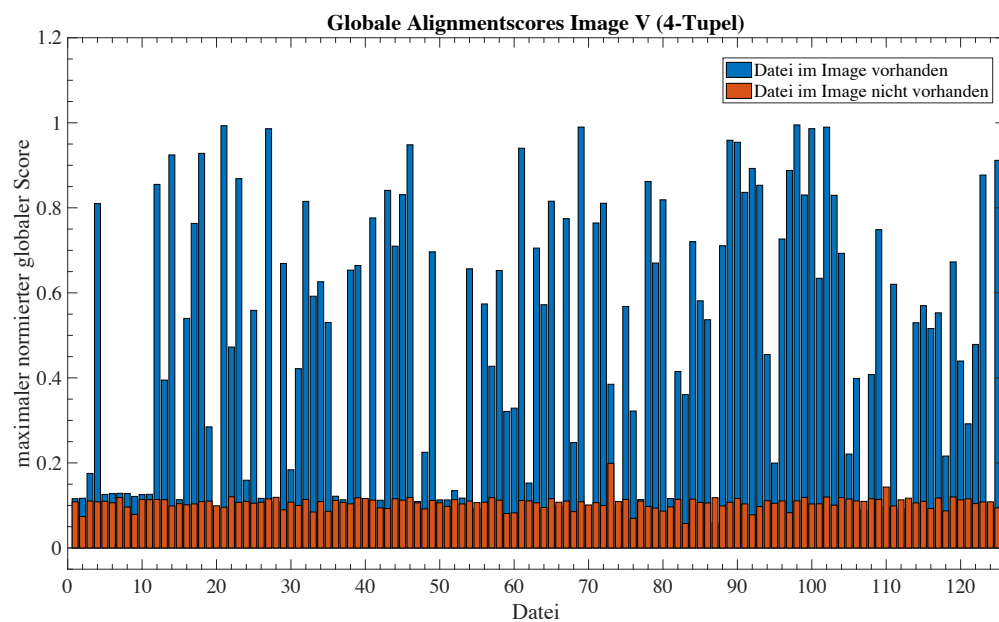


Bild 25: Globale Alignmentscores für das Image V (4-Tupel)

Die Angaben zu den Tupellängen beziehen sich auf die Ergebnisse der orientierenden Suche (*normDiffSum*).

Tabelle 21: Normierte globale Scores - unveränderte Daten

	Normierter Globaler Score		U-Test [ <i>p</i> -Wert]
	Templates im Image V <sup>+</sup>	Templates <u>nicht</u> im Image V <sup>-</sup>	
<b>Image L</b>	<b>25 Dateien</b>	<b>99 Dateien</b>	
<i>1-Tupel</i>	0,1054 (0,0237)	0,1023 (0,0238)	0,6765
<i>2-Tupel</i>	0,1113 (0,0272)	0,1088 (0,0713)	0,8469
<i>3-Tupel</i>	0,1110 (0,0272)	0,1040 (0,0309)	0,5837
<i>4-Tupel</i>	0,1110 (0,0272)	0,1050 (0,0309)	0,2703
<b>Image V</b>	<b>125 Dateien</b>	<b>125 Dateien</b>	
<i>1-Tupel</i>	0,1295 (0,1778)	0,1032 (0,0122)	0,0015
<i>2-Tupel</i>	0,3195 (0,3242)	0,1031 (0,0088)	0,0006
<i>3-Tupel</i>	0,4109 (0,3232)	0,1030 (0,0134)	$1,7588 * 10^{-13}$
<i>4-Tupel</i>	0,4735 (0,3104)	0,1030 (0,0144)	$1,2038 * 10^{-21}$
<b>Image F</b>	<b>500 Dateien</b>	<b>500 Dateien</b>	
<i>1-Tupel</i>	0,1194 (0,1039)	0,1043 (0,0053)	0,0016
<i>2-Tupel</i>	0,1601 (0,1859)	0,1067 (0,0051)	$2,2890 * 10^{-9}$
<i>3-Tupel</i>	0,1766 (0,2059)	0,1068 (0,0051)	$1,0151 * 10^{-9}$
<i>4-Tupel</i>	0,1930 (0,2195)	0,1065 (0,0055)	0,0329

Der Schwellenwert  $lt$  für globale Scores zur Berechnung der  $FPR$  und  $FNR$  wird anhand der Mittelwerte und Standardabweichungen der nicht im Image (n. i. I.) vorhandenen Templates wie folgt definiert:

$$\begin{aligned}
 lt_{global} &= t_{Tupellänge}^{Globaler\ Score\ n.i.I.} \\
 &= mean_{Tupellänge}^{Globaler\ Score\ n.i.I.} + std_{Tupellänge}^{Globaler\ Score\ n.i.I.}
 \end{aligned}
 \quad (7.8)$$

In Anlehnung an den Schwellenwert nach Formel 7.1 schließt der Schwellenwert  $lt$  für globale Scores als unterer Schwellenwert idealerweise alle im Image vorhandenen Templates ein. Im Gegensatz zu den  $normDiffSum$  wird hier mit zunehmender Übereinstimmung der Image- und Templatesequenz kein Wert nahe null, sondern ein globaler Score nahe eins erwartet. Vor diesem Hintergrund wird bei den  $normDiffSum$  der Schwellenwert als oberer und für die globalen Alignments als unterer definiert. Wie für die  $normDiffSum$  gezeigt, wird auch hier der beste Schwellenwert unter entsprechender Anwendung der Formeln 7.2 bis 7.6 ermittelt. Auch hier erweist sich der Mittelwert als günstigste Option. Für die folgenden Berechnungen unter Berücksichtigung eines unteren Schwellenwertes für den globalen Score wird somit

$$lt_{Tupellänge}^{global} = mean_{Tupellänge}^{Globaler\ Score\ n.i.I.} \quad (7.9)$$

gewählt. Die Anwendung der Formel 7.9 führt für die Images L, V und F zu den Ergebnissen der Vierfeldertafel in Tabelle 22.

Tabelle 22: Vierfeldertafel  $globale\ Scores \geq lt$  - unveränderte Daten

Image	Templates im Image $V^+$			Templates <u>nicht</u> im Image $V^-$		
	$L$	$V$	$F$	$L$	$V$	$F$
<b>E<sub>+</sub></b> 1-Tupel	11 {5}	33 {10}	268 {15}	2	37	241
2-Tupel	7 {7}	74 {58}	281 {70}	6	47	227
3-Tupel	15 {15}	96 {86}	291 {103}	12	60	233
4-Tupel	14 {14}	105 {102}	254 {131}	14	55	235
<b>E<sub>-</sub></b> 1-Tupel	7	18	40	77	39	21
2-Tupel	1	5	26	21	17	12
3-Tupel	1	0	11	52	15	14
4-Tupel	0	0	9	57	16	16

In den geschweiften Klammern ist wiederum die Anzahl der korrekten Ergebnisse angegeben. Das heißt, zunächst werden alle im Image vorhandenen Templates ermittelt,

deren wahrscheinlichste Imagelokalisation mit der tatsächlichen eine Überlappung  $> 0\%$  aufweist (Anzahl vor der geschweiften Klammer). Anschließend werden die Dateinamen dieser Templates mit den Namen der Dateien an den berechneten Imagelokalisationen verglichen, bei einer Übereinstimmung handelt es sich um ein korrektes Ergebnis. Die *FPR* und *FNR* für die oben genannten Images sind Tabelle 23 bis Tabelle 25 zu entnehmen.

Tabelle 23: *FPR* und *FNR* Image L, *globale Scores*  $\geq lt_{global}$  - unveränderte Daten

<b>Image L</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,0253 {0,0941}	0,3889 {0,5833}
<i>2-Tupel</i>	0,2222 {0,2222}	0,1250 {0,1250}
<i>3-Tupel</i>	0,1875 {0,1875}	0,0625 {0,0625}
<i>4-Tupel</i>	0,1972 {0,1972}	0,0000 {0,0000}

Tabelle 24: *FPR* und *FNR* Image V, *globale Scores*  $\geq lt_{global}$  - unveränderte Daten

<b>Image V</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,4868 {0,6061}	0,3529 {0,6429}
<i>2-Tupel</i>	0,7344 {0,7875}	0,0633 {0,0794}
<i>3-Tupel</i>	0,8000 {0,8235}	0,0000 {0,0000}
<i>4-Tupel</i>	0,7747 {0,7838}	0,0000 {0,0000}

Tabelle 25: *FPR* und *FNR* Image F, *globale Scores*  $\geq lt_{global}$  - unveränderte Daten

<b>Image F</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,9199 {0,9592}	0,1299 {0,7273}
<i>2-Tupel</i>	0,9498 {0,9733}	0,0847 {0,2708}
<i>3-Tupel</i>	0,9433 {0,9678}	0,0364 {0,0965}
<i>4-Tupel</i>	0,9363 {0,9572}	0,0342 {0,0643}

Im vorhergehenden Abschnitt 7.1 sind die Anzahlen u. a. bei der Anwendung keiner Schwellenwerte aufgeführt. Da die globalen Alignmentscores in Abhängigkeit von der *normDiffSum* ermittelt werden, ist an dieser Stelle eine isolierte Betrachtung der globalen Scores nicht sinnvoll - ohne die Berücksichtigung eines Schwellenwertes für die globalen Scores würden sich die *FPR* und *FNR* aus dem vorherigen Abschnitt ergeben. Die bisherigen Analysen der globalen Scores berücksichtigen nicht die mit der orientierenden



Suche ermittelten Schwellenwerte  $ut$  zur Unterscheidung von im Image vorhandenen und nicht vorhandenen Templates. In Tabelle 26 finden sich die Werte unter der zusätzlichen Bedingung  $normDiffSum \leq ut$  angepasst.

Tabelle 26: Vierfeldertafel  $globale Scores \geq lt, normDiffSum \leq ut$  - unveränderte Daten

<b>Image</b>	<b>Templates im Image <math>V^+</math></b>			<b>Templates <u>nicht</u> im Image <math>V^-</math></b>		
	<b><math>L</math></b>	<b><math>V</math></b>	<b><math>F</math></b>	<b><math>L</math></b>	<b><math>V</math></b>	<b><math>F</math></b>
<b>E<sub>+</sub></b> 1-Tupel	7 {2}	25 {7}	180 {10}	1	26	129
2-Tupel	5 {5}	49 {42}	151 {50}	2	19	82
3-Tupel	8 {8}	55 {52}	152 {90}	1	3	60
4-Tupel	8 {8}	56 {55}	133 {107}	1	0	17
<b>E<sub>-</sub></b> 1-Tupel	7	17	35	75	30	15
2-Tupel	0	2	16	5	6	7
3-Tupel	0	0	10	0	6	6
4-Tupel	0	0	1	0	0	3

Die hieraus errechneten  $FPR$  und  $FNR$  sind Tabelle 27 bis Tabelle 29 zu entnehmen.

Tabelle 27:  $FPR$  und  $FNR$  Image  $L$ ,  $globale Scores \geq lt, normDiffSum \leq ut$  - unveränderte Daten

<b>Image <math>L</math></b>	<b><math>FPR</math></b>	<b><math>FNR</math></b>
1-Tupel	0,0132 {0,0741}	0,5000 {0,7778}
2-Tupel	0,2857 {0,2857}	0,0000 {0,0000}
3-Tupel	1,0000 {1,0000}	0,0000 {0,0000}
4-Tupel	1,0000 {1,0000}	0,0000 {0,0000}

Tabelle 28:  $FPR$  und  $FNR$  Image  $V$ ,  $globale Scores \geq lt, normDiffSum \leq ut$  - unveränderte Daten

<b>Image <math>V</math></b>	<b><math>FPR</math></b>	<b><math>FNR</math></b>
1-Tupel	0,4643 {0,5946}	0,4048 {0,7083}
2-Tupel	0,7600 {0,8125}	0,0392 {0,0455}
3-Tupel	0,3333 {0,5000}	0,0000 {0,0000}
4-Tupel	- {1,0000}	0,0000 {0,0000}

Tabelle 29: *FPR* und *FNR* Image F, *globale Scores*  $\geq lt$ , *normDiffSum*  $\leq ut$  - unveränderte Daten

Image F	<i>FPR</i>	<i>FNR</i>
1-Tupel	0,8958 {0,9522}	0,1628 {0,7778}
2-Tupel	0,9214 {0,9632}	0,0958 {0,2424}
3-Tupel	0,9091 {0,9531}	0,0617 {0,1000}
4-Tupel	0,8500 {0,9348}	0,0075 {0,0093}

In den Tabellen aufgeführte „-“ entsprechen in *Matlab* dem Berechnungsergebnis *NaN* (Not a Number; Division durch null).

### 7.2.2 Lokale Alignments

Nachdem für jedes Template mit Hilfe der orientierenden Suche die wahrscheinlichste Lokalisation im Image bestimmt ist, erfolgt in einem weiteren Schritt die Berechnung lokaler Alignments. Der entsprechende *Matlab*-Code ist Anlage G zu entnehmen (♦). Die Berechnungen erfolgen mit einer Fenstergröße von 64.000 Basen. Die lokalen Scores sind für das Image V beispielhaft auf Basis von Basen-4-Tupeln in Bild 26 dargestellt.

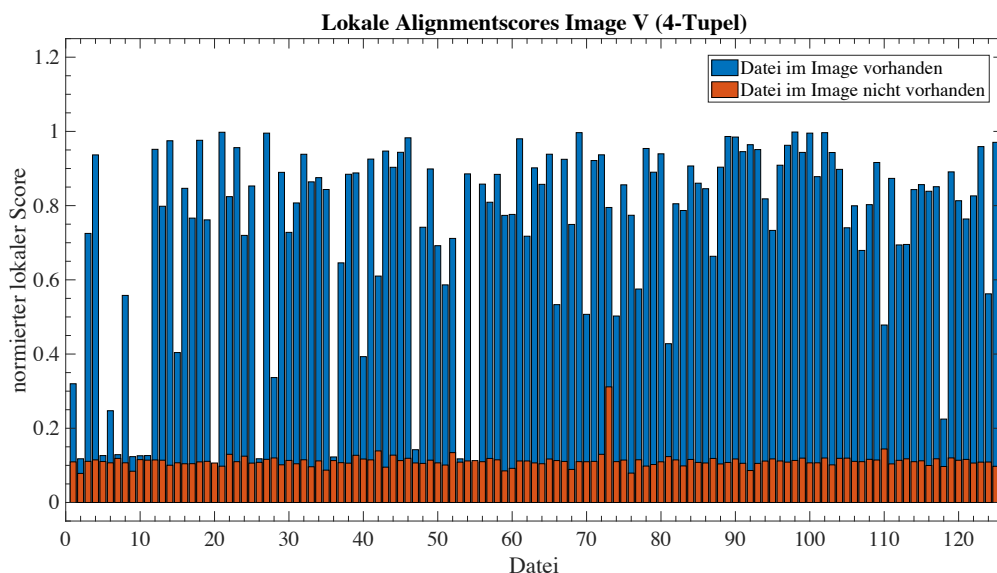


Bild 26: Lokale Alignmentsscores für das Image V (4-Tupel)

Die statistischen Kennzahlen der lokalen Scores, normiert auf die Dateilänge [Basen], ergeben die Werte in Tabelle 30. Die Angaben zu den Tupellängen beziehen sich auf die Ergebnisse der orientierenden Suche.

Tabelle 30: Normierte lokale Scores - unveränderte Daten

Normierter Lokaler Score			U-Test [ <i>p</i> -Wert]
Templates im Image $V^+$		Templates <u>nicht</u> im Image $V^-$	
Image L	25 Dateien	99 Dateien	
1-Tupel	0,1122 (0,0305)	0,1070 (0,0104)	0,2703
2-Tupel	0,2031 (0,1732)	0,1165 (0,0825)	0,0212
3-Tupel	0,2357 (0,1835)	0,1125 (0,0304)	0,0022
4-Tupel	0,2548 (0,1940)	0,1134 (0,0303)	0,0016
Image V	125 Dateien	125 Dateien	
1-Tupel	0,1683 (0,1983)	0,1076 (0,0087)	0,4852
2-Tupel	0,4431 (0,3771)	0,1072 (0,0079)	$3,6154 * 10^{-11}$
3-Tupel	0,6200 (0,3292)	0,1082 (0,0204)	$4,3145 * 10^{-29}$
4-Tupel	0,7205 (0,2655)	0,1089 (0,0209)	$7,5337 * 10^{-38}$
Image F	500 Dateien	500 Dateien	
1-Tupel	0,1271 (0,1214)	0,1069 (0,0042)	0,0001
2-Tupel	0,1938 (0,2326)	0,1084 (0,0048)	$1,6302 * 10^{-14}$
3-Tupel	0,2419 (0,2694)	0,1086 (0,0048)	$2,1052 * 10^{-26}$
4-Tupel	0,3005 (0,2955)	0,1086 (0,0050)	$1,8636 * 10^{-37}$

Legt man den unteren Schwellenwert für die lokalen Scores entsprechend der Formel 7.9 als

$$lt_{Tupell\ddot{a}nge}^{local} = mean_{Tupell\ddot{a}nge}^{Lokaler\ Score\ n.i.I.} \quad (7.10)$$

fest, ergeben sich die in der folgenden Tabelle aufgeführten Werte.

Tabelle 31: Vierfeldertafel  $lokale\ Scores \geq lt_{local}$  - unveränderte Daten

		Templates im Image $V^+$			Templates <u>nicht</u> im Image $V^-$		
Image		<i>L</i>	<i>V</i>	<i>F</i>	<i>L</i>	<i>V</i>	<i>F</i>
<b>E<sub>+</sub></b>	1-Tupel	7 {3}	49 {11}	258 {15}	2	34	219
	2-Tupel	10 {10}	87 {61}	294 {82}	2	46	226
	3-Tupel	15 {15}	112 {96}	336 {144}	6	47	233
	4-Tupel	16 {16}	120 {117}	354 {234}	7	40	232
<b>E<sub>-</sub></b>	1-Tupel	5	17	35	62	36	16
	2-Tupel	0	4	27	3	14	10
	3-Tupel	0	0	11	32	8	14
	4-Tupel	0	0	8	28	11	15

In den geschweiften Klammern ist wiederum die Anzahl der korrekten Ergebnisse angegeben. Das heißt, zunächst werden alle im Image vorhandenen Templates ermittelt, deren wahrscheinlichste Imagelokalisation mit der tatsächlichen eine Überlappung  $> 0\%$  aufweist (Anzahl vor der geschweiften Klammer). Anschließend werden die Dateinamen dieser Templates mit den Namen der Dateien an den berechneten Imagelokalisationen verglichen, bei einer Übereinstimmung handelt es sich um ein korrektes Ergebnis. Daraus lassen sich die *FPR* und *FNR* berechnen (Tabelle 32 bis Tabelle 34).

Tabelle 32: *FPR* und *FNR* Image L, *lokale Scores*  $\geq lt_{local}$  - unveränderte Daten

<b>Image L</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,0313 {0,0882}	0,4167 {0,6250}
<i>2-Tupel</i>	0,4000 {0,4000}	0,0000 {0,0000}
<i>3-Tupel</i>	0,1579 {0,1579}	0,0000 {0,0000}
<i>4-Tupel</i>	0,2000 {0,2000}	0,0000 {0,0000}

Tabelle 33: *FPR* und *FNR* Image V, *lokale Scores*  $\geq lt_{local}$  - unveränderte Daten

<b>Image V</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,4857 {0,6667}	0,2576 {0,6071}
<i>2-Tupel</i>	0,7667 {0,8372}	0,0440 {0,0615}
<i>3-Tupel</i>	0,8546 {0,8873}	0,0000 {0,0000}
<i>4-Tupel</i>	0,7843 {0,7963}	0,0000 {0,0000}

Tabelle 34: *FPR* und *FNR* Image F, *lokale Scores*  $\geq lt_{local}$  - unveränderte Daten

<b>Image F</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,9319 {0,9665}	0,1195 {0,7000}
<i>2-Tupel</i>	0,9576 {0,9777}	0,0841 {0,2477}
<i>3-Tupel</i>	0,9433 {0,9681}	0,0317 {0,0710}
<i>4-Tupel</i>	0,9393 {0,9591}	0,0221 {0,0331}

Im vorhergehenden Abschnitt 7.1 sind die Anzahlen bei der Anwendung keiner Schwellenwerte aufgeführt. Da die lokalen Alignmentsscores in Abhängigkeit von der *normDiffSum* ermittelt werden, ist an dieser Stelle eine isolierte Betrachtung der lokalen Scores nicht sinnvoll - ohne die Berücksichtigung eines Schwellenwertes für die lokalen Scores würden sich die *FPR* und *FNR* aus dem vorherigen Abschnitt ergeben. Die

bisherigen Analysen der lokalen Scores berücksichtigen nicht die mit der orientierenden Suche ermittelten Schwellenwerte zur Unterscheidung von im Image vorhandenen und nicht vorhandenen Templates. In Tabelle 35 finden sich die Werte unter der zusätzlichen Bedingung  $normDiffSum \leq ut$  angepasst.

Tabelle 35: Vierfeldertafel  $lokale\ Scores \geq lt_{local}$ ,  $normDiffSum \leq ut$  - unveränderte Daten

<b>Image</b>	<b>Templates im Image <math>V^+</math></b>			<b>Templates <u>nicht</u> im Image <math>V^-</math></b>		
	<b><i>L</i></b>	<b><i>V</i></b>	<b><i>F</i></b>	<b><i>L</i></b>	<b><i>V</i></b>	<b><i>F</i></b>
<b>E<sub>+</sub> 1-Tupel</b>	4 {1}	34 {7}	173 {10}	1	24	124
<b>2-Tupel</b>	6 {6}	53 {46}	158 {57}	2	17	78
<b>3-Tupel</b>	11 {11}	58 {56}	172 {111}	1	3	55
<b>4-Tupel</b>	13 {13}	61 {60}	183 {158}	1	0	16
<b>E<sub>-</sub> 1-Tupel</b>	5	17	32	61	27	12
<b>2-Tupel</b>	0	1	16	1	5	5
<b>3-Tupel</b>	0	0	10	0	3	6
<b>4-Tupel</b>	0	0	0	0	0	3

Die aus der oben aufgeführten Vierfeldertafel errechneten Werte *FPR* und *FNR* sind in Tabelle 36 bis Tabelle 38 zusammengestellt.

Tabelle 36: *FPR* und *FNR* Image L,  $lokale\ Scores \geq lt_{local}$ ,  $normDiffSum \leq ut$  - unveränderte Daten

<b>Image L</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<b>1-Tupel</b>	0,0161 {0,0615}	0,5556 {0,8333}
<b>2-Tupel</b>	0,6667 {0,6667}	0,0000 {0,0000}
<b>3-Tupel</b>	1,0000 {1,0000}	0,0000 {0,0000}
<b>4-Tupel</b>	1,0000 {1,0000}	0,0000 {0,0000}

Tabelle 37: *FPR* und *FNR* Image V,  $lokale\ Scores \geq lt_{local}$ ,  $normDiffSum \leq ut$  - unveränderte Daten

<b>Image V</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<b>1-Tupel</b>	0,4706 {0,6539}	0,3333 {0,7083}
<b>2-Tupel</b>	0,7727 {0,8276}	0,0185 {0,0213}
<b>3-Tupel</b>	0,5000 {0,6250}	0,0000 {0,0000}
<b>4-Tupel</b>	- {1,0000}	0,0000 {0,0000}

Tabelle 38: *FPR* und *FNR* Image F, *lokale Scores*  $\geq lt_{local}$ , *normDiffSum*  $\leq ut$  - unveränderte Daten

<b>Image F</b>	<b><i>FPR</i></b>	<b><i>FNR</i></b>
<i>1-Tupel</i>	0,9118 {0,9599}	0,1561 {0,7619}
<i>2-Tupel</i>	0,9398 {0,9728}	0,0920 {0,2192}
<i>3-Tupel</i>	0,9016 {0,9508}	0,0549 {0,0826}
<i>4-Tupel</i>	0,8421 {0,9318}	0,0000 {0,0000}

In den Tabellen aufgeführte „–“ entsprechen in *Matlab* dem Berechnungsergebnis *NaN* (Not a Number; Division durch null).

### 7.2.3 Suche auf Basis globaler und lokaler Alignments

Das Vorgehen der orientierenden Suche mit Hilfe von Basenhäufigkeiten soll in einem ersten Schritt möglichst zeitsparend relevante Imagelokalisationen identifizieren. In einem zweiten Schritt dienen die Alignments der Bestätigung oder Ablehnung dieser Imagelokalisationen. Das schrittweise Vorgehen scheint gerechtfertigt, bedingt aber, dass die Alignments die Ergebnisse nur eingeschränkt verbessern bzw. beeinflussen können. Eine falsch identifizierte Imagelokalisation kann als solche erkannt, aber nicht korrigiert, werden. In diesem Abschnitt wird die Suche ohne den Einsatz von Basenhäufigkeiten direkt mit dem Alignieren von Image- und Templatesequenz durchgeführt. Da dies sehr ressourcenaufwändig - insbesondere zeitintensiv - ist, wird ein Abschnitt des Images V gewählt und mit zehn Templates (fünf vorhanden und fünf nicht im Image vorhanden) untersucht. Der Beginn (Imagestart) der Untersuchung wird als Vielfaches der Fenstergröße von 64.000 Basen gewählt, um den Beginn der Untersuchung ab der ersten Base zu simulieren - Start im Image bei Base 92.032.000. Das Imageende wird für die Untersuchung ebenso als Vielfaches der Fenstergröße gewählt (Base 94.016.000), so dass in dem Imageabschnitt die fünf vorhandenen Templates zu finden sind. Die Suche auf Basis von Alignments ist in Bild 27 visualisiert und orientiert sich an Bild 14.

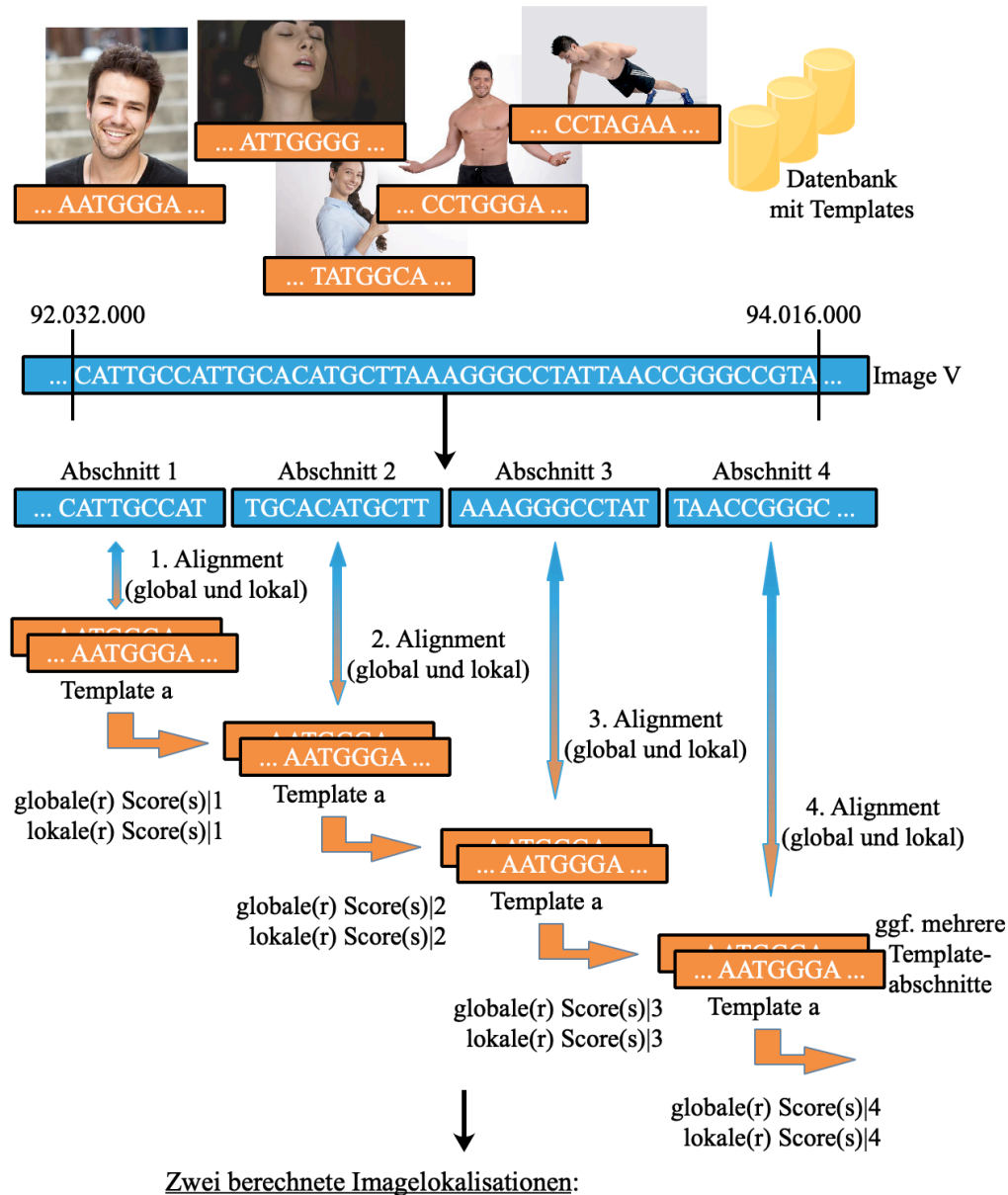


Bild 27: Suche auf Basis von Alignments

Das *Matlab*-Skript in Anlage H implementiert die oben beschriebene Vorgehensweise, die hier als Pseudocode vorgestellt wird (Symbole kennzeichnen die entsprechende Codestelle in der Anlage):

```

array_image      = aktuell zu untersuchendes Image;
array_templates = Templates für die Untersuchung;
window           = Größe des Untersuchungsfensters 64.000 Basen
                    (hardwarebestimmte maximale Größe zur
                    Alignmentberechnung);
Lade das aktuell zu untersuchende Image;
for Elemente in array_templates
  Lade ein Template;
  while maximum noch nicht gefunden
    norm_global_scorei = global_scorei / window; ❖
    norm_local_scorei = local_scorei / window; ⌘
    maximum_global_scorei = Speichere den größten globalen Alignmentsscore
    und den zugehörigen Image-/Templatebereich;-
    maximum_local_scorei = Speichere den größten lokalen Alignmentsscore
    und den zugehörigen Image-/Templatebereich;
    Verschiebe den Start des Untersuchungsfensters nach Formel 6.2 (Shift); ▽
    Durchlaufe erneut das Image und finde ggf. maximum...i+1 > maximum...i;
    wenn maximum...i+1 < maximum...i → Schleifenabbruch;

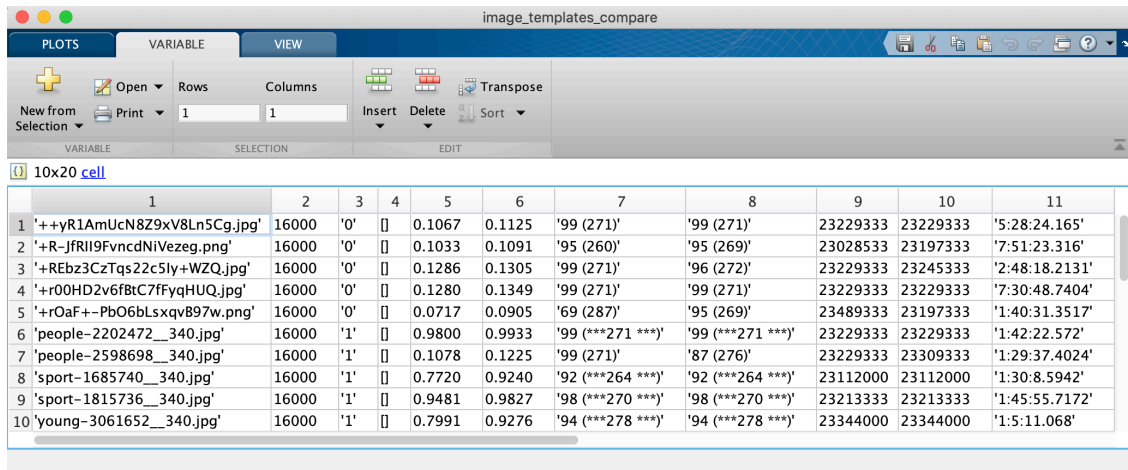
```

Auch hier wird das Untersuchungsfenster mit jedem Durchlauf verschoben (Shift), um eine möglichst optimale Fensterposition zur tatsächlichen Imagelokalisation zu erhalten. Dieser Shift berechnet sich nach Formel 6.2. Die Abbruchbedingung ergibt sich nach:

$$\begin{aligned}
 & \max(\text{globaler Score}_i) \leq \\
 & \max [\max(\text{globaler Score}_1), \dots, \max (\text{globaler Score}_{i-1})] \text{ und} \\
 & \max(\text{lokaler Score}_i) \leq \\
 & \max [\max(\text{lokaler Score}_1), \dots, \max (\text{lokaler Score}_{i-1})]
 \end{aligned} \tag{7.11}$$

Die Fenstergröße bedingt eine Aufteilung des Images und der Templates in Untersuchungsabschnitte mit einer Größe von 64.000 Basen, wobei jeder Templateabschnitt mit jedem Imageabschnitt kombiniert wird. Diese Kombination erfolgt auch bei der orientierenden Suche mit Hilfe von Basenhäufigkeiten, nicht jedoch für die im zweiten Schritt stattfindende Berechnung der Alignments. Die Suche (ausschließlich) auf Basis von Alignments verfügt nur über einen Schritt, vergleichbar dem Häufigkeitsvergleich bei der orientierenden Suche, so dass hier entsprechend die Kombination der Abschnitte stattfindet. Das Suchergebnis der 10 Templates im Image V ist in Bild 28 gezeigt.





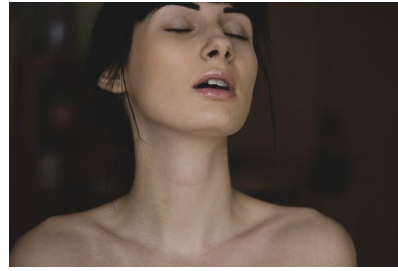
	1	2	3	4	5	6	7	8	9	10	11
1	'++yR1AmUcN8Z9xV8Ln5Cg.jpg'	16000	'0'	□	0.1067	0.1125	'99 (271)'	'99 (271)'	23229333	23229333	'5:28:24.165'
2	'+R-JFRlI9FvncdNiVezeg.png'	16000	'0'	□	0.1033	0.1091	'95 (260)'	'95 (269)'	23028533	23197333	'7:51:23.316'
3	'+REbz3CzTqs22c5ly+WZQ.jpg'	16000	'0'	□	0.1286	0.1305	'99 (271)'	'96 (272)'	23229333	23245333	'2:48:18.2131'
4	'+r00HD2v6fBtC7fFyqHUQ.jpg'	16000	'0'	□	0.1280	0.1349	'99 (271)'	'99 (271)'	23229333	23229333	'7:30:48.7404'
5	'+rOaf+-PbO6bLsxqvB97w.png'	16000	'0'	□	0.0717	0.0905	'69 (287)'	'95 (269)'	23489333	23197333	'1:40:31.3517'
6	'people-2202472__340.jpg'	16000	'1'	□	0.9800	0.9933	'99 (***271 ***)'	'99 (***271 ***)'	23229333	23229333	'1:42:22.572'
7	'people-2598698__340.jpg'	16000	'1'	□	0.1078	0.1225	'99 (271)'	'87 (276)'	23229333	23309333	'1:29:37.4024'
8	'sport-1685740__340.jpg'	16000	'1'	□	0.7720	0.9240	'92 (***264 ***)'	'92 (***264 ***)'	23112000	23112000	'1:30:8.5942'
9	'sport-1815736__340.jpg'	16000	'1'	□	0.9481	0.9827	'98 (***270 ***)'	'98 (***270 ***)'	23213333	23213333	'1:45:55.7172'
10	'young-3061652__340.jpg'	16000	'1'	□	0.7991	0.9276	'94 (***278 ***)'	'94 (***278 ***)'	23344000	23344000	'1:5:11.068'

Bild 28: Suche auf Basis von Alignments (Image V)

In der ersten Spalte sind die Templatennamen aufgeführt. Es folgt die Größe des Untersuchungsfensters in Byte (Basenzahl / 4). Die Kennzeichnung für vorhandene („1“) und nicht vorhandene („0“) Templates ist in Spalte drei dokumentiert. Die maximalen normierten globalen bzw. lokalen Alignmentsscores sind den Spalten fünf bzw. sechs zu entnehmen. Übereinstimmungen der berechneten Lokalisationen mit den tatsächlichen Imagelokalisationen sind in den Spalten sieben (globales Alignment) und acht (lokales Alignment) aufgeführt und korrekt identifizierte Dateien mit „\*\*\*“ markiert. Es folgen in den Spalten neun und 10 die unter Berücksichtigung des Shifts berechneten Startbytes der Imagelokalisationen für die globalen bzw. lokalen Alignments. In der letzten Spalte ist die Zeitdauer für die Suche aufgeführt (Stunden:Minuten:Sekunden). Die Suche dauert für die 10 Templates in einem Imageabschnitt der Länge 1.984.000 Basen ca. 35 Stunden (zum Vergleich: die Untersuchung des kompletten Images V mit Hilfe von Basenhäufigkeiten und anschließenden Alignmentberechnungen dauert für 250 Templates und der Tupellänge vier ca. 20 Stunden). Die normierten Alignmentsscores für die im Image vorhandenen Templates sind mit über 0,70 (global und lokal) deutlich höher als für die nicht vorhandenen bzw. nicht korrekt identifizierten Templates mit Werten kleiner als 0,15. Die Definition eines Schwellenwertes mit einer hohen Trennschärfe fällt hier leicht (z. B. 0,50 für globale und lokale Alignmentsscores) und führt zu vier von fünf korrekt identifizierten Dateien, die vorhandene Datei *people-2598698\_\_340.jpg* wird fälschlicherweise der Datei *people-2202472\_\_340.jpg* zugeordnet. Hierbei handelt es sich um die beiden folgenden Bilder:



people-2202472\_\_340.jpg



people-2598698\_\_340.jpg (korrektes Bild)

Die fünf nicht vorhandenen Templates würden unter Berücksichtigung eines Schwellenwertes alle korrekt negativ bewertet. Ansonsten ist in Bild 28 erkennbar, dass die nicht vorhandenen Templates hohe Lokalisationsübereinstimmungen mit anderen Imagedateien aufweisen, z. B. wird dem Template in Zeile 1 mit einer 99%igen Übereinstimmung die Imagedatei mit der Nr. 271 (*people-2202472\_\_340.jpg*) zugeordnet. Die Analyse mit Hilfe lokaler Alignments ergibt im Vergleich zu den globalen Alignments keine Verbesserung des Untersuchungsergebnisses.

In Bild 29 ist zum Vergleich das Ergebnis der orientierenden Suche mit Hilfe von Basenhäufigkeiten gezeigt. Ab der Tupellänge drei ergibt sich zur Suche auf Basis von Alignments ein vergleichbares Bild. Es wird der gleiche Imageabschnitt mit einer Fenstergröße von 64.000 Basen (16.000 Bytes gemäß Spalte zwei) abgefahren.

</

Bild 29: Orientierende Suche mit Hilfe von Basenhäufigkeiten (3-Tupel, Image V)

In Spalte vier sind wiederum die zugeordneten Imagedateien (Nummer innerhalb der Klammern) mit der prozentualen Übereinstimmung der ermittelten mit den tatsächlichen Imagelokalisationen aufgeführt (Wert vor der Klammer). In den Spalten fünf und sechs finden sich die maximalen globalen bzw. lokalen Alignmentsscores. Hierbei handelt es

sich um die höchsten Werte der Alignments mit der Größe des Untersuchungsfensters. Bei der orientierenden Suche mit Hilfe von Basenhäufigkeiten werden die ermittelten wahrscheinlichsten Imagelokalisationen mit globalen und lokalen Alignments weiter untersucht. Die hier für die Auswertung relevanten Alignmentsscores sind aber die Scoresummen der einzelnen Fensterpositionen. Ein Template der Größe 128.000 Basen wird für das Alignment in zwei Abschnitte zu je 64.000 Basen unterteilt, so dass sich der „Gesamtscore“ als Scoresumme der beiden Untersuchungsabschnitte ergibt. Die normierten Differenzsummen folgen in Spalte sieben. Spalte acht zeigt die Differenzsummen. Der Startpositionen [Byte] im Image bzw. im Template sind den Spalten neun und 10 zu entnehmen. Abschließend sind in den Spalten elf bis dreizehn die Untersuchungszeiten für die *normDiffSum* (Spalte elf), die globalen (Spalte zwölf) und lokalen (Spalte dreizehn) Alignments angegeben.

### 7.3 Untersuchung fragmentierter Daten

Die Suche nach fragmentierten Daten erfolgt mit den Images L, V und F, die mit einem *Python*-Programm in der Weise modifiziert werden, dass mit Nullen überschriebene Abschnitte fester Länge (250.000 Byte) in definierten Abständen (3.000.000 Byte) entstehen (Anlage F).<sup>21</sup> Zusätzlich erfolgt das manuelle Überschreiben von Dateiinhalten mit Nullen im Image „L Fragments“ (Anlage B). Die so bearbeiteten Images werden dann in eine Basensequenz zur weiteren Untersuchung in *Matlab* transkribiert. Die Untersuchungen erfolgen grundsätzlich mit der Templatelänge als Größe des Untersuchungsfensters - wie bei der Suche nach unveränderten Dateien - und der halben Templatelänge, auf Grund des hohen zeitlichen Aufwandes unterbleibt die Suche mit weiteren Fenstergrößen im Image „L Fragments“ und den modifizierten Images. Die Untersuchungsergebnisse mit unterschiedlichen Fenstergrößen werden „nur“ für das Image I anschließend weiter unten beschrieben. Eine Auflistung der Mittelwerte und Standardabweichungen für die normierten Differenzsummen, globalen und lokalen Scores unterbleibt, da bei den nicht bekannten Fragmentgrößen valide Werte, aus denen beispielsweise Schwellenwerte abgeleitet werden könnten, nicht zu erwarten sind.

---

<sup>21</sup> Die Parameter *size\_gap* für die Abschnittlänge und *distance\_gap* für den Abstand in der entsprechenden *Python*-Klasse *ReadImage\_CreateFragments.py* können beliebig gewählt werden. Die hier gewählten Parameterwerte sind zufällig ausgewählt und produzieren, wie auch andere Werte, Fragmente unterschiedlicher Größe.

Entscheidend ist die Ermittlung der korrekt identifizierten Fragmente. Eine Zusammenfassung der Untersuchungsergebnisse bezüglich des Images „L Fragments“ und der modifizierten Images L, V und F ist in Tabelle 39 angegeben. Die im Rahmen der Arbeit genutzte orientierende Suche und Berechnung der Alignments erfolgen templatebasiert. Von den in den Images fragmentierten Dateien sind diese von Interesse, die auch als Template verwendet werden. Damit gibt es in jedem Image relevante Fragmente (rel. Fragm.) - Templates, die im Image fragmentiert vorhanden sind - und nicht relevante Fragmente - Dateien oder Dateifragmente, die nicht als Suchmuster bzw. Template verwendet werden. Korrekt identifizierte Fragmente können auf Grund der bekannten Imageinhalte leicht gezählt werden, daraus ergibt sich dann die Anzahl der nicht erkannten Fragmente. Da die Fragmentgrößen a priori nicht bekannt sind, wird das Untersuchungsfenster das jeweilige Dateifragment in unterschiedlichen Anteilen überdecken. Als Grundlage der Auswertung wird eine Überlappung der mit der orientierenden Suche ermittelten Imagelokalisation mit der tatsächlichen von  $> 0\%$  vorausgesetzt. Die Identifizierung von im Image nicht vorhandenen Templates mit im Image vorhandenen fragmentierten Templates und einer Überlappung von  $> 0\%$  wird als falsch-positives Untersuchungsergebnis gewertet. Die Anzahl der korrekt-negativen Ergebnisse muss immer null sein, da nicht vorhandene Fragmente auch nicht korrekt nicht erkannt werden können. Damit ergibt sich nach Formel 6.4 immer eine *FPR* von 1,00 und wird hier nicht gesondert aufgeführt.

Tabelle 39: Ergebnisse der bioinformatischen Fragmentsuche, Fenstergröße = Templatelänge / halbe Templatelänge

<b>Image</b>	<b>Erkannte Fragmente</b>			
	<i>L Fragments</i>	<i>L modifiziert</i>	<i>V modifiziert</i>	<i>F modifiziert</i>
rel. Fragm.	10	4	4	44
<b>E+</b> 1-Tupel	2 / 0	1 / 1	0 / 0	0 / 1
2-Tupel	5 / 7	1 / 1	0 / 0	0 / 14
3-Tupel	6 / 6	1 / 3	0 / 2	1 / 17
4-Tupel	7 / 6	1 / 3	0 / 2	1 / 18

Die sich auf Basis der Werte in Tabelle 39 ergebenden *FNR* sind in Tabelle 40 zusammengestellt.

Tabelle 40: *FNR* der bioinformatischen Fragmentsuche, Fenstergröße = Templatelänge / halbe Templatelänge

Image	<i>FNR</i>			
	<i>L Fragmente</i>	<i>L modifiziert</i>	<i>V modifiziert</i>	<i>F modifiziert</i>
1-Tupel	0,80 / 1,00	0,75 / 0,75	1,00 / 1,00	1,00 / 0,98
2-Tupel	0,50 / 0,30	0,75 / 0,75	1,00 / 1,00	1,00 / 0,68
3-Tupel	0,40 / 0,40	0,75 / 0,25	1,00 / 0,50	0,98 / 0,61
4-Tupel	0,30 / 0,40	0,75 / 0,25	1,00 / 0,50	0,98 / 0,59

Die Fragmentsuche kann auch zum Auffinden von Dateiinhalten im Slack Space genutzt werden. Hierzu wird das Image I - vergleichbar mit Image L - manuell in der Weise bearbeitet, dass mit Hilfe von *iHex* Slack Space und ggf. angrenzende freie, nicht allozierte Cluster mit Dateiinhalten gefüllt werden (Anlage B). In Bild 30 sind drei Cluster skizziert, wobei die ersten beiden Cluster der Datei X zugewiesen sind.

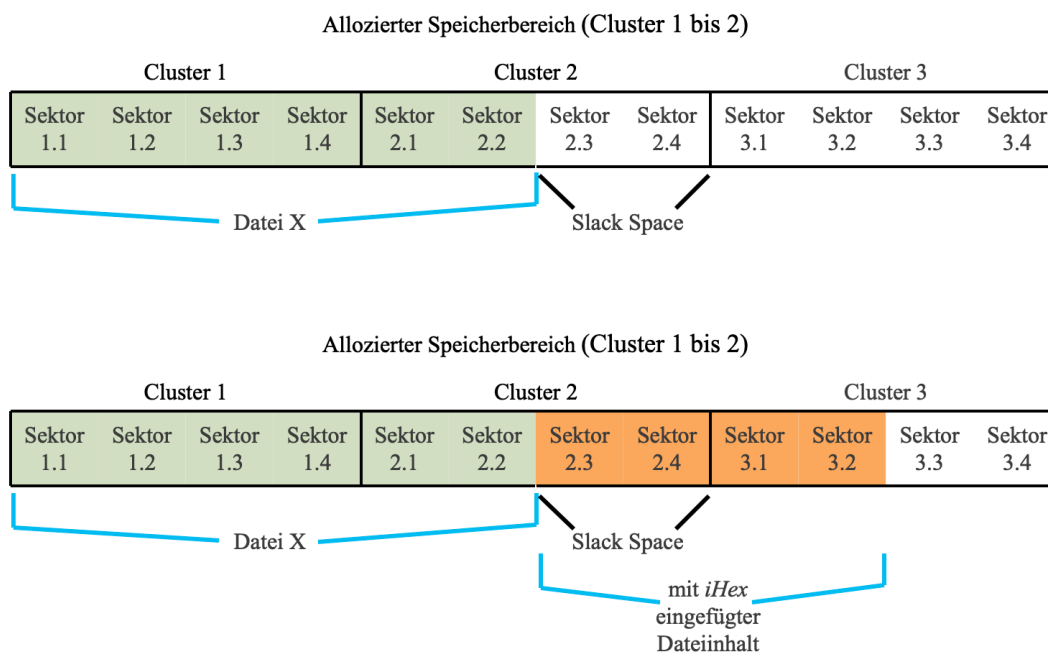


Bild 30: Allozierte Speicherbereiche, Slack Space und Bearbeitung des Images I

Die Datei X nimmt jedoch nur einen Teil des zweiten Clusters in Anspruch, die beiden Sektoren 2.3 und 2.4 bleiben frei (Slack Space). Der angrenzende dritte Cluster ist frei (nicht alloziert). Die manuelle Bearbeitung des Images I mit *iHex* ist beispielhaft im unteren Teil von Bild 30 gezeigt. Bei einer Clustergröße von 512 Bytes, wie bei den

verwendeten Images, beträgt die Größe des Slack Spaces weniger als 512 Bytes. Bezogen auf die transkribierten Images ergibt sich damit eine maximale Fenstergröße für die Untersuchung des Slack Spaces von weniger als  $4 * 512 \text{ Bytes} = 2.048 \text{ Basen}$ . Die hier vorgestellten Vorgehensweisen zur Datensuche unterscheiden nicht zwischen allozierten und nicht allozierten Speicherbereichen, auch eine dedizierte Betrachtung von Slack Space erfolgt nicht. Die Untersuchung wird auf Grund der in Tabelle 51 (Anlage B) dargestellten Fragmentgrößen im Image I mit unterschiedlichen Fenstergrößen durchgeführt. Die Ergebnisse sind Tabelle 41 zu entnehmen. Den unterschiedlichen Fenstergrößen [Basen] sind die Fragmente gegenübergestellt, in eckigen Klammern hinter dem Dateiname ist die Dateigröße [Basen] angegeben.

Tabelle 41: Anzahl korrekt identifizierter Fragmente für das Image I - kein Schwellenwert

Fenstergröße [Basen]	2.500	5.000	10.000	20.000	30.000	40.000
<b>indian.jpg [326.544]</b>						
Fragment 1: 8.960 Basen	28 <sup>2T, 3T, 4T</sup>	56 <sup>2T, 3T, 4T</sup>	-	-	-	-
<b>sheep-1642874_1280.jpg [671.684]</b>						
Fragment 2: 45.776 Basen	-	-	-	-	-	-
Fragment 3: 3.912 Basen	63 <sup>2T, 3T, 4T</sup>	-	-	-	-	-
<b>sunset-1759716_1280.jpg [1.101.424]</b>						
Fragment 4: 14.256 Basen	18 <sup>2T, 3T, 4T</sup>	35 <sup>2T, 3T, 4T</sup>	70 <sup>2T, 3T, 4T</sup>	-	-	-

Eine positive Identifizierung ist durch die Überlappung der mit der orientierenden Suche ermittelten mit der tatsächlichen Lokalisation in Prozent gekennzeichnet. „18<sup>2T, 3T, 4T</sup>“ bedeutet, dass bei der Untersuchung mit den Tupellängen zwei, drei und vier eine Überlappung der ermittelten mit der tatsächlichen Imagelokalisation in Höhe von 18 % vorliegt. Die dazu hochgestellten Angaben 1T, ..., 4T beziehen sich auf die entsprechenden Tupellängen (Tupellänge eins = 1T, ..., vier = 4T). Ergänzend wird in Tabelle 42 die Anzahl falsch-positiver Ergebnisse für jede verwendete Fenstergröße in

Abhängigkeit von den gewählten Schwellenwerten aufgezeigt ( $t$  = threshold,  $ut$  = oberer Schwellenwert für *normDiffSum*,  $lt$  = unterer Schwellenwert für Alignmentscores).

Tabelle 42: Anzahl falsch-positiver Ergebnisse für das Image I - Abhängigkeit von Schwellenwerten

Fenstergröße	kein $t$	$ut$	$ut$ und $lt_{global}$	$ut$ und $lt_{local}$
<b>2.500</b>	$1^{2T}, 4^{3T, 4T}$	-	-	-
<b>5.000</b>	$3^{2T}, 4^{3T, 4T}$	$1^{2T, 3T}$	-	-
<b>10.000</b>	-	-	-	-
<b>20.000</b>	$2^{4T}$	$1^{4T}$	-	-
<b>30.000</b>	$1^{4T}$	-	-	-
<b>40.000</b>	$1^{4T}$	-	-	-

In Tabelle 43 sind die Anzahlen der relevanten Fragmentgrößen in Bezug zur ursprünglichen Dateigröße für die verwendeten Images zusammengestellt. Beispielsweise wird das Fragment 1 aus Tabelle 41 mit einer Größe von 8.960 Basen in den Cluster  $\leq 10$  % der Tabelle 43 eingeordnet, da die ursprüngliche Dateigröße 326.544 Basen beträgt ( $8.960 \text{ Basen} / 326.544 \text{ Basen} \approx 0,027 = 2,7 \%$ ).

Tabelle 43: Anzahl der relevanten Fragmentgrößen zur ursprünglichen Dateigröße

Fragmentgröße/ ursprünglichen Dateigröße [%]	Anzahl der relevanten Fragmente				
	<i>L</i> Fragments	<i>L</i> modifiziert	<i>V</i> modifiziert	<i>F</i> modifiziert	<i>I</i>
<b><math>\leq 10</math></b>		1	1	2	4
<b>11 - 20</b>				3	
<b>21 - 30</b>				3	
<b>31 - 40</b>				5	
<b>41 - 50</b>				7	
<b>51 - 60</b>	2	1	2	4	
<b>61 - 70</b>		1	1	9	
<b>71 - 80</b>				6	
<b>81 - 90</b>		1		2	
<b>91 - 100</b>	8			3	

Die hier beschriebene Suche nach Dateifragmenten ist grundsätzlich der im vorhergehenden Abschnitt dargestellten Suche nach vollständigen, unveränderten Daten gleichzusetzen. Ein Fragment kann in diesem Sinne als eine vollständige Datei geringerer

Größe angesehen werden. Vor diesem Hintergrund und unter Berücksichtigung des hohen zeitlichen Aufwandes unterbleibt die Suche allein auf Basis globaler und lokaler Alignments, wie zuvor in Abschnitt 7.2.3.

#### **7.4 Untersuchung ähnlicher Daten (u. a. Multiple Alignments)**

In den folgenden Abschnitten wird die Suche nach ähnlichen Dateien in den Images „Similar Files“ und „Versions“ beschrieben. Hierzu erfolgt in einem ersten Schritt die Imageanalyse mit der oben beschriebenen orientierenden Suche mit Hilfe von Basenhäufigkeiten, um dann in einem weiteren Schritt die Suche „nur“ auf Basis von globalen und lokalen Alignments durchzuführen.

##### **7.4.1 Suche nach ähnlichen Dateien mit Hilfe von Basenhäufigkeiten**

In den vorherigen Abschnitten werden dedizierte Templates für die Suche verwendet. Sollten sich im Image „neue“, noch nicht bekannte, nicht in einer Datenbank als Template hinterlegte Dateien bzw. Dateifragmente befinden, sinkt die Wahrscheinlichkeit für eine erfolgreiche Suche. Der in diesem Zusammenhang hier betrachtete Lösungsansatz versucht, ähnliche Dateien aufzufinden. Mit Hilfe sogenannter Multipler Alignments können mehrere (ähnliche) Sequenzen aligniert und anschließend in eine einzelne Konsensussequenz überführt werden. Von jeweils 10 ähnlichen Bildern in den Kategorien:

1. beliebige Frauengesichter
2. beliebige Männergesichter
3. kriterienbasierte Auswahl von Frauengesichtern

wird eine Konsensussequenz gebildet. Hier sei auf Bild 10 verwiesen, in dem das Grundprinzip Multipler Alignments visualisiert ist. Da man auf Grund der begrenzten Hardwareressourcen nicht beliebig große Dateien über die gesamte Länge alignieren kann (Überschreitung der „zulässigen“ Speichergröße), ist das Multiple Alignment blockweise durchzuführen. Die einzelnen blockbezogenen Konsensussequenzen werden zu einer Sequenz konkateniert, wobei die Länge dieser Sequenz im Wesentlichen durch die kleinste Dateigröße bestimmt ist. Die grundlegende Vorgehensweise bei der Erzeugung von Konsensussequenzen ist in Bild 31 dargestellt. Insgesamt soll für drei Sequenzen ein Multiples Alignment erstellt und auf dieser Grundlage eine Konsensussequenz gebildet



werden. Bei dem oben erwähnten blockweisen Alignieren sind auf Grund der kürzesten Sequenz drei insgesamt drei vollständige Blöcke möglich (die Blockgröße beträgt 64.000 Basen). Ein vierter Block ist kürzer und erfasst die Sequenzen eins und zwei sowie die restlichen Basen der dritten Sequenz. Ein fünfter Block würde nur noch die Sequenzen eins und zwei erfassen und damit nur einen Teil der Sequenzinformationen beinhalten. Diese Möglichkeit - das Multiple Alignment ggf. mit weniger Sequenzen fortzusetzen - wird hier nicht realisiert. Die Konsensussequenzen werden ihrerseits dann als Templates zum Abgleich mit dem Image verwendet.

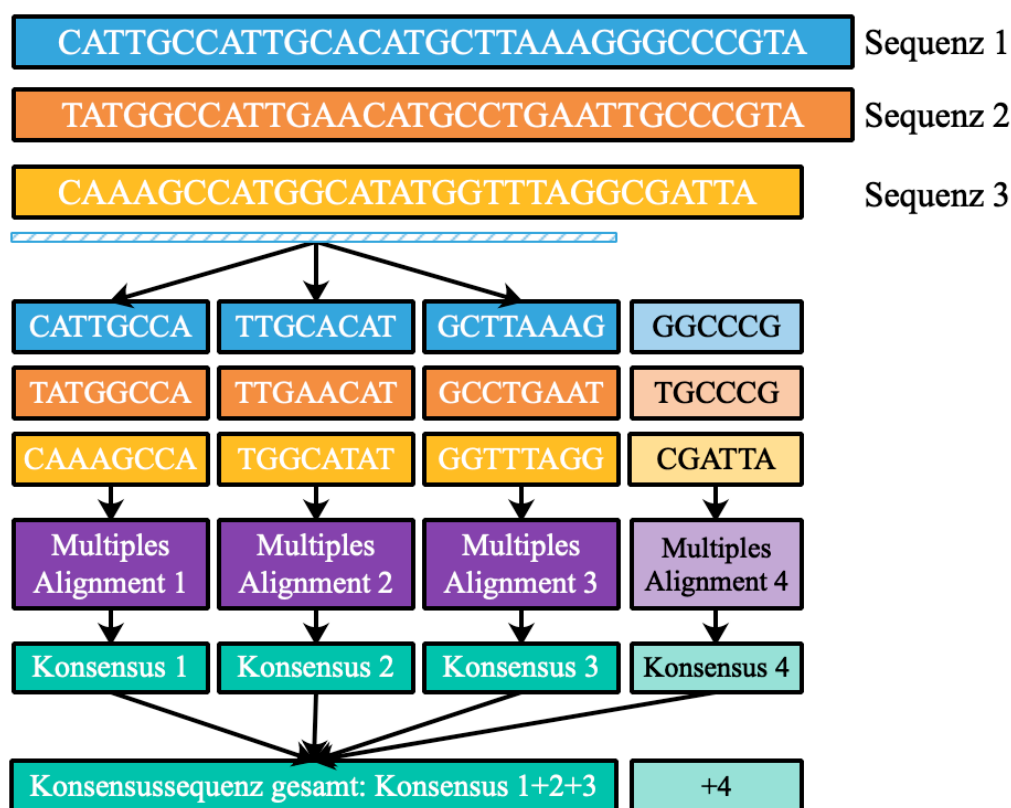


Bild 31: Erzeugung einer Konsensussequenz

Nutzt man das *Matlab*-Skript aus Anlage G zur Dokumentation weiterer möglicher Lokalisationen, indem man von der kleinsten normierten Differenzsumme eine prozentuale Abweichung nach oben zulässt, können möglicherweise ähnliche Bilder identifiziert werden. Die für die Untersuchung des Images „Similar Files“ eingesetzten Dateien *man-67467\_\_340.jpg* und *girl-3023853\_\_340.jpg* sind in Bild 32 aufgeführt.



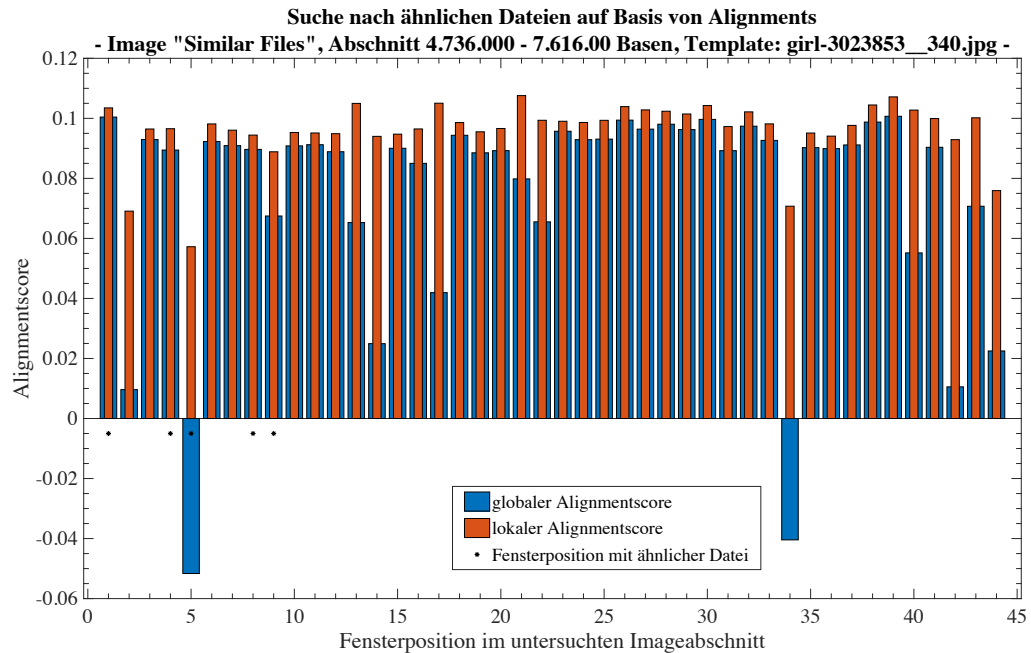
Bild 32: Bilder für die Suche nach Ähnlichkeiten im Image „Similar Files“

Hinzu kommen die Untersuchungen von drei mit Hilfe Multipler Alignments erstellter Konsensussequenzen. Um den Dateiinhalt (Basen) aller, auch größerer Dateien für die Konsensussequenz nutzen zu können, ist das Angleichen der Dateigrößen möglich und wird für die folgenden Untersuchungen umgesetzt. Hierzu werden die Bilder mit Hilfe der Software *Vorschau* zuerst in ihrer Größe so verändert, dass sie annähernd die gleiche Dateigröße erreichen. Zum Einsatz kommt zunächst die Erzeugung einer Konsensussequenz ausschließlich mit vollständigen Blöcken gemäß Bild 31 - bedingt durch die Platz- und Zeitkomplexität bei der Erzeugung Multipler Alignments. Das bedeutet, dass möglicherweise ein Block mit einer Größe  $< 64.000$  Basen nicht in die Konsensussequenz einfließt (Konsensus 4 in Bild 31). Die erste Konsensussequenz umfasst Frauengesichter, die zweite Konsensussequenz Männergesichter und die dritte Konsensussequenz entsteht auf einer kriterienbasierten Auswahl von Frauengesichtern (Faces). Die hierfür verwendeten Bilder sind in Anlage I zu finden. Der Abgleich erfolgt mit dem Image „Similar Files“, das insgesamt 110 Bilddateien (96 *jpg*-Dateien und 14 *png*-Dateien) enthält. Weiterhin lässt sich der Imageinhalt in 25 Männer- und 25 Frauengesichter differenzieren. Die übrigen 60 Bilddateien weichen von einer menschlichen Gesichtsdarstellung mehr oder weniger stark ab. Es handelt sich überwiegend um Landschaften und Gegenstände, aber auch zwei Statuenköpfe. Es werden die Frauen- bzw. Männergesichter im Image „Similar Files“ entsprechend mit „*girl*“ bzw. „*man*“ im Dateinamen markiert. Eine erfolgreiche Suche ist dann gegeben, wenn Template- und Imagedateiname den gleichen Markierungstext enthalten. Idealerweise sollte so ein „*girl*“-Template mit allen „*girl*“-Imagedateien eine Überlappung aufweisen, ansonsten liegt ein falsch-negatives Ergebnis vor.

Die Auswertung erfolgt mit den Abweichungen von 10 %, 20 % und 30 %; das heißt, von der ermittelten *normDiffSum* ist eine entsprechende Abweichung nach oben erlaubt. Für jedes Template werden alle ermittelten normierten Differenzsummen mit den zugehörigen Imagelokalisationen zwischengespeichert (unabhängig vom Durchlauf) und anschließend redundante Werte der normierten Differenzsummen für die weitere Auswertung mit den genannten Abweichungen entfernt. Hierdurch wird die Anzahl zusätzlicher Imagelokalisationen reduziert, unter Inkaufnahme des Verlustes von möglicherweise tatsächlichen Imagelokalisationen ähnlicher Dateien. Hinzu kommt, dass die Anzahl zusätzlicher Lokalisationen auf maximal 100 beschränkt wird. Nur so ist unter der bereits mehrfach angesprochenen Zeit- und Platzkomplexität eine praktikable Untersuchung des Images möglich. Die mit Hilfe der normierten Differenzsummen und zulässigen Abweichungen berechneten Dateiübereinstimmungen zeigen überwiegend absolute Werte nahe null. Valide Mittelwerte und Standardabweichungen der normierten Differenzsummen sowie globalen und lokalen Alignmentsscores mit statistisch signifikanten Unterschieden zwischen „ähnlichen“ und „nicht ähnlichen“ Dateien lassen sich nicht ermitteln.

#### 7.4.2 Suche nach ähnlichen Dateien auf Basis von Alignments

Die in Abschnitt 7.2.3 beschriebene Vorgehensweise der Suche auf Basis von globalen und lokalen Alignments findet auch hier seine Anwendung. Aus dem Image „Similar Files“ wird der Abschnitt von Base 4.736.000 bis Base 7.616.000 mit den Templates *girl-3023853\_\_340.jpg* (Bild 32) und Konsensussequenz der Frauengesichter untersucht. In dem Imageabschnitt befinden sich drei „girl“- und drei „man“-Imagedateien, zusätzlich vier als gelöscht markierte Dateien. Für jede Position des Untersuchungsfensters (und jeden Durchlauf) werden die berechneten globalen und lokalen Alignmentsscores gespeichert. In Bild 33 ist das Analyseergebnis für das Template *girl-3023853\_\_340.jpg* dargestellt. Die fünf markierten Fensterpositionen (\*) spiegeln die genannten drei ähnlichen Imagedateien wider, die Positionen vier und fünf gehören ebenso wie die Positionen acht und neun zu jeweils einer Imagedatei. Anhand der zugehörigen globalen und lokalen Alignmentsscores scheint eine sichere Identifizierung nicht möglich. Verwendet werden hier die Werte des letzten Durchlaufs (Shift), der für zumindest eine Fensterposition den größten globalen bzw. lokalen Alignmentsscore garantieren soll.

Bild 33: Suche auf Basis von Alignments in einem Imageabschnitt (*girl-3023853\_\_340.jpg*)

Analog zu Bild 33 sind Bild 34 die Suchergebnisse auf Basis der Konsensussequenz zu entnehmen. Auch hier scheinen die globalen und lokalen Alignmentsscores keine sichere Identifizierung zu ermöglichen.

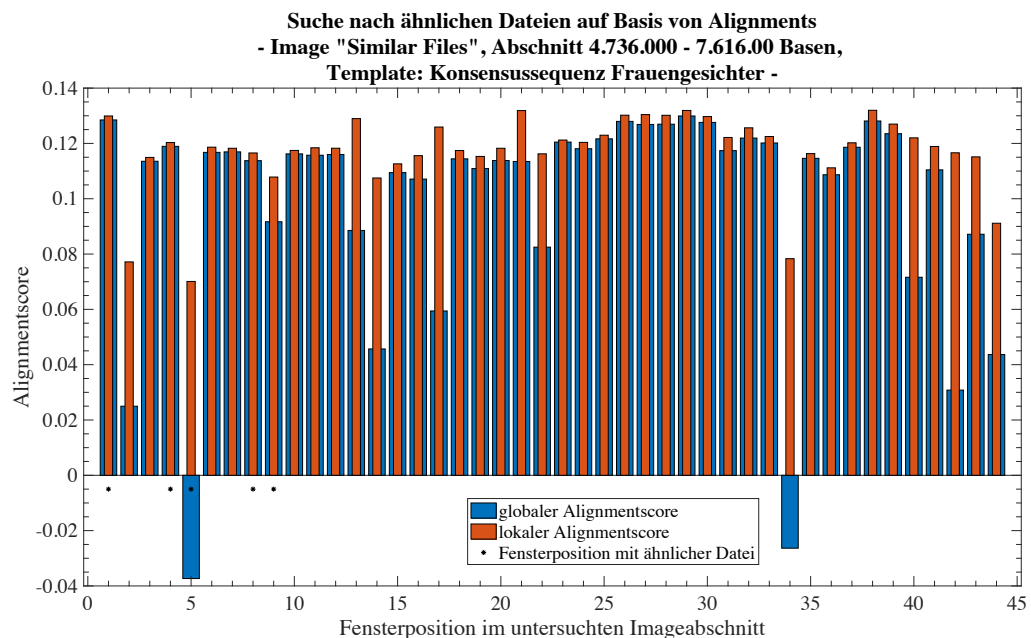


Bild 34: Suche auf Basis von Alignments in einem Imageabschnitt (Konsensussequenz)

Die vergleichbare Untersuchung mit Hilfe von Basenhäufigkeiten (Fenstergröße wie bei der Alignmentberechnung 64.000 Basen und gleicher Imageabschnitt) für das Template *girl-3023853\_\_340.jpg* ergibt zwei identifizierte ähnliche Dateien unter insgesamt 23 errechneten Imagelokalisationen bei einer zulässigen Abweichung von 30 % und einer Tupellänge von zwei. Das Untersuchungsfenster fährt den genannten Imageabschnitt ab und es wird die kleinste normierte Differenzsumme ermittelt. Hiervon ausgehend werden alle Imagelokalisationen gespeichert, für die die normierte Differenzsumme um maximal 30 % vom Minimum nach oben abweicht. In diesem Beispiel fällt auf, dass lokale Alignmentsscores über 0,10 ausschließlich bei den hier auftretenden ähnlichen Dateien vorkommen. Die globalen Scores liegen ausschließlich bei diesen beiden Dateien über 0,095. Insgesamt ist jedoch keine sichere Differenzierung zwischen ähnlichen und nicht ähnlichen Dateien möglich. In diesem Zusammenhang ergibt die Analyse mit der Konsensussequenz als Template keine positiven Ergebnisse.

#### 7.4.3 Suche nach ähnlichen Dateien - „Dateiversionen“

Den Gedanken unterschiedlicher „Dateiversionen“ - wie in Abschnitt 6.4 für *sdhash* beschrieben - aufgreifend, werden die in Bild 35 gezeigten drei Bilder<sup>22</sup> als Templates zur Suche im Image „Versions“ verwendet. In diesem Image befinden sich insgesamt 84 *jpg*-Dateien. Die 14 Originalbilder werden mit Hilfe der Bildbearbeitungssoftware *GIMP* so verändert, dass zusätzlich zum jedem Ausgangsbild fünf Versionen im gleichen *jpg*-Format existieren (Anlage I).

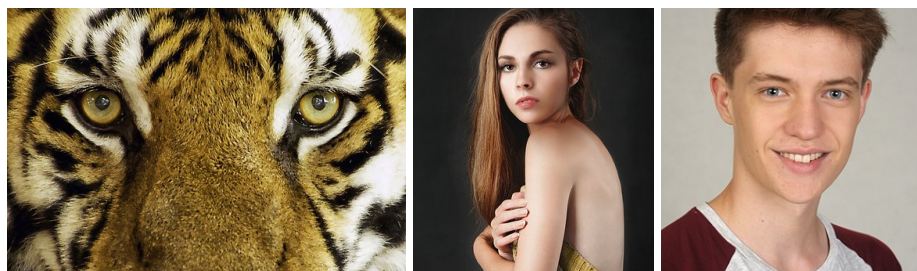


Bild 35: Templates zur Untersuchung des Images „Versions“

<sup>22</sup> Tiger: *ekklp+wFf1QIxkSJG7JfQ.jpg*;  
 Frau: *girl\_TkaAYYBly2lfL4Has37dQ.jpg*;  
 Mann: *man\_Ro5lDhrPYWtU1lRvhSC9g.jpg*.

Die Konsensussequenzen werden für jedes Template aus den drei Dateiversionen „Original“, „NL-Filter“ und „Ölgemälde“ erstellt. Zur Konsensusbildung kommt einerseits die oben beschriebene Vorgehensweise der vollständigen Blöcke - ganzzahliger Anteil der Fensterlängen, der in die kleinste Sequenz für das Multiple Alignment passt - und andererseits die vollständige Ausnutzung der kürzesten Sequenz für das Multiple Alignment (Konsensus 4 in Bild 31) zum Einsatz. Mit Hilfe der Konsensussequenz „Frau“ können ab einer Tupellänge von zwei und einer zulässigen Überschreitung der *normDiffSum* von maximal 20 % bis zu 17 der 36 vorhandenen ähnlichen Dateien (*girl*-Dateien) identifiziert werden. Dies bedeutet „nur“, dass unter den auf 100 beschränkten Imagelokalisationen innerhalb der zulässigen Abweichung ca. die Hälfte der vorhandenen ähnlichen Dateien sicher aufzufinden ist. Eine gezielte Identifizierung gelingt nicht. Ein Vergleich der Untersuchungsergebnisse der unterschiedlich erzeugten Konsensussequenzen zeigt keine wesentliche Verbesserung für die ressourcenintensivere Erstellung unter Ausnutzung der kompletten kürzesten Sequenz. Auch das Template „Konsensussequenz Tiger“ weist mit bis zu sechs identifizierten ähnlichen Dateien ein besseres Ergebnis als das Template „Tiger“ mit maximal einem korrekten Ergebnis auf. Ab einer Tupellänge von zwei und zulässigen Abweichung von 30 % befinden sich fünf oder alle sechs „Dateiversionen“ des Tiger-Bildes unter den Imagelokalisationen. Eine sichere Identifizierung bzw. weitere Eingrenzung der Imagelokalisationen gelingt nicht.

#### 7.4.4 Suche nach ähnlichen Dateien auf Basis von Alignments - „Dateiversionen“

Die Suche nach ähnlichen Dateien auf Basis von globalen und lokalen Alignments mit Hilfe der Templates „Frau“ (Bild 35, Mitte) und „Konsensussequenz Frau“ im Image „Versions“ lässt vergleichbar den Ergebnissen in Abschnitt 7.4.2 keine sichere Identifizierung ähnlicher Dateien anhand der globalen oder lokalen Alignmentsscores zu. Die Untersuchung des Images „Versions“ erfolgt über seine gesamte Länge.

In Bild 36 ist ein Auszug der Untersuchungsergebnisse mit dem Template „Frau“ zu sehen. Ein entsprechender Auszug der Resultate mit dem Template „Konsensussequenz Frau“ ist in Bild 37 dargestellt.

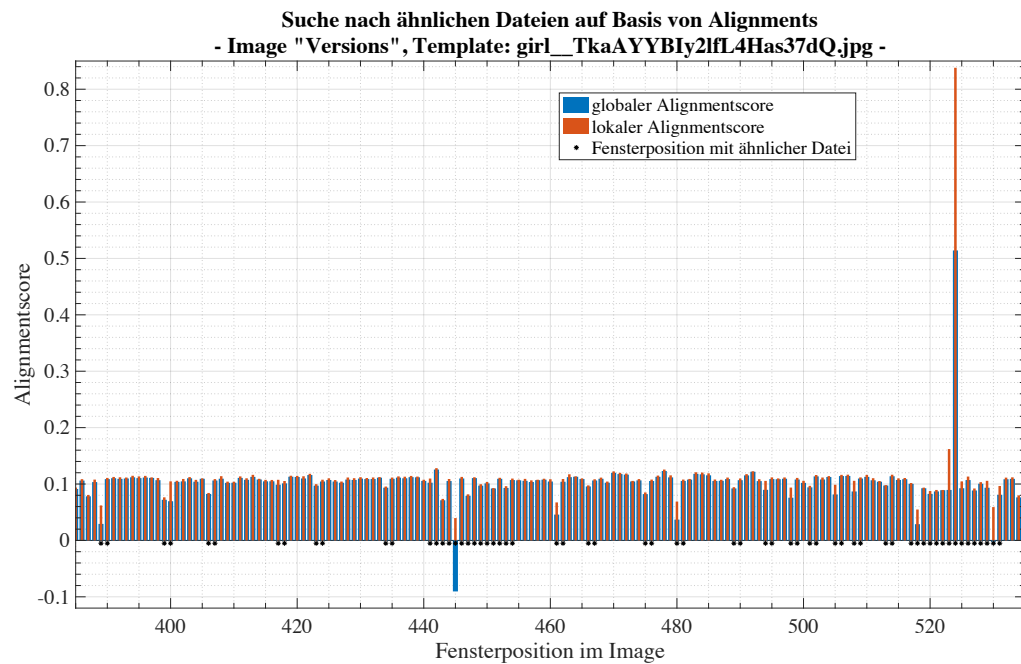


Bild 36: Suche auf Basis von Alignments im Image „Versions“ (*girl\_TkaAYYBIy2lfL4Has37dQ.jpg*)

Die Markierungen \* in Bild 36 und Bild 37 kennzeichnen die Fensterposition bzw. Imagelokalisation mit ähnlichen Dateien (*girl*-Dateien), wobei eine Datei zwei Lokalisationen umfassen kann.

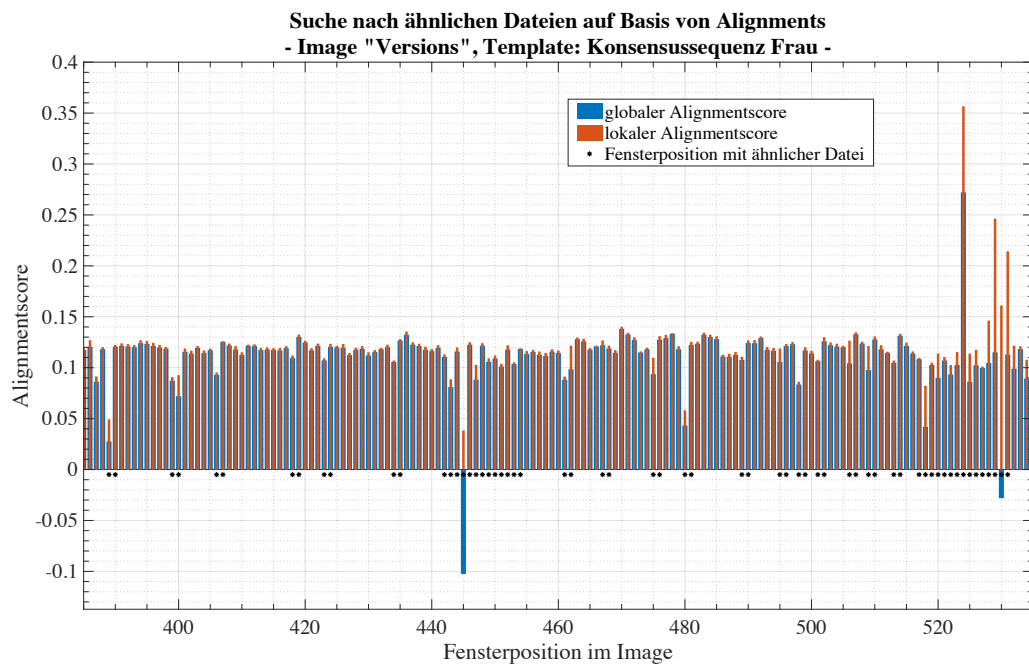


Bild 37: Suche auf Basis von Alignments im Image „Versions“ (Konsensussequenz)

## 8 Vergleichende Betrachtung mit etablierten Verfahren

Die mit bioinformatischen Methoden im vorhergehenden Kapitel ermittelten Untersuchungsergebnisse werden hier mit den Ergebnissen der Forensiktools *Autopsy/Scalpel*, *X-Ways*, *EnCase* und *sdhash* verglichen. Die Forensiktools *Autopsy*, *X-Ways* und *EnCase* analysieren zunächst das Filesystem, können Hashwertvergleiche und auch ein File Carving durchführen. Für so identifizierte Dateien sind u. a. die Imagelokalisation, der zugehörige Hashwert und die Dateigröße abrufbar. Das Programm *sdhash* dient ausschließlich dem Hashwertvergleich. Bezugnehmend auf Kapitel 4 wird die vergleichende Betrachtung mit etablierten Verfahren durchgeführt, die die Bereiche „Filesystem“, „Hashing“ und „File Carving“ abdecken.

### 8.1 Vorüberlegungen, Voruntersuchungen

Vor dem Einsatz der etablierten Verfahren sollen diese in den folgenden Abschnitten hinsichtlich des Umgangs bzw. der wesentlichen Einstellungen kurz vorgestellt werden.

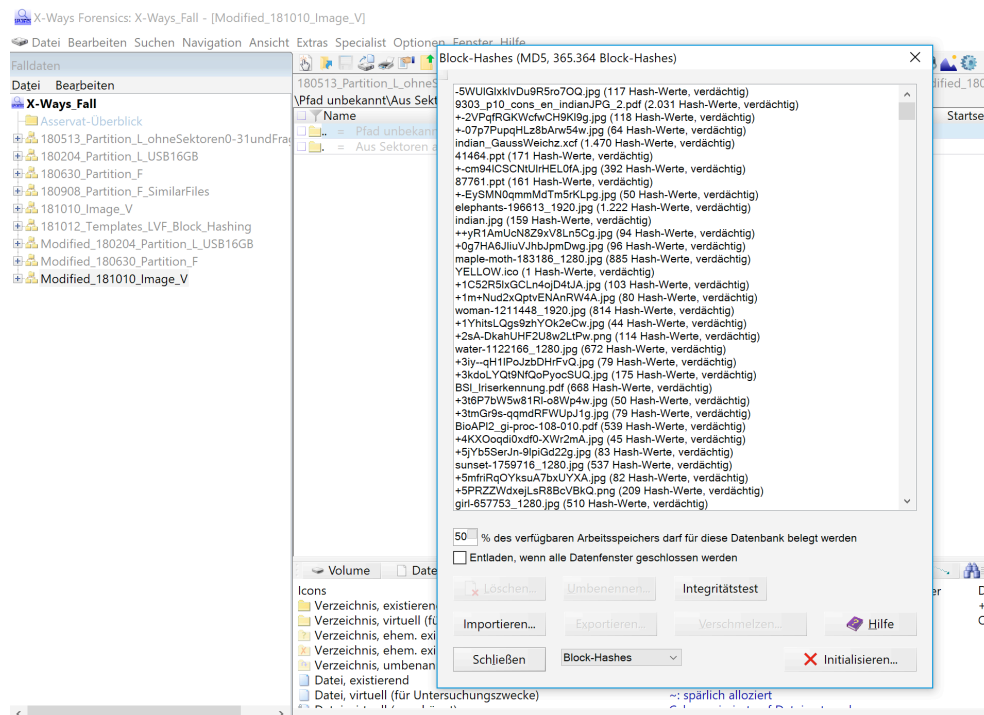
#### 8.1.1 Filesystem

Die Analyse des Filesystems ist eine Standardfunktionalität der Forensiktools *Autopsy*, *X-Ways* und *EnCase*. Eine Auflistung der im Image vorhandenen Dateien setzt ein intaktes Filesystem voraus. Hierbei erfolgt zunächst kein Abgleich der Imagedateien mit z. B. dem Inhalt von Datenbanken mit relevantem Material, ist aber Voraussetzung für weitergehende Analysen in Form des Hashwertvergleiches.

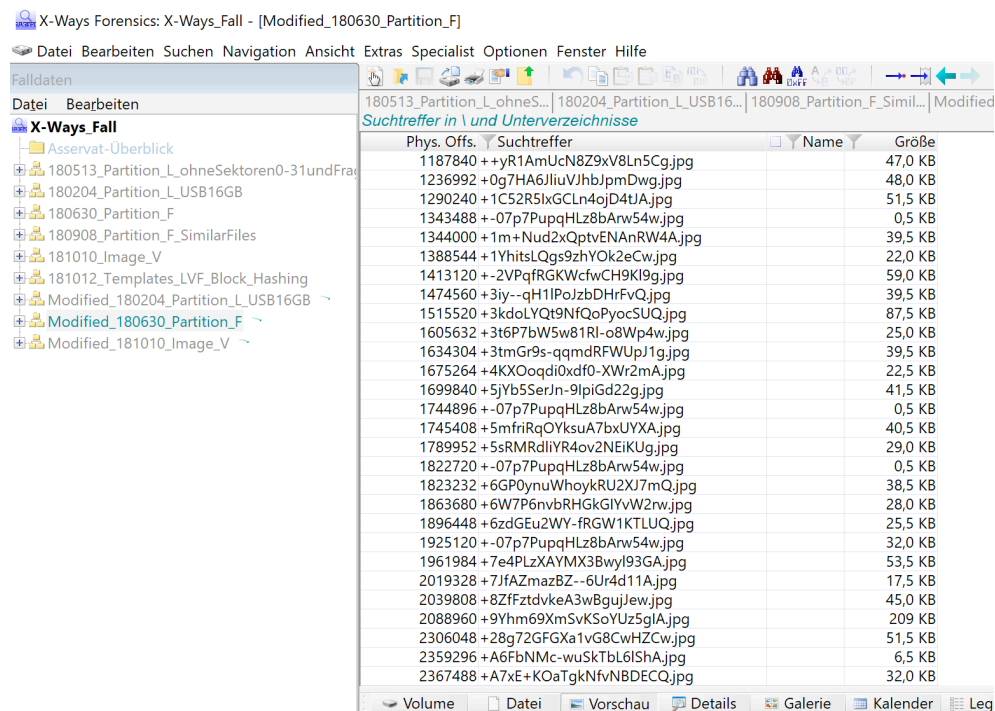
#### 8.1.2 Hashing

Mit *X-Ways* können Fragmente identifiziert werden, indem zuvor Hashwerte einzelner Dateiblöcke (Mindestgröße 512 Bytes; vgl. [17]; S. 144) berechnet und diese zusammen für jede Datei als eigenes Hashset in die Datenbank aufgenommen werden (Bild 38). Nach der erfolgten Analyse erhält der Nutzer eine Auflistung der ermittelten physikalischen Offsets, Größen und zugehörigen Templatennamen für die Suchtreffer (Bild 39).



Bild 38: Hashdatenbank mit Block Hashes in *X-Ways* (Beispiel)

Diese Auflistung kann zur weiteren Verarbeitung exportiert werden und dient als Eingabearray für die Analyse mit den Hilfsprogrammen in Anlage D.

Bild 39: Ergebnisse der Analyse auf Basis von Block Hashes in *X-Ways* (Beispiel)

Das Forensiktool ***sdhash*** kann mit dem Parameter *b* block-aligned Hashwerte, z. B. bei Hauptspeicherabbildern, berechnen und vergleichen. *Sdhash* ist für die Blockgrößen 4 KB bis 16 KB<sup>23</sup> (vgl. [36], S. 15) optimiert. Es ermöglicht grundsätzlich das Auffinden von Fragmenten und ähnlichen Dateien („Dateiversionen“, vgl. [36], S. 3). Bei Dateien größer als 128 MB erfolgt durch *sdhash* automatisch ein Splitting in 128 MB große Abschnitte. Diese Segmentierung kann man über den Parameter *z* beeinflussen und so die Lokalisation einer erfolgreichen Suche bzw. eines erfolgreichen Hashwertvergleichs eingrenzen. Die Segmentgrößen werden mit *z* = 1 MB (kleinste mögliche Segmentgröße) und 2 MB gewählt<sup>24</sup>, der Kommandozeilenbefehl lautet für *b* = 4 und *z* = 1:

*sdhash -b 4 -z 1 -t -l -gr* Eingabe-Ordner > Ausgabe-Ordner

Der Parameter *t* definiert einen Schwellenwert, nur für Scores oberhalb von *t* erfolgt die Ausgabe (der Wert -1 führt dazu, dass alle Ergebnisse ausgegeben werden, siehe hierzu auch Abschnitt 4.3). Die Angabe der Parameter *g* und *r* ermöglicht den Vergleich aller Dateien in einem Ordner. Für weitere Einzelheiten zur Verwendung des Befehls *sdhash* wird auf [36] verwiesen. Bei der Auswertung der Analyseergebnisse mit *sdhash* finden nur Scores > 0 Berücksichtigung.

### 8.1.3 File Carving

Das Softwaretool ***Scalpel*** ist in *The Sleuth Kit/Autopsy* enthalten und kann Dateien unterschiedlichen Typs anhand typischer Signaturen im Dateiheader bzw. -footer aufspüren (File Carving), der Kommandozeilenbefehl lautet:

*scalpel -b -c* [Konfigurationsdatei] *-o* [Ausgabeordner Imagedatei]

Die Angaben in [...] sind entsprechend anzupassen. Der Parameter *b* veranlasst den Algorithmus, auch Suchergebnisse aufzunehmen, bei denen sich der *Footer* („Endemarkierung“ einer Datei) außerhalb der in der Konfigurationsdatei (*scalpel.conf*) aufgeführten maximalen *Carving Size* (Dateigröße) für den jeweiligen Dateityp

---

<sup>23</sup> Standardwert, der grundsätzlich verwendet wird, Abweichungen sind gesondert ausgewiesen; der vorgegebene Schwellenwert zur Anwendung von block-aligned Hashwerten ist 16 MB als Größe der zu analysierenden Datei (vgl. [36], S. 15).

<sup>24</sup> Beim Image L existiert eine *jpg*-Datei mit einer Größe > 1 MB. Bei den Images „Similar Files“ und „Versions“ beträgt die Segmentgröße immer *z* = 1.

befindet<sup>25</sup>, z. B. beträgt die *Carving Size* für den Dateityp *jpg* im Standard von *Scalpel* 200 MB (Bild 40). In den hier verwendeten Images befinden sich deutlich kleinere Dateien, die Konfigurationsdatei wird entsprechend angepasst. *Scalpel* rekonstruiert Dateien, nummeriert diese durch und legt diese in Abhängigkeit vom Dateityp in einem entsprechenden Ergebnisordner ab.

```
[...]
#-----
# GRAPHICS FILES
#-----
[...]
# GIF and JPG files (very common)
# gif y 5000000 \x47\x49\x46\x38\x37\x61 \x00\x3b
# gif y 5000000 \x47\x49\x46\x38\x39\x61 \x00\x00\x3b
# jpg y 200000000 \xff\xd8\xff\xe0\x00\x10 \xff\xd9
# jpg y 200000000 \xff\xd8\xff\xe1 \xff\xd9
#
#
# PNG
# png y 200000000 \x50\x4e\x47? \xff\xfc\xfd\xfe
[...]
```

Bild 40: Auszug aus der Konfigurationsdatei von *Scalpel*

Die bei der Rekonstruktion verwendeten Einstellungen und eine Auflistung der *Scalpel*-Ergebnisse (u. a. Startbyte und Länge) sind in der von *Scalpel* angelegten Datei *audit.txt* gespeichert. Hieraus werden für die weitere Analyse (Anlage D) zwei Vektoren aus Startbyte und Dateilänge gebildet, jeweils für die originären Imagelokalisationen (die Images mit den tatsächlichen Dateilokalisationen und -längen sind bekannt) und die rekonstruierten Dateien. Überdecken sich beide Vektoren teilweise oder vollständig und liegt an der betrachteten Imagelokalisation ein Fragment vor, ergibt sich ein korrekter Fragmentfund. Die Fragmente bzw. Fragmentlokalisationen sind bei den Images L und I durch die manuelle Bearbeitung bekannt und durch das determinierte Überschreiben bei den modifizierten Images können diese automatisiert ermittelt werden. Die Analyse mit *Scalpel* kann durch Einstellungen in der Konfigurationsdatei beeinflusst werden. Jedem

---

<sup>25</sup> Eine Auflistung der zulässigen Parameter zum Aufruf von *Scalpel* erfolgt mit dem Terminal-Befehl *Scalpel -help*.

Dateityp ist eine maximale Dateigröße zugeordnet, innerhalb derer *Scalpel* nach dem Auffinden eines *Headers* den zugehörigen *Footer* vermutet. Die hier durchgeführten Untersuchungen basieren auf zwei Carvinggrößen, die grundsätzlich auch einen Vergleich der etablierten Verfahren untereinander ermöglichen sollen: 1 MB und 5 MB. Auch bietet *X-Ways* die Möglichkeit des File Carvings an. Ein Auszug aus der entsprechenden Konfigurationsdatei, wie bei *Scalpel*, ist in Bild 41 dargestellt.

Description	Extensions	Header	Offset	Footer	Default size	Flags
*** Pictures						
JPEG	JPG;jpeg;jpe;thm;mpo	\xFF\xD8\xFF	[\xC0\xC4\xDB\xDD\ [...]			
PNG	png	\x89PNG\x0D\x0A\x1A\x0A	0	~6	e	
GIF	gif	GIF8[79]a	0	~3	2097152/33554432	
Thumbcache fragment	cmmm	CMMM.	\x00\x00.[^\x00]	0	~84 [...]	
TIFF/NEF/CR2/DNG	tif;tiff;nef;cr2;dng;pef;nrv;arw	[...]				
Bitmap	bmp;dib	BM.....	\x00.\x00....[\x0C\x28\x38\x40\x6C\x7C]\x00\x00[...]			
Paint Shop Pro	psp;PsPImage;pfr	(Paint Shop Pro Im)(~BK\x00)	[...]			
Canon Raw	crw	HEAPCCDR	6	8200000	c	
Adobe Photoshop	PSD;pdd;p3m;p3r;p3l	8BPS	\x00\x01\x00\x00\x00\x00[...]			
[...]						

Bild 41: Konfigurationsdatei für das Carving mit *X-Ways* (Auszug)

Eine entsprechende Ergebnisauflistung für das Image „L Fragments“, bei dem die ersten 32 Sektoren überschrieben und Fragmente erzeugt wurden, ist in Bild 42 zu sehen.

Name	Erw.	Größe	Attr.	Startsektor	Offset im Date	Date	Hash <sup>1</sup> (MD5)	Hash-Set
000001 (Standard 90 Edited up).jpg	jpg	169 KB		1,559			4D90E975EBBCBAC...	
000002 (Adobe Mono 85.59 sym).jpg	jpg	2,1 KB		2,201			E7B27961D9B043E1...	
000003 (Adobe Mono 85.59 sym).jpg	jpg	3,8 KB		2,206			6D80753605F5C6B...	
000004 (Adobe Mono 85.59 sym).jpg	jpg	2,5 KB		2,214			D242503AB690145...	
000005 (Adobe Mono 95.92 sym).jpg	jpg	20,2 KB		2,219			F3BCDD779FD533D...	
000006 (Adobe Mono 85.59 sym).jpg	jpg	7,0 KB		2,260			067992A65A30B5C...	
000007 (Adobe Mono 85.59 sym).jpg	jpg	4,5 KB		2,274			5E2E6C2AC2C30739...	
000008 (Photoshop Web 80).jpg	jpg	16,0 KB		2,303			132FC1DC8ABD660...	
000009 (Photoshop Web 60).jpg	jpg	7,6 KB		2,390			81383EB8C582D83...	
000010 (Photoshop Web 60).jpg	jpg	5,4 KB		2,405			F46DC6F80914460B...	
000011 (Photoshop Web 60).jpg	jpg	5,0 KB		2,416			475A06250623444A...	
000012 (Photoshop Web 80).jpg	jpg	13,9 KB		2,516			0BE1771BC12F7760...	
000013 (Photoshop Web 60).jpg	jpg	6,4 KB		2,677			AE57F1E729873625...	
000014 (Photoshop Mono 10).jpg	jpg	336 KB		3,001			E5E3C34982189544...	
000015 (Photoshop Mono 10).jpg	jpg	64,1 KB		3,674			DB47277A6B4F7918...	
000016 (Photoshop Mono 10).jpg	jpg	24,6 KB		3,920			4E4F18DA0671E585...	
000017 (Photoshop Web 60).jpg	jpg	4,2 KB		3,970			1E2C770A0982669...	
000018 (Photoshop Web 60).jpg	jpg	3,2 KB		3,985			8B19D14537730B9F...	
000019 (Photoshop Web 60).jpg	jpg	4,8 KB		3,992			AFCC20994819119...	
000020 (Photoshop Web 60).jpg	jpg	2,7 KB		4,002			D9578A9B25FA190...	
000021 (Adobe Mono 95.92 sym).jpg	jpg	2,9 KB		4,058			7B9E231BCAE380E...	
000022 (Adobe Mono 95.92 sym).jpg	jpg	2,9 KB		4,066			1CBE6728970C02EE...	
000023 (Adobe Mono 85.59 sym).jpg	jpg	5,2 KB		4,094			5334DA8C89CA3A1...	
000024 (Adobe Mono 85.59 sym).jpg	jpg	5,2 KB		4,105			FF93A2B2EE60A12...	
000025 (Adobe Mono 95.92 sym).jpg	jpg	124 KB		4,148			9472B7FA96A96561...	
000026 (Adobe Mono 95.92 sym).jpg	jpg	123 KB		4,396			A104CA6B06B9C90...	
000027 (Adobe Mono 95.92 sym).jpg	jpg	50,8 KB		4,642			8E84F2DF3FA9C565...	
000028 (Adobe Mono 95.92 sym).jpg	jpg	46,7 KB		4,744			156C341F41870DE...	
000029 (Adobe 95.92 ~x).jpg	jpg	92,7 KB		4,838			35E79AF1A568B8F...	
000030 (Adobe 95.92 ~x).jpg	jpg	92,3 KB		5,023			2C7B88A287E3CAE...	
000031 (Photoshop Web 80).jpg	jpg	69,8 KB		5,224			AB004D62A8AB55F...	

Bild 42: Beispiel für das File Carving mit *X-Ways*

Auch diese Auflistung kann zur weiteren Verarbeitung exportiert werden und dient als Eingabearray für die Analyse mit den Hilfsprogrammen in *Matlab*.

**EnCase** bietet die Möglichkeit des File Carvings, wobei die Signatursauswahl im Standard größer ist als bei *Scalpel* (Bild 43).

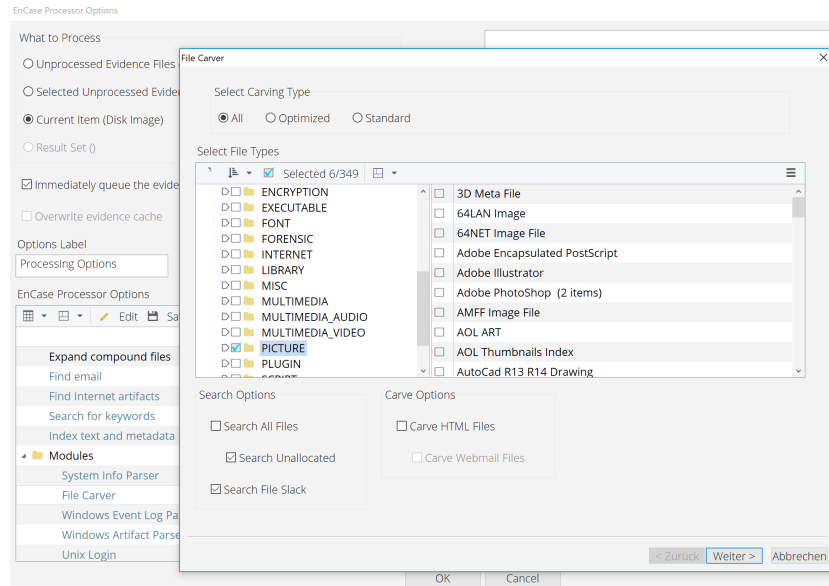


Bild 43: Auswahl der Carving-Typen in *EnCase* (Auszug)

Eine durch den Nutzer beeinflussbare typspezifische Voreinstellung für die Carvinggröße ist jedoch nicht vorgesehen. Jeder Filetyp bzw. jede Signatur verfügt über eine Standardgröße. Sofern im *Header* keine Informationen vorhanden sind, dient grundsätzlich der *Footer* als Begrenzung für das Carving. Fehlen Informationen zur Typgröße werden als Standard-Carvinggröße 4.096 Bytes verwendet (vgl. [20], S. 155ff). Allgemein kann der Nutzer die Ausgabe des Carvings über „Output File Size“ beeinflussen. Die kleinste hierfür definierbare Größe beträgt 1 MB.

## 8.2 Untersuchung unveränderter Daten

Die Ergebnisse der Datensuche nach unveränderten Daten sind in Tabelle 44 dokumentiert. In den geschweiften Klammern ist die Anzahl der Ergebnisse mit einer Überdeckung der berechneten mit der tatsächlichen Imagelokalisation von größer 0 % angegeben. Die Verfahren *Autopsy*, *X-Ways* und *EnCase* arbeiten hier standardmäßig mit der Analyse des Filesystems, spezielle Funktionalitäten in Form des Hashings oder File Carvings werden nicht benötigt. Dem gegenüber bietet *sdhash* „nur“ das Hashing an.

Tabelle 44: Vierfeldertafel für die Datensuche mit etablierten Verfahren - unveränderte Dateien

Image	Templates im Image $V^+$			Templates <u>nicht</u> im Image $V^-$		
	$L$	$V$	$F$	$L$	$V$	$F$
<b>E+</b> Autopsy	25 {25}	125 {125}	500 {500}	0	0	0
X-Ways	25 {25}	125 {125}	500 {500}	0	0	0
EnCase	25 {25}	125 {125}	500 {500}	0	0	0
sdfhash (-z 1)	25 {0}	125 {30}	500 {0}	3	3	0
sdfhash (-z 2)	25 {0}	125 {71}	500 {0}	3	3	0
<b>E-</b> Autopsy	0	0	0	99	125	500
X-Ways	0	0	0	99	125	500
EnCase	0	0	0	99	125	500
sdfhash (-z 1)	0	0	0	96	122	500
sdfhash (-z 2)	0	0	0	96	122	500

In Tabelle 45 sind die  $FPR$  und  $FNR$  etablierter Verfahren den bioinformatischen gegenübergestellt.

Tabelle 45:  $FPR$  und  $FNR$ , etablierte Verfahren  $\leftrightarrow$  Bioinformatik - unveränderte Dateien

	Autopsy	X-Ways	EnCase	sdfhash (-z 1)	sdfhash (-z 2)	Bioinformatik
<b>FPR</b>						
Image $L$	0 {0}	0 {0}	0 {0}	0,03 {0,23}	0,03 {0,23}	0,04 {0,09} Suche 0,01 {0,07} / 0,03 {0,08} Alignment
Image $V$	0 {0}	0 {0}	0 {0}	0,02 {0,45}	0,02 {0,32}	0,27 {0,43} Suche 0,33 {0,50} / 0,33 {0,50} Alignment
Image $F$	0 {0}	0 {0}	0 {0}	0,00 {0,50}	0,00 {0,50}	0,86 {0,93} Suche 0,85 {0,93} / 0,84 {0,93} Alignment
<b>FNR</b>						
Image $L$	0 {0}	0 {0}	0 {0}	0 {-}	0 {-}	0,00 {0,00} Suche 0,00 {0,00} / 0,00 {0,00} Alignment
Image $V$	0 {0}	0 {0}	0 {0}	0 {0}	0 {0}	0,00 {0,00} Suche 0,00 {0,00} / 0,00 {0,00} Alignment
Image $F$	0 {0}	0 {0}	0 {0}	0 {-}	0 {-}	0,01 {0,02} Suche 0,01 {0,01} / 0,00 {0,00} Alignment

Die Spalte „Bioinformatik“ beinhaltet zwei Angaben. Zum einen werden die *FPR* und *FNR* nur unter Anwendung der orientierenden Suche (Basenhäufigkeiten) angegeben, zum anderen unter der zusätzlichen Anwendung von Alignments. Eine vergleichende Betrachtung zu den Ergebnissen der Suche allein auf Basis von Alignments unterbleibt. Die Werte in den geschweiften Klammern spiegeln wiederum die Berechnungen nur auf Basis der korrekten positiven Ergebnisse wider. Das bedeutet, dass sich errechnete und tatsächliche Imagelokalisation überdecken und der Templatename mit dem Namen der an dieser Position befindlichen Imagedatei übereinstimmt (siehe auch Abschnitt 6.3). Aufgeführt sind die besten Ergebnisse, unabhängig von der Tupellänge und Schwellenwerten. Bei den Alignments bezieht sich der erste Wert auf globale und der zweite auf lokale Scores. In der Tabelle aufgeführte „–“ entsprechen der Division durch null, eine Wertangabe ist nicht möglich.

### 8.3 Untersuchung fragmentierter Daten

Die Ergebnisse der Datensuche nach fragmentierten Daten sind in Tabelle 46 dokumentiert. In der letzten Spalte sind die maximal errechneten bioinformatischen Ergebnisse zum Vergleich dargestellt. Das Image „L Fragments“ und die modifizierten Images lassen sich mit *Autopsy* nicht analysieren. Der Analyseversuch des Hinzufügens wird mit der Fehlermeldung:

*„Failed to add data source (critical error encountered). Click below to view the log.  
Add Image Log: Errors occurred while ingesting image  
1. Cannot determine file system type (Sector offset: 0)“*

quittiert. Daher erfolgen die Untersuchungen mit *Scalpel*, das wie *Autopsy* zu *The Sleuth Kit* gehört. Alle untersuchten Images zum Auffinden von Fragmenten zeichnen sich durch fehlende Informationen zum Filesystem aus. Eine standardmäßige Analyse ist so mit den etablierten Verfahren nicht möglich. Der Einsatz weiterer Funktionalitäten in Form des Hashings und File Carvings sind unabdingbar. Die Ergebnisse der Forensiktools werden ggf. so vorverarbeitet, dass zu jeder aufgefundenen Datei deren Lokalisation im Image und Größe in einer Tabelle zusammengefasst werden. Diese Tabelle dient dann der weiteren Bearbeitung in *Matlab*. Aus den Angaben zur Imagelokalisation und Größe wird ein Vektor gebildet und mit dem Vektor aus der tatsächlichen Imagelokalisation und Datei- bzw. Fragmentgröße verglichen.

Tabelle 46: Ergebnisse der Fragmentsuche mit *Scalpel*, *X-Ways*, *EnCase* und *sdhash*

	<i>Scalpel</i> Carving	<i>X-Ways</i> Carving	<i>EnCase</i> Carving	<i>sdhash</i> Hashes	<i>X-Ways</i> Block Hashes	Bioinformatik <sup>26</sup>
	[1 MB / 5 MB]			[z = 1 / 2]		
Image L Fragmente (10 rel. Frag.)	10 / 10	2 / 2	10 / 10	5 / 5	10	≤ 7
Image L modifiziert (4 rel. Frag.)	3 / 4	3 / 3	4 / 4	2 / 2	3	≤ 3
Image V modifiziert (4 rel. Frag.)	4 / 4	1 / 1	4 / 4	2 / 2	4	≤ 2
Image F modifiziert (44 rel. Frag.)	43 / 44	18 / 18	- / - <sup>27</sup>	36 / 39	44	≤ 18

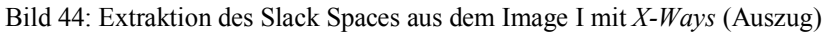
Für jedes etablierte Verfahren existieren verschiedene Einstellungen zur Durchführung des File Carvings. Im Standard gibt es bei *X-Ways* und *EnCase* deutlich mehr zur Verfügung stehende Signaturen als bei *Scalpel*. Die Einstellungen werden für die Untersuchungen so vorgenommen, dass möglichst alle Dateitypen gemäß Tabelle 4 gecarvt werden. Eine Ergänzung der Signaturdateien, z. B. ist der Typ *xcf* (Bildformat der Software *GIMP*) nur in *X-Ways* verfügbar, erfolgt nicht.

Die automatisierte Analyse von Slack Space mit etablierten Verfahren gelingt nur mit *X-Ways*. Hier existiert eine Funktion, die Slack Space-Inhalte extrahiert und in einer Textdatei abgelegt. In Bild 44 ist ein Auszug für das Image I zu sehen. In den dargestellten Clustern-Auszüge drei und vier (Bytes 1.536 bis 2.048) sind gemäß Tabelle 51 Anteile der Datei *sunset-1759716\_1280.jpg* vorhanden.

<sup>26</sup> Die Anzahl variiert in Abhängigkeit von der Tupellänge und der Größe des Untersuchungsfensters (vgl. Tabelle 39).

<sup>27</sup> *EnCase* liefert hier keine Carvingergebnisse.





## 8.4 Untersuchung ähnlicher Daten

101

betrachtet. Die Untersuchung des Images „Similar Files“ mit *sdhash* ergibt für die Blockgrößen 4, 8 und 16 KB und die Segmentgröße 1 MB keinen signifikanten Vergleichswert - ein negatives Analyseergebnis sowohl für den Vergleich zwischen Templates und Image als auch der Imagedateien untereinander. Die zur Erstellung des Images genutzten Dateien werden aber mit einer sehr hohen Wahrscheinlichkeit (Scores > 20) identifiziert. Auch der „normale“ - nicht auf Ähnlichkeiten beruhende - Hashwertvergleich der in Bild 32 gezeigten Templates mit dem Image „Similar Files“ in *X-Ways* blieb erfolglos, es wurde keine „verdächtige“ Datei gefunden. Ebenso verhält es bei *EnCase*.

Mit Hilfe von *sdhash* werden im Image „Versions“ ausschließlich die Originaldateien mit sehr hoher Wahrscheinlichkeit aufgefunden. Der Vergleich der Bilddateien („Dateiversionen“) untereinander ergibt keinen signifikanten Score. Es soll erwähnt werden, dass bei  $b = 16$  der Vergleich von *girl\_TkaAYYBly2lfL4Has37dQ\_75percent.jpg* mit *girl\_5zfFbIfWQjejlOUq-dN0g\_NLFilter.jpg* einen *sdhash*-Score in Höhe von 11 ergibt (eine marginale Wahrscheinlichkeit für deren Übereinstimmung).

Auf eine Untersuchung der Hashwerte mit *EnCase* und *X-Ways* wird auf Grund der Resultate beim Image „Similar Files“ hier verzichtet.

## 9 Diskussion

### 9.1 Orientierende Suche mit Hilfe von Basenhäufigkeiten - unveränderte Daten

Der Einsatz bioinformatischer Methoden startet mit einer orientierenden Suche nach Lokalisationen im Image, die möglicherweise strafrechtlich relevantes Datenmaterial enthalten und für weitere Untersuchungen genutzt werden. Die hier vorgestellte Vorgehensweise ist mit der Behandlung von genetischen Daten vergleichbar, in denen sich bestimmte DNA-Abschnitte (genkodierende Bereiche) in ihrer Häufigkeit sogenannter Codonen - drei Basen (3-Tupel), die eine Aminosäure kodieren - unterscheiden (vgl. [21], S. 482f). In diesem Zusammenhang unterstützen die Ergebnisse von Veenman [43] eine Datendiskriminierung mit Hilfe von Häufigkeitsverteilungen der Bytes in einem oder mehreren Clustern. Bestimmte Dateitypen lassen sich in einer Menge von Dateien grundsätzlich über die Bytehäufigkeiten identifizieren, vergleichbar können auch bestimmte Muster in einer Gensequenz über die Häufigkeiten von Basentupeln - wie oben bereits angesprochen - gefunden werden (vgl. [4], S. 212ff). Um bioinformatische Methoden zur Datensuche in der IT-Forensik einsetzen zu können, müssen die Daten in Form von Bytes (Bits) zuerst in die entsprechende Basensequenz, bestehend aus dem Alphabet {A, C, G, T}, transkribiert werden. Hierzu werden im Rahmen der Arbeit spezifische Programme in *Biopython* und *Matlab* erstellt. Der zeitliche Aufwand bei der Erzeugung der Basensequenzen beträgt von weniger als einer Sekunde für eine Dateigröße kleiner als ein Megabyte in *Biopython* bis hin zu mehreren Minuten für dieselben Dateien in *Matlab*. Vor diesem Hintergrund erfolgt die Übersetzung der Bytes in eine Basensequenz ausschließlich in *Python* (Anlage E). Der Speicheraufwand für die resultierende Datei mit der Basensequenz erhöht sich um den Faktor 4 gegenüber der ursprünglichen Dateigröße. Jedes Byte - bestehend aus 8 Bits - wird in vier Blöcke zu je zwei Bits unterteilt. Das 2-Bitmuster entspricht einer Base (Bild 12), so dass ein Byte der ursprünglichen Datei in vier Basen übersetzt wird. Eine einzelne Base ist ein ASCII-Zeichen mit einem Speicherbedarf von einem Byte, daher die Größenzunahme um den Faktor vier. Nach diesem ersten Schritt der Übersetzung in eine Basensequenz werden anschließend die Häufigkeiten bestimmt. Wie in Bild 13 gezeigt, erscheint jedoch die Berechnung von Codon-Häufigkeiten - wie in der Bioinformatik üblich - einen Informationsverlust zu bedingen, der durch eine entsprechende Anpassung vermieden

bzw. reduziert werden kann. So wird die zu untersuchende Datenmenge (Image und Template) innerhalb des Untersuchungsfensters Base für Base abgefahren - die Schrittweite (Shift) beträgt hier eins und nicht wie bei der Untersuchung von Codonen üblich drei - und die Häufigkeiten für die definierte Tupellänge bestimmt. Die Längenauswahl für die Tupel beschränkt sich auf maximal vier Basen und entspricht damit einem ursprünglichen Byte. Längere Tupel können auf Grund der Tabellengrößen zu einer Instabilität von *Matlab* führen und bedingen vor allem einen so hohen zeitlichen Aufwand, dass diese Berechnungen nicht Gegenstand der Arbeit sind. Das Untersuchungsfenster besitzt grundsätzlich die Länge des jeweils für die Untersuchung genutzten Templates in Basen und fährt das Image ab, alle Fensterpositionen zusammen ergeben die komplette Imagelänge. Die berechneten Basenhäufigkeiten im Image und Template stimmen idealerweise überein (keine Abweichung), wenn das Untersuchungsfenster die Lokalisation im Image des jeweils verwendeten Templates exakt überdeckt. Aufschluss über den Grad der Abweichung bildet die Summe der Differenzbeträge. Da die verschiedenen Templates unterschiedliche Längen - in der Folge auch die Untersuchungsfenster für die Basenhäufigkeiten - besitzen, wird die Summe der Differenzbeträge auf die Länge des Untersuchungsfensters normiert. So ist ein Vergleich der berechneten Häufigkeitswerte möglich. Je kleiner die normierte Differenzsumme ist, desto größer ist die Übereinstimmung der Imagedaten an der entsprechenden Fensterlokalisierung mit dem Template. Vor diesem Hintergrund wird für jedes Template die Imagelokalisation mit dem Minimum der normierten Differenzsummen (*normDiffSum*) als Ausgangspunkt für die weiteren Untersuchungen der globalen und lokalen Alignmentsscores genutzt, die das Suchergebnis bestätigen oder zur Ablehnung führen sollen. Die orientierende Suche bedingt, dass für jedes Template (vorhanden oder im Image nicht vorhanden) eine Imagelokalisation mit einem Minimum gefunden wird. Eine Analyse der Images ist auf Grund der grundsätzlich unbegrenzten Datenmengen bei gleichzeitig beschränkten Ressourcen der Analysehardware derzeit nur block- oder abschnittsweise (Größe des Untersuchungsfensters) möglich. Natürlich ist ein direkter Vergleich zwischen Image und Template Bit für Bit durchführbar, wobei bereits minimale Abweichungen, z. B. Änderungen des Dateinamens bei ansonsten unveränderten Dateiinhalten, dann das Auffinden unmöglich machen können. Die Verwendung von Häufigkeiten scheint robuster gegenüber solchen Veränderungen zu

sein. Die Reduzierung einer beliebig großen Datei auf ein determiniertes Häufigkeitsprofil, z. B. bestehend aus 64 3-Tupeln, muss bei Änderungen nicht zu veränderten Häufigkeitswerten führen (vgl. Tabelle 3). Eine Abgrenzung zwischen den Templates, die im Image vorhanden bzw. nicht vorhanden sind, auf Basis der normierten Differenzsummen lässt sich unter Berücksichtigung von Schwellenwerten verbessern. Zur Ermittlung dieser Schwellenwerte werden die Mittelwerte und Standardabweichungen der normierten Differenzsummen in Abhängigkeit von der Tupellänge herangezogen (Formeln 7.1 bis 7.7). Eine Festlegung dateiformatspezifischer Schwellenwerte findet nicht statt, hier dürften sie durch das häufigste Dateiformat *jpg* (Tabelle 4) bestimmt sein. Bei den Untersuchungen stellten sich die Mittelwerte als die besten Schwellenwerte heraus, um möglichst minimale *FPR* und *FNR* zu erreichen. Idealerweise können die hier errechneten Mittelwerte für die normierten Differenzsummen auch für andere Untersuchungen bzw. unbekannte Images genutzt werden. Hierbei gilt es zu beachten, dass unterschiedliche Fenstergrößen die relevanten Imagedaten verschieden stark überdecken und allgemeingültige Schwellenwerte mit der hier genutzten Vorgehensweise nur schwierig zu erhalten sind. Hierzu bedarf es dateitypspezifischer Untersuchungen mit definierten Größen des Untersuchungsfensters. In diesem Zusammenhang ist eine Beeinflussung der Untersuchungsergebnisse auch durch Extremwerte oder Ausreißer, vor allem auf Grund unterschiedlicher Dateiformate, nicht auszuschließen. Ein Ausschluss von Extremwerten oder Ausreißern bzw. deren gesonderte Betrachtung findet aber nicht statt, um die Datenbasis unverändert in die Analysen einfließen zu lassen.

Bei allen Images nehmen die Werte der normierten Differenzsummen mit größer werdender Tupellänge zu. Tupellänge und die bei der Untersuchung zu betrachtende Tupelanzahl hängen wie folgt zusammen:

$$\text{Tupelanzahl} = 4^{\text{Tupellänge}}$$

Für eine Tupellänge von zwei ergibt sich beispielsweise:

$$\text{Tupelanzahl} = 4^2 = 16 \ (\{AA, AC, AG, AT, \dots, TA, TC, TG, TT\})$$

Gemäß Formel 6.1 steigt mit zunehmender Tupelanzahl auch die Anzahl der Summanden, eine durchschnittlich größere Summe kann die Folge sein (Bild 45). In der ersten Sequenz ergibt sich für 1-Tupel:  $A_1 = 5$ ,  $C_1 = 5$ ,  $G_1 = 5$ ,  $T_1 = 5$ ; in der zweiten

Sequenz entsprechend:  $A_2 = 6, C_2 = 4, G_2 = 5, T_2 = 5$ . Als Differenzsumme *DiffSum* ergibt sich:  $DiffSum_1 = |A_1 - A_2| + |C_1 - C_2| + |G_1 - G_2| + |T_1 - T_2| = 1 + 1 + 0 + 0 = 2$ . Für 2-Tupel folgt:  $AA_1 = 4, AC_1 = 1, AG_1 = 0, AT_1 = 0, CA_1 = 0, CC_1 = 4, CG_1 = 1, CT_1 = 0, GA_1 = 0, GC_1 = 0, GG_1 = 4, GT_1 = 1, TA_1 = 0, TC_1 = 0, TG_1 = 0, TT_1 = 4$  und  $AA_2 = 5, AC_2 = 1, AG_2 = 0, AT_2 = 0, CA_2 = 0, CC_2 = 2, CG_2 = 1, CT_2 = 1, GA_2 = 0, GC_2 = 1, GG_2 = 4, GT_2 = 0, TA_2 = 0, TC_2 = 0, TG_2 = 0, TT_2 = 4$ . Damit ist  $DiffSum_2 = |AA_1 - AA_2| + \dots + |TT_1 - TT_2| = 1 + 0 + 0 + 0 + 0 + 2 + 0 + 1 + 0 + 1 + 0 + 1 + 0 + 0 + 0 + 0 = 6$ .

Sequenz 1 (z. B. Image):

**A A A A A C C C C C G G G G G T T T T T**

Sequenz 2 (z. B. Template):

**A A A A A A C C C G G G G G C T T T T T**

Differenzsumme für 1-Tupel: 2; normiert:  $2 / 20 = 0,10$

Differenzsumme für 2-Tupel: 6; normiert:  $6 / 20 = 0,30$

Bild 45: Vergleich der Differenzsummen für zwei Sequenzen in Abhängigkeit von der Tupellänge

Gleichzeitig ist zu beachten, dass die Differenzsummen von der Größe des Untersuchungsfensters abhängen. Ein direkter Vergleich der Tupelhäufigkeiten zwischen Template und Image erfolgt idealerweise mit einer Fenstergröße, die der Templatelänge entspricht. Das zu untersuchende Image wird für jedes Template mit der entsprechenden Fenstergröße abgefahren. Die hierbei berechneten Differenzsummen sollten für einen statistischen Vergleich nicht genutzt werden, da eine belastbare Interpretation der Ergebnisse kaum möglich erscheint. Im günstigen Fall beträgt die Differenzsumme null und die normierte Differenzsumme errechnet sich unabhängig von der Fenstergröße ebenfalls zu null. Im schlechten Fall weichen die Sequenzen im Template und an der Imageposition des Untersuchungsfenster in jeder Base voneinander ab, so dass die Differenzsumme maximal das Zweifache der Fenstergröße erreicht:

Seq1: ‚AAAAAAAAAA‘

Seq2: ‚TTTTTTTTTT‘

$$DiffSum = |10 - 0| + |0 - 0| + |0 - 0| + |0 - 10| = 10 + 0 + 0 + 10 = 20$$

Fenstergröße = 10 Basen

Bei einem doppelt so großen Untersuchungsfenster wäre die *DiffSum* in diesem Fall 40. Vergleicht man beide Werte, ist nicht ersichtlich, ob eine unterschiedliche Fenstergröße zur Analyse vorlag oder der kleinere Wert durch eine anteilige Übereinstimmung der Sequenzen zustande kam. Werden die Werte auf die Fenstergröße normiert, folgt für die Fenstergröße 10 Basen:  $20 / 10 = 2$  und die Fenstergröße 20 Basen:  $40 / 20 = 2$ . In beiden Fällen ist nun eine maximale Abweichung der verglichenen Sequenzen erkennbar. Normiert man also die Differenzsummen auf die jeweilige Länge des Untersuchungsfensters (Templatelänge) fällt auf, dass die durchschnittlichen Werte mit der Tupellänge zunehmen, wobei sie für im Image vorhandene Templates am kleinsten sind (Tabelle 8). Dies deckt sich mit der Erwartung hinsichtlich geringerer Abweichungen der Häufigkeitsprofile zwischen Image und dort vorhandenen Templates im Vergleich zu verwendeten Templates, die nicht im Image vorhanden sind. Die Definition des Schwellenwertes *ut* nach der Formel 7.7 mit der Bedingung  $normDiffSum \leq ut$  soll möglichst alle vorhandenen Templates auffinden und somit die Wahrscheinlichkeit für falsch-negative Ergebnisse minimieren - dies wird insbesondere für die Tupellänge vier erreicht. Der relative Anteil der korrekt identifizierten Templates ( $a'$ ) zu den Testpositiven ( $a$ ) steigt an. Am stärksten zeigt sich dies beim Image F mit einer Rate  $a'/a$  von ca. 59 % gemäß Tabelle 9 auf ca. 80 % gemäß Tabelle 13. Zwischen den normierten Differenzsummen für Templates im Image und dort nicht vorhandenen existieren ab einer Tupellänge von zwei für alle untersuchten Images statistisch signifikante Unterschiede. Damit können grundsätzlich vorhandene von nicht im Image vorhandenen Templates abgegrenzt werden. Für die untersuchten Images zeigt sich insgesamt eine vergleichsweise sehr hohe *FPR*. Damit scheint das korrekte Auffinden von Dateien mit der hier vorgestellten orientierenden Suche auf Basis von Basenhäufigkeiten alleine nicht ausreichend zu sein. Die Wahrscheinlichkeit für falsch-negative Ergebnisse nimmt in Abhängigkeit von der Tupellänge ab. Entscheidend ist also die Wahl der Tupellänge und auch die Anwendung eines Schwellenwertes beeinflusst die Werte teilweise erheblich, Reduzierungen der *FNR* von über 50 % sind möglich, dies zeigt sich z. B. für die Werte bei der Tupellänge vier für das Image F in Tabelle 12 und Tabelle 16. Bei Tupellängen größer als vier kann eine weitere Reduzierung falsch-negativer Ergebnisse vermutet werden. Je mehr Basen ein Tupel umfasst, desto häufiger gehen die einzelnen Basen in die Berechnung der Basenhäufigkeiten ein (Schrittweite von eins bei einer Tupellänge

$\geq 2$ , vgl. Bild 13) und die Annäherung der Häufigkeitsprofile von Image und Template wird wahrscheinlicher. Dies zeigt sich u. a. in der *FNR*, die mit zunehmender Tupellänge abnimmt, gleichzeitig nehmen die *FPR* zu, um spätestens ab der Tupellänge drei eine leicht fallende Tendenz aufzuweisen (Bild 23). Mit Hilfe des genannten Schwellenwertes für die normierten Differenzsummen ist eine Reduzierung der *FPR* und *FNR* möglich. In Bild 23 fallen die für jede Tupellänge geringeren *FPR* des Images L im Vergleich zu den Images V und F auf. Eine Erklärung hierfür könnte in der Dateizusammenstellung des Images L liegen, es beinhaltet verschiedene Dateiformate; die Images V und F umfassen im Wesentlichen Bilder vom Typ *jpg* (Tabelle 4). Es lässt sich vermuten, dass die Unterscheidung unterschiedlicher Dateiformate mit Hilfe der hier vorgestellten orientierenden Suche besser gelingt als die Dateidifferenzierung innerhalb des Formates *jpg*. Der Schwellenwert *ut* für die normierten Differenzsummen des Images L wird auch durch das Dateiformat *pdf* beeinflusst (Tabelle 4), nach den Studienergebnissen von Veenman [43] kann es u. a. auf Basis von Bytehäufigkeiten zu falschen Zuordnungen von *jpg*-Dateien zum Dateityp *pdf* kommen. Damit könnten die  $FPR_L$  unter Anwendung des Schwellenwertes in Bild 24 erklärt werden. Die Anwendung von Schwellenwerten geht in der Regel mit einer Reduzierung der absoluten Zahlen einher, so dass sich möglicherweise auch die Aussagekraft der Untersuchungsergebnisse verringert. Ebenso ist das Analyseergebnis von der Fenstergröße abhängig. Auf Grundlage der durchschnittlichen Dateigröße und den Ergebnissen für die *FPR* und *FNR* unter Berücksichtigung des Schwellenwertes für die *normDiffSum* lässt sich vermuten, dass eine exakte Übereinstimmung zwischen der Größe des Untersuchungsfensters und der Templategröße nicht erforderlich ist (Abschnitt 7.1.2, Tabelle 20). Das größere Untersuchungsfenster, wie hier mit 128.000 Basen, liefert gute Untersuchungsergebnisse. Betrachtet man die mit der Suche ermittelten Imagelokalisationen, die eine Überdeckung  $> 0\%$  mit der tatsächlichen Lokalisation (korrekte Identifizierung, geschweifte Klammern in den Tabellen) aufweisen, verschlechtern sich diese Ergebnisse in der Regel. Die Suchoptimierung mit einer Verschiebung des Untersuchungsfensters gemäß Formel 6.2 und Bild 16 soll eine Überdeckung der berechneten mit der tatsächlichen Imagelokalisation von über 50 % garantieren. Nicht bei allen positiven Testergebnissen wird dies erreicht. In Bild 46 ist das Ergebnis der Untersuchung des Images L mit Basen-4-Tupel exemplarisch dargestellt. Zu sehen sind gemäß Spalte drei nur Templates, die im



Image vorhanden sind („1“). Auf der linken Seite beruhen die Ergebnisse auf der Verschiebung (Shift) des Untersuchungsfensters gemäß Formel 6.2, hingegen erfolgte der Shift bei den Ergebnissen in der Tabelle auf der rechten Seite nicht. Es ist im Vergleich der Spalten vier zu erkennen, dass mit dem Shift mehr Treffer (\*\*\*) vorhanden sind und auch der vor der Klammer angegebene Prozentsatz der Überdeckung - der mit der orientierenden Suche berechneten Imagelokalisation des Templates mit der tatsächlichen - häufig höher ausfällt.

image\_templates\_compare

PLOTS

VARIABLE

VIEW

No Variable Selected

Select elements to plot

SELECTION

PLOTS

124x20 cell

Image L, Basen-4-Tupel, ohne Shift

	1	2	3	4
100 'BSI_Gesichtserkennung.pdf'	252512	'1'	'93 (***30 ***)'	
101 'BSI_Iriserkennung.pdf'	342075	'1'	'84 (***24 ***)'	
102 'BioAPI2_gi-proc-108-010.pdf'	276458	'1'	'96 (***26 ***)'	
103 'ISO19794_Overview.pdf'	151514	'1'	'89 (35)'	
104 'RossIntroMultibio_EUSIPCO07.pdf'	255233	'1'	'95 (***29 ***)'	
105 'asia-3023824_340.jpg'	65536	'1'	'73 (***42 ***)'	
106 'beautiful-girl-2003647_340.jpg'	61339	'1'	'88 (***44 ***)'	
107 'community-150124_1280.png'	109469	'1'	'73 (***38 ***)'	
108 'composing-2925179_340.jpg'	56487	'1'	'90 (***45 ***)'	
109 'elephants-196613_1920.jpg'	626021	'1'	'93 (***13 ***)'	
110 'fitness-863081_340.jpg'	50504	'1'	'78 (***39 ***)'	
111 'girl-56683_1280.jpg'	243094	'1'	'78 (***32 ***)'	
112 'girl-657753_1280.jpg'	261302	'1'	'11 (17)'	
113 'indian.jpg'	81636	'1'	'70 (***5 ***)'	
114 'indian_GaussWeichz.png'	239400	'1'	'0 (-1)'	
115 'indian_GaussWeichz.xcf'	753074	'1'	'67 (12)'	
116 'man-1508670_340.jpg'	65105	'1'	'69 (***43 ***)'	
117 'man-1508680_340.jpg'	72989	'1'	'82 (43)'	
118 'maple-moth-183186_1280.jpg'	453524	'1'	'68 (***18 ***)'	
119 'puppy-384647_1280.jpg'	226477	'1'	'65 (16)'	
120 'sheep-1642874_1280.jpg'	167921	'1'	'97 (***36 ***)'	
121 'sunset-1759716_1280.jpg'	275356	'1'	'63 (***27 ***)'	
122 'water-1122166_1280.jpg'	344276	'1'	'46 (16)'	

124x20 cell

Image L, Basen-4-Tupel, mit Shift gemäß Formel 6.2

	1	2	3	4
100 'BSI_Gesichtserkennung.pdf'	252512	'1'	'93 (***30 ***)'	
101 'BSI_Iriserkennung.pdf'	342075	'1'	'84 (***24 ***)'	
102 'BioAPI2_gi-proc-108-010.pdf'	276458	'1'	'96 (***26 ***)'	
103 'ISO19794_Overview.pdf'	151514	'1'	'89 (35)'	
104 'RossIntroMultibio_EUSIPCO07.pdf'	255233	'1'	'95 (***29 ***)'	
105 'asia-3023824_340.jpg'	65536	'1'	'90 (***42 ***)'	
106 'beautiful-girl-2003647_340.jpg'	61339	'1'	'88 (***44 ***)'	
107 'community-150124_1280.png'	109469	'1'	'90 (***38 ***)'	
108 'composing-2925179_340.jpg'	56487	'1'	'90 (***45 ***)'	
109 'elephants-196613_1920.jpg'	626021	'1'	'93 (***13 ***)'	
110 'fitness-863081_340.jpg'	50504	'1'	'78 (***39 ***)'	
111 'girl-56683_1280.jpg'	243094	'1'	'71 (***32 ***)'	
112 'girl-657753_1280.jpg'	261302	'1'	'79 (***28 ***)'	
113 'indian.jpg'	81636	'1'	'70 (***5 ***)'	
114 'indian_GaussWeichz.png'	239400	'1'	'0 (-1)'	
115 'indian_GaussWeichz.xcf'	753074	'1'	'82 (***11 ***)'	
116 'man-1508670_340.jpg'	65105	'1'	'80 (***43 ***)'	
117 'man-1508680_340.jpg'	72989	'1'	'84 (***40 ***)'	
118 'maple-moth-183186_1280.jpg'	453524	'1'	'68 (***18 ***)'	
119 'puppy-384647_1280.jpg'	226477	'1'	'83 (***34 ***)'	
120 'sheep-1642874_1280.jpg'	167921	'1'	'97 (***36 ***)'	
121 'sunset-1759716_1280.jpg'	275356	'1'	'86 (***27 ***)'	
122 'water-1122166_1280.jpg'	344276	'1'	'81 (***23 ***)'	

Bild 46: Analyseergebnisse des Images L mit Basen-4-Tupel (links ohne und rechts mit Shift)

Die Anwendung eines Shiftes zur Verbesserung des Untersuchungsergebnisses für jedes Template scheint also durchaus erfolgversprechend. Neben der Berechnung der Shiftweite, wie beispielsweise nach Formel 6.2, ist vor allem die Abbruchbedingung von Bedeutung. Die hier angewandte Bedingung gemäß Formel 6.3 kann zu dem in Bild 47 gezeigten suboptimalen Ergebnis führen. Die Position 2 erfasst die Templatelokalisation im Image deutlich besser als die erste Position. Die oben genannte Abbruchbedingung führt dazu, dass Position 1 als wahrscheinlichste Imagelokalisation ausgegeben wird.

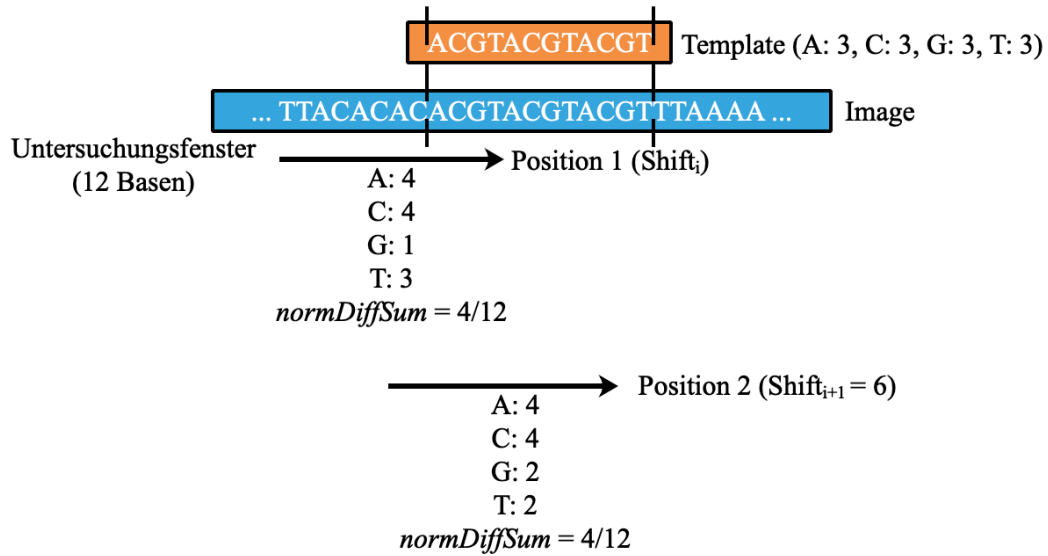


Bild 47: Anwendung des Shifts gemäß Formel 6.2

In der Folge erscheint die Anpassung der Bedingung in der Form

$$\min(normDiffSum_i) > \min [\min(normDiffSum_1), \dots, \min (normDiffSum_{i-1})] \quad (9.1)$$

ein besseres Ergebnis zu liefern, jedoch kann dies zu dem in Bild 48 gezeigten Verlauf führen. Die optimale Überdeckung des Untersuchungsfensters mit der tatsächlichen Imagelokalisation des Templates dürfte bei Position 2 oder 3 liegen. Die angepasste Abbruchbedingung bedingt jedoch weitere Verschiebungen des Untersuchungsfensters, so dass die tatsächliche Imagelokalisation möglicherweise nicht „entdeckt“ wird. Eine Korrektur der Fensterposition zum Aufdecken der wahrscheinlichsten Imagelokalisation hängt wesentlich von den Basenhäufigkeiten, der Berechnung des Shiftes und der Abbruchbedingung ab. Die in Bild 46 bis Bild 48 gezeigten Beispiele zeigen auf, dass eine vermeintliche Optimierung der Shiftberechnung in einigen Fällen auch zu einer Verschlechterung führen kann.

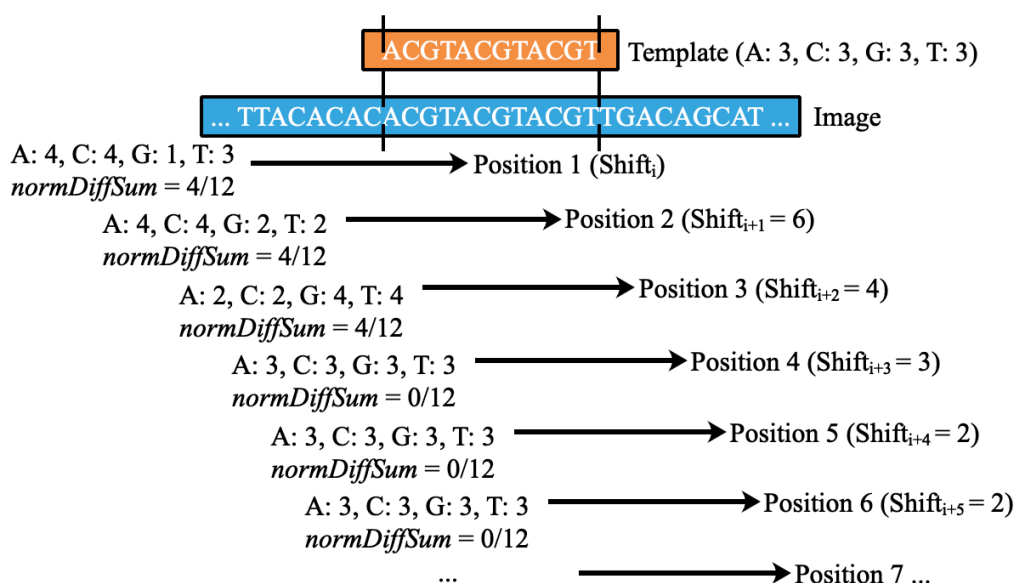


Bild 48: Beispiel für eine angepasste Abbruchbedingung (Shift)

Die Speicherung von Dateien erfolgt grundsätzlich block- bzw. clusterweise und bei nicht genutzten Clusteranteilen handelt es sich um allozierte Speicherbereiche, die keine Daten oder Daten aus dem Hauptspeicher enthalten können (Slack Space, siehe auch Bild 4). Damit kann von einer Verbesserung der orientierenden Suche bei intakten Filesystemen ausgegangen werden, indem das zu untersuchende Image nicht - wie im Rahmen dieser Arbeit - mit einer Schrittweite entsprechend der Fenstergröße (standardmäßig die Templatelänge) abgefahren wird, sondern clusterweise, bei den hier verwendeten Images 512 Bytes oder 2.048 Basen. Unterstützt wird diese Vermutung durch die Werte in Tabelle 6. Bereits das Überspringen der für das Filesystem reservierten ersten 32 Cluster (Shift 65.536) führt zu einer Annäherung an die optimalen Werte für die normierten Differenzsummen, die normierten globalen und lokalen Alignmentscores. Die Größe des Untersuchungsfensters als ein Vielfaches der Clustergröße unter Beachtung der Dateilänge zu wählen, wird vermutlich bessere Resultate ergeben. Nutzt die Speicherung einer Datei den letzten zugehörigen Cluster nur minimal, geht in die Häufigkeitsberechnung der Inhalt des Slack Space (dieser wird z. B. zum „Verstecken“ von Daten genutzt, vgl. [32]) mit ein. In Abhängigkeit von diesem Inhalt weicht das Ergebnis mehr oder weniger stark vom optimalen Wert ab. Allerdings sind ein intaktes Filesystem bzw. eine intakte Dateistruktur und ein ungenutzter Slack Space unter der Annahme krimineller Manipulationen nicht vorauszusetzen. Die hier vorgestellte „naive“ Vorgehensweise der orientierenden Suche verfügt im Gegensatz zu einem clusterweisen

Vergleich zwischen Template und Image über eine geringere Zeitkomplexität, da übliche Dateigrößen mehrere Cluster beanspruchen; das heißt, das Image wird bei der orientierenden Suche in größeren Abschnitten abgefahren.

Zusammenfassend kann für die orientierende Suche festgehalten werden, dass die Suche nach vorgegebenen Templates auf Basis von Basenhäufigkeiten möglich ist, insbesondere zum Ausschluss von Dateien. Die besten Untersuchungsergebnisse zeigen sich überwiegend für die hier untersuchte maximale Tupellänge vier, vermutlich im Wesentlichen durch den häufigsten Dateityp *jpg* und die spezifische Implementierung der orientierenden Suche in Verbindung mit der Suchoptimierung nach Formel 6.2 beeinflusst. Trotz der geschilderten Herausforderungen bei der Suche nach der optimalen Imagelokalisation und Reduzierung einer Basensequenz auf ein Häufigkeitsprofil sind die Raten der falsch-negativen Resultate (*FNR*) mit  $\leq 6\%$  ab einer Tupellänge vier ohne Anwendung von Schwellenwerten und  $\leq 2\%$  unter Berücksichtigung des Schwellenwertes *ut* für die *normDiffSum* bemerkenswert - das etablierte Verfahren *sdhash* liegt nach der Studie von Roussev bei ca. 5 % (vgl. [35]). Interessanterweise zeigen sich in dieser Studie u. a. für Bilddaten deutlich häufiger falsch-positive Ergebnisse als für Textdaten, insbesondere für *sdhash* Scores  $\leq 23$ . Gemäß Abschnitt 4.2 nimmt die Wahrscheinlichkeit für einen korrekten Treffer bei Werten unterhalb von 21 stark ab, die kleinsten Werte u. a. für die Bilddaten liegen in der genannten Studie aber bei 12 und für die Textdaten bei fünf. Die Angabe einer entsprechenden *FPR* ist leider nicht möglich. Die hier vorgestellte orientierende Suche liefert in jedem Falle eine Imagelokalisation, da immer eine kleinste normierte Differenzsumme existiert. Erst die Einführung von Schwellenwerten ermöglicht auch bei unbekannten Images eine Differenzierung zwischen vorhandenen und nicht vorhandenen Templates (Mustern). Der Häufigkeitsvergleich zwischen Template und Image benötigt ca. 11 Minuten je Template im Image F. Beschleunigen kann man den Algorithmus u. a. durch das Abspeichern des entsprechenden Basenprofils eines Templates in der Datenbank, ohne dass zuerst das Template eingelesen und die Basenhäufigkeiten ermittelt werden müssen. Die ursprüngliche Idee einer Verwendung von Basenhäufigkeiten zur orientierenden Suche im Image, die dann in einem weiteren Schritt verfeinert wird, ist nur eingeschränkt geglückt. Eine falsch ermittelte Imagelokalisation kann durch weitere Analyseschritte

lediglich als falsch bestätigt werden, eine Korrektur der Imagelokalisation ist mit dem vorgestellten Verfahren nicht möglich.

## 9.2 Alignments - unveränderte Daten

Ein weiterer Untersuchungsschritt zur Bestätigung oder Ablehnung der ermittelten Imagelokalisationen ist erforderlich, da die Untersuchung eines Images nicht clusterweise erfolgt und damit die Überdeckung des Untersuchungsfensters mit der tatsächlichen Lokalisation - wie oben beschrieben - in den wenigsten Fällen optimal sein dürfte. Vor allem bei den hohen bis sehr hohen *FPR* (Abschnitt 7.2) ist eine Reduzierung der *FNR* in Richtung 0 % wünschenswert. Hierzu soll der Einsatz von globalen und lokalen Alignments dienen. Auf Grund der in Abschnitt 6.2.4 skizzierten Platzkomplexität der eingesetzten Algorithmen nach Needleman und Wunsch (globale Alignments) sowie Smith und Waterman (lokale Alignments) ist die Berechnung der Scores durch die verwendete Hardware limitiert. Ein ressourcensparendes, abschnittsweises Alignieren von Image und Template - wie in der vorliegenden Arbeit - könnte jedoch in Abhängigkeit von der Abschnittsgröße mehr oder weniger stark vom optimalen Score (Berechnung über die gesamte Templatelänge) abweichen. Der Vergleich unterschiedlicher Größen von Untersuchungsfenstern ergibt signifikante Abweichungen der fensterspezifischen globalen und lokalen Scores ab einer Tupellänge von zwei (siehe Tabellen in Anlage C). Die Alignmentuntersuchungen werden mit der hardwareorientierten maximalen Fenstergröße von 64.000 Basen durchgeführt. Wie bei den Differenzsummen wird die Summe der Alignmentsscores auf die Templatelänge normiert. Die mit Hilfe der orientierenden Suche berechnete Lokalisation im Image wird mit dem Untersuchungsfenster für das Alignment abgefahren (die Schrittweite ist hier die Fenstergröße von 64.000 Basen) und für jede Fensterposition der globale bzw. lokale Alignmentsscore zwischen Image und Template errechnet. Alle Fensterpositionen zusammen ergeben die Länge des jeweils untersuchten Templates bzw. Imageabschnittes. Die Summe der einzelnen Scores entspricht dem Gesamtscore, der dann auf die Templatelänge normiert wird. So ist der direkte Wertevergleich auch bei unterschiedlich großen Templatedateien möglich. Für das Image L mit der kleinsten Dateianzahl zeigen sich annähernd gleiche normierte **globale Scores** für alle Tupellängen, eine Unterscheidung zwischen im Image vorhandenen und nicht vorhandenen Templates ist

statistisch signifikant nicht möglich. Erst für die Images V und F mit einer größeren Anzahl an Dateien ergeben sich gemäß Tabelle 21 signifikante Testergebnisse für alle Tupellängen. Es kann festgestellt werden, dass die normierten globalen Scores für im Image vorhandene Templates im Vergleich zu den nicht vorhandenen durchschnittlich größer sind und mit der Tupellänge zunehmen. Dies ist auch zu erwarten, da eine höhere Übereinstimmung der Basensequenzen auch mit höheren Alignmentsscores einhergeht. In Abhängigkeit von der Tupellänge wird die Lokalisationsermittlung der Templates im Image genauer und bei vorhandenen Templates wird deren Basensequenz eine höhere Übereinstimmung mit der Imagesequenz an dieser Position aufweisen als nicht vorhandene. Es fällt auf, dass hingegen die normierten globalen Scores der nicht vorhandenen Templates in allen Images und für alle Tupellängen mit einem durchschnittlichen Mittelwert in Höhe von 0,1047 (durchschnittliche Standardabweichung: 0,0189; Auswertung der entsprechenden Werte in Tabelle 21) als eher „stabil“ zu bezeichnen sind. Im Gegensatz zu den normierten Differenzsummen ergeben sich für die normierten globalen Alignmentsscores insgesamt die kleineren Werte für die im Image nicht vorhandenen Templates. Somit erfolgt die Schwellenwertdefinition analog zu den *normDiffSum* mit der Bedingung  $Alignmentscore \geq lt$  (Formel 7.9). Die *FPR* und *FNR* nach Tabelle 23 bis Tabelle 25 ergeben im Vergleich zu den Werten mit der  $normDiffSum \leq ut$  (Tabelle 14 bis Tabelle 16) nur teilweise Verbesserungen. Diese häufen und verstärken sich unter Einbeziehung des Schwellenwertes *ut* (Tabelle 27 bis Tabelle 29), so dass für Basenquadrupel eine  $FNR \leq 1\%$  erreicht wird. Ein globales Alignment startet immer bei der ersten Basenposition im Image bzw. Template und kann damit bei nicht vollständiger Übereinstimmung zwischen ermittelter und tatsächlicher Lokalisation im Image auch keine maximalen globalen Scores liefern. Es ist nachvollziehbar, dass das Alignment zweier gleicher Basensequenzen, die gegeneinander verschoben aligniert werden, geringere globale Scores liefert (Bild 49).

korrekte Position	um 4 Basen verschoben	Image
... A C C G A T T C G C C C C ...	... A T T C G C C C C ...	
A C C G A T T C G	A C C G A T T C G	
⇒ Score: 9	⇒ Score: -2	Template

Bild 49: Globale Scores für unterschiedliche Startpositionen im Image

Eine Visualisierung der globalen Alignments der Sequenzen aus Bild 49 sind mit Hilfe der *Matlab*-Funktion *showalignment()* in Bild 50 dargestellt.

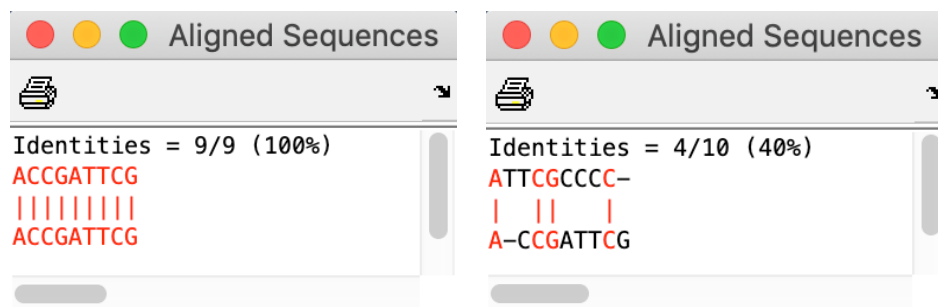


Bild 50: Visualisierung globaler Alignments mit *showalignment()* in *Matlab*

Die Berechnung maximaler **lokaler Alignmentsscores** ist hingegen auch bei abweichenden Lokalisationen grundsätzlich möglich (vgl. Bild 9), da im Gegensatz zur Betrachtung der vollständigen Basensequenz bei globalen Alignments hier Teilsequenzen identifiziert werden können. Ab der Tupellänge zwei ergeben sich in den Images L, V und F hinsichtlich der normierten lokalen Alignmentsscores signifikante Unterschiede zwischen den vorhandenen und nicht im Image vorhandenen Templates. Ebenso sind die normierten Scores für Templates, die im Image gefunden werden können, größer und mit der Tupellänge zunehmend im Vergleich zu den normierten Scores für nicht vorhandene Templates (Tabelle 30). Wie bei den normierten globalen Alignmentsscores verhalten sich auch die lokalen Scores der Templates, die nicht im Image zu finden sind, mit ihren Mittelwerten „stabil“ (durchschnittlicher Mittelwert in Höhe von 0,1095 mit einer durchschnittlichen Standardabweichung von 0,0192; Auswertung der entsprechenden Werte in Tabelle 30) in Verbindung mit deutlich kleineren Standardabweichungen in den Images V und F im Vergleich zu den Werten für vorhandene Templates. Die „Stabilität“ der Werte für die lokalen (und globalen) Alignmentsscores nicht vorhandener Templates gemäß Tabelle 30 (Tabelle 21) ist am ehesten so zu werten, dass die fehlende Übereinstimmung zwischen Imageabschnitt und Template - und dies sollte für alle untersuchten Tupellängen der Fall sein - hier zum Ausdruck kommt. Die bei den lokalen Scores höheren Werte für im Image vorhandene Templates sprechen einerseits für eine erfolgreiche Suche nach Teilsequenzen und beruhen andererseits sicherlich auch auf der Anwendung der Scoring Matrix und Bewertung von Lücken wie bei den globalen

Alignments. Dies soll die Vergleichbarkeit aller darauf beruhender Untersuchungsergebnisse gewährleisten, gleichzeitig ist damit das Aufdecken der längsten gemeinsamen Teilsequenz zwischen Image und Template nach ([38], S. 111f) nicht gegeben. Die Anwendung anderer Scoring-Werte und Lücken-Bewertungen könnte eine bessere Differenzierung zwischen vorhandenen und nicht im Image vorhandenen Templates ermöglichen (siehe hierzu auch die Ausführungen in Kapitel 5). Zwei Sequenzen mit einer geringen Übereinstimmung weisen voraussichtlich eine kleinere (längste) gemeinsame Teilsequenz auf als Sequenzen mit einer vergleichsweise höheren Übereinstimmung. In der Folge werden bei den durchgeführten Untersuchungen eher kleinere gemeinsame Teilsequenzen der Images und Templates identifiziert, die bei unterschiedlichen Übereinstimmungsgraden auftreten können. Damit steigt die Wahrscheinlichkeit für annähernd gleiche lokale Alignmentsscores. Dennoch zeigt sich zwischen den lokalen Scores der vorhandenen und nicht vorhandenen Templates ein signifikanter Unterschied ab der Tupellänge zwei in allen Images. Neben dem Alignmentsscore sind bei lokalen Alignments die Angaben zu den Startpositionen in den Sequenzen (hier Image und Template) ein wichtiges Ergebnis. Eine Verschiebung der Sequenzen gegeneinander auf Basis der Startpositionen kann mit einem besseren Alignmentsergebnis verbunden sein (Bild 51).

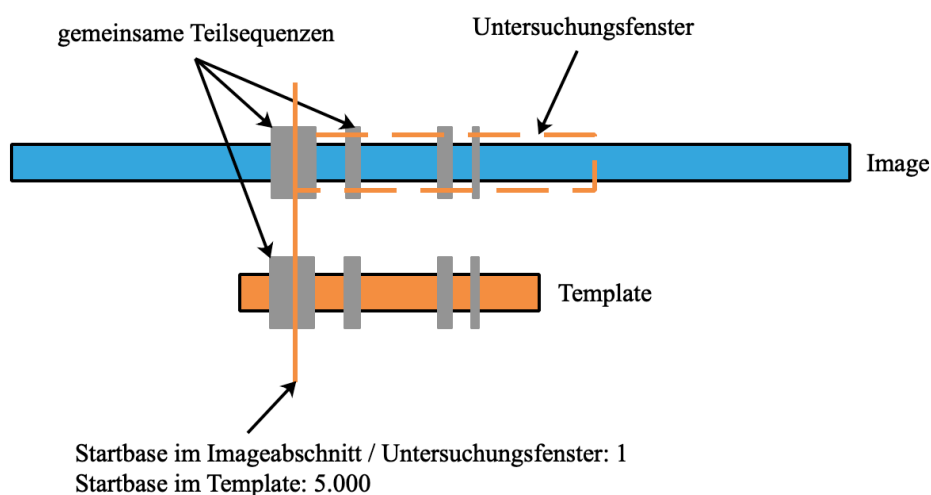


Bild 51: Teilsequenzen und Startangaben bei lokalen Alignments



Die Verschiebung des Untersuchungsfensters um 5.000 Basen nach links in Bild 51 führt zu einer vollständigen Überdeckung aller gemeinsamen Teilsequenzen mit einem maximalen lokalen Alignmentsscore für den Imageabschnitt.

Die Unterschiede der durchschnittlichen normierten lokalen Alignmentsscores zwischen den beiden Templatekategorien (vorhanden - nicht im Image vorhanden) decken sich mit der Erwartungshaltung, dass das Auffinden der Imagelokalisationen vorhandener Templates auf Grund der Sequenzenähnlichkeiten mit höheren Scores einhergeht. Idealerweise würden die Scores 1,00 betragen, dies ist hier nicht der Fall und spricht u. a. für eine suboptimale Identifizierung der Imagelokalisation. Bei den nicht vorhandenen Templates gibt es grundsätzlich keine optimale Imagelokalisation. Analog zu den globalen Alignments wird hier der Schwellenwert als Mittelwert der nicht im Image vorhandenen Templates definiert (Formel 7.10). Damit soll ein maximaler Ausschluss nicht vorhandener Templates erreicht werden. Unter Berücksichtigung dieses Schwellenwertes liegen ab einer Tupellänge von drei die *FNR* bei  $\leq 7\%$ . Die *FPR* liegen deutlich höher, gravierende Unterschiede zu den normierten globalen Scores zeigen sich nicht. Die zusätzliche Anwendung des Schwellenwertes *ut* (Tabelle 36 bis Tabelle 38) kann insbesondere die *FNR* auf 0 % reduzieren. Dass lokale Alignmentsscores im Gegensatz zu globalen Scores eine bessere Differenzierung zwischen vorhandenen und nicht im Image vorhandenen Dateien zulassen (Tabelle 30 und z. B. Vergleich von Bild 25 und Bild 26), basiert am ehesten auf dem Alignieren von Teilsequenzen. Globale Alignments versuchen beide Sequenzen vollständig in Übereinstimmung zu bringen. Da das Untersuchungsfenster hier die tatsächliche Lokalisation in der Regel nur teilweise umfasst - die Schrittweite ist bei den vorliegenden Untersuchungen die Fenstergröße und nicht eins - ist ein optimales bzw. maximales globales Alignment schwierig zu realisieren. Zusätzlich ist die Fenstergröße, über die aligniert wird, von Bedeutung. Der Gesamtscore ergibt sich als Summe aus den Alignmentsscores der einzelnen Untersuchungsabschnitte (Fensterpositionen), damit ist eine abschnittsübergreifende Berücksichtigung der „Nahtstellen“ zwischen den Fensterpositionen nicht möglich.

Führt man die orientierende Suche nicht auf Basis von Häufigkeiten durch, sondern verwendet direkt die Berechnung von Alignments (Bild 27), so sind niedrigere *FPR* und *FNR* zu erwarten. Die zunächst bemerkenswerten Ergebnisse mit einer *FPR* in Höhe von 0 % und einer *FNR* von 20 % sind beispielhaft für 10 Templates in Bild 28 gezeigt. Auf

Grund der sehr langen Untersuchungsdauer ist eine vollständige Imageanalyse nicht erfolgt, die genannten *FPR* und *FNR* sind daher wenig belastbar. Die mit den gleichen Parametern durchgeführte orientierende Suche mit Hilfe von Basenhäufigkeiten ergibt ab einer Tupellänge von drei grundsätzlich das gleiche Ergebnis, allerdings sind die Übereinstimmungen der ermittelten mit der tatsächlichen Imagelokalisation des Templates teilweise geringer und eine Differenzierung der vorhandenen von den nicht vorhandenen Templates - z. B. durch die Definition eines Schwellenwertes für die normierte Differenzsumme - erscheint schwieriger (Bild 29). Der zeitliche Aufwand ist bei der orientierenden Suche aber deutlich geringer (ca. 35 Stunden bei der „Alignment-Suche“ gegenüber weniger als eine Minute für den Vergleich der Basenhäufigkeiten).

### 9.3 Untersuchung fragmentierter Daten

Die Anwendung der beschriebenen Vorgehensweise der orientierenden Suche kann auch zum Auffinden von Dateifragmenten genutzt werden. Durch manuelles Überschreiben von Dateiinhalten, z. B. Dateiheder beim Image „L Fragments“, erzeugte Fragmente können korrekt identifiziert werden (Tabelle 39). Auch die bioinformatische Analyse automatisiert modifizierter Images deckt Fragmente auf. Da Dateifragmente kleiner als die ursprüngliche Dateilänge sind - die Größe ist a priori nicht bekannt - werden hier die bioinformatischen Untersuchungen zunächst mit zwei Fenstergrößen durchgeführt. Der erste Durchlauf erfolgt mit einer Größe des Untersuchungsfensters gleich der Templatelänge und der zweite Durchlauf findet mit der halben Templatelänge statt. Auf Grund des hohen Ressourcenaufwandes unterbleibt für die modifizierten Images eine weitere Reduzierung der Fenstergröße. Für das Image I mit den vier manuell eingefügten Dateifragmenten wird die Anpassung der Größe des Untersuchungsfensters in Abhängigkeit von der Fragmentgröße beispielhaft durchgeführt. Beim Image „L Fragments“ sind gemäß Tabelle 50 die ersten Dateiinhalte, einschließlich des Dateiheders, zu unterschiedlichen Anteilen mit Nullen überschrieben. Die eigentlichen Dateiinhalte sind größtenteils erhalten, erwartungsgemäß sind die in der Arbeit nicht explizit aufgeführten Mittelwerte und Standardabweichungen der normierten Differenzsummen sowie der globalen Alignmentsscores mit denen des Images L annähernd vergleichbar (vgl. Tabelle 8, Tabelle 21 und die Ergebnisdateien auf dem Datenträger). Die lokalen Alignmentsscores hingegen betragen teilweise nur ein Sechstel

bei der betrachteten Fenstergröße gleich der Templatelänge. Die mit Nullen überschriebenen Imageabschnitte bedingen am ehesten diese Score-Reduzierung, zumal die lokalen Scores für die im Image vorhandenen Templates mit der Halbierung des Untersuchungsfensters wieder ansteigen. So ist eine gute Differenzierung zwischen vorhandenen und nicht im Image vorhandenen Templates möglich. Die niedrigste *FPR* mit ca. 6 % ergibt sich für die Tupellänge vier und der Fenstergröße gleich der halben Templatelänge bei Betrachtung der globalen Alignmentsscores, die *FNR* beträgt dann 0 %. Unter den 15 so korrekt identifizierten Templates befinden sich auch sechs der insgesamt 10 relevanten Fragmente. Die Imageuntersuchung mit der halben Fenstergröße zeigt für die nicht vorhandenen Templates leicht erhöhte durchschnittliche Alignmentsscores auf, insgesamt auch hier ein „stabiles“ Verhalten. Diese für das Image „L Fragments“ beschriebenen Aspekte treffen gleichermaßen für die modifizierten Images zu. Die beste *FPR* mit ca. 8 % ergibt sich durch die globalen Scores im modifizierten Image L, die niedrigste *FNR* liegt in diesem Zusammenhang bei ca. 4 % im modifizierten Image V. Die meisten relevanten Fragmente im Image „L Fragments“ und den modifizierten Images werden tupellängenunabhängig mit der veränderten (halbierten) Fenstergröße korrekt identifiziert (Tabelle 39). Ein Grund dafür, dass nicht alle relevanten Fragmente aufgefunden werden, kann in der orientierenden Suche mit Hilfe von Basenhäufigkeiten liegen, bei der das Untersuchungsfenster - wie bereits ausgeführt - möglicherweise eine zu geringe Überdeckung mit den tatsächlichen Imagelokalisationen der Dateireste aufweist. Hinzu kommt, dass durch ein zu großes Untersuchungsfenster nicht zur ursprünglichen Datei gehörende Imageinhalte in die Berechnungen eingehen und das Untersuchungsergebnis „verfälschen“. Auch ein zu kleines Untersuchungsfenster (unterhalb der Fragmentgröße) kann die Fragmentsuche negativ beeinflussen. Beispielsweise ergibt die Halbierung der Fenstergröße in Bild 52 ein anderes Häufigkeitsprofil mit der Gefahr einer erhöhten Wahrscheinlichkeit für falsch-positive und falsch-negative Resultate. Die Anzahl der gefundenen Fragmente kann sich durch die Verkleinerung des Untersuchungsfensters und Angleichung an die Fragmentgröße erhöhen (Tabelle 39 und Tabelle 41). Es kann festgehalten werden, dass die Suche nach Fragmenten mit verschiedenen Fenstergrößen durchgeführt werden sollte. Erst hierdurch verbessert sich das Suchergebnis bei vorliegenden Fragmenten unterschiedlicher Länge.

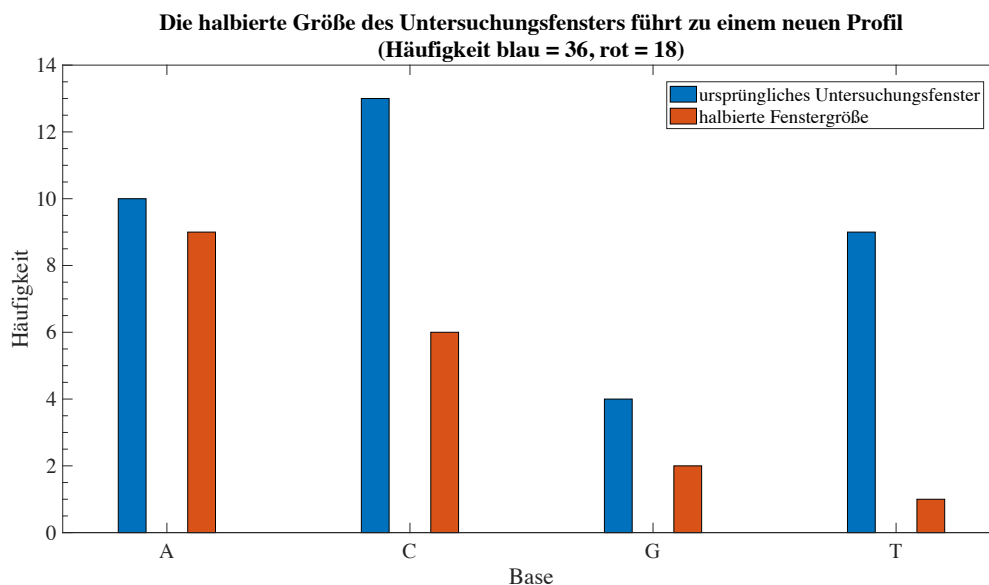


Bild 52: Profiländerung bei Halbierung der Fenstergröße

Da eine Datenfragmentierung bereits durch das Filesystem selbst und die tägliche Arbeit mit Dateien (Kopieren, Einfügen, Löschen, Ersetzen oder Verschieben) hervorgerufen sein kann, dürfte die Definition der Fenstergröße als ein Vielfaches der Blockgröße vorteilhaft sein. Im Falle krimineller Manipulationen wird man möglicherweise hiervon unabhängige, zufällig gewählte Fenstergrößen probieren müssen. Die Fenstergröße muss aber nicht der Fragmentgröße entsprechen. Mit Blick auf die *FPR* könnten Untersuchungsfenster größer als das Fragment vorteilhaft sein, kleinere Fenster scheinen mit günstigeren *FNR* einherzugehen (Abschnitt 7.1.2). Die Analyse der modifizierten Images mit einer verringerten Fenstergröße kann in der Gesamtschau fragmentierter und nicht fragmentierter Dateien die Wahrscheinlichkeiten für falsch-positive und falsch-negative Resultate verringern. Zur Bestätigung sollten die unveränderten Images mit verschiedenen Fenstergrößen untersucht werden.

#### 9.4 Untersuchung ähnlicher Daten

Die Suche nach ähnlichen Daten (Bildern) zeigt insgesamt keine guten Ergebnisse. Eine Definition des Begriffs „Ähnlichkeit“ unterbleibt bewusst, um zunächst auf naive Weise die hier vorgestellten bioinformatischen Methoden auf frei verfügbare Bilddateien anzuwenden. Belastbare Angaben von *FPR* und *FNR* sind so nicht möglich. Es wird unterstellt, dass sich die jeweils 25 Frauen- bzw. Männergesichter des Images „Similar Files“ ähnlich sind. Für einen menschlichen Betrachter ist dies durchaus nachvollziehbar.

Mit Hilfe der orientierenden Suche in Verbindung mit der Analyse von Ergebnissen innerhalb einer zulässigen Abweichung von der *normDiffSum* können jedoch ähnliche Bilder gefunden werden. Dies gelingt auch mit den durch Multiple Alignments erzeugten Konsensussequenzen. Diese Sequenzen beinhalten für jede Basenposition die über alle in das Multiple Alignment einfließenden Sequenzen ermittelte häufigste Base. Mehrere Basenprofile werden also mit Hilfe von Häufigkeiten auf ein Basenprofil reduziert. Idealerweise wird das Multiple Alignment in einem Schritt über die gesamten Längen der Basensequenzen durchgeführt. Die Erzeugung einer Konsensussequenz erfolgt hier auf Grund der begrenzten Hardwareressourcen aber blockweise, insbesondere die Platzkomplexität stellt eine Herausforderung dar. Möglicherweise verbessern sich die Ergebnisse durch den Verzicht auf das blockweise Alignieren, da dann auch die Sequenzanteile zu Beginn und am Ende eines Blockes in das Multiple Alignment einfließen. Auch gilt es zu beachten, dass unterschiedlich lange Sequenzen zu einem Alignment mit zueinander verschobenen ähnlichen Dateiabschnitten führen, z. B. das Alignieren einer Bilddatei mit einer verkleinerten Version des Bildes. Ein Ausgleich der Abweichungen durch das Einfügen von Lücken wird kaum gelingen.

Die hier vorgestellte Suche nach ähnlichen Dateien beruht grundsätzlich darauf, dass ausgehend von der kleinsten ermittelten normierten Differenzsumme für ein Template weitere Imagelokalisationen betrachtet werden, deren normierte Differenzsummen innerhalb der definierten Abweichung nach oben liegen (10 %, 20 % oder 30 %). Um Dopplungen zu vermeiden und den Aufwand für die weitere Bearbeitung zu reduzieren, werden redundante normierte Differenzsummen eliminiert. Hierdurch ist es möglich, dass Ergebnisse mit einer Übereinstimmung zwischen ermittelter und tatsächlicher Imagelokalisation herausfallen. Die Untersuchung des Images „Similar Files“ mit den vorgestellten Templates weist keine belastbaren Sucherfolge auf. Die Verwendung von „Dateiversionen“ beim Image „Versions“ hingegen schon. Mit Hilfe der Templates „Konsensussequenz Frau“ und „Tiger“ können Imagelokalisationen ähnlicher Dateien eingegrenzt werden. Auch wenn keine sichere Identifizierung gelingt, scheint der Einsatz von Basenhäufigkeiten auch hier ein weiter zu verfolgender Ansatz zu sein. Die Suche allein auf Basis von Alignments kann vermutlich Imagebereiche mit relevantem Material eingrenzen, die Untersuchungsergebnisse zeigen allerdings eine wenig praktikable Vorgehensweise auf (Abschnitt 7.4.2).

Es kann vermutet werden, dass die Angleichung der Bildergrößen bzw. Sequenzlängen für das Multiple Alignment weniger bedeutend sind als die Sequenzauswahl zur Erzeugung der Konsensussequenz. Die zufällig ausgesuchten Frauen- und Männergesichter führen trotz der Größenangleichung zu Konsensussequenzen, die als Templates zur Suche nach ähnlichen Dateien im Image „Similar Files“ erfolglos blieben. Die Suche mit dem Template „Konsensussequenz Frau“, beruhend auf Versionen einer Datei, im Image „Versions“ zeigt demgegenüber positive Ergebnisse. Sowohl die Bilderauswahl als auch der Einsatz verschiedener Fenstergrößen zur Erzeugung des Multiplen Alignments in Kombination mit unterschiedlichen Scoring Matrices und Lückenbewertungen bedarf weiterer Untersuchungen hinsichtlich der Ergebnisbeeinflussung.

Die Zulassung der oben beschriebenen Abweichungen kann - unabhängig von der Ähnlichkeitssuche - bei der Suche mit Hilfe von Basenhäufigkeiten zu einer größeren Anzahl korrekt identifizierter Dateien führen, wenn man die weiteren Imagelokalisationen innerhalb der Abweichung in die Analyse einbezieht (Bild 53).

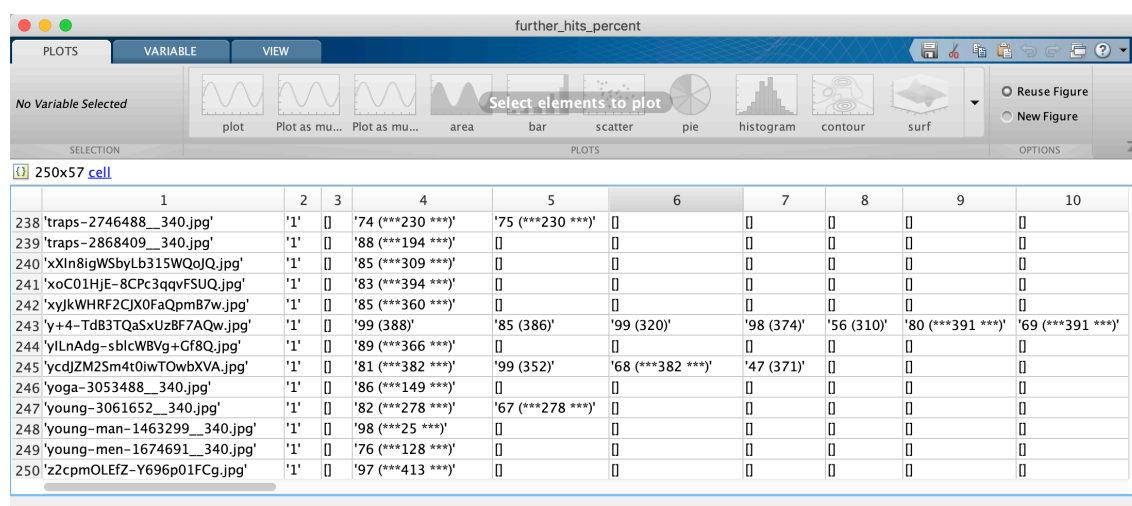


Bild 53: Weitere Imagelokalisationen/-dateien bei einer Abweichung von 10 % (4-Tupel, Image V)

In Bild 53 ist ein Auszug aus der Ergebnisdatei zur orientierenden Suche mit Hilfe von Basenhäufigkeiten im Image V mit der Tupellänge vier und einer zulässigen Abweichung von 10 % für die *normDiffSum* zu sehen. In Zeile 243 ist in Spalte neun die korrekte Identifizierung der Templatedatei erkennbar. Sich wiederholende Markierungen \*\*\* in

einer Zeile kennzeichnen einen vergleichbaren Imageabschnitt, der durch den Shift nach Formel 6.2 vom Untersuchungsfenster in unterschiedlichem Maße überdeckt wird.

## 9.5 Vergleichende Betrachtung mit etablierten Verfahren

### 9.5.1 Unveränderte Daten

Der Vergleich mit den etablierten Forensiktools zeigt deutlich, dass sie bei intakten Filesystemen das Mittel der Wahl darstellen. Sie sind sehr schnell und grundsätzlich existieren dann keine falsch-positiven und falsch-negativen Resultate. Hier zeichnen sich die Softwarepakete *Autopsy*, *X-Ways* und *EnCase* aus. Sie sind vergleichsweise einfach zu bedienen und liefern umgehend Analyseergebnisse. Auch mit *sdhash* werden gute Ergebnisse erzielt, es sind jedoch falsch-positive Ergebnisse möglich. Es fällt auf, dass die Eingrenzung der Lokalisation im Image - z. B. für weitere manuelle Überprüfungen - mit *sdhash* nicht immer gelingt (Tabelle 44). Mit den etablierten Verfahren werden grundsätzlich alle unveränderten Dateien aufgedeckt, allerdings ist der manuelle bzw. visuelle Abgleich erforderlich, da bei der Analyse des Filesystems ein Vergleich mit Templates (relevantes Datenbankmaterial) nicht stattfindet.

In Bezug auf die *FNR* sind die hier eingesetzten bioinformatischen Methoden durchaus mit den etablierten Verfahren vergleichbar. Mit *FPR* zwischen sieben und über 90 % schneiden die bioinformatischen Methoden deutlich schlechter als *Autopsy*, *X-Ways* und *EnCase* ab. Auch *sdhash* kann hier nicht überzeugen, ist jedoch vergleichsweise ein sehr schnelles Verfahren - die Vergleiche zwischen Image und Templates benötigen in der Regel weniger als eine Minute bei der eingesetzten Hardware.

### 9.5.2 Fragmentierte Daten

Bei nicht intakten Filesystemen oder Fragmenten reduzieren sich die Analysemöglichkeiten. Im Vordergrund stehen dann Hashing-Verfahren und das File Carving. Das Forensiktool *Scalpel* als der Vertreter des File Carvings ist sehr schnell, wobei nur bekannte *Header* und *Footer* genutzt werden können. Zusätzlich muss der Nutzer alle rekonstruierten Dateien anschauen bzw. Zusatzprogramme einsetzen, die die rekonstruierten Dateien mit bekannten vergleichen, um das relevante Material aufzufinden. Es soll erwähnt werden, dass das File Carving im Vergleich zu den im Image vorhandenen Dateien möglicherweise eine deutlich größere Anzahl an Dateien

rekonstruiert. Dies hängt u. a. damit zusammen, dass Dateien eingebettete Bilder enthalten - eine *pdf*-Datei mit zehn eingebetteten Bildern ergibt nach der Analyse insgesamt elf rekonstruierte Dateien. Die Untersuchungsergebnisse des File Carvings hängen vom eingesetzten Forensiktool und den Einstellungen (Carving-size) ab. Das File Carving sucht zunächst nicht nach Fragmenten, sollte jedoch die Carving-size überschritten sein, kann dies als Hinweis auf vorliegende Fragmente gewertet werden.

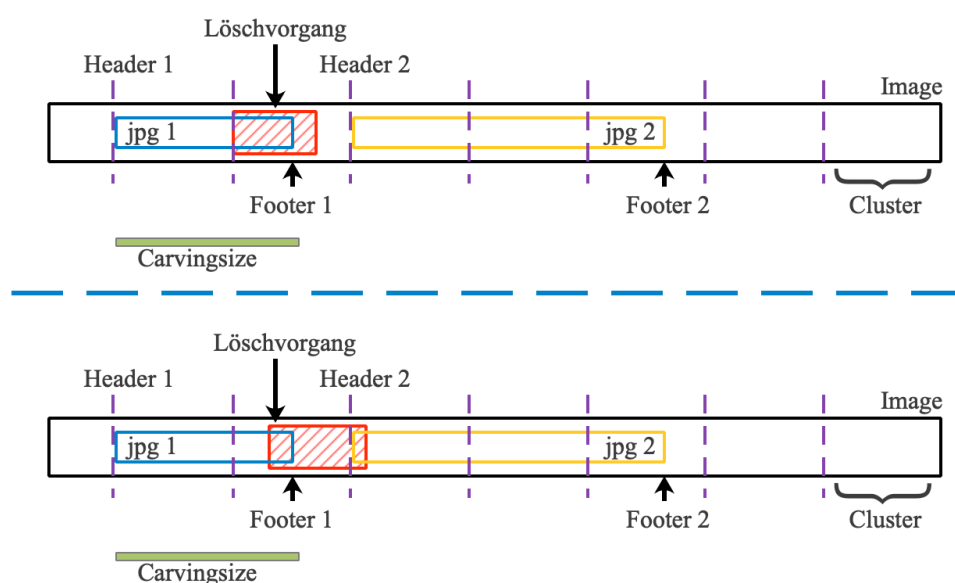


Bild 54: Löschvorgänge, die für die Rekonstruktion ein Überschreiten der Carving-size bedingen

In Bild 54 sind zwei Fälle gezeigt, wobei im oberen Teil der *Footer 1* und im unteren Teil *Footer 1* und *Header 2* gelöscht worden sind. Das Carving beginnend beim *Header* wird mit der voreingestellten Carving-size in beiden Fällen zunächst keinen zugehörigen *Footer* finden. Im ersten Fall liegt ein Fragment der Datei *jpg 1* vor, im zweiten Fall existieren jeweils ein Fragment der Datei *jpg 1* und *jpg 2*. Die Rekonstruktion der Datei kann auf die Carving-size beschränkt sein und führt auf Grund des fehlenden *Footers* in beiden Fällen zu einer „defekten“ Datei. Die Dateirekonstruktion mit dem Überschreiten der Carving-size führt ebenfalls zu einer nicht lesbaren Datei, da die Metainformationen in *Header 1* nicht zu der rekonstruierten Datei mit dem *Footer 2* passen (Bild 55).



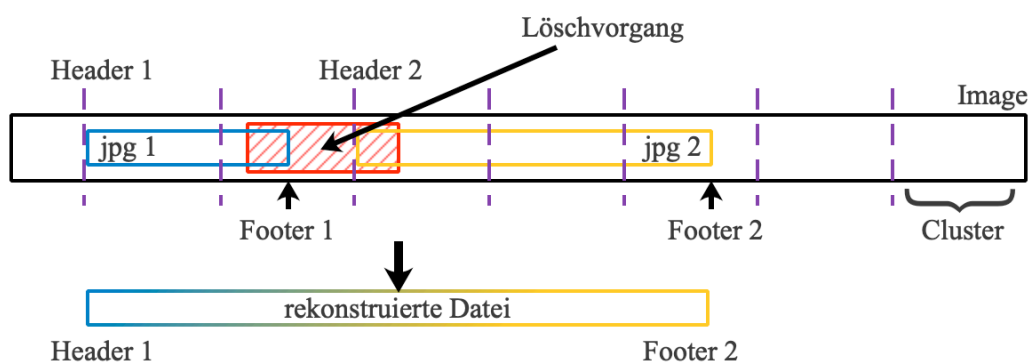


Bild 55: „Fehlerhafte“ Dateirekonstruktion (vollständige Überlappung der Lokalisationsvektoren)

Die Imagelokalisationen der rekonstruierten Datei und der in die Rekonstruktion einfließenden Imagedateien überlappen sich vollständig. Dies bedeutet, dass vor allem rekonstruierte, aber mit Standardsoftware nicht lesbare Dateien einer genaueren Betrachtung bzw. weiteren Untersuchung bedürfen, da sie fragmentierte Daten enthalten können.

Die im Rahmen der Arbeit eingesetzte Analyse der gecarvten Dateien beruht darauf, dass aus der physikalischen Lokalisation und Länge der gecarvten Datei ein Vektor gebildet und mit den Imagevektoren (tatsächliche Imagelokalisation und Dateilänge) verglichen wird. Eine Überlappung der beiden Vektoren (vollständig, wie in Bild 55 gezeigt oder unvollständig gemäß Bild 56) wird hier als Fragmentfund gewertet. Das in ein *pdf*-Dokument eingebettete *jpg*-Bild führt auf Grund der Löschvorgänge zu einer „fehlerhaft“ rekonstruierten Datei.

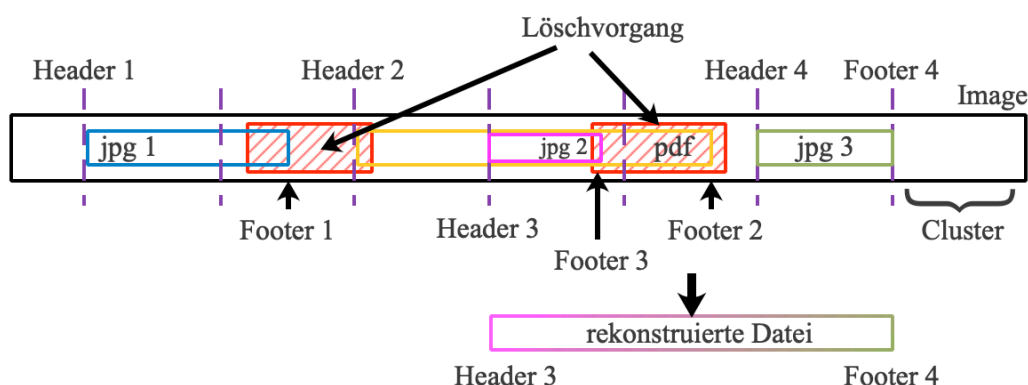


Bild 56: „Fehlerhafte“ Dateirekonstruktion (unvollständige Überlappung der Lokalisationsvektoren)

Diese Vorgehensweise ermöglicht einen Vergleich der betrachteten Verfahren, gleichzeitig ist aber eine zielgerichtete Fragmentsuche so nicht realisierbar. Die Untersuchungsergebnisse sind weniger belastbar als die Analyseergebnisse der unveränderten Dateien. Die hier eingesetzte (bioinformatische) orientierende Suche ist deutlich langsamer, ist von bekannten Mustern in Form von *Header/Footer* aber unabhängig und liefert Imagelokalisationen des relevanten Materials, da die Suche templatebasiert erfolgt. Eine auf Fragmente bezogene Suche ist durch eine Anpassung des bioinformatischen Untersuchungsfensters und dem Einsatz von Block Hashes möglich. Das Block Hashing des Forensiktools *X-Ways* liefert templatebasierte Resultate; das heißt, die zu einer Datei hinterlegten Block Hashes (aktuell ist nur eine Blockgröße von 512 Bytes möglich) können in einem unbekannten Image die Templates bzw. auch Fragmente auffinden, einschließlich der Angaben der Imagelokalisationen. Wie oben beim File Carving beschrieben, gibt es einige Erklärungsansätze für das nicht vollständige Aufdecken aller Dateien bzw. Dateifragmente in manipulierten Images. Ebenso sind beim Hashing negative Resultate möglich. In Abhängigkeit von der gewählten Blockgröße für das Hashing werden möglicherweise nicht alle Fragmente gefunden. Die in Bild 57 gezeigten Dateifragmente zwei und drei werden mit einer Blockdefinition gleich der Clustergröße nicht erkannt. Der mit \* markierte Teil des ersten Fragmentes innerhalb des ersten Dateiclusters ergibt einen anderen Hashwert als der ursprüngliche Clusterinhalt und bleibt unerkannt. Im zweiten Dateicluster befindet sich der unveränderte Dateirest, mit einer hohen Wahrscheinlichkeit für einen positiven Hashwertvergleich.

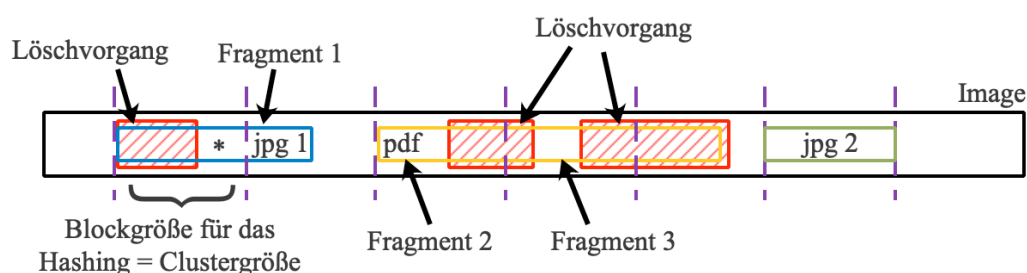


Bild 57: Block Hashing und Dateifragmente

Die zu Beginn der Arbeit erwähnten „robusten“ Hashing-Verfahren (Abschnitt 2.1) könnten hier Vorteile bieten, aber auch eine frei wählbare Verschiebung des „Hashing-

Fensters“ innerhalb des Images und Templates wird voraussichtlich fallbezogen bessere Ergebnisse liefern.

Die in Bild 44 dargestellte Extraktion der Slack Space-Inhalte mit *X-Ways* zeigt selbst unter Berücksichtigung unterschiedlicher Möglichkeiten des Text Encoding (z. B. ASCII, UTF-8, UTF-16) keine Übereinstimmung mit Inhalten des Images I bzw. der in den Clustern drei und vier vorhandenen Inhalten der Datei *sunset-1759716\_1280.jpg*. Inwieweit durch *X-Ways* eine Verarbeitung der extrahierten Daten stattfindet, ist nicht ersichtlich.

Mit allen etablierten Verfahren und auch den bioinformatischen Methoden findet keine gezielte Fragmentsuche statt, dennoch werden durch das Block Hashing Blöcke bzw. Abschnitte der zu untersuchenden Dateien gebildet (also Fragmente) und miteinander verglichen. Die vorgestellte bioinformatische Vorgehensweise scheint grundsätzlich den etablierten Verfahren unterlegen, wobei die gewählte Funktionalität (File Carving oder Hashing) einen wesentlichen Einfluss auf das Untersuchungsergebnis hat. Sowohl beim File Carving als auch Hashing werden durch die etablierten Verfahren nicht in jedem Falle die relevanten Fragmente vollumfänglich aufgedeckt (Tabelle 46). So kommt es, dass die bioinformatischen Ergebnisse unterlegen, gleichwertig oder überlegen sein können. Damit reiht sich die Bioinformatik hier in das Verfahrensportfolio ein, ist jedoch auf Grund der Zeitkomplexität nicht das Mittel der ersten Wahl.

### 9.5.3 Ähnliche Daten

Die mit *sddhash* in *X-Ways* zunächst vergleichbar erscheinende Funktionalität *PhotoDNA* kann nicht getestet werden, da sie nur Strafverfolgungsbehörden zur Verfügung steht. Gemäß ([17], S. 128ff) und ([36], S. 3) testen diese Ähnlichkeitsvergleiche Abweichungen eines Templates, sogenannte Versionen einer Datei. Die Funktionalität *PhotoDNA* dürfte das erheblich breitere Anwendungsspektrum bedienen können, da es im Gegensatz zu *sddhash* auch Veränderungen in Form von Dateitypänderungen, Größenanpassungen oder Bildverpixelungen „erkennen“ soll.<sup>28</sup> Die Analyse der Images „Similar Files“ und „Versions“ mit *sddhash* konnte für kein Template ähnliche Dateien mit einem (signifikanten) Score > 20 nachweisen. Auch die in Abschnitt 7.4.3 dargestellten „Bildversionen“ werden von *sddhash* nicht erkannt. Die hier vorgestellte

<sup>28</sup> Vergleichbare Ausführungen zum Begriff „Ähnlichkeit“ wie in [17] finden sich in [36] nicht.

bioinformatische Variante der Ähnlichkeitssuche ist ebenso templatebasiert (Sequenz eines Bildes oder Konsensussequenz). Die Suche mit Hilfe von Basenhäufigkeiten kann ähnliche Dateien erkennen. Von Bedeutung sind die Tupellänge und die Auswahl der Templates. Bei den Templates scheint die Anwendung von Konsensussequenzen vorteilhafter zu sein, wobei die zur Erzeugung der Konsensussequenz eingesetzten Dateien einen großen Einfluss haben. Mit Hilfe von Basenhäufigkeiten können also Imagelokalisationen (hier maximal 100) eingegrenzt werden, die mit hoher Wahrscheinlichkeit ähnliche Dateien enthalten. Mit Blick auf das Image „Versions“ konnten unter 100 Imagelokalisationen ca. die Hälfte der ähnlichen *girl*-Dateien und alle ähnlichen Tiger-Bilder aufgefunden werden (Abschnitt 7.4.3). Diese Ergebnisse zeigen Möglichkeiten der Ähnlichkeitssuche mit Hilfe von Konsensussequenzen auf. Das Abstützen der Suche allein auf die Berechnung von Alignmentsscores ergab im Vergleich zum Einsatz von Basenhäufigkeiten schlechtere Resultate. Dennoch ist hier Potenzial einer erfolgreichen Ähnlichkeitssuche mit bioinformatischen Methoden erkennbar. Ein wesentlicher limitierender Faktor dürfte darin bestehen, dass je nach verwendetem Bild bzw. Bildausschnitt als Template unterschiedliche Bildabschnitte verglichen werden, die zu entsprechend starken Abweichungen von der kleinsten *normDiffSum* oder den maximalen globalen und lokalen Alignmentsscores führen können. Dieser Umstand ist in Bild 58 visualisiert.

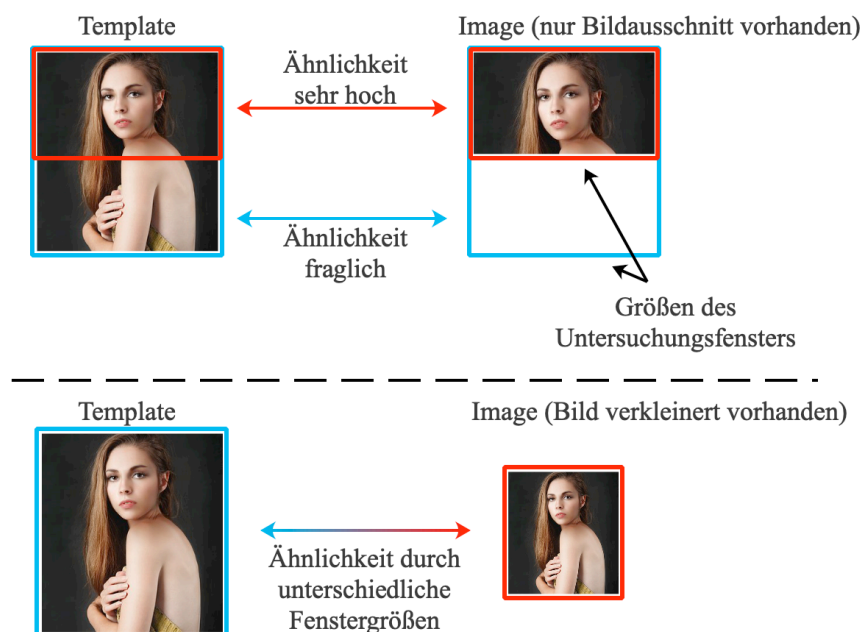


Bild 58: Suche nach ähnlichen Bildern mit unterschiedlichen Fenstergrößen

Lösungsansätze für eine Verbesserung der Ähnlichkeitssuche mit den hier aufgezeigten Vorgehensweisen könnte darin bestehen, kleinere Fenstergrößen zu verwenden (oberer Teil in Bild 58), um gezielt unterschiedliche Templateausschnitte mit dem Image abzugleichen, oder für die Analyse des Images und Templates unterschiedliche Fenstergrößen zu wählen (Bild 58, unterer Teil). Insgesamt muss man derzeit aber festhalten, dass die hier dargestellten bioinformatischen Methoden in Form von Multiplen Alignments oder der zulässigen Abweichung bei der orientierenden Suche kaum praktikabel sind.

### 9.6 Zusammenfassung in Bezug auf die Zielstellung

Die in Kapitel 3 formulierten Fragestellungen können wie folgt beantwortet werden:

1. **Kann die Analyse von Basenhäufigkeiten zur Suche nach (unveränderten) Dateien eingesetzt werden?**

Dateien bzw. Daten können mit Hilfe von Basenhäufigkeiten erfolgreich identifiziert werden. Falsch-negative Resultate kommen mit einer Wahrscheinlichkeit  $\leq 6\%$  für Basenquadrupel vor (*FNR*), dieser stehen vergleichsweise hohe falsch-positive Ergebnisse (*FPR*) gegenüber. *FPR* und *FNR* können durch den Einsatz von Schwellenwerten reduziert werden.

2. **Kann die Analyse von Basenhäufigkeiten zur Suche nach Dateifragmenten eingesetzt werden?**

Dateifragmente können mit Hilfe von Basenhäufigkeiten identifiziert werden. Der Sucherfolg ist von der Größe des Untersuchungsfensters und der eingesetzten Tupellänge abhängig.

3. **Bedarf es für die Anwendung bioinformatischer Algorithmen ggf. einer Vorverarbeitung der Eingabedaten?**

Zur Anwendung der im Rahmen der Arbeit genutzten bioinformatischen Funktionen in *Matlab* ist eine Übersetzung in das Basen- oder Aminosäuren-Alphabet erforderlich.

**4. Können bioinformatische Algorithmen zur Erzeugung von globalen und lokalen Alignments erfolgreich im Rahmen der Datensuche eingesetzt werden?**

Die Suche nach (unveränderten) Daten kann auch auf Basis von Alignments erfolgreich durchgeführt werden. Im Vordergrund steht hier jedoch die Verbesserung der Ergebnisse der orientierenden Suche mit Hilfe von Basenhäufigkeiten. Die so für das jeweilige Template ermittelte wahrscheinlichste Imagelokalisation wird mit globalen und lokalen Alignments weiter untersucht, um das Ergebnis zu bestätigen oder abzulehnen (Reduzierung der *FPR*, eine *FNR* in Höhe von 0 % ist möglich). Es ist davon auszugehen, dass andere Scoring Matrices und Lückenbewertungen, als die im Rahmen der Arbeit eingesetzten, die Ergebnisse noch verbessern können, da hierdurch u. a. bei lokalen Alignments die zwischen Image und Template gemeinsamen Teilsequenzen besser ermittelt werden.

**5. Können Multiple Alignments zur Suche nach ähnlichen Dateien eingesetzt werden?**

Multiple Alignments sind zur Suche nach ähnlichen Dateien einsetzbar. Die Dateiauswahl, die für das Alignment gewählte Fenstergröße und die für den Abgleich zwischen Image und Template gewählten Untersuchungsfenster sind wichtige Einflussfaktoren, die weiterer Untersuchungen bedürfen. Die hier vorgestellte Vorgehensweise ist für den praktischen Einsatz so nicht geeignet.

**6. Sind die Ergebnisse beim Einsatz bioinformatischer Algorithmen mit denen etablierter Forensiktools, z. B. *Autopsy*, vergleichbar?**

Bei intakten Filesystemen und unveränderten Dateien sind die *FNR* der hier vorgestellten bioinformatischen Methoden mit den etablierten Verfahren vergleichbar. Die Schnelligkeit spricht in jedem Falle für die etablierten Verfahren. Betrachtet man Dateifragmente kann die Bioinformatik bessere Ergebnisse liefern, da die etablierten Verfahren nicht immer alle Fragmente aufdecken und teilweise schlechtere „Erkennungsraten“ als die bioinformatischen Methoden aufweisen. Hinsichtlich der Suche nach ähnlichen Dateien ist ein Vergleich so nicht möglich. *Sdhash* soll als einziges hier betrachtetes etabliertes Verfahren (die entsprechende Funktionalität in *X-Ways* ist nur

Strafverfolgungsbehörden zugänglich) auch ähnliche Dateien erkennen können, wobei sich die Ähnlichkeit über Versionen definiert; grundsätzlich bedeutet das, ausgehend von einer Originaldatei werden Veränderungen beispielsweise in Form von Farbanpassungen oder Anpassungen durch Filter vorgenommen. Der Einsatz von Basenhäufigkeiten in Verbindung mit Konsensussequenzen als Template kann ähnliche Dateien aufdecken, wobei hier lediglich eine Eingrenzung der Imagelokalisationen und keine gezielte Identifizierung möglich ist. Dies gilt auch nur für die verwendeten „Dateiversionen“, eine Suche nach für den Menschen ähnlich aussehenden Bildern - mit Hilfe eines Frauengesichtes als Template sollen weitere andere Gesichter aufgefunden werden - blieb erfolglos. *Sdhash* ermittelte in den zur Ähnlichkeitssuche eingesetzten Images keine ähnlichen Dateien.

## 10 Zusammenfassung der Arbeit

Die Arbeit untersucht den Einsatz bioinformatischer Methoden zur Datensuche in der IT-Forensik. Hierzu werden in *Python* und *Matlab* spezielle Skripte erstellt, mit deren Hilfe die Imageuntersuchungen durchgeführt werden. In einem ersten Schritt erfolgt das Transkribieren des Bitmusters der Images und zum Vergleich verwendeten Templates in eine Basensequenz, bestehend aus den Buchstaben {A, C, G, T}. Nur so ist der Einsatz der bioinformatischen Algorithmen möglich. Anschließend wird jedes Image und ggf. Template mit einem Untersuchungsfenster abgefahren. Die Größe des Fensters ist variabel, beträgt standardmäßig aber die Länge des jeweiligen Templates. Für jede Fensterposition wird die Basenhäufigkeit im Image bestimmt und mit der zuvor ermittelten Basenhäufigkeit im Template verglichen (orientierende Suche). Die Fensterposition mit der geringsten auf die Fenstergröße normierten Differenzsumme zwischen diesen beiden Häufigkeitsprofilen wird als wahrscheinlichste Imagelokalisation des Templates weiter untersucht. Die Basenhäufigkeiten können für unterschiedliche Tupellängen berechnet werden. Die Untersuchungen starten mit 1-Tupel ({A, C, G, T}) und gehen über 2-Tupel ({AA, AC, AG, AT, ..., TA, TC, TG, TT}) bis zu einer maximalen Tupellänge von vier. Der für bioinformatische Algorithmen sehr hohe Ressourcenbedarf setzt hier die Grenzen. Zunächst werden unveränderte Dateien untersucht, wobei die orientierende Suche mit einer Wahrscheinlichkeit für falsch-negative Resultate (*FNR*) in einer Höhe von  $\leq 6\%$  ab einer Tupellänge vier gute Ergebnisse liefert. Hingegen zeigten die Wahrscheinlichkeiten für falsch-positive Resultate (*FPR*) mit Werten teilweise über 90 % eine eher schlechte Dateidiskriminierung auf Basis der hier verwendeten Basenhäufigkeiten auf. Die weitere Untersuchung der so gefundenen Imagelokalisation mit der Scoreberechnung globaler und lokaler Alignments ergab eine Reduzierung der *FPR* und *FNR* (u. a. eine *FNR* in Höhe von 0 % für Basenquadrupel und lokale Alignments). Mit der Einführung von Schwellenwerten für die normierten Differenzsummen und die globalen bzw. lokalen Alignmentsscores ist eine weitere Verringerung der *FPR* und *FNR* möglich. Mit Hilfe der Häufigkeitsprofile können auch unvollständige Dateien (Fragmente) aufgedeckt werden. Die hierzu manipulierten Images mit dem Überschreiben von *Header* und *Footer* können mit der orientierenden Suche erfolgreich nach Fragmenten analysiert werden. Der Erfolg



hängt von der Größe des Untersuchungsfensters und der gewählten Tupellänge ab. Grundsätzlich kann davon ausgegangen werden, dass je kleiner das Dateifragment ist, desto kleiner ist auch die Größe des Untersuchungsfensters zu wählen. Nachdem unveränderte Dateien mit der orientierenden Suche in Verbindung mit Alignments und unvollständige Dateien mit der orientierenden Suche aufgefunden werden können, erfolgt die Erweiterung der Untersuchungen um Multiple Alignments zur Erzeugung von sogenannten Konsensussequenzen, mit deren Hilfe ähnliche Daten im Image gefunden werden sollen. Die Suche nach ähnlichen Daten ist möglich, allerdings nur für „Dateiversionen“; das heißt, ausgehend von einer Originaldatei werden z. B. durch die Anwendung von Bildbearbeitungsfiltern weitere Bilder im gleichen Dateiformat erzeugt. Bei intakten Filesystemen sind die etablierten Verfahren schon auf Grund ihrer Schnelligkeit den hier verwendeten bioinformatischen Methoden deutlich überlegen. Bei nicht intakten Filesystemen oder unvollständigen Dateien kann die Bioinformatik überlegen sein. Überschriebene Speicherbereiche, wie bei den manipulierten Images, erzwingen den Einsatz von Spezialverfahren, z. B. das File Carving mit *Scalpel* oder die Analyse mit dem Vergleich von Hashwerten. Die Untersuchungsergebnisse hängen von den gewählten Blockgrößen beim Block Hashing (*X-Ways*, *sdhash*) bzw. der Größe des Untersuchungsfensters bei den bioinformatischen Methoden, dem gewählten Hashing-Verfahren, den eingesetzten *Header-/Footer*-Signaturen (*Scalpel* und *EnCase*) und den Nutzereinstellungen (z. B. Konfigurationsdatei bei *Scalpel*) ab. Das Auffinden aller Fragmente kann nicht durch jedes etablierte Verfahren sichergestellt werden. Direkt vergleichbare Funktionalitäten zur Ähnlichkeitssuche weisen die getesteten etablierten Verfahren *Autopsy*, *Scalpel*, *X-Ways*, *EnCase* und *sdhash* nicht auf. *Sdhash* als das einzige hierbei verfügbare und mit einer „Ähnlichkeitsfunktionalität“ (wie auch *X-Ways*) ausgestattete etablierte Verfahren konnte hier im Rahmen der Arbeit nicht überzeugen. Insgesamt ist festzustellen, dass bioinformatische Methoden, hier in Form des Vergleichs von Basenhäufigkeiten und Alignments, auch in der IT-Forensik eingesetzt werden können. Hierzu muss das Bitmuster in eine Basensequenz übersetzt werden, um eine Weiterverarbeitung mit bioinformatischen Methoden zu ermöglichen. Es konnte sowohl die erfolgreiche Suche nach unveränderten Daten als auch nach Fragmenten gezeigt werden. Das Auffinden ähnlicher Dateiinhalte scheint möglich, bedarf aber zwingend weiterer Untersuchungen. Insgesamt können die durchgeführten Untersuchungen den

erfolgreichen Einsatz bioinformatischer Methoden zur Datensuche in der IT-Forensik aufzeigen. Dabei werden die eingesetzten bioinformatischen Methoden - Basenhäufigkeiten, globale/lokale und multiple Alignments - möglichst breit dargestellt. Die Vielzahl an Parametrisierungsmöglichkeiten kann hier nur einen Überblick vermitteln, wobei erfolgversprechende Ansätze identifiziert werden, die in weiteren Untersuchungen betrachtet werden sollten. Im Vergleich zu den getesteten etablierten Verfahren ist davon auszugehen, dass die bioinformatischen Methoden gleichmächtig sind. Die Einschränkungen beruhen im Wesentlichen auf der deutlich höheren Zeit- und Platzkomplexität der bioinformatischen Methoden.

## 11 Ausblick

Auf Basis der gezeigten Ergebnisverbesserungen bei einer Erhöhung der Tupellänge sollten weitere Längen über vier untersucht werden. Der hierfür benötigte Ressourcenansatz kann sich durch bessere bzw. optimierte Algorithmen vermutlich erheblich reduzieren - z. B. Verringerung der Berechnungsdauer mit der Implementierung von *parfor*-Schleifen in *Matlab* oder der Einsatz von BLAST. In diesem Zusammenhang ist auch zu überlegen, die hier durchgeführte Übersetzung von jeweils 2 Bits in eine Base auf weitere Bits zu erweitern, z. B. Transkription von 3 Bits in das Aminosäuren-Alphabet (dabei werden nicht alle Alphabet-Elemente genutzt). Für einen künftigen praktischen Einsatz in der IT-Forensik sollten Programmiersprachen gewählt werden, deren Laufzeitumgebungen das Zusammenspiel zwischen Hard- und Software effizient ausnutzen - dies ist nicht immer der Fall (vgl. [3], S. 43ff; [29], S./Folie 4f oder [37], S. 83f). In diesem Zusammenhang könnten auch Cloud-Services gewinnbringend genutzt werden. Die Verbesserung der orientierenden Suche in Verbindung mit einer Reduzierung von falsch-positiven und falsch-negativen Ergebnissen ist auch durch eine sektoren-/blockbezogene Suche zu erwarten. Dies kann auf Grund der Werte in Tabelle 6 vermutet werden. Auch sollten eine Untersuchung unterschiedlicher Tupellängen und Dateiformate erfolgen, um die Eignung von Basenhäufigkeiten für eine dateitypspezifische Diskriminierung - soweit vorhanden - aufzudecken. In diesem Zusammenhang könnte die Berechnung globaler und lokaler Alignmentscores in Verbindung mit einer formatspezifischen Betrachtung (z. B. nur *pdf*-Dokumente oder Bilder im *jpg*-Format) die Wahrscheinlichkeiten für falsch-positive und falsch-negative Ergebnisse der orientierenden Suche weiter reduzieren helfen. Für die Ähnlichkeitssuche ist möglicherweise der Einsatz weiterer Aspekte Multipler Alignments, z. B. das Alignieren von Profilen, von Vorteil. Vor allem die Durchführung des Multiplen Alignments in einem Schritt über die vollständigen Sequenzlängen dürfte sich positiv auswirken. Hinzu kommt die sorgfältige Auswahl ähnlicher Dateien, für die Durchführung Multipler Alignments. Der „bioinformatische Baukasten“ enthält noch mehr Verfahren, z. B. den Sequenzvergleich durch die Berechnung von Distanzmaßen wie bei *seqpdist()* in *Matlab*, die in weiteren Untersuchungen betrachtet werden sollten. Darüber hinaus ist zu bedenken, dass bei globalen Alignments das Alignieren zweier

Sequenzen immer bei der ersten Basenposition erfolgt. Bei lokalen Alignments werden zusätzlich zum Score auch die Startpositionen in den beiden Sequenzen (Template und Image) angegeben, dies könnte zur Korrektur des Untersuchungsfensters eingesetzt werden (Shift), um auf dieser Basis einen erneuten (orientierenden) Suchdurchlauf zu optimieren. Zusätzlich dürften die Suchergebnisse durch unterschiedliche Scoring Matrices und Lückenbewertungen beeinflusst werden.

## Literaturverzeichnis

- [1] Baun Christian: Betriebssysteme kompakt. Heidelberg, Berlin : Springer-Verlag GmbH, 2017.- ISBN 978-3-662-53142-6.
- [2] Bayerisches Staatsministerium des Innern, für Sport und Integration: IT-Spezialisten bei der Bayerischen Polizei gesucht. Auf der Internetseite unter: Medien - Pressemitteilungen - Presse-Archiv, München, 2. August 2017.  
<https://www.stmi.bayern.de/med/pressemitteilungen/pressearchiv/2017/297b/index.php>  
[letzter Zugriff: 5. Juli 2019].
- [3] Beazley David M.: Python Referenz. O. O. : o. V., o. J. Online-Version der Übersetzung von David Beazleys Buch *Python Referenz*, Markt+Technik, 2001.- ISBN 3-8272-5959-2.  
[https://www.reportlab.com/media/pix/RLIMG\\_f32c7c0a6db7b442726ebb8b3d3e3d0e.PDF](https://www.reportlab.com/media/pix/RLIMG_f32c7c0a6db7b442726ebb8b3d3e3d0e.PDF)  
[letzter Zugriff: 5. Juli 2019].
- [4] Böckenhauer Hans-Joachim, Bongartz Dirk: Algorithmische Grundlagen der Bioinformatik. Modelle, Methoden und Komplexität. Serie: *Leitfäden der Informatik*. 1. Auflage Mai 2003. Wiesbaden : B. G. Teubner Verlag / GWV Fachverlage GmbH, 2003.- ISBN 3-519-00398-8.
- [5] Borchers Detlef: Bayerns Polizei sucht IT-Professionals, IT-Forensiker und IT-Kriminalisten. Heise online - News, 25. Oktober 2018, 18:13 Uhr.  
<https://www.heise.de/newsticker/meldung/Bayerns-Polizei-sucht-IT-Professionals-IT-Forensiker-und-IT-Kriminalisten-4204434.html> [letzter Zugriff: 5. Juli 2019].
- [6] Breiting Frank: On the utility of bitwise approximate matching in computer science with a special focus on digital forensics investigations. Darmstadt, Technischen Universität Darmstadt, Computer Science Department, Dissertation, 2014.
- [7] Breiting Frank, Winter Christian, Yannikos York, Fink Tobias, Seefried Michael: Using Approximate Matching to Reduce the Volume of Digital Data. In: Peterson Gilbert, Shenoj Sujeet (Eds.) : *Advances in Digital Forensics X*. International Federation for Information Processing (IFIP) Advances in Information and Communication Technology (AICT) 433, 2014, S. 149 - 163.
- [8] Buchanan William Johnston, Graves Jamie Robert, Bose Niladri: United States Patent, Digital Forensics. Inquisitive Systems Limited, Edinburgh (GB), November 11, 2014, Patent No.: US 8,887,274 B2.
- [9] Bundesamt für Sicherheit in der Informationstechnik: Leitfaden „IT-Forensik“. Version 1.0.1 (März 2011). Bonn : Bundesamt für Sicherheit in der Informationstechnik, 2010.
- [10] Bundesamt für Sicherheit in der Informationstechnik: IT-Grundschutz-Kataloge. 15. Ergänzungslieferung - 2016. Bonn : o. V., 2016.
- [11] Bundeskriminalamt: Cybercrime. Bundeslagebild 2014. Wiesbaden : Bundeskriminalamt (BKA), 2014.
- [12] Bundeskriminalamt: Cybercrime. Bundeslagebild 2015. Wiesbaden : Bundeskriminalamt (BKA), 2015.
- [13] Bundeskriminalamt: Cybercrime. Bundeslagebild 2016. Wiesbaden : Bundeskriminalamt (BKA), 2016.
- [14] Bundeskriminalamt: Cybercrime. Bundeslagebild 2017. Wiesbaden : Bundeskriminalamt (BKA), 2018.
- [15] Carrier Brian: Python Autopsy Module Tutorial #2: The Data Source Ingest Module. Autopsy, Digital Forensics, Blog, Aug 11, 2015. In Verbindung mit „Python Autopsy Module Tutorial #1: The File Ingest Module“. Autopsy, Digital Forensics, Blog, Jul 7, 2015.  
<https://www.autopsy.com/python-autopsy-module-tutorial-2-the-data-source-ingest-module/>,  
[letzter Zugriff: 5. Juli 2019].
- [16] Chang Jeff, Chapman Brad, Friedberg Iddo, Hamelryck Thomas, de Hoon Michiel, Cock Peter, Antao Tiago, Talevich Eric, Wilczyński Bartek: biopython. Biopython Tutorial and Cookbook. Last Update - 10 July 2017 (Biopython 1.70). O. O. : o. V., 2017.

- 
- [17] Fleischmann Stefan: X-Ways Software Technology AG. X-Ways Forensics/WinHex. Integrated Computer Forensics Environment. Data Recovery & IT Security Tool. Hexadecimal Editor for Files, Disks & RAM. Manual. X-Ways Software Technology AG, 2017.  
<http://www.x-ways.net/winhex/manual.pdf> [letzter Zugriff: 5. Juli 2019].
- [18] Frischholz Andreas: Personalmangel: Polizei und Geheimdienste finden keine Hacker. ComputerBase, Netzpolitik, 3. April 2017 17:44 Uhr.  
<https://www.computerbase.de/2017-04/polizei-geheimdienste-hacker-personal/> [letzter Zugriff: 5. Juli 2019].
- [19] Geschonneck Alexander: Computer Forensik. Computerstraftaten erkennen, ermitteln, aufklären. 6., aktualisierte und erweiterte Auflage. Heidelberg : dpunkt.verlag GmbH, 2014.- ISBN 978-3-86490-133-1.
- [20] Guidance Software: EnCase® Forensic. User Guide. Version 8.07. April 27, 2018.  
<http://encase-docs.opentext.com/documentation/encase/forensic/8.07/Content/Resources/External%20Files/Encase%20Forensic%20v8.07%20User%20Guide.pdf> [letzter Zugriff: 5. Juli 2019].
- [21] Gusfield Dan: Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology. New York, Melbourne : The Press Syndicate of the University of Cambridge, 1997.- ISBN13: 978-0-521-67035-7.
- [22] Harichandran Vikram S., Breiter Frank, Baggili Ibrahim: Byte-wise Approximate Matching: The Good, the Bad, and the Unknown. In: Rogers Marcus (Eds.) : *Journal of Digital Forensics, Security and Law (JDFSL)*, 2016, Volume 11, Number 2, S. 59 - 78.
- [23] Hütt Marc-Thorsten, Dehnert Manuel: Methoden der Bioinformatik. Eine Einführung zur Anwendung in Biologie und Medizin. 2. Auflage. Berlin, Heidelberg : Springer-Verlag, 2016.- ISBN 978-3-662-46149-5.
- [24] Jahankhani Hamid, Watson David Lilburn, Me Gianluigi, Leonhardt Frank: Handbook of Electronic Security and Digital Forensics. Singapore : World Scientific Publishing Co. Pte. Ltd., 2010.- ISBN 13 978-981-283-703-5.
- [25] Jain Anil K., Ross Arun A., Nandakumar Karthik: Introduction to Biometrics. New York, Dordrecht, Heidelberg, London : Springer Science+Business Media, LLC, 2011.- ISBN 978-0-387-77325-4.
- [26] Kornblum Jesse: Identifying Almost Identical Files Using Context Triggered Piecewise Hashing. In: Casey Eoghan (Eds.) : *Digital Investigation*, Volume 3, Supplement, 2006, S. S91 - S97.
- [27] Kröger Ben: Cyberkriminelle nutzen den Fachkräftemangel: Offene Stellen sind offene Einfallsstore! Axians - Cybersecurity - Managed Services, 5. April 2018.  
<https://www.axians.de/de/blog/2018/04/05/managed-security-services-cyberkriminelle-nutzen-den-fachkraeftemangel/> [letzter Zugriff: 5. Juli 2019].
- [28] Lander Eric S., Linton Lauren M., Birren Bruce et al. (International Human Genome Sequencing Consortium): Initial sequencing and analysis of the human genome. In: *Nature*, Volume 409, 2001, S. 860 - 921.
- [29] Liedlgruber Michael: Anwendungssoftware III (MATLAB), VII - Spezielle Themen. Fachbereich Computerwissenschaften, Universität Salzburg, Sommersemester 2014.  
<https://docplayer.org/9089035-Anwendungssoftware-iii-matlab.html> [letzter Zugriff: 5. Juli 2019].
- [30] Lin Alan C., Peterson Gilbert L.: Activity Pattern Discovery from Network Captures. Institute of Electrical and Electronics Engineers (IEEE) Security and Privacy Workshops, 2016, S. 334 - 342.
- [31] Merkl Rainer, Waack Stephan: Bioinformatik. Interaktiv. Grundlagen, Algorithmen, Anwendungen. 2., erweiterte und neubearbeitete Auflage. Weinheim : Wiley-VCH Verlag GmbH & Co. KGaA, 2009.- ISBN 978-3-527-32594-8.
- [32] Mulazzani Martin, Neuner Sebastian, Kieseberg Peter, Huber Markus, Schrittwieser Sebastian, Weippl Edgar: Quantifying Windows File Slack Size and Stability. In: Peterson G., Sheno S. (Eds.): *Advances in Digital Forensics IX*, International Federation for Information Processing (IFIP) Advances in Information and Communication Technology (AICT) 410, Chapter 13, 2013, S. 183 - 193.

- 
- [33] Needleman Saul B., Wunsch Christian D.: A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. In: Wright Peter (Eds.) : *Journal of Molecular Biology*, Volume 48, Issue 3, 1970, S. 443 - 453.
- [34] Rauhut Reinhard: Bioinformatik. Sequenz-Struktur-Funktion. Weinheim : Wiley-VCH Verlag GmbH, 2001.- ISBN 3-527-30355-3.
- [35] Roussev Vassil: An Evaluation of Forensic Similarity Hashes. In: Casey Eoghan (Eds.) : *Digital Investigation*, Volume 8, Supplement, 2011, S. s34 - s41.
- [36] Roussev Vassil, Quates Candice: sdhash tutorial. Release 0.8. August 06, 2013.  
<http://roussev.net/sdhash/tutorial/sdhash-tutorial.pdf>  
[letzter Zugriff: 5. Juli 2019].
- [37] Scholz Peter: Softwareentwicklung eingebetteter Systeme. Berlin, Heidelberg : Springer-Verlag, 2005.- ISBN 3-540-23405-5.
- [38] Singh Gautam B.: Fundamentals of Bioinformatics and Computational Biology. Methods and Exercises in MATLAB. Series: *Modeling and Optimization in Science and Technologies (MOST)*, Volume 6. Cham, Heidelberg, New York, Dordrecht, London : Springer International Publishing Switzerland, 2015.- ISBN 978-3-319-11402-6.
- [39] Smith T. F., Waterman M. S.: Identification of Common Molecular Subsequences. In: Wright Peter (Eds.) : *Journal of Molecular Biology*, Volume 147, Issue 1, 1981, S. 195 - 197.
- [40] Strutz Tilo: Bilddatenkompression. Grundlagen, Codierung, Wavelets, JPEG, MPEG, H.264. 4. Auflage. Wiesbaden : Vieweg + Teubner | GWV Fachverlage GmbH, 2009.
- [41] The GIMP Documentation Team: GNU Image Manipulation Program. Benutzerhandbuch, 2019.  
<https://docs.gimp.org/2.10/de/> [letzter Zugriff: 5. Juli 2019].
- [42] Tsukasa Oi, Kornblum Jesse: ssdeep Project | Getting Started with ssdeep - Quick Start. Last updated on 2017-09-25.  
<https://ssdeep-project.github.io/ssdeep/usage.html> [letzter Zugriff: 5. Juli 2019].
- [43] Veenman Cor J.: Statistical Disk Cluster Classification for File Carving. Third International Symposium on Information Assurance and Security, Manchester, 29. - 31. August 2007, S. 393 - 398.
- [44] Weiß Christel: Basiswissen Medizinische Statistik. 6., überarbeitete Auflage. Berlin, Heidelberg : Springer-Verlag, 2013.- ISBN 978-3-642-34260-8.
- [45] Wick Vanessa: Fachkräftemangel: IT-Forensiker gesucht! IT-Zoom, 15. November 2017.  
<https://www.it-zoom.de/it-director/e/fachkraeftemangel-it-forensiker-gesucht-18290/>  
[letzter Zugriff: 5. Juli 2019].
- [46] Willer Christoph: PC-Forensik. Daten suchen und wiederherstellen. 1. Auflage. Böblingen : C&L Computer und Literaturverlag, 2012.- ISBN 978-3-936546-60-6.
- [47] Wundram Martin, Sigel Alexander: Anti-Forensik - Techniken, mit denen Täter IT-forensische Maßnahmen behindern und Gegenmaßnahmen. In: Bund Deutscher Kriminalbeamter (Hrsg.): *Der Kriminalist*. Fachzeitschrift des BDK, Ausgabe 9 / September 2013, S. 37 - 42.

## Bilderverzeichnis

Bild 1: Forensischer Prozess, orientierend an den Vorgaben des BSI (vgl. [9], S. 24f) .....	10
Bild 2: Zusammenhang zwischen Image- und Template-Bytemuster (IT-Forensik) .....	14
Bild 3: Zusammenhang zwischen Image- und Template-Sequenzmuster (Bioinformatik) .....	15
Bild 4: Speicherzuweisung, Cluster und Slack Space .....	19
Bild 5: Auszug der Byte-Struktur von <i>indian.jpg</i> ( <i>iHex</i> ).....	22
Bild 6: Anwendung einer Scoring Matrix .....	24
Bild 7: Berechnung globaler Alignments (das optimale Alignment erfordert keine Lücken).....	26
Bild 8: Berechnung globaler Alignments (das optimale Alignment erfordert das Einfügen von Lücken) .....	27
Bild 9: Berechnung von lokalen Alignments.....	28
Bild 10: Beispiel für ein Multiples Alignment mit Konsensussequenz .....	29
Bild 11: Auszug aus einer FASTA-Datei .....	33
Bild 12: Übersetzung von Bytes in Nukleinbasen.....	34
Bild 13: Codon-Häufigkeit versus Basentupel-Häufigkeit .....	35
Bild 14: Abgleich zwischen Image und Template - orientierende Suche .....	36
Bild 15: Unterschied zwischen den Häufigkeitsprofilen eines Images und Templates .....	38
Bild 16: Untersuchungsfenster im Image .....	40
Bild 17: Erstellung von Testimages zur Skriptüberprüfung .....	43
Bild 18: Templates für den Algorithmustest der orientierenden Suche.....	44
Bild 19: Alignments in Ergänzung zur orientierenden Suche.....	48
Bild 20: Vorgehensweise in der Arbeit .....	53
Bild 21: Beispiel für den Vergleich von Basenhäufigkeiten, Template nicht vorhanden (2-Tupel).....	55
Bild 22: Beispiel für den Vergleich von Basenhäufigkeiten, Template vorhanden (2-Tupel).....	56
Bild 23: <i>FPR</i> und <i>FNR</i> für <i>normDiffSum</i> , kein Schwellenwert - unveränderte Daten .....	59
Bild 24: <i>FPR</i> und <i>FNR</i> für Schwellenwert <i>normDiffSum</i> ( <i>ut</i> ) - unveränderte Daten .....	62
Bild 25: Globale Alignmentsscores für das Image V (4-Tupel).....	66
Bild 26: Lokale Alignmentsscores für das Image V (4-Tupel).....	70
Bild 27: Suche auf Basis von Alignments .....	75
Bild 28: Suche auf Basis von Alignments (Image V).....	77
Bild 29: Orientierende Suche mit Hilfe von Basenhäufigkeiten (3-Tupel, Image V).....	78
Bild 30: Allozierte Speicherbereiche, Slack Space und Bearbeitung des Images I.....	81
Bild 31: Erzeugung einer Konsensussequenz .....	85
Bild 32: Bilder für die Suche nach Ähnlichkeiten im Image „Similar Files“ .....	86
Bild 33: Suche auf Basis von Alignments in einem Imageabschnitt ( <i>girl-3023853_340.jpg</i> ) .....	88
Bild 34: Suche auf Basis von Alignments in einem Imageabschnitt (Konsensussequenz).....	88
Bild 35: Templates zur Untersuchung des Images „Versions“ .....	89
Bild 36: Suche auf Basis von Alignments im Image „Versions“ ( <i>girl_TkaAYYBIy2lfL4Has37dQ.jpg</i> ) ...	91
Bild 37: Suche auf Basis von Alignments im Image „Versions“ (Konsensussequenz).....	91
Bild 38: Hashdatenbank mit Block Hashes in <i>X-Ways</i> (Beispiel).....	93
Bild 39: Ergebnisse der Analyse auf Basis von Block Hashes in <i>X-Ways</i> (Beispiel).....	93
Bild 40: Auszug aus der Konfigurationsdatei von <i>Scalpel</i> .....	95
Bild 41: Konfigurationsdatei für das Carving mit <i>X-Ways</i> (Auszug).....	96
Bild 42: Beispiel für das File Carving mit <i>X-Ways</i> .....	96
Bild 43: Auswahl der Carving-Typen in <i>EnCase</i> (Auszug).....	97
Bild 44: Extraktion des Slack Spaces aus dem Image I mit <i>X-Ways</i> (Auszug).....	101



Bild 45: Vergleich der Differenzsummen für zwei Sequenzen in Abhängigkeit von der Tupellänge.....	106
Bild 46: Analyseergebnisse des Images L mit Basen-4-Tupel (links ohne und rechts mit Shift) .....	109
Bild 47: Anwendung des Shifts gemäß Formel 6.2.....	110
Bild 48: Beispiel für eine angepasste Abbruchbedingung (Shift) .....	111
Bild 49: Globale Scores für unterschiedliche Startpositionen im Image .....	114
Bild 50: Visualisierung globaler Alignments mit <i>showalignment()</i> in <i>Matlab</i> .....	115
Bild 51: Teilsequenzen und Startangaben bei lokalen Alignments .....	116
Bild 52: Profiländerung bei Halbierung der Fenstergröße.....	120
Bild 53: Weitere Imagelokalisationen/-dateien bei einer Abweichung von 10 % (4-Tupel, Image V) ....	122
Bild 54: Löschvorgänge, die für die Rekonstruktion ein Überschreiten der Carvingsize bedingen.....	124
Bild 55: „Fehlerhafte“ Dateirekonstruktion (vollständige Überlappung der Lokalisationsvektoren).....	125
Bild 56: „Fehlerhafte“ Dateirekonstruktion (unvollständige Überlappung der Lokalisationsvektoren)...	125
Bild 57: Block Hashing und Dateifragmente.....	126
Bild 58: Suche nach ähnlichen Bildern mit unterschiedlichen Fenstergrößen.....	128
Bild 59: Auszug aus dem <i>Matlab</i> -Array <i>image_templates_compare</i> .....	157
Bild 60: Auszug aus dem <i>Matlab</i> -Array <i>globalAlign_scores</i> .....	157
Bild 61: Bildschirmausgabe der orientierenden Suche und globalen Alignmentberechnung in <i>Matlab</i> ..	159
Bild 62: Bildschirmausgabe der orientierenden Suche und lokalen Alignmentberechnung in <i>Matlab</i> ....	159
Bild 63: Auszug aus dem <i>Matlab</i> -Array <i>further_hits_percent</i> .....	160
Bild 64: Auszug aus dem <i>Matlab</i> -Array <i>align_scores</i> .....	160
Bild 65: Ablauf der orientierenden Suche mit der Berechnung von Alignments.....	165
Bild 66: Frauengesichter zur Erstellung einer Konsensussequenz.....	187
Bild 67: Männergesichter zur Erstellung einer Konsensussequenz.....	187
Bild 68: Kriterienbasierte Gesichterauswahl zur Erstellung einer Konsensussequenz.....	188
Bild 69: „Versionen“ des Templates 1, von links oben nach rechts unten: original-1.-2.-3.-4.-5. ....	189
Bild 70: „Versionen“ des Templates 2, von links oben nach rechts unten: original-1.-2.-3.-4.-5. ....	190
Bild 71: „Versionen“ des Templates 3, von links oben nach rechts unten: original-1.-2.-3.-4.-5. ....	190
Bild 72: Strukturbaum der Dateiablage auf dem Datenträger.....	191

## Tabellenverzeichnis

Tabelle 1: Vergleich von IT-Forensik und Bioinformatik.....	16
Tabelle 2: Transkription von System Calls in das Aminosäuren-Alphabet.....	17
Tabelle 3: Beispiel für Basenhäufigkeiten im Image und Template.....	38
Tabelle 4: Dateiformate für die Untersuchung von Basenhäufigkeiten (Images L / V / F).....	42
Tabelle 5: Testimages und Testtemplates (TestT) zur Überprüfung des <i>Matlab</i> -Skriptes.....	43
Tabelle 6: Ergebnisse der Untersuchung des Images „Test Algorithm“.....	45
Tabelle 7: Schema einer Vierfeldertafel.....	49
Tabelle 8: <i>NormDiffSum</i> der orientierenden Suche - unveränderte Dateien.....	56
Tabelle 9: Vierfeldertafel <i>normDiffSum</i> , kein Schwellenwert - unveränderte Daten.....	57
Tabelle 10: <i>FPR</i> und <i>FNR</i> Image L, <i>normDiffSum</i> , kein Schwellenwert - unveränderte Daten.....	58
Tabelle 11: <i>FPR</i> und <i>FNR</i> Image V, <i>normDiffSum</i> , kein Schwellenwert - unveränderte Daten.....	58
Tabelle 12: <i>FPR</i> und <i>FNR</i> Image F, <i>normDiffSum</i> , kein Schwellenwert - unveränderte Daten.....	58
Tabelle 13: Vierfeldertafel <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	61
Tabelle 14: <i>FPR</i> und <i>FNR</i> Image L, <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	61
Tabelle 15: <i>FPR</i> und <i>FNR</i> Image V, <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	61
Tabelle 16: <i>FPR</i> und <i>FNR</i> Image F, <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	62
Tabelle 17: Vierfeldertafel für Image V, kein Schwellenwert, unterschiedliche Fenstergrößen.....	64
Tabelle 18: Vierfeldertafel für Image V, <i>normDiffSum</i> $\leq ut$ , unterschiedliche Fenstergrößen.....	64
Tabelle 19: <i>FPR</i> und <i>FNR</i> Image V, keine Schwellenwerte, unterschiedliche Fenstergrößen.....	65
Tabelle 20: <i>FPR</i> und <i>FNR</i> Image V, <i>normDiffSum</i> $\leq ut$ , unterschiedliche Fenstergrößen.....	65
Tabelle 21: Normierte globale Scores - unveränderte Daten.....	66
Tabelle 22: Vierfeldertafel <i>globale Scores</i> $\geq lt$ - unveränderte Daten.....	67
Tabelle 23: <i>FPR</i> und <i>FNR</i> Image L, <i>globale Scores</i> $\geq lt_{global}$ - unveränderte Daten.....	68
Tabelle 24: <i>FPR</i> und <i>FNR</i> Image V, <i>globale Scores</i> $\geq lt_{global}$ - unveränderte Daten.....	68
Tabelle 25: <i>FPR</i> und <i>FNR</i> Image F, <i>globale Scores</i> $\geq lt_{global}$ - unveränderte Daten.....	68
Tabelle 26: Vierfeldertafel <i>globale Scores</i> $\geq lt$ , <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	69
Tabelle 27: <i>FPR</i> und <i>FNR</i> Image L, <i>globale Scores</i> $\geq lt$ , <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	69
Tabelle 28: <i>FPR</i> und <i>FNR</i> Image V, <i>globale Scores</i> $\geq lt$ , <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	69
Tabelle 29: <i>FPR</i> und <i>FNR</i> Image F, <i>globale Scores</i> $\geq lt$ , <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	70
Tabelle 30: Normierte lokale Scores - unveränderte Daten.....	71
Tabelle 31: Vierfeldertafel <i>lokale Scores</i> $\geq lt_{local}$ - unveränderte Daten.....	71
Tabelle 32: <i>FPR</i> und <i>FNR</i> Image L, <i>lokale Scores</i> $\geq lt_{local}$ - unveränderte Daten.....	72
Tabelle 33: <i>FPR</i> und <i>FNR</i> Image V, <i>lokale Scores</i> $\geq lt_{local}$ - unveränderte Daten.....	72
Tabelle 34: <i>FPR</i> und <i>FNR</i> Image F, <i>lokale Scores</i> $\geq lt_{local}$ - unveränderte Daten.....	72
Tabelle 35: Vierfeldertafel <i>lokale Scores</i> $\geq lt_{local}$ , <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	73
Tabelle 36: <i>FPR</i> und <i>FNR</i> Image L, <i>lokale Scores</i> $\geq lt_{local}$ , <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	73
Tabelle 37: <i>FPR</i> und <i>FNR</i> Image V, <i>lokale Scores</i> $\geq lt_{local}$ , <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	73
Tabelle 38: <i>FPR</i> und <i>FNR</i> Image F, <i>lokale Scores</i> $\geq lt_{local}$ , <i>normDiffSum</i> $\leq ut$ - unveränderte Daten.....	74
Tabelle 39: Ergebnisse der bioinformatischen Fragmentsuche, Fenstergröße = Templatelänge / halbe Templatelänge.....	80

Tabelle 40: <i>FNR</i> der bioinformatischen Fragmentsuche, Fenstergröße = Templatelänge / halbe Templatelänge .....	81
Tabelle 41: Anzahl korrekt identifizierter Fragmente für das Image I - kein Schwellenwert .....	82
Tabelle 42: Anzahl falsch-positiver Ergebnisse für das Image I - Abhängigkeit von Schwellenwerten ...	83
Tabelle 43: Anzahl der relevanten Fragmentgrößen zur ursprünglichen Dateigröße .....	83
Tabelle 44: Vierfeldertafel für die Datensuche mit etablierten Verfahren - unveränderte Dateien .....	98
Tabelle 45: <i>FPR</i> und <i>FNR</i> , etablierte Verfahren ↔ Bioinformatik - unveränderte Dateien .....	98
Tabelle 46: Ergebnisse der Fragmentsuche mit <i>Scalpel</i> , <i>X-Ways</i> , <i>EnCase</i> und <i>sdhash</i> .....	100
Tabelle 47: Für die Arbeit eingesetzte Hardware .....	147
Tabelle 48: Für die Untersuchungen eingesetzte Software .....	147
Tabelle 49: Auflistung der Image-Kenndaten .....	150
Tabelle 50: Dateien im Image L bzw. „L Fragments“ .....	150
Tabelle 51: Anpassungen im Image I .....	152
Tabelle 52: Globale Alignments abhängig von der Fenstergröße (1-Tupel) .....	153
Tabelle 53: Globale Alignments abhängig von der Fenstergröße (2-Tupel) .....	153
Tabelle 54: Globale Alignments abhängig von der Fenstergröße (3-Tupel) .....	153
Tabelle 55: Globale Alignments abhängig von der Fenstergröße (4-Tupel) .....	153
Tabelle 56: Lokale Alignments abhängig von der Fenstergröße (1-Tupel) .....	154
Tabelle 57: Lokale Alignments abhängig von der Fenstergröße (2-Tupel) .....	154
Tabelle 58: Lokale Alignments abhängig von der Fenstergröße (3-Tupel) .....	154
Tabelle 59: Lokale Alignments abhängig von der Fenstergröße (4-Tupel) .....	154
Tabelle 60: Hauptprogramme für die Untersuchungen .....	155
Tabelle 61: Hilfsprogramme zur Auswertung der Untersuchungsergebnisse .....	161
Tabelle 62: Übersetzung des Bitmusters in Basen .....	166

**Abkürzungsverzeichnis**

A	Adenin (Nukleinbase)
ASCII	American Standard Code for Information Interchange
aw	Alignment Window
BLAST	Basic Local Alignment Search Tool
BSI	Bundesamt für Sicherheit in der Informationstechnik
bzw.	beziehungsweise
C	Cytosin (Nukleinbase)
ca.	circa
csv	Comma Separated Values - Dateiformat
DDR	Double Data Rate
DIMM	Dual Inline Memory Module
DNA	Deoxyribonucleic acid (Desoxyribonukleinsäure, besteht aus Nukleinsäuren)
et al.	et alii / und andere (bei Literaturquellen; bei mehr als 10 Autoren werden nur die ersten drei genannt und mit et al. ergänzt)
f	folgende (bei Seitenangaben)
FASTA	Fast-All; Algorithmus zur Suche nach Ähnlichkeiten zwischen Sequenzen und ein Speicherformat für Sequenzdaten
FAT	File Allocation Table
ff	fortfolgende (bei Seitenangaben)
FNR	False Negative Rate
FPR	False Positive Rate
Fragm.	Fragmente
FTK	Forensic Toolkit
G	Guanin (Nukleinbase)
GB	Gigabyte
ggf.	gegebenenfalls
GPL	GNU General Public License
GHz	Gigahertz
gif	Graphics Interchange Format - Bilddatenformat
ico	Icon - Bilddatenformat
IT	Informationstechnik
jpg	Joint Photographic Experts Group (auch jpeg) - Bilddatenformat
KB	Kilobyte
max	Maximum
MB	Megabyte
MFT	Master File Table
MHz	Megahertz
min	Minimum
NTFS	New Technology File System
o. J.	ohne Jahresangabe (bei Literaturquellen)
o. O.	ohne Ortsangabe (bei Literaturquellen)
o. V.	ohne Verlagsangabe (bei Literaturquellen)
pdf	Portable Document Format - Dokumentenformat
png	Portable Network Graphics - Bilddatenformat

ppt	Microsoft® Office PowerPoint-Format - Präsentationsformat
RAM	Random Access Memory
RCP	Rich Client Platform
rel.	relevante
S.	Seite
t	Treshold
T	Thymin (Nukleobase)
lt	Lower Treshold
TestT	Testtemplate
TW	Template Window
txt	Dateiformat für Textdateien
u. a.	unter anderem
USB	Universal Serial Bus
ut	Upper Treshold
vgl.	vergleiche
xcf	Experimental Computing Facility - Bilddatenformat des freien Bildbearbeitungsprogramms <i>GIMP</i>
xlsx	Microsoft® Office Excel-Format - Tabellenformat
z. B.	zum Beispiel

## **Anlagenverzeichnis und Anlagen**

Anlage A - Auflistung der verwendeten Hard- und Software

Anlage B - Auflistung der Images

Anlage C - Unterschiedliche Fenstergrößen für die Untersuchung von Alignments

Anlage D - Erläuterungen zu den *Matlab-Skripten*

Anlage E - *Python-Klasse* zur Erstellung einer FASTA-Datei

Anlage F - *Python-Klasse* zur Erstellung von Dateifragmenten

Anlage G - *Matlab-Skript* zur Suche mit Hilfe von Basenhäufigkeiten und Alignments

Anlage H - *Matlab-Skript* zur Suche auf Basis von Alignments

Anlage I - Bilder zur Erzeugung von Konsensussequenzen und Bildversionen

Anlage J - Datenträgerinhalte

## Anlage A - Auflistung der verwendeten Hard- und Software

Die für die Untersuchungen eingesetzte Hardware ist in Tabelle 47 zusammengestellt.

Tabelle 47: Für die Arbeit eingesetzte Hardware

Hardware	Spezifikationen	Bemerkungen
<i>Dell Workstation Precision™ T7600 Series</i>	Prozessor: Intel® Xeon® CPU E5-2665 0 @ 2,40 GHz x 32; Arbeitsspeicher: 192 GB DIMM DDR3 1600 MHz	überwiegend Untersuchungen ( <i>Ubuntu 18.04 LTS</i> )
<i>Kingston®</i>	DTSE9 G2 USB 3.0 16 GB	partitioniert ( <i>MiniTool Partition Wizard Free 10.2.2</i> )
<i>Lenovo X220</i>	Intel® Core™ i5-2520M CPU @ 2,50 GHz; Arbeitsspeicher: 4,00 GB	Erstellung von Partitionen auf dem USB-Stick und Imageerstellung ( <i>Microsoft Windows 10 Pro</i> )
<i>MacBook Pro</i>	Retina 15“, Mitte 2015; Prozessor: Intel Core i7 2,5 GHz; Speicher: 16 GB 1600 MHz DDR3	überwiegend Analyse der Untersuchungsergebnisse ( <i>macOS Mojave + Microsoft Windows 10 Pro virtualisiert</i> )

Die für die Untersuchungen eingesetzte Software ist in Tabelle 48 zusammengestellt.

Tabelle 48: Für die Untersuchungen eingesetzte Software

Software	Spezifikationen	Bemerkungen
<i>AccessData® FTK® Imager</i>	3.4.3.3; 2015 AccessData Group, Inc.	Tool zur Erzeugung von Images für die Datensuche ( <i>Lenovo</i> )
<i>Autopsy</i>	Product Version: Autopsy 4.7.0 (Release); Sleuth Kit Version: 4.6.1; Netbeans RCP Build: 201609300101; Java: 1.8.0_152; Java HotSpot™ 64-Bit Server VM 25.152-b16	Forensiktool (virtualisiert, <i>MacBook</i> )
<i>Biopython</i>	Version 1.70	Sammlung von <i>Python</i> -Tools u. a. zur Bearbeitung bioinformatischer Vorgänge ( <i>MacBook</i> )
<i>Eclipse IDE for Java Developers</i>	Version: Oxygen.2 Release (4.7.2); Build id: 20171218-0600	Entwicklungsumgebung ( <i>MacBook</i> )

<b>Software</b>	<b>Spezifikationen</b>	<b>Bemerkungen</b>
<i>Edraw Max</i>	Version: 8.4; Release Date: July 30, 2016; Edraw Max Subscription License; EdrawSoft	Programm zur Erstellung der Skizzen und Grafiken in der Arbeit ( <i>MacBook</i> )
<i>EnCase® Forensic</i>	Version: 8.07.00.93; Platform: x86; Company: Guidance Software; Build: qaau0000010B 04/26/18 08:30:25 PM; Dongle ID G3002091056	Forensiktool (virtualisiert, <i>MacBook</i> )
<i>FreeFileSync 10.1</i>	Build: 03.06.18 - Unicode x64	Ordnervergleich und Synchronisation (virtualisiert, <i>MacBook</i> )
<i>GIMP</i>	Version 2.8.22	GNU Image Manipulation Program ( <i>MacBook</i> )
<i>iHex</i>	Version 2.3 (23); hewbo.com	Hex-Editor ( <i>MacBook</i> )
<i>macOS Mojave</i>	Version 10.14.5	Betriebssystem ( <i>MacBook</i> )
<i>MATLAB®</i>	R2018a (9.4.0.813654); 64-bit (maci64 und glnx64); February 23, 2018; License Number: STUDENT	Mathematik-Tool, Bioinformatics Toolbox version 4.10 ( <i>Dell und MacBook</i> )
<i>Microsoft® Office Excel</i>	Version 16.16.11 (190609); Produkt-ID: 02981-000-127802; Lizenz: Einzelhandelslizenz 2016	Tabellenkalkulationsprogramm ( <i>MacBook</i> )
<i>Microsoft® Office Word</i>	Version 16.16.11 (190609); Produkt-ID: 02981-000-127802; Lizenz: Einzelhandelslizenz 2016	Textverarbeitungsprogramm zur Erstellung der Arbeit ( <i>MacBook</i> )
<i>Microsoft Windows 10 Pro</i>	Version 1709; Betriebssystembuild 16299.344; Systemtyp: 64-bit	Betriebssystem (virtualisiert, <i>MacBook</i> )
<i>Microsoft Windows 10 Pro</i>	Version 1803; Betriebssystembuild 17134.829; 64-Bit-Betriebssystem, x64-basierter Prozessor	Betriebssystem ( <i>Lenovo</i> )
<i>MiniTool Partition Wizard Free 10.2.2</i>	Building Date: Jul 27 2017; Building Time: 13:59:30; MiniTool Solution Ltd.	Partitionierungstool ( <i>Lenovo</i> )
<i>Parallels Desktop® 14 für Mac Pro Edition</i>	Version 14.1.3 (45485); Parallels International GmbH	Virtualisierungstool ( <i>MacBook</i> )
<i>PicaLoader</i>	V1.71; VOWSoft, Ltd.	Bild-Downloader (virtualisiert, <i>MacBook</i> )



---

<b>Software</b>	<b>Spezifikationen</b>	<b>Bemerkungen</b>
<i>PyDev for Eclipse</i>	Version: 7.2.1.201904261721	<i>Python</i> für Eclipse DIE ( <i>MacBook</i> )
<i>Scalpel</i>	Version 2.0 Written by Golden G. Richard III and Lodovico Marziale	Forensiktool für das File Carving ( <i>MacBook</i> )
<i>sdhash</i>	3.1 by Vassil Roussev, Candice Quates, December 2012	Hashing-Tool ( <i>MacBook</i> )
<i>Ubuntu 18.04.2 LTS</i>	GNOME 3.28.2; 64-bit	Betriebssystem ( <i>Dell</i> )
<i>Vorschau</i>	Version 10.0 (944.5)	Programm zur Bildbearbeitung ( <i>MacBook</i> )
<i>X-Ways Forensics</i>	19.7 SR-3 x64 (04.10.2018); kostenlose Lizenz für zwei Monate	Forensiktool (virtualisiert, <i>MacBook</i> )

## Anlage B - Auflistung der Images

Die Image-Kenndaten sind in Tabelle 49 zusammengestellt. Die verwendeten Images sind auf dem beigefügten Datenträger enthalten.

Tabelle 49: Auflistung der Image-Kenndaten

Imagebezeichnung in der Arbeit <sup>29</sup>	Filesystem	Bytes Per Sector	Sector Count	Source Data Size [MB] <sup>30</sup>
F <sup>31</sup>	FAT	512	1.044.162	535
I <sup>32</sup> , L <sup>33</sup> , „L Fragments“ <sup>34</sup>	FAT	512	96.327	49
„Similar Files“ <sup>35</sup>	FAT	512	32.067	16
„Test Algorithm“ <sup>36</sup>	FAT	512	48.132	25
V <sup>37</sup>	NTFS	512	96.327	49
„Versions“	NTFS	512	18.845	10

Die Fragmentsuche erfolgt mit dem Image „L Fragments“ (Tabelle 50), dem Image I (Tabelle 51) und den modifizierten Images L, V und F<sup>38</sup>.

Tabelle 50: Dateien im Image L bzw. „L Fragments“

Dateiname (gemäß Filesystem)	Dateigröße [Bytes] <sup>39</sup>	Lokalisation [Startbyte] <sup>40</sup>	Bytes mit Nullen überschrieben
_70803~1.PDF (gelöschte Datei)	11.940.888	1.166.848	
170804_Biometrie_PraesenzFolienDSU pdt.pdf	12.152.328	22.567.936	140 (Header)

<sup>29</sup> Zur besseren Übersicht werden in der Arbeit die hier aufgeführten Kurzbezeichnungen verwendet. In den *Matlab*-Ergebnisdateien sind die vollständigen Imagenamen zu finden; diese ergeben sich nach der Übersetzung in eine Basensequenz wie folgt: *GenSeq\_<Imagename>\_dna.faa* oder *GenSeq\_<Imagename>\_dna.faa*.

<sup>30</sup> Werte gerundet.

<sup>31</sup> 180630\_Partition\_F.001

<sup>32</sup> 180227\_Partition\_I.001

<sup>33</sup> 180204\_Partition\_L\_USB16GB.001

<sup>34</sup> 180513\_Partition\_L\_ohneSektoren0-31undFragments.001

<sup>35</sup> 180908\_Partition\_F\_SimilarFiles.001

<sup>36</sup> 181022\_Image\_Test\_Algorithm.001

<sup>37</sup> 181010\_Image\_V.001

<sup>38</sup> Modified\_180204\_Partition\_L\_USB16GB.001, Modified\_181010\_Image\_V.001, Modified\_180630\_Partition\_F.001

<sup>39</sup> *Logical Size* gemäß der Auswertung mit *EnCase*.

<sup>40</sup> *Physical Location* gemäß der Auswertung mit *EnCase*.

<b>Dateiname</b> (gemäß Filesystem)	<b>Dateigröße</b> <b>[Bytes]<sup>39</sup></b>	<b>Lokalisation</b> <b>[Startbyte]<sup>40</sup></b>	<b>Bytes mit Nullen</b> <b>überschrieben</b>
9303_p10_cons_en.pdf	864.260	36.978.176	560 (Header)
asia-3023824__340.jpg (†)	65.536	49.069.056	
beautiful-girl-2003647__340.jpg (†)	61.339	49.200.640	
BioAPI2_gi-proc-108-010.pdf (†)	276.458	45.882.368	700 (Header)
Biometric_Standards_White_Paper_March2009.pdf	443.487	43.227.648	
Box-CAIQ-v3.0.1-2016-11-03.xlsx	249.561	47.204.864	
BSI_Fingerabdruckerkennung_Historie_eingebettet_indian.pdf	200.320	48.164.864	
BSI_Gesichtserkennung_JPG_eingebettet.pdf	473.419	42.300.416	
BSI_Gesichtserkennung.pdf (†)	252.512	46.951.936	
BSI_Iriserkennung.pdf (†)	342.075	45.208.064	280 (Header)
CAIQ_Atlassian_Box_Azure.xlsx	79.810	34.721.280	
community-150124_1280.png (†)	109.469	48.685.056	1.400 (Header)
composing-2925179__340.jpg (†)	56.487	49.262.080	
Einfuehrung_Gesichtserkennung.pdf	388.385	44.089.344	
elephants-196613_1920.jpg (†)	626.021	40.106.496	
fitness-863081__340.jpg (†)	50.504	48.877.056	23.240 (Header)
girl-56683_1280.jpg (†)	243.094	47.454.720	
girl-657753_1280.jpg (†)	261.302	46.434.304	2.800 (Header)
ICAO_Fotomustertafel.pdf	699.637	39.406.592	
indian_GaussWeichz.png (†)	239.400	47.697.920	
indian_GaussWeichz.xcf (†)	753.074	38.653.440	
indian.jpg (†)	81.636	48.794.624	
Introduction+to+Biometrics.pdf	21.778.373	789.504	280 (Header)
ISO19794_Overview.pdf (†)	151.514	48.533.504	
ISOIEC_2382-37_2012.pdf	385.307	44.477.952	
ISOIEC19794-4.pdf	809.974	37.842.944	
man-1508670__340.jpg (†)	65.105	49.134.592	5.600 (Header)
man-1508680__340.jpg (†)	72.989	48.927.744	140 (Header)
maple-moth-183186_1280.jpg (†)	453.524	42.774.016	
people-1013546_1920.jpg	552.008	40.732.672	
people-279457_1920.jpg	1.167.946	34.801.152	12.460 (Header)
pretty-351884_1920.jpg	481.239	41.819.136	

<b>Dateiname</b> (gemäß Filesystem)	<b>Dateigröße</b> <b>[Bytes]<sup>39</sup></b>	<b>Lokalisation</b> <b>[Startbyte]<sup>40</sup></b>	<b>Bytes mit Nullen</b> <b>überschrieben</b>
puppy-384647_1280.jpg (†)	226.477	47.938.048	
RossIntroMultibio_EUSIPCO07.pdf (†)	255.233	46.696.448	103.880 (Header)
sailors-903044_1920.jpg	1.008.348	35.969.536	
schwerpunkte.pdf	331.046	45.551.104	
sheep-1642874_1280.jpg (†)	167.921	48.365.568	
Studienanweisung_ Angewandte biometrische Systeme.pdf	533.202	41.285.120	
sunset-1759716_1280.jpg (†)	275.356	46.158.848	420 (Header)
water-1122166_1280.jpg (†)	344.276	44.863.488	
woman-1211448_1920.jpg (†)	416.814	43.672.064	
women-936549__340.jpg (†)	67.904	49.000.960	140 (Header)

In Tabelle 50 sind die Dateien und überschriebenen Bereiche im Image „L Fragments“ zusammengestellt. In allen entsprechenden Fällen ist der Header beginnend ab dem Startbyte im Image überschrieben. Der Zusatz (†) hinter dem Dateinamen bedeutet, dass diese Datei als Template für die Untersuchungen genutzt wird.

Die Modifikation der Images L, V und F umfasst das automatisierte Überschreiben von Imageabschnitten (250.000 Bytes) mit Nullen. Die Entfernung zwischen diesen Abschnitten beträgt 3.000.000 Bytes.

Die Anpassungen des Images I sind in Tabelle 51 zusammengestellt.

Tabelle 51: Anpassungen im Image I

<b>Template</b> (Länge [Bytes])	<b>Template</b> Start - Ende [Byte]	<b>Image</b> Start - Ende [Byte]
sunset-1759716_1280.jpg (275.356)	271.792 - 275.356	532 - 4.095
indian.jpg (81.636)	0 - 2.239	98.112 - 100.351
sheep-1642874_1280.jpg (167.921)	624 - 12.067	201.544 - 212.987
sheep-1642874_1280.jpg (167.921)	4 - 981	49.318.446 - 49.319.423

# Anlage C - Unterschiedliche Fenstergrößen für die Untersuchung von Alignments

Die Berechnung der globalen Alignments erfolgt für das Image V mit 100 Templates (im Image vorhanden) und den Fenstergrößen 16.000, 32.000, 48.00 und 64.000 Basen. Die errechneten Alignmentsscores der unterschiedlichen Fenstergrößen werden statistisch auf signifikante Unterschiede untersucht. Die tupelabhängigen Ergebnisse sind in Tabelle 52 bis Tabelle 55 zusammengefasst.

Tabelle 52: Globale Alignments abhängig von der Fenstergröße (1-Tupel)

U-Test [ <i>p</i> -Wert]	16.000	32.000	48.000	64.000
16.000	-	-	-	-
32.000	0,6460	-	-	-
48.000	0,5316	0,8202	-	-
64.000	0,4650	0,7471	0,8931	-

Tabelle 53: Globale Alignments abhängig von der Fenstergröße (2-Tupel)

U-Test [ <i>p</i> -Wert]	16.000	32.000	48.000	64.000
16.000	-	-	-	-
32.000	0,0159	-	-	-
48.000	0,0022	0,3506	-	-
64.000	0,0004	0,1440	0,5365	-

Tabelle 54: Globale Alignments abhängig von der Fenstergröße (3-Tupel)

U-Test [ <i>p</i> -Wert]	16.000	32.000	48.000	64.000
16.000	-	-	-	-
32.000	0,0013	-	-	-
48.000	$1,0322 \cdot 10^{-5}$	0,1258	-	-
64.000	$5,2135 \cdot 10^{-8}$	0,0083	0,2087	-

Tabelle 55: Globale Alignments abhängig von der Fenstergröße (4-Tupel)

U-Test [ <i>p</i> -Wert]	16.000	32.000	48.000	64.000
16.000	-	-	-	-
32.000	0,0007	-	-	-
48.000	$5,8310 \cdot 10^{-7}$	0,0616	-	-
64.000	$8,8978 \cdot 10^{-10}$	0,0018	0,1701	-

Die Untersuchung der Fenstergrößen wird ebenso für lokale Alignments durchgeführt. Die Ergebnisse sind Tabelle 56 bis Tabelle 59 zu entnehmen.

Tabelle 56: Lokale Alignments abhängig von der Fenstergröße (1-Tupel)

U-Test [ $p$ -Wert]	16.000	32.000	48.000	64.000
16.000	-	-	-	-
32.000	0,5592	-	-	-
48.000	0,4816	0,8815	-	-
64.000	0,3979	0,7250	0,8623	-

Tabelle 57: Lokale Alignments abhängig von der Fenstergröße (2-Tupel)

U-Test [ $p$ -Wert]	16.000	32.000	48.000	64.000
16.000	-	-	-	-
32.000	0,1196	-	-	-
48.000	0,0452	0,5389	-	-
64.000	0,0316	0,3751	0,7323	-

Tabelle 58: Lokale Alignments abhängig von der Fenstergröße (3-Tupel)

U-Test [ $p$ -Wert]	16.000	32.000	48.000	64.000
16.000	-	-	-	-
32.000	0,0040	-	-	-
48.000	$1,8017 \cdot 10^{-5}$	0,1133	-	-
64.000	$8,6109 \cdot 10^{-7}$	0,0173	0,3456	-

Tabelle 59: Lokale Alignments abhängig von der Fenstergröße (4-Tupel)

U-Test [ $p$ -Wert]	16.000	32.000	48.000	64.000
16.000	-	-	-	-
32.000	0,0012	-	-	-
48.000	$3,6351 \cdot 10^{-7}$	0,0482	-	-
64.000	$2,8951 \cdot 10^{-9}$	0,0023	0,2399	-

## Anlage D - Erläuterungen zu den *Matlab*-Skripten

### Hauptprogramme

In der folgenden Tabelle sind die für die Untersuchungen verwendeten Hauptprogramme aufgeführt, zu finden auf dem beigefügten Datenträger. Alle Programme können mit einem beliebigen Editor geöffnet und bearbeitet werden.

Tabelle 60: Hauptprogramme für die Untersuchungen

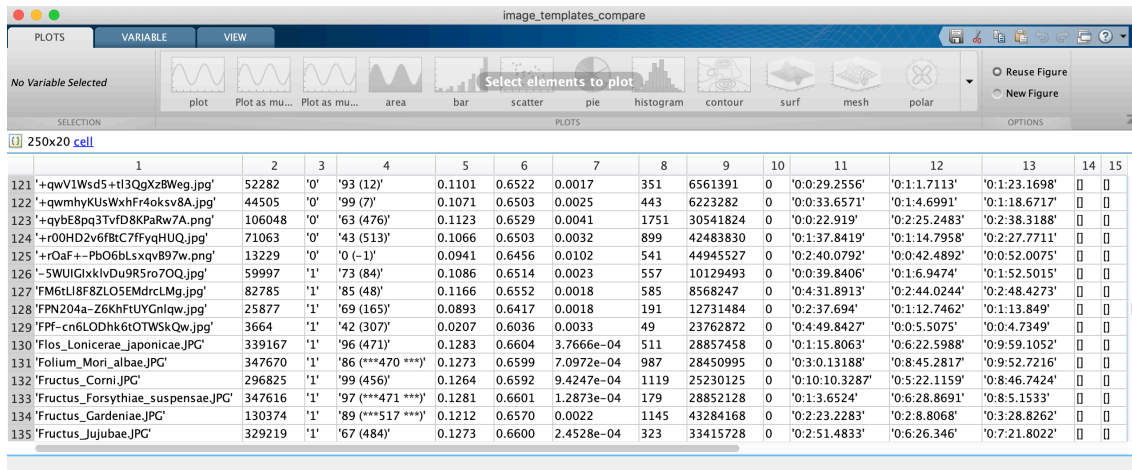
Nr.	Hauptprogramm	Beschreibung
1	<b>BaseRate_Alignments.m</b>	Suche nach der <b>wahrscheinlichsten Lokalisation</b> des Templates <b>im Image</b> mit Hilfe von Basenhäufigkeiten; auf Basis dieser Lokalisation Berechnung des maximalen globalen und lokalen Alignmentsscores; wesentliches Ausgabe-Array: <i>image_templates_compare</i>
2	<b>BaseRate_Alignments_SimilarFiles.m</b>	Suche nach der wahrscheinlichsten Lokalisation des Templates im Image wie bei Nr. 1 - darüber hinaus können <b>weitere Lokalisationen</b> mit einer definierten Abweichung ermittelt werden; auf Basis der ermittelten Lokalisationen jeweils Berechnung der maximalen globalen und lokalen Alignmentsscores; wesentliches Ausgabe-Array: <i>further_hits_percent</i>
3	<b>Alignments.m</b>	Suche nach der <b>wahrscheinlichsten Lokalisation</b> des Templates <b>im Image mit Hilfe allein von globalen und lokalen Alignmentsscores</b> ; wesentliches Ausgabe-Array: <i>image_templates_compare</i>
4	<b>Alignments_SimilarFiles.m</b>	Berechnung der globalen und lokalen <b>Alignmentsscores für alle Fensterpositionen</b> ; anschließend können die Scores für ähnliche und nicht ähnliche Dateien ausgewertet werden; wesentliches Ausgabe-Array: <i>align_scores</i>
5	<b>MultiAlignment.m</b>	Berechnung eines multiplen Alignments mit der <b>Erzeugung einer Konsensussequenz</b> , die als Template (Eingabe) für das Hauptprogramm Nr. 2 dient; Ausgabe: Konsensussequenz als Datei im FASTA-Format

Die Untersuchungsergebnisse sind in sogenannten Cell Arrays im *Matlab*-Workspace abgelegt. Die wesentlichen Ergebnisarrays sollen hier für das Verständnis erläutert werden. Der Array *image\_templates\_compare* enthält zu jedem Template der

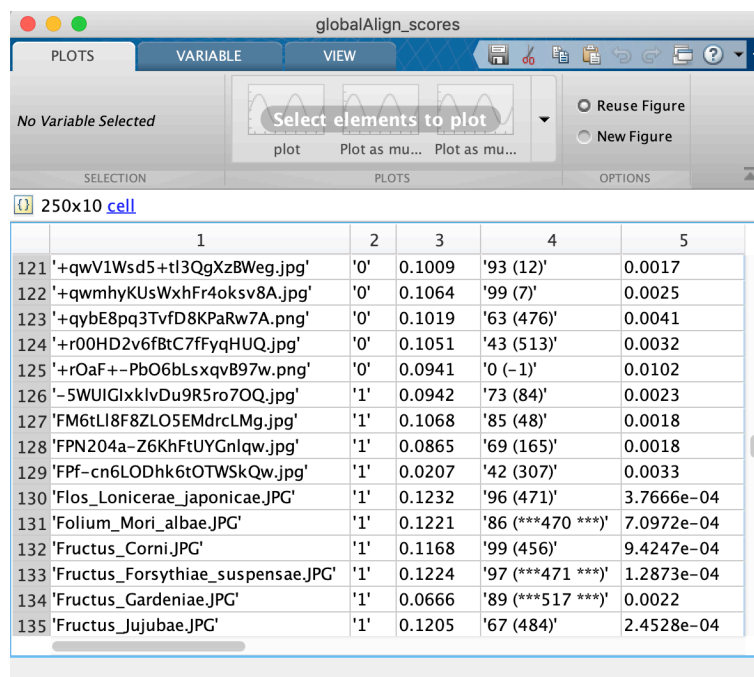
Untersuchung die wichtigsten Kenndaten und ist in Bild 59 dargestellt. Die Spaltenbedeutungen ergeben sich grundsätzlich wie folgt:

- Spalte 1: Templatename
- Spalte 2: Dateilänge [Byte]
- Spalte 3: im Image vorhanden („1“), nicht vorhanden („0“)
- Spalte 4: prozentuale Überdeckung mit einer Datei, deren Position im *Matlab*-Array *image\_locations* in Klammern angegeben ist; Ausgangspunkt ist die Imagelokalisation mit der kleinsten normierten Differenzsumme (die Markierung \*\*\* bedeutet, dass für die ermittelte Imagelokalisation der Templatename mit der Bezeichnung der Imagedatei übereinstimmt → korrektes Auffinden des Templates im Image)
- Spalte 5: größter globaler Score (wird nicht verwendet, keine Scoresumme, Analyse erfolgt über gesonderte Tabelle)
- Spalte 6: größter lokaler Score (wird nicht verwendet, keine Scoresumme, Analyse erfolgt über gesonderte Tabelle)
- Spalte 7: **kleinste normierte Differenzsumme**
- Spalte 8: Differenzsumme
- Spalte 9: Imagelokalisation (Startpunkt in Byte) mit der kleinsten normierten Differenzsumme
- Spalte 10: Templatelokalisation (Startpunkt in Byte) mit der kleinsten normierten Differenzsumme, erforderlich bei mehreren Templateblöcken (standardmäßig gilt: Fenstergröße = Templatelänge)
- Spalte 11: Zeit für die orientierende Suche (Vergleich der Basenhäufigkeiten)
- Spalte 12: Zeit für die Berechnung der globalen Alignments
- Spalte 13: Zeit für die Berechnung der lokalen Alignments
- Spalte 14: bleibt frei
- Spalte 15: ab dieser Spalte werden weitere mögliche Imagelokalisationen aufgeführt (z. B. Suche nach ähnlichen Dateien)




Bild 59: Auszug aus dem Matlab-Array `image_templates_compare`

Die **globalen bzw. lokalen Alignmentsscores** werden in den gesonderten Arrays `globalAlign_scores` (Bild 60) und `localAlign_scores` aufgeführt.


Bild 60: Auszug aus dem Matlab-Array `globalAlign_scores`

Die Spaltenbedeutungen ergeben sich wie folgt:

Spalte 1: Templatename

Spalte 2: im Image vorhanden („1“), nicht vorhanden („0“)

- Spalte 3: **maximaler globaler bzw. lokaler Score** (als Scoresumme der maximalen Fenstergröße in Höhe von 64.000 Basen für das Alignment)
- Spalte 4: prozentuale Überdeckung mit einer Datei, deren Position im *Matlab*-Array *image\_locations* in Klammern angegeben ist; Ausgangspunkt ist die Imagelokalisation mit der kleinsten normierten Differenzsumme
- Spalte 5: kleinste normierte Differenzsumme

Einen Eindruck hinsichtlich typischer Bildschirmausgaben des Hauptprogramms *BaseRate\_Alignments.m* sollen die Beispiele in Bild 61 und Bild 62 vermitteln. Im oberen Bereich von Bild 61 sieht man Angaben zu den untersuchten Dateien: Imagenname, Templatename, Dateilängen [Basen], Anzahl der Blöcke, verwendeter Shift, verwendeter Schwellenwert und die Größe des Untersuchungsfensters. Direkt darunter werden die einzelnen Suchdurchläufe (Verschiebung des Untersuchungsfensters) aufgelistet - hier insgesamt zwei. Es folgt das Ergebnis der orientierenden Suche, wobei neben der kleinsten normierten Differenzsumme u. a. der Start der zugehörigen Imagelokalisation in Basen und Bytes angegeben wird. Auf Basis der mit der orientierenden Suche identifizierten wahrscheinlichsten Imagelokalisation werden in einem weiteren Schritt die globalen Alignmentsscores berechnet. Es werden Image und Template blockweise aligniert, wobei der Start im Image der ermittelten wahrscheinlichsten Imagelokalisation entspricht. Es soll erwähnt werden, dass die Abweichung um eine Base (Imagestart für die *normDiffSum* und die Startbase im ersten Imageblock für das Alignment) dadurch entsteht, dass der Start im Image oder Template immer bei eins beginnen muss. Ein Shift der Größe 0 startet ebenfalls bei 1, wobei das Ergebnis in Basen im Ergebnisarray als Byte-Angabe gespeichert wird (der Basenwert wird durch vier geteilt und mit Hilfe der *Matlab*-Funktion *fix()* auf den nächsten Integer-Wert nahe null „fixiert“):  $\frac{187.868.929}{4} = 46.967.232,25 \rightarrow \text{fix}(46.967.232,25) = 46.967.232$ . Der Imagestart ergibt sich dann wie folgt:  $46.967.232 * 4 = 187.868.928$ . Da sich im Image unterschiedlich große Dateien befinden und der Start für die Dateispeicherung im Image grundsätzlich nicht bekannt ist, wird das Untersuchungsfenster die einzelnen Imagedateien kaum vollständig überdecken. Die Wahrscheinlichkeit für ein vorhandenes Template die entsprechende Imagelokalisation mit dem Untersuchungsfenster exakt aufzufinden, dürfte eher gering sein. Vor diesem Hintergrund und den Ergebnissen in Abschnitt 7.2 scheint die

Verschiebung zweier Sequenzen um eine Base mit Blick auf den Alignmentscore unschädlich zu sein.

```

Command Window
5/124-----
Vergleich Basenhäufigkeiten: GenSeq_Modified_180204_Partition_L_USB16GB.001_dna_.faa <-> GenSeq_BSI_Gesichtserkennung.pdf_dna.faa
Länge Image [Basen]: 197277696
Länge Template [Basen]: 1010048
Image-Blöcke: 196
Template-Blöcke: 1
Shift: 0
Schwellenwert: 0
Untersuchungsfenster: 1010048
***** 1. Untersuchung gestartet ***** (196 Blöcke)
***** 2. Untersuchung gestartet ***** (195 Blöcke)
Der kleinste normDiffSum beträgt (Image-/Templetestart: 187868929 / 1): 0.045347 [Byte: 46967232-47219744]

--Global-----Global-----Global-----Global-----Global--
Starte Berechnung globaler Alignments ...
Untersuchungsfenster: 64000
Untersuchungsblöcke: 16

Image-Block 1 (187868928-187932928) <-> Template-Block 1 (1-64001): 29259
Image-Block 4 (188060928-188124928) <-> Template-Block 4 (192001-256001): 34181
Image-Block 2 (187932928-187996928) <-> Template-Block 2 (64001-128001): 8070
Image-Block 10 (188444928-188508928) <-> Template-Block 10 (576001-640001): 19931
Image-Block 6 (188188928-188252928) <-> Template-Block 6 (320001-384001): 9320
Image-Block 8 (188316928-188380928) <-> Template-Block 8 (448001-512001): 7878
Image-Block 11 (188508928-188572928) <-> Template-Block 11 (640001-704001): 5402
Image-Block 5 (188124928-188188928) <-> Template-Block 5 (256001-320001): 39913
Image-Block 3 (187996928-188060928) <-> Template-Block 3 (128001-192001): 15287
Image-Block 9 (188380928-188444928) <-> Template-Block 9 (512001-576001): 6851
Image-Block 12 (188572928-188636928) <-> Template-Block 12 (704001-768001): 6978
Image-Block 7 (188252928-188316928) <-> Template-Block 7 (384001-448001): 5917
Image-Block 16 (188828928-188892928) <-> Template-Block 16 (960001-1024001): -1153
Image-Block 14 (188700928-188764928) <-> Template-Block 14 (832001-896001): 3734
Image-Block 13 (188636928-188700928) <-> Template-Block 13 (768001-832001): 7023
Image-Block 15 (188764928-188828928) <-> Template-Block 15 (896001-960001): 9329
Die Berechnung der globalen Alignments dauerte (h:m:s): 0:2:47.3236

--Local-----Local-----Local-----Local-----Local--
Starte Berechnung lokaler Alignments ...

```

Bild 61: Bildschirmausgabe der orientierenden Suche und globalen Alignmentberechnung in *Matlab*

Die Berechnung der lokalen Alignmentscores erfolgt analog zu den globalen Scores.

```

Command Window
--Local-----Local-----Local-----Local-----Local--
Starte Berechnung lokaler Alignments ...

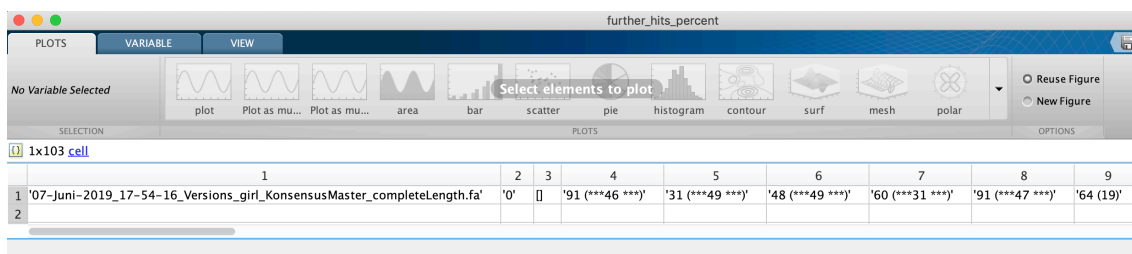
Image-Block 1 (1126400-1142400) <-> Template-Block 1 (1-16001): 16000 (Start@: [2;1])
Image-Block 2 (1142400-1158400) <-> Template-Block 2 (16001-32001): 16000 (Start@: [2;1])
Image-Block 4 (1174400-1190400) <-> Template-Block 4 (48001-64001): 16000 (Start@: [2;1])
Image-Block 7 (1222400-1238400) <-> Template-Block 7 (96001-112001): 16000 (Start@: [2;1])
Image-Block 12 (1302400-1318400) <-> Template-Block 12 (176001-192001): 16000 (Start@: [2;1])
Image-Block 17 (1382400-1398400) <-> Template-Block 17 (256001-272001): 16000 (Start@: [2;1])
Image-Block 22 (1462400-1478400) <-> Template-Block 22 (336001-352001): 16000 (Start@: [2;1])
Image-Block 3 (1158400-1174400) <-> Template-Block 3 (32001-48001): 16000 (Start@: [2;1])
Image-Block 27 (1542400-1558400) <-> Template-Block 27 (416001-432001): 16000 (Start@: [2;1])
Image-Block 6 (1206400-1222400) <-> Template-Block 6 (80001-96001): 16000 (Start@: [2;1])
Image-Block 11 (1286400-1302400) <-> Template-Block 11 (160001-176001): 16000 (Start@: [2;1])
Image-Block 16 (1366400-1382400) <-> Template-Block 16 (240001-256001): 16000 (Start@: [2;1])
Image-Block 21 (1446400-1462400) <-> Template-Block 21 (320001-336001): 16000 (Start@: [2;1])
Image-Block 26 (1526400-1542400) <-> Template-Block 26 (400001-416001): 16000 (Start@: [2;1])
Image-Block 31 (1606400-1622400) <-> Template-Block 31 (480001-496001): 16000 (Start@: [2;1])
Image-Block 5 (1190400-1206400) <-> Template-Block 5 (64001-80001): 16000 (Start@: [2;1])
Image-Block 10 (1270400-1286400) <-> Template-Block 10 (144001-160001): 16000 (Start@: [2;1])
Image-Block 15 (1350400-1366400) <-> Template-Block 15 (224001-240001): 16000 (Start@: [2;1])
Image-Block 20 (1430400-1446400) <-> Template-Block 20 (304001-320001): 16000 (Start@: [2;1])
Image-Block 25 (1510400-1526400) <-> Template-Block 25 (384001-400001): 16000 (Start@: [2;1])
Image-Block 30 (1590400-1606400) <-> Template-Block 30 (464001-480001): 16000 (Start@: [2;1])
Image-Block 34 (1654400-1670400) <-> Template-Block 34 (528001-544001): 16000 (Start@: [2;1])
Image-Block 9 (1254400-1270400) <-> Template-Block 9 (128001-144001): 16000 (Start@: [2;1])
Image-Block 14 (1334400-1350400) <-> Template-Block 14 (208001-224001): 16000 (Start@: [2;1])
Image-Block 19 (1414400-1430400) <-> Template-Block 19 (288001-304001): 16000 (Start@: [2;1])
Image-Block 24 (1494400-1510400) <-> Template-Block 24 (368001-384001): 16000 (Start@: [2;1])
Image-Block 29 (1574400-1590400) <-> Template-Block 29 (448001-464001): 16000 (Start@: [2;1])
Image-Block 33 (1638400-1654400) <-> Template-Block 33 (512001-528001): 16000 (Start@: [2;1])
Image-Block 8 (1238400-1254400) <-> Template-Block 8 (112001-128001): 16000 (Start@: [2;1])
Image-Block 13 (1318400-1334400) <-> Template-Block 13 (192001-208001): 16000 (Start@: [2;1])
Image-Block 18 (1398400-1414400) <-> Template-Block 18 (272001-288001): 16000 (Start@: [2;1])
Image-Block 23 (1478400-1494400) <-> Template-Block 23 (352001-368001): 16000 (Start@: [2;1])

```

Bild 62: Bildschirmausgabe der orientierenden Suche und lokalen Alignmentberechnung in *Matlab*

In Bild 62 ist jedoch eine zusätzliche Angabe mit dem Start des maximalen lokalen Alignments im Image bzw. Template erkennbar (Start@). Die Angabe „Start@: [2; 1]“ bedeutet, das lokale Alignment startet im Image an der Position 2 und im Template an der Position 1.

Der Array *further\_hits\_percent* für weitere Imagelokalisationen, z. B. zum Auffinden von ähnlichen Dateien ist in Bild 63 dargestellt.

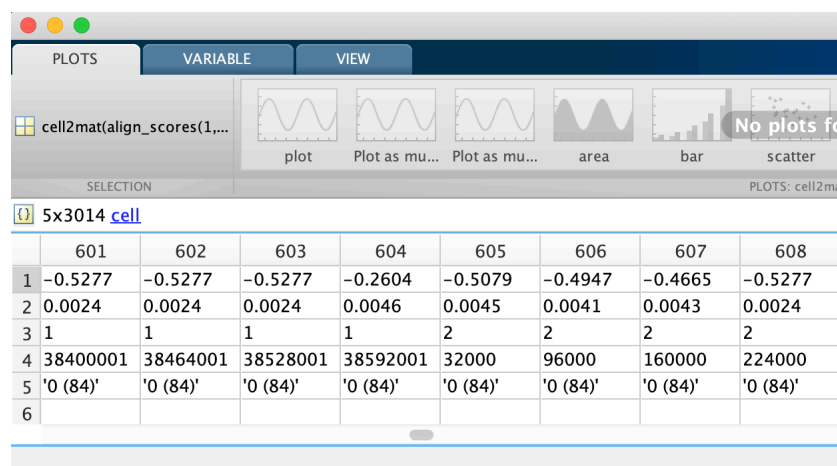


	1	2	3	4	5	6	7	8	9
1	'07-Juni-2019_17-54-16_Versions_girl_KonsensusMaster_completeLength.fa'	'0'		'91 (***46 ***)'	'31 (***49 ***)'	'48 (***49 ***)'	'60 (***31 ***)'	'91 (***47 ***)'	'64 (19)'
2									

Bild 63: Auszug aus dem Matlab-Array *further\_hits\_percent*

Die im Array *image\_templates\_compare* ab Spalte 15 angegebenen weiteren wahrscheinlichen Imagelokalisationen für das jeweilige Template werden im Array *further\_hits\_percent* als Überlappungen mit Imagedateien - wie oben bereits beschrieben - aufgelöst.

Der Array *align\_scores* bei der Imageuntersuchung allein mit globalen und lokalen Alignments ist in Bild 64 dargestellt.



	601	602	603	604	605	606	607	608
1	-0.5277	-0.5277	-0.5277	-0.2604	-0.5079	-0.4947	-0.4665	-0.5277
2	0.0024	0.0024	0.0024	0.0046	0.0045	0.0041	0.0043	0.0024
3	1	1	1	1	2	2	2	2
4	38400001	38464001	38528001	38592001	32000	96000	160000	224000
5	'0 (84)'	'0 (84)'	'0 (84)'	'0 (84)'	'0 (84)'	'0 (84)'	'0 (84)'	'0 (84)'
6								

Bild 64: Auszug aus dem Matlab-Array *align\_scores*

In Zeile eins finden sich die maximalen globalen und in Zeile zwei die maximalen lokalen Alignmentsscores. Die Angabe zum Suchdurchlauf befindet sich in Zeile drei. Die Startposition in Basen des untersuchten Imageabschnittes findet sich in Zeile vier. Eine in diesem Imageabschnitt vorhandene Überlappung mit einer Imagedatei ist in Zeile fünf angegeben. Mit Hilfe dieses Arrays sind Bild 36 und Bild 37 erstellt.

### **Hilfsprogramme**

Die großen Mengen an Daten, die bei den einzelnen Untersuchungen anfallen, können manuell nicht ausgewertet werden. Aus diesem Grunde wird der *Matlab*-Workspace mit den Berechnungsergebnissen gespeichert (*mat*-Dateien) und mit Hilfe von speziellen Analyseprogrammen ausgewertet. Neben den *Matlab*-Ergebnisdateien finden sich auf dem beiliegenden Datenträger auch *txt*- und *csv*-Dateien (z. B. Exporte aus *EnCase* und Ergebnisse mit *sdfhash*), die zur besseren Übersicht beispielsweise in Microsoft® Office Excel importiert werden können. In Tabelle 61 sind die Analyseprogramme - ebenfalls auf dem beigefügten Datenträger zu finden - aufgeführt. Diese Hilfsprogramme lassen sich grob in fünf Gruppen einteilen:

1. Analyse der Ergebnisse unveränderter Daten, einschließlich der Ähnlichkeitssuche (Nr. 1 bis 4)
2. Vorverarbeitung der Ergebnisse etablierter Verfahren zur Analyse (Nr. 5 und 6)
3. Analyse der Ergebnisse veränderter Daten - Fragmente (Nr. 7 bis 9)
4. Analyse der Größe des Untersuchungsfensters (Nr. 10)
5. Analyse der Überprüfungsergebnisse des Hauptprogramms Nr. 1 (Nr. 11)

Tabelle 61: Hilfsprogramme zur Auswertung der Untersuchungsergebnisse

Nr.	Hilfsprogramm	Beschreibung	Eingabe
1	<b>Analyze_BaseRate_Alignments_MatLabFiles.m</b>	Auswertung der Ergebnisse des Hauptprogramms <i>BaseRate Alignments.m</i> : Mittelwerte, Standardabweichungen, Vierfeldertafeln, <i>FPR</i> und <i>FNR</i> ; <u>Zusammenfassung der Ergebnisse</u> : <i>output_matrix</i>	<i>image_templates_compare</i> , <i>globalAlign_scores</i> , <i>localAlign_scores</i>

Nr.	Hilfsprogramm	Beschreibung	Eingabe
2	<b>Analyze_BaseRate_ Alignments_ MatLabFiles_ Tresholds.m</b>	Auswertung der Ergebnisse des Hauptprogramms <i>BaseRate_Alignments.m</i> : Mittelwerte, Standardabweichungen, <i>FPR</i> und <i>FNR</i> zur Ermittlung eines geeigneten Schwellenwertes; <u>Zusammenfassung der Ergebnisse</u> : <i>analyze_tresholds_&lt;normDiffSum   globalScores   localScores&gt;</i>	<i>image_templates_compare</i> , <i>globalAlign_scores</i> , <i>localAlign_scores</i>
3	<b>Analyze_BaseRate_ Alignments_ SimilarFiles.m</b>	Auswertung der Ergebnisse des Hauptprogramms <i>BaseRate_Alignments_SimilarFiles.m</i> : Mittelwerte, Standardabweichungen, Vierfeldertafeln, <i>FPR</i> und <i>FNR</i> ; <u>Zusammenfassung der Ergebnisse</u> : <i>output_matrix</i>	<i>image_locations</i> , <i>further_hits_percent</i> , <i>further_hits_norm</i> , <i>globalAlign_scores</i> , <i>localAlign_scores</i>
4	<b>Analyze_Results_ sdhash.m</b>	Verarbeitung der mit <i>sdhash</i> untersuchten unveränderten Dateien: Extrahierung des physikalischen Startbytes mit dem maximalen Score und Vergleich mit den tatsächlichen Imagelokalisationen; <u>Zusammenfassung der Ergebnisse</u> : <i>sdhash_all_results</i>	importierte Ergebnisse <i>sdhash</i> <u>und</u> <i>image_locations</i> , <i>image_templates_compare</i>
5	<b>Work_On_Results_ EnCase_ CarvedFiles.m</b>	Verarbeitung der mit <i>EnCase</i> gecarvten Dateien: Extrahierung des physikalischen Sektors des File Offsets und Einlesen der Datei zur Bestimmung der Dateilänge → Analyse dann mit den Hilfsprogrammen Nr. 7 und 8; <u>Zusammenfassung der Ergebnisse</u> : Ausgabedatei wird unter <i>results_file_name</i> benannt	Angabe des Pfades der gecarvten Dateien
6	<b>Work_On_Results_ sdhash.m</b>	Verarbeitung der mit <i>sdhash</i> untersuchten Dateien: Extrahierung des physikalischen Startbytes mit dem maximalen Score → Analyse dann mit den Hilfsprogrammen Nr. 7 und 8; <u>Zusammenfassung der Ergebnisse</u> : <i>results_</i>	importierte Ergebnisse <i>sdhash</i>

Nr.	Hilfsprogramm	Beschreibung	Eingabe
7	<b>Analyze_Results_Modified_Images.m</b>	Auswertung der mit <i>X-Ways</i> , <i>Scalpel</i> , <i>EnCase</i> und <i>sdhash</i> ermittelten Ergebnisse im Vergleich zu den bioinformatischen; <u>Zusammenfassung der Ergebnisse:</u> <i>results_fragments</i>	<i>Matlab</i> -Dateien mit den Ergebnissen der Forensiktools <u>und</u> <i>image_locations</i> , <i>image_templates</i> <i>compare</i>
8	<b>Analyze_Results_L_Fragments.m</b>	Auswertung der mit <i>X-Ways</i> , <i>Scalpel</i> , <i>EnCase</i> und <i>sdhash</i> ermittelten Ergebnisse im Vergleich zu den bioinformatischen; <u>Zusammenfassung der Ergebnisse:</u> <i>results_fragments</i>	<i>Matlab</i> -Dateien mit den Ergebnissen der Forensiktools <u>und</u> <i>image_locations</i> , <i>image_templates</i> <i>compare</i>
9	<b>Analyze_Results_I_Fragments.m</b>	Auswertung der Ergebnisse des Hauptprogramms <i>BaseRate_Alignments.m</i> für das Image I: Übereinstimmung der errechneten mit den tatsächlichen Fragmentlokalisationen; <u>Zusammenfassung der Ergebnisse:</u> Ausgabe im <i>Matlab-Command Window</i>	<i>files_template</i> , <i>image_templates</i> <i>compare</i>
10	<b>Analyze_Results_WindowSize.m</b>	Auswertung der Ergebnisse des Hauptprogramms <i>BaseRate_Alignments.m</i> für unterschiedliche Größen des Untersuchungsfensters: <i>p</i> -Werte des U-Tests nach Mann und Whitney; <u>Zusammenfassung der Ergebnisse:</u> <i>mean_std_scores</i> , <i>p_scores</i> <1T   2T   3T   4T>_<global   local>	<i>globalAlign_scores</i> , <i>localAlign_scores</i>
11	<b>Analyze_Results_TestAlgorithm.m</b>	Zusammenfassende Darstellung der Ergebnisse des Hauptprogramms <i>BaseRate_Alignments.m</i> für die Testimages; <u>Zusammenfassung der Ergebnisse:</u> <i>results_test_algorithm</i>	<i>image_templates</i> <i>compare</i>

Die Algorithmen für die Auswertungen benötigen ggf. bestimmte Arrays aus dem *Matlab*-Workspace der Untersuchungsergebnisse, diese sind ebenfalls in Tabelle 61 zu finden. Beispielsweise muss der Vergleich zwischen den mit *Scalpel* rekonstruierten Dateien und den mit bioinformatischen Methoden identifizierten Dateien auf das Array

*image\_templates\_compare* zurückgreifen, das bei der Anwendung des *Matlab*-Skriptes *BaseRate\_Alignments.m* erstellt wird. Weitere Angaben zu ggf. erforderlichen Codeanpassungen und Dateiimporten sind zu Beginn eines jeden Skriptes als Kommentarblock eingefügt.

Die genannten importierten Ergebnisse der Forensiktools müssen folgende Struktur aufweisen:

*Physikalische Lokalisation | Dateiname | Dateilänge*

Eine entsprechende *Matlab*-Datei wird für *Scalpel* beispielsweise über den Import der Datei *audit.txt* (die Dateien auf dem Datenträger wurden im Namen um das untersuchte Image und die Carving Size erweitert) in Microsoft® Office Excel und anschließendem Import in *Matlab* erzeugt. Die Erstellung von Hilfsprogrammen, die eine der oben genannten Gruppen vollständig abdecken, ist möglich, da aber die etablierten Verfahren die Analyseergebnisse teilweise unterschiedlich abspeichern bzw. zur Verfügung stellen, kann eine getrennte Programmierung die Komplexität reduzieren.

Um das Zusammenspiel zwischen den Haupt- und Hilfsprogrammen zu verdeutlichen, soll hier die orientierende Suche mit der Berechnung globaler und lokaler Alignmentsscores in den Grundzügen visualisiert werden. In einem ersten Schritt müssen die physikalischen Lokalisationen der einzelnen Imagedateien, z. B. mit *EnCase* wie in dieser Arbeit, bestimmt werden. Der *EnCase*-Export als *txt*-Datei (Name und Physical Location) dient zum Import in *Matlab* als Cell Array *image\_locations* für die Hauptprogramme Nr. 1 bis 4. Mit der anschließenden Anwendung des Hilfsprogramms Nr. 1 erfolgt die Analyse gemäß Bild 65. Nachdem durch den Vergleich von Basenhäufigkeiten (*normDiffSum*) die wahrscheinlichste Imagelokalisation für das aktuell untersuchte Template ermittelt wurde (Position 2 des Untersuchungsfensters in Bild 65), erfolgt die Berechnung des globalen und lokalen Alignmentsscores für diese Imagelokalisation. Sollte die Größe des Untersuchungsfensters für die Basenhäufigkeiten größer als die maximal verwendete Größe für die Alignments sein (64.000 Basen), so findet die Berechnung der Alignmentsscores nach Bild 19 statt. Die hierbei erzeugten Cell Arrays dienen als Eingabe für die Hilfsprogramme zur Auswertung der Untersuchungsergebnisse.



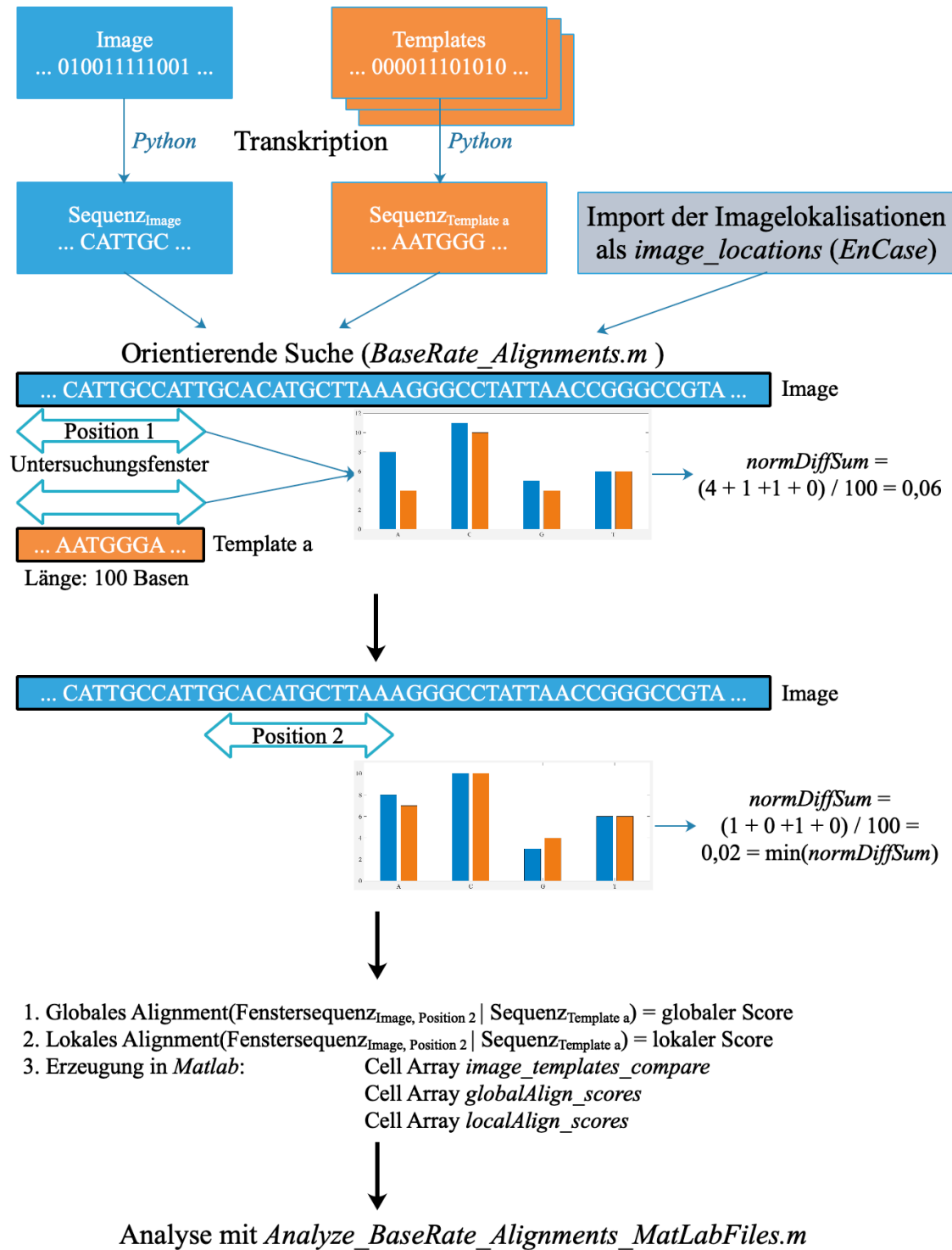


Bild 65: Ablauf der orientierenden Suche mit der Berechnung von Alignments

## Anlage E - Python-Klasse zur Erstellung einer FASTA-Datei

Die folgende *Python*-Klasse übersetzt eine Datei Byte für Byte in eine Gensequenz. Hierzu wird jedes Byte in 2-Bit-Tupel unterteilt, um anschließend in Abhängigkeit von den belegten Bits das Basenmuster zu erstellen:

Tabelle 62: Übersetzung des Bitmusters in Basen

Bitmuster	Base	Zeile im Programmcode
00	A(denin)	80 - 81
01	C(ytosin)	82 - 83
10	G(uanin)	84 - 85
11	T(hymin)	86 - 87

Das *Python*-Programm lässt sich mit jedem beliebigen Editor öffnen und bearbeiten, zu empfehlen sind die kostenlosen Entwicklungsumgebungen *Eclipse* und *Python*.

Start der *Python*-Klasse:

```

1  import os
2  import time
3  from Bio import SeqIO
4  from Bio.Seq import Seq
5  from Bio.SeqRecord import SeqRecord
6
7  #Diese Klasse übersetzt ein Bitmuster in die Nukleinsäuren 'A', 'C', 'G', 'T'.
8  class TranslateToFASTA_dna:
9
10     #Das Image-File öffnen.
11     def getFileOpen(self, path_image):
12         fimage_in = open(path_image, "rb")
13         return fimage_in
14
15     #Das Image-File schließen.
16     def setFileClose(self, fimage_in):
17         fimage_in.close()
18
19     #Die Länge des Image-File bestimmen.
20     def getFileLength(self, path_image):
21         imageLength = os.path.getsize(path_image)
22         return imageLength
23

```

```

24 #Das Bitmuster wird umgewandelt (DNA-Sequenz): 00='A', 01='C', 10='G', 11='T'.
25 def translation(self, file_in, fileLength, blocks, file_name):
26
27     #Zur Auswahl von zwei Bits in einem Byte werden hier die erforderlichen Masken
28     #definiert.
29     mask = []
30     mask.append(192) #Maske für die ersten 2 Bits (11000000)
31     mask.append(48) #Maske für Bits 3/4 (00110000)
32     mask.append(12) #Maske für Bits 5/6 (00001100)
33     mask.append(3) #Maske für die letzten 2 Bits (00000011)
34
35     #String für die DNA-Sequenz 'leer' definieren.
36     genStr = ""
37
38     #Anzahl der gem. Nutzereingabe erforderlichen Ausgabedateien initialisieren.
39     pieces = 0
40
41     #Ist die Nutzereingabe größer oder gleich der Imagelänge bzw. fehlt die
42     #Nutzereingabe wird nur eine Ausgabedatei erstellt,
43     if(blocks == 0 or blocks == fileLength or blocks > fileLength):
44         end = fileLength #Schleifenende definieren.
45         pieces = 1
46     else:
47         #ansonsten wird die Anzahl benötigter Ausgabedateien berechnet.
48         end = blocks #Schleifenende definieren.
49         pieces = int(fileLength / blocks) + 1
50
51     #Schleife zur Erstellung der Ausgabedateien.
52     for t in range(0, pieces):
53         #Die Bezeichnung der Ausgabedatei ändert sich in Abhängigkeit von der
54         #Anzahl.
55         if(pieces == 1):
56             outfile_name = "GenSeq_" + file_name + "_dna"
57         else:
58             outfile_name = "GenSeq_" + file_name + "_" + str(t) + \
59                 "_dna"
60
61         #Für den Fall, dass beim letzten Schleifendurchgang der Dateirest kleiner als
62         #die Nutzereingabe ist,
63         rest = fileLength - (t * blocks)
64         if(rest < blocks):
65             end = rest #wird das Schleifenende neu definiert.
66
67         #Schleife zum Auslesen des File.
68         for x in range(0, end):
69             file_in.seek(x) #Position im File einnehmen.
70             byte = file_in.read(1) #Byte einlesen.
71

```

```
72         #Byte in Integer umwandeln.
73         byteToInt = int.from_bytes(byte, byteorder='big')
74
75         #Schleife zur Umwandlung der 8 Bits (eingelienes Byte) in eine
76         #DNA-Sequenz.
77         for y in range(0, 4):
78             #2 Bits gem. Maske auswählen.
79             base = byteToInt & mask[y]
80             if(base == 0):
81                 genStr = genStr + "A"
82             elif(base == 1 or base == 4 or base == 16 or base == 64):
83                 genStr = genStr + "C"
84             elif(base == 2 or base == 8 or base == 32 or base == 128):
85                 genStr = genStr + "G"
86             elif(base == 3 or base == 12 or base == 48 or base == 192):
87                 genStr = genStr + "T"
88
89         sequence = Seq(genStr)
90         record = SeqRecord(sequence, id=file_name, description="Fasta-Sequenz")
91         SeqIO.write(record, path + outfile_name + "_fda", "fasta")
92         #String für die DNA-Sequenz wieder 'leer' definieren.
93         genStr = ""
94         #String für die Ausgabedatei wieder 'leer' definieren.
95         outfile_name = ""
96
97     #Programm starten.
98     if __name__ == "__main__":
99         #Variable fuer Zeitnahme initialisieren.
100         time_period = 0
101
102         #Instanz erstellen.
103         case = TranslateToFASTA_dna()
104
105         #Ordner für die Ausgabe festlegen.
106         path = <...>
107
108         #Ordner für die zu bearbeitenden Dateien festlegen.
109         root = <...>
110
111         #Inhalt des Ordners auslesen.
112         content = os.listdir(root)
113
114         #Anzahl der pro Durchgang zu verarbeitende Bytes eingeben.
115         eingabe_str_1 = "Bitte geben Sie die Sequenzlänge ein "
116         eingabe_str_2 = " ('Return', die Sequenzen werden nicht auf "
117         eingabe_str_3 = "mehrere Ausgaben verteilt): "
118         eingabe = input(eingabe_str_1 + eingabe_str_2 + eingabe_str_3)
119
```

```
120     #Bei 'leerer' Nutzereingabe erfolgt kein Splittung der Eingabedatei,
121     if(eingabe == ""):
122         blocks = 0
123     else: #sonst gibt es mehrere Ausgabedateien.
124         blocks = int(eingabe)
125
126     #Jede Datei des Eingabeordners wird gem. Nutzereingabe verarbeitet.
127     for e in range(0, len(content)):
128         #Zeitnahme starten.
129         start_time = time.time()
130
131         #Arbeitsschritt auf Bildschirm/Konsole ausgeben.
132         print((e + 1), "/", len(content), ": ", content[e])
133         #Eingabe-File öffnen.
134         file_in = case.getFileOpen(root + content[e])
135         #Länge Eingabe-File berechnen.
136         fileLength = case.getFileLength(root + content[e])
137         #Eingabe-File verarbeiten.
138         case.translation(file_in, fileLength, blocks, content[e])
139
140         #Zeitnahme stoppen.
141         end_time = time.time()
142         time_period = time_period + round(end_time - start_time, 2)
143
144         #Eingabe-File schließen.
145         case.setFileClose(file_in)
146
147     #Durchschnittliche Bearbeitungszeit pro Datei ausgeben.
148     average_time = round((time_period / len(content)), 2)
149     print("Die durchschnittliche Bearbeitungszeit pro Datei beträgt: ", \
150         average_time, " sec.")
```

Ende der Python-Klasse.

## Anlage F - *Python*-Klasse zur Erstellung von Dateifragmenten

Die folgende *Python*-Klasse erlaubt das Überschreiben von Bytes mit Nullen in einer eingeleseenen Datei. Der Nutzer kann den Beginn (*start\_image*), die Größe des zu überschreibenden Bereiches (*size\_gap*) und den Abstand zwischen diesen Bereichen (*distance\_gap*) beim Programmstart festlegen.

Das *Python*-Programm lässt sich mit jedem beliebigen Editor öffnen und bearbeiten, zu empfehlen sind die kostenlosen Entwicklungsumgebungen *Eclipse* und *Python*.

Start der *Python*-Klasse:

```
1 import os
2 import time
3
4 #Diese Klasse liest ein Image Byte für Byte ein und erzeugt ein neues Image, bei dem
5 #definierte Bereiche mit Nullen überschrieben sind.
6 class ReadImage_CreateFragments:
7
8     #Das Image-File öffnen.
9     def getFileOpen(self, path_image):
10         fimage_in = open(path_image, "rb")
11         return fimage_in
12
13     #Das Image-File schließen.
14     def setFileClose(self, fimage_in):
15         fimage_in.close()
16
17     #Die Länge des Image-File bestimmen.
18     def getFileLength(self, path_image):
19         imageLength = os.path.getsize(path_image)
20         return imageLength
21
22     #Das Eingabe-Image in ein Ausgabe-Image mit überschriebenen Bereichen
23     #umwandeln.
24     def workOnImage(self, file_in, file_length, start_image, distance_gap, \
25                     size_gap, file_name, path):
26         start = start_image
27         end = size_gap
28         overwrite = True
29         file_str = path + "Modified_" + file_name
30         file_out = open(file_str, 'bw')
31         #Schleife zum Auslesen des Files.
32         while end <= file_length:
33             if overwrite:
34                 for x in range(start, end):
```

```
35         file_out.write(b'\x00')
36         overwrite = False
37         start = end
38         if (start + distance_gap) > file_length:
39             end = file_length
40         else:
41             end = start + distance_gap
42     else:
43         file_in.seek(start) #Position im File einnehmen.
44         file_bytes = file_in.read(end - start) #Byte einlesen.
45         file_out.write(file_bytes)
46         overwrite = True
47         start = end
48         end = start + size_gap
49     case.setFileClose(file_out)
50
51     #Programm starten.
52     if __name__ == "__main__":
53         #Variable für die Zeitnahme initialisieren.
54         time_period = 0
55
56         #Instanz erstellen.
57         case = ReadImage_CreateFragments()
58
59         #Ordner für die Ausgabe festlegen.
60         path = <...>
61
62         #Inhalt des Ordners auslesen.
63         content = os.listdir(path)
64
65         #Anzahl der pro Durchgang zu verarbeitende Bytes eingeben.
66         start_image = input("Start des Überschreibens (Lücken) im Image: ")
67         distance_gap = input("Abstand der Lücken: ")
68         size_gap = input("Größe der Lücken: ")
69
70         #Bei 'leerer' Nutzereingabe wird der Start mit Null festgelegt.
71         if(start_image == ""):
72             start_image = 0
73         else:
74             start_image = int(start_image)
75
76         #Bei 'leerer' Nutzereingabe wird der Abstand mit Null festgelegt.
77         if(distance_gap == ""):
78             distance_gap = 0
79         else:
80             distance_gap = int(distance_gap)
81
82
```

```
83     #Bei 'leerer' Nutzereingabe wird die Größe mit Null festgelegt.
84     if(size_gap == ""):
85         size_gap = 0
86     else:
87         size_gap = int(size_gap)
88
89     #Jede Datei des Eingabeordners wird gem. Nutzereingabe verarbeitet.
90     for e in range(0, len(content)):
91         #Zeitnahme starten.
92         start_time = time.time()
93
94         #Arbeitsschritt auf Bildschirm/Konsole ausgeben.
95         print((e + 1), "/", len(content), ": ", content[e])
96         #Eingabe-File öffnen.
97         file_in = case.getFileOpen(path + content[e])
98         #Länge Eingabe-File berechnen.
99         file_length = case.getFileLength(path + content[e])
100        #Eingabe-File verarbeiten.
101        case.workOnImage(file_in, file_length, start_image, distance_gap, size_gap, \
102                        content[e], path)
103
104        #Zeitnahme stoppen.
105        end_time = time.time()
106        time_period = time_period + round(end_time - start_time, 2)
107
108        #Eingabe-File schließen.
109        case.setFileClose(file_in)
110
111        #Durchschnittliche Bearbeitungszeit pro Datei ausgeben.
112        average_time = round((time_period / len(content)), 2)
113        print("Die durchschnittliche Bearbeitungszeit pro Datei beträgt: ", average_time, \
114            " sec.")
```

Ende der Python-Klasse.



## Anlage G - *Matlab*-Skript zur Suche mit Hilfe von Basenhäufigkeiten und Alignments

Das folgende *Matlab*-Skript errechnet die Basenhäufigkeiten im Template und vergleicht diese mit jedem Imageabschnitt. Die Lokalisation im Image mit der geringsten Abweichung zwischen den Häufigkeiten kann anschließend zur Berechnung der globalen und lokalen Alignment scores verwendet werden. Es sind hier nur die wesentlichen Codeanteile aufgeführt (Auslassungen sind durch [...] gekennzeichnet), der vollständige Skriptcode ist auf dem beigefügten Datenträger in der Datei *BaseRate\_Alignments.m* zu finden.

Start des *Matlab*-Skriptes:

```

1  %*****
2  % Dieses Skript ermittelt die Haeufigkeiten einzelner oder mehrerer Nukleinbasen bzw.
3  % Basen-Tupel in Templates fuer den Vergleich mit Images.
4  % Nachdem so die optimale Position im Image gefunden wurde, erfolgt die Alignierung
5  % zwischen Template und Image.
6  %-----Nur die erste Lokalisation (kleinste normDiffSum) wird aligniert-----
7  %
8  % Eingabe: importierte Imagelokalisationen als 'Image_locations', z.B. aus EnCase
9  % exportierte Datei
10 % -> 'path_images' anpassen
11 % -> 'path_templates' anpassen
12 % -> 'file_root' anpassen
13 %
14 % Ausgabe: 'image_templates_compare'
15 %*****
16
17 %-----
18 %  Vorbereitende Massnahmen
19 %-----
20 % Variablen intialisieren.
21 path_images = <...>;
22 path_templates = <...>;
23 file_root = <...>;
24 memory_limit_align = 64000; % Dieser Wert ist von der RAM-Groesse, Anzahl der
25                             % Prozessoren und u. a. dem Swap-Bereich abhaengig
26                             % und sollte ggf. im jeweiligen System getestet
27                             % werden.
28 % Scoring Matrices initialisieren.
29 scoring_matrix_global = [1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1];
30 scoring_matrix_local = [1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1];
31

```

```
32 % Bewertung von Luecken initialisieren.
33 gap_open_global = 1;
34 gap_open_local = 1;
35 extend_gap_global = 1;
36 extend_gap_local = 1;
37 % Beim Skriptstart Nutzer-Fragen stellen.
38 questions = true;
39 % Auflistung der Image-Dateien aus dem Pfad erstellen (hier immer nur eine Datei).
40 files_image = dir(path_images);
41 % Auflistung der Template-Dateien aus dem Pfad erstellen.
42 files_template = dir(path_templates);
43
44 [...]
45
46 % Mit Hilfe dieser Schleife werden die Untersuchungen mit den unterschiedlichen
47 % Tupellaengen durchgefuehrt.
48 for aa = 1 : 4
49     base = {'A', 'C', 'G', 'T'};
50     tupel_str = {'A', 'C', 'G', 'T'};
51     compare_actual = 0;
52     location = 1;
53     next = true;
54
55     % Die Laenge der Basentupel fuer die Haeufigkeitsuntersuchung festlegen.
56     base_str_length = aa;
57     if base_str_length > 1
58         base_str_help = cell(length(base) * length(tupel_str));
59
60         % Basentupel in Array speichern.
61         while next
62             ende_base_str = length(tupel_str);
63             ende_base = length(base);
64
65             % Schleifen zur Kombination/Konkatenation von Basen-Strings.
66             for g = 1 : ende_base_str
67                 for h = 1 : 4
68                     base_str_help(1, location) = strcat(tupel_str(1, g), base(1, h));
69                     location = location + 1;
70                 end
71             end
72
73             tupel_str = base_str_help(1, :);
74             base_str_help = cell(length(base) * length(tupel_str));
75
76             % Abbruchbedingung fuer Uebergeordnete while-Schleife festlegen.
77             if location == 4^base_str_length + 1
78                 next = false;
79             end
```

```

80
81     location = 1;
82     end
83 end
84
85 [...]
86
87     % Der Nutzer kann die Parameter "Untersuchungsfenster", "Shift",
88     % "Image-Laenge", "Schwellenwert" und „Alignments“ fuer jede Tupellaenge
89     % neu festlegen.
90     if questions
91         [new_window, shift_default, new_image_length, threshold, questions, ...
92          align_decision, factor_] = user_input(questions);
93     end
94     file_name = [file_root, num2str(aa), 'T'];
95     % Diese Schleife liest nacheinander jedes Template fuer die
96     % Untersuchung ein.
97     for y = 1 : array_templates_width
98
99         % Template einlesen und Laenge bestimmen.
100        [template, template_length] = file_input(path_templates, ...
101         compare_files_template{1, y});
102
103    [...]
104
105    % Start und Ende fuer die Untersuchung im Image initialisieren.
106    if new_image_length > 0
107        seq_end_image = new_image_length;
108        image_length = new_image_length;
109    else
110        seq_end_image = image_length;
111    end
112
113    % Laenge des Untersuchungsfensters definieren, Nutzereingabe oder
114    % Template-Laenge.
115    if new_window > 0
116        window = new_window;
117    elseif isequal(new_window, 0) && isequal(factor_, 0)
118        window = template_length;
119    elseif factor_ > 0
120        window = fix(template_length / factor_);
121    end
122
123    % Anzahl der Untersuchungsblöcke im Image berechnen.
124    if mod((image_length - shift_default), window) == 0
125        image_pieces = (image_length - shift_default) / window;
126    else
127        image_pieces = fix((image_length - shift_default) / window) + 1;

```

```

128     end
129
130     % Anzahl der Untersuchungsblöcke im Template berechnen.
131     if mod(template_length, window) == 0
132         template_pieces = template_length / window;
133     else
134         template_pieces = fix(template_length / window) + 1;
135     end
136
137     [...]
138
139     % Die Start- und Endpunkte fuer die Untersuchungsfenster im Template
140     % festlegen.
141     [start_points_template, end_points_template] = ...
142         start_end_points_template(1, template_length, window);
143
144     % Auf Basis der einzelnen Untersuchungsfenster die entsprechenden
145     % Sequenzen aus dem Template gesondert abspeichern, um den Workload
146     % fuer die parallele Verarbeitung zu reduzieren.
147     template_sequences = sequences_template(template, start_points_template, ...
148         end_points_template);
149
150     % Die Tupel-Häufigkeiten fuer die einzelnen Template-Sequenzen □
151     % berechnen.
152     tupel_rates_template = base_rates_template(template_sequences, tupel_str, ...
153         base_str_length);
154
155     [...]
156
157     % -----
158     %           Untersuchung der Häufigkeiten
159     % -----
160     % Diese erste Schleife fuer mehrere Untersuchungen durch, bis
161     % die kleinste mittlere Abweichung der Tupel-Häufigkeiten
162     % zwischen Image und Template gefunden wird.
163     while shift_steps
164
165         % Die Start- und Endpunkte fuer die Untersuchungsfenster im
166         % Image festlegen.
167         [start_points_image, end_points_image] = ...
168             start_end_points_image(seq_start_image, seq_end_image, window);
169
170         % Auf Basis der einzelnen Untersuchungsfenster die
171         % entsprechenden Sequenzen aus dem Image gesondert
172         % abspeichern, um den Workload fuer die parallele
173         % Verarbeitung zu reduzieren.
174         image_sequences = sequences_image(image, start_points_image, ...
175             end_points_image);

```

```

176
177 % Die Tupel-Haeufigkeiten fuer die einzelnen Image-Sequenzen ●
178 % berechnen.
179 tuple_rates_image = base_rates_image(image_sequences, tuple_str, ...
180     base_str_length);
181
182 [...]
183
184 % Aufruf der einzelnen Image-Blocke zur Analyse. Das Image muss
185 % auf Grund des Untersuchungsfensters (Standard: Template-Laenge)
186 % in Blocke unterteilt werden.
187 for m = 1 : image_pieces
188
189     % Stoppuhr 2 fuer den Vergleich mit einem Image-Block starten.
190     start_time_2 = tic;
191
192     % Aufruf der einzelnen Template-Blocke zum Vergleich mit den
193     % Image-Blocken. Das Template muss auf Grund des
194     % Untersuchungsfensters (Standard: Template-Laenge) in Blocke
195     % unterteilt werden.
196     for f = 1 : template_pieces
197
198         % Summe der Betraege der Differenzen zwischen ★
199         % den Haeufigkeiten.
200         sum_ = sum(abs(cell2mat(tuple_rates_image(m, :)) - ...
201             cell2mat(template_rates(f, :))));
202
203         % Differenzsumme auf Templatelaenge normieren und als
204         % Vektor ablegen.
205         norm_ = sum_ / window;
206
207         % Kleinste Summe mit dem zugehoerigen Image-Block
208         % ermitteln.
209         if sum_ < min_sum_image
210             min_sum_image = sum_;
211             min_sum_block_image = m;
212         end
213
214         % Kleinsten Mittelwert mit dem zugehoerigen
215         % Image-Block ermitteln.
216         if norm_ < min_norm_image
217             min_norm_image = norm_;
218             min_norm_block_image_start = start_points_image(1, m);
219             min_norm_block_image_end = end_points_image(1, m);
220             min_norm_block_image_step = steps;
221             min_norm_block_template_start = start_points_template(1, f);
222             min_norm_block_template_end = end_points_template(1, f);
223         end

```

```

224
225     % Stoppuhr 2 anhalten.
226     stop_time_2 = toc(start_time_2);
227
228     % Zaehlvariabel um 1 erhoehen.
229     counter = counter + 1;
230     end
231 end
232
233 % Variable neu initialisieren.
234 steps = steps + 1;
235
236 % Fuer den naechsten Image-Durchlauf den Start des
237 % Untersuchungsfenster neu festlegen.
238 if min_norm_image < global_min_norm_image
239     global_min_norm_image = min_norm_image;
240     shift = fix(shift + window / steps);
241     if shift < image_length
242         seq_start_image = shift;
243     else
244         shift_steps = false;
245     end
246 else
247     shift_steps = false;
248 end
249
250 % Anzahl der Untersuchungsblöcke im Image neu berechnen.
251 if mod((image_length - shift), window) == 0
252     image_pieces = (image_length - shift) / window;
253 else
254     image_pieces = fix((image_length - shift) / window) + 1;
255 end
256 end
257
258 % Den Shift fuer den naechsten Templatevergleich zuruecksetzen.
259 shift = shift_default;
260
261 [...]
262
263 % -----
264 %           Globales Alignment berechnen
265 % -----
266 % Funktion fuer die Berechnung globaler Alignments aufrufen.
267 [globAlignment_scores] = ...
268     globalAlignment(template_sequences_align, ...
269     image_sequences_align, ...
270     start_points_image_align, end_points_image_align, ...

```

▽

❖

```

271     start_points_template_align, end_points_template_align,
272     exam_window_global, ...
273     scoring_matrix_global, gap_open_global, extend_gap_global);
274
275     globalAlign_scores(y, 3) = num2cell(sum(globAlignment_scores) / window);
276
277     [...]
278
279     image_templates_compare(y, 5) = ...
280     num2cell(max_score_global / exam_window_align);
281
282     [...]
283
284     % -----
285     %           Lokale Alignments berechnen ⌘
286     % -----
287     % Funktion fuer die Berechnung lokaler Alignments aufrufen.
288     [locAlignment_scores, locAlignment_starts] = ...
289     localAlignment(template_sequences_align, ...
290     image_sequences_align, ...
291     start_points_image_align, end_points_image_align, ...
292     start_points_template_align, end_points_template_align, ...
293     scoring_matrix_local, gap_open_local, extend_gap_local);
294
295     localAlign_scores(y, 3) = num2cell(sum(locAlignment_scores) / window);
296
297     [...]
298
299     image_templates_compare(y, 6) = ...
300     num2cell(max_score_local / exam_window_align);
301     end
302 end
303
304     % Workspace in einer externen Datei abspeichern.
305     save(file_name);
306 end
307
308 % *****
309 % *****
310 %           FUNKTIONEN
311 % *****
312 % *****
313
314 % -----
315 % Diese Funktion berechnet die Tupel-Haeufigkeiten in Template-Sequenzen. □
316 % -----
317 function tuple_rates_template = ...
318     base_rates_template(template_sequences, tuple_str, base_str_length)

```

```

319
320 % Matrix fuer die Tupel-Haeufigkeiten (Spalten) in Template-Sequenzen (Zeilen)
321 % initialisieren.
322 tuple_rates_template = cell(length(template_sequences(1, :)), length(tuple_str(1, :)));
323 for r = 1 : length(tuple_rates_template(1, :))
324     for s = 1 : length(tuple_rates_template(:, 1))
325         tuple_rates_template(s, r) = num2cell(0);
326     end
327 end
328
329 % Die Tupel-Haeufigkeiten bestimmen und in der Matrix ablegen.
330 length_tuple_str = length(tuple_str(1, :));
331 parfor p = 1 : length(template_sequences(1, :))
332     tuple_count = nmercount(template_sequences{1, p}, base_str_length);
333     for o = 1 : length(tuple_count(:, 1))
334         for q = 1 : length_tuple_str
335             if tuple_count{o, 1} == tuple_str{1, q}
336                 tuple_rates_template(p, q) = tuple_count(o, 2);
337             end
338         end
339     end
340 end
341 clearvars tuple_count;
342 end
343
344 % -----
345 % Diese Funktion berechnet die Tupel-Haeufigkeiten in Image-Sequenzen. ●
346 % -----
347 function tuple_rates_image = ...
348     base_rates_image(image_sequences, tuple_str, base_str_length)
349
350 % Matrix fuer die Tupel-Haeufigkeiten (Spalten) in Image-Sequenzen (Zeilen)
351 % initialisieren.
352 tuple_rates_image = cell(length(image_sequences(1, :)), length(tuple_str(1, :)));
353 for r = 1 : length(tuple_rates_image(1, :))
354     for s = 1 : length(tuple_rates_image(:, 1))
355         tuple_rates_image(s, r) = num2cell(0);
356     end
357 end
358
359 % Die Tupel-Haeufigkeiten bestimmen und in der Matrix ablegen.
360 length_tuple_str = length(tuple_str(1, :));
361 parfor p = 1 : length(image_sequences(1, :))
362     tuple_count = nmercount(image_sequences{1, p}, base_str_length);
363     for o = 1 : length(tuple_count(:, 1))
364         for q = 1 : length_tuple_str
365             if tuple_count{o, 1} == tuple_str{1, q}
366                 tuple_rates_image(p, q) = tuple_count(o, 2);

```



```

367         end
368     end
369 end
370 end
371 clearvars tuple_count;
372 end
373
374 [...]
375
376 % -----
377 % Diese Funktion berechnet das globale Alignment fuer den Imagebereich
378 % mit dem kleinsten Mittelwert der Tupel-Haeufigkeit. ❖
379 % -----
380 function [globAlignment_scores] = ...
381     globalAlignment(template_sequences_align, image_sequences_align, ...
382         start_points_image_align, end_points_image_align, ...
383         start_points_template_align, end_points_template_align, exam_window_global, ...
384         scoring_matrix_global, gap_open_global, extend_gap_global)
385
386 [...]
387
388 % Schleife zur Berechnung der Scores der globalen Alignments.
389 parfor i = 1 : count_sequences_gloAlign
390     [score, ~, ~] = nwalign(image_sequences_align{i}, ...
391         template_sequences_align{i}, 'ScoringMatrix', ...
392         scoring_matrix_global, 'extendgap', extend_gap_global, 'gapopen',
393         gap_open_global, 'alphabet', 'nt');
394     globAlignment_scores(i) = score;
395
396 [...]
397
398 end
399
400 % -----
401 % Diese Funktion berechnet das lokale Alignment fuer den Imagebereich
402 % mit dem kleinsten Mittelwert der Tupel-Haeufigkeit. ⌘
403 % -----
404 function [locAlignment_scores, locAlignment_starts] = ...
405     localAlignment(template_sequences_align, image_sequences_align, ...
406         start_points_image_align, end_points_image_align, ...
407         start_points_template_align, end_points_template_align, ...
408         scoring_matrix_local, gap_open_local, extend_gap_local)
409
410 [...]
411
412 % Schleife zur Berechnung der Scores der lokalen Alignments.
413 parfor i = 1 : count_sequences_locAlign
414     [score, ~, startat] = swalign(image_sequences_align{1, i}, ...

```

```

415     template_sequences_align{1, i}, 'ScoringMatrix', ...
416     scoring_matrix_local, 'extendgap', extend_gap_local, 'gapopen', gap_open_local, ...
417     'alphabet', 'nt');
418     locAlignment_scores(i) = score;
419     locAlignment_starts(i) = cellstr(mat2str(startat));
420
421     [...]
422
423 end
424
425 % -----
426 % Diese Funktion liest die Dateien ein und bestimmt die File-Laenge anhand
427 % der Basenanzahl.
428 % -----
429 function [file, file_length] = file_input(path, compare_files)
430
431 % Fasta-File einlesen.
432 file_in = [path, compare_files];
433 file = fastaread(file_in);
434
435 % File-Laenge anhand der Basen-Anzahlen bestimmen.
436 counts_bases = basecount(file);
437 file_length = counts_bases.A + counts_bases.C + counts_bases.T + counts_bases.G;
438 end
439
440 % -----
441 % Diese Funktion liest Nutzereingaben fuer die Analyse von Image und
442 % Template ein.
443 % -----
444 function [new_window, shift_default, new_image_length, treshold, ...
445     questions, align_decision, factor_] = user_input(questions)
446
447     [...]
448
449 end

```



Ende des *Matlab*-Skriptes.

## Anlage H - *Matlab*-Skript zur Suche auf Basis von Alignments

Das folgende *Matlab*-Skript führt die Suche nach Templates ausschließlich auf Basis von globalen und lokalen Alignments durch. Es basiert auf dem Skript in Anlage G - *Matlab*-Skript zur Suche mit Hilfe von Basenhäufigkeiten, daher sind hier nur die wesentlichen, abgewandelten Codeanteile aufgeführt (Auslassungen sind durch [...] gekennzeichnet), der vollständige Skriptcode ist auf dem beigelegten Datenträger in der Datei *Alignments.m* zu finden.

Start des *Matlab*-Skriptes:

```
1  %*****
2  % Dieses Skript ermittelt die Alignmentsscores fuer jede Position des
3  % Untersuchungsfensters beim Abfahren des Images.
4  %
5  % -> 'path_images' anpassen
6  % -> 'path_templates' anpassen
7  % -> 'file_name' anpassen
8  %
9  % Ausgabe: 'image_templates_compare'
10 %*****
11
12 %-----
13 %  Vorbereitende Massnahmen
14 %-----
15 % Variablen initialisieren.
16 path_images = <...>;
17 path_templates = <...>;
18 file_name = <...>;
19 memory_limit_align = 64000;  % Dieser Wert ist von der RAM-Groesse, Anzahl der
20                               % Prozessoren und u. a. dem Swap-Bereich abhaengig
21                               % und sollte ggf. im jeweiligen System getestet
22                               % werden.
23 % Scoring Matrices initialisieren.
24 scoring_matrix_global = [1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1];
25 scoring_matrix_local = [1 -1 -1 -1; -1 1 -1 -1; -1 -1 1 -1; -1 -1 -1 1];
26 % Bewertung von Luecken initialisieren.
27 gap_open_global = 1;
28 gap_open_local = 1;
29 extend_gap_global = 1;
30 extend_gap_local = 1;
31
32 [...]
33
```


```
34 %-----
35 %           Globales Alignment berechnen 
36 %-----
37 % Funktion fuer die Berechnung globaler Alignments aufrufen.
38 [globAlignment_scores] = ...
39     globalAlignment(template_sequences, image_sequences, ...
40     start_points_image, end_points_image, start_points_template, ...
41     end_points_template, image_pieces, template_pieces, window, ...
42     scoring_matrix_global, gap_open_global, extend_gap_global);
43
44 % Den maximalen globalen Score ermitteln. Die Anzahl der
45 % Tabellenspalten gibt die Anzahl der Imageabschnitte wieder. Die
46 % Anzahl der Zeilen entspricht den Templateabschnitten.
47 [~, I] = max(globAlignment_scores(:));
48 [global_row, global_col] = ind2sub(size(globAlignment_scores), I);
49 if globAlignment_scores(global_row, global_col) > max_score_global
50     max_score_global = globAlignment_scores(global_row, global_col);
51     max_score_start_global_image = ...
52         fix(start_points_image(1, global_col) / 4);
53 end
54
55 % Den maximalen globalen Score abspeichern.
56 image_templates_compare(y, 5) = num2cell(max_score_global);
57
58 %-----
59 %           Lokale Alignments berechnen 
60 %-----
61 % Funktion fuer die Berechnung lokaler Alignments aufrufen.
62 [locAlignment_scores, locAlignment_starts] = ...
63     localAlignment(template_sequences, image_sequences, ...
64     start_points_image, end_points_image, ...
65     start_points_template, end_points_template, ...
66     image_pieces, template_pieces, window, ...
67     scoring_matrix_local, gap_open_local, extend_gap_local);
68
69 % Den maximalen lokalen Score ermitteln. Die Anzahl der
70 % Tabellenspalten gibt die Anzahl der Imageabschnitte wieder. Die
71 % Anzahl der Zeilen entspricht den Templateabschnitten.
72 [M, I] = max(locAlignment_scores(:));
73 [local_row, local_col] = ind2sub(size(locAlignment_scores), I);
74 if locAlignment_scores(local_row, local_col) > max_score_local
75     max_score_local = locAlignment_scores(local_row, local_col);
76     max_score_start_local_image = fix(start_points_image(1, local_col) / 4);
77 end
78
79 % Den maximalen lokalen Score abspeichern.
80 image_templates_compare(y, 6) = num2cell(max_score_local);
81
```

```

82      % Variable neu initialisieren.
83      steps = steps + 1;
84
85      % Fuer den naechsten Image-Durchlauf den Start des
86      % Untersuchungsfensters neu festlegen. ▽
87      if (max_score_global > global_max_globScore_image) || ...
88          (max_score_local > global_max_locScore_image)
89          if max_score_global > global_max_globScore_image
90              global_max_globScore_image = max_score_global;
91          elseif max_score_local > global_max_locScore_image
92              global_max_locScore_image = max_score_local;
93          end
94          shift = fix(shift + window / steps);
95          if shift < image_length
96              seq_start_image = shift;
97          else
98              shift_steps = false;
99          end
100      else
101          shift_steps = false;
102      end
103
104      [...]
105
106      end
107      % Den Workspace extern in einer Datei abspeichern.
108      save(file_name);
109
110      %*****
111      %*****
112      %              FUNKTIONEN
113      %*****
114      %*****
115
116      [...]
117
118      %-----
119      % Diese Funktion berechnet das globale Alignment. ◆
120      %-----
121      function [globAlignment_scores] = ...
122          globalAlignment(template_sequences, image_sequences, ...
123              start_points_image, end_points_image, ...
124              start_points_template, end_points_template, image_pieces, ...
125              template_pieces, window, scoring_matrix_global, gap_open_global, ...
126              extend_gap_global)
127
128      [...]

```

```

129
130 % Schleife zur Berechnung der Scores der globalen Alignments.
131 parfor i = 1 : image_pieces
132     for j = 1 : template_pieces
133         [score, ~, ~] = nwalgn(image_sequences{i}, ...
134             template_sequences{j}, 'ScoringMatrix', ...
135             scoring_matrix_global, 'extendgap', extend_gap_global, ...
136             'gapopen', gap_open_global, 'alphabet', 'nt');
137         globAlignment_scores(j, i) = score / window;
138
139     [...]
140
141 end
142
143 %-----
144 % Diese Funktion berechnet das lokale Alignment. 
145 %-----
146 function [locAlignment_scores, locAlignment_starts] = ...
147     localAlignment(template_sequences, image_sequences, ...
148     start_points_image, end_points_image, ...
149     start_points_template, end_points_template, image_pieces, ...
150     template_pieces, window, scoring_matrix_local, gap_open_local, ...
151     extend_gap_local)
152
153 [...]
154
155 % Schleife zur Berechnung der Scores der lokalen Alignments.
156 parfor i = 1 : image_pieces
157     for j = 1 : template_pieces
158         [score, ~, startat] = swalign(image_sequences{i}, ...
159             template_sequences{j}, 'ScoringMatrix', ...
160             scoring_matrix_local, 'extendgap', extend_gap_local, ...
161             'gapopen', gap_open_local, 'alphabet', 'nt');
162         locAlignment_scores(j, i) = score / window;
163         locAlignment_starts(j, i) = cellstr(mat2str(startat));
164
165     [...]
166
167 end
168
169 [...]

```

Ende des *Matlab*-Skriptes.

## Anlage I - Bilder zur Erzeugung von Konsensussequenzen und Bildversionen

Alle hier genutzten Bilder sind frei verfügbar und dürfen verwendet werden (Abschnitt 6.2). Die in Bild 66, Bild 67 und Bild 68 gezeigten Bilder werden in ihrer Größe angepasst, um annähernd gleich große Dateien zu erhalten. Der Einsatz Multipler Alignments in Form des *Matlab*-Algorithmus *multialign()* setzt in etwa gleich lange Sequenzen voraus. Die zur Erzeugung der ersten Konsensussequenz verwendeten 10 Frauengesichter sind in Bild 66 aufgeführt.



Bild 66: Frauengesichter zur Erstellung einer Konsensussequenz

Die entsprechenden 10 Männergesichter sind in Bild 67 aufgeführt (zweite Konsensussequenz).



Bild 67: Männergesichter zur Erstellung einer Konsensussequenz



Abschließend werden gezielt Frauengesichter gesucht, die folgende Kriterien erfüllen:

- Gesicht frontal fotografiert
- keine Verdeckungen, z. B. durch Brillen oder Hüte
- möglichst homogener Hintergrund
- nur Farbfotografien
- möglichst wenig Schminke
- möglichst homogene Bildausleuchtung
- das gesamte Gesicht sollte sichtbar sein

Die auf Basis dieser Kriterien zur Erstellung der dritten Konsensussequenz (Faces) eingesetzten Fotos sind in Bild 68 zusammengestellt.



Bild 68: Kriterienbasierte Gesichterauswahl zur Erstellung einer Konsensussequenz

Die Fotos werden bei Bedarf auf den Gesichtsabschnitt zugeschnitten und in der Größe verändert, um annähernd gleiche Dateigrößen zu erhalten. Gemäß Bild 31 werden zur Untersuchung des Images „**Similar Files**“ mit den drei oben beschriebenen Konsensussequenzen nur vollständige Blöcke verwendet. Hierbei bedingt die Größenangleichung der Bilder, dass bei allen Sequenzen annähernd die gleiche Anzahl Basen nicht in die Konsensussequenz einfließt.



Zur Erzeugung unterschiedlicher „Bildversionen“ und Untersuchung des Images „**Versions**“ werden die Templates in Bild 69, Bild 70 und Bild 71 genutzt. Die Originalbilder im *jpg*-Format dienen als Ausgangsbasis für die Bearbeitung mit der Software *GIMP*, wie folgt (vgl. [41]):

1. Reduzierung der Bildgröße auf 75 % (Skalieren) der ursprünglichen Größe.
2. Erhöhung der Bildgröße auf 125 % (Skalieren) der ursprünglichen Größe.
3. Weichzeichnen des Bildes mit dem Standard-Filter „Gaußscher Weichzeichner“.
4. Verbessern des Bildes mit dem Standard-Filter „NL-Filter“.
5. Künstlerische Verarbeitung des Bildes mit dem Standard-Filter „Ölgemälde“.

Die unterschiedlichen „Bildversionen“ werden im *jpg*-Format exportiert. Für alle *GIMP*-Filter finden keine Veränderungen der Voreinstellungen statt.

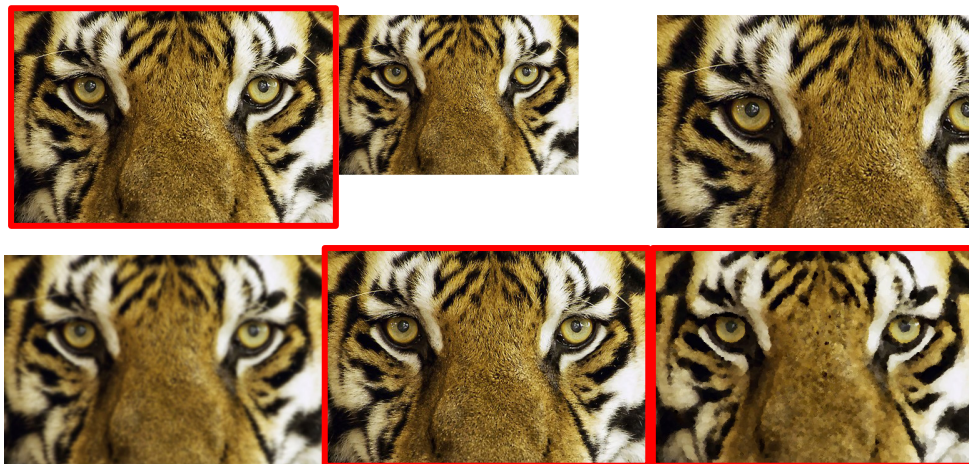


Bild 69: „Versionen“ des Templates 1, von links oben nach rechts unten: original-1.-2.-3.-4.-5.

Auf Basis dieser drei Templates werden zusätzlich aus dem Originalbild und den mit *GIMP* nach Nr. 4. und 5. veränderten Bildern (rote Umrandung) drei Konsensussequenzen erzeugt. Die Konsensussequenzen werden zum einen über die kleinste zur Erzeugung genutzten Templatesequenz gebildet, so dass die Summe der Untersuchungsabschnitte der Sequenzlänge

$$\max(n * \text{Fenstergröße}) \leq \text{Länge der kleinsten Templatesequenz} \quad (I.1)$$

$$n \in \mathbb{N}^{>0}$$

entspricht (Bild 31, Konsensus 1 bis 3).

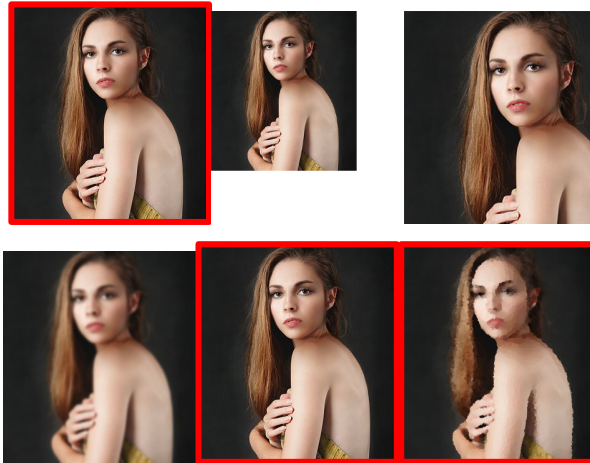


Bild 70: „Versionen“ des Templates 2, von links oben nach rechts unten: original-1.-2.-3.-4.-5.

Zum anderen wird die Länge der kleinsten Templatesequenz vollständig zur Konsensusbildung genutzt.

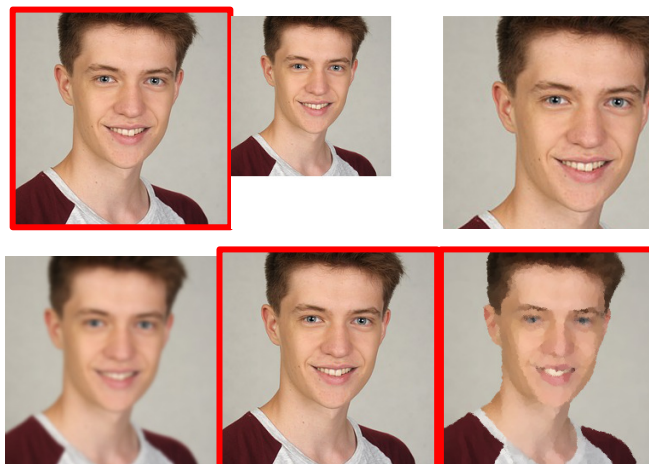


Bild 71: „Versionen“ des Templates 3, von links oben nach rechts unten: original-1.-2.-3.-4.-5.

## Anlage J - Datenträgerinhalte

Bei dem beiliegenden Datenträger handelt es sich um eine Blu-ray Disc (sogenannte Millenium Disc), auf der alle genutzten bzw. im Rahmen der Arbeit erzeugten Daten gespeichert sind. Die erstellten *Matlab*-Skripte zur Untersuchung und Analyse sind ebenfalls abgelegt, die eingesetzte Software gemäß Anlage A ist nicht enthalten. Der Strukturbaum der Dateiablage ergibt sich bis in die dritte Ebene wie in Bild 72 gezeigt.

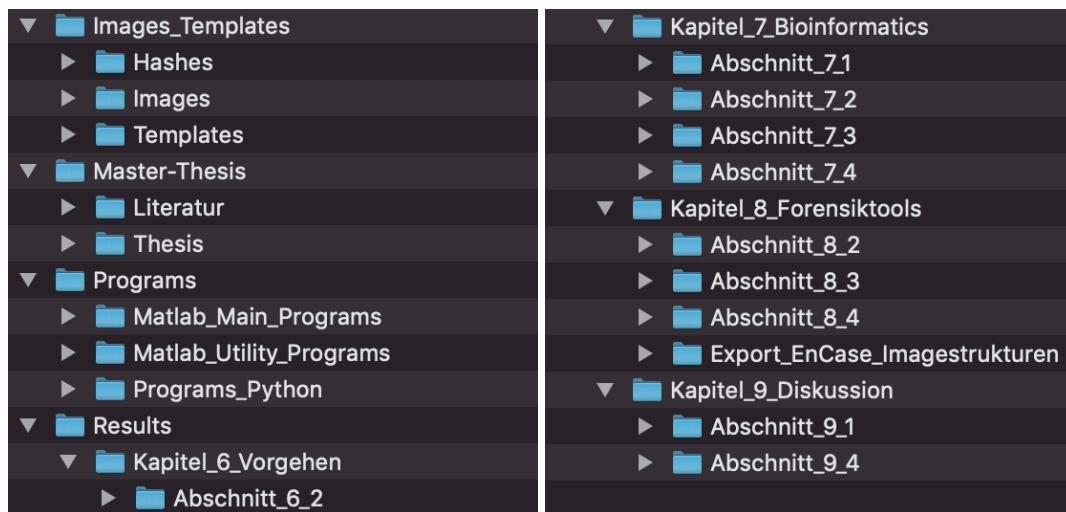


Bild 72: Strukturbaum der Dateiablage auf dem Datenträger

Neben den *Matlab*-Ergebnisdateien finden sich auf dem Datenträger auch *txt*- und *csv*-Dateien, die zur besseren Übersicht beispielsweise in *Microsoft® Office Excel* importiert werden können.

Die im Rahmen der Untersuchungen gecarvten Dateien befinden sich nicht auf dem Datenträger, da sie zusätzliche Datenträger erforderlich machen würden; jedoch sind nur Auflistung der Dateinamen, der Imagelokalisationen und Größen der gecarvten Dateien für die Analysen von Bedeutung. Diese Auflistungen sind im Ordner „Abschnitt\_8\_3“ auf dem Datenträger abgelegt.

Die Inhalte der wesentlichen *Matlab*-Ergebnisarrays sind für jeden Untersuchungsabschnitt zusätzlich als Kopie im *Excel*-Format abgelegt.

## **Erklärung**

Hiermit erkläre ich, dass ich die hier vorliegende Arbeit selbstständig, ohne unerlaubte fremde Hilfe und nur unter Verwendung der in der Arbeit aufgeführten Hilfsmittel angefertigt habe.

Ahrweiler, den 6. Juli 2019

**Thesen**

1. Die Analyse von Basenhäufigkeiten ist geeignet, unveränderte Dateien auf Basis bekannter Muster (Templates) zu identifizieren (orientierende Suche).
2. Das Suchergebnis verbessert sich mit zunehmender Tupellänge (die hier maximal untersuchte Tupellänge beträgt vier {AAAA, ..., TTTT}).
3. Mit Hilfe der Analyse von Basenhäufigkeiten können auch Dateifragmente erfolgreich aufgefunden werden.
4. Das Vorhandensein eines Kontextes in Form eines intakten Filesystems oder Datei-Header/-Footer ist für den Einsatz bioinformatischer Methoden nicht erforderlich.
5. Die Berechnung globaler und lokaler Alignmentsscores kann die Wahrscheinlichkeiten für falsch-positive und falsch-negative Ergebnisse in Verbindung mit der zuvor erfolgten Analyse von Basenhäufigkeiten reduzieren.
6. Templates (Muster) in Form von Konsensussequenzen Multipler Alignments können zur Suche nach ähnlichen Dateien eingesetzt werden. Der Sucherfolg wird u. a. wesentlich durch die Auswahl der Sequenzen für das Multiple Alignment beeinflusst. Das Suchergebnis dürfte sich in Abhängigkeit von den Größen der Untersuchungsfenster im Image und Template verbessern.
7. Die Anwendung von Basenhäufigkeiten und Alignments zum Auffinden von Dateien bedingt keine clusterorientierte Suche (eine Verbesserung der Suchergebnisse hierdurch ist jedoch zu vermuten).
8. Die Datensuche erfolgt im Image und ggf. Template abschnittsweise, wobei die Größe des Untersuchungsfensters nicht der Größe der zu suchenden Daten entsprechen muss. Gleichwohl ist der Sucherfolg von der Fenstergröße abhängig.