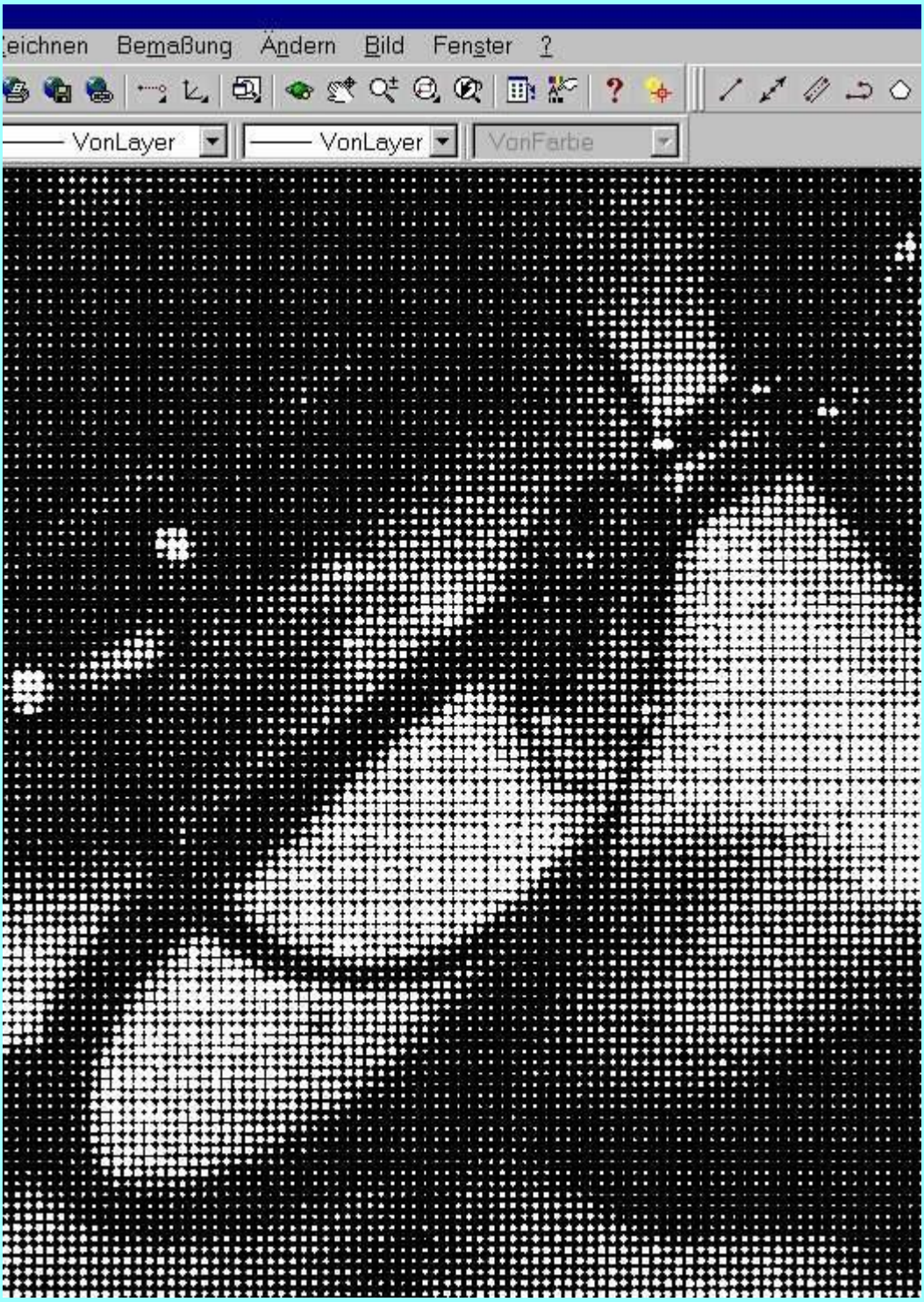
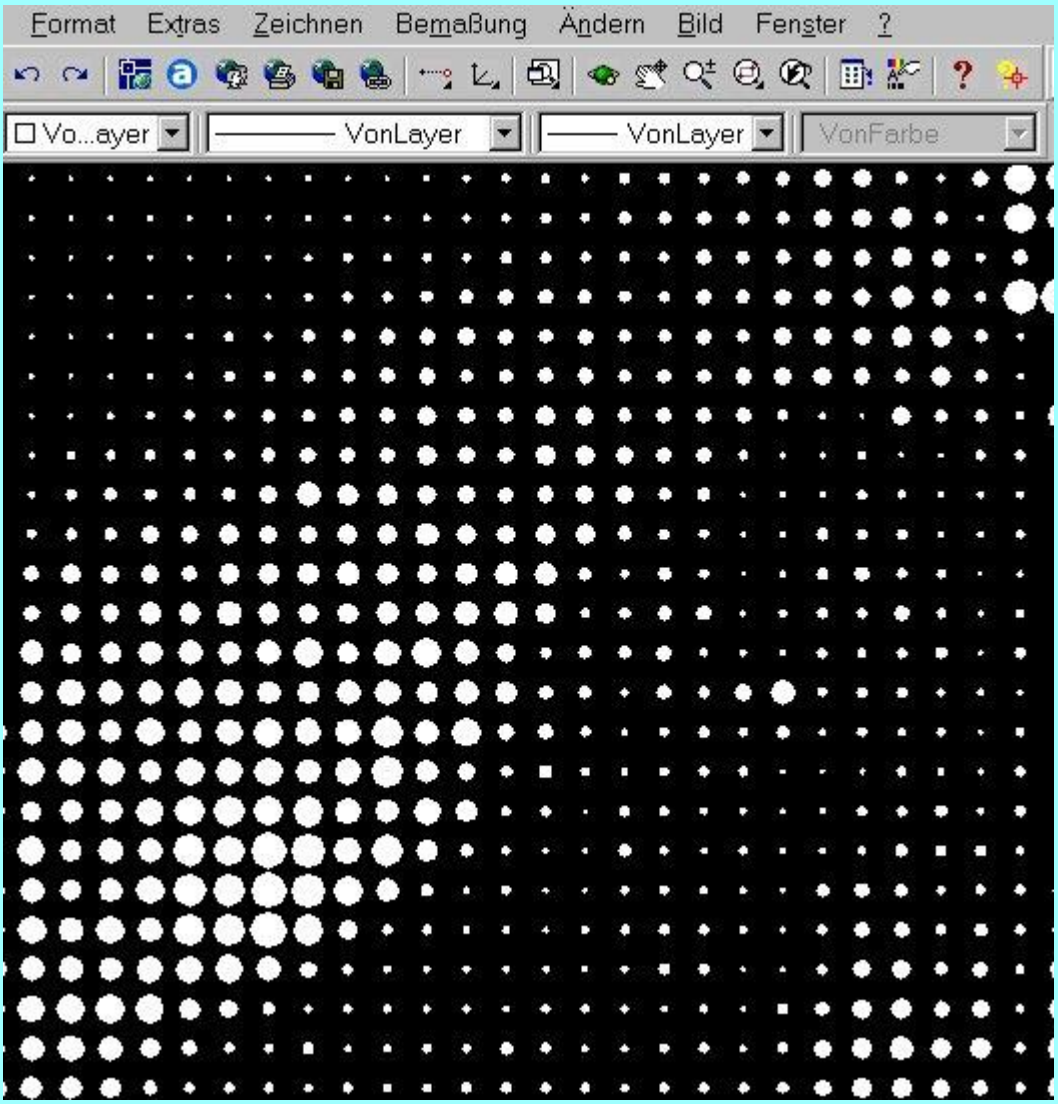
BILDER AUS BLECH

Die Foto-Vorlage



Ein Ausschnitt des Rasterbildes in AutoCAD (Screenshot 1:1)



Stärker gezoomt in AutoCAD (Screenshot)

Die Anregung zu diesem Projekt bekam ich kürzlich in einem Diskussionsforum für AutoCAD-Programmierer, und zwar von Stefan Michel. Der fragte dort sinngemäss: 'Wie kann ich die Pixel eines Schwarzweissfotos so nach AutoCAD übertragen, dass man ein gerastertes Bild erhält, welches man dann in ein Lochblech stanzen kann? Das Blech wird dann schwarz hinterlegt, und mit einem gewissen Betrachtungsabstand kann man dann sehr schön das Foto erkennen.'

Da ich in einem Lochblech-verarbeitenden Betrieb arbeite, hat mich diese Idee natürlich sofort fasziniert. Die Grundlagen sind überhaupt nicht schwierig. Die Löcher im Blech müssen nur in ihrem Durchmesser die entsprechenden Graustufen des Fotos wiedergeben. Das ist nichts anderes als das Rastern eines Fotos für den Zeitungsdruck oder was ein Laserdrucker macht, wenn er Bilder in Schwarzweiss druckt.

Bei einem hellen Blech mit schwarzer Hinterlegung muss also ein helles Pixel sehr klein im Durchmesser werden, nimmt man ein schwarz eloxiertes Alublech mit weissem Hintergrund, muss es eben umgekehrt sein.

Ein Bild, das Stefan schon gemacht hat.
Hier ist ein Ausschnitt zu sehen.

Nochmal das Bild von Stefan.
Es hat nur 8 verschieden Durchmesser (Farbstufen)!

Was braucht man, um das hier nachvollziehen zu können?

- Ein installiertes JRE (Java Runtime Environment). Das gibt es (kostenlos) bei http://web.archive.org/web/20070808041314/http://java.sun.com/. Damit lässt sich der Java-Teil des Programms, den du hier als fertige Class-Datei und hier als Quellcode herunterladen kannst, auf der DOS-Kommandozeile (bzw. in der shell bei Linux) ausführen. Du solltest aber auf jeden Fall vorher prüfen, ob es nicht schon installiert ist (nach java.exe suchen!). Wer Netscape oder Opera als Browser hat, wird sich diese Installation wahrscheinlich sparen können!
- AutoCAD ist nicht unbedingt erforderlich. Ich werde zwar auch ein kleines Lisp-Programm, das du hier herunterladen kannst, verwenden, aber ich werde am Ende auch einen Weg für diejenigen zeigen, die kein AutoCAD zur Verfügung haben.
- Schliesslich braucht man noch das Material selbst, um so einen 'Foto-Print' zu realisieren. Das überlasse ich euch. Das einfachste ist sicher, so ein Bild einfach auf

einem Laserdrucker auszugeben. Wer es wirklich in Blech haben will, wird ein paar Mark investieren müssen und das Blech entweder stanzen oder aber mit Laser oder Wasserstahl schneiden lassen. Wenn das Blech nicht allzu groß sein soll (A3-Format), dann kommt auch ein galvanotechnisches Verfahren in Frage.

Und nun zu den Programm-Grundlagen: Wir müssen zunächst aus dem Bild die Pixel-Informationen herauslesen. Direkt geht das natürlich kaum, da GIF- und JPG-Dateien komprimiert sind. Aber Java stellt uns die notwendigen Werkzeuge zur Verfügung. Um die Bälle hier flach zu halten, habe ich den primitivsten Weg gewählt: Ein kleines Java-Programm, das die Informationen in der DOS-Kommandozeile ausgibt. Diese Ausgabe kann man in eine Datei umlenken und speichern.

Jedes Bildpixel besteht aus den Farbanteilen für Rot, Grün und Blau. Bei Schwarzweissbildern sind diese Anteile gleich, es entstehen Graustufen von 0 (schwarz) bis 255 (weiss). Das Programm wertet nur den Rotanteil aus (natürlich könnte es genauso gut Grün oder Blau sein, das spielt keine Rolle).

Die Ausgabe sieht so aus: In der ersten Zeile steht die Breite des Bildes, in der zweiten Zeile die Höhe. Ab der dritten Zeile folgt jeweils eine Zahl zwischen 0 und 255 für ein Pixel. Wenn die Auflösung also beispielsweise 300 x 200 ist, hat die Ausgabe 60002 Zeilen. Aber keine Angst, die Datei ist trotzdem in wenigen Sekunden geschrieben!

Die Ausgabedatei kann nun im Prinzip von jedem anderen Programm gelesen und verarbeitet werden - oder auch von einem Heimwerker, der mit 256 Bohrern 60000 Löcher in eine mit Millimeterpapier beklebte Sperrholzplatte bohren möchte. Ich mach's mir jedenfalls erstmal einfacher: Ich nehme AutoCAD!

Die kleine Lisproutine liest die Ausgabedatei ein und zeichnet in AutoCAD die Kreise mit entsprechenden Durchmessern. Diese Datei kann man dann an eine Firma schicken, die das Lochblech herstellt. Das Programm beginnt bei den Koordinaten 0,0 und zeichnet Kreise mit Durchmessern zwischen 0,01 und 1. Nach jedem Loch springt es um 1,1 weiter, und wenn eine Zeile voll ist, springt es an den nächsten Zeilenanfang.

Das Zeichnen in AutoCAD dauert erheblich länger als der Java-Teil. Du solltest auch möglichst erstmal Tests mit Auflösungen von 75x50 oder 150x100 machen, bevor du ein Bild deiner brandneuen 6-Megapixel-Digicam in ein armes AutoCAD jagst. Tröste dich damit, dass so große Bleche sowieso nicht hergestellt werden!

Und nun zur Bedienungsanleitung: Das Java-Programm muss eine Applikation sein, da ein Browser-Applet aus Sicherheitsgründen nicht auf die Festplatte deines Rechners schreiben darf. Nach der Installation des JRE (siehe oben) musst du folgendes machen: Das Bild (ich nenne es mal 'test.gif') und die Datei 'Lochblech.class' in irgendein neues Verzeichnis (ich nenne es mal 'lochblech') kopieren. Dann eine DOS-Shell aufrufen und in das Verzeichnis wechseln. Folgende Eingabe abschicken:

C:\lochblech>c:\Programme\JRE1.3.1\bin\java Lochblech test.gif > test.txt

Die Pfade musst du natürlich an die gegebenen Verhältnisse anpassen! Gross- und Kleinschreibung ist im Prinzip egal, nur bei 'Lochblech' nicht. Das Bild kann eine GIF- oder JPG-Datei sein. Dann startest du AutoCAD mit einer neuen Zeichnung. Das Lisp-Programm lädst du mit

(load "c:/lochblech/lochblech.lsp")

Starten kannst du das Programm in AutoCAD nun mit der Eingabe von

(lochblech "c:/lochblech/test.txt")

Und nun viel Spass beim Zuschauen, das kann ein paar Minuten dauern, bis die Zeichnung fertig ist! In der Lisproutine ist die Zeile, die die Kreise tatsächlich zeichnen soll, derzeit auskommentiert und durch eine Zeile ersetzt, die gefüllte Scheiben erzeugt. Das sieht auf dem Bildschirm besser aus und ist auch zum Plotten besser. Zum Lasern usw. sollte man aber wieder die Kreise einsetzen!

Ach ja, wer nicht nur das JRE (Java Runtime Environment) zur Verfügung, sondern sich das JDK (Java Development Kit) installiert hat, der kann an dem Java-Programm auch Veränderungen vornehmen und den Sourcecode dann neu kompilieren! Das JRE kann nur die fertige Class-Datei ausführen.

Hier nun die Sourcecodes der Programme (nicht lang!), damit jeder, der gerne möchte, damit spielen kann:

Der Java-Teil:

```
import java.lang.*;
import java.awt.*;
import java.io.*;
import java.awt.image.*;

public class Lochblech extends Frame{
  private static Lochblech application;

  // Application starten
  public static void main(String[] args){
    application = new Lochblech(args[0]);
    // ...und beenden
    System.exit(0);
  }

  // Konstruktor (macht alles)
  public Lochblech(String imageName){
    // Farbkanäle
    int red,green,blue;
    // Bild
    Image image = Toolkit.getDefaultToolkit().getImage(imageName);
    // Laden des Bildes überwachen
```

```
    MediaTracker mt = new MediaTracker(this);
    mt.addImage(image, 0);
    // Warten, bis Bild vollständig geladen
    try{
      mt.waitForID(0);
    }catch(Exception e){
      // Wenn was schiefgeht
      System.exit(0);
    }

    // Breite und Höhe ermitteln
    int w = image.getWidth(this);
    int h = image.getHeight(this);
    // Speicherplatz Breite x Höhe reservieren
    int[] pixels = new int[w * h];
    // Die Pixel auslesen
    try{
      PixelGrabber pg = new PixelGrabber(image,
                              0, 0, w, h, pixels, 0, w);
      pg.grabPixels();
    }catch(Exception e){

    }

    // Breite und Höhe ausgeben
    System.out.println(w);
    System.out.println(h);
    // Den Rotanteil der Pixel ausgeben
    for(int i = 0; i < w * h; i++){
      red = (pixels[i] & 0x00ff0000) >> 16;
      //green = (pixels[i] & 0x0000ff00) >> 8;
      //blue = (pixels[i] & 0x000000ff);

      System.out.println(red);
    }
  }
}
```

Und der Lisp-Teil:

```
(defun lochblech(filename / fh b w h p r row col)
  (setvar "cmdecho" 0)
  (setvar "osmode" 0)
  (setq fh(open filename "r"))
  (setq w(atoi(read-line fh)))
  (setq h(atoi(read-line fh)))
  (setq row 0)
  (setq p '(0 0))
  (setq r (/ 1 255.0))
  (repeat h
    (setq col 0)
    (repeat w
      (setq p(list(* 1.1 col)(* 1.1 row)))
      (setq b(1+(atoi(read-line fh))))
      ;(command "_circle" p (* b r))
      (command "_donut" 0 (* b r) p "")
      (setq col(1+ col))
    )
    (setq row(1- row))
  )
  (close fh)
)
```

Noch eine Ergänzung, es haben mich einige Mails erreicht mit der Frage, wie man die Kreisdurchmesser an vorhandene Werkzeuge anpassen kann - d.h. nicht mehr alle möglichen Durchmesser sollen vorkommen können, sondern es soll auf einige verschiedene Durchmesser reduziert werden, für die Stanzwerkzeuge vorhanden sind.

Natürlich gibt es da verschiedene Wege - auf die naheliegende Möglichkeit, das Bild einfach als 8- oder 16-Farben-GIF abzuspeichern, ist wohl niemand gekommen. Aber natürlich kann es auch sein, dass jemand nicht 8 oder 16 Werkzeuge zur Verfügung hat, sondern vielleicht ausgerechnet 11! Ein 11-Farben-GIF gibt es leider nicht...

Hier noch mal eine Variante des Lisp-Teils, der (beispielhaft) die ganze Sache für 11 verschiedene Werkzeugdurchmesser erzeugt. Zum Abändern reicht es, neue Werkzeuge einfach in die Liste einzutragen oder daraus zu entfernen, es muss sonst nichts am Programm geändert werden.

```
(defun lochblech(filename / fh b w h p r row col
                            maximum werkzeuge abstufen)
  ; Lokale Funktion zum Abstufen der Radien
  (defun abstufen(zahl werkzeuge / )
    (terpri)
    (nth(fix(/ zahl(/ 1.0(1-(length werkzeuge))))))werkzeuge)
  )
  ; **************
  (setvar "cmdecho" 0)
  (setvar "osmode" 0)

  (setq werkzeuge'(1 1.5 2 2.5 3.5 4.5 6 7.5 9 11 13))
    ; Hier die vorhandenen Werkzeug-Radien eintragen!!!!

  (setq maximum(apply'max werkzeuge))
  (setq fh(open filename "r"))
  (setq w(atoi(read-line fh)))
  (setq h(atoi(read-line fh)))
  (setq row 0)
  (setq p '(0 0))
  (setq r(/ 1 255.0))
  (repeat h
    (setq col 0)
    (repeat w
      (setq p(list(* 2.2 maximum col)(* 2.2 maximum row)))
      (setq b(1+(atoi(read-line fh))))
      ;(command "_circle" p (abstufen(* b r)werkzeuge))
      (command "_donut" 0 (abstufen(* b r)werkzeuge) p "")
      (setq col(1+ col))
    )
    (setq row(1- row))
  )
  (close fh)
)
```

Und noch ein kleiner Tipp: Wer mit dem Lisp-Teil überhaupt nicht klarkommt, der kann ja mal einen Blick in mein Autolisp-Tutorial werfen - da werden Sie geholfen!

Du hast kein AutoCAD, willst aber dein Sparschwein opfern und dir so ein Blech stanzen lassen? Ich kann dir folgende Vorschläge machen:

- Ändere das Java-Programm so ab, dass nicht die Farbnummer allein ausgegeben wird, sondern (schon umgerechnet) der Kreisradius sowie die dazugehörenden Koordinaten. Die so ausgegebene Textdatei wird ein Stanzbetrieb statt einer AutoCAD-Zeichnung auch akzeptieren.
- Mit ein wenig Programmierkenntnissen lässt sich der Lisp-Teil auch in andere Sprachen übertragen, er ist ja wirklich nicht kompliziert! Wenn's z.B. um's Drucken geht: In Corel Draw! oder Photopaint gibt's eine Basic-Schnittstelle, mit der sich mit wenigen Zeilen ein Programm für das Erstellen einer Druckvorlage realisieren liesse! Betriebe, die Folien mit Schneidplottern bearbeiten, werden wahrscheinlich mit Corel Draw! kein Problem haben.
- Und dann gibt es noch eine Variante des Java-Programms, die aus dem GIF oder JPEG direkt eine DXF-Datei erzeugt. Diese Datei kann weitergegeben werden, sie kann aber auch mit AutoCAD (alle Versionen), Corel Draw! usw. geöffnet werden. Der enthaltene DXF-Printer ist ein Beispiel, wie das Schreiben von DXF-Dateien in Java realisiert werden kann. Das Paket mit class-Dateien, Sourcecode und Anleitung kannst du hier gezippt herunterladen. DXF ist das **D**ata E**x**change **F**ormat von AutoCAD, das von fast allen Programmen verstanden wird.