Praktikum Bildverarbeitung

Programmthema: Polarcoding/Polarabtastung
Programmiersprache: Object Pascal
Programmname: BVA.exe

Klante, Rüdiger (B_TInf 2381, 4. Fachsemester)
Bertram, Alexander (B_TInf 2616, 4. Fachsemester)

Inhaltsverzeichnis

1. Allgemeine Problemstellung

Im Rahmen dieses Praktikums sollen Algorithmen zur Polarabtastung implementiert und ihre Anwendungsmöglichkeiten sowie ihre Grenzen experemintell untersucht werden.

Dieses Programm deckt alle Teile des Problems ab.

2. Benutzerhandbuch

2.1. Ablaufbedingungen

Für den Ablauf des Programms sind folgende Hard- und Softwarekomponenten erforderlich:

Hardware:

- IBM kompatibler PC
- Grafikkarte
- Monitor
- Maus

Software:

- MS Windows (ab '95)

Bildmaterial:

Die zu verwendenden Bilder müssen am BVA-Tisch mit dem Standard-Objektiv aufgenommen werden. Blendeneinstellungen können soweit variiert werden, dass auf dem aufgenommenen Bild das Objekt mit bloßem Auge noch ganz zu erkennen ist, anderenfalls treten Fehler in der Vorverarbeitung sowie der Klassifikation auf.

Die Ausleuchtung sollte relativ gleichmäßig sein, da ein globales Schwellwertverfahren angewendet wird.

Das Objekt auf dem Bild sollte nicht zu nah am Rand liegen, da der Rand bei den Filterungsoperationen nicht verarbeitet wird.

Weiterhin darf sich nur ein(!) Objekt auf dem Bild befinden.

Die mitgelieferten Bilder befinden sich im Unterverzeichnis „Bilder".

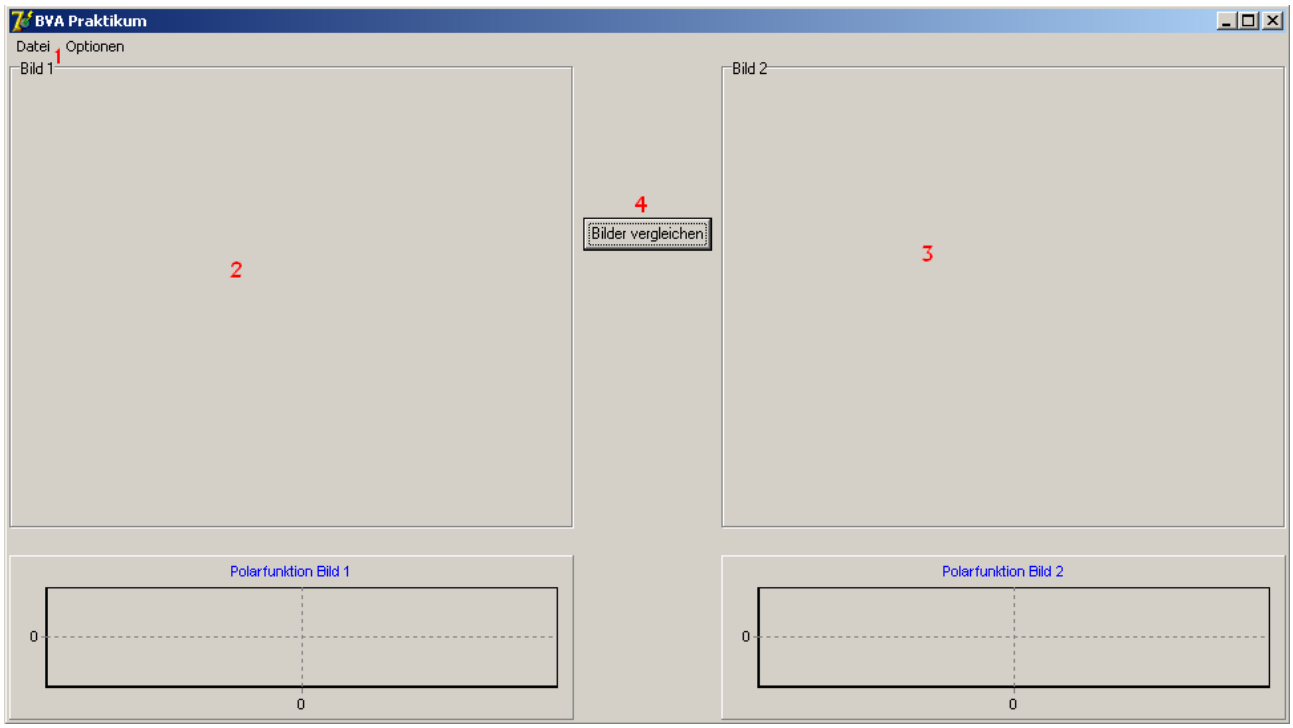
2.2. Programminstallation und Programmstart

Das Programm BVA.exe wird durch einen Doppelklick gestartet und ist sofort einsatzbereit.

2.3. Bedienungsanleitung

2.3.1. Programmstart

Starten Sie das Programm wie unter 2.2 beschrieben. Sie sehen folgendes Fenster auf dem Monitor:



1: Hauptmenü

2: Bild 1

3: Bild 2

4: Bildervergleich

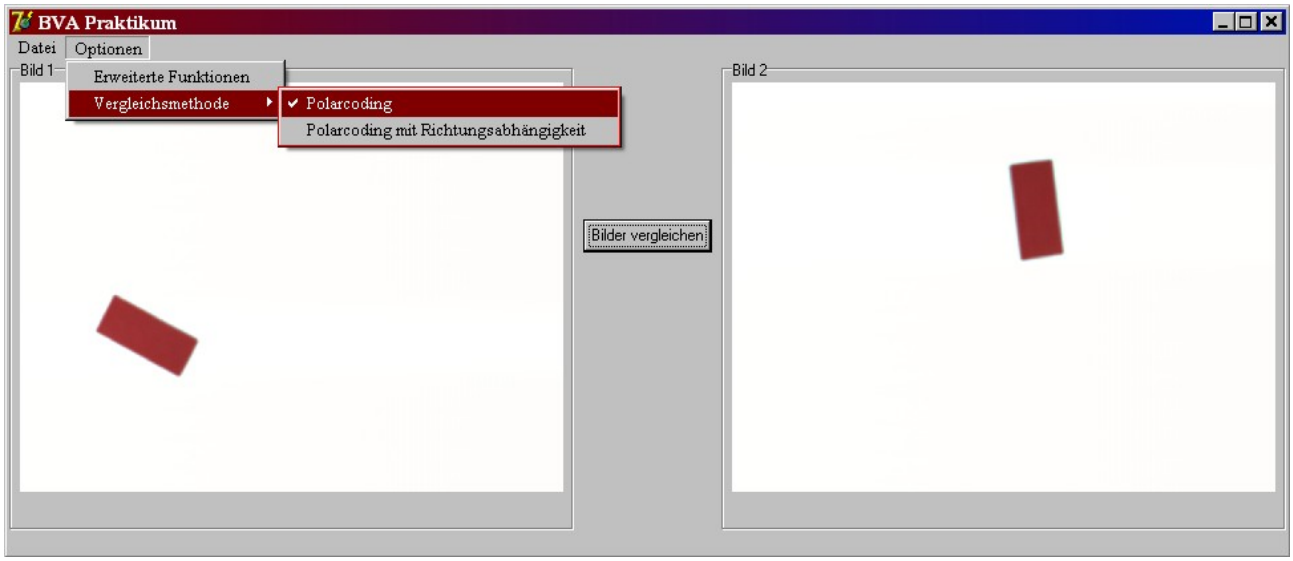
2.3.2. Bilder laden

Klicken sie als Erstes im Hauptmenü auf Datei => Bild 1 öffnen. Es öffnet sich ein Bildauswahldialog, in dem Sie das erste Bild auswählen können. Klicken Sie nun auf Öffnen. Das Bild wird in der linken Hälfte angezeigt. Analog gehen Sie beim zweiten Bild vor. Dieses Bild wird in der rechten Hälfte angezeigt.

Voraussetzung für richtige Funktion des Programms sind Bilder, die nur ein (!) einziges Objekt beinhalten!

2.3.3. Bildervergleich

Um die Bilder zu vergleichen, wählen Sie zunächst eine der beiden Vergleichsmethoden aus.



Auf die Unterschiede zwischen den beiden zur Auswahl stehenden Methoden wird später eingegangen.

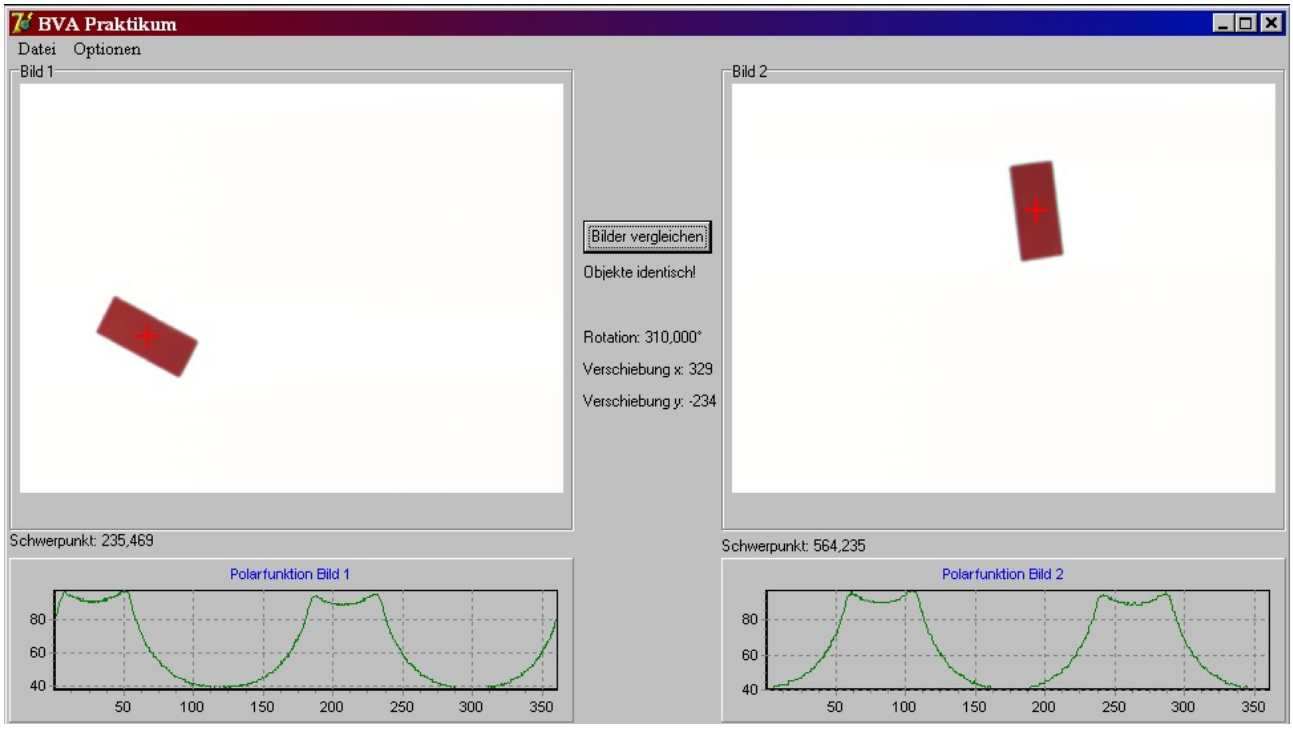
Nach Auswahl des gewünschten Verfahrens, klicken Sie nun den Button „Bilder vergleichen" in der Fenstermitte an. Der Vergleichsvorgang nimmt einige Sekunden in Anspruch, da aufwändige Filterungen in der Vorverarbeitung angewandt werden.

Nachdem der Mauszeiger von der zuvor angezeigten Sanduhr wieder zum Zeiger geworden ist, ist der Vergleich durchgeführt.

Nach abgeschlossenem Vergleich werden nun unter den beiden Bildern die Koordinaten der jeweiligen Flächenschwerpunkte (im Bild selbst durch ein rotes Kreuz markiert) sowie die Abtastfunktionen der gewählten Vergleichsmethode angezeigt.

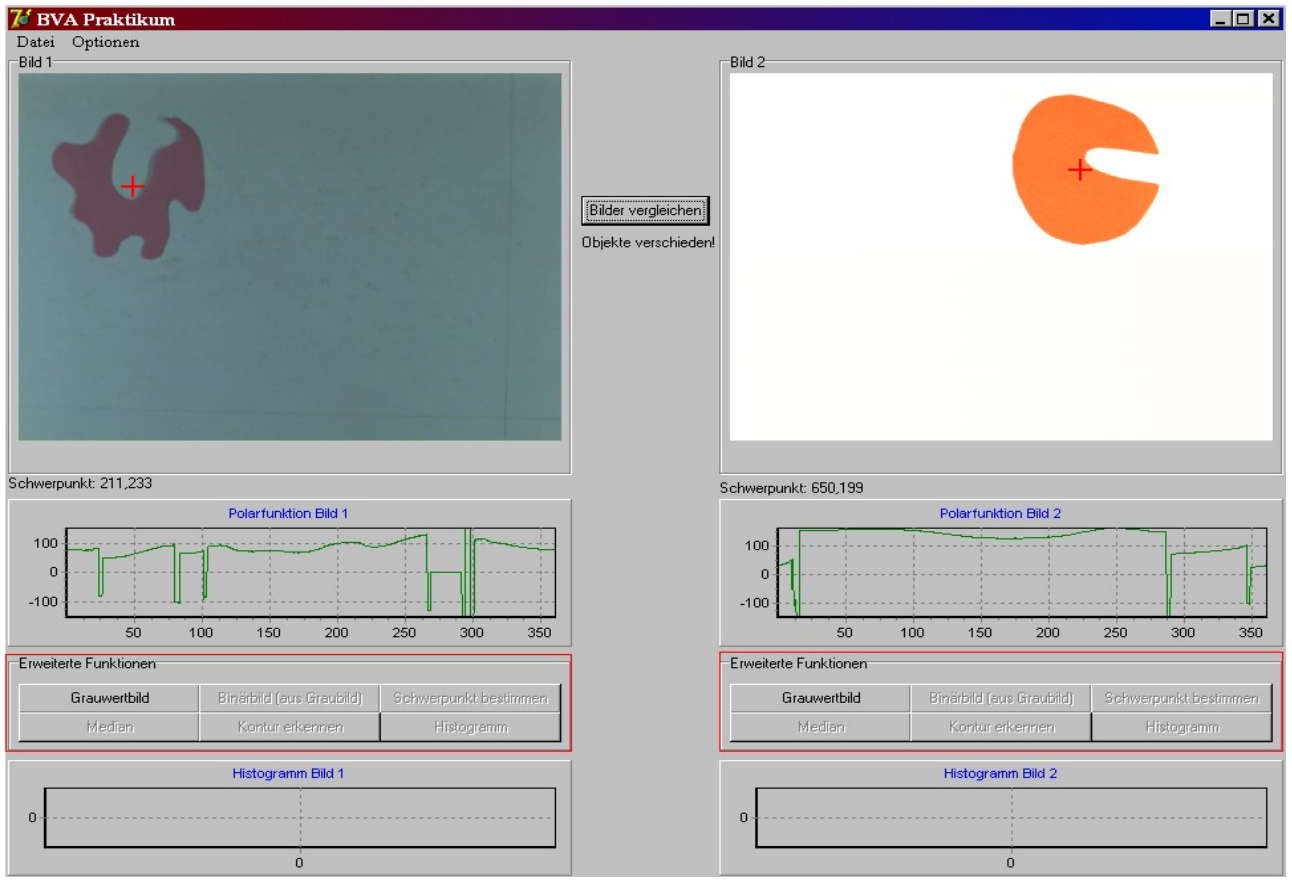
Bei Übereinstimmung der Bilder erscheint in der Bildmitte, neben einer Erfolgsmeldung, die Rotation des rechten gegenüber dem linken Bild in Grad, sowie die translatorischen Freiheitsgrade gemessen in Pixeln. (siehe Screenshot auf der folgenden Seite)

Bei fehlgeschlagenem Vergleich erscheint die Meldung „Bilder verschieden", es werden keine Freiheitsgrade angezeigt.



2.3.4. Zusätzliche Funktionen

Im Hauptmenü finden Sie unter Optionen den Menüpunkt „Erweiterte Funktionen“. Schalten Sie diese ein, sieht das Programmfenster wie folgt aus:



Sie sehen nun zusätzliche Buttons unter den Bildern:

- Grauwertbild: Erstellt aus dem Farbbild ein Grauwertbild.
- Binärbild: Erstellt aus dem Grauwertbild ein Binärbild, welches aus den Farben schwarz (Hintergrund) und weiss (Objekt) besteht.
- Schwerpunkt bestimmen: Der Schwerpunkt des Objektes wird berechnet und mit einem roten Kreuz markiert. Die Pixelkoordianten werden zusätzlich unter dem Bild ausgegeben.
- Median: Wendet einen Medianfilter auf das Bild mit einer Maskengröße von 5x5 an.
- Kontur erkennen: Markiert die Kontur des Objekts rot und zeigt im Diagramm unter dem jeweiligen Bild ihre Funktion an.
- Histogramm: Stellt die Häufigkeiten der einzelnen Grauwerte im Diagramm unter den Buttons dar.

2.4. Fehlermeldungen

Fehlermeldung	***Ursache***	***Maßnahme***
Nicht genug Bilder geladen.	Eins der beiden bzw. beide Bilder, die verglichen werden sollen, sind leer.	Laden Sie zwei Bilder.

2.5. Wiederanlaufbedingungen

Sollte es zum Programmabsturz kommen, kann es wie unter Punkt 2.2 neu gestartet werden.

3. Programmierhandbuch

3.1. Entwicklungskonfiguration

Hardware:

- IBM kompatibler PC
- Monitor
- Tastatur
- Maus

Software:

- Betriebssystem: Windows XP mit Service Pack 2
- IDE: Delphi 7

3.2. Problemanalyse und Realisation

Das Programm lässt sich in 2 wesentlich Probleme unterteilen: Bildvorverarbeitung und Klassifizierung.

3.2.1. Bildvorverarbeitung:

Als erster Schritt findet in der Bildverarbeitung die Bildvorverarbeitung statt. Hier werden z.B. Störungen ausgeblendet, ein Binärbild erzeugt oder die Konturen verstärkt. Dazu gibt es lokale und globale Verfahren.

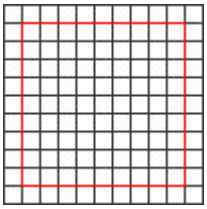
Lokale Verfahren betrachten einen Ausschnitt des Bildes und verarbeiten diesen. Im Extremfall sogar einzelne Pixel.

Zu den lokalen Verfahren zählt die lokale Filterung, in der auf jedes Pixel des Bildes ein Filteroperator, der linear oder nicht linear sein kann, mit einer Maske der Größe m (m = ungerade, typisch 3, 7, 9, ...) angewendet wird. Im Gegensatz zu den nicht linearen Filtern können lineare Filter durch eine Faltung beschrieben werden:

$$g(x,y)=\frac{1}{m^2}\sum_{i=-k}^{k}\sum_{j=-k}^{k}s(x+i,y+j)*h(i,j) \qquad k=\frac{m-1}{2}$$

s(x, y): Ursprungsbild

h(i, j): Faltungskern



g(x, y): Ergebnisbild

Das neu entstandene Bild ist rot markiert. Was an den Rändern passieren soll, ist je nach Problemstellung zu entscheiden.

Globale Verfahren werden nicht auf einzelne Pixel sondern auf das ganze Bild angewendet. Die Fouriertransformation ist eine der Methoden, die die globalen Verfahren vertritt. Bei dieser Methode wird das Bild aus dem Ortsbereich in den Frequenzbereich transformiert, die relevanten Anteile werden heraus gefiltert und das Bild zurück in den Ortsbereich transformiert.

In unserem Programm finden folgende Vorverarbeitungsschritte statt:

- Erzeugen eines Grauwertbildes
- Filterung des Bildes mit dem Medianfilter
- Erzeugung eines Binärbildes

Erzeugen eines Grauwertbildes:

Aus dem Eingangsbild wird ein Grauwertbild erzeugt, indem über die drei Farbkanäle (Rot, Grün, Blau) der einzelnen Pixel gemittelt wird.

Filterung des Bildes mit dem Medianfilter:

Der Medianfilter gehört zu den nicht linearen Glättungsfiltern. Er eignet sich sehr gut, um Ausreisser zu eliminieren und funktioniert folgendermaßen:

Die Umgebung des zu setzenden Pixels wird abgetastet. Die Grauwerte werden aufsteigend sortiert und der mittlere (nicht der Mittelwert!) Grauwert wird an der Stelle gesetzt.

In unserem Falle wird der Medianfilter mit einer Maske von 5 Pixeln verwendet. Der Rand von zwei Pixeln, der dadurch entsteht, wird nicht verändert.

Für unsere Problemstellung hat sich durch Experimente herausgestellt, dass dieser Filter am Besten geeignet ist, um anschließend ein Binärbild zu erzeugen.

Erzeugung eines Binärbildes:

Für die Erzeugung eines Binärbildes wird eine Schwelle gebraucht, mit deren Hilfe entschieden wird, ob ein Grauwert zum Objekt oder Hintergrund gehört.

$$s'(x,y)=\begin{cases}0, falls & s(x,y)\geqslant Schwelle\\ 1, sonst & \end{cases}$$

Für die Bestimmung der Schwelle gibt es mehrere Verfahren:

Fixe Schwelle:

Eine fixe Schwelle wird vorgegeben und bei allen Bildern verwendet. Bei unterschiedlichen Beleuchtungssituationen ist dieses Verfahren nicht zu gebrauchen.

Adaptive Schwelle:

Lokal:

Die Schwelle wird für jedes Pixel einzeln berechnet. Und zwar analog zu der weiter oben beschriebenen lokalen Filterung:

$$c=\frac{1}{m^2}\sum_{i=0}^{m-1}\sum_{j=0}^{m-1}s(x+k-j\,,\,y+k-j) \qquad\qquad k=\frac{m-1}{2}$$

Dieses Verfahren hat sich als ungeeignet für unsere Problemstellung herausgestellt, da wir beim Experimentieren ein „Binärbild" erzeugt haben, in dem nur die Konturen verstärkt waren und das sonst weiße und schwarze Pixel durcheinander gewürfelt („binäres Rauschen") enthielt.

Global: mit bekannter Objektgröße (relativ):

Belegt ein Objekt p% der Bildfläche, ist c die maximale Zahl für die gilt:

$$\frac{\sum_{g=0}^{c}h(g)}{100}\leqslant 1-\left(\frac{p}{100}\right)$$

g: Grauwert

h(g): Häufigkeit des Grauwertes

Dieses Verfahren hat sich ebenfalls als nicht geeignet herausgestellt, da man vor der Verarbeitung wissen muss, wieviel Prozent der Bildfläche ein Objekt belegt.

Global: Verfahren von Otsu:

Das Histogramm eines Bildes wird in 2 Klassen unterteilt: Objekt- und Hintergrundpixel. Es wird ein Grauwert gesucht, bei dem die Varianz zwischen den Klassen maximiert wird.

Dieses Verfahren haben wir in unser Programm implementiert, da es mit Hilfe von statistischen Mitteln arbeitet und bei unserer Problemstellung die besten Ergebnisse gezeigt hat.

Dieses Verfahren zeigt seine Grenzen, wenn das Bild an verschiedenen Stellen unterschiedlich stark ausgeleuchtet ist.

3.2.2. Klassifizierung

Bei der Klassifizierung wird ein Objekt „gelernt" und mit einem zweiten Objekt verglichen. Durch die Aufgabenstellung war das Polarcoding-Verfahren vorgegeben, welches wir in zwei Varianten implementiert haben, die später erläutert werden.

Beim Polarcoding-Verfahren wird ein Objekt ausgehend von einem Referenzpunkt, z.B. Flächenschwerpunkt, in festen Winkelschritten abgetastet und die Schnittpunkte mit der Kontur in ein Koordinatensystem eingetragen. Es entsteht die so genannte Polarfunktion. Die Abtasteigenschaften können durch Bewertung der Krümmung erweitert werden.

Wir haben das Polarcoding, wie oben erwähnt, in zwei Varianten implementiert. Die erste Variante ist simples Polarcoding, bei dem der Abstand mittels eines

vom Flächenschwerpunkt ausgesandten Strahls zur Kontur gemessen wird, ohne dabei die Krümmungen der Kontur oder Durchbrüche im Objekt zu beachten.

In der zweiten Variante haben wir den Algorithmus um das Erkennen von Durchbrüchen sowie die Bewertung der Krümmung in der Kontur erweitert.

Bei mehreren Schnittpunkten je Winkelschritt, also bei auftretenden Durchbrüchen, wird der Mittelwert aus allen Abständen von Schwerpunkt zu den Schnittpunkten gebildet.

Die Krümmung wird anschließend wie folgt berechnet: Es wird der Abstand eines Abtastwertes und seiner Nachbarn zum Flächenschwerpunkt betrachtet. Sind die Abstände beider Nachbarn kleiner als der Abstand des Referenzpunktes, ist die Krümmung positiv, sonst ist sie negativ. Negative Abtastwerte repräsentieren Wendepunkte in der Kontur.

Als Referenzpunkt haben wir den Flächenschwerpunkt ausgewält, der durch Mittelung der x und y Koordinaten des Objektes über die Objektfläche berechnet wird. Durch dieses Verfahren kann bei einigen Objekten der Flächenschwerpunkt ausserhalb des Objektes liegen. Das haben wir jedoch dabei belassen, da der Flächenschwerpunkt bei jeder Objektrotation und -translation relativ zum Objekt gleich bleibt.

Um zwei Objekte auf Ähnlichkeit zu überprüfen, werden beide Polarfunktionen immer wieder um einen Abtastschritt gegeneinander verschoben und eine Differenz der Absolutwerte gebildet. Beim Entdecken von Ausreissern, die größer sind, als eine bestimmte Toleranzschwelle, wird die Anzahl der Fehler inkrementiert. Ist die Anzahl am Ende des Vergleichs größer als eine bestimmte Schwelle, sind die Objekte nicht identisch.

Stimmen die Objekte überein, wird aus dem aktuellen Versatz der beiden Kurven gegeneinander der Winkel errechnet, um den ein Objekt gegenüber dem anderen gedreht ist.

Als drittes haben wir mit einem Verfahren experimentiert, das nicht vom Schwerpunkt ausgeht und die Kontur „sucht", sondern auf der Kontur „entlangwandert" und von jedem Pixel des Objektrandes den Abstand zum Objektschwerpunkt ausmisst um diesen dann als Ergebnis zu speichern.

Da dieses Verfahren Probleme beim Vergleich von Objekten mit sich bringt, weil die Zahl der Pixel des Randes bei translatorischen und rotatorischen Verschiebungen varriert, haben wir uns gegen eine Implementation zum Objektvergleich entschieden. Weiterhin entspricht dieses Verfahren nicht der Vorgabe, Polarcoding zu realisieren, da nicht in äquidistanten Winkelschritten abgetastet wird.

Dieses letzte Verfahren liefert für manche Objekte allerdings so exakte Abtastungen, dass wir es für sinnvoll hielten, sich die Abtastfunktionen für beliebige Objekte unter den „erweiterten Funktionen" anzeigen lassen zu können.

3.3. Beschreibung grundlegender Datenstrukturen

In diesem Projekt werden ausschließlich Arrays zur Speicherung von Objektdaten verwendet.

Zum Ablegen der Ergebnisse des Polarcodings werden für die Weiterverarbeitung teilweise die von Delphi 7 angebotenen dynamischen Arrays, für die endgültige Speicherung normale, statische Arrays genutzt.

Für die Speicherung der Daten nach Konturabtastung werden ebenfalls dynamische Arrays eingesetzt, deren Länge der Anzahl der Konturpixel entspricht.

Typen im Detail:

- *TPolarArray*: statisches Array, eindimensional. Länge entspricht der Anzahl der Winkelschritte, mit der abgetastet wird.
- *TPolarArrayDynamic*: dynamisches Array, eindimensional. Verwendet zur Speicherung der Daten aus der Konturabtastung. Länge entspricht der Anzahl gefundener Pixel auf der Kontur.
- *TPolarArrayMultiJunction*: dynamisches, zweidimensionales Array. Die erste Dimension ist dabei ein statisches Array, dessen Länge durch die Anzahl der Winkelschritte, mit der abgetastet wird, bestimmt ist. Die zweite Dimension repräsentiert die Schnittpunkte mit der Kontur. Ihre Länge entspricht jeweils der Anzahl der Übergänge Objekt/Hintergrund sowie Hintergrund/Objekt.
- *TBitmapArray*: dynamisches, zweidimensionales Array. Dient zur Speicherung von Grauwerten eines Bitmaps zur schnelleren Verarbeitung.
- *TGrayLevelArray*: statisches, eindimensionales Array. Dient zur Speicherung von Histogrammen. Jedes Arrayelement entspricht dabei der Häufigkeit des Grauwerts (0-255) mit dem jeweiligen Arrayindex.

3.4. Programmstruktur und Pseudocode

3.4.1. Modularisierung

Das Projekt ist aufgeteilt in drei units:

- unitMain: Verwaltung der Oberfläche, aufrufen der Prozeduren und Funktionen zur Bildverarbeitung.
- unitImage: Prozeduren und Funktionen zur Vorverarbeitung und Klassifikation der Bilder.
- unitTypes: eigene Datentypen.

Abhängigkeiten:

- In unitMain werden sowohl die unitImage als auch die unitTypes verwendet.
- In unitImage wird die unitTypes verwendet.
- In unitTypes werden keine weiteren units verwendet.

3.4.2. Pseudocode zu ausgewählten Prozeduren/Funktionen

Funktion zur Durchführung des „Polarcoding“:

```
getPolarArray;
begin
  for i := {Winkel von 0-360° in festgelegten Schritten} do
    {laufe im aktuellen Winkel entlang eines Strahls nach außen}
    {wenn das Objekt verlassen wurde, trage die aktuelle Strahllänge
      in das Ergebnisarray an Stelle i ein}
  endfor;
end;
```

Funktion zur Durchführung des „Polarcoding mit Richtungsabhängigkeit“:

```
getPolarArrayDirections;
begin
  for i := {Winkel von 0-360° in festgelegten Schritten} do
  while {Bildbereich ist nicht verlassen} do
    {laufe im aktuellen Winkel entlang eines Strahles nach aussen}
    {wenn ein Übergang Objekt/Hintergrund oder Hintergrund/Objekt
      erreicht ist, erhöhe Zahl gefundener Schnittpunkte}
```

```
    {trage aktuelle Strahllänge in Ergebnisarray an Stelle
      [i,"Nummer des Schnittpunktes] ein}
   endwhile;
  endfor;
  {Bilde den Mittelwert über die Strahllängen an den Schnittpunken für jede
    Winkelposition des Arrays}
  {bewerte Richtungsabhängigkeit der Kontur und modifiziere Ergebnisse}
end;
```

Funktion zur Durchführung des Vergleichs zwischen zwei Objekten:

```
compareObjects;
begin
  for versatz := {Winkel von 0-360° in festgelegten Schritten} do
   for index := {Winkel von 0-360° in festgelegten Schritten} do
    {Bilde die Differenz der Absolutwerte an Stelle [index] im Array von
     Objekt 1 und Stelle[index + versatz] }
      {ist diese Differenz Größer als eine Schwelle, erhöhe die Anzahl der
       aufgetretenen Fehler}
    endfor;
   {Ist die Fehlerzahl kleiner als eine Schwelle, markiere die Objekte als gleich}
  endfor;
end;
```

3.5. Randbedingungen, Wahl der Parameter, Diskussion der Ergebnisse

Schwellwert:

Wie schon weiter oben erwähnt, darf das Bild keine starken Beleuchtungsschwankungen haben, denn spätesten hier gibt das Verfahren von Otsu auf.

Winkelschritte bei der Abtastung:

Diese sollten nicht zu klein gewählt werden. Zwei Schritte pro Grad ist schon ein ganz gutes Maß. Außer mehr Rechenaufwand bringt eine feinere Abtastung nicht viel, da bei der Berechnung auf ganze Zahlen gerundet wird. Somit wird bei sehr kleinen Schritten, z.B. vier Schritte pro Grad, mehrmals pro Winkel auf dem gleichen Pixel gearbeitet. Dadurch werden die Ergebnisse der Polarabtastung nicht besser.

Werden die Schritte zu groß eingestellt, wird das Objekt nur sehr grob abgetastet. Bei der kleinsten Drehung besteht die Gefahr, dass gleiche Objekte nicht wiedererkannt werden, da zum Beispiel eine Ausbuchtung einfach übersprungen werden könnte.

Maskengröße der Filter:

Diese sollte nicht zu klein gewählt werden, denn sonst werden nicht genug Störungen eliminiert, um mit dem Bild weiter arbeiten zu können.

Wird jedoch eine zu große Maske gewählt, werden die Ergebnisse zwar besser, dafür steigt der Rechenaufwand aber exponentiell an.

Richtungsabhängigkeit der Kontur:

Mit dieser Erweiterung können komlexere Objekte gelernt und wiedererkannt werden. Leider ist es uns bei unseren Experimenten nicht gelungen, solche Objekte zu finden.

Flächenschwerpunkt:

Werden komplexere Obejkte verglichen, kann es vorkommen, dass bei einem der Objekte der Flächenschwerpunkt ausserhalb des Objektes und bei dem anderen innerhalb bzw. genau auf dem Rand berechnet wird. Je nach Objekt kann das zu Problemen führen.

Einsatbereiche des Verfahrens:

Dieses Verfahren kann, vorausgesetzt, es wird noch ein Bisschen, auf Geschwindigkeit bezogen, optimiert und evtl. in Hardware gegossen, ohne Probleme in der Industrie verwendet werden. Weitere Voraussetzungen dafür sind 2D-Objekte, die nicht zu komplex sein dürfen, z.B. Objekte mit zu vielen Löchern.

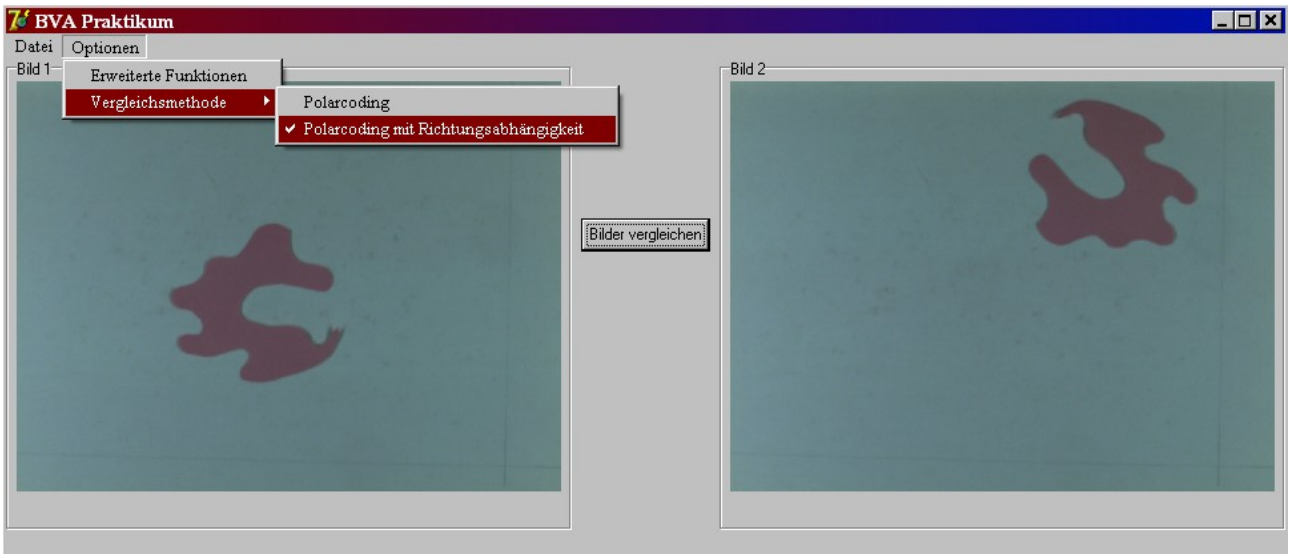
4. Exemplarischer Programmablauf

4.1. Beispiel für wiedererkanntes Objekt

Bild 1: amoebe_1.bmp

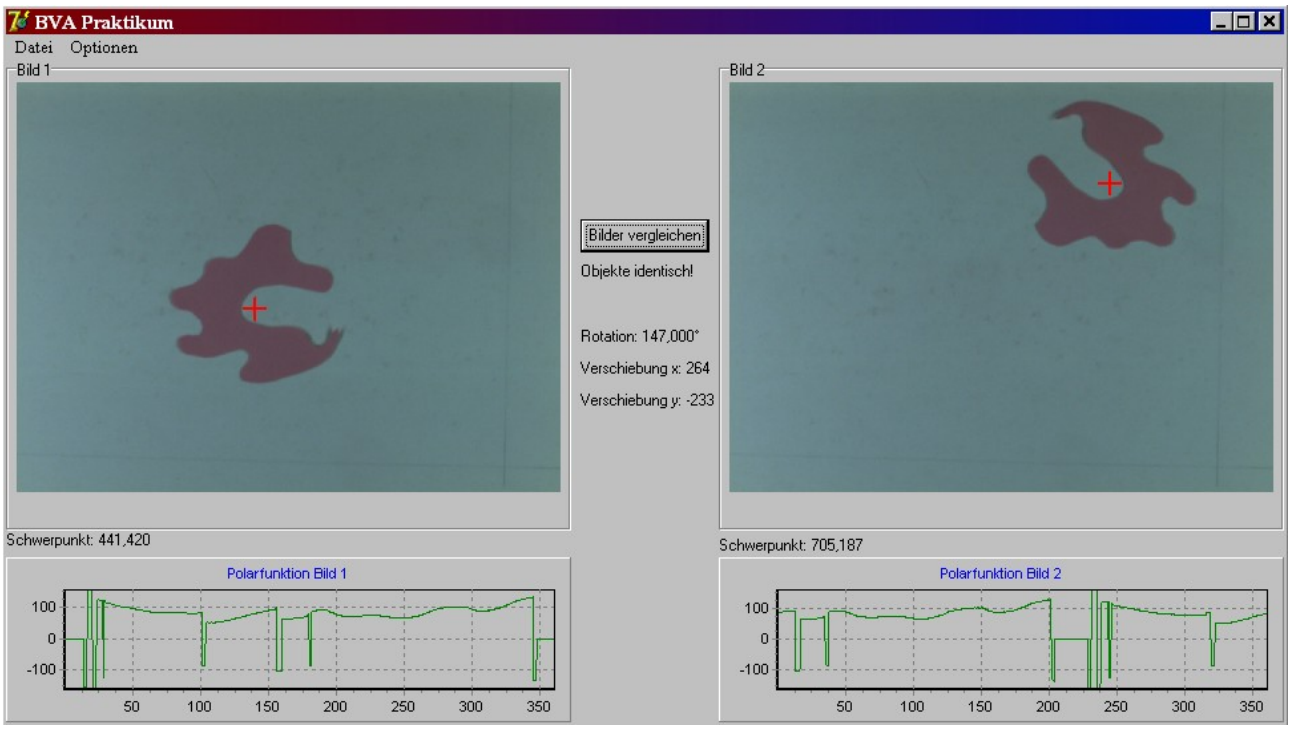
Bild 2 : amoebe_3.bmp

Vergleichsmethode: „Polarcoding mit Richtungsabhängigkeit“



Die Bilder amoebe_1.bmp und amoebe_3.bmp enthalten das selbe Pappobjekt. Demzufolge sollte beim Vergleich Übereinstimmung festgestellt und die Freiheitsgrade angezeigt werden.

Nach vollzogenem Vergleich:



Die Objekte wurden korrekt als identisch erkannt.

Die Schwerpunkte beider Objekte wurden eingezeichnet und ihre Koordinaten angezeigt, die Funktionen des Klassifizierungsverfahrens sind in den Diagrammen unter dem jeweiligen Bild zu sehen.

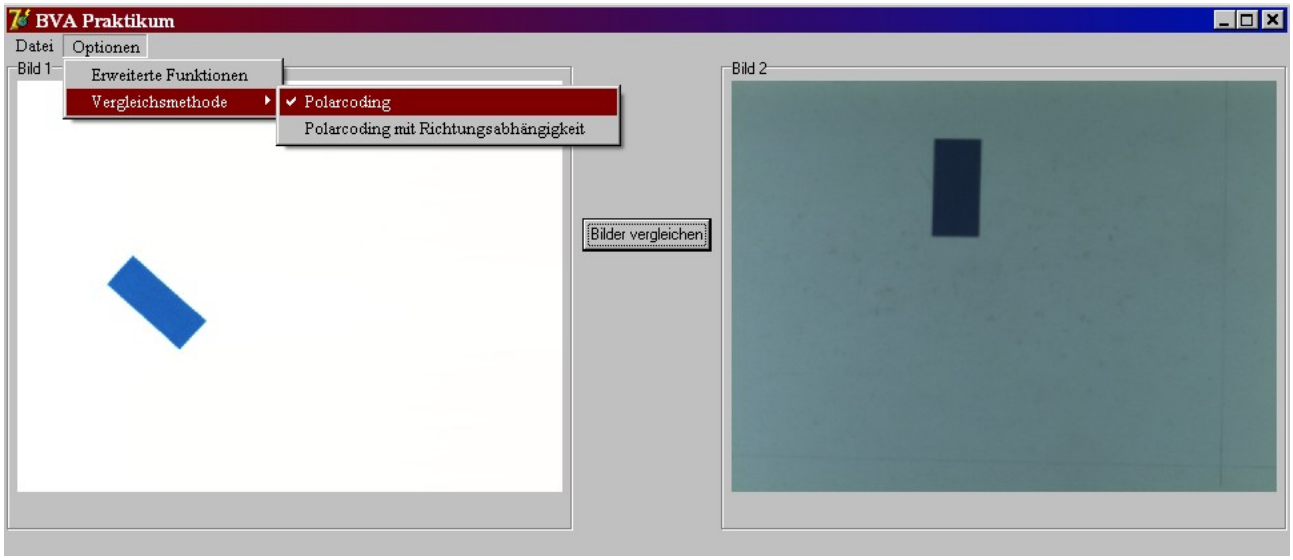
Die Rotation vom Objekt in Bild 1 gegenüber dem in Bild 2 beträgt 147.0°, das Objekt in Bild 2 ist um 264 Pixel in x-Richtung sowie -233 Pixel in y-Richtung verschoben.

4.2. Beispiel für nicht wiedererkanntes Objekt

Bild 1: rechteck_2.bmp

Bild 2: rechteck_anders_4.bmp

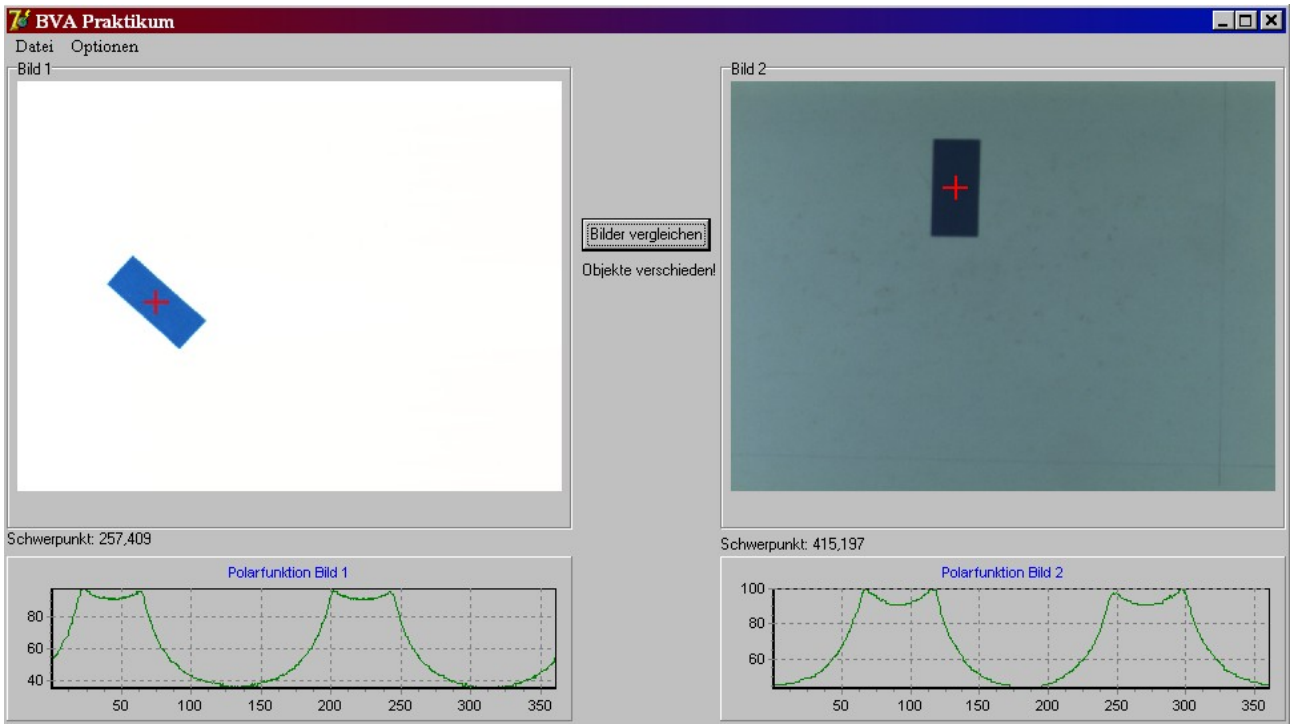
Vergleichsmethode: „Polarcoding"



Die Bilder rechteck_2.bmp und rechteck_anders_4.bmp enthalten ähnliche Pappobjekte, jeweils Rechtecke, die aber nicht identisch sind.

Demzufolge sollte beim Vergleich keine Übereinstimmung festgestellt werden.

Nach vollzogenem Vergleich:



Die Objekte wurden korrekt als verschieden erkannt.

Die Schwerpunkte beider Objekte wurden eingezeichnet und ihre Koordinaten angezeigt, die Funktionen des Klassifizierungsverfahrens sind in den Diagrammen unter dem jeweiligen Bild zu sehen.

4.3. Manuelle Vorverarbeitung und Vorteile des Median-Filters

Bild 1: smiley_4.bmp

Bild 2: nicht vorhanden

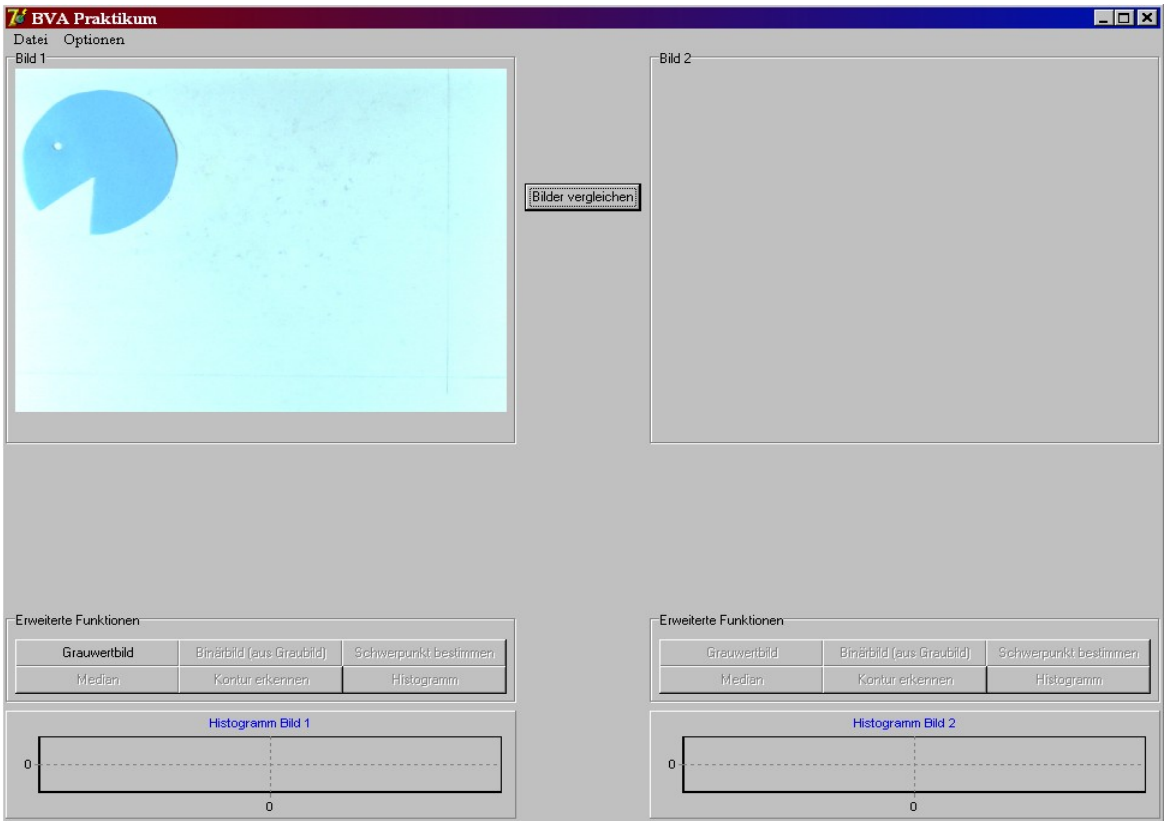
Zunächst werden, wie im Benutzerhandbuch beschrieben, die erweiterten Funktionen aktiviert.

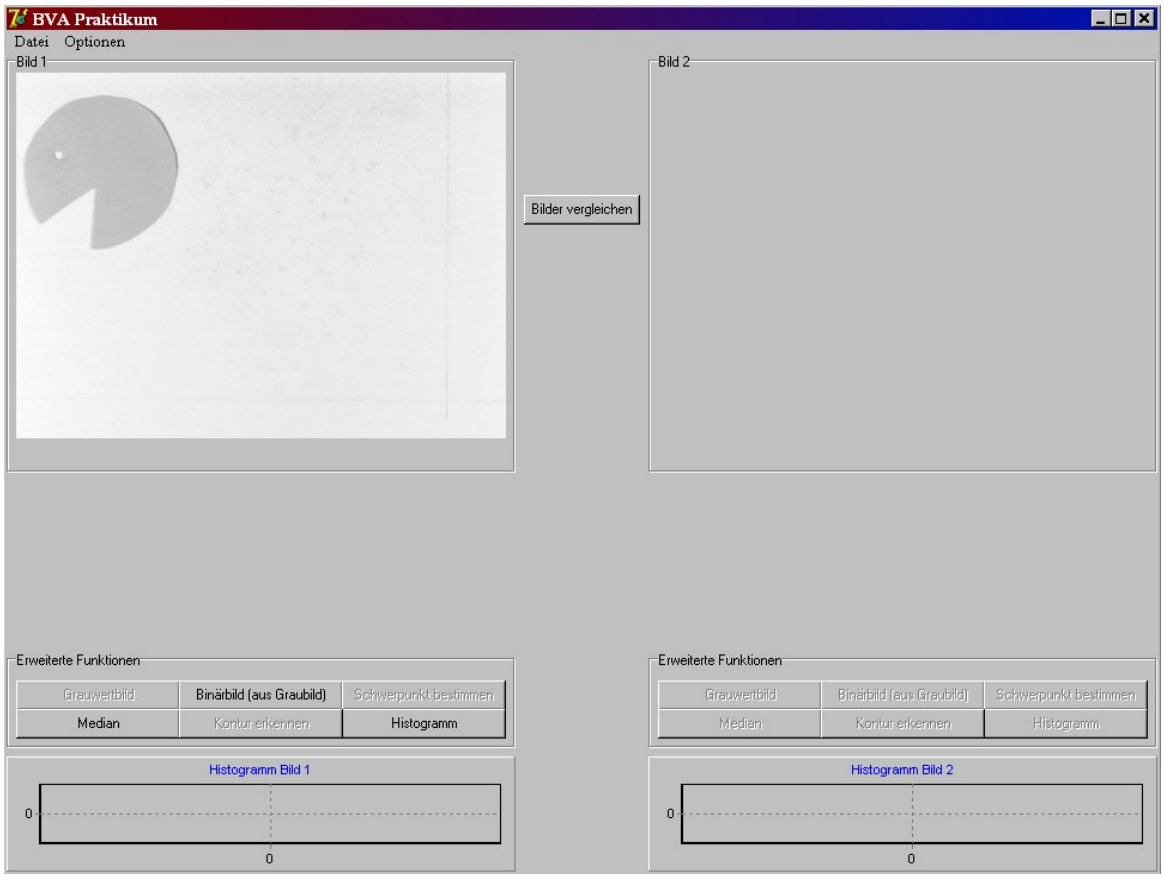
Anschliessend werden die Vorverarbeitungsschritte bis hin zum Binärbild durchgeführt.

Einmal geschieht dies mit dem Zwischenschritt der Median-Filterung, einmal ohne diesen.

Beim Durchlauf ohne Filterung sollten noch Störungen im Binärbild zu sehen sein, die durch den Median-Filter vermeidbar gewesen wären.

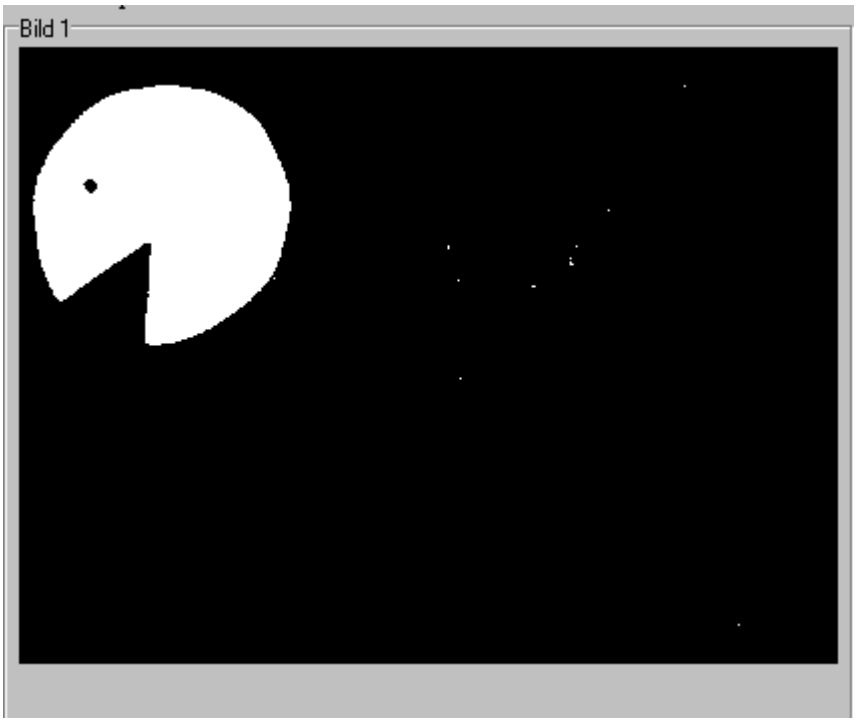
Ausgangsbild und Grauwertbild sehen in beiden Fällen wie folgt aus:


BVA Praktikum
Datei Optionen
Bild 1
Bild 2
Bilder vergleichen
Erweiterte Funktionen
Grauwertbild
Binärbild (aus Graubild)
Schwerpunkt bestimmen
Median
Kontur erkennen
Histogramm
Histogramm Bild 1
Histogramm Bild 2
0
0


BVA Praktikum
Datei Optionen
Bild 1
Bild 2
Bilder vergleichen
Erweiterte Funktionen
Grauwertbild
Binärbild (aus Graubild)
Schwerpunkt bestimmen
Median
Kontur erkennen
Histogramm
Histogramm Bild 1
Histogramm Bild 2
0
0

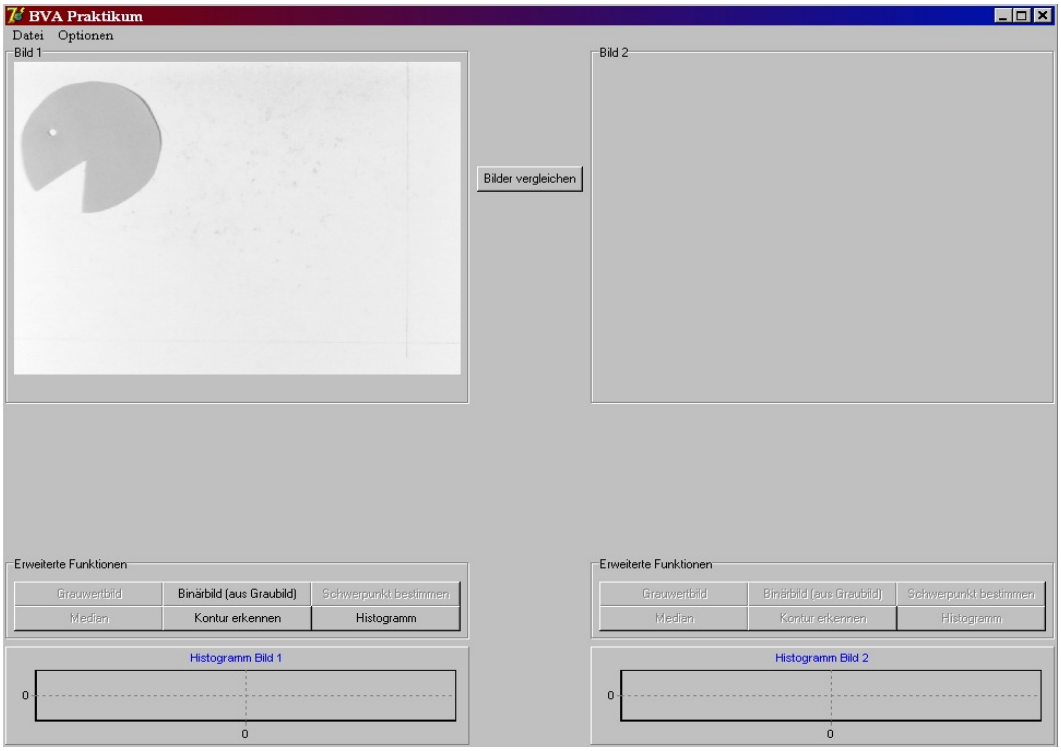
4.3.1. Durchlauf ohne Median-Filter

Im Binärbild ohne vorangehende Median-Filterung sind in der rechten Bildhälfte noch deutlich Störungen zu erkennen, die beim späteren „Polarcoding mit Richtungsabhängigkeit" zu Problemen führen könnten.



4.3.2. Durchlauf mit Median-Filter

Nach vollzogener Median-Filterung ist mit bloßem Auge noch kein Unterschied zum Grauwertbild zu erkennen:



Setzt man nun jedoch den Algorithmus zur Binärbilderzeugung auf obiges Bild an, erhält man folgendes, völlig störungsfreies Bild: