Masterarbeit

Filterbänke für Bildzerlegung

eingereicht bei: **Prof. Dr. Tomas Sauer**

am **11.12.2017**

Name, Vorname:	Wasmeier, Ewald
Matrikelnummer:	63660
Anschrift:	In der Point 1
	94554 Moos
Semesterzahl:	12
Studiengang:	Master Informatik
Telefonnummer:	0160/91835874
E-Mail:	wasmeier@fim.uni-passau.de

Inhaltsverzeichnis

1 Einleitung

Die Zahl automatisierter und nicht von Menschenhand gesteuerter Gegenstände im Internet nimmt drastisch zu. Sie bilden zusammen das Internet-of-Things (IoT) und bestehen vorwiegend aus Sensoren und Kameras. Diese erzeugen eine gigantische Flut an Daten, welche maschinell verarbeitet werden müssen. Die effektive Abarbeitung von Bilddaten ist hierbei eine der wichtigsten Aufgaben. Ganze Serverfarmen werten Tag und Nacht Kamerabilder aus und speichern diese. Trotz fortschrittlichster Technologie mit immer leistungsstärkeren Festplatten ist hier eine hohe Kompression der Bilder mit geringem Qualitätsverlust unabdingbar. Auch das Erkennen von signifikanten Merkmalen im Bild, wie z.B. Kanten, ist etwa in der Fahndung und Terrorbekämpfung wie auch in den sozialen Medien nicht mehr wegzudenken. In dieser Arbeit werden verschiedene Filterbänke vorgestellt, mit denen sowohl eine effektive Bildkompression als auch das Auffinden von Kanten im Bild möglich ist. Die mitgelieferte Software lässt sehr viel Spielraum bei der Einstellung der Parameter, wodurch sie an beliebige Anwendungsbeispiele angepasst werden kann.
Im Abschnitt 2.1 über die Funktionsdokumentation werden sämtliche Matlab-Funktionen genau erklärt, dokumentiert und mit Beispielprozeduren versehen. Dabei sind die wichtigsten Funktionen in den Bereichen Anzeigefunktionen (siehe 2.1.1) und Hauptfunktionen (siehe 2.1.2) verzeichnet. Die bereits implementierten Filter sind im Abschnitt Filterdatenbanken aufgelistet. Der Abschnitt 2.3 zur Behandlung des Verschiebungsvektors gibt Aufschluss über die Funktionsweise der Filterbänke und die Speicherung der Positionsdaten des $(0, 0)$-Elements in den einzelnen Bildmatrizen. Anschließend werden die Ergebnisse der unterschiedlichen Filterbänke diskutiert und untersucht. Das Ende des Hauptteils enthält eine kurze Anleitung zur Verwendung des Programms.

2 Hauptteil

Ausgangslage jeder Zerlegung bilden eine Bilddatei und eine Filterbank. Werden mehrere Filterbänke simultan verwendet, so ist darauf zu achten, dass deren Dilatationsmatrizen gemeinsam expandieren (engl. jointly expanding) [1]. Jede Filterbank besteht aus einer Dilatationsmatrix D, einer Tiefpasszerlegungsfiltermatrix $L_{DecFilter}$ und theoretisch beliebig vielen Hochpasszerlegungsfiltermatrizen $H1_{DecFilter}$ bis $Hn_{DecFilter}$ für $n \in \mathbb{N}$. Die Anzahl der Hochpassfiltermatrizen hängt von der Dilatationsmatrix ab. Dazu kommen die zugehörigen Rekonstruktionsfiltermatrizen sowie die den einzelnen Filtermatrizen entsprechenden Verschiebungsvektoren.
Die Bilddatei wird in jedem Zerlegungsschritt mit den Zerlegungsfiltermatrizen gefaltet. Man erhält einen Tiefpassbildanteil und entsprechend viele Hochpassbildanteile. Diese werden nun einem Downsampling und einer Transformation (Sampling, Drehung und Scherung entsprechend der Dilatationsmatrix) unterzogen.
Die maximal mögliche Zerlegungstiefe eines Bildes hängt von dessen Größe ab. Führt ein Downsampling in irgendeinem Zerlegungsschritt dazu, dass statt einer Teilbildmatrix nur noch ein einzelner Wert übrig bleibt, so ist die maximale Tiefe erreicht.
Anschließend kann unter Zuhilfenahme der Rekonstruktionsfiltermatrizen, einem Upsamplingprozess und der Inversion der Transformationen in den einzelnen Schritten das ursprüngliche Bild vollständig wiederhergestellt werden. Wie die Position des $0,0$-Elements einer jeden Teilbildmatrix gespeichert und bei der Rekonstruktion verwendet wird, kann im Abschnitt 2.3 nachgelesen werden.
Während der Rekonstruktion kann Thresholding angewendet werden. Auch die Rekonstruktion des Originalbildes allein aus den Hochpassinformationen zur Kantenerkennung ist möglich.
Die einzelnen Zerlegungs- und Rekonstruktionsschritte werden detailliert im Abschnitt 2.3 vorgefunden.[7]

2.1 Funktionsdokumentation

2.1.1 Anzeigefunktionen

Die Anzeigefunktionen dienen zur Visualisierung und dem Vergleich von Ergebnissen. Verschiedene Filterpfade oder Thresholdwerte können miteinander verglichen werden. Vor allem bei größeren Bildern kann die Ausführung der Funktionen länger dauern, da viele verschiedene Rekonstruktionspfade berechnet werden müssen.

CompareEdgeLevels.m

```
function imageArray = CompareEdgeLevels(decArray, recCellIndex)
```

Die Funktion `CompareEdgeLevels` erhält als Parameter ein Cell-Array `decArray` mit Zerlegungsinformationen. Zudem wird der Index `recCellIndex` der Zelle im Cell-Array übergeben, von welcher aus die Rekonstruktion gestartet werden soll. Mit der Angabe der untersten Rekonstruktionszelle wird ein eindeutiger Rekonstruktionspfad festgelegt. Die Funktion startet in jedem Tiefenlevel aus der entsprechenden Zelle des Rekonstruktionspfades eine eigenständige Rekonstruktion mit dem optionalen Parameter `'highpassOnly'`. Anschließend werden die Rekonstruktionsergebnisse aus den verschiedenen Tiefenleveln zum besseren Vergleich nebeneinander dargestellt und als Cell-Array zurückgegeben. Bei Bildern mit mehreren Layern werden die Hochpassergebnisse der einzelnen Layer am Ende addiert und durch die Anzahl der Layer geteilt.

Beispiel 2.1.1: Rekonstruktion der Hochpassinformationen aus verschiedenen Tiefenleveln
Folgende Matlab-Routine lädt das Bild `'serrano.png'`[5] und zerlegt es anschließend gemäß der Filterbank `'MO1'` bis zu einer Tiefe von 4. Das Bild `'peppersSmall.png'`[5] wird unter Verwendung des Filters `'M11'` bis zu einer Tiefe von 5 zerlegt. Die Bildränder werden erweitert, sodass dort Faltungsartefakte vermieden werden. Anschließend wird die Funktion `CompareEdgeLevels` auf beide Zerlegungen angewandt. Die Ergebnisse sind in Abbildung 2.1 und 2.2 zu sehen.

```
» image = imread('Testbilder/serrano.png');
» decArray = WaveletDecomposition(image, 'MO1', 4);
» imageArray = CompareEdgeLevels(decArray, [5, 1]);

» image = imread('Testbilder/peppersSmall.png');
» decArray = WaveletDecomposition(image, 'M11', 5, 'expandEdges', 25);
» imageArray = CompareEdgeLevels(decArray, [6, 1])

imageArray =

Columns 1 through 3

[123x94 uint8]     [123x94 uint8]     [123x94 uint8]

Columns 4 through 6

[123x94 uint8]     [123x94 uint8]     [123x94 uint8]
```

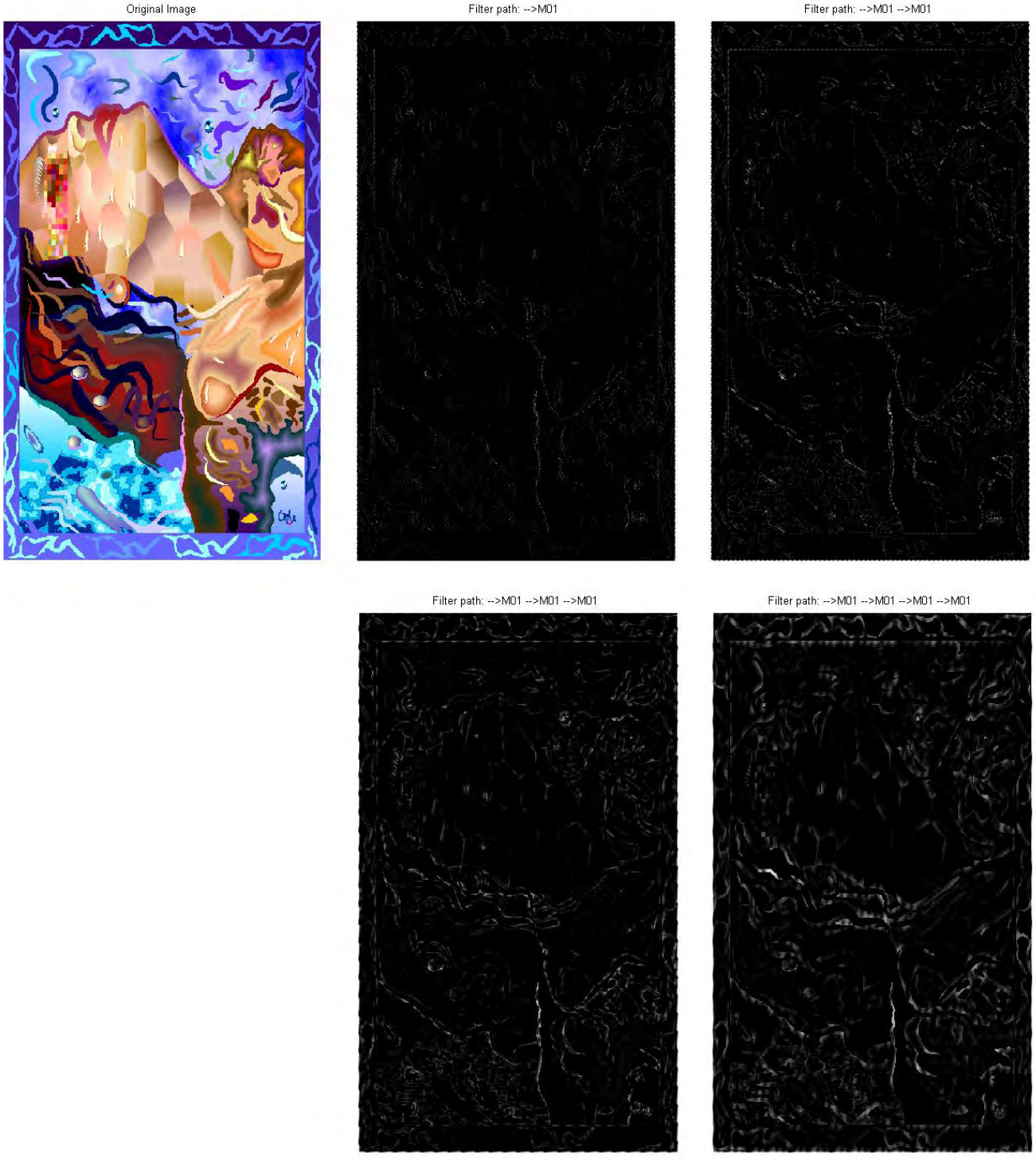

Abbildung 2.1: Das Bild `'serrano.png'`[5] wurde mit dem Filter `'M01'` bis zu einer Tiefe von 4 zerlegt. Dargestellt sind die Hochpassrekonstruktionen aus den unterschiedlichen Tiefenleveln.

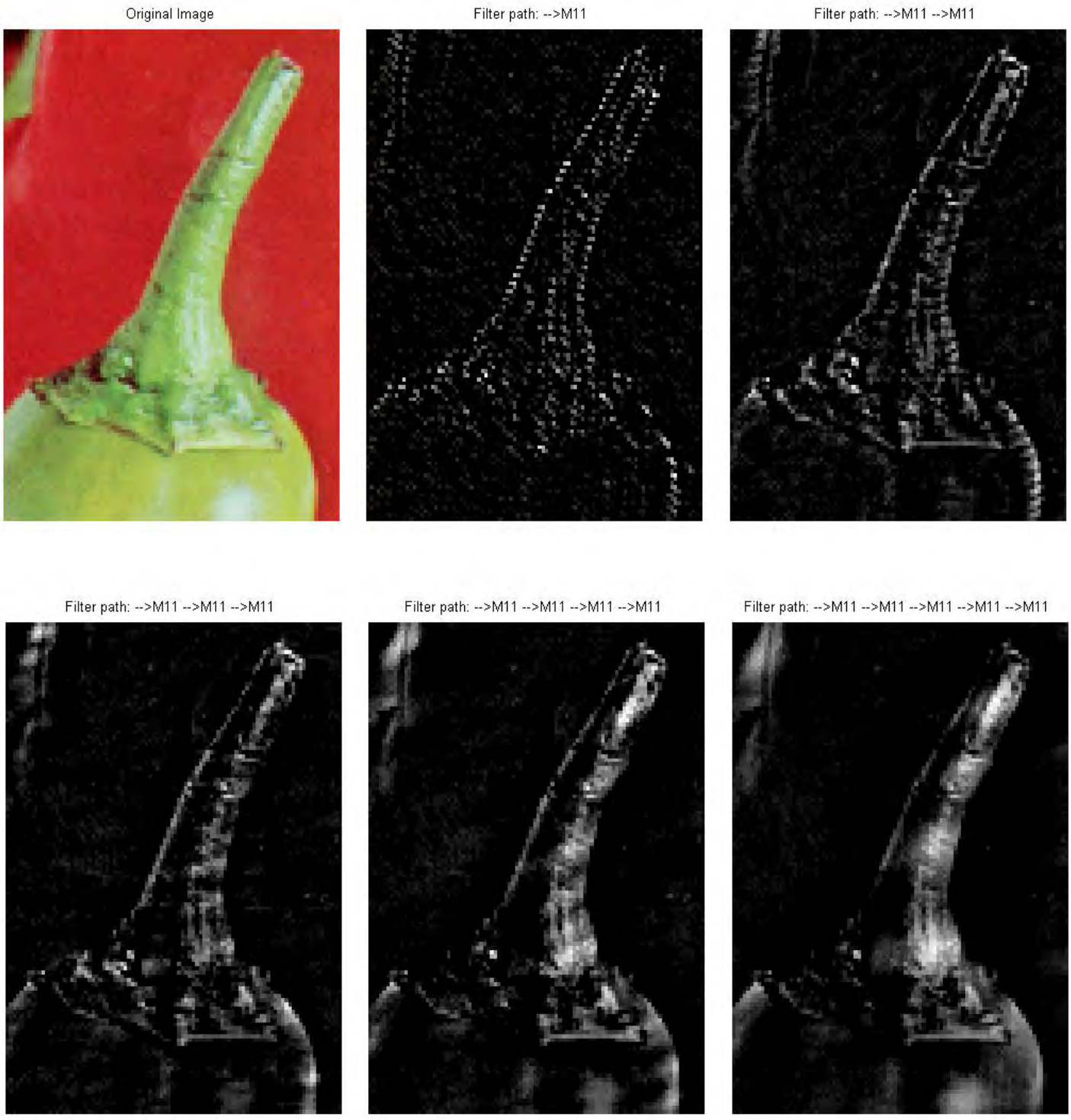

Abbildung 2.2: Das Bild `'peppersSmall.png'` wurde mit dem Filter `'M11'` bis zu einer Tiefe von 5 zerlegt. Dargestellt sind die Hochpassrekonstruktionen aus den unterschiedlichen Tiefenleveln.

CompareEdgePathes.m

```
function imageArray = CompareEdgePathes(decArray)
```

Die Funktion `CompareEdgePathes` vergleicht die aus verschiedenen Dekompositionspfaden gewonnenen Hochpassinformationen. Je nach gewähltem Pfad erkennt man die Kanten im Bild besser oder schlechter. Die Funktion erhält eine Cell-Array `decArray` mit Dekompositionsinformationen und startet für jede Zelle im untersten Zerlegungslevel eine eigenständige Rekonstruktion mit dem Schlüsselwort `'highpassOnly'`. Die verschiedenen Ergebnisse werden schließlich grafisch gegenübergestellt und als Cell-Array zurückgegeben. Besteht das Originalbild aus mehreren Layern, so werden die Kanteninformationen der einzelnen Layer nach der Rekonstruktion addiert und durch die Anzahl der Layer geteilt.

Beispiel 2.1.2: Rekonstruktion der Hochpassinformationen aus verschiedenen Filterpfaden
Folgende Matlab-Routine lädt das Bild `'pool.tif'`[5] und zerlegt es anschließend gemäß der Filterbänke `'E00'` und `'E01'` bis zu einer Tiefe von 2. Anschließend wird die Funktion `CompareEdgePathes` auf das Cell-Array mit den Dekompositionsinformationen angewandt. Das Ergebnis wird in Abbildung 2.1 dargestellt.

```
» image = imread('Testbilder/pool.tif');
» decArray = WaveletDecomposition(image, ['E00','E01'], 2);
» imageArray = CompareEdgePathes(decArray)

imageArray =

Columns 1 through 3

[383x510 uint8]     [383x510 uint8]     [383x510 uint8]

Columns 4 through 6

[383x510 uint8]     [383x510 uint8]     [383x510 uint8]
```

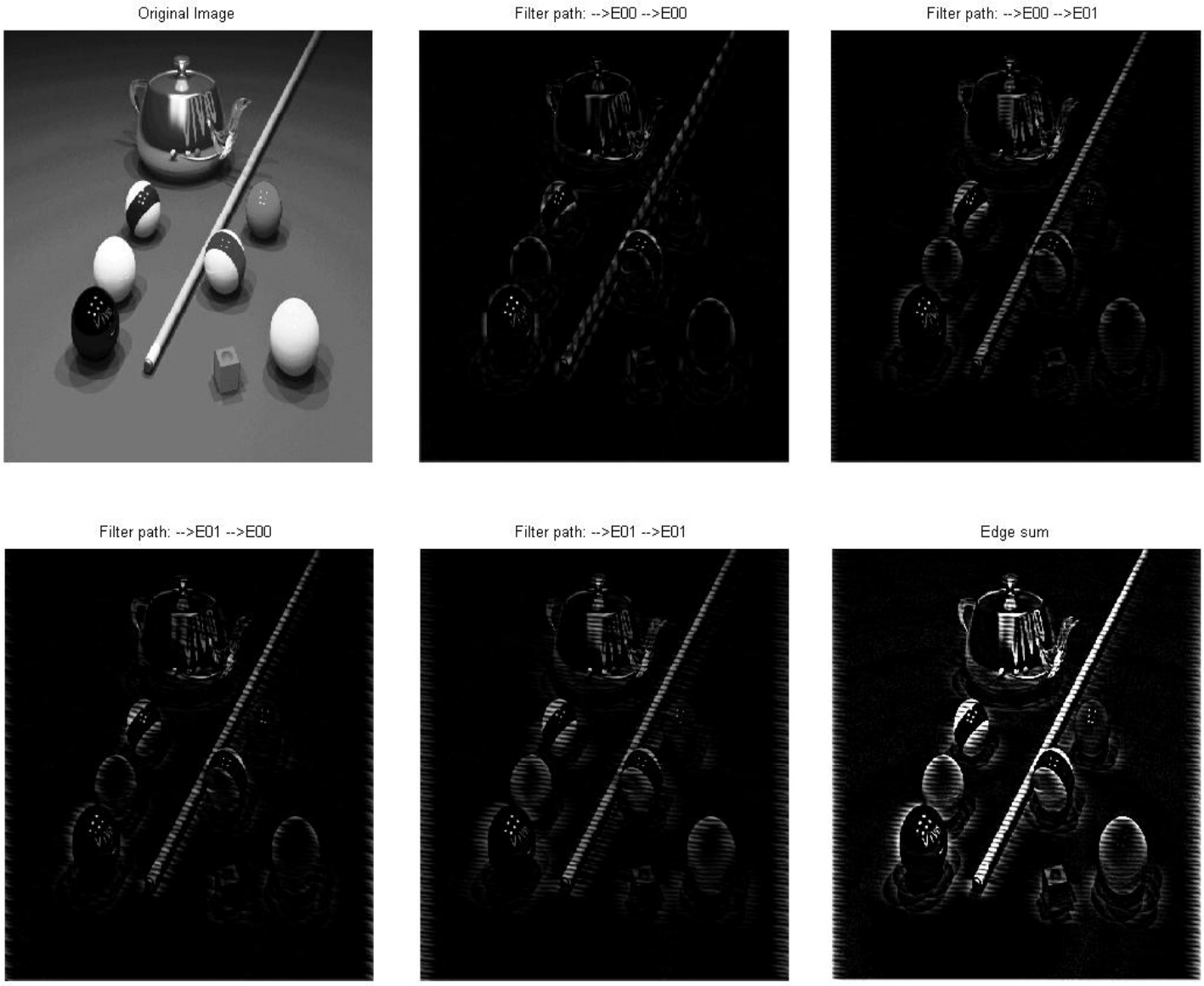

Abbildung 2.3: Das Bild `'pool.tif'`[5] wurde mit den Filterbänken `'E00'` und `'E01'` bis zu einer Tiefe von 2 zerlegt. Dargestellt sind die Hochpassrekonstruktionen aus den einzelnen Dekompositionspfaden sowie deren Gesamtsumme.
Beim Zerlegungsschritt wurde nicht der optionale Parameter `'expandEdges'` verwendet. Man erkennt an den Bildrändern relativ große, unerwünschte Artefakte, da die einzelnen Filtermatrizen der Filterbänke `'E00'` und `'E01'` sehr breit sind. Die Erweiterung der Bildränder des Originalbildes um mindestens $25px$ (Abk. für 25 Pixel) würde hier Abhilfe schaffen.

CompareThresholdLevels.m

```
function imageArray = CompareThresholdLevels(decArray, recCellIndex, ...
     threshold, varargin)
```

Die Funktion `CompareThresholdLevels` vergleicht die Speicherplatzersparnis bei der Rekonstruktion aus verschiedenen Tiefenleveln. Dazu erhält sie ein Cell-Array `decArray` mit Zerlegungsinformationen und den Index `recCellIndex` der Zelle, von welcher aus der Rekonstruktionsprozess gestartet werden soll. Wie bereits vorangehend beschrieben, ist der Rekonstruktionspfad durch die Angabe eines Zellindex eindeutig bestimmt. Nun wird jeweils für die entsprechende Zelle des Rekonstruktionspfades eines Tiefenlevels eine eigenständige Rekonstruktion mit dem optionalen Parameter `'threshold'` gefolgt von dem angegebenen Schwellwert `threshold` gestartet. Die Ergebnisse werden zum Vergleich grafisch dargestellt und als Cell-Array zurückgegeben. Soll die Abklingrate der Hochpasskoeffizienten berücksichtigt werden, so fügt man als zusätzlichen Parameter beim Funktionsaufruf `'decay'` hinzu. Auch der Parameter `'uint8'` wird erkannt, die Bildpixelwerte werden vor der Rückgabe von den intern verwendeten Fließkommazahlen in Ganzzahlen zwischen 0 und 255 umgewandelt.

Beispiel 2.1.3: Threshold-Rekonstruktion aus verschiedenen Tiefen
Folgende Matlab-Routine lädt das Bild `'tulips.png'`[5] und zerlegt es anschließend unter Verwendung der Filterbank `'M11'` bis zu einer Tiefe von 5. Anschließend wird die Funktion `CompareThresholdLevels` mit Schwellwert 20 auf das Cell-Array mit den Dekompositionsinformationen angewandt. Das Ergebnis wird in Abbildung 2.4 dargestellt.
Das Bild `'monarchSmall.png'`[5] wird derselben Prozedur unterzogen, allerdings wird hier die Filterbank `'M01'` und ein Schwellwert von 35 verwendet. Das Ergebnis wird in Abbildung 2.5 dargestellt.

```
» image = imread('Testbilder/tulips.png');
» decArray = WaveletDecomposition(image, 'M11', 5);
» imageArray = CompareThresholdLevels(decArray, [6, 1], 20)

imageArray =

Columns 1 through 3

[512x768x3 uint8]     [512x768x3 uint8]     [512x768x3 uint8]

Columns 4 through 6

[512x768x3 uint8]     [512x768x3 uint8]     [512x768x3 uint8]
```

```
» image = imread('Testbilder/monarchSmall.png');
» decArray = WaveletDecomposition(image, 'M01', 5);
» imageArray = CompareThresholdLevels(decArray, [6, 1], 35);
```

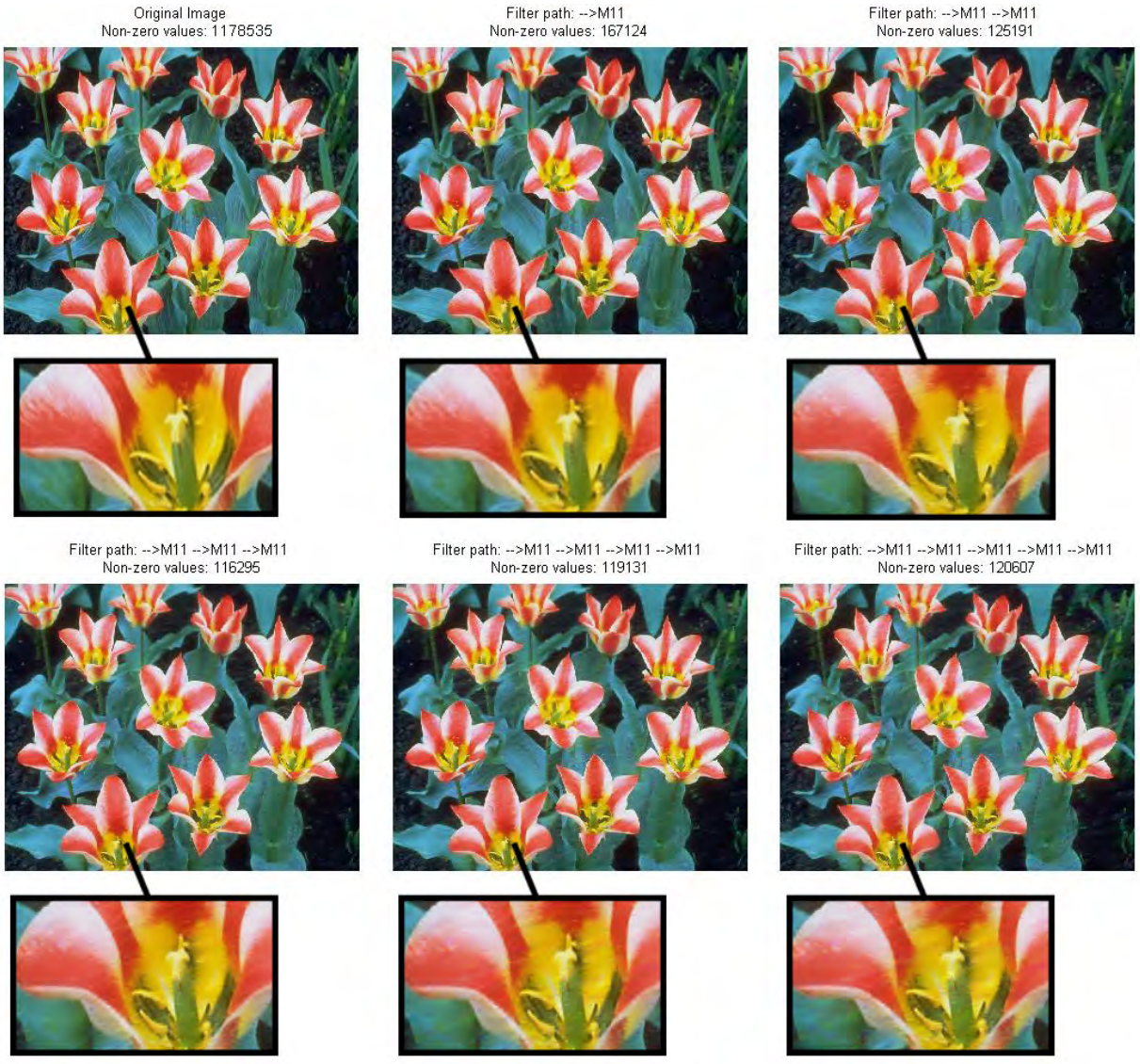

Abbildung 2.4: Das Bild `'tulips.png'`[5] wurde mit der Filterbank `'M11'` bis zu einer Tiefe von 5 zerlegt. Dargestellt sind die Thresholdrekonstruktionen mit Schwellwert 20 aus den einzelnen Tiefenleveln.
Auffällig ist, dass ab einer gewissen Zerlegungstiefe die Anzahl der von Null verschiedenen Werte bei der Threshold-Rekonstruktion wieder steigt.

Abbildung 2.5: Das Bild `'monarchSmall.png'`[5] wurde mit der Filterbank `'M01'` bis zu einer Tiefe von 5 zerlegt. Dargestellt sind die Thresholdrekonstruktionen mit Schwellwert 35 aus den einzelnen Tiefenleveln.

CompareThresholdPathes.m

```
function imageArray = CompareThresholdPathes(decArray, threshold, ...
     varargin)
```

Mit Hilfe der Funktion `CompareThresholdPathes` können die Ergebnisse eines Thresholding bei verschiedenen Rekonstruktionspfaden verglichen werden. Die Funktion erhält ein Cell-Array `decArray` mit Zerlegungsinformationen sowie einen Schwellwert `threshold`. Anschließend wird für jede Zelle in der untersten Zerlegungsschicht eine eigenständige Rekonstruktion mit dem optionalen Parameter `'threshold'` gefolgt vom zu verwendenden Schwellwert `threshold` gestartet. Alternativ kann anstelle des absoluten Schwellwerts auch der gewünschte Prozentsatz, der bei der Kompression erreicht werden soll, angegeben werden (z.B. `'15%'`). Die Ergebnisse werden zum Vergleich neben dem Originalbild grafisch dargestellt und als Cell-Array zurückgegeben. Zusätzlich kann der Parameter `'decay'` angegeben werden, wenn bei den Rekonstruktionen die Abklingrate der Hochpasskoeffizienten berücksichtigt werden soll.

Beispiel 2.1.4: Threshold-Rekonstruktion aus verschiedenen Pfaden
Folgende Matlab-Routine lädt das Bild `'girlSmall.png'`[5] und zerlegt es anschließend unter Verwendung der Filterbänke `'E00'` und `'E01'` bis zu einer Tiefe von 2. Anschließend wird die Funktion `CompareThresholdPathes` mit Schwellwert 30 auf die tiefste Ebene des Cell-Arrays mit den Dekompositionsinformationen angewandt. Das Ergebnis wird in Abbildung 2.6 dargestellt.

```
» image = imread('Testbilder/girlSmall.png');
» decArray = WaveletDecomposition(image, ['E00';'E01'], 5);
» imageArray = CompareThresholdPathes(decArray, 30)

imageArray =

Columns 1 through 3

[263x400x3 uint8]     [263x400x3 uint8]     [263x400x3 uint8]

Columns 4 through 5

[263x400x3 uint8]     [263x400x3 uint8]
```

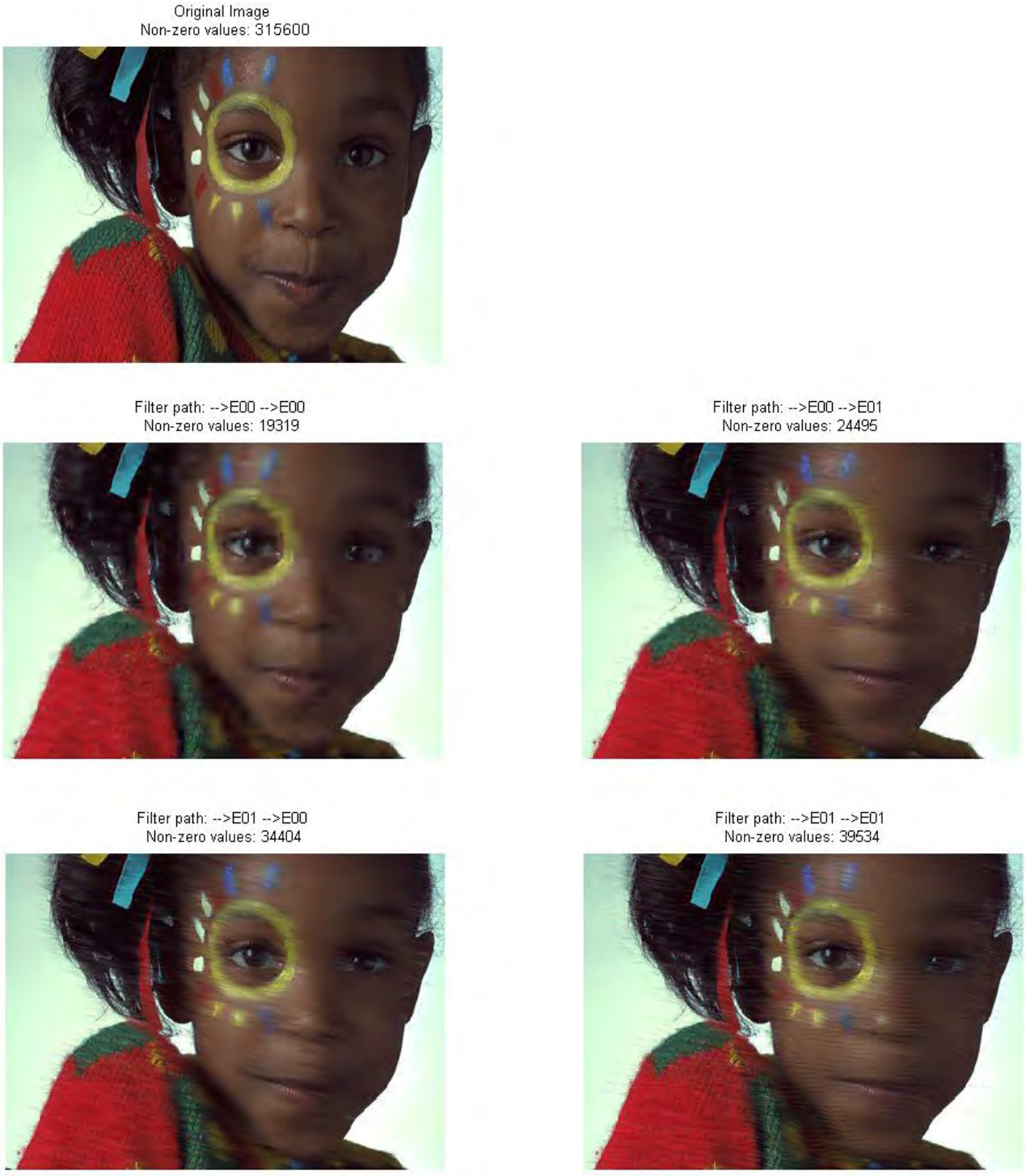

Abbildung 2.6: Das Bild `'girlSmall.png'`[5] wurde mit den Filterbänken `'E00'` und `'E01'` bis zu einer Tiefe von 2 zerlegt. Dargestellt sind die Thresholdrekonstruktionen mit Schwellwert 30 aus den einzelnen Rekonstruktionspfaden.

CompareThresholdValues.m

```
function imageArray = CompareThresholdValues(decArray, recCellIndex, ...
     thresholds, varargin)
```

Die Funktion `CompareThresholdValues` erlaubt es, ein zerlegtes Bild unter Verwendung verschiedener Threshold-Schwellwerte zu rekonstruieren und die Ergebnisse grafisch zu präsentieren. Dazu erhält die Funktion ein Cell-Array `decArray` mit Zerlegungsinformationen eines Bildes sowie dem Index `recCellIndex` der Ausgangszelle im Cell-Array für die Rekonstruktion. Der Zeilenvektor `thresholds` enthält die zu verwendenden Schwellwerte für die einzelnen Thresholdrekonstruktionen. Als Startzelle wird diejenige mit dem angegebenen Index `recCellIndex` verwendet. Alle Ergebnisse werden grafisch dargestellt und als Cell-Array zurückgegeben. Als zusätzlicher Parameter kann `'decay'` angegeben werden. Bei der Rekonstruktion wird dann die Abklingrate der Hochpasskoeffizienten berücksichtigt.

Beispiel 2.1.5: Threshold-Rekonstruktion mit verschiedenen Schwellwerten
Die nachfolgende Matlab-Routine lädt das Bild `'girlSmall.png'`[5] und zerlegt es anschließend unter Verwendung der Filterbank `'MO1'` bis zu einer Tiefe von 3. Anschließend startet die Funktion `CompareThresholdValues` für die Schwellwerte $5, 10, 20, 30$ und 50 jeweils eine eigenständige Threshold-Rekonstruktion. Das Ergebnis wird in Abbildung 2.7 dargestellt.

```
» image = imread('Testbilder/girlSmall.png');
» decArray = WaveletDecomposition(image, 'MO1', 3);
» imageArray = CompareThresholdValues(decArray, [4, 1], ...
[5, 10, 20, 30, 50]);
```

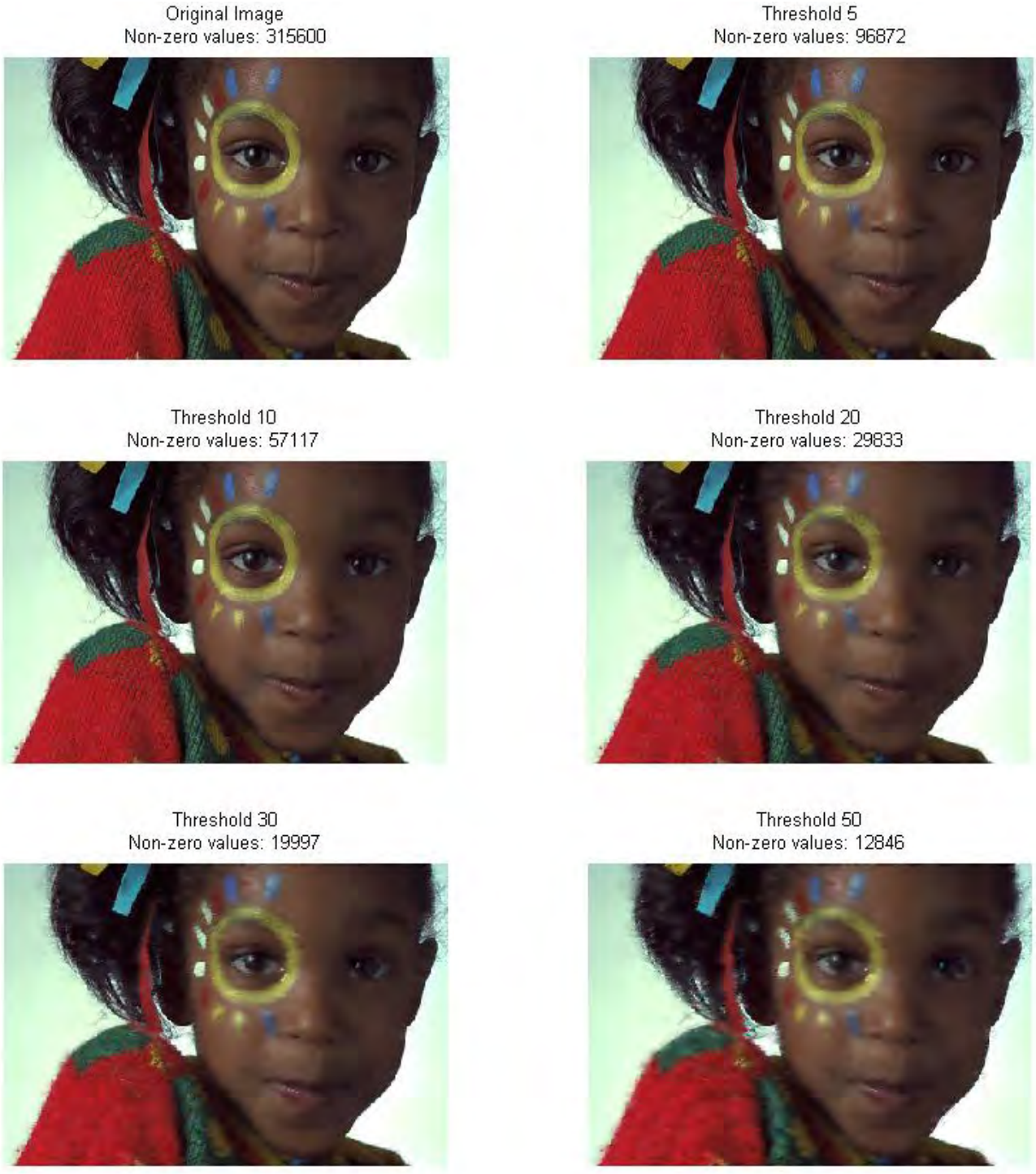

Abbildung 2.7: Das Bild `'girlSmall.png'`[5] wurde mit der Filterbank `'MO1'` bis zu einer Tiefe von 3 zerlegt. Dargestellt sind die Thresholdrekonstruktionen mit den Schwellwerten $5, 10, 20, 30$ und 50 aus dem untersten Level.

ShowDecompositionImages.m

```
function ShowDecompositionImages(decArray)
```

Die Funktion `ShowDecompositionImages.m` zeigt die Inhalte eines Cell-Arrays mit Zerlegungsinformationen an, sodass verschiedene Tiefenlevel miteinander verglichen werden können. Bei einem RGB-Bild wird der Übersicht halber lediglich der erste Layer angezeigt. Der Matlab-Befehl `decArray(:, :, i)` würde den i-ten Layer auswählen.

Beispiel 2.1.6: Anzeigen eines Cell-Arrays mit Zerlegungsinformationen
Die nachfolgende Matlab-Routine lädt das Bild `'watch.tif'`[5] und zerlegt es anschließend unter Verwendung der beiden Filterbänke `'MO1'` und `'M11'` bis zu einer Tiefe von 2. Anschließend werden die Zerlegungsinformationen mit Hilfe der Funktion `ShowDecompositionImages` grafisch dargestellt (siehe Abbildung 2.8).
Weiter wird das Bild `'watch.tif'`[5] mit der Filterbank `'EO1'` bis zu einer Tiefe von 3 zerlegt und das Ergebnis grafisch präsentiert (siehe Abbildung 2.9).

```
» image = imread('Testbilder/watch.tif');
» decArray = WaveletDecomposition(image, ['MO1', 'M11'], 2);
» ShowDecompositionImages(decArray);

» decArray = WaveletDecomposition(image, 'EO1', 3);
» ShowDecompositionImages(decArray);
```

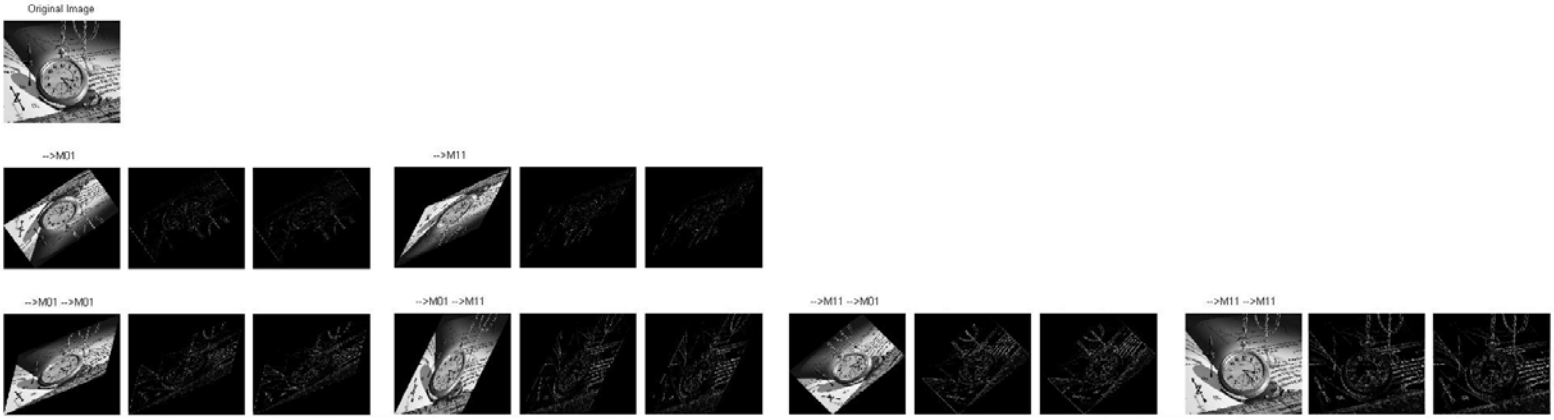

Abbildung 2.8: Das Bild `'watch.tif'`[5] wurde mit den Filterbänken `'MO1'` und `'M11'` bis zu einer Tiefe von 2 zerlegt. Dargestellt ist das gesamte Cell-Array mit Zerlegungsinformationen.

Abbildung 2.9: Das Bild `'watch.tif'`[5] wurde mit der Filterbank `'E01'` bis zu einer Tiefe von 3 zerlegt. Dargestellt ist das gesamte Cell-Array mit Zerlegungsinformationen.

2.1.2 Hauptfunktionen

Die Hauptfunktionen sind für die Koordination des gesamten Ablaufs der Zerlegung sowie Rekonstruktion zuständig. Sie werden vorwiegend vom Benutzer aufgerufen und verwendet.

WaveletDecomposition.m

```
function decArray = WaveletDecomposition(image, filters, maxLevel, varargin)
```

Die Funktion `WaveletDecomposition` zerlegt eine Bildmatrix `image` unter Verwendung einer oder mehrerer Filterbänke, deren Namen im Array `filters` aufgelistet sind. Dabei werden die Bildinformationen in `maxLevel` aufeinanderfolgenden Einzelschritten in Tiefpass- und Hochpassanteile aufgespalten.
Dabei wird ein Cell-Array angelegt, das für jeden Zerlegungsschritt und jede Filterbank eine Zelle zur Speicherung der Informationen bereithält. Abbildung 2.24 zeigt den Inhalt einer jeden Zelle des Cell-Arrays und Abbildung 2.25 das gesamte Cell-Array im nachfolgenden Beispielfall.

Beispiel 2.1.7: Zerlegung eines Bildes
Folgende Matlab-Routine lädt das Bild `'cameraman.tif'[5]` und zerlegt es anschließend gemäß der Filterbänke `'MO1'` und `'M11'` bis zu einer Tiefe von 2.

```
» image = imread('Testbilder/cameraman.tif');
» decArray = WaveletDecomposition(image, ['MO1';'M11'], 2);
```

Der Inhalt des Cell-Arrays kann mit Hilfe der Funktion `ShowDecompositionImages.m` (siehe Abschnitt 2.1.1) anschaulich dargestellt werden. Unter Verwendung weiterer Anzeigefunktionen aus demselben Abschnitt können auf Basis der erfolgten Zerlegung verschiedene Threshold- oder Kantenerkennungseinstellungen miteinander verglichen werden.
Leider werden bei der Rekonstruktion des Bildes allein aus den Hochpassinformationen die Bildränder als Kanten erkannt, was das Ergebnis etwas verfälscht. Dies kann verhindert werden, wenn die Ränder des Originalbildes nach außen hin erweitert werden. Im einfachsten Fall werden die Randpixel über die Ränder hinaus wiederholt. Intern wird dies mit dem Matlab-Befehl `padarray` realisiert.

Beispiel 2.1.8: Erweiterung der Bildränder
Bei der einfachen Matrix `A` werden die Bildränder durch Wiederholung der Randpixel über den Rand hinaus erweitert.

```
» A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]

A =

1     2     3     4
5     6     7     8
9    10    11    12

» padarray(A, [2, 2], 'replicate')
```

```
ans =

1     1     1     2     3     4     4     4
1     1     1     2     3     4     4     4
1     1     1     2     3     4     4     4
5     5     5     6     7     8     8     8
9     9     9    10    11    12    12    12
9     9     9    10    11    12    12    12
9     9     9    10    11    12    12    12
```

Am Ende der anschließenden Rekonstruktion können diese künstlich erzeugten Ränder wieder entfernt werden und die rekonstruierten Kanteninformationen besitzen keine verfälschten Werte an den Bildrändern.

Beispiel 2.1.9: Zerlegung eines Bildes mit erweiterten Rändern
Folgende Matlab-Routine lädt das Bild `'cat.png'`[5], erweitert dessen Ränder in jede Richtung um $40px$ durch Wiederholen der äußersten Pixel und zerlegt es anschließend gemäß der Filterbank `'E01'` bis zu einer Tiefe von 3. Abbildung 2.10 zeigt das Bild `'cat.png'`[5] mit erweiterten Rändern vor der Zerlegung an. Die unerwünschten Artefakte an den Bildrändern treten bei der Rekonstruktion allein aus den Hochpassinformationen zwar immer noch an den äußeren Bildrändern auf, diese enthalten allerdings keine relevanten Bildinformationen und werden bei der Rekonstruktion am Ende wieder entfernt. Der Prozess wird in Abbildung 2.11 nochmals verdeutlicht.

```
» image = imread('Testbilder/cat.png');
» decArray = WaveletDecomposition(image, 'E01', 3, 'expandEdges', 40);
```

Abbildung 2.10: Beim Bild `'cat.png'`[5] wurden die Ränder durch Wiederholen der äußersten Pixel nach außen hin in jede Richtung zur Veranschaulichung um 40px erweitert. Je nach verwendetem Filter reichen 3px bis 20px jedoch vollkommen aus, um die Bildränder frei von Artefakten zu bekommen.

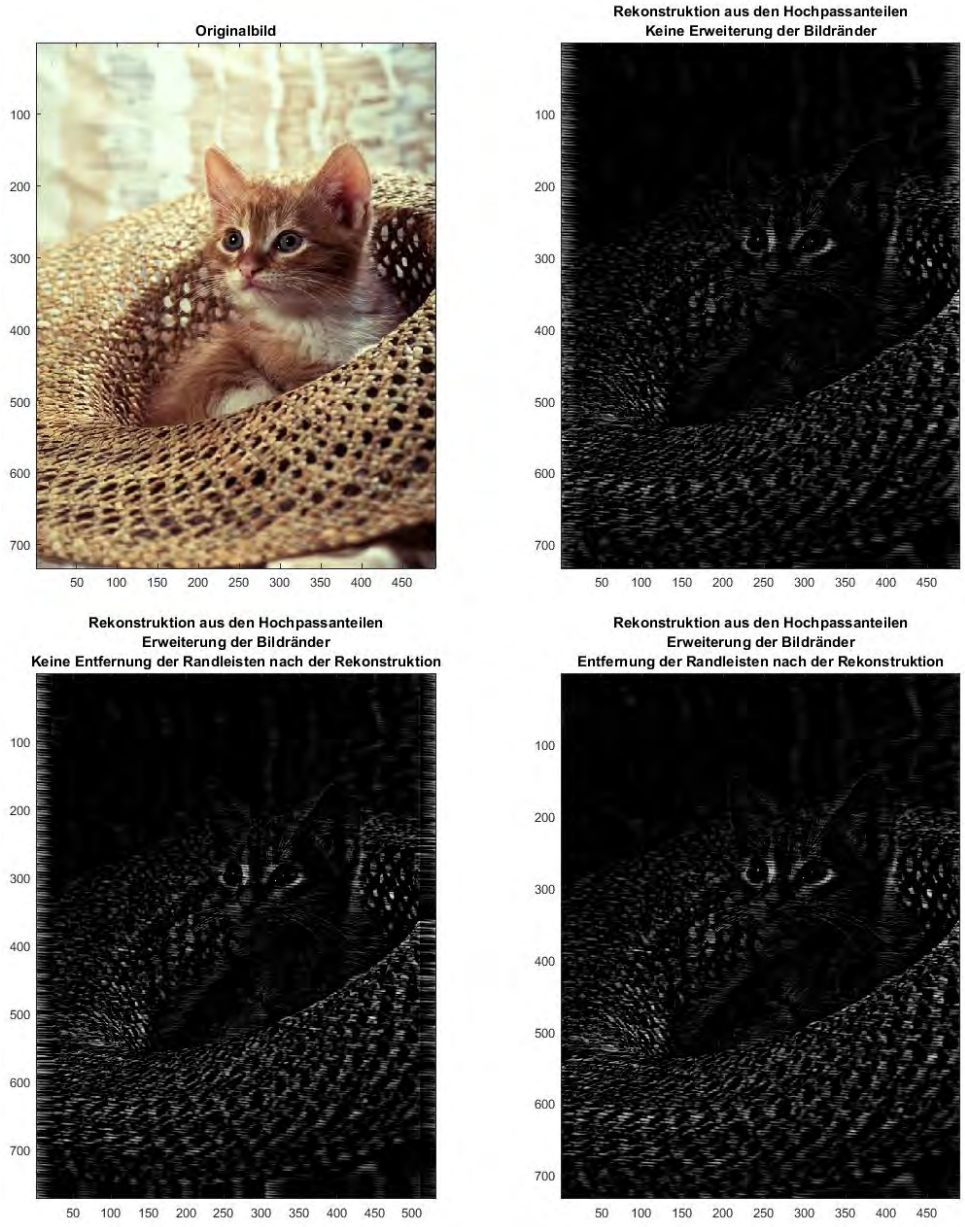

Abbildung 2.11: Der Unterschied zwischen der Standardrekonstruktion ohne Erweiterung der Ränder (rechts oben) und derjenigen mit erweiterten Rändern (rechts unten) ist deutlich erkennbar. Durch die Erweiterung der Bildränder durch Wiederholen der äußersten Pixel verschwinden die unerwünschten Kantenartefakte an den Rändern fast vollständig. Links oben ist zum Vergleich noch das Originalbild dargestellt. Rechts unten wurde das Bild mit erweiterten Rändern zerlegt und rekonstruiert, die hinzugefügten Ränder wurden jedoch nach der Rekonstruktion beibehalten. Man erkennt deutlich, dass sich die Randartefakte bei der passenden Größenwahl der erweiterten Ränder fast ausschließlich in den hinzugefügten Randleisten befinden.

WaveletDecomposition_1Level.m

```
function [lowpassData, highpassData] = ...
     WaveletDecomposition_1Level(originalLowpassData, filter)
```

Die Funktion `WaveletDecomposition_1Level` erhält als Übergabeparameter ein Cell-Array `originalLowpassData` mit dem zu zerlegenden Bild und den zugehörigen Verwaltungsinformationen wie etwa der Position des `(0,0)`-Elements. Zudem wird der Name `filter` der zu verwendenden Filterbank übergeben. Es wird eine 1-Level-Dekomposition durchgeführt, d.h. das ursprüngliche Bild wird in einem Schritt in Tiefpass- und Hochpassanteile zerlegt. Ist das Übergabebild bereits Tiefpassanteil eines Bildes, so wird dieser erneut zerlegt und weitere Hochpassinformationen abgespalten. Als Rückgabewert erhält man zwei Cell-Arrays, die einerseits den Tiefpassanteil `lowpassData` und andererseits den Hochpassanteil `highpassData` samt aller Verwaltungsinformationen über das Bild enthalten.

Beispiel 2.1.10: 1-Level Zerlegung eines Bildes
Die nachfolgende Matlab-Routine lädt das Bild `'mountain.png'`[5]. Anschließend wird das Cell-Array `originalLowpassData` vorbereitet. Da es sich um ein unzerlegtes Originalbild handelt, ist das `(0,0)`-Element das linkeste, unterste und es fand noch kein Austausch des `(0,0)`-Elements aufgrund eines Downsamplings statt. Das erzeugte Cell-Array wird nun an die Funktion `WaveletDecomposition_1Level` zusammen mit dem Namen `'E01'` der zu verwendenden Filterbank (siehe Abschnitt 2.2.4) übergeben. Nun kann ein Zerlegungsschritt durchgeführt werden. Das Ergebnis der Zerlegung wird in Abbildung 2.12 visualisiert.

```
» image = imread('Testbilder/mountain.png');
» originalLowpassData = cell(1, 3);
» originalLowpassData{1} = image;
» originalLowpassData{2} = [0, 0];
» originalLowpassData{3} = [];
» [lowpassData, highpassData] =
            WaveletDecomposition_1Level(originalLowpassData, 'E01');
```

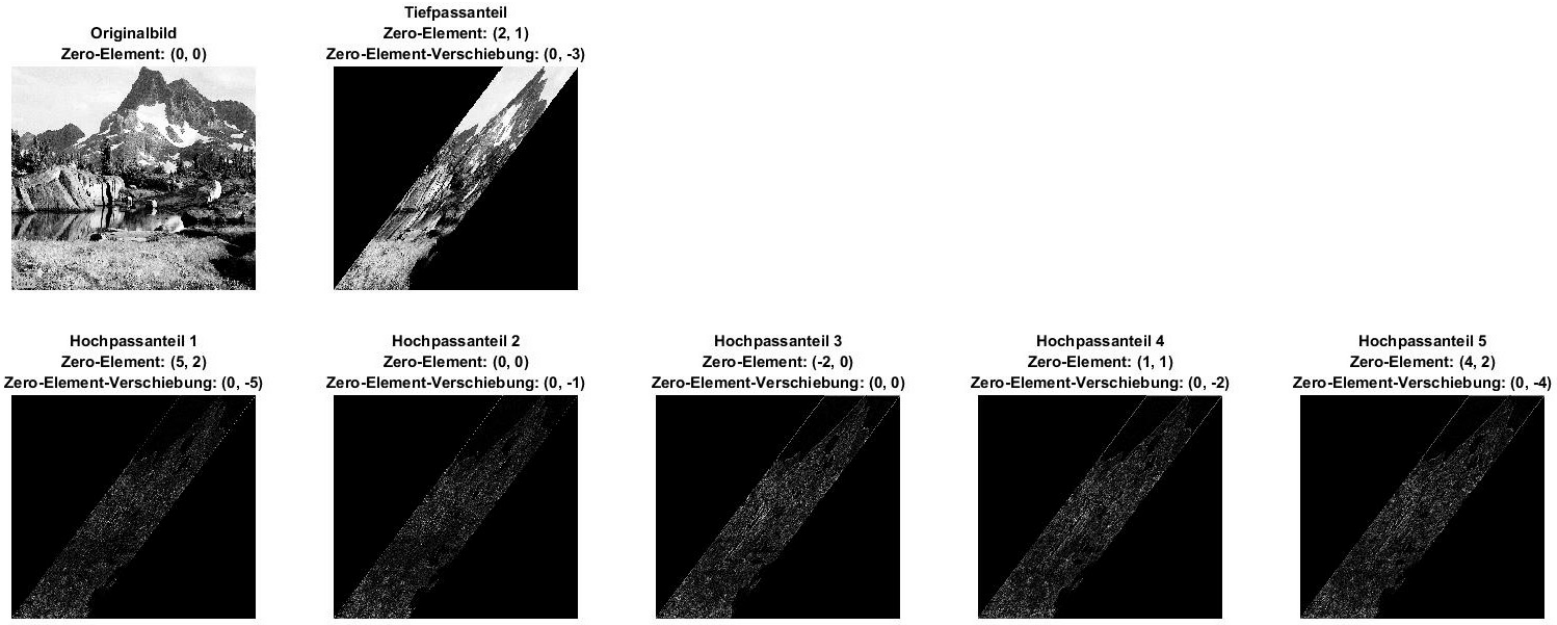

Abbildung 2.12: Das Bild `'mountain.png'`[5] wird unter Verwendung der Filterbank `'E01'` [4] einem einzelnen Zerlegungsschritt unterzogen.

WaveletReconstruction.m

```
function [image, completeThresholdInfo] = ...
     WaveletReconstruction(decArray, recCellIndex, varargin)
```

Wurde ein Bild in einem oder mehreren Schritten in Tiefpass- und Hochpassanteile zerlegt, so kann es mit Hilfe der gespeicherten Verwaltungsinformationen wie der Position des `(0,0)`-Elements durch die Funktion `WaveletReconstruction.m` wieder vollständig rekonstruiert werden. Dabei wird das Dekomposition-Cell-Array `decArray` sowie der Index `recCellIndex` der Zelle angegeben, aus welcher das Bild wiederhergestellt werden soll. Solange als zusätzliche Parameter weder `'threshold'` noch das Schlüsselwort `'highpassOnly'` verwendet werden, ist eine perfekte Rekonstruktion möglich.

Beispiel 2.1.11: Zerlegung und perfekte Rekonstruktion eines Bildes
In dieser Matlab-Routine wird das Bild `'boat.png'`[5] zuerst mit dem Filter `'M11'` (siehe Abschnitt 2.2.2) bis zur Tiefe 2 zerlegt und anschließend wieder vollständig rekonstruiert. Zum Schluss überprüft die Funktion `CompareImages`, ob die Rekonstruktion erfolgreich war und das rekonstruierte Bild exakt dem Originalbild entspricht. Abbildung 2.13 zeigt das Zwischenergebnis nach der zweischrittigen Zerlegung an.

```
» image = imread('Testbilder/boat.png');
» decArray = WaveletDecomposition(image, 'M11', 2);
```

```
» newImage = WaveletReconstruction(decArray, [3, 1]);
» CompareImages(image, newImage)

ans =

Images are identically.
```

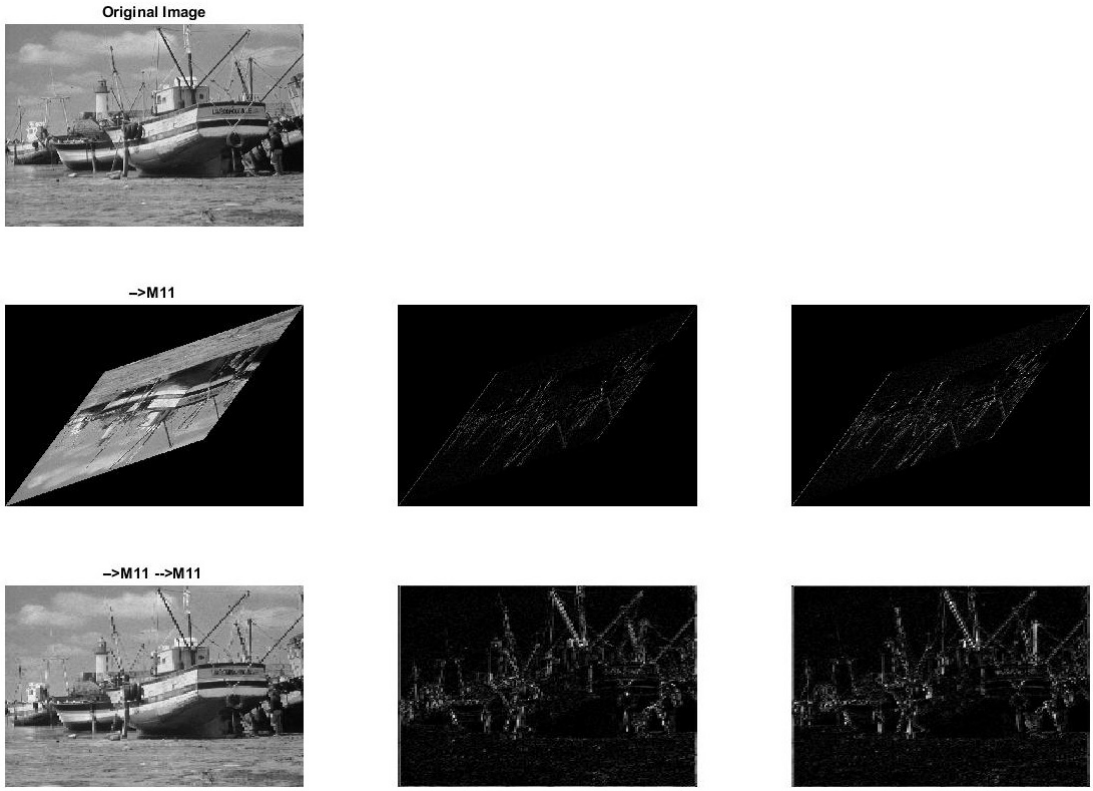

Abbildung 2.13: Das Bild `'boat.png'`[5] wird unter Verwendung der Filterbank `'M11'` [1] zuerst bis zu einer Tiefe von 2 zerlegt und anschließend aus der untersten Zerlegung wieder vollständig rekonstruiert Die Abbildung zeigt das Zwischenergebnis nach der Zerlegung.

Wird als zusätzlicher Parameter `'highpassOnly'` angegeben, so wird das Bild lediglich aus den Hochpassanteilen rekonstruiert. Eine perfekte Rekonstruktion ist damit nicht mehr möglich, jedoch eignet sich dieser Ansatz optimal zur Kantenerkennung im Bild. Der unterste Tiefpassanteil wird dazu gelöscht.

Beispiel 2.1.12: Rekonstruktion eines Bildes aus dem Hochpassanteil
In dieser Matlab-Routine wird das Bild `'goldhill.png'`[5] zuerst mit dem Filter `'M01'` (siehe Abschnitt 2.2.1) bis zur Tiefe 3 zerlegt und anschließend nur aus den Hochpassinformationen rekonstruiert. Dazu wird der Tiefpassanteil in der untersten Zerlegungsebene gelöscht und somit bei der Rekonstruktion nicht berücksichtigt. Abbildung 2.14 veranschaulicht das Vorgehen. In Abbildung 2.15 ist das Ergebnis der Rekonstruktion allein aus den Hochpassinformationen dargestellt.

```
» image = imread('Testbilder/goldhill.png');
» decArray = WaveletDecomposition(image, 'M01', 3);
» newImage = WaveletReconstruction(decArray, [4, 1], 'highpassOnly');
```

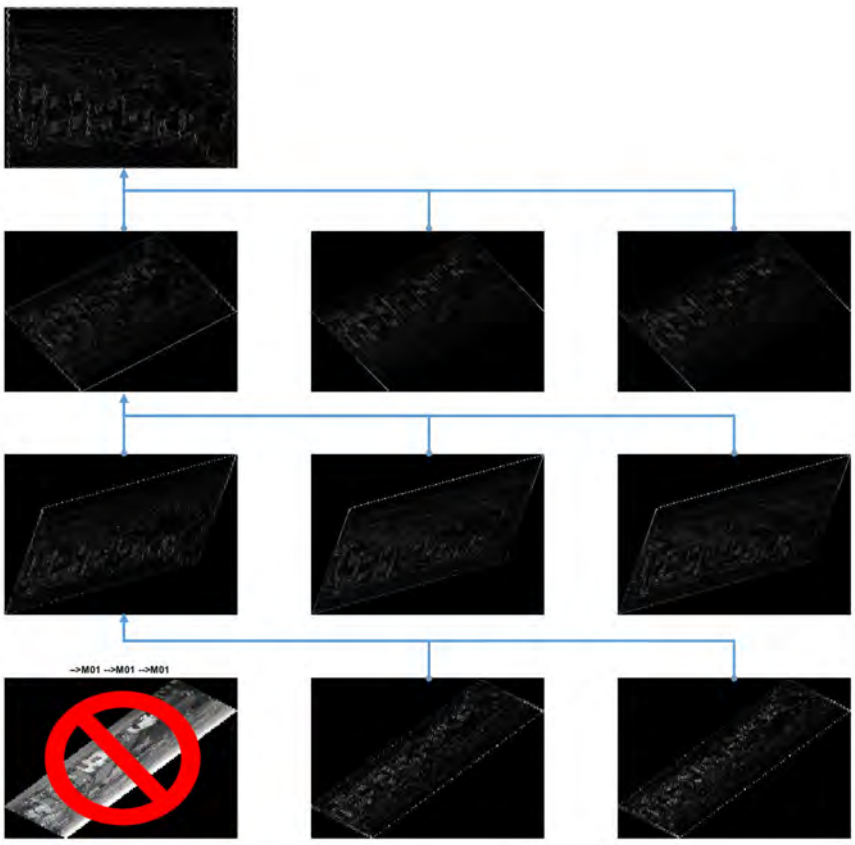

Abbildung 2.14: Das Bild `'goldhill.png'`[5] wird unter Verwendung der Filterbank `'M01'` [1] zerlegt. Der Tiefpassanteil in der untersten Ebene wird gelöscht und anschließend werden in jedem Rekonstruktionsschritt nur die Hochpassanteile zur Rekonstruktion verwendet.

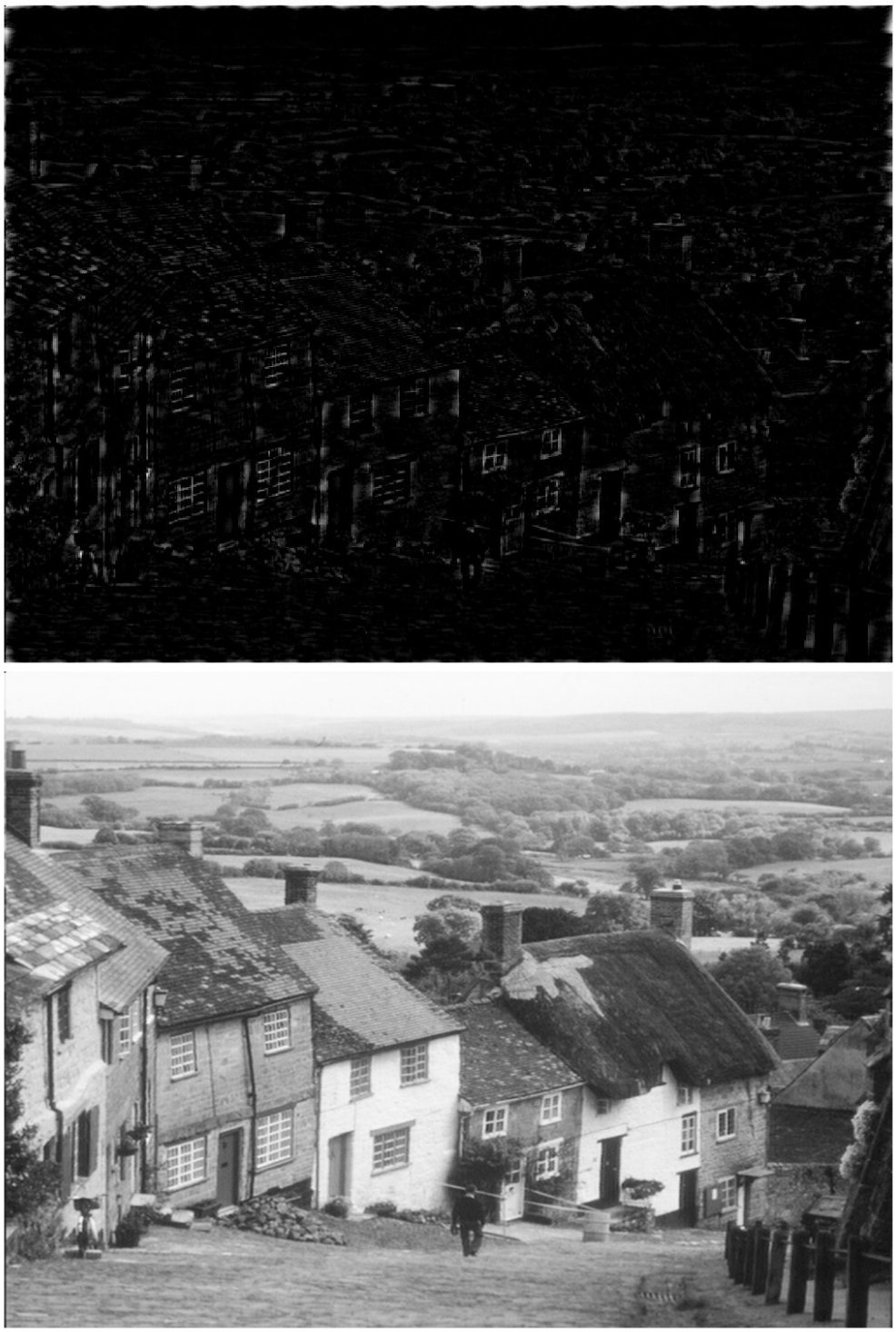

Abbildung 2.15: Die Abbildung zeigt oben das Ergebnis der Rekonstruktion des Bildes `'goldhill.png'`[5] allein aus den Hochpassinformationen unter Verwendung der Filterbank `'MO1'` [1]. Man erkennt deutlich die Kanten im Bild. Unten ist das Originalbild zum Vergleich noch einmal abgebildet.

Bei der Rekonstruktion des Bildes nur aus den Hochpassinformationen entstehen an den Bildrändern unerwünschte Artefakte. Dies ist in Abbildung 2.15 eindeutig erkennbar. Um diesen Effekt zu vermeiden, werden die Bildränder bereits vor der Dekomposition durch Wiederholung der äußersten Pixel erweitert. Durch die richtige Wahl der hinzugefügten Rahmenstärke können derartige Artefakte fast vollständig vermieden werden. Der Prozess ist in Abbildung 2.11 beschrieben.

Wird nun als zusätzlicher Parameter `'threshold'` gefolgt von einem Schwellwert angegeben, so wird bei der Rekonstruktion in den Hochpassanteilen jeder Bildpixel, der betragsmäßig kleiner als der angegebene Thresholdwert ist, verworfen und gleich 0 gesetzt. Beim Tiefpassanteil passiert dies natürlich nicht, da das Ziel des Prozesses eine möglichst optimale Kompression der Bilddaten unter Inkaufnahme des Verlustes weniger Detailinformationen ist. Hier ist nicht nur ein absoluter Schwellwert erlaubt, auch Prozentwerte wie `'10%'` sind möglich. Der tatsächliche Schwellwert wird dann intern derart gewählt, dass das Bild schließlich aus dem angegebenen Prozentsatz an von Null verschiedenen Werten im Hinblick auf das Originalbild rekonstruiert werden kann. Gibt man zudem den Übergabeparameter `'decay'` an, so wird die Abklingrate der Hochpasskoeffizienten berücksichtig und die Auswahl der Hochpasskoeffizienten in den einzelnen Tiefenleveln individuell durchgeführt. Im Tiefenlevel $i \in \{1, 2, ...\}$ werden die Hochpassbildanteile für die Koeffizientenwahl mit dem Faktor $|det(M)|^i$ multipliziert, wobei M die zum verwendeten Filter gehörige Dilatationsmatrix ist. Anschließend werden die zu kleinen Hochpasskoeffizienten gleich Null gesetzt.

Beispiel 2.1.13: Anwenden eines Threshold auf ein Bild
Nachfolgende Matlab-Routine zerlegt das Bild `'peppers.png'`[5] und unterzieht es bei der Rekonstruktion einem Threshold mit Schwellwert 20. D.h. es werden sämtliche Pixel der Hochpassanteile, deren Betrag kleiner als 20 ist gleich 0 gesetzt. Abbildung 2.16 zeigt das Ergebnis der Rekonstruktion.

```
» image = imread('Testbilder/peppers.png');
» decArray = WaveletDecomposition(image, 'M11', 3);
» newImage = WaveletReconstruction(decArray, [4, 1], 'threshold', 20);
» newImageDecay = WaveletReconstruction(decArray, [4, 1], 'threshold',
        150, 'decay');
```

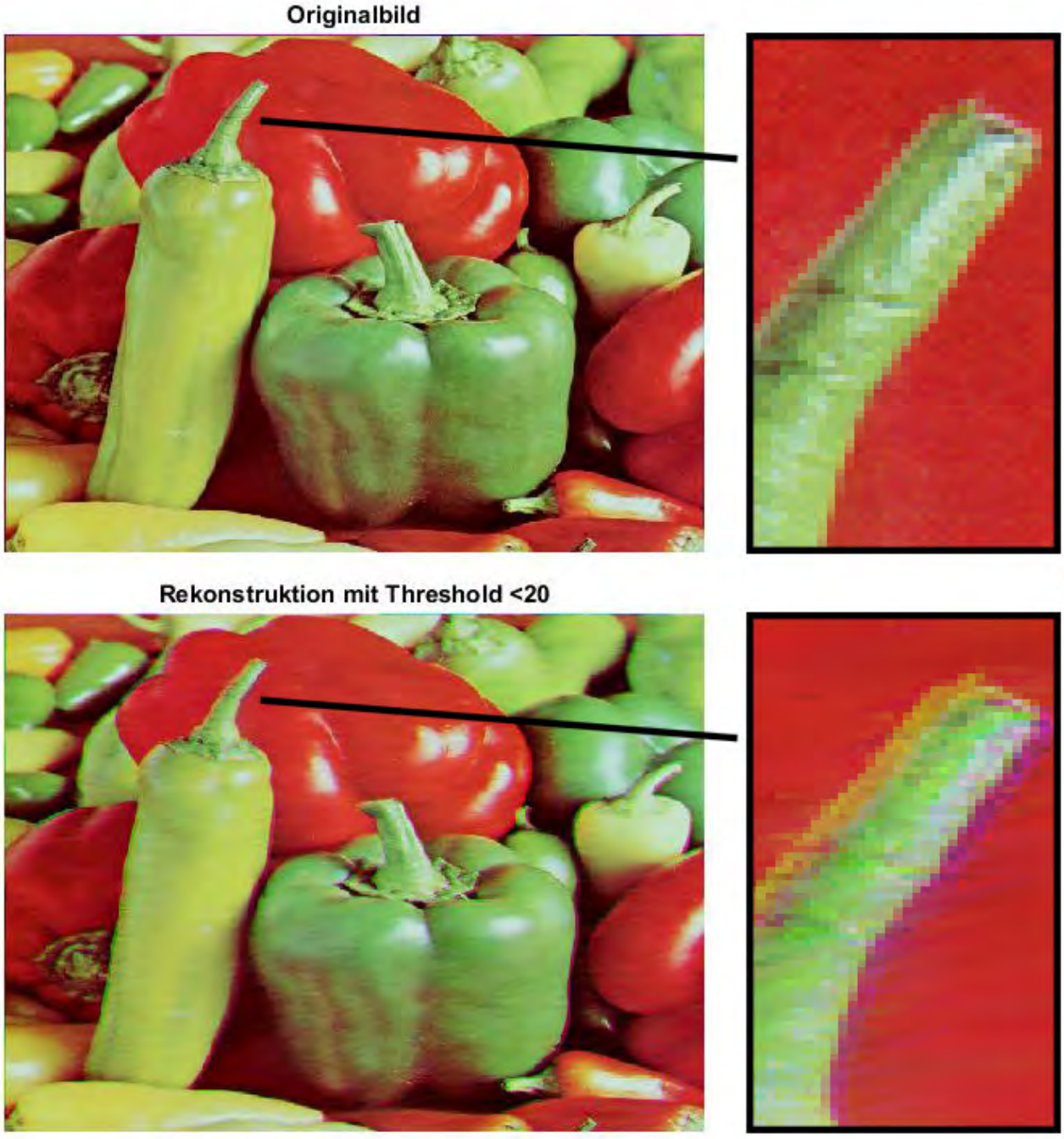

Abbildung 2.16: Oben sieht man das Originalbild `'peppers.png'`[5], unten dessen Rekonstruktion mittels Thresholding. Das Originalbild besteht insgesamt aus 756.307 von Null verschiedenen Elementen. Nach der Anwendung des Thresholds werden für die Rekonstruktion noch 82.386 von Null verschiedene Werte benötigt. Zu erkennen ist auch, dass bei der Anwendung des Thresholds fast ausschließlich Informationen über die Texturen, nicht aber über Kanten im Bild verloren gehen.

WaveletReconstruction_1Level.m

```
function [newLowpassData, thresholdInfo] = ...
     WaveletReconstruction_1Level(lowpassData, highpassData, filter, varargin)
```

Die Funktion `WaveletReconstruction_1Level.m` wird von der Funktion `WaveletReconstruction` (siehe Abschnitt 2.1.2) benötigt und führt einen einzelnen Rekonstruktionsschritt durch. Als Parameter werden zwei Cell-Arrays `lowpassData` und `highpassData` mit den Tiefpass- und Hochpassinformationen sowie der verwendete Filter übergeben. Optional können wie beim Funktionsaufruf die Schlüsselbegriffe `'highpassOnly'` und `'threshold'` gefolgt von einem Schwellwert angegeben werden. Wird die Funktion `WaveletReconstruction_1Level.m` intern von der Funktion `WaveletReconstruction.m` aufgerufen und wurde diese mit dem Parameter `'decay'` gestartet, so wurde für das aktuelle Tiefenlevel bereits der passende Abklingfaktor berechnet. Dieser wird im Funktionsaufruf nach dem Schlüsselwort `'decay'` übergeben.
Die Funktion rekonstruiert aus den erhaltenen Daten den Tiefpassanteil `lowpassData` des darüberliegenden Levels. Sollte es sich um Level 1 handeln, wird das Originalbild erzeugt. Wurde ein Schwellwert für den Threshold gewählt, werden zusätzlich noch Informationen über die Kompressionsrate beim Threshold zurückgegeben.

Beispiel 2.1.14: 1-Level Rekonstruktion eines Bildes
Die nachfolgende Matlab-Routine zerlegt das Bild `'zelda.png'`[5] unter Verwendung des Filters `'MO1'` bis zu einer Tiefe von 2. Anschließend wird der erste Rekonstruktionsschritt durchgeführt, d.h. aus den Daten der untersten Ebene wird der Tiefpass der mittleren Ebene erzeugt. Die schematische Darstellung des Prozesses findet sich in Abbildung 2.17 wieder.

```
» image = imread('Testbilder/zelda.png');
» decArray = WaveletDecomposition(image, 'MO1', 2);
» lowpassData = decArray{3, 1}{1};
» highpassData = decArray{3, 1}{2};
» newLowpassData =
      WaveletReconstruction_1Level(lowpassData, highpassData, 'MO1')
```

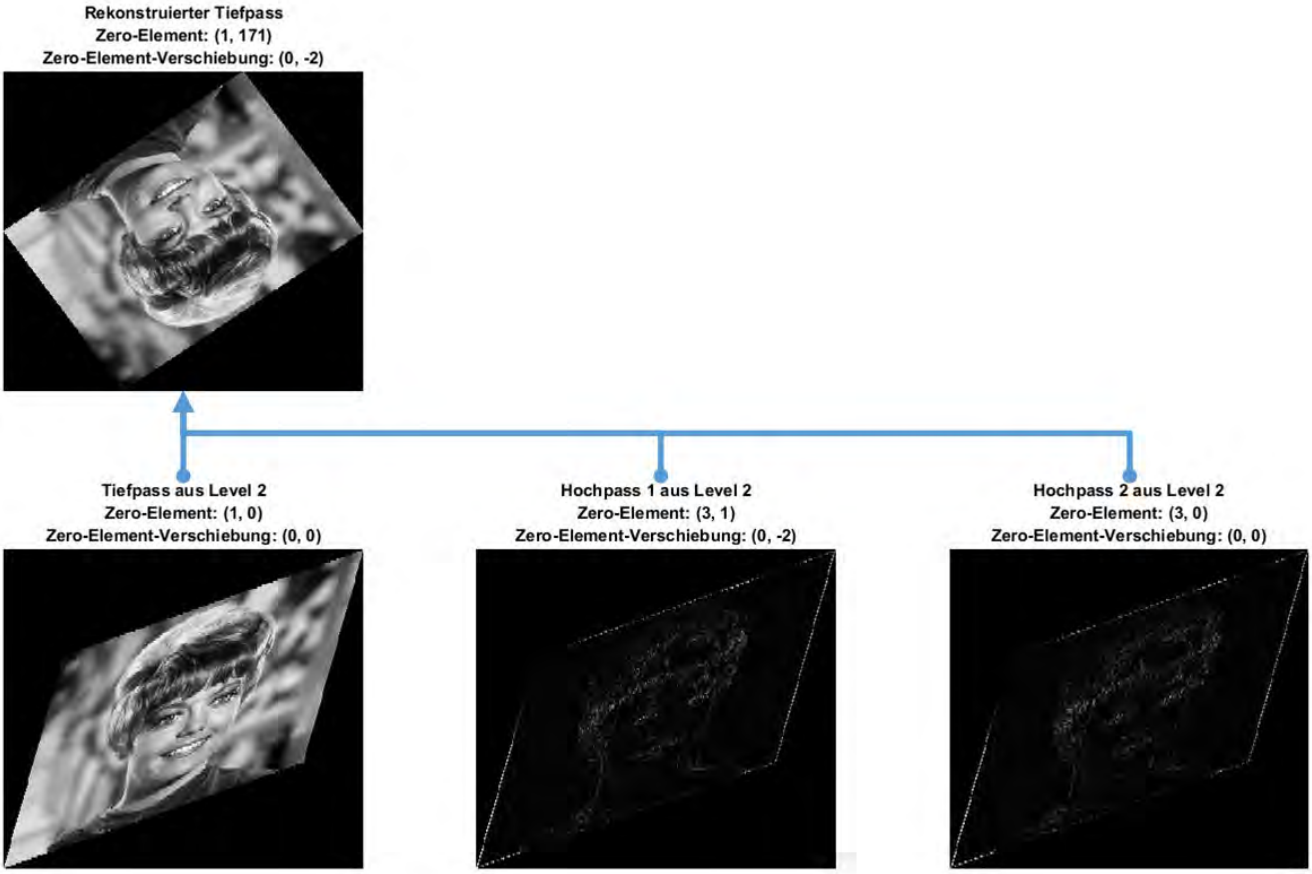

Abbildung 2.17: Aus den Bildinformationen des untersten Levels wird der Tiefpassanteil des darüberliegenden Levels rekonstruiert. Dieser Vorgang wird solange wiederholt, bis man sich auf Level 0 befindet und das Originalbild rekonstruiert wurde.

AllPathesReconstruction.m

```
function newImage = AllPathesReconstruction(decArray, threshold, varargin)
```

Die Funktion `AllPathesReconstruction.m` erhält ein Cell-Array mit Zerlegungsinformationen und einen anzuwendenden Schwellwert. Aus jedem Zerlegungspfad wird eine eigenständige Threshold-Rekonstruktion gestartet. Die resultierenden Bilder werden aufsummiert. Das Rückgabebild besteht aus der Mittelung aller Bilder aus den einzelnen Pfaden. Optional kann noch der Parameter `'decay'` angegeben werden, wenn während der Rekonstruktion auf die Abklingrate der Hochpasskoeffizienten Rücksicht genommen werden soll.
Durch die Mittelung werden potentiell bessere Ergebnisse erzielt. Allerdings ist der Speicherplatzbedarf höher, da alle Zerlegungspfade gespeichert werden müssen.

Beispiel 2.1.15: Rekonstruktion eines Bildes aus allen Pfaden
Die nachfolgende Matlab-Routine zerlegt das Bild `'boots.jpg'`[5] unter Verwendung der Filterbänke `'E00'` und `'E01'` bis zu einer Tiefe von 3. Anschließend wird das Bild als Mittelung aus allen Rekonstruktionspfaden unter Anwendung eines Thresholds wiederhergestellt. Dabei wird das Schlüsselwort `'decay'` verwendet, um die Auswahl der Hochpasskoeffizienten unter Berücksichtigung der Abklingrate durchzuführen.

```
» image = imread('Testbilder/boots.jpg');
» image = image(:,:,1);
» decArray = WaveletDecomposition(image, ['E00';E01], 3);
» newImage = AllPathesReconstruction(decArray, 200, 'decay');
```

2.1.3 Hilfsfunktionen

Die Hilfsfunktionen übernehmen kleinere Tätigkeiten, bei denen es sinnvoll war, sie aus den wichtigen Funktionen auszulagern.

CompareImages.m

```
function compareString = CompareImages(image1, image2)
```

Die Funktion `CompareImages` erhält zwei Bildmatrizen und überprüft Pixel für Pixel, ob diese identisch zueinander sind. Stimmen die Dimensionen der Bilder nicht überein, wird eine Fehlermeldung ausgegeben. Dabei sind auch Bilder mit mehreren Layern (z.B. RGB oder A-RGB) erlaubt. Haben zwei Bilder dieselbe Größe, unterscheiden sich aber voneinander, so wird die Anzahl der Pixel ausgegeben, in denen sie jeweils unterschiedliche Werte haben.
Die Funktion eignet sich hervorragend zur Prüfung, ob die Rekonstruktion fehlerfrei und perfekt verlaufen ist oder nicht.

Beispiel 2.1.16: Vergleich zweier Bilder
Die nachfolgende Matlab-Routine soll aufzeigen, wie die Funktion `CompareImages` zu verwenden ist.

```
» image1 = [1, 2, 3; 4, 5, 6]

image1 =

1     2     3
4     5     6

» image2 = [1, 2, 3; 4, 5, 6; 7, 8, 9]
```

```
image2 =

1     2     3
4     5     6
7     8     9

» CompareImages(image1, image2)

ans =

Image dimensions do not coincide.

» image1 = [image1; 9, 8, 7]

image1 =

1     2     3
4     5     6
9     8     7

» CompareImages(image1, image2)

ans =

2 of 9 pixels are different.

» image1 = image2

image1 =

1     2     3
4     5     6
7     8     9

» CompareImages(image1, image2)

ans =

Images are identically.
```

ConvertCoordinates.m

```
function newCoordinates = ConvertCoordinates(coordinates, matrix, type)
```

In Matlab werden Einträge von Matrizen mit der entsprechenden Zeilen- und Spaltennummer von links oben mit 1 beginnend angesprochen. Für die Bildzerlegung und Rekonstruktion benötigen wir häufig kartesische Koordinaten, wobei das linkeste unterste Element der Matrix dem Koordinatenursprung entspricht. Die Funktion `ConvertCoordinates` wandelt die beiden Koordinatentypen ineinander um. Dazu werden die umzuwandelnden Koordinaten `coordinates`, die zugrunde liegende Matrix `matrix` sowie der Ausgangstyp der Koordinaten angegeben. Als Rückgabewert erhält man die in den jeweils anderen Koordinatentyp umgewandelten Koordinaten.

Beispiel 2.1.17: Umwandlung von Koordinaten
Die nachfolgende Matlab-Routine soll verdeutlichen, wie die jeweiligen Koordinatentypen ineinander umgewandelt werden können. Die Matlab-Koordinaten in Zeilen- und Spaltenform werden in kartesische Koordinaten (x, y) umgewandelt, wobei das linkeste unterste Element (in diesem Fall die 7) als Koordinatenursprung $(0, 0)$ fungiert.

```
» matrix = [1, 2, 3; 4, 5, 6; 7, 8, 9]

matrix =
1     2     3
4     5     6
7     8     9

» matrix(3, 2)

ans =

8

» ConvertCoordinates([3, 2], matrix, 'matlab')

ans =

1     0
```

ConvolveImageWithFilter.m

```
function convolvedImage = ConvolveImageWithFilter(image, filter)
```

Die Funktion `ConvolveImageWithFilter` erhält als Parameter eine Bildmatrix und eine Faltungsmatrix. Anschließend wird eine zweidimensionale Faltung mit Hilfe der Matlab-Funktion `conv2` durchgeführt.

Beispiel 2.1.18: Zweidimensionale Faltung
Die nachfolgende Matlab-Routine stellt anschaulich dar, was bei einer Faltung im Zweidimensionalen passiert. Die Filtermatrix wird wie bei einer Faltung üblich von links über die Bildmatrix geschoben.

```
» image = ones(4, 6)

image =

1     1     1     1     1     1
1     1     1     1     1     1
1     1     1     1     1     1
1     1     1     1     1     1

» filter = [0, 0, 0; 0.5, 1, 0.5; 0, 0, 0]

filter =

0         0         0
0.5000    1.0000    0.5000
0         0         0

» ConvolveImageWithFilter(image, filter)

ans =

0        0        0        0        0        0        0        0
0.500    1.500    2.000    2.000    2.000    2.000    1.500    0.500
0.500    1.500    2.000    2.000    2.000    2.000    1.500    0.500
0.500    1.500    2.000    2.000    2.000    2.000    1.500    0.500
0.500    1.500    2.000    2.000    2.000    2.000    1.500    0.500
0        0        0        0        0        0        0        0
```

ThresholdStorageSaving.m

```
function [nonZeroBefore, nonZeroAfter] = ThresholdStorageSaving(decArray, ...
     recCellIndex, threshold, varargin)
```

Die Funktion `ThresholdStorageSaving` berechnet für einen vollständigen Rekonstruktionspfad die Anzahl an benötigten von Null verschiedenen Elementen, damit das Bild vollständig wiederhergestellt werden kann. Dabei wird das Zerlegungs-Cell-Array sowie der Index der Zelle, von der aus rekonstruiert werden soll, übergeben. Zusätzlich wird der zu verwendende Threshold angegeben. Optional kann dieser um das Schlüsselwort

'decay' erweitert werden, wenn die Abklingrate der Hochpasskoeffizienten beim Thresholding berücksichtigt werden soll. Die Funktion gibt zudem die Anzahl der von Null verschiedenen Elemente im Originalbild an.

Beispiel 2.1.19: Berechnung der Speicherplatzersparnis durch Thresholding
Die nachfolgende Matlab-Routine zeigt auf, wie die Anzahl der von Null verschiedenen für die Rekonstruktion notwendigen Elemente bestimmt werden kann.

```
» image = imread('Testbilder/boots.jpg');
» image = image(:,:,1);
» decArray = WaveletDecomposition(image, 'E00', 3);
» [nonZeroBefore, nonZeroAfter] = ThresholdStorageSaving(decArray, ...
        [4, 1], 180, 'decay')

nonZeroBefore =

1259712

nonZeroAfter =

69021
```

ThresholdStorageSavingAllPathesReconstruction.m

```
function [nonZeroBefore, nonZeroAfter] = ...
     ThresholdStorageSavingAllPathesReconstruction(decArray, ...
     threshold, varargin)
```

Die Funktion `ThresholdStorageSavingAllPathesReconstruction` berechnet die Anzahl der von Null verschiedenen Elementen, welche für die Rekonstruktion benötigt werden, wenn das Bild als Mittelwert aus sämtlichen Zerlegungspfaden wiederhergestellt werden soll. Auch hier kann das Schlüsselwort 'decay' angegeben werden.

Beispiel 2.1.20: Berechnung der Speicherplatzersparnis durch Thresholding
Die nachfolgende Matlab-Routine zeigt auf, wie die Anzahl der von Null verschiedenen für die Rekonstruktion notwendigen Elemente bestimmt werden kann, wenn das resultierende Bild aus der Mittelung der einzelnen Ergebnisse aller Zerlegungspfade hervorgehen soll.

```
» image = imread('Testbilder/boots.jpg');
» image = image(:,:,1);
» decArray = WaveletDecomposition(image, ['E00';'E01'], 3);
```

```
» [nonZeroBefore, nonZeroAfterSum] =
      ThresholdStorageSavingAllPathesReconstruction(decArray, 70)

nonZeroBefore =

1259712

nonZeroAfterSum =

84883
```

2.1.4 Sampling-Funktionen

Die einzelnen Sampling-Funktionen übernehmen das Down- sowie Upsampling bei Dekomposition und Rekonstruktion. Einerseits werden Zeilen und Spalten der Bildmatrix gelöscht, andererseits werden Nullzeilen und -spalten hinzugefügt.

CalculateZeroElementPositionAfterDownsampling.m

```
function [newZeroElementPosition, changeVector] = ...
     CalculateZeroElementPositionAfterDownsampling(matrix, ...
     scaleFactorRows, scaleFactorColumns, zeroElementPosition, changeVector)
```

Während des Downsamplingprozesses kann das `(0,0)`-Element wegfallen. Um trotzdem perfekte Rekonstruktion garantieren zu können, muss also ein neues `(0,0)`-Element ausgewählt werden. In der Regel wird hier das Element aus der nächsthöheren Zeile und nächstlinkeren Spalte genommen, die beim Downsampling nicht wegfallen. Dies wird dann in den Verwaltungsdaten als Verschiebungsvektor notiert. Die Berechnung der neuen Position `newZeroElementPosition` des `(0,0)`-Elements und des gegebenenfalls entstehenden Verschiebungsvektors `changeVector` übernimmt nun die Funktion `CalculateZeroElementPositionAfterDownsampling`. Diese erhält als Parameter die betroffene Matrix sowie die Zeilen- und Spaltenfaktoren fürs Downsampling. Zudem werden die aktuelle Position des `(0,0)`-Elements (vor dem Downsampling) sowie ein möglicherweise bereits existierender Verschiebungsvektor übergeben. Vergleiche hierzu den Abschnitt 2.3.2.
Abbildung 2.18 zeigt bildlich, was in der Funktion `CalculateZeroElementPositionAfter Downsampling` berechnet wird.

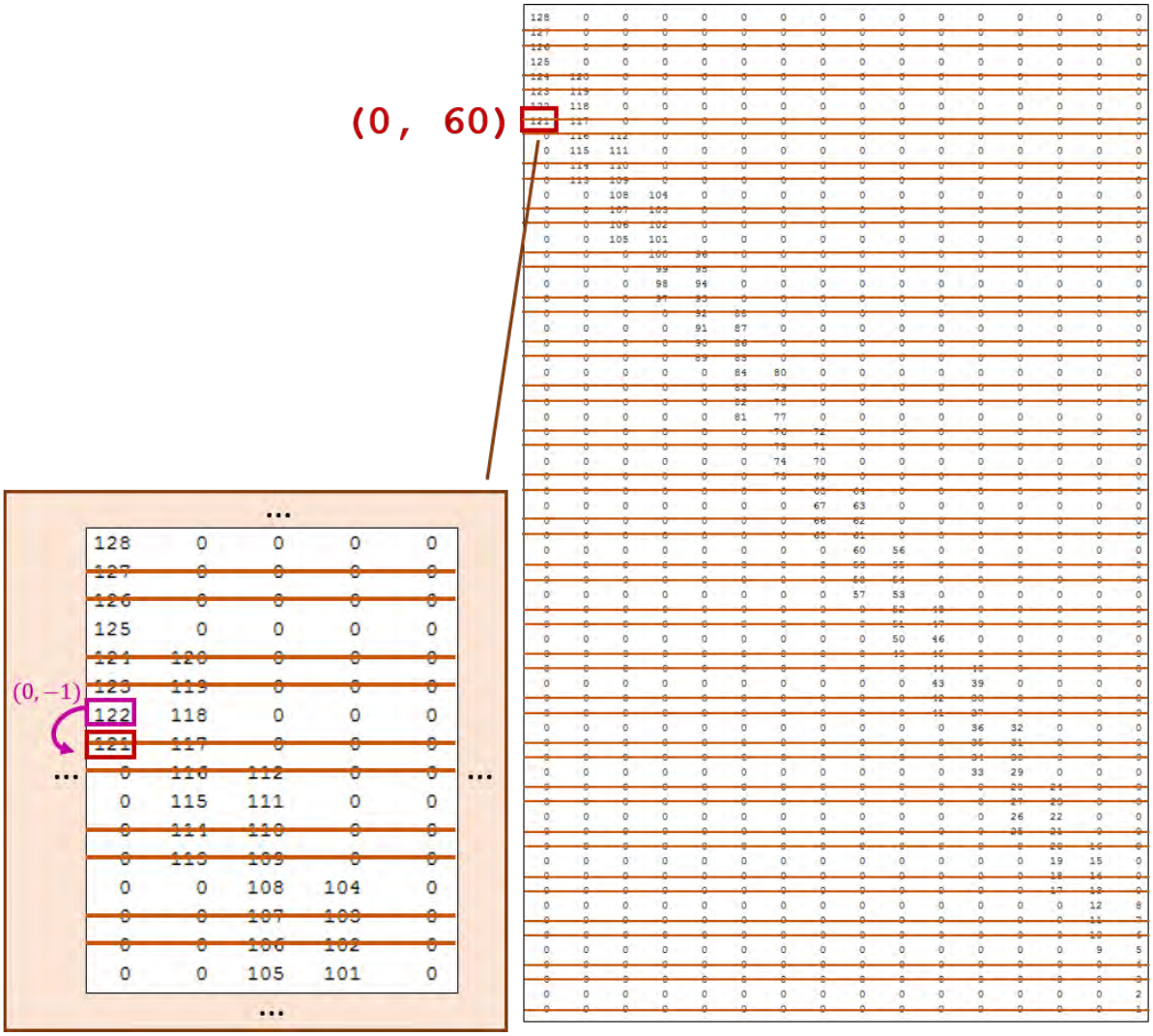

Abbildung 2.18: Das alte `(0,0)`-Element (rot) fällt beim Downsampling weg. Für die Rekonstruktion muss ein neues Element (pink) gewählt werden.

CalculateZeroElementPositionAfterUpsampling.m

```
function [newZeroElementPosition, changeVector] = ...
    CalculateZeroElementPositionAfterUpsampling(matrix, ...
    scaleFactorRows, scaleFactorColumns, zeroElementPosition, changeVector)
```

Der in Abschnitt 2.1.4 durchgeführte Prozess muss während der Rekonstruktion, genauer nach dem Upsampling, wieder rückgängig gemacht werden. Bei Bedarf muss dann auch das ursprüngliche `(0,0)`-Element wieder ausgewählt werden. Dies übernimmt die Funktion `CalculateZeroElementPositionAfterUpsampling`. Vergleiche dazu den Abschnitt 2.3.3.

Abbildung 2.19 zeigt exemplarisch, was bei dem Vorgang passiert.

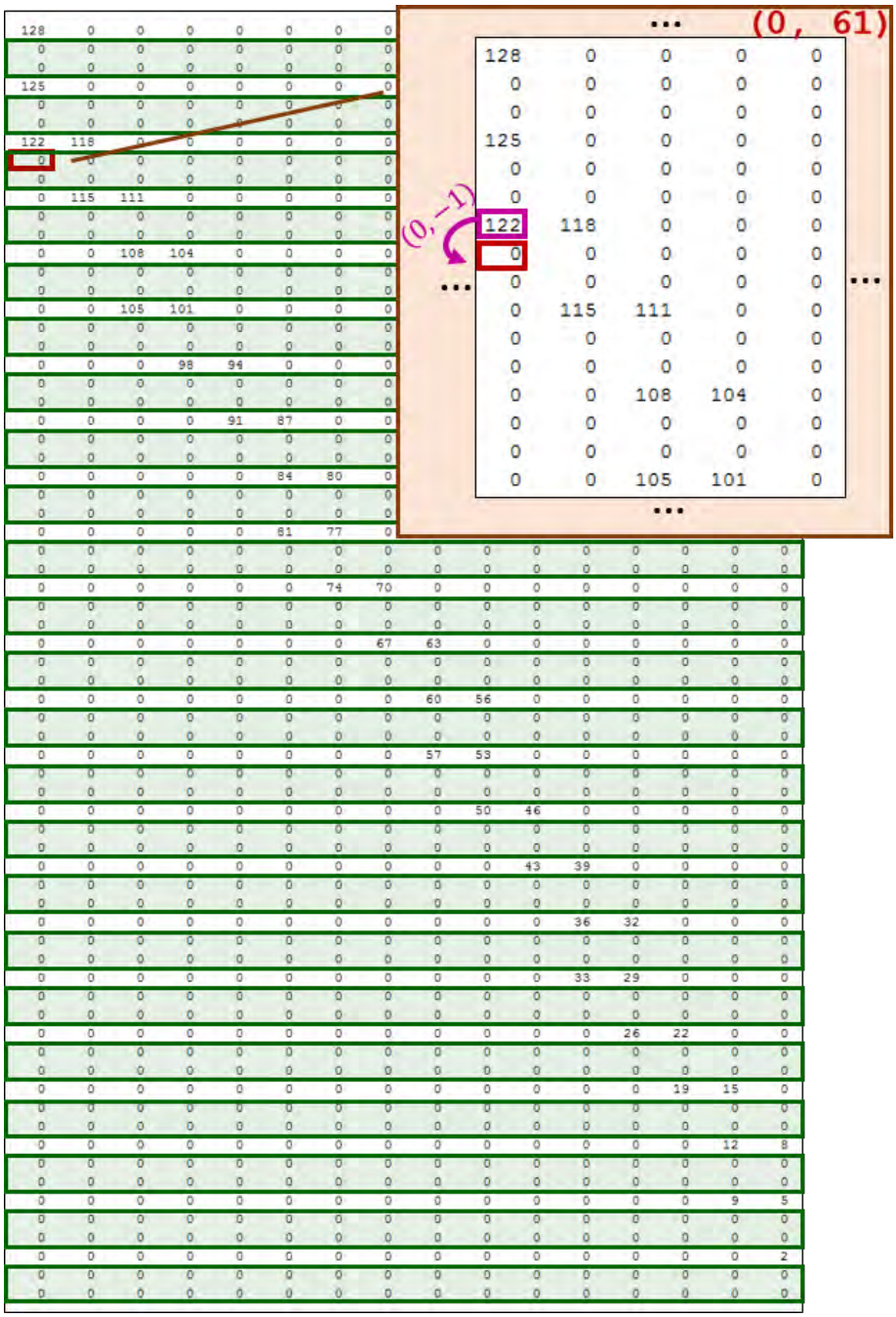

Abbildung 2.19: Ist das `(0,0)`-Element beim Downsampling weggefallen, muss dieses im entsprechenden Upsamplingprozess wieder korrekt ausgewählt werden, damit perfekte Rekonstruktion möglich ist. Dies geschieht anhand des in den Verwaltungsdaten gespeicherten `changeVector`.

Downsampling2D_1Dimension

```
function downsampledMatrix = Downsampling2D_1Dimension(matrix, ...
     dimension, scaleFactor)
```

Die Funktion `Downsampling2D_1Dimension` führt auf einer Matrix `matrix` ein Downsampling entlang der Dimension `dimension` (`'row'` oder `'column'`) um den Faktor `scaleFactor` durch.

Beispiel 2.1.21: Eindimensionales Downsampling
Folgende Matlab-Routine zeigt auf, wie die Funktion `Downsampling2D_1Dimension` arbeitet. Wird etwa entlang der Zeilendimension der Downsamplingfaktor 3 gewählt, so wird nur jede dritte Zeile behalten.

```
» image = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12];
» image = [image; image + 12]

image =

1     2     3     4
5     6     7     8
9     10    11    12
13    14    15    16
17    18    19    20
21    22    23    24

» image = Downsampling2D_1Dimension(image, 'row', 3)

image =

1     2     3     4
13    14    15    16

» image = Downsampling2D_1Dimension(image, 'column', 2)

image =

1     3
13    15
```

Downsampling2D_DilationMatrix

```
function imageData = Downsampling2D_DilationMatrix(imageData, dilationMatrix)
```

Nach der Faltung mit den Filtermatrizen wird die Bildmatrix in jedem Zerlegungsschritt einem Downsampling unter Berücksichtigung der Dilatationsmatrix der jeweiligen Filterbank unterzogen. Dies übernimmt die Funktion `Downsampling2D_Dilationmatrix`. Sie erhält als Parameter ein Cell-Array `imageData` mit der Bildmatrix sowie den Verwaltungsdaten (z.B. die Position des `(0,0)`-Element) und die Dilatationsmatrix der verwendeten Filterbank. Diese wird zuerst mit der Funktion `SmithFactorization` (im Programm enthalten) in die Smith-Normalform gebracht. Dadurch können die Samplingfaktoren sowie die Transformationsmatrizen abgelesen werden.
Während des gesamten Funktionsablaufs wird die Bildmatrix unter Zuhilfenahme der Funktion `TransformMatrixShift` (siehe Abschnitt 2.1.5) zweimal transformiert und dazwischen einem Downsampling gemäß der Funktion `Downsampling2D_1Dimension` (siehe Abschnitt 2.1.4) unterzogen. Dabei muss die Positionsänderung des `(0,0)`-Elements unter allen Umständen mitprotokolliert werden. Abbildung 2.20 zeigt die erste Transformation einer Beispielmatrix unter Verwendung der Dilatationsmatrix der Filterbank `'M01'` (siehe dazu Abschnitt 2.2.1). In Abbildung 2.21 werden die einzelnen Schritte des gesamten Prozesses deutlich.

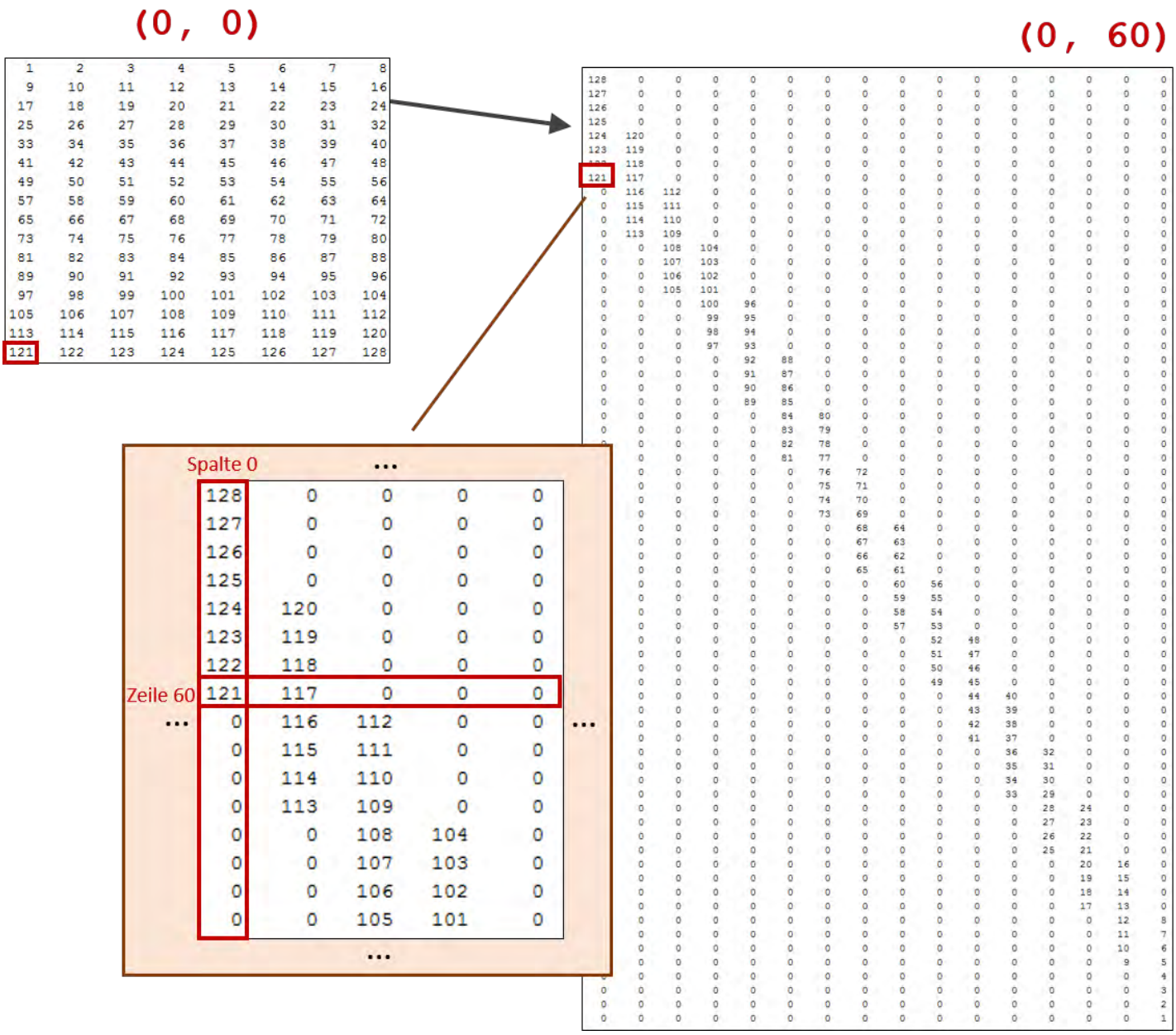

Abbildung 2.20: Die Beispielmatrix wird der ersten Transformation während des Downsamplings unterzogen. Die Funktion `TransformMatrixShift` berechnet neben der Matrixtransformation zudem auch die neue Position des `(0,0)`-Elements.

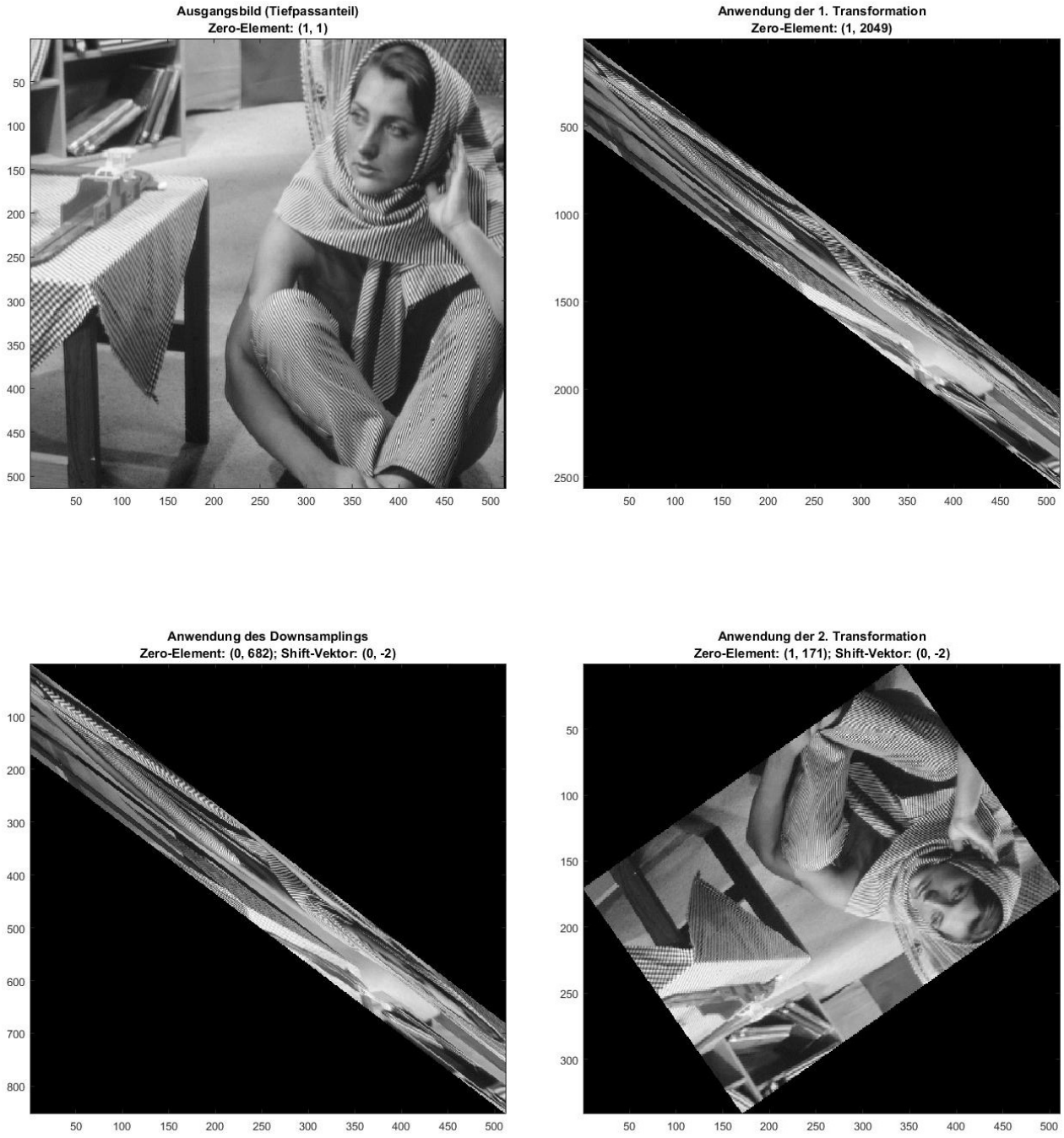

Abbildung 2.21: Das Bild `'barbara.png'`[5] wird transformiert und entlang beider Dimensionen einem Downsampling unterzogen. Während des Downsamplings fällt das `(0,0)`-Element weg, ein neues Element muss dessen Platz einnehmen. In diesem Schritt wird der entsprechende Shift-Vektor in den Verwaltungsdaten vermerkt, sodass die Aktion während der Rekonstruktion nach dem Upsampling wieder rückgängig gemacht werden kann.

Upsampling2D_1Dimension

```
function upsampledMatrix = Upsampling2D_1Dimension(matrix, ...
     dimension, scaleFactor)
```

Die Funktion `Upsampling2D_1Dimension` führt auf einer Matrix `matrix` ein Upsampling entlang der Dimension `dimension` (`'row'` oder `'column'`) um den Faktor `scaleFactor` durch.

Beispiel 2.1.22: Eindimensionales Upsampling
Folgende Matlab-Routine soll die Funktionsweise der Upsampling-Prozedur darlegen.

```
» image = [1, 2, 3; 4, 5, 6;]

image =

1     2     3
4     5     6

» image = Upsampling2D_1Dimension(image, 'row', 2)

image =

1     2     3
0     0     0
4     5     6
0     0     0

» image = Upsampling2D_1Dimension(image, 'column', 3)

image =

1     0     0     2     0     0     3     0     0
0     0     0     0     0     0     0     0     0
4     0     0     5     0     0     6     0     0
0     0     0     0     0     0     0     0     0
```

Upsampling2D_DilationMatrix

```
function imageData = Upsampling2D_DilationMatrix(imageData, dilationMatrix)
```

Die Funktion `Upsampling2D_DilationMatrix` stellt das genaue Gegenstück zur Funktion `Downsampling2D_DilationMatrix` (siehe Abschnitt 2.1.4) dar und wird während jedes Rekonstruktionsschrittes vor der Faltung mit den Rekonstruktionsfiltermatrizen benötigt.

Auch sie greift bei den anstehenden Transformationen auf die Funktion `TransformMatrixShift` (siehe Abschnitt 2.1.5) zurück. Während des gesamten Upsamplingprozesses wird die Position des `(0,0)`-Elements mitprotokolliert. Zudem muss ein gegebenenfalls stattgefundener Tausch des `(0,0)`-Elements wieder rückgängig gemacht werden. In Abbildung 2.22 wird die Bildmatrix nach ihrer ersten Transformation einem Upsampling unterzogen und die Position des `(0,0)`-Elements anschließend passend aktualisiert. Die Abbildung 2.23 verdeutlicht die Einzelschritte während des Prozesses.

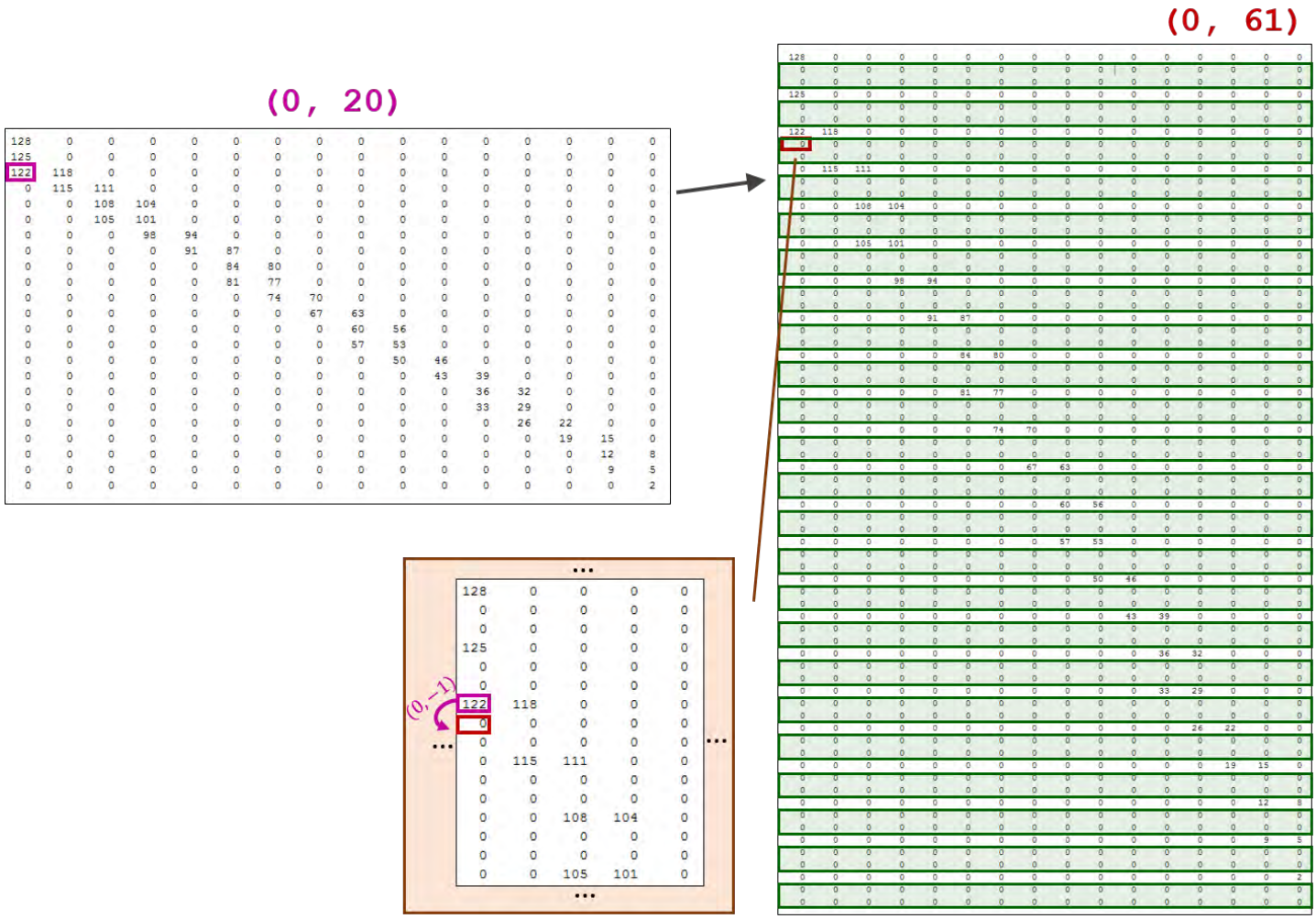

Abbildung 2.22: Nach dem Upsampling muss die gegebenenfalls beim Downsampling erfolgte Auswahl eines neuen `(0,0)`-Elements wieder rückgängig gemacht werden.

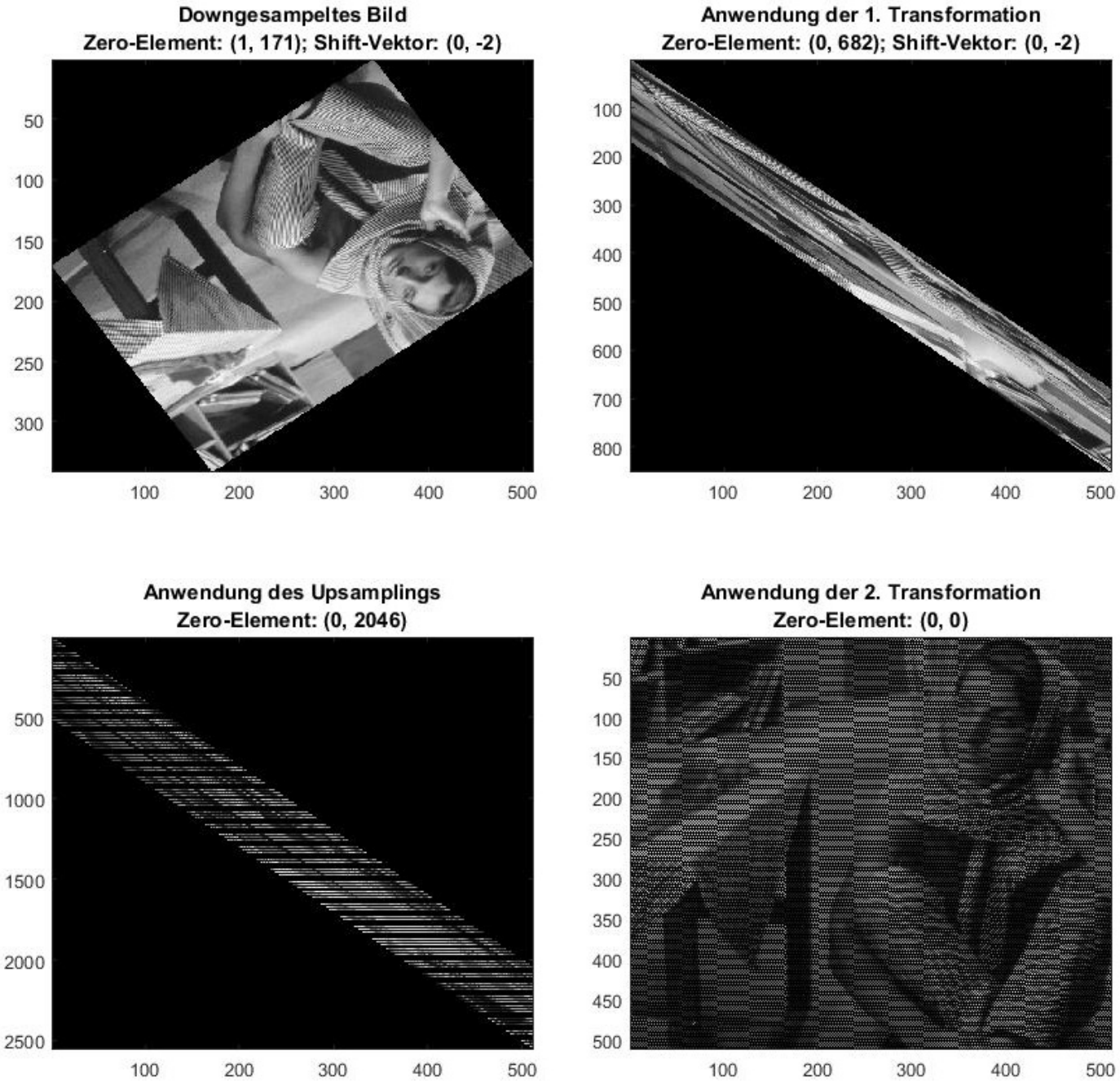

Abbildung 2.23: Das Bild `'barbara.png'`[5] wird transformiert und entlang beider Dimensionen einem Upsampling unterzogen. Nach dem Upsampling muss der Shift-Vektor auf die Position des `(0,0)`-Elements angewendet werden.

2.1.5 Transformationsfunktionen

Die Transformationsfunktionen verschieben, drehen und scheren Matrizen und löschen Nullzeilen und -spalten.

AlignMatrices.m

```
function [matrices, zeroElementPositions] = ...
     AlignMatrices(matrices, zeroElementPositions)
```

Die Funktion `AlignMatrices` wird benötigt, wenn im Programm zusätzliche Filterbänke implementiert werden sollen. Für den korrekten Programmablauf ist es zwingend notwendig, dass die Positionen des `(0,0)`-Elements gleich sind. Dabei werden entsprechend Nullzeilen und -spalten eingefügt, die das Faltungsresultat nicht verändern, das `(0,0)`-Element jedoch an die passende Stelle verschieben.

Beispiel 2.1.23: Ausrichtung der Filtermatrizen
Untenstehende Matlab-Routine soll die Funktionsweise der `AlignMatrices`-Prozedur darstellen. Es werden zwei Matrizen `A` und `B` erzeugt. Diese enthalten jeweils an genau einer Stelle den Wert 2. Die Matrizen sollen derart verschoben werden, dass der Wert 2 bei beiden an derselben Position steht. Dazu werden Nullzeilen und -spalten eingefügt. Zudem sollen die Matrizen am Ende die gleiche Größe haben.
Zu beachten ist hier, dass die Positionen der `(0,0)`-Elemente (in unserem Fall die Positionen des Wertes 2) in kartesischen Koordinaten mit Ursprung links unten und nicht in Matlab-Koordinaten angegeben werden müssen.

```
» A = ones(3, 5);
» A(2, 2) = 2

A =

1     1     1     1     1
1     2     1     1     1
1     1     1     1     1

» B = ones(4, 4);
» B(2, 3) = 2

B =

1     1     1     1
1     1     2     1
1     1     1     1
1     1     1     1
```

```
» matrices{1} = A;
» matrices{2} = B

matrices =

[3x5 double]     [4x4 double]

» zeroElementPositions = [1, 1; 2, 2]

zeroElementPositions =

1     1
2     2

» [matrices, zeroElementPositions] = AlignMatrices(matrices,
zeroElementPositions)

matrices =

» A = ones(3, 5);
» A(2, 2) = 2

A =

1     1     1     1     1
1     2     1     1     1
1     1     1     1     1

» B = ones(4, 4);
» B(2, 3) = 2

B =

1     1     1     1
1     1     2     1
1     1     1     1
1     1     1     1

» matrices{1} = A;
» matrices{2} = B

matrices =
```

```
[4x6 double]     [4x6 double]

zeroElementPositions =

2     2
2     2

» matrices{1}

ans =

0     1     1     1     1     1
0     1     2     1     1     1
0     1     1     1     1     1
0     0     0     0     0     0

» matrices{2}

ans =

1     1     1     1     0     0
1     1     2     1     0     0
1     1     1     1     0     0
1     1     1     1     0     0
```

RemoveZeroBorder.m

```
function [slimImage, rowsDeletedAtBeginning, rowsDeletedAtEnd, ...
     colsDeletedAtBeginning, colsDeletedAtEnd] = RemoveZeroBorder(image, varargin)
```

Nach den Transformationen und Samplingfunktionen kann es sein, dass eine Bildmatrix `image` eine nicht unbeachtliche Anzahl an Nullzeilen- oder Spalten zu Beginn oder am Ende enthalten. Diese verfälschen das Ergebnis nicht, kosten aber Rechenzeit. Die Funktion `RemoveZeroBorder` sorgt dafür, dass nur noch die tatsächliche Bildmatrix `slimImage` übrig bleibt. Zudem wird die Anzahl der gelöschten Nullzeilen und -spalten zurückgegeben, damit die neue Position des `(0,0)`-Elements effektiv berechnet werden kann.
Beim Erkennen von Nullzeilen und -spalten wird mit der Zeilennorm und einem kleinen Vielfachen von *eps* gearbeitet. Wird als zusätzlicher Parameter das Schlüsselwort `'artefacts'` angegeben, so wird der verwendete Schwellwert für die Norm derart angepasst, dass Zeilen oder Spalten mit kleineren Artefakte der einzelnen Prozesse (z.B. Faltung) ebenfalls entfernt werden. Hierdurch werden Rundungsfehler zurechtgebogen.

Beispiel 2.1.24: Nullrand entfernen
Folgende Matlab-Routine soll darlegen, wie die Nullzeilen und -spalten einer Matrix entfernt werden.

```
» image = zeros(6, 9);

» image(2:3, 3:5) = [1, 2, 3; 4, 5, 6];

» image(2, 1) = eps

image =

0         0         0         0         0         0         0         0         0
0.0000    0    1.0000    2.0000    3.0000         0         0         0         0
0         0    4.0000    5.0000    6.0000         0         0         0         0
0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0
0         0         0         0         0         0         0         0         0

» [slimImage, rowsDeletedAtBeginning, rowsDeletedAtEnd,
colsDeletedAtBeginning, colsDeletedAtEnd] = RemoveZeroBorder(image)

slimImage =

1     2     3
4     5     6

rowsDeletedAtBeginning =

1

rowsDeletedAtEnd =

3

colsDeletedAtBeginning =

2

colsDeletedAtEnd =

4
```

ShiftMatrix.m

```
function shiftedImage = ShiftMatrix(image, shiftVector)
```

Während des gesamten Verarbeitungsprozesses müssen an vielen verschiedenen Stellen Matrizen an den Positionen ihrer (0,0)-Elementen ausgerichtet werden. Dazu werden passend Nullzeilen und -spalten eingefügt. Dies übernimmt die Funktion `ShiftMatrix`. Sie erhält eine Bildmatrix `image` und einen Verschiebungsvektor `shiftVector` und verschiebt die Matrix im Anschluss um diesen Vektor durch Einfügen oder Löschen von Nullzeilen oder -spalten.

Beispiel 2.1.25: Matrix verschieben
Untenstehende Matlab-Routine zeigt deutlich, wie die Funktion Matrizen verschiebt. Dabei wird mit Matlab-Koordinaten und nicht mit kartesischen Koordinaten gearbeitet.

```
» image = [1, 2, 3; 4, 5, 6]

image =

1     2     3
4     5     6

» image = ShiftMatrix(image, [2, 3])

image =

0     0     0     0     0     0
0     0     0     0     0     0
0     0     0     1     2     3
0     0     0     4     5     6

» image = ShiftMatrix(image, [-1, 0])

image =

0     0     0     0     0     0
0     0     0     1     2     3
0     0     0     4     5     6
```

TransformMatrixShift.m

```
function [transformedMatrix, newPositionZeroElement] = ...
     TransformMatrixShift(matrix, transformationMatrix, positionZeroElement)
```

Während des Down- bzw. Upsamplings wird eine Bildmatrix `matrix` mehrmals unter Verwendung einer Transformationsmatrix `transformationMatrix` transformiert. Dies kommt einer Drehscherung gleich. Den Transformationsvorgang übernimmt die Funktion `TransformMatrixShift`. Sie berechnet insbesondere noch die neue Position des `(0,0)`-Elements `newPositionZeroElement` in der transformierten Bildmatrix `transformedMatrix`.

Beispiel 2.1.26: Drehscherung einer Matrix
In der folgenden Matlab-Routine kann man erkennen, wie eine derartige Transformation funktioniert (für detaillierte, bebilderte Beispiele siehe Abschnitt 2.1.4).

```
» matrix = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]

matrix =

1     2     3     4
5     6     7     8
9    10    11    12

» transformationMatrix = [0, 1; 1, -4]

transformationMatrix =

0     1
1    -4

» [transformedMatrix, newPositionZeroElement] = ...
TransformMatrixShift(matrix, transformationMatrix, positionZeroElement)

transformedMatrix =

12    0     0
11    0     0
10    0     0
9     0     0
0     8     0
0     7     0
0     6     0
0     5     0
0     0     4
0     0     3
0     0     2
0     0     1
```

```
newPositionZeroElement =

0     8
```

2.1.6 Verwaltung der Filterbänke

Die Daten der Filterbänke werden im Ordner `'Filterbanks'` in einzelnen Textdateien gespeichert. Alternativ kann auch eine SQLite-Datenbank verwendet werden. Dafür wird die Database-Toolbox von Matlab benötigt. Im Folgenden gehen wir von einer Speicherung in Textdateien aus.

AddFilter.m

```
function AddFilter(name, dilationMatrix, cosets, zeroElementsDecomposition, ...
     decompositionFilters, zeroElementsReconstruction, reconstructionFilters)
```

Die Funktion `AddFilter` fügt eine neue Filterbank zum Filterspeicher hinzu. Dazu wird im Ordner `'Filterbanks'` ein neuer Unterordner mit dem Namen `name` der Filterbank angelegt. Anschließend werden Dilatationsmatrix `dilationsMatrix`, die Cosets der Filtermatrizen `cosets`, die Positionen der `(0,0)`-Elemente der Zerlegung `zeroElementsDecomposition` sowie der Rekonstruktion `zeroElementsReconstruction` und nicht zuletzt die Filtermatrizen `decompositionFilters` und `reconstructionFilters` darin gespeichert.

Beispiel 2.1.27: Einfügen einer Filterbank
Im Folgenden soll die Filterbank `'M01'` in den Filterspeicher eingefügt werden. Die jeweiligen `(0,0)`-Elemente sind rot eingefärbt.

```
» name = 'M01'

name =

M01

» dilationMatrix = [1, 1; 1, -2]

dilationMatrix =

1     1
1    -2

» cosets = [0, 0; 1, 0; 1, -1]
```

```
cosets =

0     0
1     0
1    -1

» zeroElementsDecomposition = [1, 1; 1, 1; 1, 1]

zeroElementsDecomposition =

1     1
1     1
1     1

» zeroElementsReconstruction = [2, 1; 2, 1; 2, 1]

zeroElementsReconstruction =

2     1
2     1
2     1

» Ldec = [0, 0, 0, 0, 0; 0, 1, 0, 0, 0; 0, 0, 0, 0, 0]

Ldec =

0     0     0     0     0
0     1     0     0     0
0     0     0     0     0

» H1dec = [0, 0, 0, 0, 0; 0, -2/3, 1, 0, -1/3; 0, 0, 0, 0, 0]

H1dec =

0         0         0         0         0
0   -0.6667    1.0000         0   -0.3333
0         0         0         0         0

» H2dec = [0, 0, 0, 0, 0; 0, 0, 0, 0, 0; -1/3, 0, 1, -2/3, 0]

H2dec =
```

```
0         0         0         0         0
0         0         0         0         0
-0.3333   0    1.0000   -0.6667         0

» decompositionFilters = {Ldec, H1dec, H2dec}

decompositionFilters =

1×3 cell array

[3×5 double]     [3×5 double]     [3×5 double]

» Lrec = [0, 0, 0, 0, 0; 1/3, 2/3, 1, 2/3, 1/3; 0, 0, 0, 0, 0]

Lrec =

0         0         0         0         0
0.3333    0.6667    1.0000    0.6667    0.3333
0         0         0         0         0

» H1rec = [0, 0, 0, 0, 0; 0, 0, 0, 1, 0; 0, 0, 0, 0, 0]

H1rec =

0     0     0     0     0
0     0     0     1     0
0     0     0     0     0

» H2rec = [0, 0, 0, 0, 0; 0, 0, 0, 0, 0; 0, 0, 0, 1, 0]

H2rec =

0     0     0     0     0
0     0     0     0     0
0     0     0     1     0

» reconstructionFilters = {Lrec, H1rec, H2rec}

reconstructionFilters =

1×3 cell array
```

```
[3×5 double]     [3×5 double]     [3×5 double]

» AddFilter(name, dilationMatrix, cosets, zeroElementsDecomposition,
decompositionFilters, zeroElementsReconstruction, reconstructionFilters)
```

DisplayFilter.m

```
function DisplayFilter(name)
```

Die Funktion `DisplayFilter` gibt alle gespeicherten Daten der Filterbank mit dem Namen `name` auf der Konsole aus. Sie verwendet dazu intern die Funktion `GetFilter` (siehe 2.1.6). Wird kein Filter mit dem angegebenen Namen gefunden, so wird eine Fehlermeldung ausgegeben.

GetFilter.m

```
function [name, dilationMatrix, shiftVectorsDecomposition, ...
     shiftVectorsReconstruction, lowpassDecompositionFilter, ...
     highpassDecompositionFilters, lowpassReconstructionFilter, ...
     highpassReconstructionFilters] = GetFilter(name)
```

Die Funktion `GetFilter` ruft alle Daten der Filterbank mit dem Namen `name` aus dem Filterspeicher ab. Ist kein Filter mit dem angegebenen Namen vorhanden, so wird eine Fehlermeldung ausgegeben.

ListFilters.m

```
function ListFilters()
```

Mit Hilfe der Funktion `ListFilters` kann man überprüfen, welche Filterbänke im Filterspeicher hinterlegt sind. Die Funktion gibt die Namen aller Filter auf der Konsole aus.

Beispiel 2.1.28: Auflistung aller Filterbänke
Die folgende Matlab-Routine listet die Namen aller im Filterspeicher hinterlegten Filterbänke auf.

```
» ListFilters()

E00
E01
M01
M11
```

2.2 Filterdatenbanken

Folgende Filterbänke befinden sich bereits in der Filterdatenbank. Es können nach Belieben neue Filterbänke eingefügt werden (siehe dazu Abschnitt 2.1.6).
Im Folgenden bezeichne D die Dilatationsmatrix, S die Verschiebungsvektoren sowie L, $H1$ und $H2$ die Filtermatrizen der jeweiligen Filterbank.
Die beiden Dilatationsmatrizen D_{M01} und D_{M11} heißen **gemeinsam expandierend** [1]. Gleiches gilt für die Matrizen D_{E00} und D_{E01}. Die zugehörigen Filterbänke können für eine simultane Bildzerlegung mit mehr als nur einem Filter gleichzeitig verwendet werden. Dabei ist darauf zu achten, dass die Dilatationsmatrizen der gleichzeitig verwendeten Filterbänke gemeinsam expandieren (engl. jointly expanding).

2.2.1 M01

Für den Filter `'M01'` ([1]) gilt:

$$D_{M01} = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix} \tag{2.1}$$

$$S_{M01}^{Dec} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 2 \end{pmatrix} \tag{2.2}$$

$$S_{M01}^{Rec} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.3}$$

$$L_{M01}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.4}$$

$$H1_{M01}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -0.6667 & 1 & 0 & -0.3333 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.5}$$

$$H2_{M01}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.3333 & 0 & 1 & -0.6667 & 0 \end{pmatrix} \tag{2.6}$$

$$L_{M01}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.3333 & 0.6667 & 1 & 0.6667 & 0.3333 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.7}$$

$$H1_{M01}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.8}$$

$$H2_{M01}^{Rec} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.9}$$

2.2.2 M11

Für den Filter `'M11'` ([1]) gilt:

$$D_{M11} = \begin{pmatrix} 2 & -1 \\ 1 & -2 \end{pmatrix} \tag{2.10}$$

$$S_{M11}^{Dec} = \begin{pmatrix} 2 & 1 \\ 2 & 0 \\ 3 & 1 \end{pmatrix} \tag{2.11}$$

$$S_{M11}^{Rec} = \begin{pmatrix} 2 & 1 \\ 2 & 2 \\ 1 & 1 \end{pmatrix} \tag{2.12}$$

$$L_{M11}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.13}$$

$$H1_{M11}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.3333 & 0 & 1 & -0.6667 & 0 & 0 \end{pmatrix} \tag{2.14}$$

$$H2_{M11}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.6667 & 1 & 0 & -0.3333 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.15}$$

$$L_{M11}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.3333 & 0.6667 & 1 & 0.6667 & 0.3333 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.16}$$

$$H1_{M11}^{Rec} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.17}$$

$$H2_{M11}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.18}$$

2.2.3 E00

Für den Filter `'E00'` ([4]) gilt:

$$D_{E00} = \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \tag{2.19}$$

$$S_{E00}^{Dec} = \begin{pmatrix} 2 & 2 \\ 3 & 2 \\ 4 & 2 \\ 2 & 3 \\ 3 & 3 \\ 4 & 3 \end{pmatrix} \tag{2.20}$$

$$S_{E00}^{Rec} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \\ 0 & 1 \\ 2 & 0 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{2.21}$$

$$L_{E00}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.22}$$

$$H1_{E00}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.6667 & 1 & 0 & -0.3333 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.23}$$

$$H2_{E00}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.3333 & 0 & 1 & -0.6667 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.24}$$

$$H3_{E00}^{Dec} = \begin{pmatrix} 0 & 0 & -0.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.25}$$

$$H4_{E00}^{Dec} = \begin{pmatrix} 0 & 0 & -0.3333 & 0 & 0 & -0.1667 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -0.3333 & 0 & 0 & -0.1667 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.26}$$

$$H5_{E00}^{Dec} = \begin{pmatrix} 0 & 0 & -0.1667 & 0 & 0 & -0.3333 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -0.1667 & 0 & 0 & -0.3333 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.27}$$

$$L_{E00}^{Rec} = \begin{pmatrix} 0.1667 & 0.3333 & 0.5 & 0.3333 & 0.1667 \\ 0.3333 & 0.6667 & 1 & 0.6667 & 0.3333 \\ 0.1667 & 0.3333 & 0.5 & 0.3333 & 0.1667 \end{pmatrix} \tag{2.28}$$

$$H1_{E00}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.29}$$

$$H2_{E00}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.30}$$

$$H3_{E00}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.31}$$

$$H4_{E00}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{2.32}$$

$$H5_{E00}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.33}$$

2.2.4 E01

Für den Filter `'E01'` ([4]) gilt:

$$D_{E01} = \begin{pmatrix} 3 & -6 \\ 0 & 2 \end{pmatrix} \tag{2.34}$$

$$S_{E01}^{Dec} = \begin{pmatrix} 6 & 1 \\ 7 & 1 \\ 8 & 1 \\ 3 & 2 \\ 4 & 2 \\ 5 & 2 \end{pmatrix} \tag{2.35}$$

$$S_{E01}^{Rec} = \begin{pmatrix} 5 & 1 \\ 4 & 1 \\ 3 & 1 \\ 8 & 0 \\ 7 & 0 \\ 6 & 0 \end{pmatrix} \tag{2.36}$$

$$L_{E01}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.37}$$

$$H1_{E01}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.6667 & 1 & 0 & -0.3333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.38}$$

$$H2_{E01}^{Dec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.3333 & 0 & 1 & -0.6667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.39}$$

$$H3_{E01}^{Dec} = \begin{pmatrix} -0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.40}$$

$$H4_{E01}^{Dec} = \begin{pmatrix} -0.3333 & 0 & 0 & -0.1667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.3333 & 0 & 0 & -0.1667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.41}$$

$$H5_{E01}^{Dec} = \begin{pmatrix} -0.1667 & 0 & 0 & -0.3333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.1667 & 0 & 0 & -0.3333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.42}$$

$$L_{E01}^{Rec} = \begin{pmatrix} 0.1667 & 0.3333 & 0.5 & 0.3333 & 0.1667 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3333 & 0.6667 & 1 & 0.6667 & 0.3333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1667 & 0.3333 & 0.5 & 0.3333 & 0.1667 \end{pmatrix} \tag{2.43}$$

$$H1_{E01}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.44}$$

$$H2_{E01}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.45}$$

$$H3_{E01}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.46}$$

$$H4_{E01}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{2.47}$$

$$H5_{E01}^{Rec} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.48}$$

2.3 Behandlung des Verschiebungsvektors

Während der Zerlegung und anschließenden Rekonstruktion ist es maßgeblich von Bedeutung, das `(0,0)`-Element jeder Bildmatrix zu speichern. Denn durch Faltung und Downsampling werden zusammengehörige Bildmatrizen ungleich verschoben. Bei der Rekonstruktion müssen der Tiefpassanteil und die verschiedenen Hochpassanteile passend übereinandergelegt werden. Dazu werden die Matrizen am `(0,0)`-Element ausgerichtet. Unglücklicherweise kann dieses etwa beim Downsampling sogar verschwinden, ein anderes Element muss seinen Platz einnehmen. Im entsprechenden Rekonstruktionsschritt muss dies berücksichtigt und rückgängig gemacht werden. Die Filterfunktionen kommen also nicht ohne einen gewissen Overhead an Daten aus. Diese Verwaltungsdaten besitzen jedoch konstante Speicherplatzkomplexität, sind also in keinster Weise von der Größe oder Art des analysierten Bildes abhängig.

2.3.1 Verwaltungsdaten

Der Inhalt einer Zelle des Decomposition-Cell-Arrays ist gemäß Abbildung 2.24 festgelegt. Dabei enthält `lowpassData` erneut ein Cell-Array mit dem tatsächlichen Tiefpass sowie der Position des `(0,0)`-Elements. Zusätzlich wird bei Bedarf gespeichert, ob just in diesem Schritt das `(0,0)`-Element beim Downsampling verloren gegangen und durch ein neues ersetzt worden ist.
Exakt dieselben Informationen werden für jeden einzelnen Hochpass gespeichert.
In der Zelle `filterHistory` werden alle bisher auf das Bild angewandte Filter in der entsprechenden Reihenfolge gespeichert, sodass bei der Rekonstruktion die passenden Rekonstruktionsfiltermatrizen ausgewählt werden können. Die Zelle `filterPath` wird benötigt, um bei der gleichzeitigen Verwendung mehrerer Filterbänke bei der Rekonstruktion die passenden Hochpassinformationen zum rekonstruierten Tiefpass zu finden. In der letzten Zelle `imageSize` schließlich wird die ursprüngliche Bildgröße vor der Zerlegung gespeichert, sodass nach dem zugehörigen Rekonstruktionsschritt die Nullzeilen und -spalten korrekt entfernt werden können.

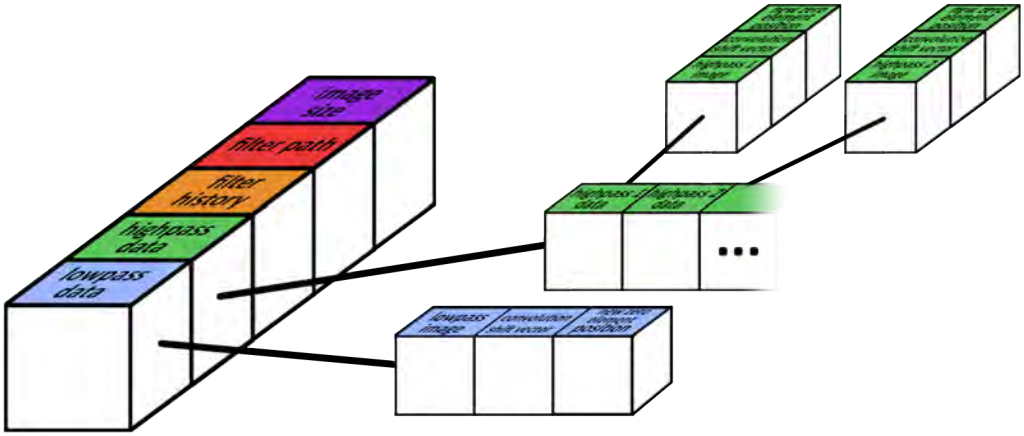

Abbildung 2.24: Die Ergebnisse eines jeden Zerlegungsschrittes werden in der entsprechenden Zelle des Cell-Arrays gespeichert. Zusätzlich werden Overhead-Daten wie etwa die Position des `(0,0)`-Elements gespeichert, um später eine perfekte Rekonstruktion zu ermöglichen.

2.3.2 Bildzerlegung

Für ein tiefer gehendes Verständnis über die Speicherung und Aktualisierung der Position des `(0,0)`-Elements in jedem Schritt, wird eine Bildmatrix beispielhaft zerlegt und wieder zusammengefügt.
Seien dazu unsere Bildmatrix M und der Filtername F gegeben durch

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 \end{pmatrix} \text{ und} \tag{2.49}$$

$$F =' M01'. \tag{2.50}$$

Als Zerlegungstiefe wählen wir $L = 1$. Jeder weitere Zerlegungsschritt erfolgt völlig analog zu diesem mit dem Tiefpassergebnis des vorherigen Schrittes als Ausgangsbild.

```
decArray = WaveletDecomposition(M, F, L);
```

Dieser Befehl ruft die Funktion `WaveletDecomposition` (siehe Abschnitt 2.1.2) auf, welche mit der Vorbereitung des übergebenen Bildes auf die Zerlegung beginnt.

Vorbereitung des Bildes

Je nach Anzahl der gewählten Filter und gewünschter Dekompositionstiefe wird ein passendes Cell-Array erstellt, in welches die zerlegten Bildinformationen an den jeweiligen Stellen geschrieben werden.

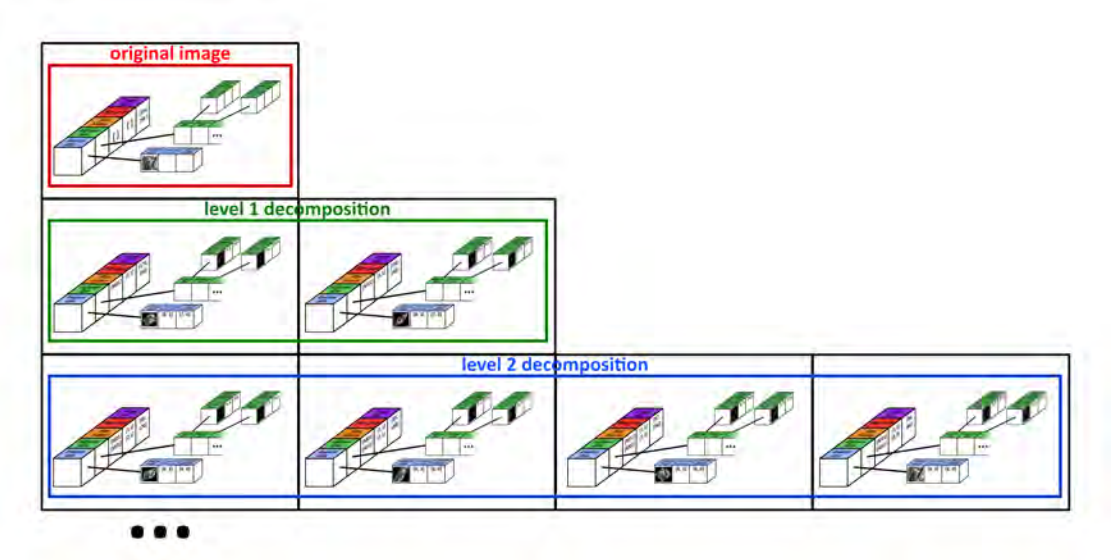

Abbildung 2.25: Für die Dekomposition des Bildes wurden die beiden Filter $'M01'$ (siehe Abschnitt 2.2.1) und $'M11'$ (siehe Abschnitt 2.2.2) benutzt. In jedem Schritt wird jeder Tiefpass mit beiden Filtern zerlegt. Die Anzahl der Filterergebnisse steigt exponentiell zur Tiefe.

Jede Zelle dieses Cell-Arrays enthält ein weiteres Cell-Array mit den Bildinformationen und Metadaten. Für das Originalbild wird die erste Zelle passend beschrieben. Als `(0,0)`-Element identifizieren wir das linkeste, unterste Pixel des Bildes. Das Bild selbst liegt damit im ersten Quadranten eines kartesischen Koordinatensystems mit dem `(0,0)`-Element im Ursprung.

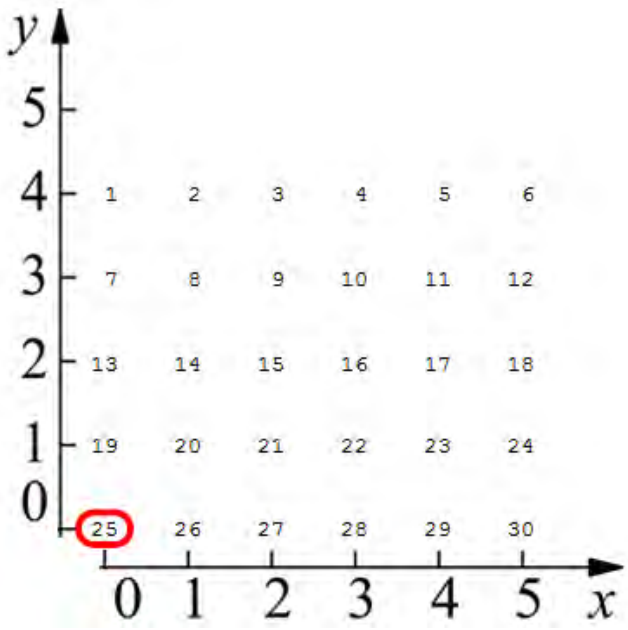

Abbildung 2.26: Das Originalbild liegt im ersten Quadranten eines kartesischen Koordinatensystems und hat als `(0,0)`-Element das im Ursprung liegende Pixel.

In die erste Zelle wird das Originalbild geschrieben. Die Einträge für die Hochpassinformationen werden gleich Null gesetzt und das `(0,0)`-Element liegt im Koordinatenursprung `(0,0)`.
Nun wird intern die Funktion `WaveletDecomposition_1Level` (siehe Abschnitt 2.1.2) aufgerufen, welche für die einzige von uns gewählte Filterbank $M01$ (siehe Abschnitt 2.2.1) einen Zerlegungsschritt der Bildmatrix durchführt.

Abrufen der Filterbank

Die Dilatationsmatrix sowie die Verschiebungsvektoren und die Zerlegungsfiltermatrizen werden aus der Datenbank geholt.
Seien also weiterhin die Dilatationsmatrix D und die Matrix der Verschiebungsvektoren S_{Dec} für die Zerlegung bestimmt durch

$$D = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix} \text{ und} \tag{2.51}$$

$$S_{Dec} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 2 \end{pmatrix}. \tag{2.52}$$

Die Tiefpassfiltermatrix $L_{DecFilter}$ und die beiden Hochpassfiltermatrizen $H1_{DecFilter}$

und $H2_{DecFilter}$ sind

$$L_{DecFilter} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.53}$$

$$H1_{DecFilter} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -0.66667 & 1 & 0 & -0.33333 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.54}$$

$$H2_{DecFilter} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.33333 & 0 & 1 & -0.66667 & 0 \end{pmatrix}. \tag{2.55}$$

Anschließend kann mit der Faltung der Bildmatrix mit den Dekompositionsfiltern begonnen werden.

Faltung mit den Filtermatrizen

Das Originalbild wird mit dem Tiefpassfilter sowie den beiden Hochpassfiltern gefaltet. Will man im Anschluss eine Hochpassrekonstruktion zur Kantenerkennung durchführen, so übergibt man den optionalen Parameter `'expandEdges'`, der dafür sorgt, dass die Faltungsartefakte an den Bildrändern verschwinden. Als Faltungsergebnis erhält man für den Tiefpassfilter L_{Dec}^{Conv}, für die beiden Hochpassfilter $H1_{Dec}^{Conv}$ und $H2_{Dec}^{Conv}$ und die entsprechenden Positionen $L_{ZeroPos}^{Conv}$, $H1_{ZeroPos}^{Conv}$ und $H2_{ZeroPos}^{Conv}$ der $(0,0)$-Elemente.

$$L_{Dec}^{Conv} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 0 & 0 & 0 \\ 0 & 7 & 8 & 9 & 10 & 11 & 12 & 0 & 0 & 0 \\ 0 & 13 & 14 & 15 & 16 & 17 & 18 & 0 & 0 & 0 \\ 0 & 19 & 20 & 21 & 22 & 23 & 24 & 0 & 0 & 0 \\ 0 & 25 & 26 & 27 & 28 & 29 & 30 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.56}$$

$$L_{ZeroPos}^{Conv} = (1,1) \tag{2.57}$$

$$H1_{Dec}^{Conv} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.66667 & -0.33334 & 0 & 0 & 0 & 0 & 4.6667 & -1.6667 & -2 \\ 0 & -4.6667 & 1.6666 & 2 & 0 & 0 & 0 & 8.6667 & -3.6666 & -4 \\ 0 & -8.6667 & 3.6666 & 4 & 0 & 0 & 0 & 12.6667 & -5.6666 & 6 \\ 0 & -12.6667 & 5.6666 & 6 & 0 & 0 & 0 & 16.6667 & -7.6666 & 8 \\ 0 & -16.6668 & 7.6666 & 8 & 0 & 0 & 0 & 20.6668 & -9.6666 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.58}$$

$$H1_{ZeroPos}^{Conv} = (2,1) \tag{2.59}$$

$$H2_{Dec}^{Conv} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.3333 & -0.6667 & 0 & 0 & 0 & 0 & 2.3333 & 2.6667 & -4 & 0 \\ -2.3333 & -2.6667 & 4 & 0 & 0 & 0 & 4.3333 & 4.6667 & -8 & 0 \\ -4.3333 & -4.6667 & 8 & 0 & 0 & 0 & 6.3333 & 6.6667 & -12 & 0 \\ -6.3333 & -6.6667 & 12 & 0 & 0 & 0 & 8.3333 & 8.6667 & -16 & 0 \\ -8.3333 & -8.6667 & 16 & 0 & 0 & 0 & 10.3333 & 10.6667 & -20 & 0 \end{pmatrix} \tag{2.60}$$

$$H2_{ZeroPos}^{Conv} = (2, 2) \tag{2.61}$$

Transformation und Downsampling

Anschließend werden die gefalteten Bildinformationen einer Transformation und Downsampling gemäß der Dilatationsmatrix D (siehe 2.51) unterzogen.

Für die Tiefpassinformationen gilt dann nach der ersten Transformation

$$L_{Dec}^{Transform1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 0 & 0 \\ 0 & 29 & 0 & 0 & 0 & 0 & 0 \\ 0 & 28 & 0 & 0 & 0 & 0 & 0 \\ 0 & 27 & 0 & 0 & 0 & 0 & 0 \\ 0 & 26 & 24 & 0 & 0 & 0 & 0 \\ 0 & 25 & 23 & 0 & 0 & 0 & 0 \\ 0 & 0 & 22 & 0 & 0 & 0 & 0 \\ 0 & 0 & 21 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 18 & 0 & 0 & 0 \\ 0 & 0 & 19 & 17 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 14 & 12 & 0 & 0 \\ 0 & 0 & 0 & 13 & 11 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 6 & 0 \\ 0 & 0 & 0 & 0 & 7 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.62}$$

$$L_{ZeroPos}^{Transform1} = (1, 21) \tag{2.63}$$

Im nächsten Schritt wird ein Downsampling durchgeführt, was zur folgenden Bildmatrix führt.

$$L_{Dec}^{Downsampling} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 28 & 0 & 0 & 0 & 0 & 0 \\ 0 & 25 & 23 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 18 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 7 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.64}$$

$$L_{ZeroPos}^{Downsampling} = (1, 7) \tag{2.65}$$

Die anschließende Transformation ergibt nun

$$L_{Dec}^{Transform2} = \begin{pmatrix} 0 & 0 & 28 & 23 & 18 \\ 25 & 20 & 15 & 10 & 5 \\ 0 & 7 & 2 & 0 & 0 \end{pmatrix} \tag{2.66}$$

$$L_{ZeroPos}^{Transform2} = (0, 1). \tag{2.67}$$

Das `(0,0)`-Element ist beim Downsampling nicht weggefallen, dieser Fall muss also nicht gesondert betrachtet werden. Für die Hochpassinformationen des ersten Hochpassfilters

folgt dann

$$H1_{Dec}^{Transform1} = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -10 & 0 & 0 & 0 & 0 & 0 \\
0 & -9.6667 & 0 & 0 & 0 & 0 & 0 \\
0 & 20.6667 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -8 & 0 & 0 & 0 & 0 \\
0 & 0 & -7.6667 & 0 & 0 & 0 & 0 \\
0 & 8 & 16.6667 & 0 & 0 & 0 & 0 \\
0 & 7.6667 & 0 & 0 & 0 & 0 & 0 \\
0 & -16.6667 & 0 & -6 & 0 & 0 & 0 \\
0 & 0 & 0 & -5.6667 & 0 & 0 & 0 \\
0 & 0 & 6 & 12.6667 & 0 & 0 & 0 \\
0 & 0 & 5.6667 & 0 & 0 & 0 & 0 \\
0 & 0 & -12.6667 & 0 & -4 & 0 & 0 \\
0 & 0 & 0 & 0 & -3.6667 & 0 & 0 \\
0 & 0 & 0 & 4 & 8.6667 & 0 & 0 \\
0 & 0 & 0 & 3.6667 & 0 & 0 & 0 \\
0 & 0 & 0 & -8.6667 & 0 & -2 & 0 \\
0 & 0 & 0 & 0 & 0 & -1.6667 & 0 \\
0 & 0 & 0 & 0 & 2 & 4.6667 & 0 \\
0 & 0 & 0 & 0 & 1.6667 & 0 & 0 \\
0 & 0 & 0 & 0 & -4.6667 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.3333 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.6667 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix} \tag{2.68}$$

$$H1_{ZeroPos}^{Transform1} = (1, 22). \tag{2.69}$$

Beim anschließenden Downsampling fällt das `(0,0)`-Element weg. Deshalb muss ein neues gewählt und diese Wahl in Form eines Verschiebungsvektors ZV in den Metadaten vermerkt werden. Hier wird das nächste Element darüber gewählt, das beim Downsampling

erhalten bleibt.

$$H1_{Dec}^{Downsampling} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -7.6667 & 0 & 0 & 0 & 0 \\ 0 & -16.6667 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 5.6667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 8.6667 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1.6667 & 0 \\ 0 & 0 & 0 & 0 & -4.6667 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.3333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.70}$$

$$H1_{ZeroPos}^{Downsampling} = (1, 8) \text{ und } H1_{ZeroShift}^{Downsampling} = (0, -2) \tag{2.71}$$

Schließlich erfolgt die Rücktransformation und das Entfernen von Nullzeilen und Spalten.

$$H1_{Dec}^{Transform2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 20.6667 & -7.6667 & -6 \\ 0 & 0 & 0 & 0 & 0 & 8.6667 & -1.6667 \\ -16.6667 & 5.6667 & 4 & 0 & 0 & 0 & 0 \\ 0 & -4.6667 & -0.3333 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.72}$$

$$H1_{ZeroPos}^{Transform2} = (2, 2) \text{ und } H1_{ZeroShift}^{Transform2} = (0, -2). \tag{2.73}$$

Dasselbe erfolgt nun noch für den zweiten Hochpassfilter.

$$H2_{Dec}^{Transform1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -20 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10.6667 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10.3333 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -16 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8.6667 & 0 & 0 & 0 & 0 & 0 \\ 16 & 8.3333 & 0 & 0 & 0 & 0 & 0 \\ -8.6667 & 0 & 0 & 0 & 0 & 0 & 0 \\ -8.3333 & 0 & -12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6.6667 & 0 & 0 & 0 & 0 \\ 0 & 12 & 6.3333 & 0 & 0 & 0 & 0 \\ 0 & -6.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & -6.3333 & 0 & -8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4.6667 & 0 & 0 & 0 \\ 0 & 0 & 8 & 4.3333 & 0 & 0 & 0 \\ 0 & 0 & -4.6667 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4.3333 & 0 & -4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.6667 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2.3333 & 0 & 0 \\ 0 & 0 & 0 & -2.6667 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2.3333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.6667 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.3333 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.74}$$

$$H2_{ZeroPos}^{Transform1} = (2, 18) \tag{2.75}$$

$$H2_{Dec}^{Downsampling} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10.3333 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 8.6667 & 0 & 0 & 0 & 0 & 0 \\ -8.3333 & 0 & -12 & 0 & 0 & 0 & 0 \\ 0 & -6.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 4.3333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.6667 & 0 & 0 \\ 0 & 0 & 0 & -2.3333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.6667 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.76}$$

$$H2_{ZeroPos}^{Downsampling} = (2, 6) \tag{2.77}$$

$$H2_{Dec}^{Transform2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 10.3333 & 8.6667 & -12 \\ 0 & 0 & 0 & 0 & 0 & 4.3333 & 2.6667 \\ -8.3333 & -6.6667 & 8 & 0 & 0 & 0 & 0 \\ 0 & -2.3333 & -0.6667 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.78}$$

$$H2_{ZeroPos}^{Transform2} = (2, 1) \tag{2.79}$$

2.3.3 Bildrekonstruktion

Mit Hilfe der drei Teilbilder und den jeweiligen Positionen des $(0, 0)$-Elements kann nun das ursprüngliche Bild wieder vollständig hergestellt werden.

```
newImage = WaveletReconstruction(decArray, [2,1]);
```

Vorbereitungen und Abrufen der Filterbank

Für die Rekonstruktion werden zuerst ein paar Vorbereitungen getroffen. Aus dem Cell-Array wird der verwendete Filter ausgelesen. Anschließend werden die Rekonstruktionsfiltermatrizen $L_{RecFilter}$, $H1_{RecFilter}$ und $H2_{RecFilter}$ sowie die Dilatationsmatrix D und die Verschiebungsvektoren S_{Rec} für die Rekonstruktion aus der Datenbank abgerufen.

$$D = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix} \tag{2.80}$$

$$S_{Rec} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.81}$$

$$L_{RecFilter} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.3333 & 0.6667 & 1 & 0.6667 & 0.3333 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.82}$$

$$H1_{RecFilter} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.83}$$

$$H2_{RecFilter} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.84}$$

Transformation und Upsampling

Analog zur Dekomposition werden für die Rekonstruktion zwei Transformationsschritte und in diesem Fall ein Upsamplingschritt benötigt.
Wir beginnen mit den Transformationen und dem Upsampling des Tiefpassbildanteils. Sollen die Teilbildmatrizen einem Thresholding unterzogen werden, dann wird die Auswahl der Hochpasskoeffizienten für die Rekonstruktion in diesem Schritt durchgeführt. Soll die Rekonstruktion ausschließlich aus den Hochpassinformationen gestartet werden, um Kanten im Bild zu lokalisieren, so werden hier in der untersten Zerlegungsebene die Tiefpassinformationen weggeworfen.

$$L_{Rec}^{Transform1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 25 & 23 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 18 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.85}$$

$$L_{ZeroPos}^{Transform1} = (2, 7) \tag{2.86}$$

$$L_{Rec}^{Upsampling} = \begin{pmatrix} 28 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 25 & 23 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 18 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.87}$$

$$L_{ZeroPos}^{Upsampling} = (0, 17) \tag{2.88}$$

$$L_{Rec}^{Transform2} = \begin{pmatrix} 0 & 2 & 0 & 0 & 5 & 0 \\ 7 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 & 18 \\ 0 & 20 & 0 & 0 & 23 & 0 \\ 25 & 0 & 0 & 28 & 0 & 0 \end{pmatrix} \tag{2.89}$$

$$L_{ZeroPos}^{Transform2} = (0, 0) \tag{2.90}$$

Als nächstes folgt die Rekonstruktion des ersten Hochpassbildanteils.

$$H1_{Rec}^{Transform1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20.6667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -7.6667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -16.6667 & 0 & -6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5.6667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 8.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.6667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.3333 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.91}$$

$$H1_{ZeroPos}^{Transform1} = (4, 10) \tag{2.92}$$

$$H1_{Rec}^{Upsampling} = \begin{pmatrix} 20.6667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -7.6667 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -16.6667 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 5.6667 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 8.6667 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1.6667 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4.6667 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.3333 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.93}$$

$$H1_{ZeroPos}^{Upsampling} = (0, 18) \tag{2.94}$$

$$H1_{Rec}^{Transform2} = \begin{pmatrix} 0 & -0.3333 & 0 & 0 & 0 & 0 & 0 & -1.6667 & 0 \\ -4.6667 & 0 & 0 & 0 & 0 & 0 & 8.6667 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & -6 \\ 0 & 5.6667 & 0 & 0 & 0 & 0 & 0 & -7.6667 & 0 \\ -16.6667 & 0 & 0 & 0 & 0 & 0 & 20.6667 & 0 & 0 \end{pmatrix} \tag{2.95}$$

$$H1_{ZeroPos}^{Transform2} = (1, 0) \tag{2.96}$$

Und schließlich erfolgt die Rekonstruktion aus dem zweiten und letzten Hochpass.

$$H2_{Rec}^{Transform1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10.3333 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8.6667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -8.3333 & 0 & -12 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -6.6667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & 4.3333 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.6667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.3333 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.6667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.97}$$

$$H2_{ZeroPos}^{Transform1} = (6, 7) \tag{2.98}$$

$$H2_{Rec}^{Upsampling} = \begin{pmatrix} 10.3333 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 8.6667 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -8.3333 & 0 & -12 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -6.6667 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 4.3333 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.6667 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2.3333 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.6667 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.99}$$

$$H2_{ZeroPos}^{Upsampling} = (2, 11) \tag{2.100}$$

$$H2_{Rec}^{Transform2} = \begin{pmatrix} 0 & -0.6667 & 0 & 0 & 0 & 0 & 0 & 2.6667 & 0 \\ -2.3333 & 0 & 0 & 0 & 0 & 0 & 4.3333 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & -12 \\ 0 & -6.6667 & 0 & 0 & 0 & 0 & 0 & 8.6667 & 0 \\ -8.3333 & 0 & 0 & 0 & 0 & 0 & 10.3333 & 0 & 0 \end{pmatrix} \tag{2.101}$$

$$H2_{ZeroPos}^{Transform2} = (2, 2) \tag{2.102}$$

Faltung mit den Filtermatrizen

Vor dem Zusammenfügen der einzelnen Bildanteile zum ursprünglichen Bild müssen die Teilbilder jeweils noch mit den entsprechenden Rekonstruktionsfiltermatrizen gefaltet werden.

Für den TiefpassanteilL_{Rec}^{Conv} nach der Faltung gilt nun:

$$L_{Rec}^{Conv} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.66666 & 1.3333 & 2 & 3 & 4 & 5 & 3.3333 & 1.6667 & 0 \\ 2.3333 & 4.6667 & 7 & 8 & 9 & 10 & 6.6667 & 3.3333 & 0 & 0 \\ 0 & 0 & 5 & 10 & 15 & 16 & 17 & 18 & 12 & 6 \\ 0 & 6.6667 & 13.3333 & 20 & 21 & 22 & 23 & 15.3333 & 7.6667 & 0 \\ 8.3332 & 16.6667 & 25 & 26 & 27 & 28 & 18.6667 & 9.3333 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.103}$$

$$L_{ZeroPos}^{Conv} = (2,1) \tag{2.104}$$

Analog folgt für die Hochpassanteile:

$$H1_{Rec}^{Conv} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.3333 & 0 & 0 & 0 & 0 & 0 & -1.6667 & 0 & 0 & 0 & 0 \\ 0 & -4.6667 & 0 & 0 & 0 & 0 & 0 & 8.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 5.6667 & 0 & 0 & 0 & 0 & 0 & -7.6667 & 0 & 0 & 0 & 0 \\ 0 & -16.6667 & 0 & 0 & 0 & 0 & 0 & 20.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.105}$$

$$H1_{ZeroPos}^{Conv} = (2,1) \tag{2.106}$$

$$H2_{Rec}^{Conv} = \begin{pmatrix} 0 & 0 & -0.6667 & 0 & 0 & 0 & 0 & 0 & 2.6667 & 0 & 0 & 0 & 0 \\ 0 & -2.3333 & 0 & 0 & 0 & 0 & 0 & 4.3333 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & -12 & 0 & 0 & 0 \\ 0 & 0 & -6.6667 & 0 & 0 & 0 & 0 & 0 & 8.6667 & 0 & 0 & 0 & 0 \\ 0 & -8.3333 & 0 & 0 & 0 & 0 & 0 & 10.3333 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.107}$$

$$H2_{ZeroPos}^{Conv} = (3,2) \tag{2.108}$$

Zusammenfügen der Bildanteile

Schließlich werden die einzelnen Bildanteile am $(0,0)$-Element ausgerichtet und zum Originalbild M_{new} addiert.

$$L_{Rec}^{Reposition} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6667 & 1.3333 & 2 & 3 & 4 & 5 & 3.3333 & 1.6667 & 0 & 0 & 0 & 0 \\ 2.3333 & 4.6667 & 7 & 8 & 9 & 10 & 6.6667 & 3.3333 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 10 & 15 & 16 & 17 & 18 & 12 & 6 & 0 & 0 & 0 \\ 0 & 6.6666 & 13.3333 & 20 & 21 & 22 & 23 & 15.3333 & 7.6667 & 0 & 0 & 0 & 0 \\ 8.3333 & 16.6667 & 25 & 26 & 27 & 28 & 18.6667 & 9.3333 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.109}$$

$$H1_{Rec}^{Reposition} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.3333 & 0 & 0 & 0 & 0 & 0 & -1.6667 & 0 & 0 & 0 & 0 \\ 0 & -4.6667 & 0 & 0 & 0 & 0 & 0 & 8.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 5.6667 & 0 & 0 & 0 & 0 & 0 & -7.6667 & 0 & 0 & 0 & 0 \\ 0 & -16.6667 & 0 & 0 & 0 & 0 & 0 & 20.6667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.110}$$

$$H2_{Rec}^{Reposition} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.6667 & 0 & 0 & 0 & 0 & 0 & 2.6667 & 0 & 0 & 0 & 0 & 0 \\ -2.3333 & 0 & 0 & 0 & 0 & 0 & 4.3333 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & -12 & 0 & 0 & 0 & 0 \\ 0 & -6.6667 & 0 & 0 & 0 & 0 & 0 & 8.6667 & 0 & 0 & 0 & 0 & 0 \\ -8.3333 & 0 & 0 & 0 & 0 & 0 & 10.3333 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.111}$$

Nach der Addition der Bildanteile und dem Entfernen der Nullzeilen und -spalten erhalten wir exakt die ursprüngliche Bildmatrix M_{new}.

$$M_{new} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \\ 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 \end{pmatrix} \tag{2.112}$$

2.4 Statistiken und Ergebnisse

Jeder verwendete Filter verspricht eine perfekte Rekonstruktion der Bilder. Allerdings eignet sich nicht jede Filterbank gleichermaßen gut für die Kantenerkennung oder die Bildkompression durch Thresholding. Im Folgenden wurden die verschiedenen Filterbänke auf Bilder aus dem Universitätsalltag angewandt und die Ergebnisse verglichen.
Sämtliche Abbildungen werden zudem hochauflösend sowie als Matlab-Figure für die detaillierte Betrachtung mitgeliefert.

2.4.1 Kantenerkennung

Folgende empirische Aussagen können bezüglich der Kantenerkennung bei der Rekonstruktion allein aus den Hochpassinformationen getroffen werden. Sämtliche Kantenerkennungsprozesse werden auf einer einzigen Bildebene durchgeführt. Bei RGB-Bildern kann ein beliebiger Layer verwendet werden. Grundsätzlich ermöglichen es die Matlab-Funktionen, die Kantenerkennung auf den einzelnen Bildlayern separat durchzuführen und die Ergebnisse im Anschluss zu addieren. Dazu wird einfach das Originalbild mit allen Layern als Funktionsparameter übergeben.

Rekonstruktionstiefe

Das Ergebnis einer Kantenerkennungsprozedur hängt maßgeblich von der Zerlegungs- bzw. Rekonstruktionstiefe ab. Dabei gilt insbesondere **nicht** die naheliegende Regel *'Je tiefer das Zerlegungslevel, desto besser die Kantenerkennung'*. Wird eine zu große Rekonstruktionstiefe gewählt, so verschwimmen die Kanten durch die vielen Faltungen mit den Filtermatrizen. Wird die Rekonstruktionstiefe zu gering gewählt, so werden unter Umständen nur wenige Kanten im Bild erkannt.
Die gewählte Tiefe ist mitunter von der Größe des Originalbildes abhängig.
In den Abbildungen 2.27 und 2.28 ist zu erkennen, dass unter Verwendung der Filterbank `'MO1'` eine Rekonstruktionstiefe von2, 3 oder 4 bei gängigen Bildgrößen (Seitenlängen zwischen 500 und 5000 Pixeln) jeweils für optimale Ergebnisse sorgt. Bei zu hohen Rekonstruktionstiefen nimmt man das unerwünschte Verschwimmen der Kanten sehr deutlich wahr.

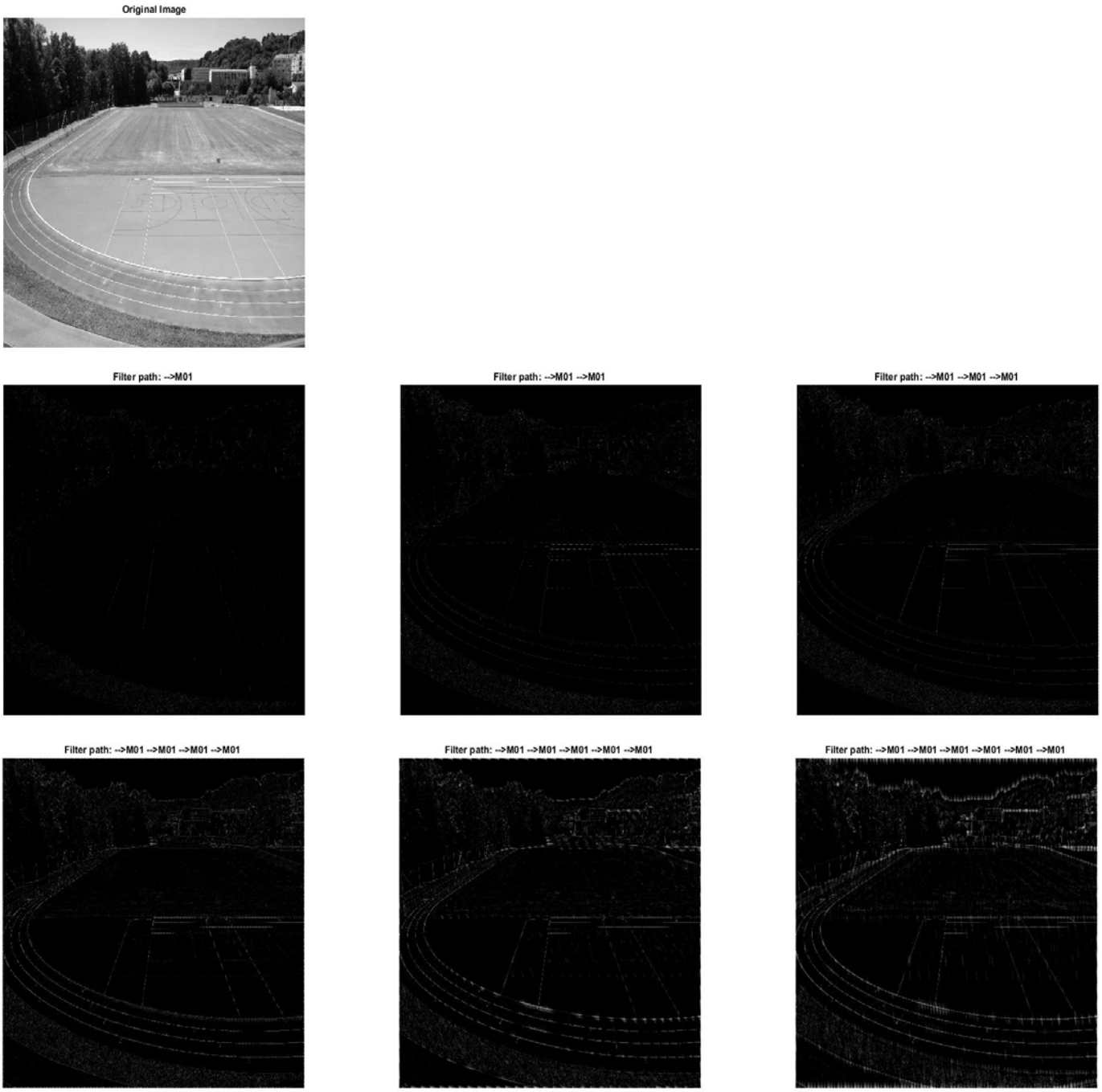

Abbildung 2.27: Das Bild `'sportsfield.jpg'` (Originalgröße: 3888×2592 Pixel) wurde unter Verwendung der Filterbank `'M01'` bis zu einer Tiefe von 6 zerlegt. Aus jedem Tiefenlevel wurde eine Hochpassrekonstruktion zur Kantenerkennung gestartet.

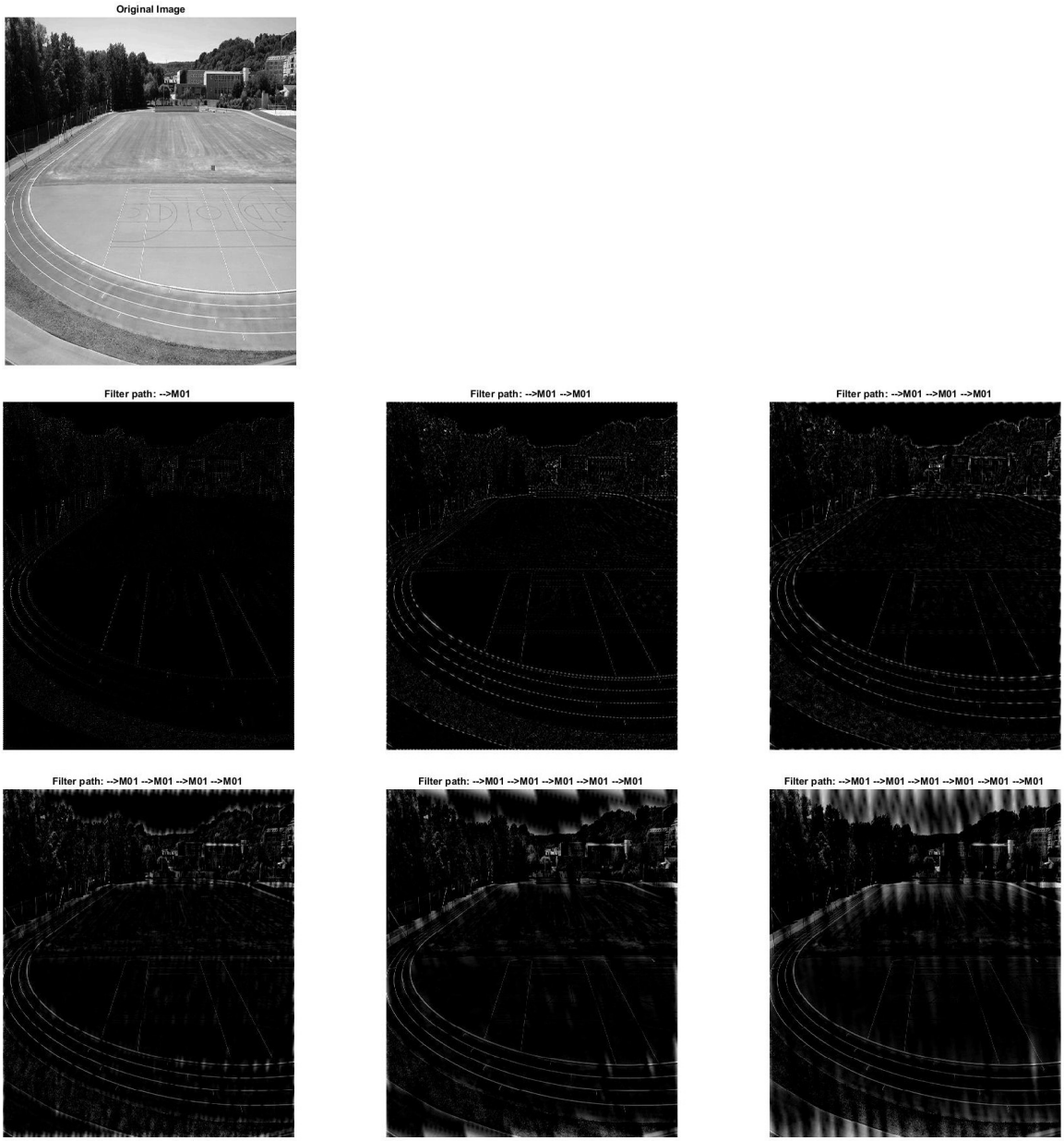

Abbildung 2.28: Eine verkleinerte Version des Bildes `'sportsfield.jpg'` (Neue Größe: 800×533 Pixel) wurde unter Verwendung der Filterbank `'MO1'` bis zu einer Tiefe von 6 zerlegt. Aus jedem Tiefenlevel wurde eine Hochpassrekonstruktion zur Kantenerkennung gestartet.

Zwischen den beiden Filterbänken `'MO1'` und `'M11'` besteht nur ein geringer Unterschied bei der Kantenerkennung (vergleiche dazu die Abbildungen 2.27 und 2.29). Verschiedene Experimente mit den unterschiedlichsten Testbildern haben ergeben, dass sich bei Ver-

wendung von lediglich einer einzigen Filterbank der Filter `'M11'` tendenziell etwas besser zur Kantenerkennung eignet.

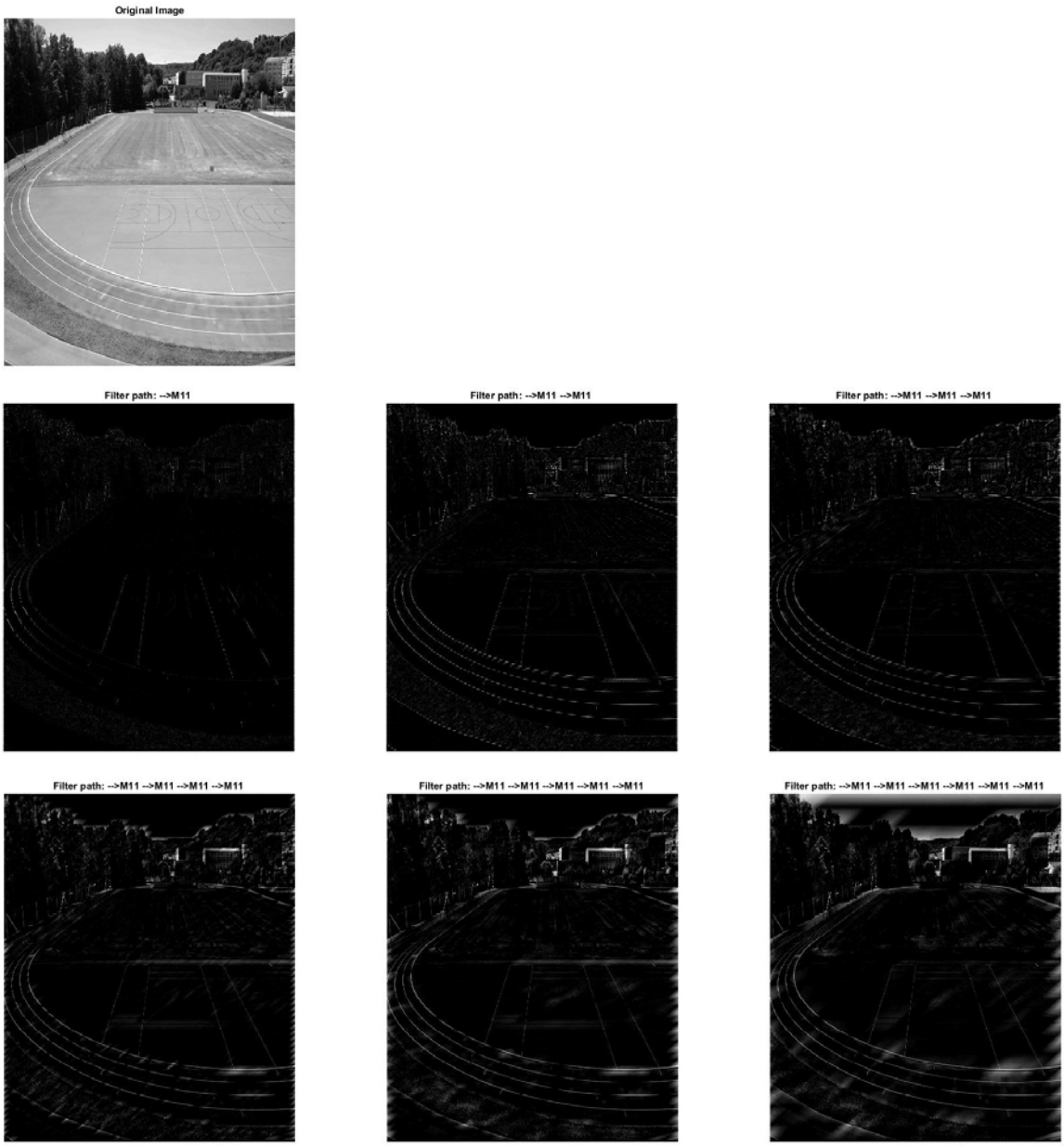

Abbildung 2.29: Das Bild `'sportsfield.jpg'` wurde unter Verwendung der Filterbank `'M11'` bis zu einer Tiefe von 6 zerlegt. Aus jedem Tiefenlevel wurde eine Hochpassrekonstruktion zur Kantenerkennung gestartet.

Rekonstruktionspfade

Im Folgenden wird versucht, einen optimalen Filterpfad für Kantenerkennungsprozesse mit einer Zerlegungstiefe von 3 zu bestimmen.
Wie aus zahlreichen empirischen Ergebnissen hervorgeht, wäre ein optimaler Pfad zur Kantenerkennung unter Verwendung der Filterbänke `'M01'` und `'M11'` gegeben durch:

- `'M11'` >> `'M01'` >> `'M11'`

Für die Filterbänke `'E00'` und `'E01'` wiederum wäre folgender Pfad optimal für die Kantenerkennung, wobei grundsätzlich keine großen Unterschiede zwischen den einzelnen Rekonstruktionspfaden festgestellt werden können.

- `'E00'` >> `'E01'` >> `'E00'`

2.4.2 Thresholding (PSNR-Quality-Check)

Um die Qualität eines rekonstruierten Bildes I_{new} im Hinblick auf das Originalbild I_{orig} beurteilen zu können, wird die Peak signal-to-noise ratio ($PSNR$) verwendet [4].

$$PSNR = 10 \cdot log_{10}\left(\frac{255^2}{MSE(I_{orig}, I_{new})}\right) \tag{2.113}$$

$$MSE(I_{orig}, I_{new}) = \frac{1}{rows \cdot cols} \cdot \sum_{r=1}^{rows} \sum_{c=1}^{cols} \big(I_{orig}(r,c) - I_{new}(r,c)\big)^2 \tag{2.114}$$

Für die nachfolgende Analyse werden Fehlerbild I_{err} und Quotientenbild I_{quot} berechnet.[4]

$$I_{err}(r,c) = \frac{\big|I_{orig}(r,c) - I_{new}(r,c)\big|}{255} \tag{2.115}$$

$$I_{quot}(r,c) = \left|\frac{I_{new}(r,c)}{I_{orig}(r,c)}\right| \tag{2.116}$$

Rekonstruktionspfade

Das Bild `'boots.jpg'` wird mit Hilfe der Filterbänke `'M01'` und `'M11'` bis zu einer maximalen Tiefe von 3 zerlegt. Anschließend wird für jeden Filterpfad eine Rekonstruktion mit verschiedenen Thresholdeinstellungen gestartet. In den nachfolgenden Tabellen 2.1 bis 2.4 wird jeweils die Anzahl der für die Rekonstruktion benötigten Elemente ungleich Null (non-zero elements, NZE), $PSNR$ sowie der mittlere quadratische Fehler (mean-square-error, MSE) dargestellt. Die Spalte Speicherplatzersparnis (storage space savings, SSS) berechnet die maximale mögliche Einsparung von Speicherplatz, sofern die einzelnen dünn besiedelten Hochpassmatrizen optimal mittels Lauflängenkodierung oder ähnlichen Verfahren gespeichert werden können.
Zum Vergleich: das Originalbild hat eine Größe von $972px \times 1.296px$ und besitzt 1.259.712 von Null verschiedene Werte. In den Abbildungen 2.30 bis 2.41 ist für die Werte $t \in$

$\{1, 5, 10, 20\}$ das aus dem Pfad 'M01'>'M01'>'M01' rekonstruierte Bild sowie das zugehörige Fehlerbild und Quotientenbild zu sehen.

Filterpfad	Threshold 1				Threshold 2			
	NZE	SSS	PSNR	MSE	NZE	SSS	PSNR	MSE
>M01>M01>M01	453280	64.02%	56.36	0.1503	351728	72.08%	49.54	0.7230
>M01>M01>M11	453360	64.01%	56.35	0.1506	351784	72.07%	49.52	0.7255
>M01>M11>M01	454515	63.92%	56.41	0.1486	353788	71.92%	49.61	0.7115
>M01>M11>M11	454629	63.91%	56.40	0.1491	354004	71.90%	49.63	0.7088
>M11>M01>M01	460251	63.46%	56.51	0.1453	360771	71.36%	49.62	0.7100
>M11>M01>M11	460160	63.47%	56.48	0.1461	360630	71.37%	49.60	0.7125
>M11>M11>M01	465334	63.06%	56.70	0.1391	367552	70.82%	49.88	0.6689
>M11>M11>M11	465419	63.05%	56.71	0.1388	367620	70.82%	49.88	0.6681

Tabelle 2.1: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes 'boots.jpg' mit den Schwellwerten 1 und 2 aus verschiedenen Filterpfaden

Filterpfad	Threshold 3				Threshold 5			
	NZE	SSS	PSNR	MSE	NZE	SSS	PSNR	MSE
>M01>M01>M01	286349	77.27%	45.63	1.7796	208347	83.46%	41.14	4.9997
>M01>M01>M11	286328	77.27%	45.62	1.7845	208397	83.46%	41.15	4.9884
>M01>M11>M01	288652	77.09%	45.69	1.7536	210755	83.27%	41.17	4.9630
>M01>M11>M11	288985	77.06%	45.72	1.7412	210985	83.25%	41.19	4.9456
>M11>M01>M01	296972	76.43%	45.72	1.7403	219069	82.61%	41.22	4.9137
>M11>M01>M11	296776	76.44%	45.73	1.7364	218888	82.62%	41.22	4.9147
>M11>M11>M01	304672	75.81%	45.94	1.6551	227273	81.96%	41.31	4.8038
>M11>M11>M11	304812	75.80%	45.95	1.6519	227411	81.95%	41.34	4.7800

Tabelle 2.2: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes 'boots.jpg' mit den Schwellwerten 3 und 5 aus verschiedenen Filterpfaden

Filterpfad	Threshold 10				Threshold 20			
	NZE	SSS	PSNR	MSE	NZE	SSS	PSNR	MSE
>M01>M01>M01	125014	90.08%	35.88	16.7874	76527	93.93%	31.68	44.1792
>M01>M01>M11	125085	90.07%	35.90	16.7208	76538	93.92%	31.71	43.8882
>M01>M11>M01	127123	89.91%	35.75	17.2905	78148	93.80%	31.39	47.2152
>M01>M11>M11	127413	89.89%	35.80	17.0872	78153	93.80%	31.47	46.3269
>M11>M01>M01	133937	89.37%	35.88	16.7838	81261	93.55%	31.65	44.4592
>M11>M01>M11	133890	89.37%	35.89	16.7616	81188	93.56%	31.64	44.6218
>M11>M11>M01	140694	88.83%	35.72	17.4173	84789	93.27%	31.11	50.3496
>M11>M11>M11	140703	88.83%	35.72	17.4249	84771	93.27%	31.14	50.0682

Tabelle 2.3: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes 'boots.jpg' mit den Schwellwerten 10 und 20 aus verschiedenen Filterpfaden

Filterpfad	Threshold 30				Threshold 50			
	NZE	SSS	PSNR	MSE	NZE	SSS	PSNR	MSE
>M01>M01>M01	61699	95.10%	29.73	69.1776	51537	95.91%	27.76	108.9028
>M01>M01>M11	61789	95.09%	29.78	68.4130	51562	95.91%	27.84	107.0013
>M01>M11>M01	62805	95.01%	29.29	76.6038	52006	95.87%	27.03	128.7776
>M01>M11>M11	62691	95.02%	29.37	75.1723	51961	95.88%	27.19	124.1484
>M11>M01>M01	64455	94.88%	29.74	69.0288	52552	95.83%	27.74	109.4109
>M11>M01>M11	64392	94.89%	29.73	69.1921	52493	95.83%	27.75	109.1614
>M11>M11>M01	66391	94.73%	28.94	82.9337	53152	95.78%	26.70	139.1466
>M11>M11>M11	66405	94.73%	29.00	81.8802	53130	95.78%	26.75	137.3394

Tabelle 2.4: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes 'boots.jpg' mit den Schwellwerten 30 und 50 aus verschiedenen Filterpfaden

Abbildung 2.30: Thresholdrekonstruktion von `'boots.jpg'` mit Schwellwert 1

Abbildung 2.31: Fehlerbild der Rekonstruktion von `'boots.jpg'` mit Threshold 1

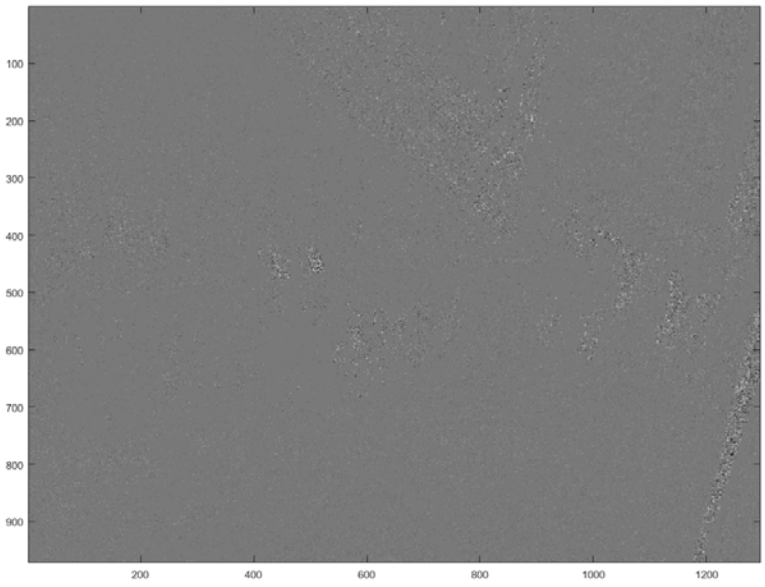

Abbildung 2.32: Quotientenbild der Rekonstruktion von `'boots.jpg'` mit Threshold 1

Abbildung 2.33: Thresholdrekonstruktion von `'boots.jpg'` mit Schwellwert 5

Abbildung 2.34: Fehlerbild der Rekonstruktion von `'boots.jpg'` mit Threshold 5

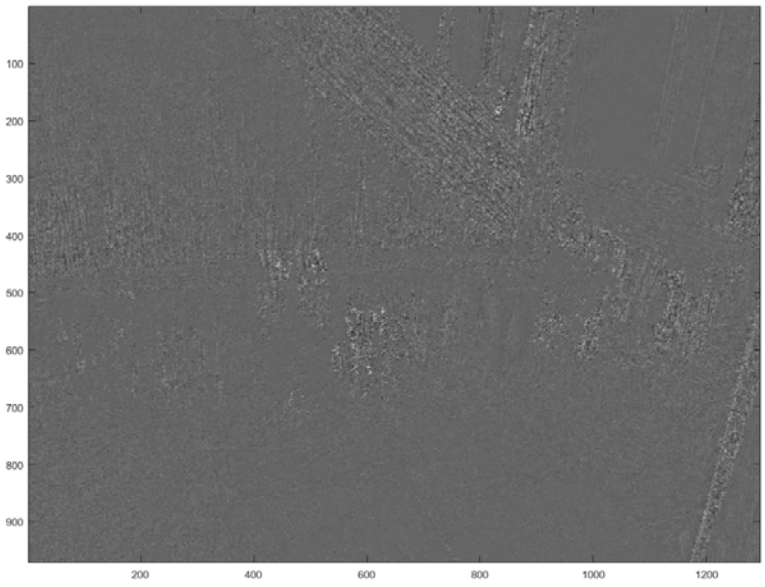

Abbildung 2.35: Quotientenbild der Rekonstruktion von `'boots.jpg'` mit Threshold 5

Abbildung 2.36: Thresholdrekonstruktion von `'boots.jpg'` mit Schwellwert 10

Abbildung 2.37: Fehlerbild der Rekonstruktion von `'boots.jpg'` mit Threshold 10

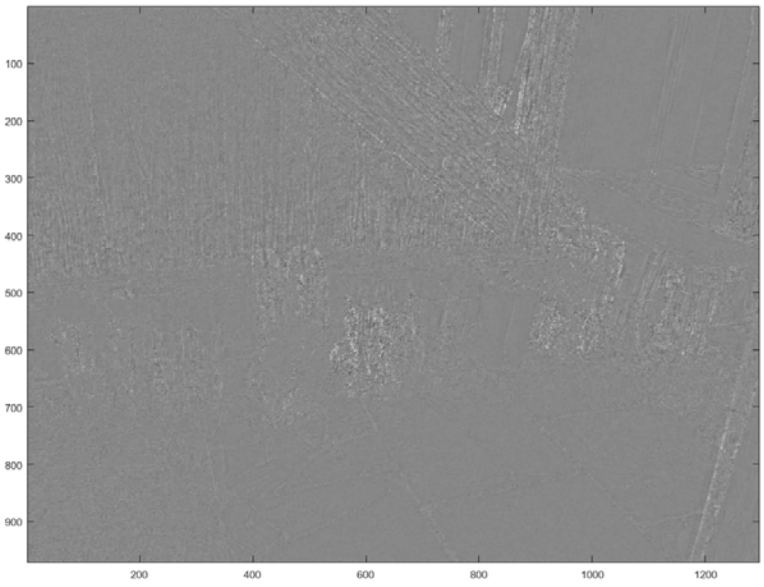

Abbildung 2.38: Quotientenbild der Rekonstruktion von `'boots.jpg'` mit Threshold 10

Abbildung 2.39: Thresholdrekonstruktion von `'boots.jpg'` mit Schwellwert 20

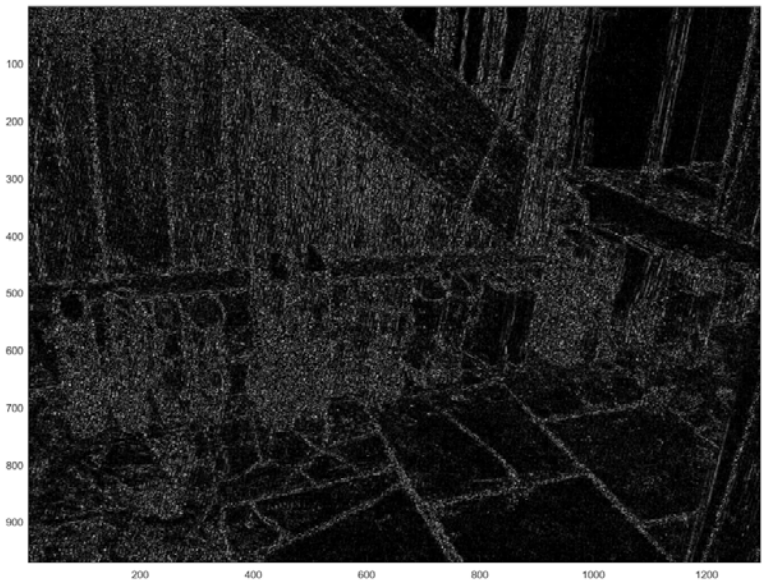

Abbildung 2.40: Fehlerbild der Rekonstruktion von `'boots.jpg'` mit Threshold 20

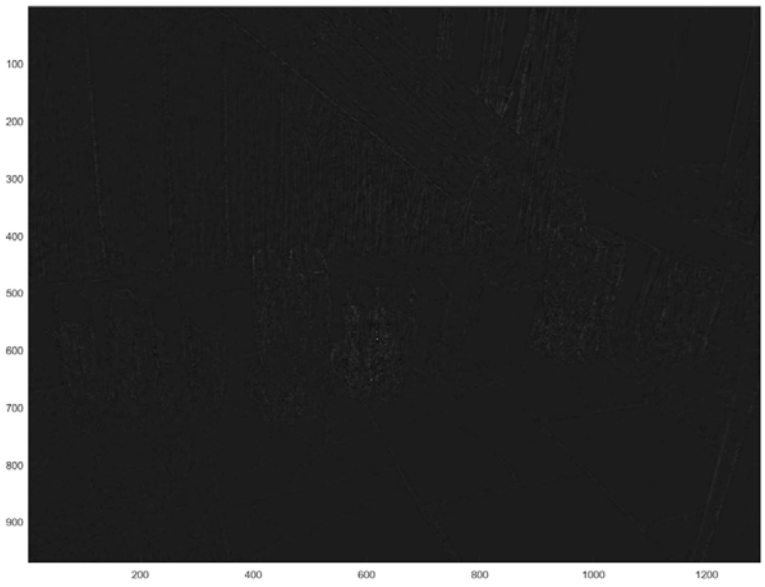

Abbildung 2.41: Quotientenbild der Rekonstruktion von `'boots.jpg'` mit Threshold 20

Rekonstruktionstiefe

Das Bild `'boots.jpg'` wird unter Verwendung der Filterbank `'M01'` bis zu einer maximalen Tiefe von 5 zerlegt. Anschließend wird aus jedem Tiefenlevel eine Rekonstruktion mit verschiedenen Thresholdeinstellungen gestartet. In den nachfolgenden Tabellen 2.5 bis 2.8 werden gemäß obigem Schema statistisch relevante Werte der Rekonstruktionen dargestellt. Die Abbildungen 2.42 bis 2.47 visualisieren die Thesholdrekonstruktion mit Schwellwert 20 aus verschiedenen Tiefenleveln sowie deren Quotienten- und Fehlerbild.

Filterpfad	Threshold 1				Threshold 2			
	NZE	SSS	PSNR	MSE	NZE	SSS	PSNR	MSE
>M01	683783	45.72%	58.45	0.0929	611368	51.47%	52.52	0.3638
>M01>M01	508635	59.62%	56.97	0.1305	413673	67.16%	50.45	0.5868
>M01>M01>M01	453280	64.02%	56.36	0.1503	351728	72.08%	49.54	0.7230
>M01>M01>M01>M01	435603	65.42%	55.95	0.1654	332182	73.63%	48.98	0.8229
>M01>M01>M01>M01>M01	430063	65.86%	55.71	0.1745	326238	74.10%	48.67	0.8825

Tabelle 2.5: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes `'boots.jpg'` mit dem Schwellwerten 1 und 2 aus verschiedenen Tiefenleveln

Filterpfad	Threshold 3				Threshold 5			
	NZE	SSS	PSNR	MSE	NZE	SSS	PSNR	MSE
>M01	566812	55.00%	49.02	0.8150	515918	59.04%	44.92	2.0957
>M01>M01	353307	71.95%	46.68	1.3975	281924	77.62%	42.36	3.7745
>M01>M01>M01	286349	77.27%	45.63	1.7796	208347	83.46%	41.14	4.9997
>M01>M01>M01>M01	265317	78.94%	44.98	2.0643	185222	85.30%	40.37	5.9777
>M01>M01>M01>M01>M01	259030	79.44%	44.63	2.2380	178360	85.84%	39.91	6.6457

Tabelle 2.6: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes `'boots.jpg'` mit dem Schwellwerten 3 und 5 aus verschiedenen Tiefenleveln

Filterpfad	Threshold 10				Threshold 20			
	NZE	SSS	PSNR	MSE	NZE	SSS	PSNR	MSE
>M01	464730	63.11%	40.11	6.3411	436222	65.37%	36.25	15.4120
>M01>M01	207210	83.55%	37.36	11.9287	164873	86.91%	33.53	28.8567
>M01>M01>M01	125014	90.08%	35.88	16.7874	76527	93.93%	31.68	44.1792
>M01>M01>M01>M01	98808	92.16%	34.85	21.2623	47752	96.21%	30.37	59.6615
>M01>M01>M01>M01>M01	90991	92.78%	34.15	25.0127	38883	96.91%	29.27	76.9785

Tabelle 2.7: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes `'boots.jpg'` mit dem Schwellwerten 10 und 20 aus verschiedenen Tiefenleveln

Filterpfad	Threshold 30				Threshold 50			
	NZE	SSS	PSNR	MSE	NZE	SSS	PSNR	MSE
>M01	427703	66.05%	34.43	23.4198	422062	66.50%	32.65	35.2909
>M01>M01	152301	87.91%	31.77	43.2746	143772	88.59%	30.07	63.9975
>M01>M01>M01	61699	95.10%	29.73	69.1776	51537	95.91%	27.76	108.9028
>M01>M01>M01>M01	31914	97.47%	28.28	96.7327	20979	98.33%	26.15	157.7793
>M01>M01>M01>M01>M01	22516	98.21%	26.89	133.1909	11067	99.12%	24.39	236.5221

Tabelle 2.8: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes `'boots.jpg'` mit dem Schwellwerten 30 und 50 aus verschiedenen Tiefenleveln

Abbildung 2.42: Thresholdrekonstruktion von `'boots.jpg'` mit Schwellwert 20 aus Tiefenlevel 1

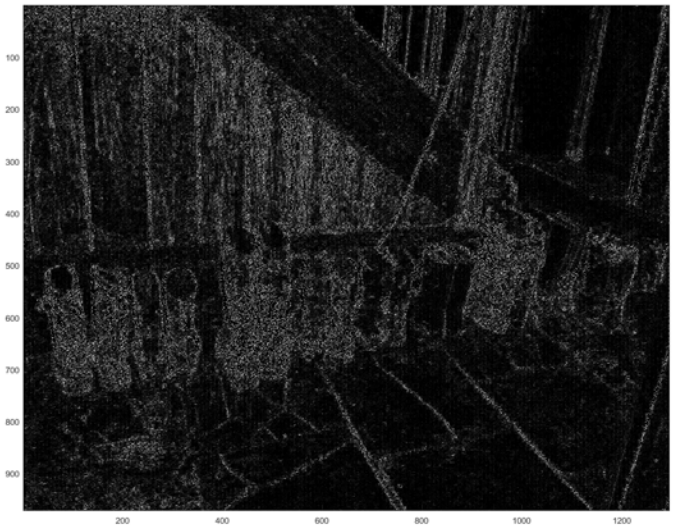

Abbildung 2.43: Fehlerbild der Rekonstruktion von `'boots.jpg'` mit Threshold 20 aus Tiefenlevel 1

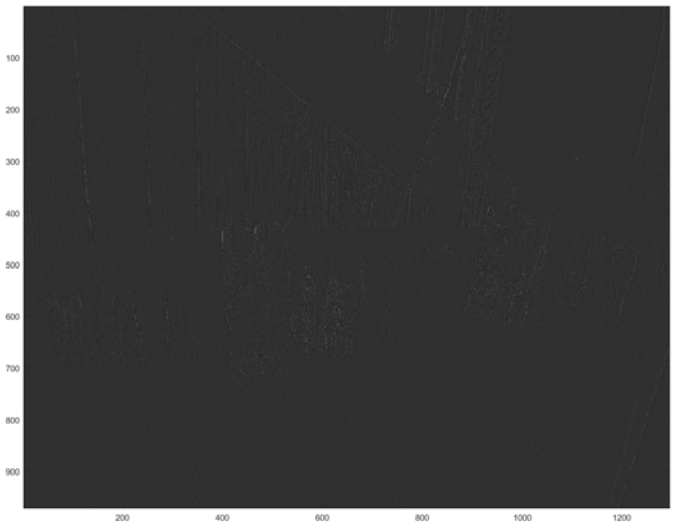

Abbildung 2.44: Quotientenbild der Rekonstruktion von `'boots.jpg'` mit Threshold 20 aus Tiefenlevel 1

Abbildung 2.45: Thresholdrekonstruktion von `'boots.jpg'` mit Schwellwert 20 aus Tiefenlevel 3

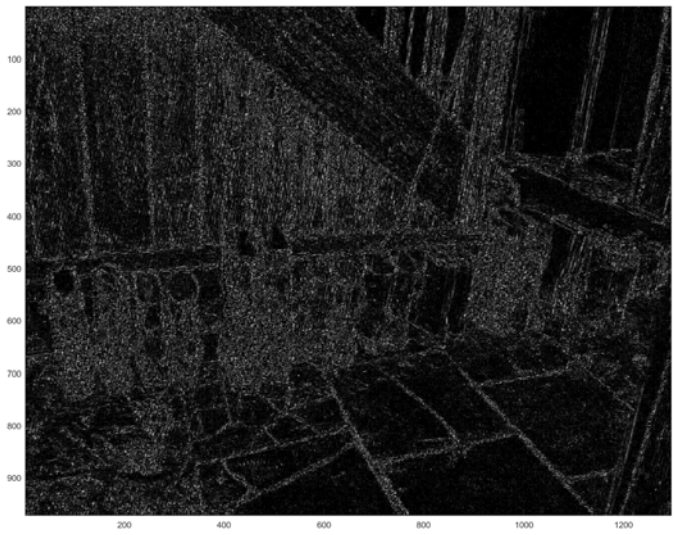

Abbildung 2.46: Fehlerbild der Rekonstruktion von `'boots.jpg'` mit Threshold 20 aus Tiefenlevel 3

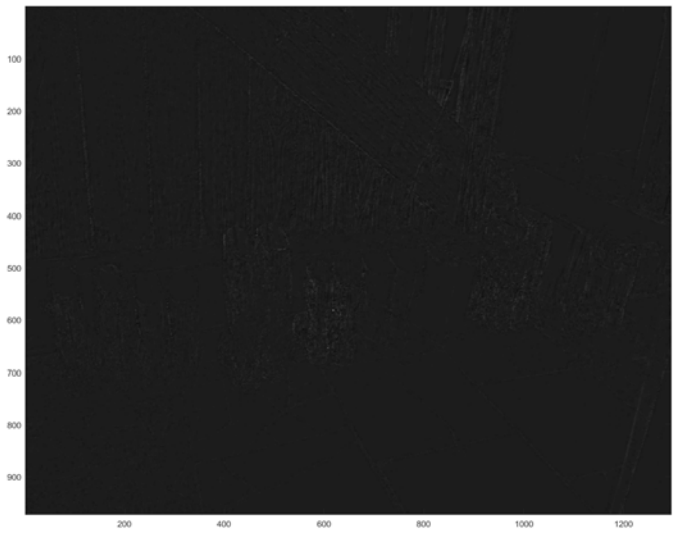

Abbildung 2.47: Quotientenbild der Rekonstruktion von `'boots.jpg'` mit Threshold 20 aus Tiefenlevel 3

2.4.3 Thresholding (SSIM-Quality-Check)

Um die Qualität eines rekonstruierten Bildes I_{new} im Hinblick auf das Originalbild I_{orig} beurteilen zu können, kann ebenso der Index struktureller Ähnlichkeit $SSIM$ (engl. structural similarity) verwendet [6] werden. Dabei wird der SSIM-Index über verschiedene Bildteile berechnet. Üblicherweise verwendet man kleine quadratische Fenster, welche Punkt für Punkt über das Bild verschoben werden. Dadurch erhält man für den Vergleich von je zwei korrespondierenden Bildpixeln einen dezimalen Wert zwischen -1 und 1. Aus der Mittelung aller Einzelwerte über die Gesamtzahl der Bildpixel resultiert der SSIM-Index. Je näher der dezimale Index am Wert 1 liegt, desto ähnlicher sind sich die beiden Bilder.
Das Verfahren ist eine Schätzung der tatsächlich wahrgenommenen Ähnlichkeit zwischen zwei Bildern, welches im Gegensatz zum obig angewandten PSNR-Verfahren eine größere Übereinstimmung mit menschlicher visueller Wahrnehmung erzielt.
Der SSIM-Index zweier korrespondierender Fenster $x \subset I_{orig}$ und $y \subset I_{new}$ gleicher Größe wird wie folgt berechnet:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{x,y} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.117}$$

Dabei bezeichnet μ_x den Mittelwert des Fensters x, μ_y den Mittelwert des Fensters y, σ_x^2 die Varianz von x, σ_y^2 die Varianz von y und $\sigma_{x,y}$ die Kovarianz von x und y. Die Konstanten c_1 und c_2 dienen zur Stabilisierung der Division bei kleinen Nennern und haben bei den von uns betrachteten Bildern die Werte $c_1 = 2.55^2$ und $c_2 = 7.65^2$.
Im Folgenden wird das Bild `'boots.jpg'` bis zu einer Tiefe von 3 zerlegt und anschließend mit einem Threshold 30 rekonstruiert.

Filterpfad	NZE	SSS	PSNR	SSIM
>M01>M01>M01	61699	95.10%	29.73	**0.805703**
>M01>M01>M11	61789	95.09%	29.78	**0.802524**
>M01>M11>M01	62805	95.01%	29.29	**0.794434**
>M01>M11>M11	62691	95.02%	29.37	**0.782467**
>M11>M01>M01	64455	94.88%	29.74	**0.801865**
>M11>M01>M11	64392	94.89%	29.73	**0.801172**
>M11>M11>M01	66391	94.73%	28.94	**0.765794**
>M11>M11>M11	66405	94.73%	29.00	**0.767938**
Alle Pfade	510627	59.46%	31.10	**0.831141**

Tabelle 2.9: Statistisch relevante Werte zur Thresholdrekonstruktion des Bildes `'boots.jpg'` mit dem Schwellwert 30 aus verschiedenen Pfaden unter Verwendung des SSIM-Maßes

2.4.4 Thresholding (Berücksichtigung der Abklingrate)

Wird die Abklingrate der Hochpasskoeffizienten berücksichtigt, so werden die Hochpassmatrizen im Tiefenlevel i für die Koeffizientenauswahl mit dem Faktor $|det(M)|^i$ multipliziert, wobei M die zum verwendeten Filter gehörige Dilatationsmatrix ist.

Beispiel 2.4.29: Koeffizientenauswahl unter Berücksichtigung der Abklingrate
Folgende Matlab-Routine zeigt detailliert das Vorgehen bei der Koeffizientenauswahl. Wir befinden uns hier im Tiefenlevel 2, der zweiten Zerlegungsstufe also. Verwendet wurde die Filterbank `'MO1'` mit der Dilatationsmatrix $M = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix}$. Der Abklingfaktor berechnet sich demnach durch $|det(M)|^2 = 9$. Als Threshold wird der Wert 200 verwendet.

```
highpass =

      0        0        0       0        0       0 28.3331 -19.6665 22.3331
      0        0        0 55.3334 -50.6665 34.3332 24.6667  23.0001 16.6665
      0 101.6666 -39.6671  9.6667  -3.9999 57.6666  7.6665        0       0
-9.3334  72.6666 -42.0003 -8.6667        0       0       0        0       0

» factor = abs(det(dilationMatrix))^depth

factor =

9

» tmpHighpass = highpass;
» tmpHighpass = abs(tmpHighpass) * factor

tmpHighpass =

     0       0       0       0       0       0 254.995 176.992 200.990
     0       0       0 498.009 455.991 308.994 222.006 207.001 149.995
     0 914.991 357.004  87.004  35.997 518.993  68.995       0       0
84.004 653.994 378.002  78.004       0       0       0       0       0

» highpass(tmpHighpass < threshold) = 0

highpass =

0        0        0        0        0        0 28.3331        0 22.3331
0        0        0  55.3334 -50.6665  34.3332 24.6667  23.0001       0
```

```
0 101.6666 -39.6671         0         0  57.6666         0         0         0
0  72.6666 -42.0003         0         0         0         0         0         0
```

Betrachtet man nun die rekonstruierten Bilder mit und ohne Berücksichtigung der Abklingrate, so stellt man fest, dass bei etwa gleicher Speicherersparnis diejenigen unter Berücksichtigung der Abklingrate deutlich bessere Ergebnisse liefern. Wird PSNR oder SSIM betrachtet, so unterscheiden sich die Maßwerte der Bilder unwesentlich, mit dem bloßen Auge sind jedoch starke Qualitätsunterschiede zu erkennen (siehe dazu Abbildungen 2.48 und 2.49).

Abbildung 2.48: Das Bild `boots.jpg` wird mit der Filterbank `'M01'` bis zu einer Tiefe von 4 zerlegt. Anschließend wird eine Rekonstruktion unter Verwendung des Schwellwertes 25 durchgeführt. Für die gesamte Rekonstruktion sind 67291 von Null verschiedene Werte nötig. Das Originalbild besitzt insgesamt 1259712 von Null verschiedene Werte.

Abbildung 2.49: Das Bild `boots.jpg` wird mit der Filterbank `'MO1'` bis zu einer Tiefe von 4 zerlegt. Anschließend wird eine Rekonstruktion unter Verwendung des Schwellwertes 220 unter Berücksichtigung der Abklingrate der Koeffizienten in den Hochpassbildanteilten durchgeführt. Für die gesamte Rekonstruktion sind 67027 von Null verschiedene Werte nötig. Das Originalbild besitzt insgesamt 1259712 von Null verschiedene Werte.

2.5 Bedienung des Programms

Für die Bedienung des Programms sind vorwiegend die Anzeigefunktionen (siehe Abschnitt 2.1.1) sowie die Hauptfunktionen (siehe Abschnitt 2.1.2) von Bedeutung. Alle weiteren Funktionen werden nur intern für Zerlegung oder Rekonstruktion verwendet.

2.5.1 Bildzerlegung

Die Zerlegung eines Bildes wird mit der Funktion `WaveletDecomposition` (siehe Abschnitt 2.1.2) realisiert. Diese erhält als Parameter das zu zerlegende Bild, die zu verwendenden Filterbänke sowie die maximale Zerlegungstiefe.
Das Bild wird üblicherweise mit der Matlab-Funktion `imread` geladen. Es kann theoretisch beliebig viele Layer (z.B. RGB) besitzen. Bei der normalen Rekonstruktion werden alle Layer separat behandelt, bei der Kantenerkennung werden die Hochpassrekonstruktionsergebnisse der einzelnen Layer am Ende addiert und durch die Anzahl der Layer geteilt.

Wird nur eine Filterbank verwendet, so wird ganz einfach deren Name angegeben (z.B. `'E01'`). Sollen zwei oder mehrere Filterbänke parallel zur Dekomposition verwendet werden, so übergibt man deren Namen zeilenweise in einem Array (z.B. `['M01';'M11']`). Soll im Anschluss eine Hochpassrekonstruktion gestartet werden und will man unschöne Artefakte an den Bildrändern vermeiden, so kann man den optionalen Parameter `'expandEdges'` verwenden.

Beispiel 2.5.30: Zerlegung eines Bildes
Im Folgenden sind mögliche Zerlegungsbefehle aufgelistet.

```
» image = imread('Testbilder/cameraman.tif');
» decArray = WaveletDecomposition(image, 'M01', 1)

decArray =

2×1 cell array

{1×6 cell}
{1×5 cell}

» decArray = WaveletDecomposition(image, ['M01'; 'M11'], 2)

decArray =

3×4 cell array

{1×6 cell}            []            []            []
{1×5 cell}    {1×5 cell}            []            []
{1×5 cell}    {1×5 cell}    {1×5 cell}    {1×5 cell}

decArray = WaveletDecomposition(image, ['E00'], 3)

decArray =

4×1 cell array

{1×6 cell}
{1×5 cell}
{1×5 cell}
{1×5 cell}
```

2.5.2 Bildrekonstruktion

Die Rekonstruktion eines Bildes aus einem Dekompositionspfad wird mit der Funktion `WaveletReconstruction` (siehe Abschnitt 2.1.2) durchgeführt. Die Funktion erhält als Parameter ein Cell-Array `decArray` mit Zerlegungsinformationen. Zudem wird der Index der Zelle angegeben, aus welcher die Rekonstruktion gestartet werden soll. Die Funktion berechnet anschließend automatisch den korrekten Rekonstruktionspfad sowie die zu verwendenden Rekonstruktionsfilter.
Als optionaler Parameter kann das Schlüsselwort `'threshold'` gefolgt von einem Schwellwert angefügt werden. Der Threshold wird dann in jedem Rekonstruktionsschritt auf die Hochpassinformationen angewandt.
Auch kann der optionale Parameter `'highpassOnly'` dem Funktionsaufruf hinzugefügt werden. Damit wird eine Rekonstruktion rein aus den Hochpassinformationen gestartet. Die optionalen Parameter können auch gleichzeitig verwendet werden, allerdings ist die Sinnhaftigkeit der Anwendung eines Thresholds bei der Kantenerkennung fragwürdig.

Beispiel 2.5.31: Rekonstruktion eines Bildes
Im Folgenden sind mögliche Rekonstruktionsbefehle aufgelistet.

```
» image = imread('Testbilder/cameraman.tif');
» decArray = WaveletDecomposition(image, ['M01'; 'M11'], 2);
» newImage = WaveletReconstruction(decArray, [3, 2]);
» highpassInfo = WaveletReconstruction(decArray, [3, 3], ...
            'highpassOnly');
» threshold20 = WaveletReconstruction(decArray, [3, 4], ...
            'threshold', 20);
```

2.5.3 Vergleich von Ergebnissen

Wenn mehrere Filterpfade oder Tiefenlevel im Hinblick auf Thresholdeffizienz oder Kantenerkennung miteinander verglichen werden sollen, werden die Anzeigefunktionen (siehe Abschnitt 2.1.1) verwendet. Sämtliche Plots werden in einem Cell-Array zurückgegeben und können bei Bedarf weiter analysiert werden.

3 Schluss

Die hier angeführten Codebeispiele sollen bewusst kein Patentrezept für Kantenerkennung und Bildkompression darstellen. Sie geben vielmehr Einblick in die Benutzung des Programms und ermöglichen es dem Nutzer die Funktionsparameter an seine individuellen Anforderungen innerhalb kürzester Zeit anzupassen. Der Vergleich der unterschiedlichen Tiefenlevel sowie Rekonstruktionspfade lässt erkennen, dass sich für verschiedene Anwendungen die implementierten Filterbänke unterschiedlich gut eignen. Existiert für einen Spezialfall tatsächlich keine passende Filterbank, so kann ohne großen Aufwand ein selbst erstellter Filter hinzugefügt werden. Das somit sehr modulare und evolutionsfähige Programm eignet sich für einen Großteil der im Internet-of-Things anfallenden Bildanalyseaufgaben.

4 Anhang

Auf der beigelegten CD-ROM befinden sich sämtliche MatLab-Funktionen sowie eine Auswahl an interessanten Testbildern. Alle im Dokument abgebildeten, analysierten Bilder sind in Originalgröße sowie als MatLab-Figure zur genaueren Betrachtung beigelegt.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Mariantonia Cotronei, Daniele Ghisi, Milvia Rossini, Tomas Sauer; 'An anisotropic directional subdivision and multiresolution', *Advances in Computational Mathematics*, 2014.

[2] Werner Bäni; 'Wavelets - Eine Einführung für Ingenieure', Oldenburg Wissenschaftsverlag, 1. Auflage, 2002.

[3] Tomas Sauer; 'Shearlet multiresolution and multiple refinement', In: Gitta Kutyniok 'Shearlets', Springer, 2011.

[4] Milvia Rossini, Elena Volontè; 'On directional scaling matrices in dimension d=2', 2017.

[5] University of Wisconsin-Madison, Computer-Aided Engineering: Public domain test images"; `http://homepages.cae.wisc.edu/~ece533/images/index.html`

[6] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli; 'Image quality assessment: From error visibility to structural similarity,' IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.

[7] Ewald Wasmeier; 'Filterbänke für Bildzerlegung', Bachelorarbeit, 2015.

Erklärung zur selbstständigen Anfertigung

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Passau, den 11.12.2017