1. Initialisierung

Als erstes werden ein paar Pakete geladen: Um Arrays zu manipulieren benötigen wir das Packet `numpy` (abgekürzt als `np`) und um Bilder darzustellen `matplotlib.pyplot` (abgekürzt als `plt`).

```
import numpy as np
import matplotlib.pyplot as plt
```

2. Daten einlesen

Der Befehl `open` öffnet die Datei `leber.vol` und liest den Inhalt Byte für Byte aus. Die ersten 291 Bytes können wir überspringen und gehen damit gleich an die interessante Stelle.

Die verschiedenen Abschnitte der binären Datei sind durch Tags gekennzeichnet und der 3D Datensatz beginnt mit dem (hexadezimalen) Schlüsselwort `0x0050 (hexadezimal)` (2 Byte). Wir interpretieren diesen Wert als unsigned short mit Hilfe der Funktion `frombuffer` und das Ergebnis sollte `0x0050 (hexadezimal) = 80 (unsigned short)` sein.

Darauf folgt eine Angabe darüber wie groß der folgende Datensatz ist (unsigned long = 4 Byte). In unserem Fall sind es 83344800 Bytes.

```
f = open("leberExport.vol1.vol","r") # Öffne die Datei
f.read(291) # Überspringe die ersten 291 Bytes

data       = f.read(2) # Schlüsselwort / Tag
tag        = np.frombuffer(data, np.uint8)[0] # interpretiere als unsigned short (2 Byte)

data       = f.read(4) # Größe des Datensatzes
tag_length = np.frombuffer(data, np.uint32)[0] # interpretiere als unsigned long (4 Byte)

print "Sind wir an der richtigen Stelle?", tag == int(0x0050) # Tag: 80 = int(0x0050)
print "Die Größe des Datensatzes in Bytes:", tag_length
```

```
Sind wir an der richtigen Stelle? True
Die Größe des Datensatzes in Bytes: 83344800
```

Nachdem wir die Daten ausgelesen haben. Speichern wir den Inhalt in einem (numpy) Array ab. Die volumetrischen Daten sind als unsigned int Werte gespeichert, die einer Graustufe entsprechen. Außerdem sind die x,y,z Koordinaten als koninuierliche Zahlenfolge gespeichert und um den Datensatz intuitiver auslesen zu können, werden wir ihn mit der Methode `reshape` in die Form eines Quaders bringen. Die Auflösung / Dimension des Quader ist:

- z-Auflösung: 616
- y-Auflösung: 330
- x-Auflösung: 410

Das heißt, die Daten liegen in der folgenden Form vor:

```
(z[1], y[1], x[1]),     (z[1], y[1], x[2]),     ... , (z[1], y[1], x[410])
.
.
.
(z[1], y[330], x[1]),   (z[1], y[330], x[2]),   ... , (z[1], y[330], x[410])
.
.
.
(z[616], y[330], x[1]), (z[616], y[330], x[2]), ... , (z[616], y[330], x[410])
```

```
data   = f.read(tag_length) # 3D Datensatz auslesen
VOLUME = np.frombuffer(data, np.uint8).reshape(616,330,410)#interpretiere als unsigned short
print "Der Datensatz ist als Quader gespeichert. Die Dimensionen sind:",VOLUME.shape
```

```
Der Datensatz ist als Quader gespeichert. Die Dimensionen sind: (616, 330, 410)
```

Ein Schnitt durch diesen Datensatz entspricht jetzt dem Bild eines Sono-Echos. Der einfachste Schnitt ist entlang einer Dimension, hier `z = 300`. Der Doppelpunkt bedeutet (ähnlich wie bei Fortran), dass alle Werte entlang dieser Dimension betrachtet werden. Wir könnten die Zeile auch etwas ausführlicher und C-näher schreiben:

```
z = 300
y_range = range(0,330,1) # 0,1,2,...,329
x_range = range(0,410,1) # 0,1,2,...,409

image = np.zeros(( 330  , 410 )) # 2D Schnittebene
for y in y_range:
    for x in x_range:
        image[y,x] = DATA[z,y,x]
```

Um den 2D Querschnitt zu plotten nutzen wir die Funktion `pcolormesh`. An den Parameter `cmap` übergeben wir die Anweisung `plt.cm.Greys_r`, dass die Werte als Graustufen interpretiert werden und speichern schließlich das Ergebnis in der Datei “querschnitt.png” ab.

```
image = VOLUME[300,:,:] # Querschnitt durch die z-Achse bei z = 300

plt.pcolormesh(image,cmap=plt.cm.Greys_r) #plotten
plt.savefig("querschnitt.png") #Plot speichern
```

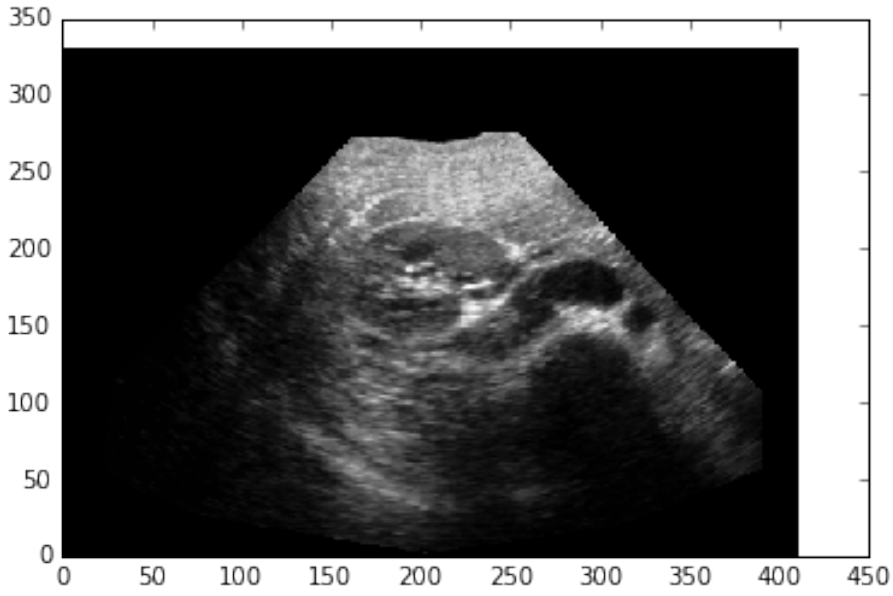

Figure 1: png

Die einfachste Animation bestünde nun darin, nacheinander alle z-Werte durchzugehen. Beliebige Drehungen lassen sich dadurch erreichen, dass man die Koordinaten der Schnittebene transformiert: https://de.wikipedia.org/wiki/Koordinatentransformation. Auf diese Art übertragen die Geräte der Firma Sonofit, von denen wir die Daten haben, die Position des Schallkopfs auf die entsprechende Schnittebene.