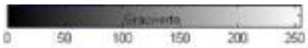
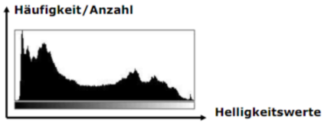
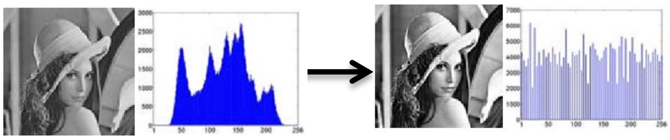
Histogramm & Histogrammäqualisierung

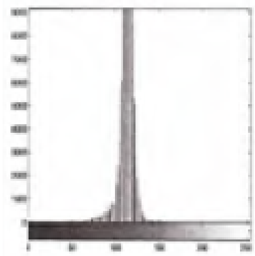
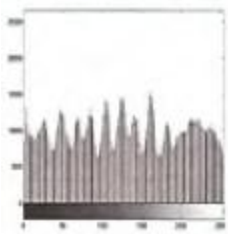
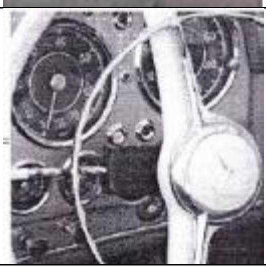
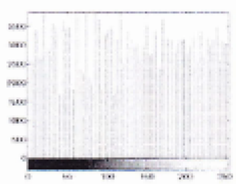
Histogramm: Häufigkeiten der Grauwerte eines Bildes geordnet nach Helligkeit: oft relative Häufigkeit





Histogrammäqualisierung: Ziel möglichst gleichmäßige Verteilung der Grauwerte über die gesamte Grauwertskala → hoher Bildkontrast

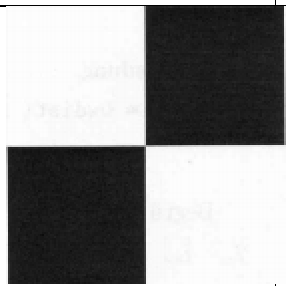
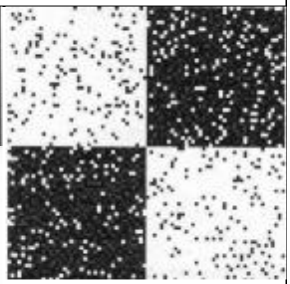
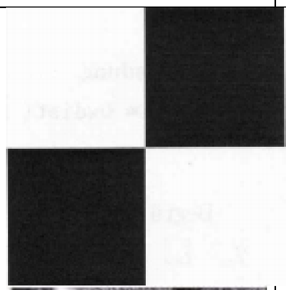
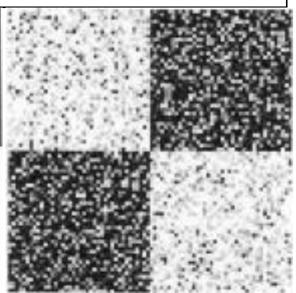
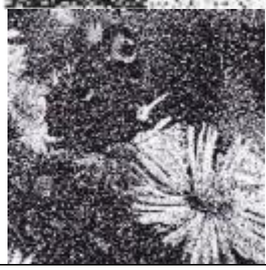


Matlab & Begründung	Input-Image (`I`)	Output-Image (`E`)
`E = imhist(I)` Histogramm: Das Eingangsbild `I` hat nur wenig verschiedene eher mittelgraue Grauwerte.		
`E = imhist(I)` Histogramm: Im Eingangsbild `I` kommen alle Grauwerte gleichverteilt vor.		
`E = histeq(I)` Histogrammäqualisierung → vom Eingangsbild werden die Grauwerte gleichverteilt; Kontrast wird erhöht		
`E = imhist(histeq(I))` Histogramm: `histeq(I)` → Histogrammäqualisierung: alle Grauwerte im Eingangsbild `I` werden gleichverteilt; `E` liefert dann das Histrogramm vom Bild mit der Histogrammäqualisierung → Grauwerte sind im Histogramm gleichverteilt		

Rauschen

<u>Salz & Pfeffer Rauschen</u>:
nur wenige Pixel sind betroffen und nehmen entweder den hellsten Grauwert 255 (weiß) oder den dunkelsten Grauwert 0 (schwarz) an. Der Grauwert 255 repräsentiert das „Salz", der Grauwert 0 den „Pfeffer". → zufälliges Auftreten von schwarzen und weißen Pixeln

<u>Gaußrauschen</u>:
"Rauschen, bei dem die Amplituden der einzelnen Frequenzen gaußverteilt sind.

Matlab & Begründung	Input-Image (`I`)	Output-Image (`E`)
`E = imnoise(I,'salt&pepper',0.1)` Salz & Pfeffer Rauschen: Auf dem Eingangsbild wird Salz- und Pfeffer- Rauschen hinzugefügt, wobei der Wert 0.1 die Rauschdichte angibt. Nur wenige Pixel sind betroffen und nehmen entweder den hellsten Grauwert 255 (weiß) oder den dunkelsten Grauwert 0 (schwarz) an. Der Grauwert 255 repräsentiert das „Salz", der Grauwert 0 den „Pfeffer". → zufälliges Auftreten von schwarzen und weißen Pixeln		
`E = imnoise(I,'salt&pepper',0.2)` Salz & Pfeffer Rauschen: Auf dem Eingangsbild wird Salz- und Pfeffer- Rauschen hinzugefügt, wobei der Wert 0.2 die Rauschdichte angibt. Nur wenige Pixel sind betroffen und nehmen entweder den hellsten Grauwert 255 (weiß) oder den dunkelsten Grauwert 0 (schwarz) an. Der Grauwert 255 repräsentiert das „Salz", der Grauwert 0 den „Pfeffer". → zufälliges Auftreten von schwarzen und weißen Pixeln		
`E = imnoise(I,'gaussian',0,0.2)` Gaußrauschen: Auf dem Eingangsbild wird Gaußrauschen hinzugefügt, wobei 0 den Mittelwert (mittlere Helligkeit) und 0.2 die Varianz (mittlerer Kontrast) angibt. Die betroffenen Pixel nehmen alle Grauwerte an.	 	 

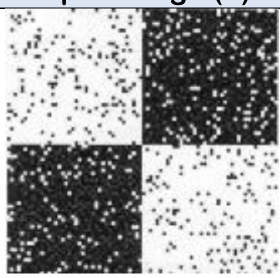
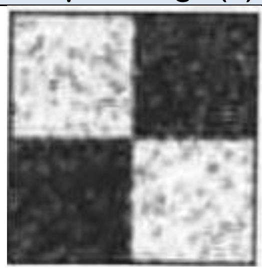
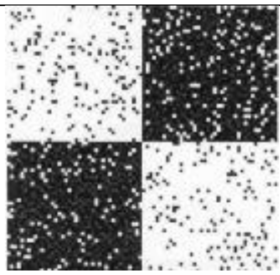
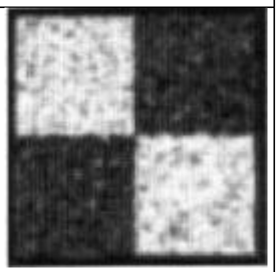
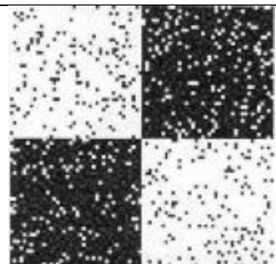
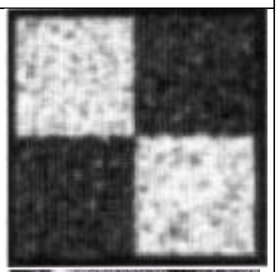
Rauschunterdrückung

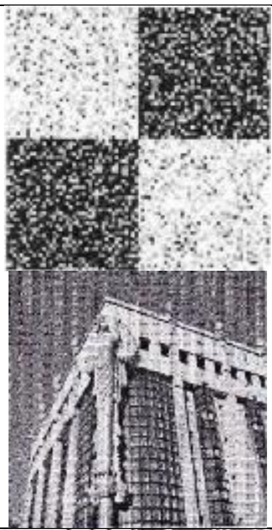
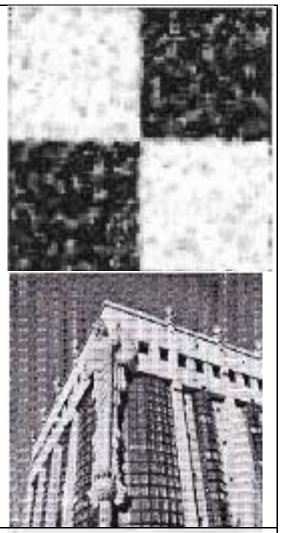
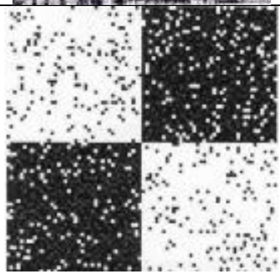
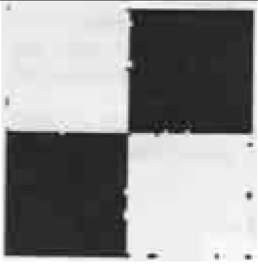
Mittelwertfilter: (Tiefpassfilter, Glättungsfilter, Rauschunterdrückungsfilter, lienaler Filter)
Der Mittelwertfilter berechnet den arithmetischen Mittelwert der Grauwerte der Pixel in der Nachbarschaft unter der Filtermaske.
Merkmale: glättet Rauschen, verwischt Kanten, Kontrastverlust von schmalen Objekten

Gaußfilter: (Tiefpassfilter, Glättungsfilter, Rauschunterdrückungsfilter, lienaler Filter)
Der Gaußfilter ist im Prinzip nur eine Erweiterung des gewichteten Mittelwertfilters. Das Zentrum wird analog zur Gaußglocken-Funktion stärker gewichtet als der Rand und der Einfluss entfernter Pixel wird reduziert.
Merkmale: hat eine gleichmäßige Wirkung auf hochfrequente Bildanteile wie Kanten und isolierte Störungen: selektiver Weichzeichner, erhält starke Kontraste, originale Bildinformation wird im gefilterten Bild stärker betont, Kanten werden nicht so stark abgeflacht, wie beim Mittelwertfilter

Medianfilter: (Tiefpassfilter, Glättungsfilter, Rauschunterdrückungsfilter, nicht lienaler Filter, Rangordnungsfilter)
Beim Medianfilter werden die Grauwerte der Pixel in einer definierten Umgebung eines Pixels aufgesammelt und der Größe nach sortiert. Der mittlere Wert (Median) der sortierten Liste wird zurückgegeben und der Wert des zentralen Pixels wird durch ihn ersetzt. Dabei werden Ausreißerpixel, auch "Salt and Pepper" genannt, durch Pixel aus der Umgebung ersetzt.
Merkmale: Glättung des Bildes bei Erhalt der Kantensteilheit → die Kanten werden nicht verschmiert - das Bild verliert kaum an Schärfe, es entstehen keine neuen Grauwerte, kleine sporadische Bildstörungen werden beseitigt; die Ecken werden abgerundet, und Punkten und Linien mit Dicke von 1 Pixel verschwinden

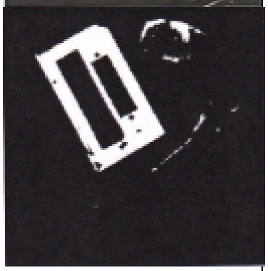
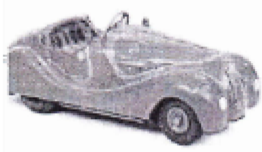
Faltung: Ein Pixel des Ausgabebildes wird aus mehreren Pixel des Eingabebildes berechnet

Matlab & Begründung	Input-Image (`I`)	Output-Image (`E`)
`E = conv2(I,[1/9 1/9 1/9; 1/9 1/9 1/9; 1/9 1/9 1/9])` Mittelwertfilter (Tiefpassfilter, Glättungsfilter, Rauschunterdrückungsfilter, lienaler Filter): Eingangsbild wir mit einem 3x3-Filterkern geglättet → Im Ergebnisbild wird das „Salz & Pfeffer“ - Rauschen mit der Glättung breiter, es verliert an Kontrast aber verschwindet nicht; das Bild wird unscharf; Kantenübergänge werden verwischt/geglättet; Der schwarze Rand entsteht durch die Behandlung von Randpixel.		
`E = conv2(I,fspecial('gaussian', [7 7], 1))` Gaußfilter (Tiefpassfilter, Glättungsfilter, Rauschunterdrückungsfilter, lienaler Filter): Eingangsbild wir mit einem 7x7-Filterkern und einer Standartabweichung von 1 geglättet → Im Ergebnisbild wird das „Salz & Pfeffer“ - Rauschen mit der Glättung breiter, es verliert an Kontrast aber verschwindet nicht; das Bild wird unscharf; Kantenübergänge werden verwischt/geglättet, jedoch nicht so stark wie beim Mittelwertfilter; Der schwarze Rand entsteht durch die Behandlung von Randpixel.		
`E = conv2(I,fspecial('gaussian', [5 5], 5))` Gaußfilter (Tiefpassfilter, Glättungsfilter, Rauschunterdrückungsfilter, lienaler Filter): Eingangsbild wir mit einem 5x5-Filterkern und einer Standartabweichung von 5 geglättet → Im Ergebnisbild wird das „Salz & Pfeffer“ - Rauschen mit der Glättung breiter, es verliert an Kontrast aber verschwindet nicht; das Bild wird unscharf; Kantenübergänge werden verwischt/geglättet, jedoch nicht so stark wie beim Mittelwertfilter; Der schwarze Rand entsteht durch die Behandlung von Randpixel.	 	 

`E = medfilt2(I)` Medianfilter (Tiefpassfilter, Glättungsfilter, Rauschunterdrückungsfilter, nicht lienaler Filter, Rangordnungsfilter): Medianfilter mit einem 3x3 Filterkern → Beseitigt Störungen und Rauschen, indem diese der Umgebung angepasst werden: Es werden keine neuen Grauwerte erzeugt und die Glättung des Ausgangsbildes ist nicht zu groß → Bild verliert kaum an Schärfe. Kanten sind gut erkennbar. Befinden sich in einer gegebenen Umgebung mehr Störpixel als Bildpunkte, so werden diese nicht restlos eliminiert.		
`E = medfilt2(I,[5 5],5)` Medianfilter (Tiefpassfilter, Glättungsfilter, Rauschunterdrückungsfilter, nicht lienaler Filter, Rangordnungsfilter): Medianfilter mit einem 5x5 Filterkern → Beseitigt Störungen und Rauschen, indem diese der Umgebung angepasst werden: Es werden keine neuen Grauwerte erzeugt und die Glättung des Ausgangsbildes ist nicht zu groß → Bild verliert kaum an Schärfe. Kanten sind gut erkennbar. Befinden sich in einer gegebenen Umgebung mehr Störpixel als Bildpunkte, so werden diese nicht restlos eliminiert. Da jedoch der Filterkern groß genug ist, wird der „Salz & Pfeffer“ – Rauschen entfernt.		

Binärbilder

Bilder, die nur Binärwerte 0 (schwarz) und 1 (weiß) enthalten. Umwandlung von Grauwertbildern in Binärbilder: Grauwerte ≤ Schwellwert → schwarz; Grauwerte > Schwellwert → weiß

Matlab & Begründung	Input-Image (`I`)	Output-Image (`E`)
`E = im2bw(I,100/255)` Alle in `I` vorkommenden Grauwerte ≤ 100 → schwarz; restlichen Grauwerte 101-255 → weiß		
`E = im2bw(I,110/255)` Alle in `I` vorkommenden Grauwerte ≤ 110 → schwarz; restlichen Grauwerte 111-255 → weiß		
`E = im2bw(I,200/255)` Alle in `I` vorkommenden Grauwerte ≤ 200 → schwarz; restlichen Grauwerte 201-255 → weiß	 	 
`E = 1-im2bw(I,100/255)` `im2bw` ohne „1-“ macht folgendes: alle in `I` vorkommenden Grauwerte ≤ 100 → schwarz; restlichen Grauwerte 101-255 → weiß: `1-im2bw` invertiert das Binärbild → die im Original- Binärbild schwarzen Pixel werden weiß und die weißen Pixel werden schwarz	 	 

Morphologie

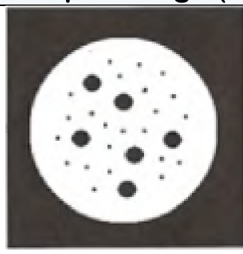
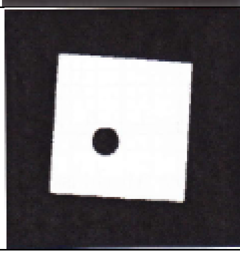
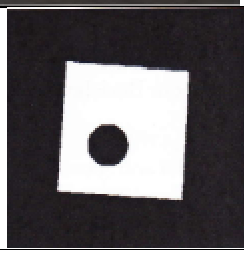
<u>Dilatation (Ausdehnung)</u> :
Das Strukturelement H, wird auf jeden Pixel im Originalbild gelegt. Bei der Dilatation wird es im Ergebnis $I \oplus H$ an jedem Punkt des Originalbildes repliziert (auf 1 gesetzt) → ändert ein Pixel von '0' (schwarz) zu '1'(weiß) wenn mindestens ein Nachbarpixel '1' (weiß) ist.
<u>Merkmale</u>: Löcher, dünne Risse werden kleiner oder verschwinden; dünne Linien und Objekte, die das Strukturelement nicht ganz aufnehmen können werden größer; nahe Objekte verschmelzen (wenn Riss länger als Strukturelement); dünn verbundene Objekte werden verbunden größer; alle anderen Objekte werden größer.

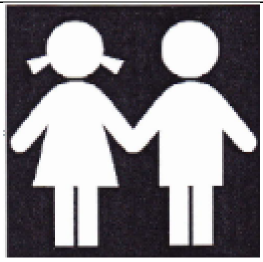
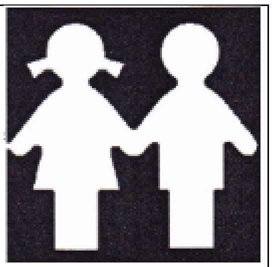
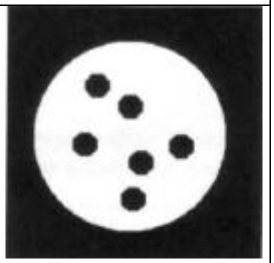
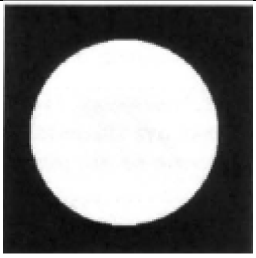
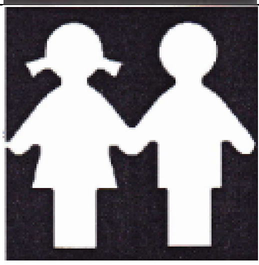
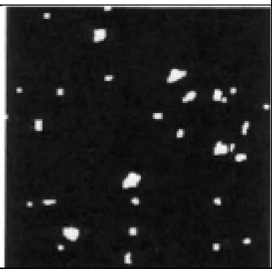
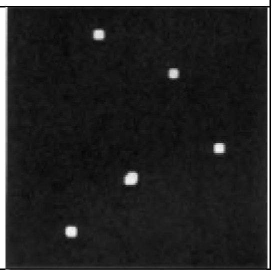
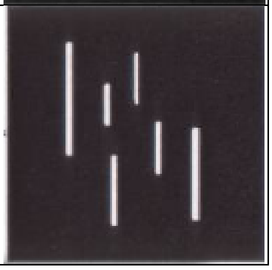
<u>Erosion (Schrumpfung)</u> :
Das Strukturelement H, wird auf jeden Pixel im Originalbild gelegt. Passt das Strukturelement H vollständig in die Region, so bleibt das Referenzpixel erhalten und die restlichen Pixel werden entfernt (auf 0 gesetzt); andernfalls werden alle Pixel entfernt. Das Ergebnis ist dann $I \ominus H$ → ändert ein Pixel von '1' (weiß) zu '0' (schwarz) wenn mindestens ein Nachbarpixel '0' (schwarz) ist
<u>Merkmale</u>: Löcher, dünne Risse werden größer; dünne Linien und Objekte, die das Strukturelement nicht ganz aufnehmen können verschwinden; nahe Objekte werden weiter entfernt; dünn verbundene Objekte werden getrennt; alle anderen Objekte werden kleiner.

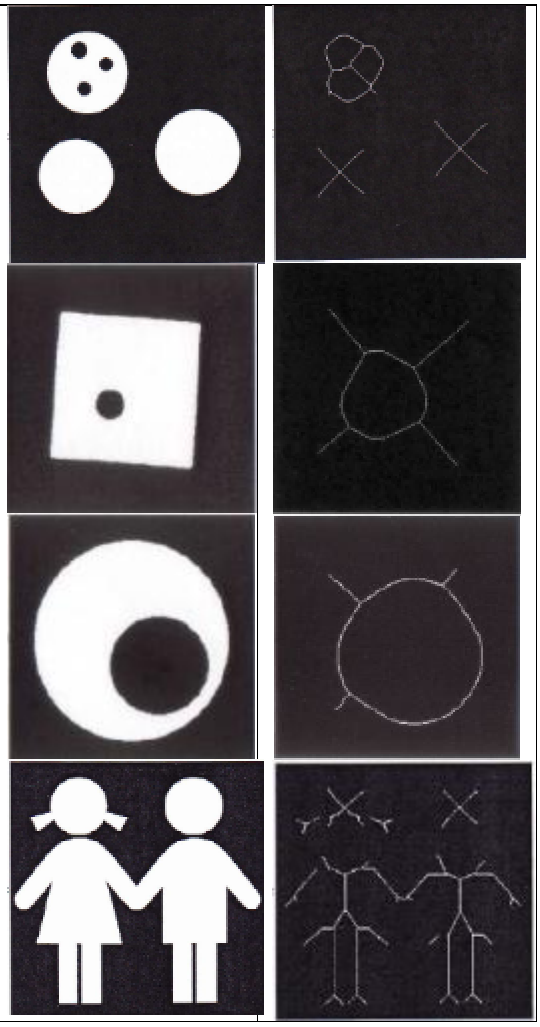
<u>Closing (Schließung):</u>
ist eine Dilation (Schließung kleiner Löcher kleiner als das Strukturelement) gefolgt von einer Erosion (Wiederherstellung der ursprünglichen Größe des Objekts) mit demselben Strukturelement.
<u>Merkmale</u>: kleine Löcher und dünne Risse werden geschlossen; nahe Objekte verschmelzen

<u>Opening (Öffnung):</u>
ist eine Erosion (Entfernung aller Strukturen kleiner als das Strukturelement) gefolgt von einer Dilation (Wiederherstellung der ursprünglichen Größe, mit Ausnahme der vollständig entfernten Strukturen) mit demselben Strukturelement.
Merkmale: öffnet/trennt dünn verbundene Objekte; eliminiert kleine Störobjekte und dünne Linien

<u>Medialachsentransformation/Mittelachsentransformation</u>:
ist ein Skelettierungsalgorithmus, das ein Objekt eines Bildes auf einen 1px reduziert, während die Topologie erhalten bleibt. Die ausgedünnte Form wird Skelett genannt. Dieser Algorithmus stellt sicher, dass das Skelett in der Mitte des Objekts liegt. Die Mittelachse eines Objekts ist die Menge aller inneren Objektpixel, für welche die beiden kürzesten Abstände zum Objektrand gleich sind. Das Skelett ist der Ort der Mittelpunkte aller vollständig im Objekt liegenden Kreise, die den Objektrand mindestens zweimal berühren.

Matlab & Begründung	Input-Image (`I`)	Output-Image (`E`)
`E = imdilate(I,strel('disk',5))` Das Strukturelement (hier ein Kreis mit Durchmesser von 5px) besteht aus weißen Pixel → alle Kreise werden um 5px verkleinert. Da der Kreisring einen Durchmesser kleiner als 5px hat, verschwindet dieser.		
`E = imerode(I,strel('disk',5))` Das Strukturelement (hier ein Kreis mit Durchmesser von 5px) besteht aus weißen Pixel → alle Kreise werden um 5px vergrößert. Da der Kreisring einen Durchmesser kleiner als 5px hat, verschwindet dieser.		
`E = imerode(I,strel('disk',11))` Das Strukturelement (hier ein Kreis mit Durchmesser von 11px) besteht aus weißen Pixel → der Kreis wird um 11px vergrößert.		

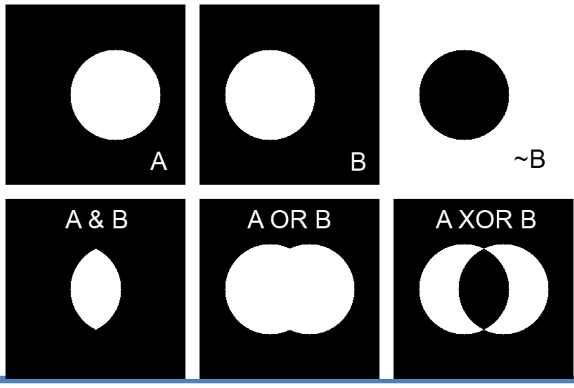
`E = imclose(I,strel('disk',10))` Das Strukturelement (hier ein Kreis mit Durchmesser von 10px) besteht aus weißen Pixel → Closing: Köpfe verschmelzen mit dem Körper (Hals); Zwischenräume zwischen den Beinen verschmelzen. genauer: Closing ist ***Dilatation*** [Köpfe verschmelzen mit dem Körper (Hals); Zwischenräume zwischen den Beinen verschmelzen; Objekte werden um 11px größer] gefolgt von ***Erosion*** [Objekte werden um 11px kleiner und haben somit die ursprüngliche Größe]		
`E = imclose(I,strel('disk',11))` Das Strukturelement (hier ein Kreis mit Durchmesser von 11px) besteht aus weißen Pixel → Alle Kreise mit dem Durchmesser kleiner gleich als 11px werden geschlossen, der Kreisring wird aufgefüllt. genauer: Closing ist ***Dilatation*** [Kreisring sowie die kleinen Kreise verschwinden; die größeren Kreise werden um 11px verkleinert] gefolgt von ***Erosion*** [die Kreise die aus der Dilatation resultieren werden um 11px vergrößert und haben somit die ursprüngliche Größe]		
`E = imclose(I,strel('disk',30))` Das Strukturelement (hier ein Kreis mit Durchmesser von 30px) besteht aus weißen Pixel → Alle Kreise mit dem Durchmesser kleiner gleich als 30px werden geschlossen, der Kreisring wird aufgefüllt. genauer: Closing ist ***Dilatation*** [Kreisring sowie die Kreise verschwinden] gefolgt von ***Erosion*** [der Kreis der aus der Dilatation resultiert, bleibt erhalten]		
`E = imopen(~I,strel('disk',10))` Bevor das Opening angewendet wird, wird das Eingangsbild invertiert → aus weiß wird schwarz und umgekehrt. Das Strukturelement (hier ein Kreis mit Durchmesser von 10px) besteht aus weißen Pixel → Konturen des Eingangsbildes werden im Ausgangsbild abgerundet (wegen Strukturelement). genauer: Closing ist ***Erosion*** [Objekte werden um 10px verkleinert]; gefolgt von ***Dilatation*** [das aus der Dilatation resultieren Bild wird um 10px vergrößert]		
`E = imopen(I,strel('disk',3))` Das Strukturelement (hier ein Kreis mit Durchmesser von 3px) besteht aus weißen Pixel → kleine Störpixel werden eliminiert; übrig bleiben nur die Kreise mit dem Durchmesser von 3px. genauer: Closing ist ***Erosion*** [Objekte werden um 3px verkleinert; Kreise gleiner gleich 3px verschwinden] gefolgt von ***Dilatation*** [der Kreis der aus der Dilatation resultiert, wird um 3px vergrößert]		
`E = imopen(I,strel('line',11,90))` Das Strukturelement (hier eine vertikale 11px lange Linie) besteht aus weißen Pixel → alle horizontalen sowie diagonalen Linien werden eliminiert; übrig bleiben nur die vertikalen Linien. genauer: Closing ist ***Erosion*** [nur die vertikalen Linien bleiben erhalten und werden um 11px verkleinert; restlichen Linien verschwinden] gefolgt von ***Dilatation*** [die vertikalen Linien, die aus der Dilatation resultieren, wird um 11px verlängert]		

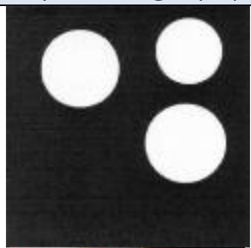
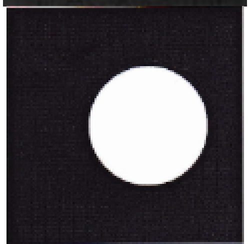
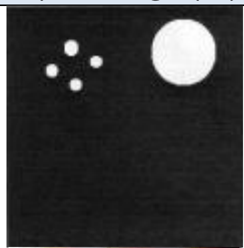
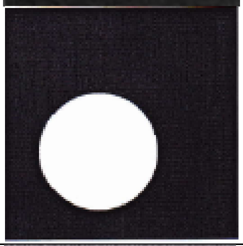
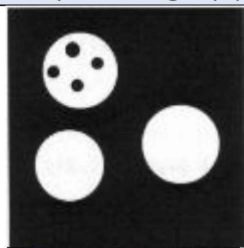
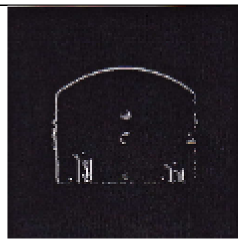
`E = bwmorph(I,'skel',Inf)` **Medialachsentransformation/Mittelachsentransformation:** Objekte werden auf 1px reduziert, während die Topologie erhalten bleibt. Die ausgedünnte Form wird Skelett genannt und liegt in der Mitte des Objektes. Das Skelett ist der Ort der Mittelpunkte aller vollständig im Objekt liegenden Kreise, die den Objektrand mindestens zweimal berühren.	

Logische Operationen

„and"-Verknüpfung:
Bei der und-Verknüpfung wird ein Bit eines Pixels des Ergebnisbildes nur dann gesetzt, wenn die entsprechenden Bits in den beiden Eingangsbildern gesetzt sind. → bestimmte Bildbereiche ausmaskieren, bestimmte Bitebenen selektieren

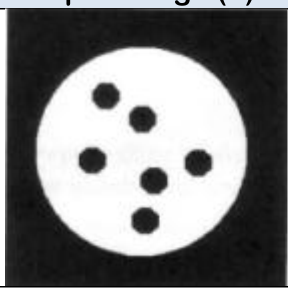
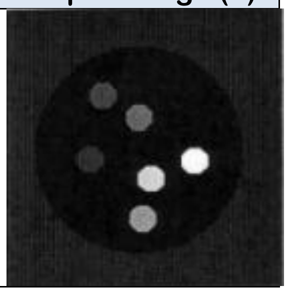
„or"-Verknüpfung:
Ein Bit eines Pixels des Ergebnisbildes wird dann gesetzt, wenn die entsprechenden Bits in mindestens einem der beiden Eingangsbildern gesetzt sind. → hellere Bereiche setzten sich durch.



Matlab & Begründung	Input-Image (I_1)	Input-Image (I_2)	Output-Image (E)
E = and(I_1,not I_2) Das Bild I_2 wird mit „not" invertiert (weiße Pixel werden schwarz und umgekehrt). Danach werden beide Bilder mit „und" verknüpft → dies liefert das Ergebnisbild E: Bei der und-Verknüpfung wird ein Bit eines Pixels des Ergebnisbildes nur dann gesetzt, wenn die entsprechenden Bits in den beiden Eingangsbildern gesetzt sind. → bestimmte Bildbereiche ausmaskieren, bestimmte Bitebenen selektieren	 	 	 
E = or(im2bw(I_1,160/255), im2bw(I_2,160/255)) Beide Eingabebilder werden zuerst in Binärbilder umgewandelt: Alle vorkommenden Grauwerte ≤ 160 → schwarz; restlichen Grauwerte 161-255 → weiß. Die beiden resultierenden Binärbilder werden dann mit „or" verknüpft und liefern das Ergebnisbild E: Ein Bit eines Pixels des Ergebnisbildes wird dann gesetzt, wenn die entsprechenden Bits in mindestens einem der beiden Eingangsbildern gesetzt sind → hellere Bereiche setzten sich durch.			

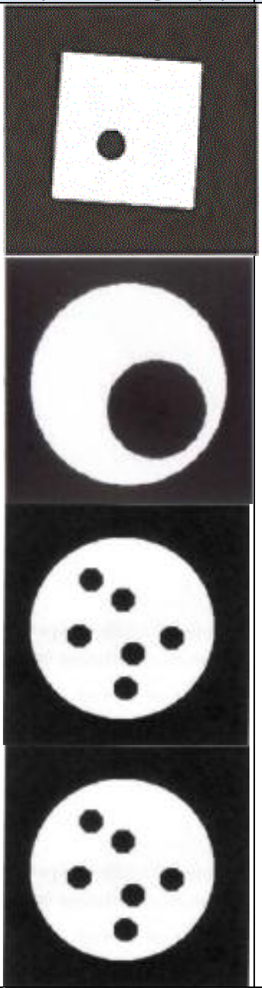
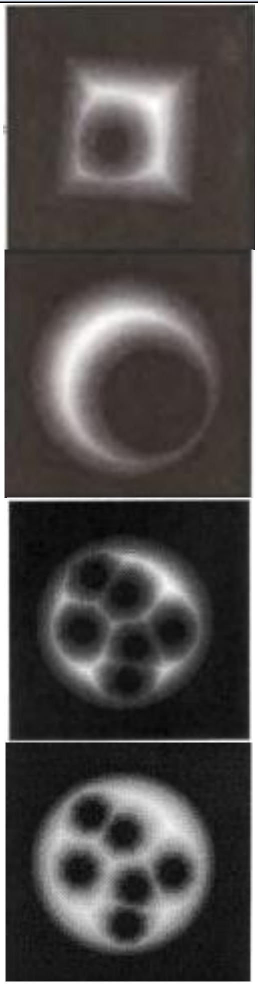
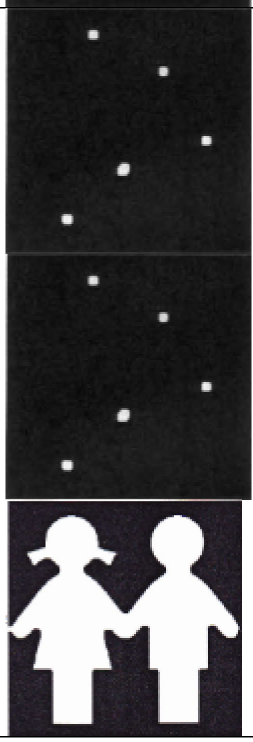
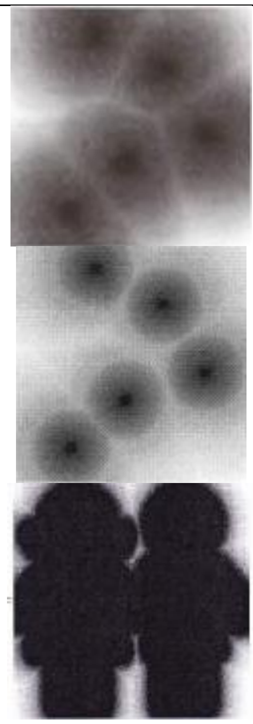
Labeling

Jede zusammengehörige Region (Blob) wird ein Label zugewiesen. Pixel mit dem Nullwert gehören zum Hintergrund. Um die Zuteilung sichtbar zu machen, bekommt jeder Blob eine eigene Farbe/Grauwert.

Matlab & Begründung	Input-Image (I)	Output-Image (E)
E = bwlabel(~I) Labeling (Etikettierung): Jede zusammengehörige Region (Blob) wird ein Label zugewiesen. Pixel mit dem Nullwert gehören zum Hintergrund. Um die Zuteilung sichtbar zu machen, bekommt jeder Blob eine eigene Farbe/Grauwert.		

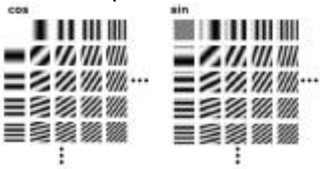
Distanztransformation

Dabei wird für jedes Objektpixel der Wert 1 (weiß) durch den Wert des kürzesten Abstands (Berechnung mittels: Euklidischer-, Cityblock-, oder Chessboard-Distanz) zum Objektrand ersetzt → Distanzkarte: Je heller der Grauwert, desto größer ist die minimale Distanz eines Bildpunktes vom Rand.

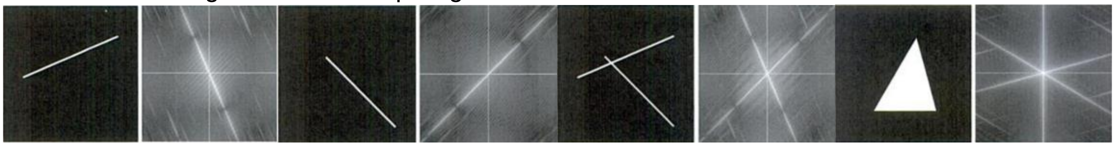
Matlab & Begründung	Input-Image (`I`)	Output-Image (`E`)
`E = bwdist(~I)` Durch die Distanztransformation (mittels der Euklidischer-Distanz) invertiert sich das Eingangsbild, daher wird es invertiert um dies umzukehren → dadurch erhält man die Distanzkarte vom Eingangsbild. Je heller der Grauwert, desto größer ist die minimale Distanz eines Bildpunktes vom Rand.		
`E = bwdist(I)` Durch die Distanztransformation (mittels der Euklidischer-Distanz) invertiert sich das Eingangsbild → man erhält die Distanzkarte vom invertierten Eingangsbild. Je heller der Grauwert, desto größer ist die minimale Distanz eines Bildpunktes vom Rand.		

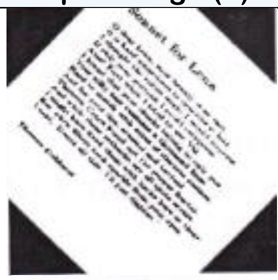
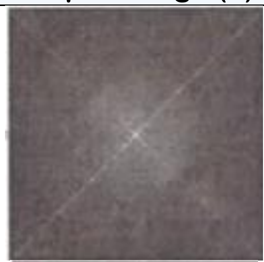
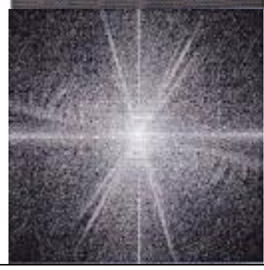
Fourier-Transformation

Fourier-Transformation (FT):
Darstellung im Frequenzraum durch die Ähnlichkeit mit cos und sin Funktionen verschiedener Frequenzen:
Das Bild im Ortsraum wird auf das cos und sin Muster projiziert: Bild und Muster werden punktweise multipliziert und die Produkte werden summiert: man erhält zwei reelle Zahlen welche die Ähnlichkeit des Bildes mit dem Musterpaar beschreibt. Die Projektion wird für alle möglichen Frequenzen und Orientierungen berechnet und auf das Ausgangsbild projiziert.



Fast Fourier Transform (FFT): Effiziente Implementierung der diskreten FT durch divide and conquer Algorithmus: Rekursive Aufteilung der Summe in gerade und ungerade Komponenten bis nur noch eine Zahl zu transformieren ist; dies ist einfach die Zahl selbst; rekursive Kombination der Terme.
Anwendung: effiziente Faltung, Analyse & Manipulation der Bildfrequenzen, Datenkompression
➔ zur Darstellung: Kompression des Amplitudenspektrums (oder Phasensprektrums): Die niedrigen Frequenzen sind im Zentrum dargestellt und hell wiedergegeben. Im Allgemeinen überwiegen sie bei einem Bild. Die höheren Frequenzen sind außen dargestellt. Der Ursprung ist in der Mitte des Plots.



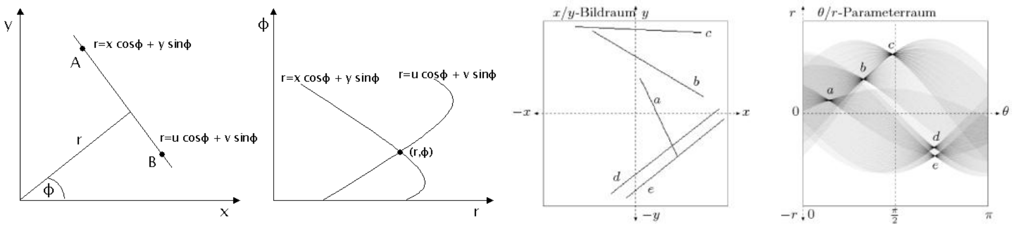
Matlab & Begründung	Input-Image (`I`)	Output-Image (`E`)
`E = log1p(abs(fftshift(fft2(I))))` **Fourier-Transformation:** Darstellung im Fourier-Spektrum bzw. Amplitudenspektrum: Die niedrigen Frequenzen sind im Zentrum dargestellt und hell wiedergegeben. Im Allgemeinen überwiegen sie bei einem Bild. Die höheren Frequenzen sind außen dargestellt. Der Ursprung ist in der Mitte des Plots. Features, die a) horizontal, b) vertikal, c) Diagonal von links unten nach rechts oben, d) Diagonal von links oben nach rechts unten im Originalbild laufen, resultieren in a)vertikalen, b) horizontalen c) Diagonal von links oben nach rechts unten, d) Diagonal von links unten nach rechts oben Komponenten im Fourier-Spektrum → Features im Ortsraum (Originalbild) werden in entgegengesetzter Richtung im Fourier-Spektrum dargestellt.	   	   

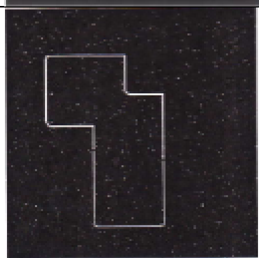
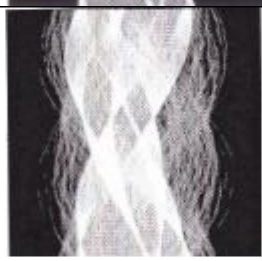
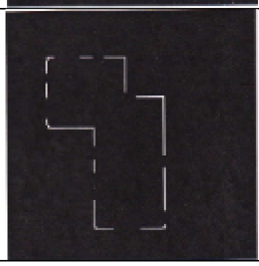
Hough-Transformation

Verfahren zur Detektion von geometrischen Strukturen (Linien, Kreise) in einem Binärbild; ist sehr robust gegenüber Bildstörungen und teilweisen Verdeckungen der Objekte

Liniendetektion: für jedes Vordergrundpixel Berechnung eines Geradenbüschels (Schar von Sinuskurven); Übertragung vom Bildraum ins Houghraum → jede Gerade entspricht einen Punkt im Houghraum (Schnittpunkt der n Sinuskurven) → jedes Geradenbüschel entspricht damit einer Kurve im Transformationsraum

- ⇨ Berechnung eines Geradenbüschel (Schar von Sinuskurven) für jedes Vordergrundpixel
- ⇨ Übertragung jedes Vordergrundpixel vom Koordinatenraum (x,y) in den Hough-Raum (r,phi) →Umwandlung mittels Hesse'sche Normalform **r = x*cos(phi) + y*sin(phi)**
- ⇨ Jede Gerade im (x,y)-Raum entspricht genau einem Punkt im (r,phi)-Raum - jedes Geradenbüschel entspricht damit einer Kurve im Transformationsraum, dem Akkumulator: (Schnittpunkt aller Sinuskurven eines Geradenbüschels = eine Gerade im Houghraum)
  - liegen viele Vordergrundpixel des Originalbildes auf einer Geraden → schneiden sich entsprechend viele Kurven im Akkumulator
  - Wird bei der Transformation der anfänglich leer (schwarz) initialisierte Akkumulator für jede Gerade inkrementiert, entstehen an den Stellen im Akkumulator besonders helle Pixel, auf dessen Gerade viele kollineare Punkte des binarisierten Ausgangsbildes liegen



Matlab & Begründung	Input-Image (I)	Output-Image (E)
`E = „Hough-Transform" (I)` Alle in I vorkommenden Geraden (Segmente) entsprechen einem Geradenbüschel in E : Schnittpunkt aller Sinuskurven eines Geradenbüschels = eine Gerade im Houghraum		
`E = „Hough-Transform" (I)` Alle in I vorkommenden Geraden (Segmente) entsprechen einem Geradenbüschel in E : Schnittpunkt aller Sinuskurven eines Geradenbüschels = eine Gerade im Houghraum; die Störungen im Bild I entsprechen die dünnen Geradenbüschel im Ergebnisbild		
`E = „Hough-Transform" (I)` Alle in I vorkommenden Geraden (Segmente) entsprechen einem Geradenbüschel in E : Schnittpunkt aller Sinuskurven eines Geradenbüschels = eine Gerade im Houghraum ; die Zwischenräume im I Bild entsprechen die Zwischenräume in E		

Kantendetektor

<u>Canny-Kantendetektor</u>: (nicht linearer Filter, Hochpassfilter)
Der Canny-Kantendetektor bedient sich der 1.Ableitungen des Bildes. Zusätzlich benötigt dieser einen unteren und einen oberen Schwellwert: Werte über dem oberen Schwellwert werden als sichere Kanten und Werte unter dem unteren Schwellwert als keine Kanten betrachtet. Für die Werte zwischen den beiden Grenzwerten gilt folgende Regel: Sind diese Werte mit sicheren Kanten verbunden, dann werden sie zu den Kanten gezählt, ansonsten werden sie nicht berücksichtigt → versucht zusammenhängende Kantenverläufe zu schließen.

- Sigma: gibt die Breite des Gaußkerns an (Stärke der Glättung) → je größer, desto weniger Details und desto ungenauere Kanten
- ist der obere Schwellwert hoch und niedriger Schwellwert niedrig → Kanten ohne starken Kontrast fehlen
- ist der obere Schwellwert niedrig und niedriger Schwellwert hoch → gestörte Kanten können zerbrechen

<u>Sobel-Kantendetektor</u>: (linearer Filter, Hochpassfilter)
Der Sobel-Operator ist ein richtungsabhängiger Kantenfilter und verwendet einen Differenzenfilter der zusätzlich eine Grass'sche Mittelung durchführt. Somit sind die Kanten nach der Operation geglättet. Das resultierende Kantenbild entspricht der ersten Ableitung des Originalbildes in horizontaler und vertikaler Richtung. Anders orientierte Kanten werden der Neigung entsprechend weniger stark hervorgehoben.
Man unterscheidet: horizontale, vertikale und diagonale Sobel-Operatoren. Diese Operatoren weisen unterschiedliche Filterkerne auf.
<u>Vorteile</u>: Rauschunterdrückung durch gewichtete Mittelung, Kanten werden auch bei einem relativ flachen Grauwertübergang deutlich hervorgehoben
<u>Nachteil</u>: es können ausschließlich Kanten detektiert werden, die sich senkrecht zur Laufrichtung befinden, es kommt zu einer Verbreiterung der Kanten.

Horizontaler Sobelfilter

-1	-2	-1
0	0	0
1	2	1

Vertikaler Sobelfilter

-1	0	1
-2	0	2
-1	0	1

Diagonaler Sobelfilter

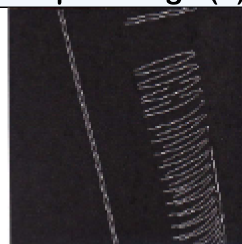
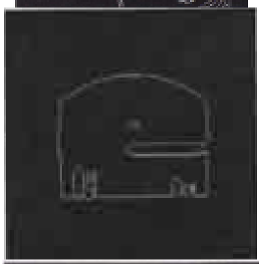
-2	-1	0
-1	0	1
0	1	2

<u>Lablace-Kantendetektor</u>: (linearer Filter, Hochpassfilter)
Der Laplace-Operator ist ein richtungsunabhängiger Kantenfilter, der sowohl in vertikale, horizontale als auch diagonale Richtung arbeitet und die 2. Ableitung des Bildes darstellt. Jede Kante führt beim Laplace-Filter zu Doppelkonturen. Der Kantenort der Nulldurchgang (Zero-Crossing).
<u>Vorteile</u>: bevorzugt steile Grauwertdifferenzen, setzt alle homogenen Bildareale auf Null
<u>Nachteil</u>: Rauschen wird verstärkt

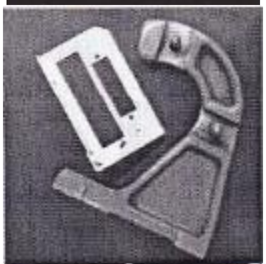
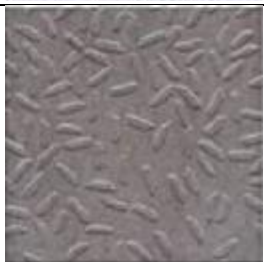
<table>
<tr><th>Matlab & Begründung</th><th>Input-Image (<code>I</code>)</th><th>Output-Image (<code>E</code>)</th></tr>
<tr><td>
<code>E = edge(I,'canny',[0.4,0.5],1)</code>
Canny-Kantendetektor: Aus dem Grauwertbild wird ein Binärbild erzeugt, bei dem nur noch die Kanten des Ursprungbildes zu sehen sind.
• Sigma: gibt die Breite des Gaußkerns an (Stärke der Glättung) → je größer, desto weniger Details und desto ungenauere Kanten
• ist der obere Schwellwert hoch und niedriger Schwellwert niedrig → Kanten ohne starken Kontrast fehlen
• ist der obere Schwellwert niedrig und niedriger Schwellwert hoch → gestörte Kanten können zerbrechen

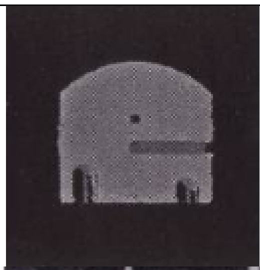
Werte über dem oberen Schwellwert werden als sichere Kanten und Werte unter dem unteren Schwellwert als keine Kanten betrachtet. Für die Werte zwischen den beiden Grenzwerten gilt folgende Regel: Sind diese Werte mit sicheren Kanten verbunden, dann werden sie zu den Kanten gezählt, ansonsten werden sie nicht berücksichtigt → versucht zusammenhängende Kantenverläufe zu schließen.
</td><td>



</td><td>



</td></tr>
</table>

`E = conv2(I,[1 0 -1; 2 0 -2; 1 0 -1])` Bild `I` wird mit dem Faltungskern [1 0 **-1**; 2 0 **-2**; 1 0 **-1**] gefaltet → wegen den negativen Werten in der 3. Spalte der Matrix des Faltungskerns werden die **vertikalen Linien** mehr betont, als die horizontalen. → Hochpassfilter: Sobel Kantendetektor in Gx-Richtung (Gradientenfilter)	    	    
`E = conv2(I,[1 2 1; 0 0 0; -1 -2 -1])` Bild `I` wird mit dem Faltungskern [1 2 1; 0 0 0; **-1 -2 -1**] gefaltet → wegen den negativen Werten in der 3. Zeile der Matrix des Faltungskerns werden die **horizontalen Linien** mehr betont, als die vertikalen. → Hochpassfilter: Sobel Kantendetektor in Gy-Richtung (Gradientenfilter)	 	 

	 	 
`E = conv2(I,fspecial('laplacian'))` Der Laplace-Operator ist ein richtungsunabhängiger Kantenfilter → hebt Schatten, die an den Rändern von Objekten entstehen, hervor. Jede Kante führt beim Laplace-Filter zu Doppelkonturen → stellt die 2.Ableitung des Bildes dar