Implementierung von morphologischen Filtern in ein Bildverarbeitungssystem

PRAXISBERICHT

für die Prüfung zum

Bachelor of Science

des Studienganges Angewandte Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Robert Illner

01.10.2012

Bearbeitungszeitraum:	13 Wochen
Matrikelnummer:	8582612
Kurs:	TINF11B2
Ausbildungsfirma:	Institut für Angewandte Informatik, Karlsruher Institut für Technologie, Karlsruhe
Betreuer:	Dr.-Ing. Bernd Köhler

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Aus den benutzten Quellen direkt oder indirekt übernommene Gedanken habe ich als solche kenntlich gemacht.
Diese Arbeit wurde bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ort, Datum	Unterschrift

Inhaltsverzeichnis

1 Einleitung

Am Institut für Angewandte Informatik des Karlsruher Instituts für Technologie (KIT) entwickelt eine Bildverarbeitungsgruppe seit vielen Jahren das Bildverarbeitungssystem DIPLOM (**D**igital **I**mage **P**rocessing **L**ibrary f**o**r **M**icrostructures) [1], welches permanent erweitert und aktualisiert wird. Das System findet sowohl im internen Bereich des KITs, als auch im industriellen Sektor Anwendung. Somit bietet DIPLOM neben den Standardfunktionen der Bildverarbeitung (wie z.B. die Binärbilderzeugung mit einem Schwellwert) auch spezialisierte Funktionen für die Anwendung auf Microstrukturen wie beispielsweise das Zusammensetzen von Mosaikbildern. Das Programm nutzt die MIL (**M**atrox **I**maging **L**ibrary) als Basisbibliothek und wird im Microsoft Visual Studio 2010 in C++ implementiert, wobei Apache Subversion (SVN) [2] zur Quellcodeverwaltung verwendet wird. Die MIL stellt umfangreiche bildverarbeitende Module wie beispielsweise die konturbasierte Mustersuche oder den Einzug von Bildern für Wissenschaft und Industrie bereit. Eine der Standardfunktionen welche in der MIL enthalten sind, sind die sogenannten „morphologischen Filter“. DIPLOM soll nun durch dieses Modul erweitert werden.

Der Begriff Morphologie kommt aus dem Griechischen (*morphé* - „Gestalt, Form“ und *lógos* - „Wort, Lehre, Vernunft“) und beschreibt die Lehre von Formen [3]. Von der Wortabstammung abgeleitet sind morphologische Filter also in der Lage, die Gestalt von Bildern methodisch zu verändern. Zunächst waren diese Filter nur für Binärbilder vorgesehen. Inzwischen finden morphologische Filter aber auch bei Grauwert- und Farbbildern Anwendung, auch wenn sich diese Methoden stark von den binären Operationen unterscheiden [4].

2 Morphologische Filter

In diesem Kapitel werden zunächst wichtige Grundlagen der digitalen Bildverarbeitung vermittelt. Es wird erklärt, was Punkt- und Nachbarschaftsoperationen sind und was man im Allgemeinen unter „Filter“ oder „Nachbarschaft“ in der Bildverarbeitung versteht. Zudem wird auf die genaue Funktionsweise der moprhologischen Filter eingegangen, sowie die Darstellung der Operationen mit Hilfe von Strukturelementen und Punktmengen nähergebracht. Vorerst wird die Funktionsweise von morphologischen Filtern anhand von Binärbildern gezeigt, in 2.4 wird diese dann auf Grauwert- und Farbbilder erweitert.

2.1 Grundlagen

Um ein besseres Verständnis dieser Arbeit zu ermöglichen werden hier zunächst grundlegende Begriffe und elementare Themenbereiche der Bildverarbeitung angesprochen und erklärt. Diese Grundlagen werden in „Digitale Bildverarbeitung“ von Bernd Jähne behandelt [5]. Im Folgenden werden diese Grundlagen basierend auf Jähnes Werk geschildert.
Die Bildverarbeitung beschäftigt sich mit vielen unterschiedlichen Aufgabenstellungen. Oft müssen Kanten, Linien oder Ecken lokalisiert und bearbeitet werden, oder es sollen sogar ganze Bereiche mit nahezu konstanten Grauwerten verarbeitet werden. Auch die Ausbesserung von Bildern mit Bewegungsunschärfe oder fehlerhafter Fokussierung, oder die Korrektur von Schäden die beispielsweise bei der Übertragung entstehen, sind Beispiele für Prozeduren, die nicht mittels Punktoperationen durchgeführt werden können. Punktoperationen zeichnen sich dadurch aus, dass der neue Grau- oder Farbwert eines Pixels allein in Abhängigkeit von dem eigenen bisherigen Wert und der eigenen bisherigen Position berechnet wird, ohne Werte oder Eigenschaften der umliegenden Pixel zu beachten. Diese Punktoperationen können meist nur zur Vorverarbeitung der Bildverarbeitung verwendet werden, wie zum Beispiel der Kontrastverbesserung.

Für die zu Beginn genannten Anwendungen werden also Operationen benötigt, die hier zum Beispiel Kanten von Bereichen mit konstanten Grauwerten unterscheiden kann. Diese Operationen nennt man Nachbarschaftsoperationen. Sie beziehen die Werte und Positionen von umliegenden Pixeln mit in die Berechnung ein und liefern ein neues Ergebnisbild. Nachbarschaftsoperatoren sind also in der Lage, Kanten mit höheren Pixelwerten von nahezu homogenen Bereichen zu unterscheiden und somit den kompletten Teilbereich des

Bildes als Ganzes zu bearbeiten. Allerdings geht bei der Anwendung eines Nachbarschaftsoperators immer Information verloren. Nicht immer kann durch eine andere Operation wieder das gleiche Ursprungsbild erstellt werden. Deshalb werden Nachbarschaftsoperationen auch Filter genannt. Bei den morphologischen Filtern handelt es sich um Nachbarschaftsoperationen. Spricht man von Nachbarschaft, so muss man zwei verschiedene Arten unterscheiden: werden nur direkt angrenzende Pixel entfernt oder angefügt, so spricht man in diesem Falle von einer so genannten 4er-Nachbarschaft. Neben dieser gibt es noch die 8er-Nachbarschaft, welche zusätzlich die über die Diagonalen angrenzenden Pixel beinhaltet. Geht man also von einer 8er-Nachbarschaft aus, so werden mehr Strukturflächen entfernt beziehungsweise angefügt. Abbildung 2.1 zeigt den Unterschied.

Abbildung 2.1: 4er-Nachbarschaft (links), 8er-Nachbarschaft (rechts)

Die wesentliche Verfahrensweise von morphologischen Filtern besteht im „Schrumpfen“ und „Wachsen“ von Strukturen. In der Bildbearbeitung werden die Operationen Schrumpfen und Wachsen auch als Erosion und Dilation (auch Dilatation) bezeichnet.

2.2 Prinzip von morphologischen Filtern

Grundsätzlich werden bei der Erosion diejenigen Randpixel entfernt, die direkt an die Hintergrundpixel angrenzen (Abbildung 2.2), bei der Dilation hingegen werden diejenigen Hintergrundpixel dem Bild zugefügt, die direkt an den Rand angrenzen (Abbildung 2.3). In diesen Beispielen wird von einer 4er-Nachbarschaft ausgegangen.

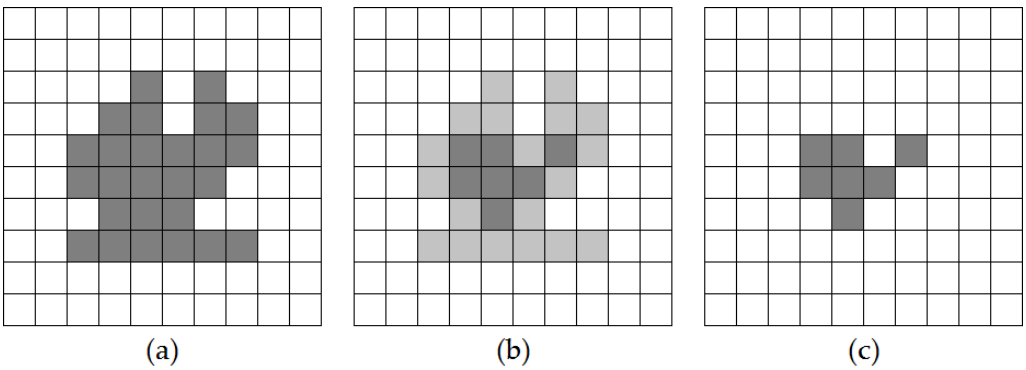

Abbildung 2.2: Erosion: Quellbild (a), zu entfernende Pixel (b), Ergebnis (c) [4]

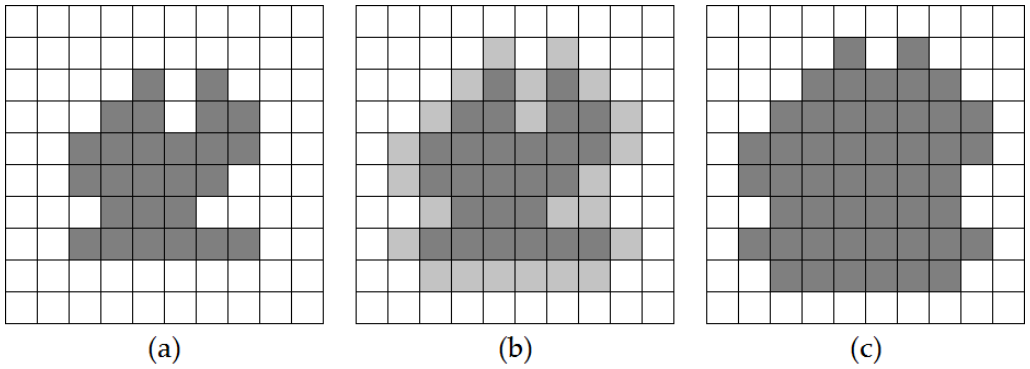

Abbildung 2.3: Dilation: Quellbild (a), hinzuzufügende Pixel (b), Ergebnis (c) [4]

2.3 Strukturelement und Punktmengen

Das Verhalten morphologischer Filter wird durch eine Matrix festgelegt, welche als Strukturelement (H) bezeichnet wird. Soll ein Binärbild (I) bearbeitet werden, wird davon ausgegangen, dass auch das Strukturelement nur die Werte `0` und `1` annimmt. Die Bildpunkte des Strukturelements werden zur besseren Veranschaulichung als Koordinatenpaare betrachtet, woraus sich die Punktmenge Q_H bildet. So ergibt sich ein zweidimensionales Koordinatensystem, in dem jeder Bildpunkt seine eigene Koordinate hat. Strukturelemente besitzen einen zentralen „Hot-Spot“, der als Ausgangspunkt für morphologische Operationen dient. Auch die Punktmenge Q_I des Binärbildes kann aus Koordinatenpaaren zusammengestellt werden. Abbildung 2.4 zeigt die Darstellung eines Binärbildes und eines Strukturelements in einem Koordinatensystem und als Punktmenge.

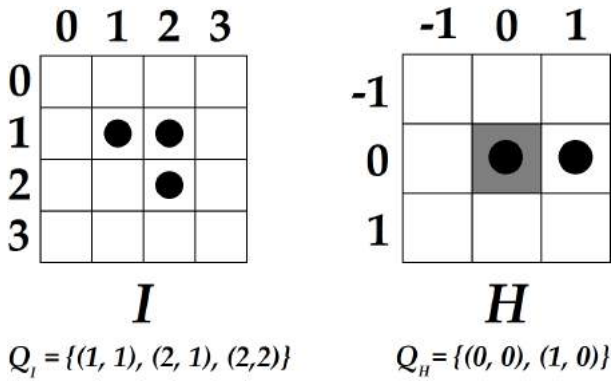

Abbildung 2.4: Binärbild I und binäres Strukturelement H als Mengen von Q_I und Q_H

Wird nun eine morphologische Operation durchgeführt, so wird das Strukturelement mit dem Bild abgeglichen. Dabei wird der Hot-Spot nacheinander auf jeden Bildpunkt ausgerichtet und es wird geprüft, welchen Wert der Bildpunkt an dieser Stelle besitzt. Bei der Dilation wird überprüft, ob das Bild an dieser Stelle den Wert `1` hat. Ist dies der

Fall, wird das Strukturelement an dieser Stelle auf das Bild kopiert. Ausgehend vom bereits gezeigten Binärbild I und Strukturelement H, werden diese Schritte für eine Dilation durchlaufen (Abbildung 2.5):

- Hot-Spot bei `(1, 1)`: Pixel `(2, 1)` bekommt Wert `1` (schon vorhanden)
- Hot-Spot bei `(2, 1)`: Pixel `(3, 1)` bekommt Wert `1`
- Hot-Spot bei `(2, 2)`: Pixel `(3, 2)` bekommt Wert `1`

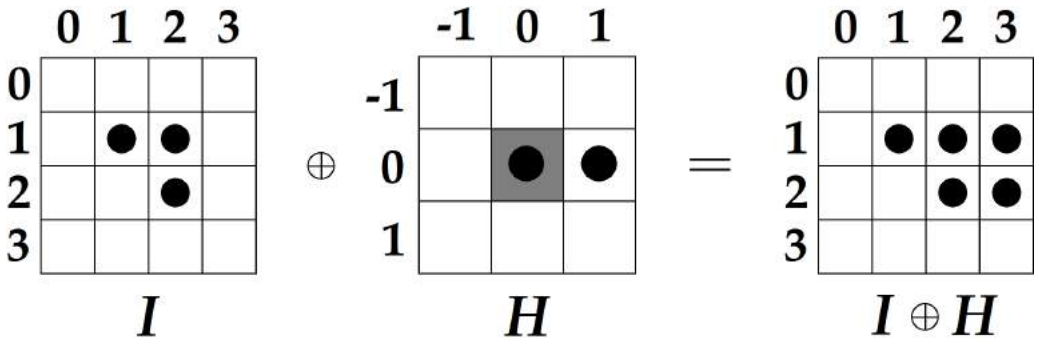

Abbildung 2.5: Dilation von I mit H. H wird an jedem Punkt des Originalbilds I repliziert.

Bei der Erosion von I mit H bleiben nur diejenigen Pixel mit dem Wert `1` erhalten, bei denen das Strukturelement vollständig ist. Hier geschieht für die Pixel Folgendes (Abbildung 2.6):

- Hot-Spot bei `(1, 1)`: Pixel `(2, 1)` bekommt Wert `0`
- Hot-Spot bei `(2, 1)`: Pixel `(3, 1)` bekommt Wert `0` (schon vorhanden)
- Hot-Spot bei `(2, 2)`: Pixel `(3, 2)` bekommt Wert `0`

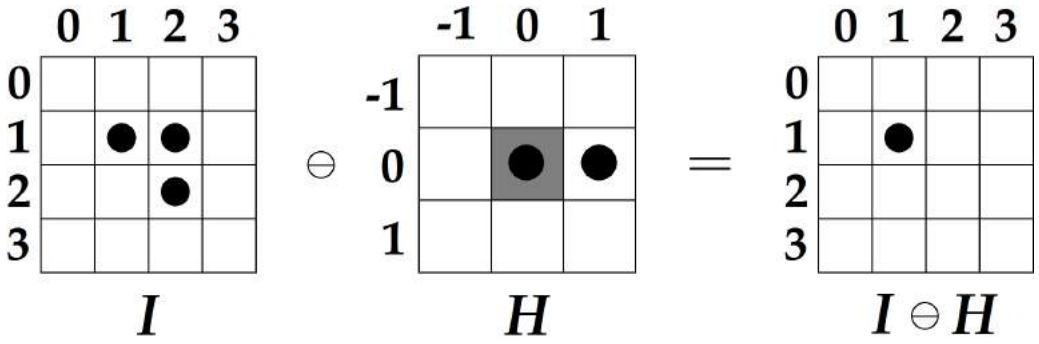

Abbildung 2.6: Erosion von I mit H. H ist nur bei `(1, 1)` vollständig in I eingebettet.

Die Kombination von Erosion und Dilation führt zu unterschiedlichen Ergebnissen. Folgt eine Dilation auf eine Erosion (Opening), so werden zuerst alle Strukturen verkleinert (Erosion) und anschließend wieder in gleichem Maße vergrößert (Dilation). Dies kann dazu führen, dass sehr kleine Bildelemente komplett eliminiert werden. Folgt analog dazu eine Erosion auf eine Dilation (Closing), so werden zuerst alle Strukturen vergrößert (Dilation) und anschließend wieder in gleichem Maße verkleinert (Erosion). Im Gegensatz zum Opening kann das Closing dazu führen, dass Löcher oder Lücken im Bild gefüllt werden. Ob bei einem Opening Strukturen beseitigt oder bei einem Closing Lücken geschlossen werden, hängt von dem verwendeten Strukturelement ab, welches ausschlaggebend für das Ergebnis der Operationen ist [4].

2.4 Anwendung auf Grauwert- und Farbbilder

Im Gegensatz zur Anwendung auf Binärbilder werden die Strukturelemente bei Grauwertbildern als 2D-Funktionen mit beliebigen Werten dargestellt. Das Verfahren der morphologischen Operationen selbst ist jedoch ähnlich: Der Hot-Spot des Strukturelements H wird nacheinander auf die Bildpunkte des Grauwertbildes I ausgerichtet. Bei der Dilation werden die Werte der Pixel des Grauwertbildes jeweils zu den einzelnen Werten von H addiert. Der Maximalwert der aktuellen Filterposition wird dann in das Ergebnisbild an der Stelle des Hot-Spots eingetragen. Diese Vorgehensweise ist analog auf die Erosion zu übertragen. Hier werden die einzelnen Werte subtrahiert und schließlich der Minimalwert der aktuellen Filterposition im Ergebnisbild an der Position des Hot-Spots eingetragen. Die Abbildungen 2.7 und 2.8 zeigen das Ergebnis einer Grauwert-Dilation und Erosion für jeweils vier Filterpositionen.

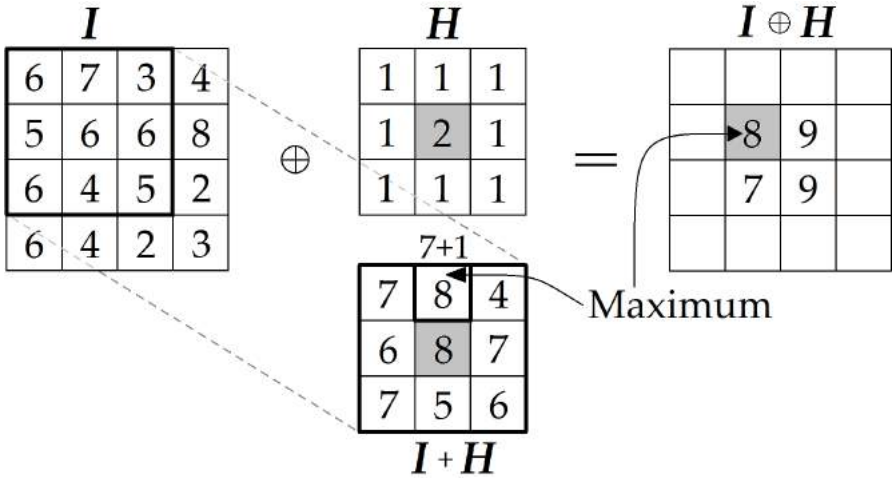

Abbildung 2.7: Grauwertbild-Dilation: Originalbild I, 3x3-Strukturelement H, Zwischenergebnis $I+H$, Ergebnisbild $I \oplus H$ [4]

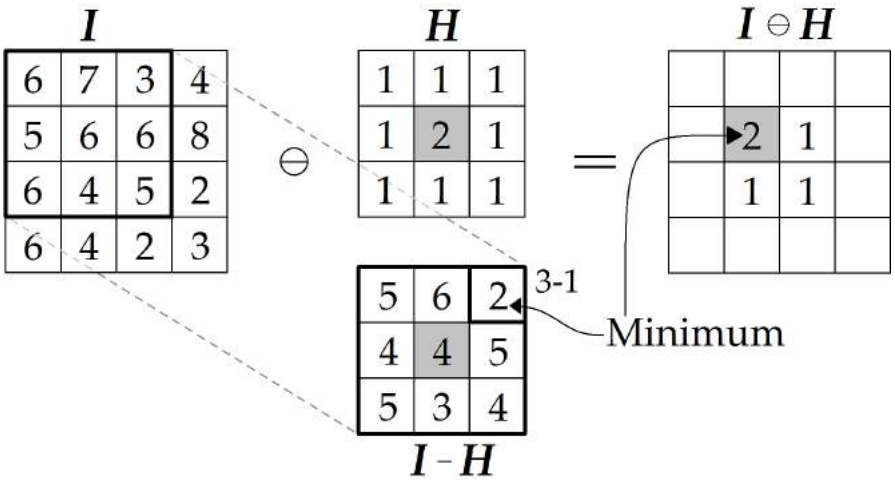

Abbildung 2.8: Grauwertbild-Erosion: Originalbild I, 3x3-Strukturelement H, Zwischenergebnis I-H, Ergebnisbild $I \ominus H$ [4]

Bei dieser Verfahrensweise ist allerdings zu beachten, dass auch Nullwerte das Ergebnis beeinflussen. Daher besteht die Möglichkeit, einen Bildpunkt mit einem X („don't care") zu kennzeichnen, was einer leeren Zelle entspricht. Für die Anwendung auf Farbbilder gilt die gleiche Vorgehensweise wie bei Grauwertbildern. Allerdings wird bei morphologischen Operationen auf Farbbilder die Berechnung separat für jeden Farbkanal durchgeführt [4].

2.5 Verschiedene Bildbeispiele

Die folgenden Abbildungen veranschaulichen die Wirkungsweise morphologischer Filter auf Binärbilder. Abbildung 2.9 zeigt das ursprüngliche Binärbild, auf das die vier Operationen Erosion, Dilation, Opening beziehungsweise Closing jeweils 1, 3, 5, 10 und 15 mal angewandt werden.

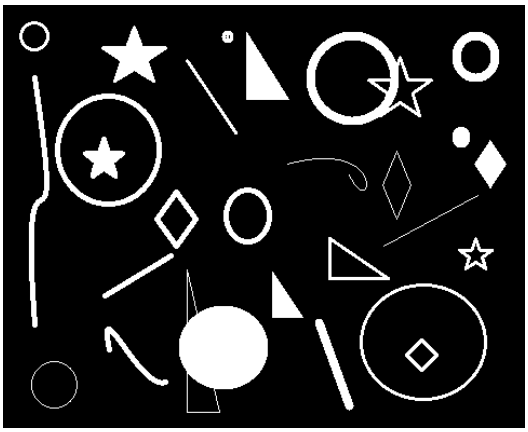

Abbildung 2.9: Originalbild: Unterschiedliche weiße Strukturen und Formen verschiedener Größe auf schwarzem Hintergrund

Bei diesen Beispielen ist die bereits in Kapitel 2.3 angesprochene Wirkung von Opening und Closing gut zu erkennen: Das Opening führt zur Beseitigung kleinster Bildelemente, was beispielsweise an den kleinen Kreisen und Strichen erkannt werden kann, welche nach Durchführung eines Openings kaum noch vorhanden sind. Hierbei kommt es auf die Größe des verwendeten Strukturelements an. Vordergrundstrukturen, die kleiner als das Strukturelement sind, werden vollständig eliminiert, etwas größere Strukturen werden hingegen nur verkleinert. Vergleichsweise sehr große Strukturen werden allerdings kaum beeinflusst, was an dem großen weißen Fleck beim Opening in Abbildung 2.10 zu erkennen ist. Ebenfalls zeigt das Beispiel des Closings, wie mit dieser Operation kleinere dunkle Lücken gefüllt werden können. In diesem Fall werden die weißen Pixel als Vordergrund und die schwarzen als Hintergrund angesehen.

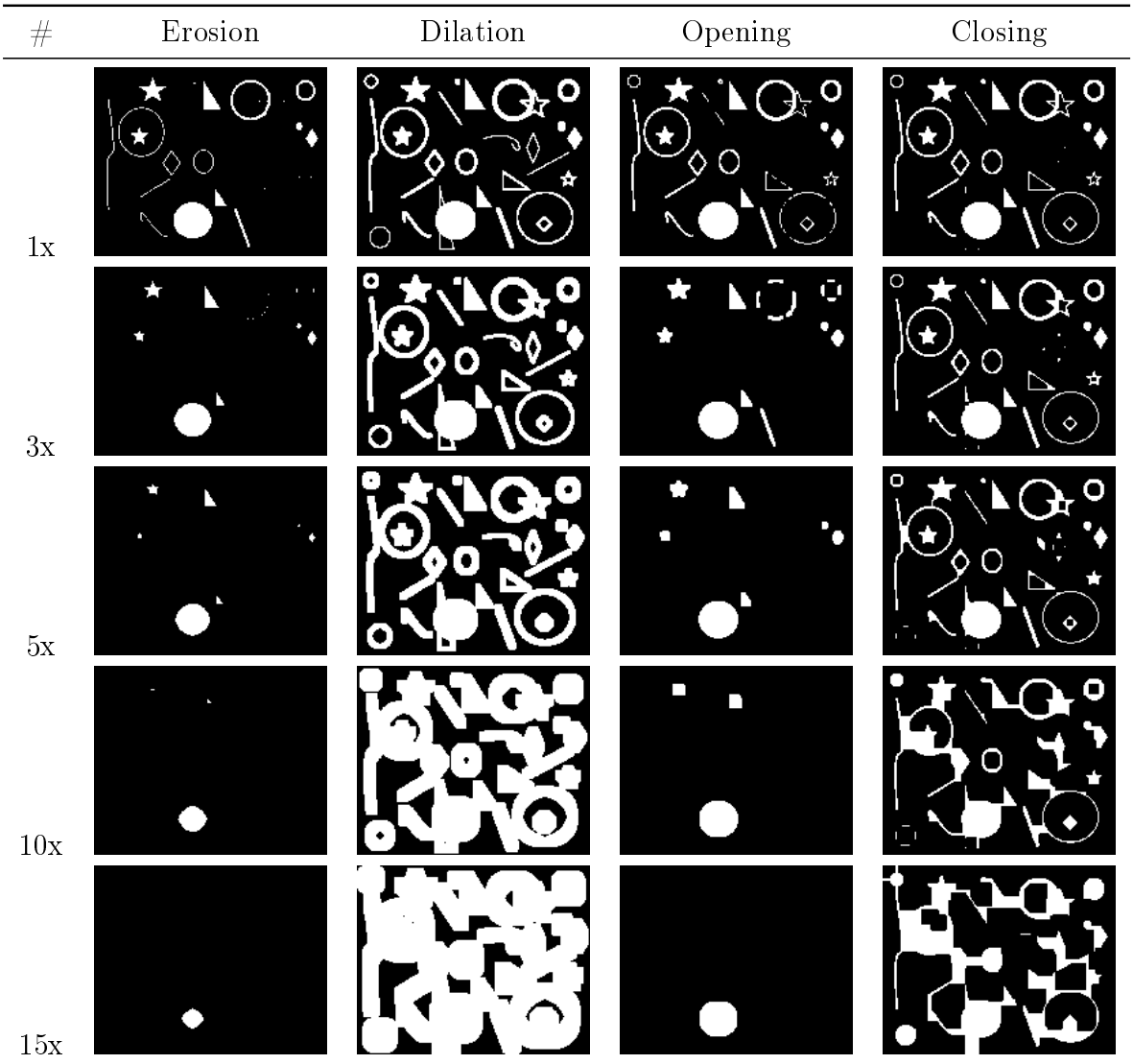

Abbildung 2.10: Auf ein Binärbild angewandte Erosion, Dilation, Opening und Closing mit unterschiedlicher Wiederholungsanzahl

3 Implementierung

In diesem Kapitel wird die Implementierung des Moduls „Morphologische Filter“ im DIPLOM-System beschrieben. Dabei wird zunächst auf die erforderlichen Funktionen des Moduls eingegangen und anschließend der Programmablauf behandelt. Bei der Darstellung des Programmablaufs werden die einzelnen Schritte des Algorithmus jeweils erläutert und der Umgang mit Sonderfällen besprochen.

3.1 Anforderungen

Da die bereits in der MIL vorhandenen Funktionen für Erosion, Dilation, Opening und Closing bestimmte Parameter verlangen, müssen diese über den Dialog weitergegeben werden. Bei diesen Parametern handelt es sich jeweils um die Bezeichnung des Quell- und des Ziel-Bildpuffers, sowie die Angabe der Wiederholungsanzahl der Operation und ob es sich um ein Binär- oder Grauwertbild handelt. Das Modul soll so aufgebaut sein, dass von vornherein keine falschen Parameter gesetzt werden können oder es muss entsprechend damit umgehen können und eine falsche Benutzereingabe (wie zum Beispiel ein Sonderzeichen) so anpassen, dass es zu keinem Fehler kommen kann und dadurch die Robustheit des Moduls gewährleistet wird. Da sich beispielsweise bei einer Erosion eines Bildes der Größe von 300 auf 350 Pixeln spätestens nach 350 Wiederholungen ein komplett schwarzes Bild entwickelt, ist es sinnvoll die Eingabe der Wiederholungsanzahl auf diese Werte zu begrenzen, um unnötigen Rechenaufwand zu vermeiden. Nebenbei ist es auch wichtig, den Dialog so kompakt und zugleich übersichtlich wie möglich zu gestalten, damit eine schnelle und unkomplizierte Benutzerinteraktion gewährleistet werden kann.

Zur Wahl eines Quellbildes soll es die Möglichkeit geben zwischen in DIPLOM geöffneten Bildern oder (falls vorhanden) einem Live-Bild zu wählen. Zwischen den zur Verfügung stehenden Quellen soll dann schnell und einfach gewechselt werden können. Außerdem soll zusätzlich zum Dialog ein Vorschaufenster vorhanden sein, das während dem Ändern der Parameter eine Vorschau des Ergebnisses anzeigt.

Um ein Ergebnis abzuspeichern oder weiterverarbeiten zu können soll auf Knopfdruck eine neue Ansicht mit dem im Vorschaufenster angezeigten Ergebnisbild geöffnet werden. Der Titel der Ansicht und somit auch der Titel des Ergebnisbildes soll entsprechend der durchgeführten Operationen benannt sein. Das Bild kann dann auf Wunsch über die Be-

nutzeroberfläche von DIPLOM abgespeichert werden.

Ein weiterer wichtiger Faktor ist die Schnelligkeit des Moduls während des Programmdurchlaufs. Da das Filter bei Verwendung eines Live-Bildes auf jedes ankommende Bild angewandt werden muss, muss hier im Vergleich zur Verwendung eines Standbildes als Quelle ein enormer Rechenaufwand berücksichtigt werden. Deshalb ist es notwendig bei der Implementierung auf Effizienz zu achten, damit ein Algorithmus entwickelt wird, der den Programmablauf möglichst nicht verlangsamt.

3.2 Programmablauf

Ist über die Benutzeroberfläche von DIPLOM eine Bildquelle geöffnet, so kann das Modul für die morphologischen Filter über den Menüpunkt „Bildverarbeitung“ geöffnet werden (Abbildung 3.1).

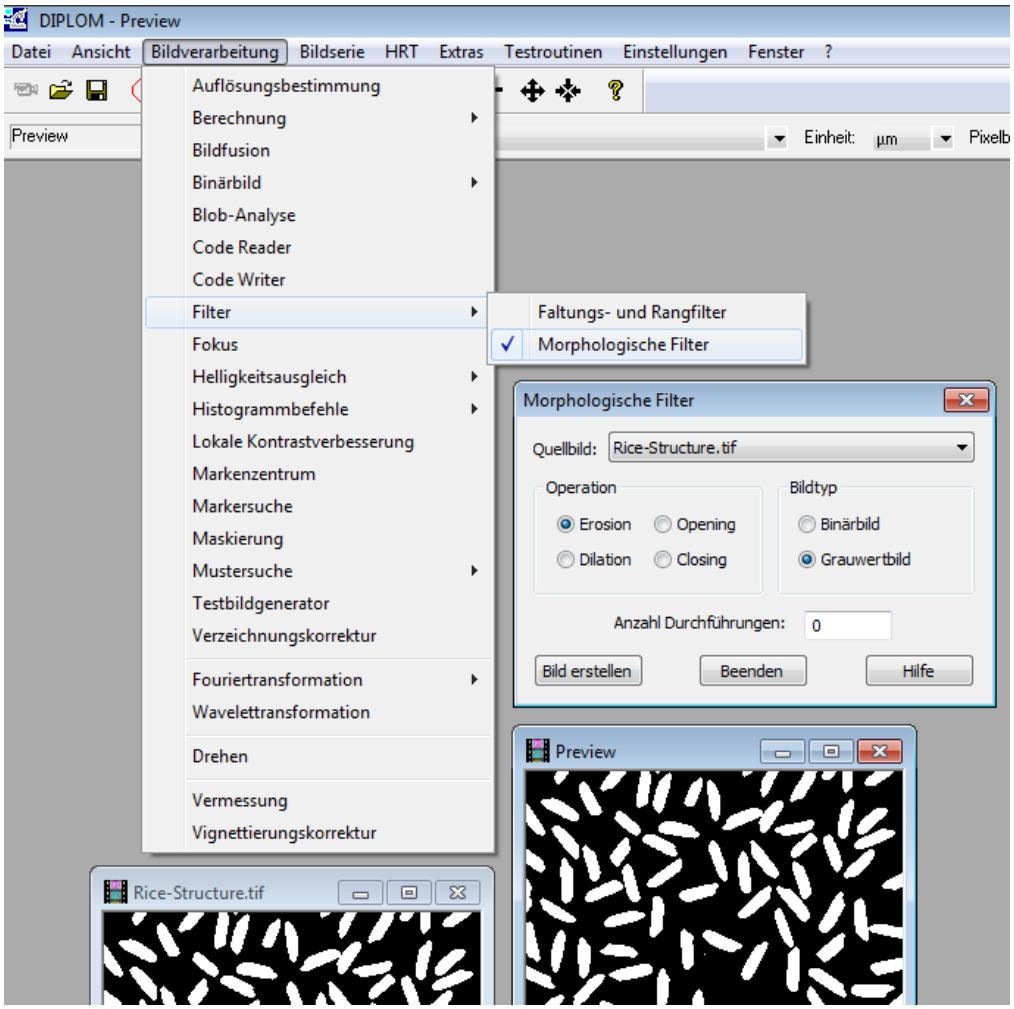

Abbildung 3.1: Layout und Öffnen des Dialogs über das DIPLOM-Menü

Abbildung 3.2 zeigt ein Aktivitätsdiagramm als Übersicht über die Aktionen des Moduls. Der Benutzer hat die Möglichkeit, die drei Hauptaktionen „Quellbild ändern“, „Bild erstellen“ und „Parameter ändern“ auszulösen. Aktivitäten, die mit einer stilisierten Harke gekennzeichnet sind, werden an entsprechender Stelle detaillierter behandelt.

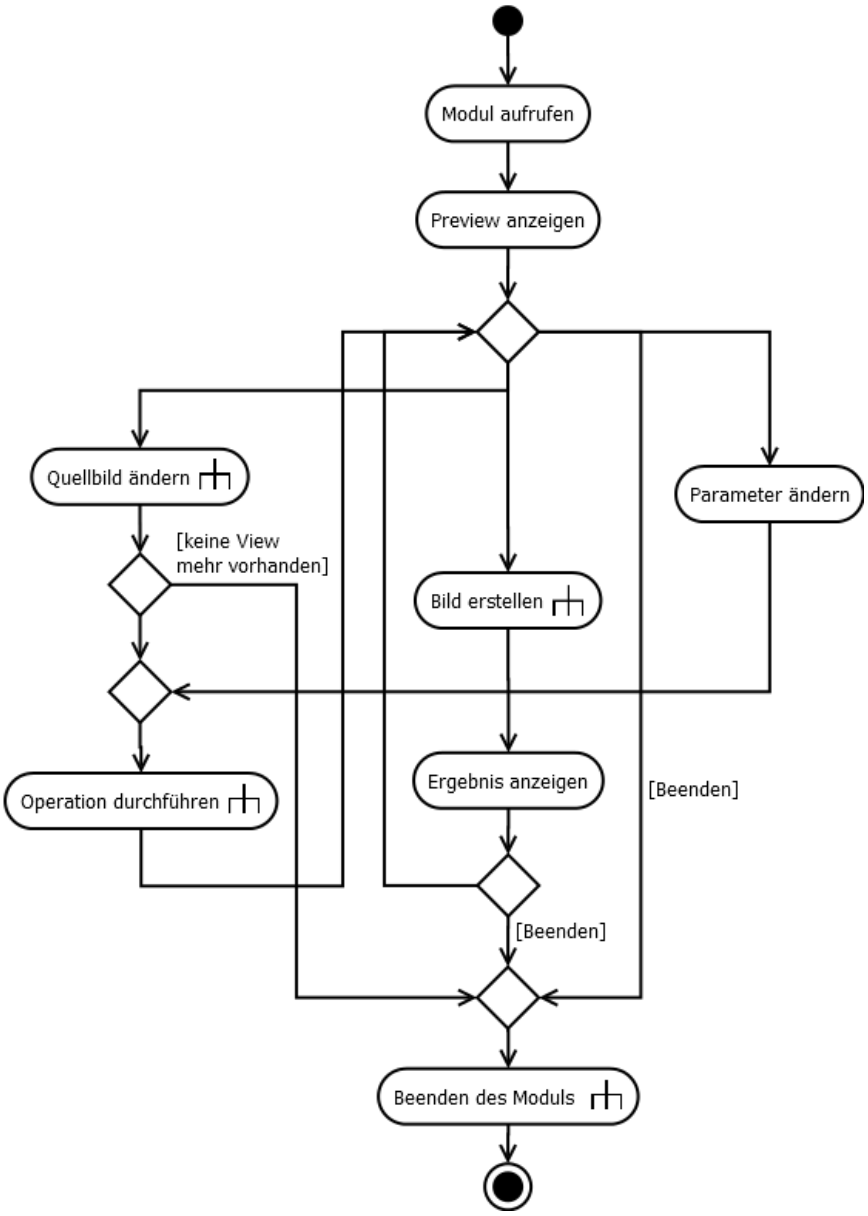

Abbildung 3.2: Übersicht des Moduls, Benutzeraktionen „Quellbild ändern“, „Bild erstellen“ und „Parameter ändern“

3.2.1 Initialisierung

Beim Starten dieses Moduls initialisiert der Konstruktor der Klasse `CMorphDlg` zunächst die Variablen mit vorgegebenen Standardwerten. Die Wahl der Operation ist zu Beginn auf „Erosion“ gesetzt und die Anzahl an Wiederholungen der gewünschten Operation ist standardmäßig auf 0 eingestellt. Mit diesen Einstellungen entspricht das Ergebnisbild dem Originalbild, da 0 Operationsdurchläufe auf das Originalbild angewandt werden.

Da die Instanz des Dialogs beim Beenden nicht zerstört, sondern nur das Fenster selbst ausgeblendet wird, werden in der Funktion `OnInitDialog` ebenfalls die gleichen Variablen wie im Konstruktor auf ihre Standardwerte gesetzt. Diese Initialisierungen sind hier zusätzlich notwendig, damit die Variablen beim Schließen und erneutem Öffnen des Dialogs zurückgesetzt werden, da der Konstruktor bei neuem Aufruf des Dialogs nicht erneut ausgeführt wird, `OnInitDialog` wird jedoch immer beim Öffnen des Dialogs aufgerufen.

Zur einfachen Auswahl von Quellbildern wird ein Kombinationsfeld verwendet. Deshalb ruft `OnInitDialog` als nächstes die Funktion `InitComboBox` auf, in welcher das Kombinationsfeld mit Einträgen der offenen Bildquellen gefüllt wird. Die Methode initialisiert das Kombinationsfeld, welches die verschiedenen Bildquellen auflisten soll. Dazu werden nacheinander alle offenen Ansichten beziehungsweise Bildquellen durchgegangen, wobei das Vorschaufenster ignoriert wird, da dieses nicht als Quelle sondern lediglich zur Vorschau dient. Dabei wird jeweils der Titel der zu überprüfenden Bildquelle ermittelt. Wenn es sich bei der betroffenen Ansicht nicht um das Vorschaufenster handelt, wird der zugehörige Titel dem Kombinationsfeld hinzugefügt und der jeweilige Index des Eintrags zwischengespeichert. Anschließend wird Index und Ansicht miteinander verknüpft, sodass über die Indizes im Kombinationsfeld auf die entsprechenden Quellen zugegriffen werden kann. Die Einträge im Kombinationsfeld sind dabei nicht alphabetisch sortiert, sondern nach ihrer Anordnung im Fenstermanager (vorderste zuerst). Also stehen aktivierte beziehungsweise ausgewählte Quellen im Kombinationsfeld immer an erster Stelle.

Nachdem `InitComboBox` vollständig abgearbeitet wurde, überprüft `OnInitDialog`, ob es sich bei der ausgewählten Bildquelle um ein Live-Bild handelt. Ist dies der Fall, so wird der Bildeinzug angehalten und dafür ein Thread gestartet. Der Bildeinzug muss in einem separaten Thread ablaufen, da die Funktion, die für den einzelnen Bildeinzug zuständig ist, durchgehend darauf wartet, ein Bild einzuziehen und somit keine anderen Aktionen wie Parameteränderungen oder auch das Beenden des Dialogs möglich wären.

Nach dem Starten des Threads werden die Maße des Quellbildes ermittelt und eine neue Ansicht mit den Abmessungen des Quellbildes und dem gleichen Bild erstellt. Die neue Ansicht stellt dann das Vorschaufenster - die so genannte Preview - dar. Sollte das Erstel-

len des Vorschaufensters nicht erfolgreich gewesen sein, also die Variable nicht initialisiert ist, so wird eine Fehlermeldung ausgegeben und der Dialog vorzeitig beendet. Nach Erzeugung des Vorschaufensters wird diesem noch der Titel „Preview“ gegeben. Letztendlich wird in `OnInitDialog` im Falle eines ausgewählten Livebildes noch ein einzelnes Bild eingezogen und der Dialog beim Hauptfenster registriert, worauf in 3.2.3 näher eingegangen wird.

3.2.2 Parameterprüfung

Bei Änderungen der Anzahl von Wiederholungen der Operation, wird die aktuelle Eingabe auf ihre Gültigkeit getestet und gegebenenfalls angepasst, bevor dieser Wert weiter verwendet wird. Dazu wird die implementierte Funktion `AdjustIterationInput()` angewandt, in welcher die Überprüfung und gegebenfalls die Anpassung stattfindet.

In dieser Methode wird zunächst getestet, ob die aktuelle Eingabe für die Wiederholungsanzahl eine Zahl ist. Dazu wird jede einzelne Stelle des Eingabestrings untersucht, da nicht bekannt ist, an welcher Stelle der Eingabe ein Zeichen geändert wurde. Wird dabei ein Zeichen gefunden, das keine Ziffer ist, so wird dieses Zeichen wieder aus dem Eingabestring gelöscht und der Cursor im Eingabefeld an die vorherige Position gesetzt. Zusätzlich werden (falls vorhanden) Nullen am Anfang der Eingabe entfernt. Ist das bis hier beschriebene Verfahren durchlaufen, so handelt es sich bei der Eingabe nun um eine natürliche Zahl (ohne Null) oder der gegebenfalls bearbeitete Eingabestring ist aufgrund von Entfernen von ungültigen Zeichen oder führenden Nullen nun leer. Der Inhalt des Eingabestrings wird nun in die richtige Wiederholungsvariable vom Typ Integer geschrieben. Falls die Zeichenkette jedoch leer ist, wird in die Wiederholungsvariable der Wert `0` geschrieben und der Cursor im Eingabefeld dahinter gesetzt. Die Variable hat nun in jedem Falle einen gültigen Wert. Um jedoch unnötigen Rechenaufwand durch zu hohe Eingaben zu vermeiden, ist es sinnvoll die Anzahl an Wiederholungen auf die Bildgröße der Quelle zu begrenzen, wie bereits in Kapitel 3.1 erwähnt wurde. Deshalb wird die Auflösung des ausgewählten Bildes ermittelt und der größere der beiden Werte (Höhe oder Breite in Pixel) zwischengespeichert. Letztendlich vergleicht `AdjustIterationInput()` den zwischengespeicherten Wert mit dem Wert der Wiederholungsvariable. Ist die Angabe der Anzahl an Wiederholungen größer als der zwischengespeicherte Wert, wird die Boolean-Variable `EnableCreate` auf `false` gesetzt, was die Deaktivierung des Buttons zur Aufnahme eines Ergebnisbildes zur Folge hat. Außerdem wird eine Fehlermeldung ausgegeben und die Wiederholungsvariable auf 0 gesetzt. Ist aber der zwischengespeicherte Wert größer, so besitzt die Wiederholungsvariable einen erlaubten Wert und wird nicht mehr verändert. `EnableCreate` wird auf `true` gesetzt und erlaubt somit die Aufnahme eines Ergebnisbildes.

3.2.3 Nachrichtenbehandlung

Am Ende von `OnInitDialog` wird im Falle eines ausgewählten Livebildes noch ein einzelnes Bild eingezogen und der Dialog beim Hauptfenster von DIPLOM registriert, um über Änderungen von Ansichten informiert zu werden. Diese Registrierung ist notwendig, da der Dialog auf Verwendung von Nachrichten bezüglich der Änderung von Ansichten, wie beispielsweise das Aktivieren oder Deaktivieren von Quellen angewiesen ist. Ansichten beziehungsweise offene Bildquellen sind in DIPLOM immer vom Typ `CMilVideoView`. Wird nun eine solche `CMilVideoView` deaktiviert, aktiviert, geöffnet, geschlossen oder aktualisiert, so wird eine entsprechende Nachricht an das Applikationsfenster geschickt. Diese Nachrichten sind in der Datei `Messages.h` definiert. Notwendigerweise muss der Dialog beim Beenden auch beim Hauptfenster abgemeldet werden. Um eine übersichtliche Implementierung zu gewährleisten, ist es von Vorteil die Funktion `UnregAndDestroyDlg()` zu erstellen, damit diese bei Bedarf aufgerufen werden kann, anstatt mehrmals die gleichen Codefragmente zu verwenden. Es gibt insgesamt vier Stellen im Code, an denen genau diese Schritte notwendig sind. Dabei handelt es sich um das Schließen des Vorschaufensters, das Schließen des letzten offenen Quellbildes, das Schließen des Dialogs über den Beenden-Button oder das X-Symbol in der Titelleiste.

In dieser Methode wird die bereits vorhandene Funktion `UnregViewDependentWnd()` der Klasse `CMainFrame` benutzt, um die Abmeldung beim Hauptfenster vorzunehmen. Außerdem wird der Dialog geschlossen, beziehungsweise ausgeblendet, wobei die Variablen und die Instanz des Dialogs weiterhin bestehen. Schließlich wird in `UnregAndDestroyDlg()` noch überprüft, ob der Bildeinzug angehalten wurde und ein Live-Bild existiert. Ist dies der Fall, so wird der kontinuierliche Bildeinzug wieder aktiviert. Abbildung 3.3 veranschaulicht den Beenden-Vorgang des Moduls.

Wird eine Ansicht aktiviert, deaktiviert, geschlossen, erstellt oder aktualisiert, so gibt es dafür jeweils eine Methode die aufgerufen wird, nachdem dem Dialog die entsprechende Nachricht mitgeteilt wurde. Nach Erstellung einer Ansicht (also bei Hinzukommen einer neuen Bildquelle) wird der Titel dieser Ansicht (sofern diese initialisiert ist und es sich nicht um das Vorschaufenster handelt) dem Kombinationsfeld hinzugefügt und der zugehörige Index im Kombinationsfeld mit der Ansicht verknüpft, wie bereits in `InitComboBox()` beschrieben wurde.

Wenn eine Nachricht bezüglich der Aktualisierung einer Ansicht empfangen wird, kann es sich um eine Titeländerung der Ansicht oder um den Empfang eines neuen Einzelbildes (z.B. eines Live-Bildes) handeln. Teilt die Nachricht eine Änderung des Titels mit, wird die betroffene Ansicht im Kombinationsfeld gesucht. Der alte Titel der Ansicht im Kombinationsfeld wird dann durch den neuen Titel ersetzt, der Index wieder mit der Ansicht

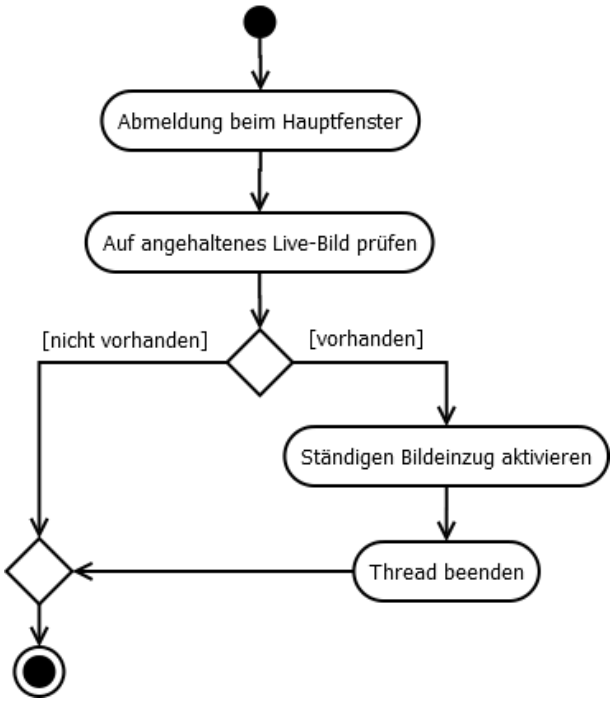

Abbildung 3.3: Aktivitätsdiagramm „Beenden des Moduls“

verknüpft und der neue Eintrag im Kombinationsfeld ausgewählt. Handelt es sich bei der Nachricht aber um den Empfang eines neuen Einzelbildes und das Live-Bild ist als Quelle ausgewählt, wird die Methode `UpdatePre()` ausgeführt, welche die gewählte Operation mit den gewünschten Parametern auf das Quellbild anwendet und die Vorschau im Vorschaufenster anzeigt. Auf diese Funktion wird in 3.2.4 weiter eingegangen. Ist also ein Live-Bild ausgewählt, so wird kontinuierlich `UpdatePre()` aufgerufen, da das Live-Bild immer wieder ein neues Bild liefert und eine Nachricht gesendet wird.

Empfängt der Dialog eine Nachricht, dass eine Ansicht geschlossen wurde, wird zuerst untersucht, ob es sich bei dieser Ansicht um das Live-Bild handelt. Wird diese geschlossen, wird die Variable, die registriert, ob ein Live-Bild angehalten wurde in jedem Falle auf `false` gesetzt. Handelt es sich um das Vorschaufenster, so soll auch der Dialog mittels `UnregAndDestroyDlg()` geschlossen werden. Wird jedoch eine andere Ansicht geschlossen, so wird lediglich der Eintrag der Quelle im Kombinationsfeld entfernt. Sollte allerdings nach dem Schließen dieser Ansicht keine andere Quelle mehr offen sein, so wird der Dialog wieder mit `UnregAndDestroyDlg()` beendet und das Vorschaufenster ebenfalls geschlossen. Das Aktivitätsdiagramm in Abbildung 3.4 zeigt die Vorgehensweise beim Schließen von Ansichten.

Für das Deaktivieren beziehungsweise das Aktivieren von Ansichten (siehe Abbildung 3.5) wird ebenfalls eine Nachricht an das Applikationsfenster gesendet und eine bestimmte

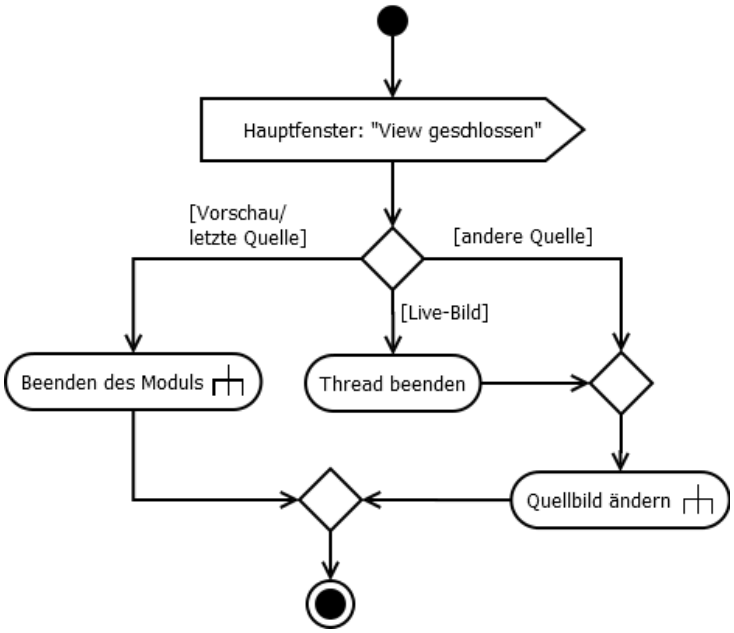

Abbildung 3.4: Aktivitätsdiagramm „Schließen einer Ansicht“

Methode ausgeführt. Diese Methode unterscheidet mittels einer if-Abfrage ob eine Deaktivierung oder eine Aktivierung stattgefunden hat und führt dann entsprechenden Code aus.

Hinsichtlich der Deaktivierung von Ansichten muss nur der Fall eines deaktivierten Live-Bildes berücksichtigt werden. Bei diesem Ereignis wird der ständige Bildeinzug aktiviert und es wird auf die Beendigung des Threads gewartet.

Stellt die Funktion fest, dass eine Aktivierung einer Ansicht stattgefunden hat, so wird zunächst untersucht, ob es sich bei dieser um ein Live-Bild handelt, welches kontinuierlich ein neues Bild empfängt. Trifft dies zu, wird bis auf weiteres die ID des Live-Bildes gespeichert, um in späteren Aktionen auf diese zugreifen zu können. Außerdem wird der Bildeinzug angehalten und ein Thread dafür gestartet um die Laufzeit zu verbessern. Als nächstes muss die aktivierte Ansicht im Kombinationsfeld gefunden werden und der entsprechende Eintrag ausgewählt werden. Wird jedoch das Vorschaufenster aktiviert, so soll dieses ignoriert werden und die nächste Ansicht aktiviert werden. Ist die nächste Ansicht das Live-Bild, muss der Bildeinzug gegebenenfalls wieder angehalten werden und ein Thread gestartet werden. Unabhängig davon, ob die nächste Ansicht ein Live-Bild ist, wird bei Aktivierung des Vorschaufensters in jedem Fall die nächste Ansicht aktiviert. Exisitiert wiederum keine nächste Ansicht nach dem Vorschaufenster, so wird der Dialog mit `UnregAndDestroyDlg()` beendet. Zu diesem Zeitpunkt steht nun sicher fest, welche Ansicht aktiviert werden soll und jetzt kann im Kombinationsfeld diese Ansicht gesucht und ausgewählt werden. Zum Schluss wird bei aktiviertem Live-Bild ein einzelnes Bild

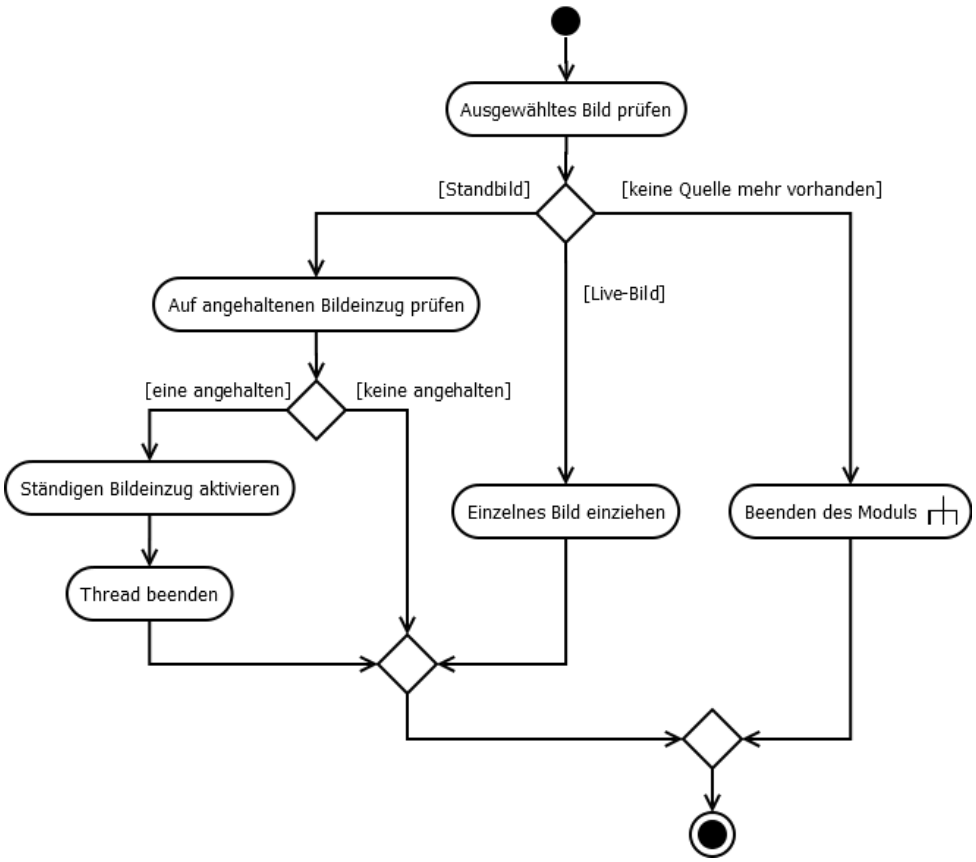

Abbildung 3.5: Aktivitätsdiagramm „Quellbild ändern“

eingezogen um ein Bild bearbeiten zu können, bevor das nächste Bild aufgenommen wird. Wurde statt dem Live-Bild ein anderes aktiviert, dann wird bei angehaltenem Bildeinzug dieser wieder auf kontinuierlich gesetzt. Nachdem schließlich alle möglichen Optionen für die Aktivierung durchgegangen und entsprechend behandelt wurden, wird `UpdatePre()` aufgerufen, um das Vorschaufenster mit dem neuen Bild zu aktualisieren.

3.2.4 Durchführung der Filteroperation

Die Methode `UpdatePre()` kann als eigentlicher Kern der Klasse `CMorphDlg` angesehen werden, da hier nun das Filter auf ein ausgewähltes Quellbild angewandt wird. Bevor die von der MIL bereits zur Verfügung gestellten Methoden benutzt werden können, müssen diese allerdings zuerst in der Klasse `CMilImage` gekapselt werden, damit diese auf ein MilImage-Objekt angewandt werden können.

In `UpdatePre()` wird zunächst für ein aktives Live-Bild der Bildeinzug angehalten. Anschließend werden die benötigten Parameter wie gewünschte Operation, Anzahl der Wiederholungen und Typ des Quellbildes abgefragt, die zuvor über die Benutzeroberfläche gesetzt worden sind. Schließlich wird die gewünschte Operation auf das Bild des Vorschaufensters angewandt, das bis zu diesem Zeitpunkt noch das gleiche Bild wie die Quelle bekommen hatte. Das Vorschaufenster bekommt von nun an jedes mal ein neues Bild, wenn `UpdatePre()` aufgerufen wird. Abbildung 3.6 veranschaulicht die Durchführung der Operationen.

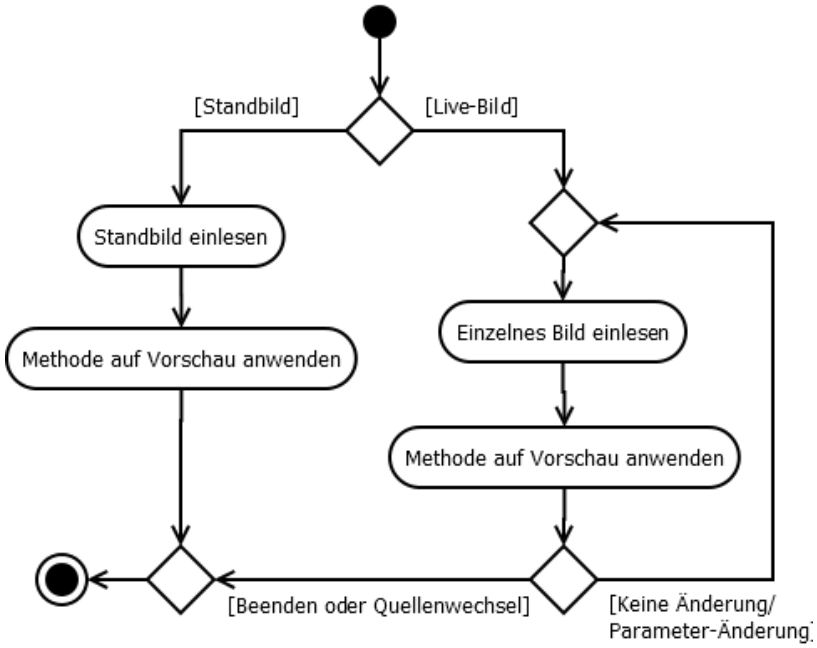

Abbildung 3.6: Aktivitätsdiagramm „Operation durchführen“

Des Weiteren gibt es einige Event-Handling-Funktionen, die bei bestimmten Benutzeraktionen, wie zum Beispiel einem Klick auf „Erosion“ oder einer Auswahl über das Kombinationsfeld ausgeführt werden. Diese Methoden beinhalten größtenteils lediglich weitere Aufrufe von Methoden, die bereits besprochen wurden. `UpdatePre()` wird beispielsweise beim Ändern der Operation, des Bildtyps oder der Wiederholungsanzahl aufgerufen. Hat man das gewünschte Ergebnis erreicht, oder möchte das aktuelle Zwischenergebnis

abspeichern, kann über den Button „Bild erstellen“ eine neue Ansicht mit dem Bild der Vorschau erzeugt werden. Das Bild kann dann abgespeichert werden.

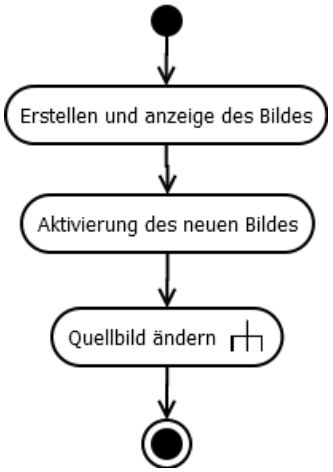

Abbildung 3.7: Aktivitätsdiagramm „Neues Bild erstellen“

4 Anwendungsbeispiel

Dieses Kapitel behandelt den Aufbau und das Layout des Moduls für morphologische Filter. Dabei wird auf die einzelnen Steuerelemente eingegangen und die Steuerung durch den Benutzer beschrieben. Außerdem wird der Einsatz von morphologischen Filtern am Beispiel eines aktuellen Projekt aus der Praxis gezeigt.

4.1 Dialoggestaltung

Abbildung 4.1 zeigt einen Ausschnitt der Benutzeroberfläche von DIPLOM mit dem geöffneten Modul für die morphologischen Filter (2). Zu sehen sind zwei geöffnete Bildquellen (1), die auch anstatt via direktem Anklicken schnell über das Kombinationsfeld des Dialogs ausgewählt werden können. Diese Möglichkeit ist vor allem bei vielen gleichzeitig geöffneten, sich überlappenden Quellen vorteilhaft. Außerdem wird eine leichte und eindeutige Auswahl der vier Operationen und des Bildtyps mit Optionsfeldern realisiert. Die Angabe der Wiederholungsanzahl wird manuell über die Tastatur in ein Textfeld eingegeben, wobei algorithmisch sichergestellt wird, dass nur gültige Werte eingegeben werden können. Zeigt die Vorschau (3) nach getätigten Einstellungen das gewünschte Ergebnis an, so kann über den Button „Bild erstellen“ ein neues Fenster mit dem Ergebnisbild erstellt werden. Dieses kann dann entweder über die DIPLOM-Oberfläche abgespeichert werden, oder es kann direkt zur Weiterverarbeitung verwendet werden. Der Dialog und das Vorschaufenster können entweder über den Beenden-Button oder über das Schließen-Symbol von einem der beiden Fenster beendet werden.

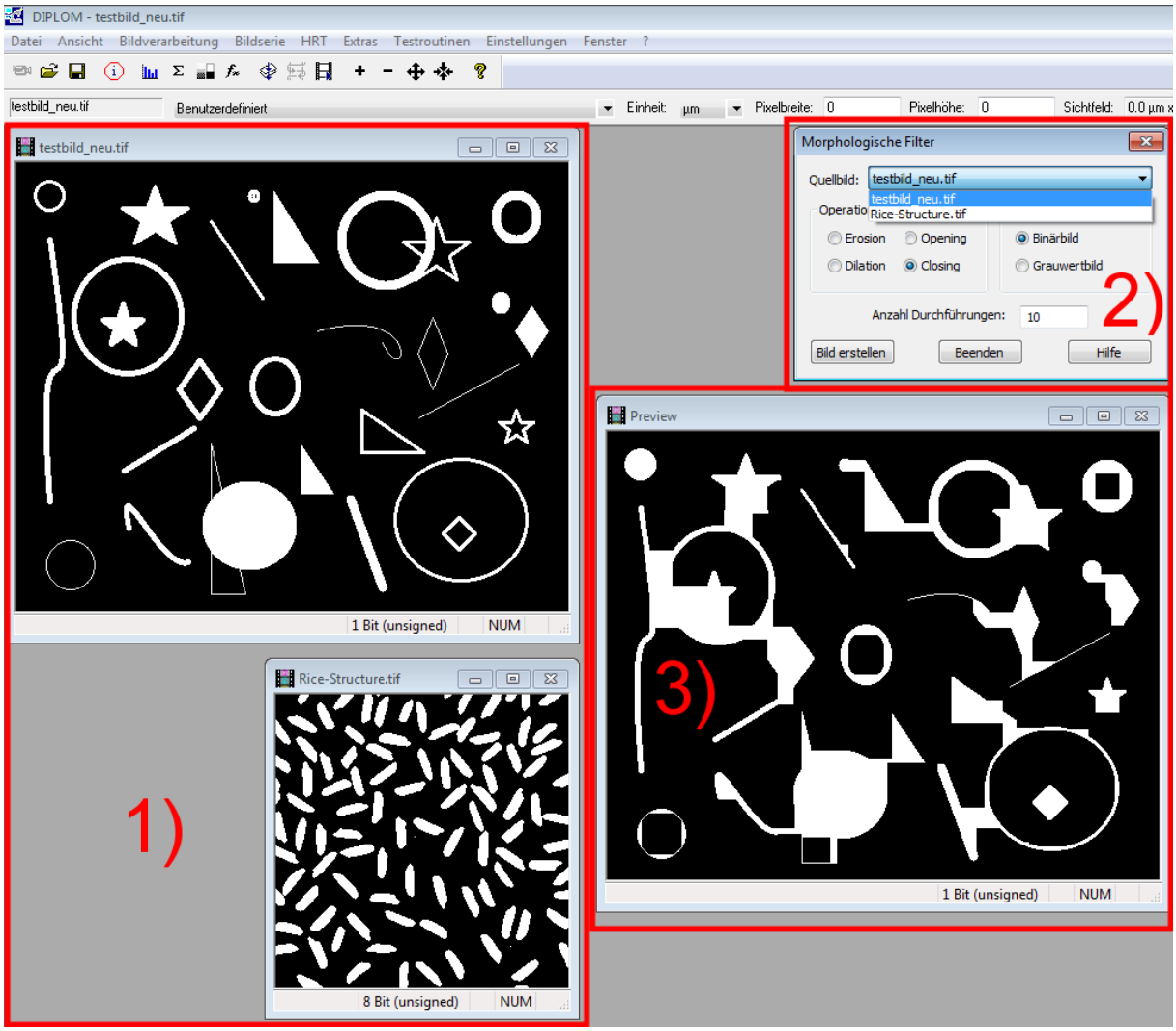

Abbildung 4.1: Layout und Bedienung des Moduls: 1) - Geöffnete Quellbilder; 2) - Dialog zur Bedienung; 3) - Zugehörige Preview als Vorschau

4.2 Anwendung in der Praxis

Das Projekt „Computerkontrollierte Analyse von Radon Detektoren“ (CARD) [6], bei dem bereits die MIL-Methode für die Erosion zum Einsatz kam, wird in Kooperation mit dem Sicherheitsmanagement des KIT (KSM) [7] bearbeitet. Das KSM beschäftigt sich unter anderem mit der Messung der Radon-Konzentration in der Luft. Radon ist ein radioaktives Edelgas und gesundheitsschädigend. Die Radon-Proben liegen auf kleinen Probe-Scheiben zur Untersuchung unter Mikroskopen vor. Um die Konzentration zu bestimmen, müssen die Radon-Krater auf den Proben gezählt werden. Bevor jedoch der Zählalgorithmus angewendet werden kann, müssen die Bilder der Mikroskopaufnahmen zuerst bearbeitet werden, da die einzelnen Radon-Krater auf den unbearbeiteten Bildern teilweise zusammenhängen und damit mehrere Krater zu einem zusammengefasst werden könnten. Jeder einzelne Krater jedoch hat einen helleren Kern, über welchen die einzel-

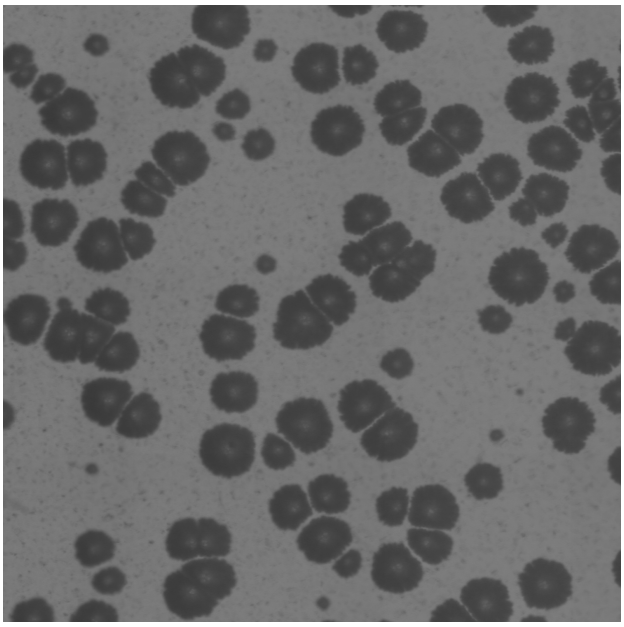

Abbildung 4.2: Ausschnitt einer unbearbeiteten Mikroskopaufnahme einer Radon-Probe

nen Krater unterschieden werden können. Abbildung 4.2 zeigt das unbearbeitete Bild der Mikroskopaufnahme. Die Verarbeitung dieser Aufnahmen wird über DIPLOM realisiert. Dazu wird das Bild zunächst in ein Binärbild umgewandelt und anschließend die scheinbar zusammenhängenden Krater voneinander getrennt (Segmentierung). Danach wird eine Erosion auf das Binärbild angewendet, damit die äußerste Schicht der Strukturen weggenommen wird. Wird das Bild nun wieder in ein Grauwertbild gewandelt, sind die einzelnen Krater voneinander getrennt und durch die Erosion ist der leichte weiße Rand der Krater nicht mehr vorhanden. Jetzt kann der Zählalgorithmus angewendet werden. Abbildung 4.3 zeigt das segmentierte und erodierte Binärbild und das fertig bearbeitete Grauwertbild.

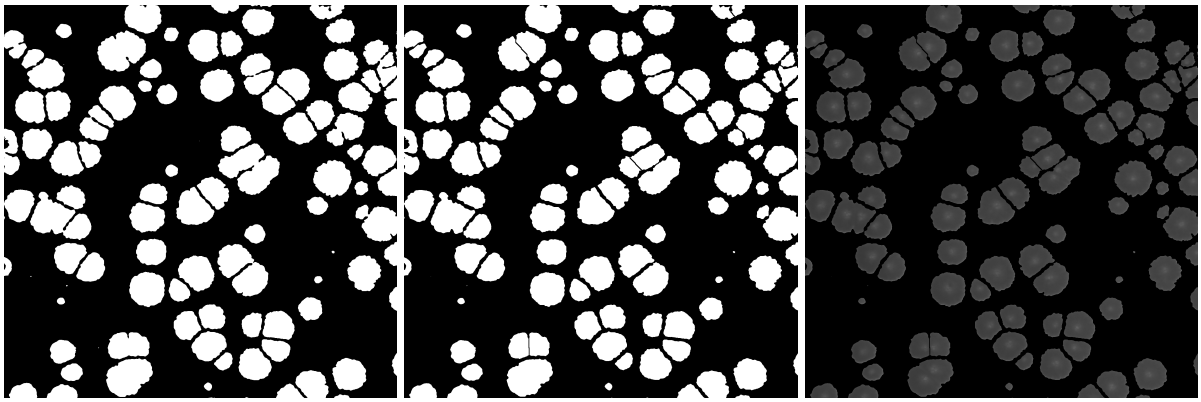

Abbildung 4.3: Binärbild (links),segmentiertes und erodiertes Binärbild (mitte), Ergebnisbild (rechts)

5 Zusammenfassung und Ausblick

Bereits vor der Implementierung des Moduls für die morphologischen Filter wurden im Projekt „Computerkontrollierte Analyse von Radon Detektoren“ (CARD) bereits morphologische Operationen in einem komplexeren Algorithmus eingesetzt. Das neu implementierte Modul kann nun für kleinere Zwischenschritte eines Bildverarbeitungsprozesses herangezogen werden. Jetzt besteht die Möglichkeit, die morphologischen Filter über einen eigenen Dialog zu bedienen und die Filter auf beliebige Bilder anzuwenden. Es ist sogar möglich, die verschiedenen Operationen in Verbindung mit einem Live-Bild einzusetzen. Zusätzlich gibt es ein Vorschaufenster, mit Hilfe dessen man sofort bei Änderung der Parameter das Ergebnis betrachten kann.

Vor der Planung und Realisierung des Projekts war das Darstellen von Grundlagen der Bildverarbeitung und morphologischen Filtern erforderlich. Beim Entwurf und der Implementierung wurde großen Wert auf Übersicht der Benutzeroberfläche gelegt, um eine möglichst einfache und schnelle Bedienung durch den Benutzer zu ermöglichen. Außerdem war eine effiziente Implementierung wichtig, da die Anwendung der morphologischen Filter mit großer Wiederholungsanzahl sehr rechenaufwendig ist.

Während der Testphase des Moduls hat sich herausgestellt, dass auch bei Verwendung von Farbbildern ein scheinbar korrektes Ergebnis erzeugt wird, wenn „Grauwertbild“ als Bildtyp angegeben wird. Da allerdings in der Dokumentation der MIL keine Angaben über die Anwendung der morphologischen Operationen auf Farbbilder gefunden werden konnten, sind Recherchen eingeleitet worden, die derzeit noch anhalten.

Als Erweiterung des Moduls ist eine Implementierung denkbar, mit der man den gewünschten Filter auf ein ROI (Region of Interest) anwenden kann. Diese Zusatzfunktion wäre hilfreich, wenn nur bestimmte Bereiche eines Bildes bearbeitet werden sollen, während andere Bereiche unberührt bleiben sollen.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Webseite des DIPLOM-Systems; 2012; http://www.iai.kit.edu/www-extern/index.php?id=408 (zuletzt besucht: 25.09.12)

[2] Webseite der freien Versionsverwaltung Subversion; 2012; http://subversion.apache.org/ (zuletzt besucht: 27.09.12)

[3] Wiktionary: Morphologie; 2012; http://de.wiktionary.org/wiki/Morphologie (zuletzt besucht: 06.09.12)

[4] Wilhelm Burger / Mark James Burge; Digitale Bildverarbeitung. Eine Einführung mit Java und ImageJ; 2. Auflage; Springer Verlag Berlin Heidelberg; 2006

[5] Bernd Jähne; Digitale Bildverarbeitung; Springer Verlag Berlin Heidelberg; 2005

[6] Klaus-Martin Reichert; Software zur automatischen Messung der Radonexposition; Bachelorarbeit an der Dualen Hochschule Karlsruhe; 2010

[7] Webseite des KIT-Sicherheitsmanagements; 2012; http://www.ksm.kit.edu/ (zuletzt besucht: 26.09.12)