

平成28年度 卒業研究論文

疑似ラインセンサによる
SP盤の非接触再生

Noncontact sound reproduction from SP discs
by means of a pseudoline sensor

2017年2月

北海学園大学工学部 電子情報工学科
魚住研究室

4513213
クーン トビアス

目 次

第 1 章 はじめに	1
第 2 章 実験の概要	2
2.1 実験装置	2
2.1.1 機器の接続	3
2.1.2 CCD カメラ	3
2.1.3 パルスステージ	4
2.2 音溝形状と光照射	4
第 3 章 部分読み出し法	7
3.1 撮像段階での問題	7
3.1.1 重複部分問題	7
3.1.2 湾曲問題	10
3.1.3 うねり問題	12
3.2 部分読み出し法による画像合成の利点・欠点	13
第 4 章 画像データ生成	14
4.1 SPRecAnalyzer	14
4.1.1 GPIB Interface タブ	15
4.1.2 Camera タブ	16
4.1.3 物理的距離検出	17
4.1.4 β 角算出	17
4.1.5 キャリブレーション機能	19
4.1.6 自動撮像機能	19
第 5 章 画像処理	20
5.1 SpliceBot	20
5.1.1 処理内容	21
5.2 SigExBot	22
5.2.1 処理内容	22
5.2.2 信号修正機能	24
5.2.3 GapFilling	25
5.2.4 TryToRecover	27
5.3 Needle	28

5.3.1 处理内容	29
第 6 章 結論	31
謝辞	33
参考文献	34
付 錄 A SPRecAnalyzer の使用法	35
付 錄 B 物理的距離検出プログラム	36
B.1 PixelProcessing.m	36
B.2 ObjectExtraction.m	37
付 錄 C 画像処理プログラム	40
C.1 SpliceBot.m	40
C.2 SigExBot.m	44
C.3 Needle.m	54
C.4 fftf.m	57

第1章 はじめに

1888年から1950年代後半まで生産されたSPレコードは今現在も数多く残っている。本来は蓄音機で再生されるように設計されたとはいえ、針を用いた再生法では記録媒体が痛み、再生する度に音質の劣化を伴うことから、非接触再生法がこの貴重な文化財を傷を付けずにデジタル化できる技術として注目を浴びるようになった[1, 2]。非接触法はレコードに損害を与えずに再生できるばかりでなく、割れやひびが入ったレコードも再生可能であるというメリットを持っている。

本研究では、SPレコード表面を顕微鏡付きカメラで撮像することにより、全レコードの音溝データを収集する。その際の1フレームの画像の幅が僅か32ピクセルであることがこの研究で用いる「部分読み出し法」の特徴である。本論文で説明するように、幅が非常に小さい画像データを取得することにより、画像処理段階で生じる幾何学的な問題を避けることができ、結果として画像処理をより高速で行うことができる。

幅が非常に小さい画像でレコード全面に渡る音溝データを記録しなければならないことから、本研究では巨大なデータ量を管理し、一枚のレコードの完全デジタル化を自動的に行うソフトウェア「SPRecAnalyzer」を開発した。また、SPRecAnalyzerが生成したデータを対象とする一連の画像処理アルゴリズムを提案した。この画像処理アルゴリズムを実行することにより、PCで再生できるwavファイルが得られ、レコードの音データをいつでも再現することが可能となる。

第2章 実験の概要

2.1 実験装置

SP レコードの撮像に用いた実験装置の模式図を図 2.1 に示す。基本的な構成は、昨年度までの卒業研究の実験装置と同様である [3]。図中の D で示すレコードは、2 つのパルスステージの上に設置されている。このうち、パルスモータ E で平行移動用パルスステージはレコードをカメラの位置に対して水平面内で平行に（図 2.1 で言うと左右に）動かす。この平行移動をパルスモータ F による回転運動と組み合わせることにより、SP レコード上面の任意の領域を固定されている顕微鏡 B の下に持つことができる。

- A CCD カメラ
- B 顕微鏡
- C 照射装置
- D SP レコード
- E パルスモータ、平行移動用
- F パルスモータ、回転用

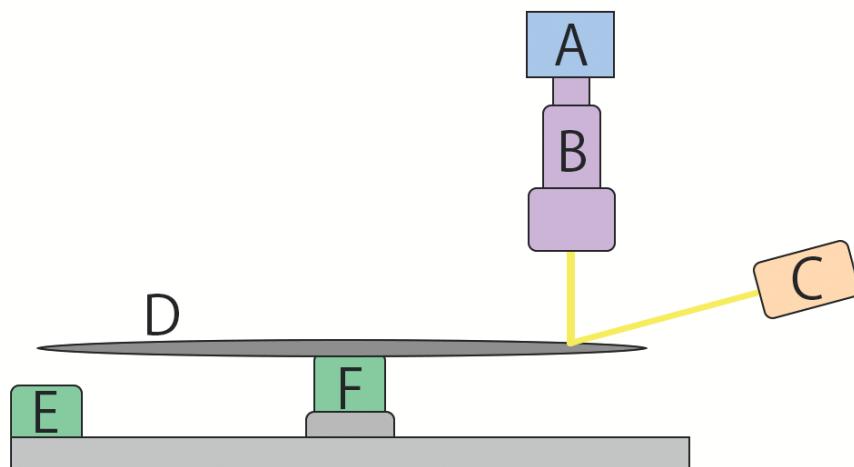


図 2.1: 実験装置の構成

2.1.1 機器の接続

実験装置を構成する各機器がどのようにしてパーソナルコンピュータ（PC）と繋がっているかを図 2.2 に示す。GigE バスは PC の NIC (Network Interface Card) と接続される。ここで GPIB は「General Purpose Interface Bus」の略であり、その仕様は Hewlett-Packard 社が定め、後に IEEE の標準バスに採用された「HP-IB」、「IEEE-488」とも知られている。一方、GigE は「Gigabit Ethernet」の略であり、イーサネットを通して通信速度が 1Gbit 毎秒であることに由来する。

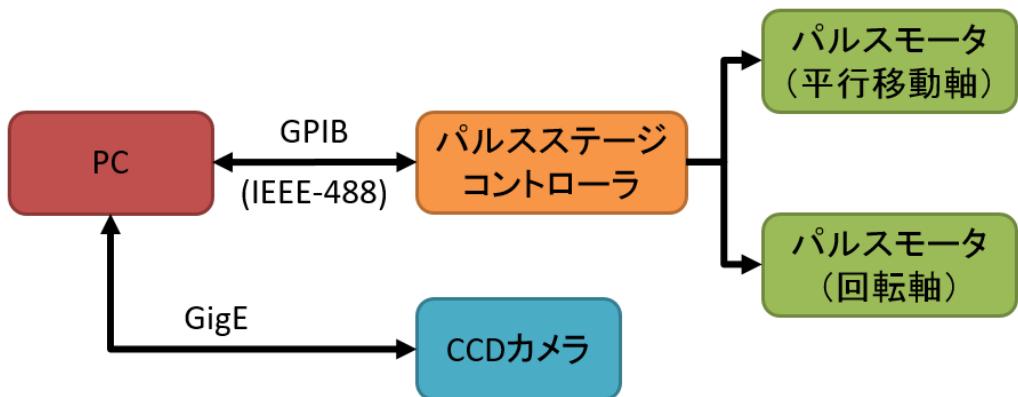


図 2.2: 実験装置の接続

2.1.2 CCD カメラ

本研究で使用するカメラは JAI 社の「BB-500GE」である。カメラの最大解像度は $2456(h) \times 2058(v)$ であるが、Variable Partial Scan 機能を利用することにより、実際にはより幅の狭い領域のデータを PC に送ることもできる。表 2.1 に本研究で用いたカメラのパラメータの設定を示す。

画像データを生成する段階で、SP レコードを一定の回転速度で回転させながら順次撮像を行うため、ExposureTimeRaw パラメータ（無単位）を実験的にぶれが生じないように設定した。表 2.1 に示すパラメータは全て PC から設定しており、その設定を行うプログラムを C# により作成した。詳細な説明は第 4 章を参照されたい。SoftwareTrigger を実行して画像一枚を取得するためのコードを図 2.3 に示す。

```

// We need to "pulse" the Software Trigger feature in order to trigger the camera!
myCamera.GetNode("SoftwareTrigger0").Value = 0;
myCamera.GetNode("SoftwareTrigger0").Value = 1;
myCamera.GetNode("SoftwareTrigger0").Value = 0;
  
```

図 2.3: SoftwareTrigger を実行させるための C# プログラムのコード

表 2.1: JAI BB-500GE のパラメータの設定

ROI 範囲設定	2456 (h) x 32 (v)
GainRaw	550
ExposureTimeRaw	222
AcquisitionMode	Continuous
ExposureMode	EdgePreSelect
PartialScan	Variable Partial Scan
VariablePartialScanStartLine	1013
VariablePartialScanNumOfLines	32
TriggerSelector	FrameStart
TriggerMode	On
TriggerSource	Software
LineSelector	CameraTrigger0
LineSource	SoftwareTrigger0
LineInverter	ActiveHight

2.1.3 パルスステージ

本研究で用いた回転ステージはシグマ光機株式会社の SGSP-60YAW-W-0B であり、平行移動ステージには同社の SGSP26-200(Z) を用いた。この 2 つのステージを制御するコントローラには同社の SHOT-102 を用いた。パルスステージの仕様を表 2.2 に示す。

表 2.2: パルスステージの仕様

degrees per pulse (回転運動)	0.0025
mm per pulse (平行移動)	0.002

パルスモータは位置制御に適しており、本研究でもその性質を活用している。研究の初期段階で、回転ステージ上に置かれたレコードが滑ることが判明した。この滑りが原因で、正確な位置制御が困難であった。滑りの発生を阻止するため、回転ステージの上に直接置かれていたアクリル板と回転ステージの間に円形のゴムマットを置き、さらにアクリル板とレコードの間にもゴムマットを設置した。

2.2 音溝形状と光照射

顕微鏡を通して撮像された光学像から SP レコードの音溝に記録されたデータを復元するのが本研究の目標であるから、撮像されるデータに音溝の形状が何らかの

方法で読み取られなければならない。これを実現するため、光をレコード面に対して斜めに照射して、音溝の壁の一部が明るく写るようにするのが有力な方法であることが過去の研究でわかっている。図2.4に斜め照射による光の反射の様子を模式的に表す。

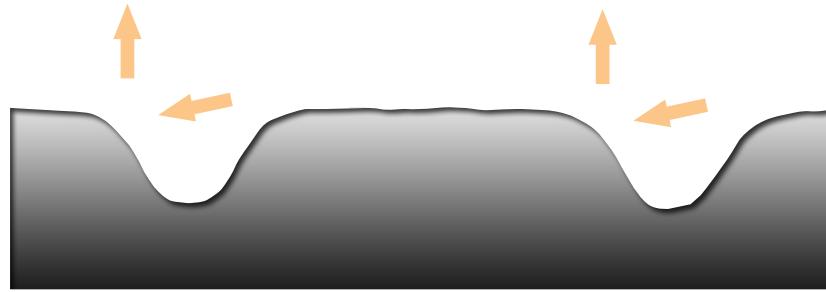


図2.4: 斜め光照射における反射の仕方

図2.4に示すように、照射角度を調節することにより、SP盤に対して垂直方向に反射する光のうち、音溝の壁部分からの寄与が最も多くなるようになることが可能である。その場合、音溝以外の平面部分で反射する光は垂直方向には反射せず、顕微鏡の対物レンズにはほとんど入射しない。この光照射によりカメラと顕微鏡を用いて撮像した画像を図2.5に示す。

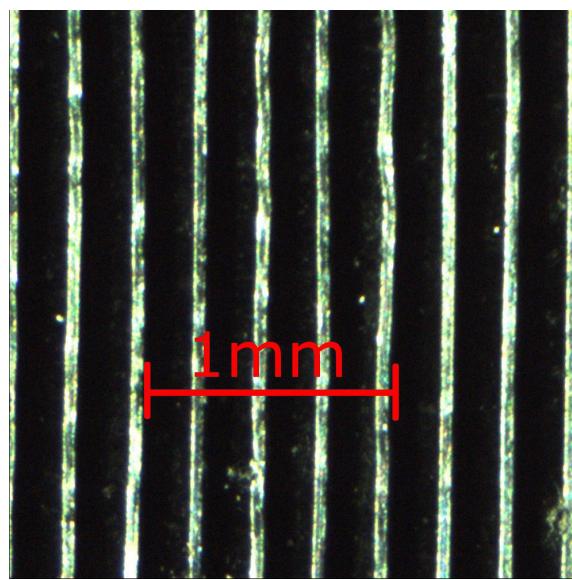


図2.5: 反射光をカメラで撮像して得られた画像データ

本研究の初期段階では照射装置に接続する照射ヘッドについて比較実験を行った。昨年度まで用いてきた円形の照射ヘッドで得られた撮像データには、音溝が強く振動している部分において画像処理段階で処理が困難になると考えられる黒い影が生

じた。これに対し、横長の矩形ヘッドを用いた場合には、このような問題は確認されなかった。そのため、円形照射ヘッドを短形照射ヘッドへと変更した。

第3章 部分読み出し法

カメラが撮影した画像データの一部のみを用いる部分読み出し法には利点もあれば欠点もあり、撮像段階で直面する問題とその解決の説明を通じてそれらについて考察する。

3.1 撮像段階での問題

画像処理技術を用いて音溝の画像から音データを抽出するためには、画像上の音溝データが連続的であり、かつ余分な重複がないものでなければならぬ。そのため、矩形の画像を撮影するカメラでその下を回転する円盤の表面を撮像する際にどのような問題が生じるかについて、解析的に考察する。

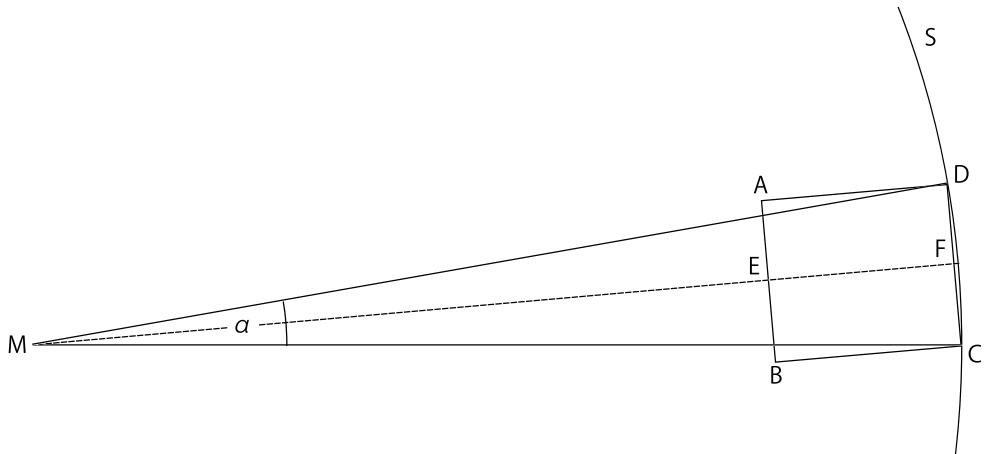


図 3.1: 撮像の様子を幾何学的に表した図

図 3.1において、長方形 ABCD をカメラの撮像範囲、S を SP レコードの外側、M を SP レコードの中心、 α を次の画像を取るための回転角度とする。

3.1.1 重複部分問題

図 3.2 の長方形 Q_1 , Q_2 , Q_3 が順番に撮像された画像範囲であるとする。このとき、それぞれの撮像範囲に重複部分があることがわかる。撮像範囲が長方形である

ことを前提とすると、画像をどのように撮像しても重複部分、または情報損失が生じる。音溝から音を抽出する画像処理段階で、情報損失があってはならない。また、重複部分も画像処理を困難にするため、何らかの方法でそれを最小限に抑える必要がある。

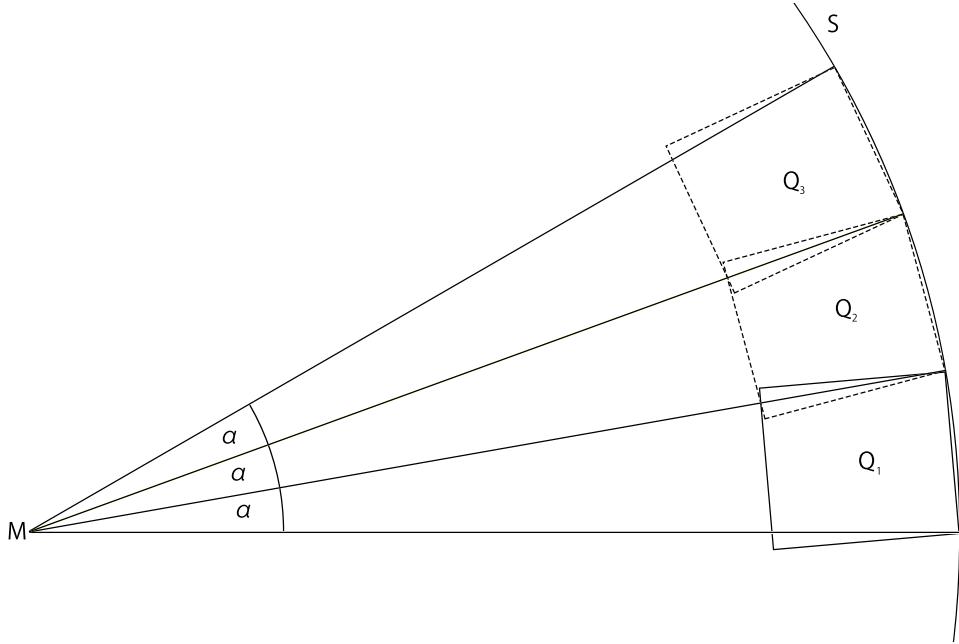


図 3.2: 長方形画像の連続的撮像による重複部分の発生

本研究では、撮像範囲の幅（図 3.3 の $\overline{AB} = \overline{CD}$ ）を小さくし、重複部分の影響を無視できる程度に抑えることで、この問題に対処する。そのため、 $\overline{AB} = \overline{CD}$ がどのくらいの幅であれば画像の重複を無視できるかを調べる。図 3.2 から分かるように、図 3.3 中の ϵ が一画像の片方の重複部分であり、隣接する画像間のピクセルの重複部分は 2ϵ となる。

ϵ を求めるために、まず直角三角形 MCFにおいて頂角 $\alpha/2$ を考えると、

$$\frac{\alpha}{2} = \arctan \left(\frac{\overline{BE}}{\overline{MF}} \right) \quad (3.1)$$

が得られる。また、 $\alpha/2$ において

$$\overline{BE} - \overline{BG} = \tan \left(\frac{\alpha}{2} \right) \overline{ME} \quad (3.2)$$

が成り立ち、したがって

$$\epsilon = \overline{BG} = \overline{BE} - \tan \left(\frac{\alpha}{2} \right) \overline{ME} \quad (3.3)$$

であるから、式 (3.1) を式 (3.3) に代入して整理すると

$$\epsilon = \overline{BE} \left(1 - \frac{\overline{ME}}{\overline{MF}} \right) \quad (3.4)$$

が得られる。本研究で作成したC#プログラム（詳細な説明は第4章を参照）の中のキャリブレーション機能を利用した後のパルスステージの場合、変数 \overline{BE} , \overline{ME} と \overline{MF} は常に把握できていることから、SP レコードの任意の位置においての ϵ を求めることができる。 ϵ の性質を調べるために、まずカメラの最大解像度で撮像を行った場合の計算を進めてみる。ただし、撮像場所においてはSP レコードの一番外側と内側と2つのケースに分けることにする。カメラに直結している顕微鏡の拡大率が1.5xに設定してあるときの次の定数を計算に用いる。

$$\gamma = 0.002366 \text{ [mm/pixel]}, \quad (3.5)$$

$$\gamma^{-1} = 422.65 \text{ [mm/pixel]}. \quad (3.6)$$

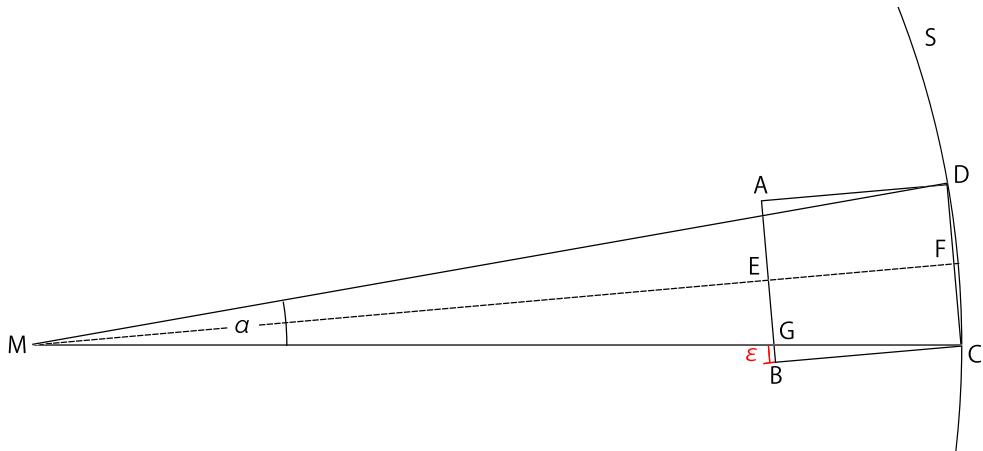


図 3.3: 重複問題を数値化する ρ の幾何学的図

最大解像度で撮像した画像の実際のサイズは

$$\text{height} = \overline{EF} = 5.81 \text{ [mm]}, \quad (3.7)$$

$$\text{width} = 2 \cdot \overline{BE} = 4.87 \text{ [mm]} \quad (3.8)$$

となり、またレコードの外側で撮像したときの \overline{ME} は

$$\overline{ME} = 120.0 \text{ [mm]} \quad (3.9)$$

となる。 $\overline{MF} = \overline{ME} + \overline{EF}$ を考慮しながら式(3.7), (3.8), (3.9)を式(3.4)に代入して計算すると

$$\epsilon_1 = 0.1124 \text{ [mm]} \quad (3.10)$$

となり、式(3.6)の単位換算定数 γ^{-1} をかけると

$$\epsilon_1 = 47.52 \text{ [pixel]} \quad (3.11)$$

という結果が得られる。同様に SP レコードの内側での ϵ を $\overline{ME} = 50.0 \text{ [mm]}$ として求めると

$$\epsilon_2 = 0.2534 \text{ [mm]} = 107.14 \text{ [pixel]} \quad (3.12)$$

となる。ここで、前述したように、隣接する画像の隣接重複距離が 2ϵ であることを思い出すと、最悪の場合の隣接重複距離が

$$2\epsilon_2 = 0.5068 \text{ [mm]} = 214.28 \text{ [pixel]} \quad (3.13)$$

となり、これだけのピクセル数の情報が重複していると無視できない問題と判断せざるを得ない。しかしながら、上記の計算は画像の縦幅 $\overline{AB} = \overline{DC}$ がカメラの最大解像度に設定した場合の ϵ_3 の計算結果であり、縦幅 32 ピクセルに指定した際の計算は以下の通りになる。

$$\text{width} = 2 \cdot \overline{BE} = 0.0378 \text{ [mm]}, \quad (3.14)$$

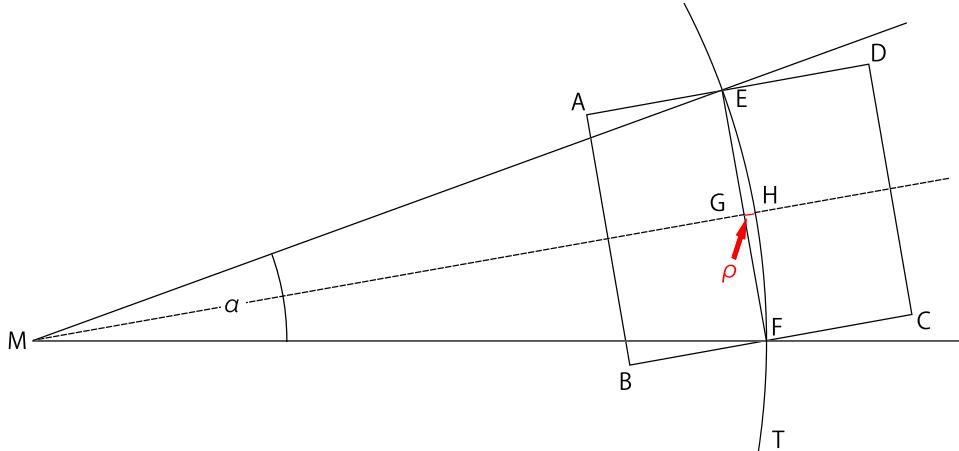
$$2\epsilon_3 = 0.00782 \text{ [mm]} = 3.331 \text{ [pixel]}. \quad (3.15)$$

式(3.15)の結果は最悪条件の下で計算したものであることを考慮しながら、実際の撮像データで幅 3 ピクセルの音溝に大きい変化があり得ないことが確認でき、この程度の重複であれば無視しても良いと判断できる。

3.1.2 湾曲問題

SP レコードに刻まれた音溝はらせん状にレコードの中心へと近づいていく仕組みであるから、レコードのどの位置で音溝の形状を撮像しても湾曲を伴った形状として画像情報が記録される。したがって、複数の連続的に撮像された画像を並べると音溝の湾曲成分が周期的な上下振動として残ることになり、音検出段階でこのゆっくりとした振動が間違って音情報として検出されることを防ぐためには、湾曲の影響を最低限に抑える必要がある。

図 3.4において、長方形 ABCD をカメラの撮像範囲、T を SP レコードの音溝、M を SP レコードの中心、 α を次の画像を取得するための回転角度とする。また、 $\overline{FG} = \overline{GE}$ であり、角 MGF は直角である。このとき、

図 3.4: 音溝湾曲を数値化した変数 ρ

$$\overline{GF} = \frac{1}{2}\overline{AB}, \quad (3.16)$$

$$\frac{\alpha}{2} = \arcsin\left(\frac{\overline{GF}}{\overline{MF}}\right) = \arcsin\left(\frac{\overline{GF}}{\overline{MH}}\right), \quad (3.17)$$

$$\overline{MG} = \cos\left(\frac{\alpha}{2}\right)\overline{MF}, \quad (3.18)$$

$$\rho = \overline{MH} - \overline{MG} \quad (3.19)$$

が成り立つ。式 (3.16)～式 (3.19) より

$$\rho = \overline{MH} \left\{ 1 - \cos \left[\arcsin \left(\frac{\overline{AB}}{2\overline{MH}} \right) \right] \right\} \quad (3.20)$$

重複問題の計算と同様にレコードの最外周と最内周で ρ を求めることにする。まず、画像の横幅を 32 ピクセルと設定し、実際の画像幅を

$$\text{width} = \overline{AB} = 0.0378 [\text{mm}] \quad (3.21)$$

と置いておく（単位換算には式 (3.6) の γ を用いる）。レコードの外周では

$$\overline{MH} = 120.0 [\text{mm}] \quad (3.22)$$

になる。このときの湾曲問題係数 ρ は

$$\rho_1 = 0.001488 [\mu\text{m}] = 0.000629 [\text{pixel}] \quad (3.23)$$

となる。また、レコードの内周では

$$\overline{MH} = 50.0 [\text{mm}] \quad (3.24)$$

であるので、湾曲問題係数は

$$\rho_2 = 0.03572 [\mu\text{m}] = 0.0015 [\text{pixel}] \quad (3.25)$$

となる。比較のため、以下に悪条件（レコードの内側での撮像）のもとで画像幅 2058 ピクセル ($\overline{AB} = 4.87 [\text{mm}]$) のときの湾曲問題係数を示す。

$$\rho_3 = 59.18 [\mu\text{m}] = 25.0 [\text{pixel}] . \quad (3.26)$$

以上の計算より、横幅 32 ピクセルで画像撮像を行った場合の湾曲問題による影響は極めて小さく、無視できる程度であることがわかる。

3.1.3 うねり問題

うねり問題はレコードに開けられた穴が偏心しているために生じる。ここまで幾何学的问题はレコードの原理的性質、すなわち音溝がらせん状に円形のレコードに刻まれていることや長方形の撮像範囲を持つカメラの下でレコードが回転することに起因している。しかし、うねり問題は湾曲問題や重複問題と異なり、不可避の問題ではない。完璧に作られたレコードとそれに正確に適合した実験装置であれば、うねり問題は生じないと考えられる。

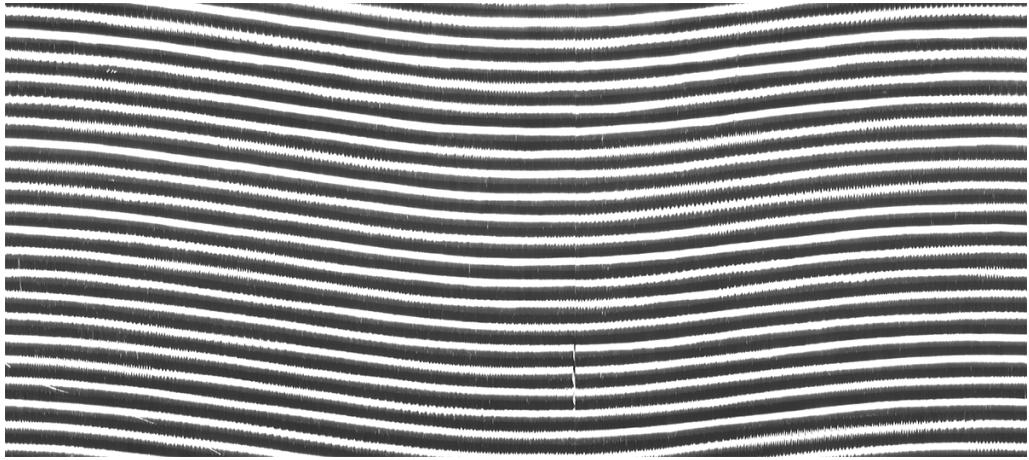


図 3.5: レコード 1 周分の画像データ。横幅は圧縮して表示している。

図 3.5 にレコードの 1 周分の画像データを示す。ページ内に収まるように、横幅は圧縮してある。この画像から音溝がゆっくりと上下に振動していることが分かる。この振動は明らかに音情報ではなく、単にレコードの中央の穴が偏心しているために生じている。本研究においては、この音溝のうねり問題の影響を画像処理ではなく信号処理により抑制する。

3.2 部分読み出し法による画像合成の利点・欠点

撮像時に直面する問題とその解決法については前節で詳しく説明した。本節では、部分読み出し法の利点および欠点について述べる。この議論を進めるため、前節の問題をもう一度リストアップする。

- 重複部分問題
- 湾曲問題
- うねり問題

重複問題と湾曲問題は本研究と同じ実験設定においては必ず発生する。その解決法に関しては、カメラの全画像を使った画像を用い、画像処理を通じて克服する手法もあるが、前節に述べたように、部分読み出しの機能を採用すれば重複問題と湾曲問題の影響が無視できる程度に収まる。これが部分読み出しの利点の一つである。

一方、欠点として挙げられるのは画像結合が増えることである。幅が非常に狭い（本研究では 32 ピクセル）画像データを数多く取得し、画像処理を通して連結させる過程において、隣接する画像データが重複も欠損もなく実際の音溝を連続的かつ正確に再現できるかどうかが部分読み出し法の制度を決める重要なポイントであり、その実現が必ずしも容易ではないことが、この方法の難点である。

第4章 画像データ生成

本研究の画像取得には JAI 社のフルカラーカメラ BB-500GE を用いる。このカメラからの画像取得方法には、「ビデオ（動画像）として取得」と「静止画像として取得」と 2 つがある。前者の方法は昨年の研究で適応され、圧縮をかけずに動画撮影を行えば Matlab で各フレームごとに処理を施し、それらを接続することができる。この方法の利点は、フレームレートを一旦設定した後はカメラが等時間間隔で画像を取得し動画ファイルに格納するため、フレームレートと回転速度の設定さえ正しければ隣接する画像データ同士は重複部分も情報損失も起きず、良好な連続的画像データが得られることである。一方、画像データに対し Matlab で一枚ずつ画像処理を行いたい場合には、ビデオ形式で取得する必要がなく、最初から静止画像形式で取得した方がデータとして扱いやすく、それが動画にする手取の欠点として考えられる。

それに対し、後者の「静止画像として取得する」手法の利点と欠点はどうなるだろうか。最初から画像ファイルとして保存されるので、画像処理には向いている。しかし、画像取得タイミングを何らかの基準に基づき、画像データが連続的になるようにカメラに撮像開始命令を送らなければならない。本研究ではこの方法を適用し、回転ステージの回転角度をマイクロ秒単位で取得し、このデータを基準に撮像命令のタイミングを決める。

4.1 SPRecAnalyzer

カメラやパルスステージとの通信を制御し、大量の画像データを管理するためのプログラム SPRecAnalyzer を作成した。このソフトウェアは C で書かれており、JAI 社が公開しているカメラ制御用 DLL ファイル¹を活用する。パルスステージとの通信には同様に National Instruments 社の GPIB コントローラ専用 DLL ファイル²を使用した。図 4.1 に SPRecAnalyzer の UI を示す。SPRecAnalyzer の目的は画像データの取得であり、キャリブレーションを行った後に AIA (Automatic Image Acquisition) タブで「スターと」ボタンを押すことにより、自動的にレコードの全面画像データ取得プログラムが開始できる。

¹Jai.FactoryDotNET.dll

²NationalInstruments.NI4882.dll

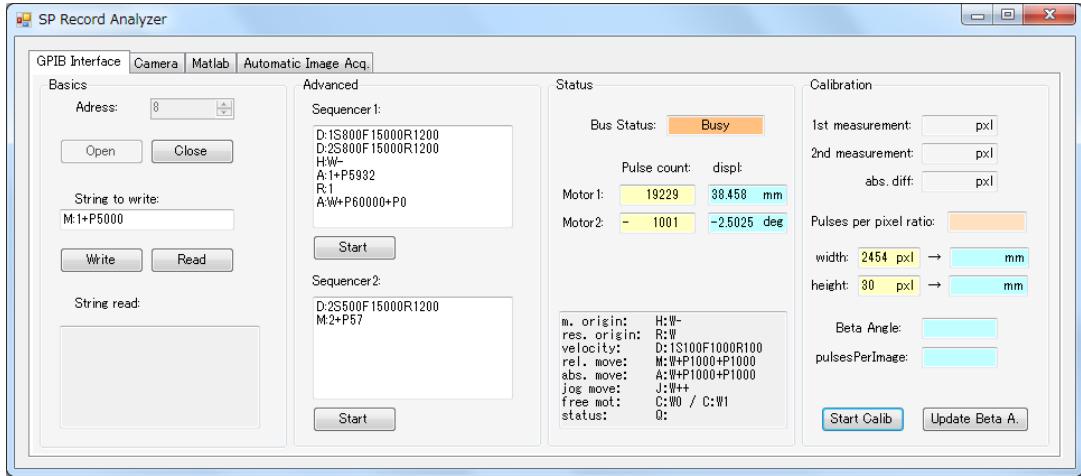


図 4.1: SPRecAnalyzer のユーザーインターフェース

SPRecAnalyzer の UI には重要なタブが 3 つあり、それらの役割について表 4.1 にまとめる。なお、「Matlab」タブもあるが、デバッグ目的で作成したもので、画像データ生成や各装置のセットアップには不要である。

表 4.1: SPRecAnalyzer のタブとその機能

タブ名	機能
GPIB Interface	GPIB 設定、パルスステージ状況表示、キャリブレーション
Camera	カメラ設定
Automatic Image Acq.	自動画像取得

4.1.1 GPIB Interface タブ

このタブの一番左に「Basics」のセクションがあり、そこで GPIB をセットアップできる。パルスステージは標準でアドレスが 8 になっているため、変更する必要がない。「Open」ボタンで通信開始した後に「Write」・「Read」ボタンでバスへの書き込みとバスバッファからの読出しができる。

「Advanced」セクションではパルスステージに対する命令を複数行記述し「Start」ボタンを押すことで順次実行できる。「Status」セクションはバス状況についての情報が表示される。「Calibration」セクションはキャリブレーション機能を実行し、その結果を確認するために設置された。なお、キャリブレーション機能を実行するためには GPIB とカメラがセットアップ済みでなければならない。GPIB をセットアップした後の UI を図 4.2 に示す。

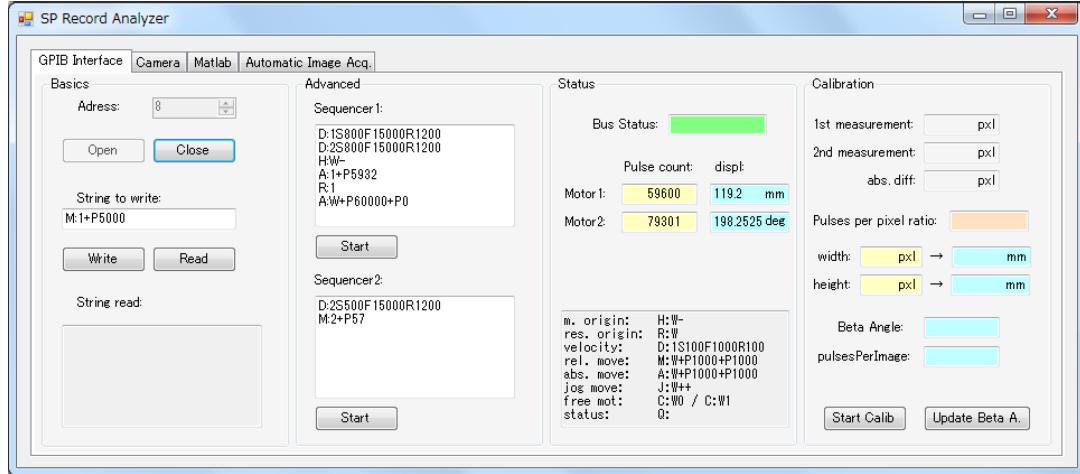


図 4.2: GPIB をセットアップした後の GPIB Interface タブの様子

4.1.2 Camera タブ

Camera タブではカメラ設定を行い画像データを取得することができる。本研究ではカメラの Software Trigger 機能を利用し撮影毎にトリガーを送ることで画像データの取得タイミングを決める。「Gain Control」スライダーではカメラのアナログ利得を設定可能であり、「Exposure Control」スライダーではカメラの露光時間を設定できる。本研究の撮像設定では、レコードが一定速度で回転しながら撮像を行うため、露光時間の値は小さく設定しないと画像データに回転によるぶれが生じる。本研究で用いた設定の詳細については付録 A を参照のこと。

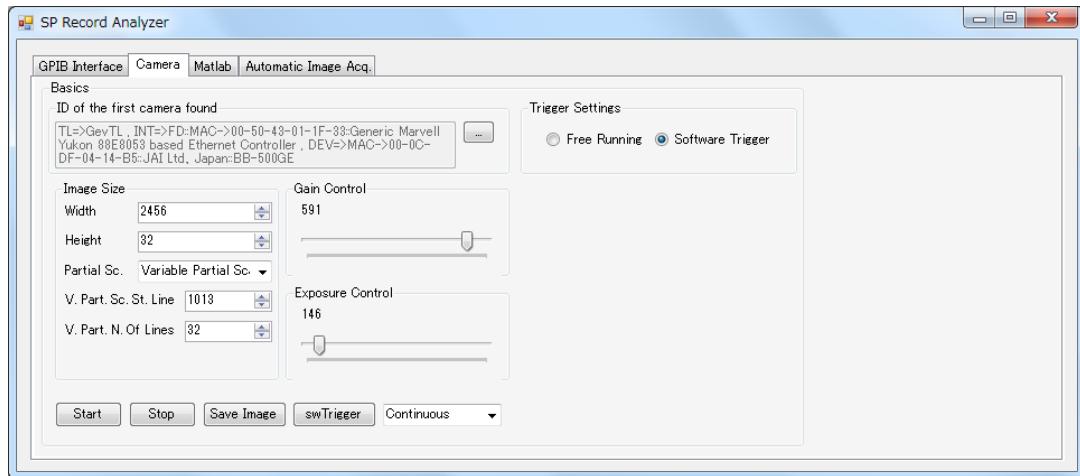


図 4.3: SPRecAnalyzer のユーザーインターフェース

4.1.3 物理的距離検出

物理的距離検出は本研究において非常に重要な機能である。なぜなら、カメラに撮像命令を送信した際に、回転パルスステージが何パルス進んだ後に次の画像を取るべきかを判定するためには回転パルスステージにおいて 1 パルス当たり横方向に何画素進むかという情報が必要不可欠だからである。SPRecAnalyzer では、この重要な変数を「Pulses per pixel ratio」と呼ぶ。

物理的距離検出アルゴリズムの内容を以下に示す。

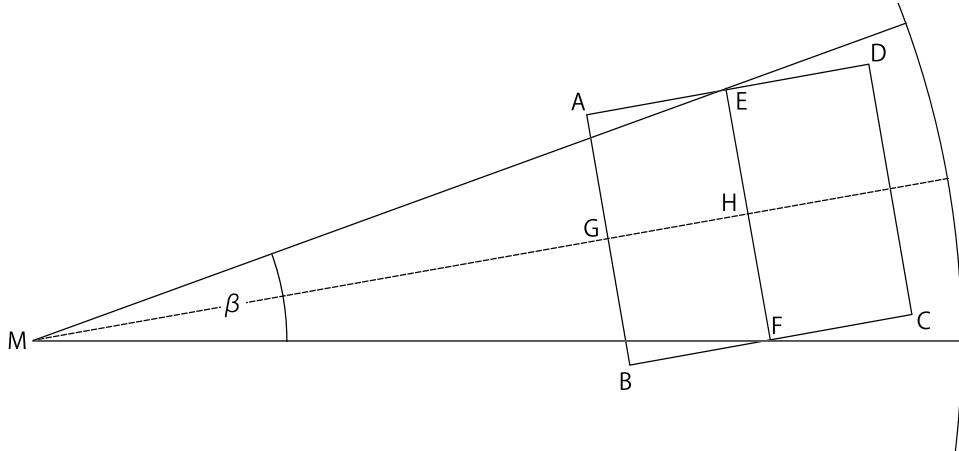
1. レコードの外側の音溝がカメラ撮像範囲の半分くらいに入る位置まで移動
2. 画像を 1 枚撮影
3. Matlab で画像処理を行い、レコードの外側に最も番近い音溝の位置情報を検出
4. 平行移動パルスステージを 400 パルスだけレコード中心に向かって移動
5. 画像を 1 枚撮影
6. Matlab で画像処理を行い、レコードの外側に最も番近い音溝の位置情報を検出
7. 2 回行った画像処理の位置情報を比較し、平行移動後、音溝が何ピクセル移動したかを計算

400 パルス分移動した際に、表 2.2 の定数を利用することで何 mm 平行移動したかが容易に計算できる。この結果を用い、mm/pixel の割合を計算することができる。これがいわゆる 1 ピクセルの物理的サイズであり、これを 1 画像当たりの回転パルス数の計算に利用する。

物理的距離検出のための画像処理アルゴリズムを付録 B に示す。画像処理内容は 2 値化、黒枠削除、ラベリング、および音溝検出にまとめることができる。ここでいう「音溝検出」はあくまでも一定のピクセル数を超える集合を音溝であると判断する処理のことである。

4.1.4 β 角算出

β 角は画像 1 枚当たりに回転すべき角度（単位は度）であり、それをパルス数に換算した値が pulsesPerImage である。SP レコードの回転速度を一定に保ちながらレコードの外側から内側へと移動するに連れて、レコードの表面速度（線速度）が低下する。したがって、レコードの内側に近づくほど、一定幅の画像を撮像する際の「pulsesPerImage」の値が大きくなると考えられる。

図 4.4: 1 画像当たり回転すべき角度 β

この関係を数学的に突き求めてみよう。図 4.4において、カメラの撮像範囲を長方形 ABCD, \overline{MH} をレコードの中心からカメラ撮像領域中心点までの距離とする。また, \overline{EF} を撮像画像幅とする。MHE が直角三角形であるから

$$\frac{\beta}{2} = \arctan\left(\frac{\overline{HE}}{\overline{MH}}\right) \quad (4.1)$$

よって,

$$\beta = 2 \arctan\left(\frac{\overline{HE}}{\overline{MH}}\right) \quad (4.2)$$

と求まる。物理的距離検出の物理的ピクセルサイズと表 2.2 に示す定数を利用し、1 画像当たりの回転パルス数 pulsesPerImage を計算できる。

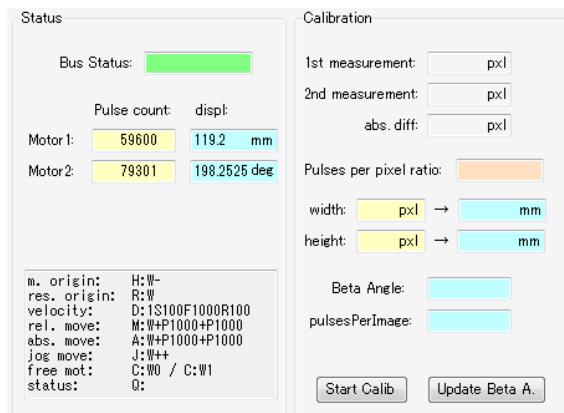


図 4.5: GPIB Interface タブで表示される「Beta Angle」と「pulsesPerImage」

4.1.5 キャリブレーション機能

キャリブレーション機能は GPIB Interface タブに配置された「Start Calib」ボタンで実行できる。自動撮像機能を使う前にこの機能を実行する必要がある。キャリブレーション機能の処理内容を以下に示す。

1. パルスコントローラに機械原点復帰命令を送信
2. パルスコントローラの論理原点をレコードの中心に設定
3. 物理的距離検出を実行
4. β 角算出を実行

なお、パルスコントローラの論理原点設定においては、どのようなレコードを使用しても、その中心の位置が変わらないことから、機械原点から一定のパルス数だけ進むことにより、レコードの中心をカメラの中心に合わせることができる。

4.1.6 自動撮像機能

「Automatic Image Acq.」タブでは自動撮像機能関連の操作ができる。物理的距離検出アルゴリズムでカメラに直結した顕微鏡の拡大率に関わらず正確に 1 ピクセルの物理的サイズが計算できるため、顕微鏡の拡大率を変えても、自動撮像機能で取得する画像データが連続的になる。

またパルス対ピクセルの割合が計算で求まるため、レコードを一周した後に正確に 1500 ピクセルだけ、レコードの中心へと平行移動することができる。平行移動後に撮像する画像データは前の周のデータと比べて正確に 1500 ピクセルずれているため、音検出アルゴリズム「SigExBot」は例えば、Round1 から Round2 のデータに移行するときに、現在追跡中の音溝を新しい画像データの中で容易に特定することができる。

取得される画像データは「SPRecordAnalyzer」フォルダ内に「Round1」、「Round2」、「Round3」... と周別にフォルダに格納される。このデータは次章の画像処理アルゴリズムの対象となる。

第5章 画像処理

SPRecAnalyzer を利用して取得した画像データは数多くの 2456×32 ピクセルの画像ファイルから構成される。原理的には Matlab で直接この画像データを個別にメモリに読み込み画像処理を行うことも可能ではあるが、画像処理アルゴリズムのデバグ作業などを考えると、この非常に幅が狭い画像データを結合して幅が広い画像ファイルにまとめてから画像処理を行った方が、画像データ生成メカニズムの開発やその後の画像処理での結果確認が行いややすくなる。

幅が狭い画像データを数十枚結合し幅が広い画像データを作成しておくのが SpliceBot アルゴリズムの役割である。SpliceBot が作成した画像データを元に音溝に沿って信号データを抽出するのが SigExBot である。そして、最後に、Needle が SigExBot の信号データに信号処理を施して音信号の形に変形を行う。本研究の画像処理は全て Matlab で行うが、Matlab の基本機能のみを用いており、Toolbox は使用しない。

5.1 SpliceBot

Splice¹Bot は Matlab 用の画像合成アルゴリズムである。SpliceBot が正常に作動するためには、画像ファイル名、フォルダ名とフォルダ構造を守らなければならぬ。すなわち、アルゴリズムファイル「SpliceBot.m」と同じフォルダに「Round**」フォルダを置く。**は 1 から始まる画像データ生成の際の周番号であり、例えば、「Round1」フォルダにはレコードの最外周を一周したときの画像データを格納する。アルゴリズムが要求する Round** フォルダの中のフォルダ構造を図 5.1 に示す。



図 5.1: フォルダ Round** フォルダの内容

¹ 「splice」の意味合いについて。Oxford Dictionary より： Join (pieces of timber, film, or tape) at the ends

BWImages フォルダには 2 値化画像データが格納される。アルゴリズムの中では「ConvertToBWImages」メソッドがこのデータを作成する。次段階の画像処理アルゴリズムは 2 値化データを必要としないため、処理時間短縮を目指すならコメントアウトしてよいメソッドである。2 値化データはあくまでも画像データ確認の一つの手段にすぎない。

rawdata フォルダには画像データ生成で作成される画像データを格納する。画像データのファイル形式は「AIA ○○.bmp」。○○は 1 から始まる通し番号である。SpliceBot はこのフォルダの画像データを処理対象にする。splicedImages フォルダは SpliceBot が画像合成処理を行った後の画像データを格納する場所である。画像合成後に作成された画像の例を図 5.2 に示す。markedImages フォルダには SigExBot が追跡マークをプロットした後の画像データが入る。

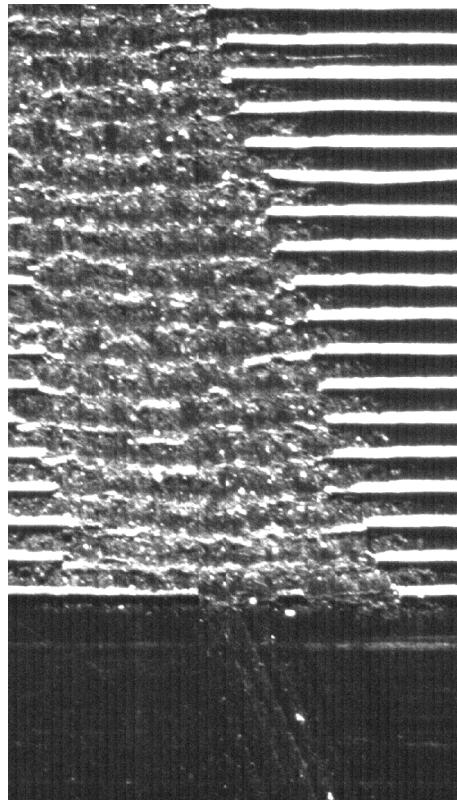


図 5.2: spliceBot が作成した合成画像データ（レコードの表面が削れた箇所）

5.1.1 処理内容

Matlab 環境内でパスを SpliceBot.m が格納されているフォルダに設定した後に Command Window に「spliceBot = SpliceBot();」を入力すると SpliceBot オブジェクトが生成される。生成時に、クラスファイルと同じフォルダにある Round** フォルダが全てオブジェクト内に登録される。

アルゴリズムを実行するには「spliceBot.Start();」を入力する。実行開始後、全ての登録されている Round ○○ フォルダに対し、以下の処理が行われる。

1. rawdata フォルダ内の bmp ファイル数を検出
2. 画像データを 50 枚ずつ 1 つの bmp ファイルに合成し、splicedImages フォルダに格納する
rawdata の画像データを束ねる前に各画像に対して以下の処理を行う：
 - (a) 幅 1 ピクセルの枠を削除（画像データのサイズが縦横 2 ピクセル縮小）
 - (b) 上下反転、90 度回転、上下反転
3. 残り画像データが 50 枚以下の場合、残り画像数だけ 2. と同じ手法で合成する
4. splicedImages フォルダ内の.bmp ファイルの個数を検出
5. 合成画像データをグレースケールに変換する
6. 合成画像データを 2 値化する
7. 処理対象データパスを次の登録 Round フォルダに変更して 1. – 6. をフォルダがなくなるまで繰り返す

5.2 SigExBot

このアルゴリズムは本研究の画像処理の中で中心的な役割を果たしている。処理対象となるのは SpliceBot が作成した画像データである。SigExBot²は SpliceBot と同じフォルダ構造を要求する。本研究の画像処理アルゴリズムは殆ど例外なく全てクラスファイルとして定義されている。したがって、SigExBot も SpliceBot と同様にクラスのインスタンス化が最初に必要である。Matlab の Command Window に「sigExBot = SigExBot();」でクラスをインスタンス化し、「sigExBot.Start();」で処理を開始する。

5.2.1 処理内容

「sigExBot = SigExBot();」と入力してオブジェクトを作成したとき、以下の処理が行われる。

1. 様々のパラメータの初期化
2. Round** フォルダの個数を検出し、処理対象フォルダを Round1 と設定する
3. Round1 フォルダ内の bmp ファイル数を検出
4. Round1 フォルダの検出された画像データを全て RAM に読み込む

² 「Signal Extraction Bot」の略称

「sigExBot.Start();」と入力して処理を開始させることにより、以下の処理が実行される。ただし、信号修正機能については後に説明するため、ここでは省力する。

1. 開始直後、設定された画像データのピクセル値を左下から上へと向かって調べていき、音溝を探す
2. 縦軸に対する音溝の中心を計算し、現在の位置をそれに合わせる
3. 現在位置のピクセルに色を付ける（デバッグ用）
4. 設定されたステップサイズの値だけ右へと進む
5. 2.に戻り、繰り返し処理を行う

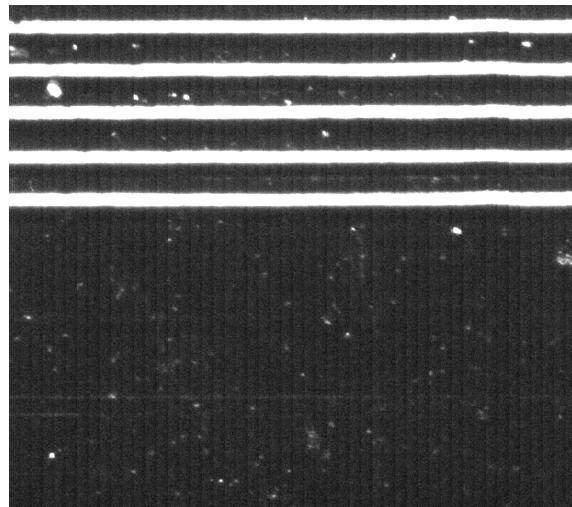


図 5.3: 処理開始直後、アルゴリズムは左下から上に向かって音溝を探す

基本的な処理内容は上記のようになっている。ただし、右へと進んでいき画像の右端に到達すると、次の画像が現在処理対象画像として設定される。また、画像データを何回か一周した後には画像の上部に到達し、ある一定のピクセル数を超えると Round 番号がインクリメントされ、次の画像データが RAM に格納され、上記の処理が繰り返し行われる。

なお、画像データ生成ソフトウェアで Roundx と Roundx+1 の撮像縦座標の差が正確に 1500 ピクセルになるように設定されているため、Roundx から Roundx+1 へと進むときは、現在の縦座標パラメータから 1500 ピクセルを引くことで、途切れることなく音溝追跡処理が続けられる。

4. のステップサイズに関して説明する。蓄音機でレコードを再生するとき、レコードの回転角速度が一定であるため、針が外側から内側へと向かっていくに連れて、針の直下を回るレコードの線速度が減少する。たとえば、同じ 1000Hz の正弦信号をレコードの外側と内側とで比較すると、内側の方が音溝の周期が短くなる。これは線速度が減少しても、時間軸に対しては同じ 1000Hz の信号を保たなければならない

からである。この音溝の横軸に対する収縮に対応するため、処理対象 Round フォルダが変わるたびにステップサイズを調節する。中心に近づけば近づくほど、ステップサイズが小さくなる。

また、Round フォルダが変わるたびに、アルゴリズムの追跡マークが付けられた画像データが markedImages フォルダ内に保存される。これはあくまでもデバッグ用であり、次段階の信号処理アルゴリズムが必要とするデータではない。図 5.4 に追跡マーク付きの画像を示す。したがって、処理時間短縮を目指すなら追跡マーク関連のメソッドをコメントアウトするとよい。図 5.4 に示された追跡マークは実際に検出されるデータと一致する。

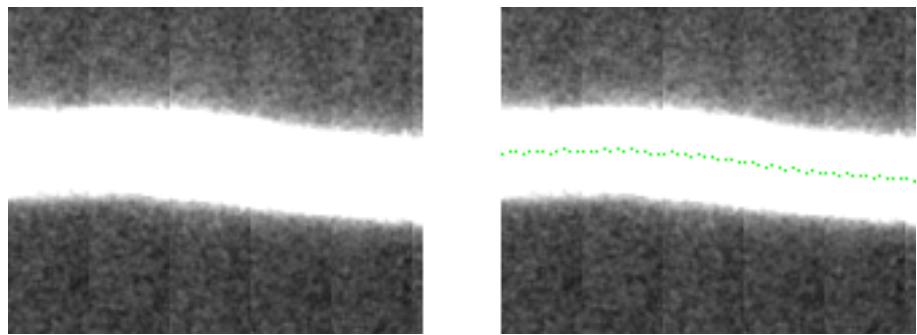


図 5.4: 元画像（左）と追跡アルゴリズムが適応された後の画像（右）

5.2.2 信号修正機能

本研究で音検出対象となる SP 盤レコードには擦り傷、割れやひびが入っているものが多い。それに加え、音溝から反射する光が音溝読み取り手段になるため、蓄音機と異なり微細なほこりにも影響を受ける。これらの悪影響に対応できるように音信号検出アルゴリズムに様々な工夫を施した。乱れた信号を修正するための機能を以下にリストアップする。

- GapFilling
- TryToRecover

Matlab プログラムとの関連性がわかるように機能名は英語で表記した。これらの機能について、以下に説明する。

5.2.3 GapFilling

GapFilling³機能はほこり、擦り傷やひびに対応するために実装した。この機能を説明するためには、まず最初に SigExBot の状態について記述する必要がある。SigExBot が取り得る状態を以下に示す。

- ライントレース中（緑）
- 警戒中（赤）
- ギャップ充填中（青）

かっこの中の色はデバッグ用の追跡マークの色に該当する。通常、音信号検出アルゴリズムは「ライントレース中」の状態にある。アルゴリズムがなぞろうとしている白い線の幅が急激に 5⁴ピクセル以上変化すると、アルゴリズム状態が「警戒中」へと遷移する。警戒中の追跡点線は赤くなり、幅が平均的な値に戻らない限りアルゴリズムは白い線の中心を追わず幅の回復が確認できるまではまっすぐ進んでいく。なお、アルゴリズムが追いかける白い線は音溝の壁に反射する光にすぎず、当然のことながら、反射光で撮像する画像データ内の白い線の幅は一定にはならない。その幅は、レコードの回転に伴って伸縮する。映り具合が多少変わることによる幅の自然な変化と、ほこりやひびによる幅の変化を区別するため、アルゴリズムの中で平均的な幅を常に計算している。そのため、100 ステップに渡って画像データ内の音溝幅が 5 ピクセル大きくなてもアルゴリズム状態は「警戒中」とならず、追跡処理が通常通りに続けられる。

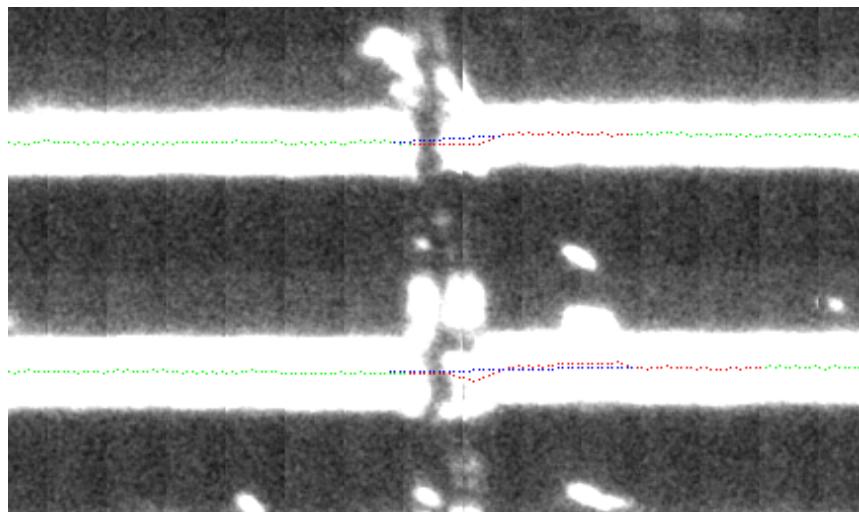


図 5.5: GapFilling 機能によるひびで乱れたデータの修正

³ギャップ充填

⁴執筆時の値

一旦アルゴリズム状態が「警戒中」となった後は、音溝データの幅が平均値の5ピクセル以内に戻らなければ回復できない。音溝幅が平均値 ± 5 ピクセル以内になつたとしても、その状態が30ステップ続かなければならない。これらの条件が満たされれば、アルゴリズムは「警戒中」から「ライントレース中」に戻り、追跡マークが赤から緑に変わる。

しかし、警戒中で進んだ間の信号データが満足のできないものになる場合もあり、GapFilling の中心的な機能である「ギャップ充填」が警戒中からライントレース中に戻る直前に行われる。この機能は、信号データを上書きする機能である。上書きする対象となるのは警戒中に入ったところから、ライントレース中に戻ったところの26ステップ前までである。この範囲をギャップとし、2点が直線で結ばれる用にアルゴリズムが追跡マークを打ち、信号データを上書きする。この処理を行っている間はアルゴリズム状態は「充填中」となり、追跡マークは青に変わる。埋め立て処理が終わった後、アルゴリズムはギャップ充填処理直前の位置に戻り、状態は「ライントレース中」となる。

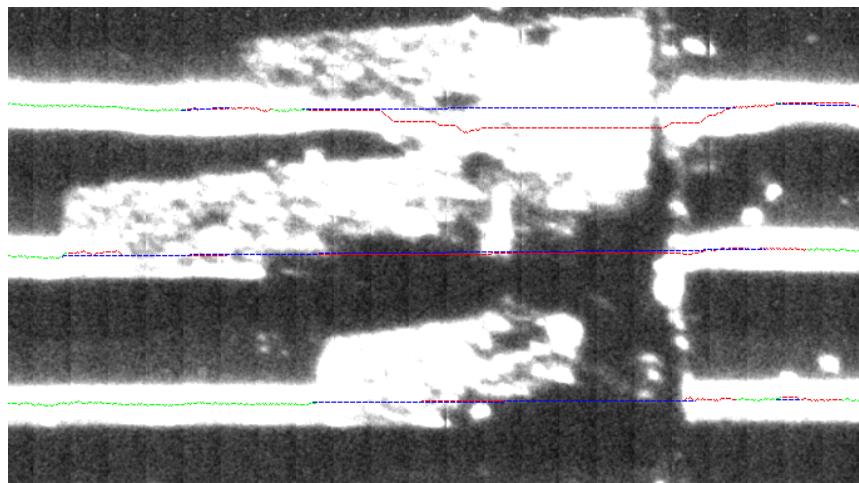


図 5.6: GapFilling 機能による割れで乱れたデータの修正

図 5.6 はレコードが完全に割れた画像の場合の追跡マークである。青い点線が赤点線のデータを上書きしている。これは GapFilling 機能が正常に動いている例であり、アルゴリズムは乱れた音溝データの両端を直線で結び、データ修正が成功している。同様に、図 5.5 にはひびが入った場合の充填機能の例である。

一方、図 5.7 は GapFilling 機能が判断を誤り、音信号を誤って修正した例である。これは音信号データを悪化させることになり、阻止すべきものであるが、ギャップ判定の判断基準を変えるなどしない限り、GapFilling 機能の望ましからぬ副作用として黙視せざるを得ない。

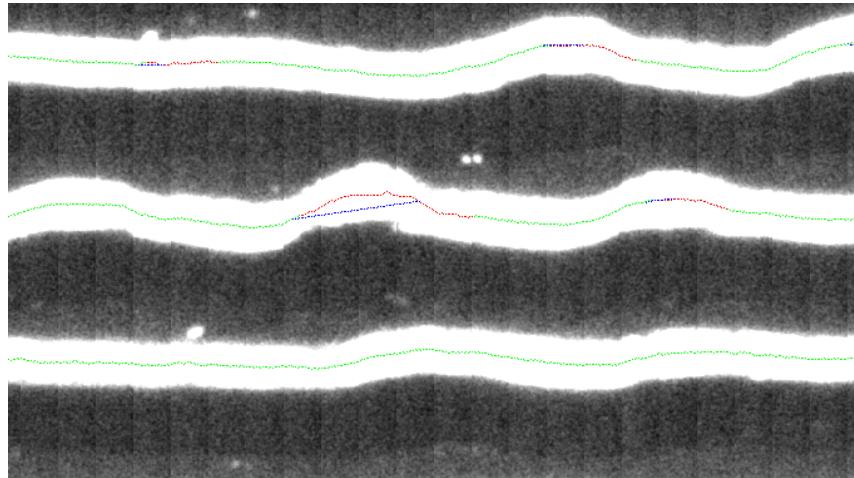


図 5.7: GapFilling 機能が正常な音信号データを誤って修正した例

5.2.4 TryToRecover

図 5.8 の中心にある音溝データを見てみると赤い点線が、アルゴリズムの仕様の結果、黒領域に入っていることが確認できる。いうまでもなく、一旦黒領域に入ると抜け出すことは困難であり、アルゴリズムは何万ステップも警戒中状態となり、ひたすら直進する状態に陥る。こうなったときに、TryToRecover 機能が呼び出される。

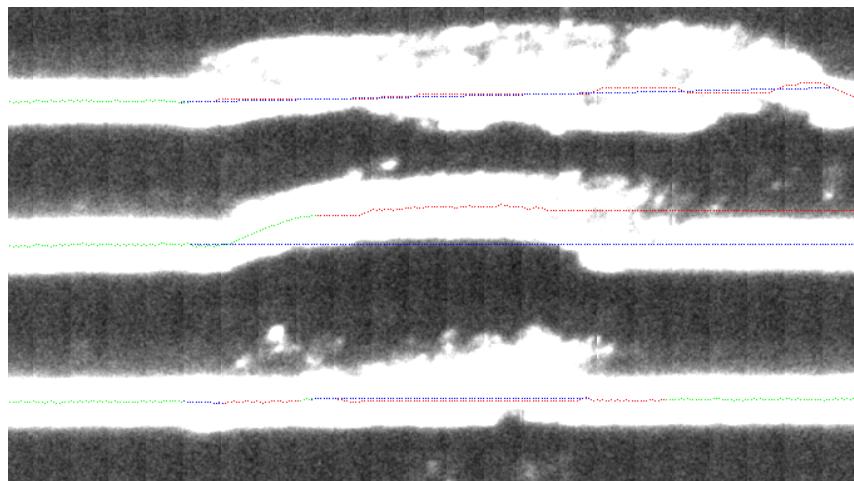


図 5.8: TryToRecover 機能による無限警戒状態からの回復

TryToRecover 機能の呼び出し条件は「黒領域（アルゴリズムの閾値以下のピクセル領域）を 600 ステップ以上連続的に進んだ」である。呼び出された後は現在の縦座標⁵を 750 ステップ前（まだ黒領域ではなかった箇所）の縦座標の値に設定し、

⁵SigExBot.m のプログラム中でいえば CurrentY

これでできた点を終点とおく。次に 750 ステップ前の座標点を読み込み、この点と終点を直線で結び⁶、アルゴリズムが終点からライントレースを再開する。

5.3 Needle

このアルゴリズムは SigExBot が検出した信号を処理対象とし、フーリエ変換や振幅調整などを通じて元信号を wav ファイルへと変換できる形にしてから実際に wav ファイルとして保存する。SigExBot が音溝画像データから抽出した信号データを図 5.9 に示す。この信号データは処理が終わった後に「sigExBot.Signal」でアクセスできる。Needle アルゴリズムは前述の画像処理アルゴリズムと同様に「needle = Needle(sigExBot.Signal);」と入力してインスタンス化できる。インスタンス化の際には処理対象信号データを引数として渡す必要がある。処理開始には「needle.Start();」と Matlab の Command Window に入力する。

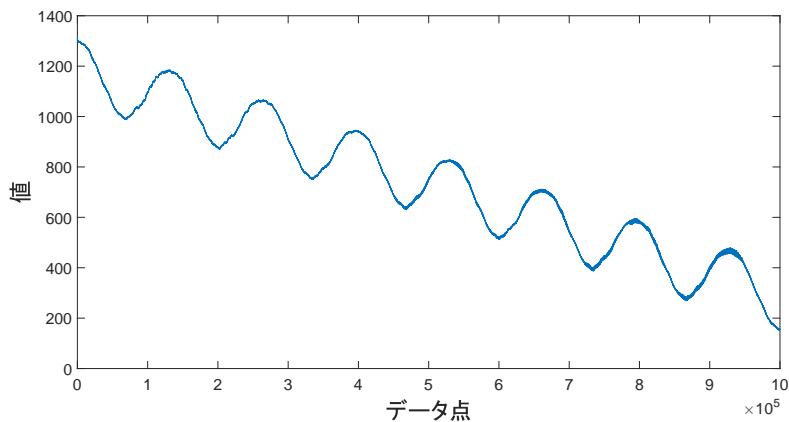


図 5.9: SigExBot が出力した信号データ

図 5.9 に示す信号データについて考察する。まず特徴的な点としてゆっくりとした正弦波に近い振動成分が挙げられる。これはうねり問題の影響である。レコードの穴が偏心していることによる音溝撮像への影響が理論上正弦波になることと、音信号より遙かに振幅が大きく、周期が長いことを考慮すれば、間違いなくそう判断できる。また、この信号を構成するもう一つの成分として、左上から右下に向かう線形のトレンドが挙げられる。これは単に音溝が直線であるために、それをなぞることによって、レコードの中心に近付いていくことの影響である。

音信号データはもっと振幅が小さく、図 5.9 ではその存在の確認が難しい。図 5.9 の一部を拡大した図が図 5.10 である。音信号が確かにうねり信号に重畠していることが分かる。

⁶GapFilling と同様にデータは上書きされる

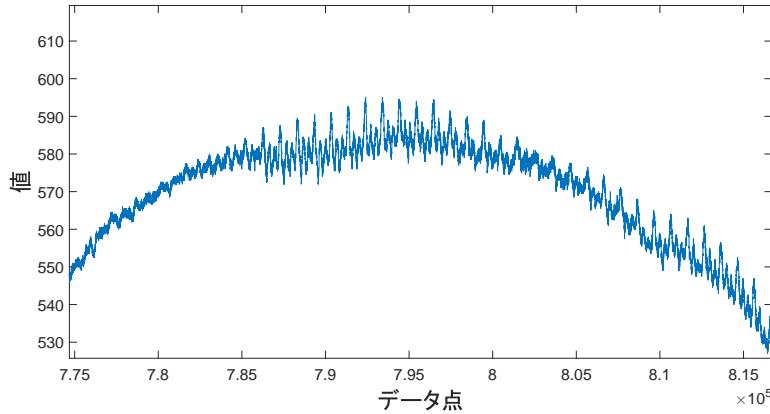


図 5.10: SigExBot が output した信号データを一部拡大した結果

5.3.1 処理内容

図 5.9 に示す信号データからうねり成分と直線成分を取り除き、音ファイルへと変換できるように振幅を調節し、極力ノイズを除法するのが Needle アルゴリズムの機能である。アルゴリズムの個別の機能を以下にリストアップする。Matlab プログラムとの関連性がわかるように処理名は英語で表記している。

1. TrendCorrectSignal
2. FFT
3. Crop
4. AdjustAmplitude
5. NeedleSimulation
6. Savewav

TrendCorrectSignal はいわゆるトレンド修正処理である。全信号を対象とする 1 次多項式近似で直線成分を抽出し、それを信号データから減算する。Matlab が用意する「polyfit」、「polyval」で実装している。

FFT は Fast Fourier Transform の略であり、高速フーリエ変換処理のことである。この処理の目的は「うねり成分削除」と「ノイズ除法」である。実装には Shmuel Ben-Ezra 氏が Matlab の公式ファイル交換サイト（Matlab File Exchange）に投稿した「fftf.m」を利用した。ただし、バンドパスフィルタリングができるよう改造を行った。バンドパスフィルタリングで 100–15,000Hz 以外の周波数成分を全部削除する。これにより低周波数ゆらぎと高周波数ノイズをある程度抑制することに成功した。しかしながら、それでもまだノイズが残っており、フィルタリングされる周波数領域を例えば、100Hz–5,000Hz に設定すると、高周波数ノイズがさらに減少する半面、抽出しようとしている音声データに影響が生じ、「つ、ち」などのいわゆる破

擦音が聞こえなくなり、音声が全体的にこもる結果となった。したがって、ノイズ除去法には別の手段で対応する必要がある。

Crop は信号の前後のデータ点を数百個を除法する処理である。高速フーリエ変換はその数学的原理上信号が周期関数であることを仮定しており、フーリエ変換に際して信号の前後のデータ点が接続しているものとして処理が行われる。これにより信号は前後のデータが乱れており、それを削除することでこの問題に対応することにした。次に AdjustAmplitude 処理が実行される。文字通り、信号の振幅を調節する処理であり、最高振幅が 1 を超えないように設定した。この処理は後で Matlab の audiowrite 関数を利用するため必要な処理である。NeedleSimulation は仮想針の動きをシミュレートすることにより蓄音機と同等レベルの音質が得られないかと考えて試しに作成した機能である。仮想針の結果は FFT 段階の信号とともに自動的に wav ファイルとして保存される。

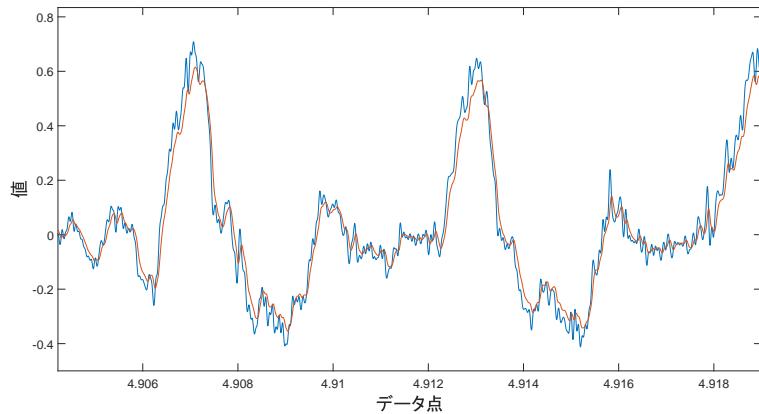


図 5.11:

図 5.11 に Needle 機能を適用した信号（オレンジ）と適用しない信号（青）を比較して示す。仮想針の信号処理後の信号の方がややゆっくり変化しており、ギザギザが少なくなることがわかる。しかしながら、Needle のパラメータを調整して高周波ノイズを抑制し、信号をさらにスムーズにしようとしたところ、フーリエフィルタで通過領域を過渡に狭くした場合と同様の現象が起こり、高周波ノイズが減少するとともに、音声データも悪化する結果となった。

第6章 結論

本研究では使用したカメラが持つ部分読み出し機能を用いてSPレコードからの音検出を試みた。レコード1枚の音溝画像を完全にデータ化するプログラム「SPRecAnalyzer」を作成し、それにより生成したデータを新たに開発した画像処理アルゴリズムにより解析した。その過程で作成された音信号ファイルを再生すると人間の声が聞こえており、ある程度目的を達成したといえる。しかし、画像処理を用いた方法は原理的には針を用いる再生法よりも高音質な音データが得られるはずであるにも関わらず、本研究で用いた方法では通常の蓄音機に匹敵する音質が得られなかつた。蓄音機の音質と比べると、本研究で得られた再生音にホワイトノイズの成分がより強くなっていることが分かる。このホワイトノイズを低減するためFFTのバンドパスフィルタリングで処理したところ、高周波成分が減衰した半面、音信号成分も劣化し、より籠った感じの音になった。これは、人間の音声にも周波数8kHz以上の成分があり、ノイズと同領域に存在しているため、音声信号とノイズを明確に区別することが困難であることに起源している。動的フィルタの適応により、ノイズ削除の効率を向上させられると考えられる。しかし、オープンソース音データ編集ソフトウェア「Audacity」の動的ノイズ削除フィルタ¹を適応しても、蓄音機による再生音より音質が低いことから、高品質の音声が取得できるためには、画像データ生成段階でより音溝に忠実な形の画像が得られるように工夫を重ねるべきと考えられる。

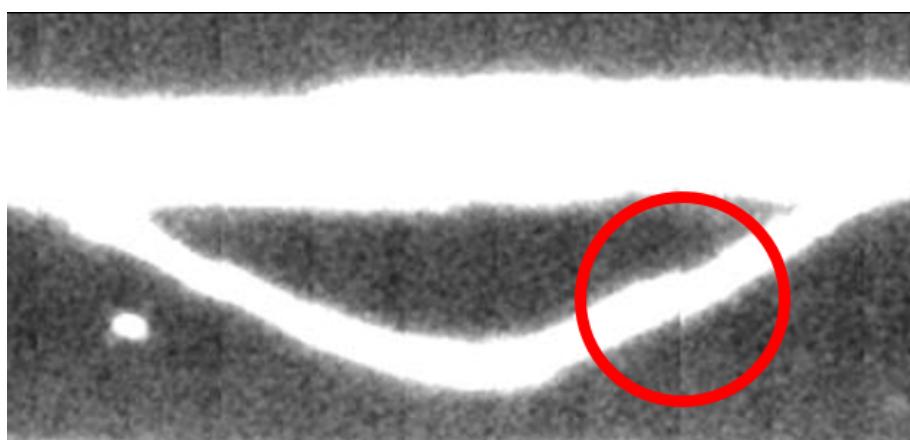


図 6.1: SpliceBot が作成した連続的画像データの一部にずれが生じている

¹ 「エフェクト→ノイズの削除」により適応できる

また、図6.1に示すように、本研究で用いた手法では、隣接する画像同士が8ピクセルほどずれことがある。これはパルスステージ回転軸の角度情報が離散的²であること、および、現在位置情報をパルスコントローラから取得する頻度に上限があることが原因であると考えられる。したがって、ノイズ削除アルゴリズムの見直しとズレ発生メカニズムの研究と対策が今後の課題である。画像データを静止画像ファイルとして取得するのではなく、動画像の形で保存する手法もズレ問題の解決に繋がるかもしれない。

²パルス数

謝辞

本研究を進めるにあたり、お忙しい中、終始、御指導、御鞭撻をいただいた北海学園大学工学部電子情報工学科魚住純教授に心より感謝いたします。

参考文献

- [1] 魚住 純：光と画像による古レコードの非接触再生＝蟻管・SP 盤を針を使わず
に再生する＝，光アライアンス，23，5，pp. 21–25，2012.
- [2] 魚住 純：画像処理によるモノラル円盤レコードからの音声再生，北海学園大学
工学部研究報告，No. 35，pp. 119–129，2008.
- [3] 敦谷 蓮：画像処理による SP 盤の音再生＝CCD の部分読み出しの利用＝，北
海学園大学電子情報工学科，魚住研究室，pp. 2–3，2015.

付 錄 A SPRecAnalyzer の使用法

SPRecAnalyzer は画像データ生成処理過程を管理しやすくするために作成された。市販のソフトウェアと異なり、十分にテストされているわけではなく、予想外の操作を受けた場合にはクラッシュすることもあると考えられる。SP レコード 1 枚の音溝を完全画像データ化するための基本的な操作法を以下に示す。

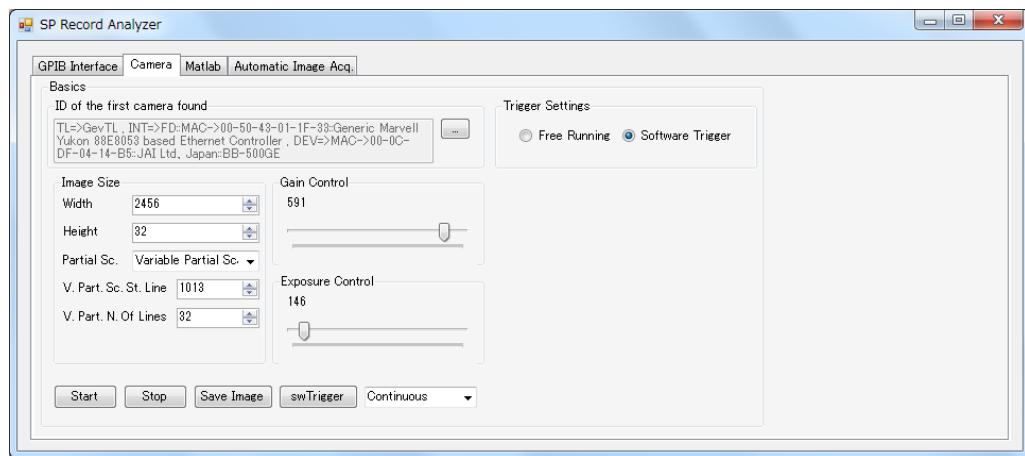


図 A.1: SPRecAnalyzer のユーザーインターフェース

1. 光源、カメラ、パルスステージコントローラに電源を供給し ON にする
2. パルスステージコントローラの「HOST/MANUAL」ボタンを押す
3. プログラムを立ち上げて「GPIB Interface」タブが開いていることを確認
4. 「Open」ボタンでパルスステージとの通信を開始
5. 「Camera」タブを開く
6. カメラ設定が図 A.1 と同じになるように設定
7. 「Start」ボタンを押す。「BB-500GE」ウィンドウが開くことを確認
8. 「swTrigger」ボタンを押す
9. 「BB-500GE」ウィンドウに画像データが表示されることを確認
10. 「GPIB Interface」タブに戻り「Start Calib」ボタンを押す
11. 「Automatic Image Acq.」タブを開き「Start」ボタンを押す

付録B 物理的距離検出プログラム

B.1 PixelProcessing.m

```

1 % -----
2 % -- - Load Image Data - - -
3 % -----
4 tic
5 % load the original image
6 imgOriginal = imread('img/SmallP59000FL.bmp', 'bmp');
7 % load the original image
8 %imgOriginal = imread('img/P60400FL.bmp', 'bmp');
9 img = uint16(imgOriginal);
10 [imgHeight, imgWidth] = size(img(:,:,1));
11 %
12 % -- - GrayScale Bitmap - - -
13 %
14 grayImg = img(:,:,1) + img(:,:,2) + img(:,:,3);
15 grayImg = grayImg / 3;
16 imgNmbrOfPixels = length(grayImg(:));
17 %
18 % -- - - - - - - - - - - - - - -
19 % -- - - create TwoValImg - - -
20 %
21 threshold = 80;
22 TwoValImg = uint16(zeros(imgHeight, imgWidth));
23 % loop through all the pixels.
24 % The Pixels values go from 0 to 255*3
25 for pixel = 1:imgHeight*imgWidth
26     if grayImg(pixel) > threshold
27         TwoValImg(pixel) = 1;
28     else
29         TwoValImg(pixel) = 0;
30     end
31 end
32 %
33 %
34 %imageData = ImageDataClass(imgWidth, imgHeight);
35 %imageData.setup(grayImg, threshold);
36 %
37 % -- - - - - - - - - - - - - - -
38 % -- - - Show Results - - -
39 %
40 figure(1);

```

```

41 imagesc(imgOriginal);
42 axis image;
43
44 figure(2);
45 % display the image in matlab
46 imagesc(grayImg); colormap(gray);
47 axis image;
48
49 figure(3);
50 % display the image in matlab
51 imagesc(TwoValImg); colormap(gray);
52 axis image;
53
54 % - - - - - - - - - - - - - - - -
55 % - - - Pixel Processing - - - -
56 % - - - - - - - - - - - - - - - -
57 ObjectExtraction();

```

B.2 ObjectExtraction.m

```

1 X = TwoValImg;
2 [m,n] = size(X);
3
4 % - - - - - - - - - - - - - - - -
5 % - - - DELETE FRAME PIXELS - - -
6 % - - - - - - - - - - - - - - - -
7 X(1,:) = 0;
8 X(m,:) = 0;
9 X(:,1) = 0;
10 X(:,n) = 0;
11
12 % - - - - - - - - - - - - - - - -
13 % - - - - - - - LABELING - - - -
14 % - - - - - - - - - - - - - - - -
15 label = 0;
16 for kx = 2:m-1;
17     for ky = 2:n-1;
18         W = zeros(3);
19         W(1:3,1:3) = X(kx-1:kx+1, ky-1:ky+1);
20         maxW = max(W(:));
21         if X(kx,ky) == 1 && X(kx,ky-1) == 0 && maxW == 1;
22             label = label + 1;
23             X(kx,ky) = label;
24         end
25         if X(kx,ky) == 1 && maxW >= 1;
26             X(kx,ky) = maxW;
27         end
28     end
29 end
30 figure(4);
31 imagesc(X); axis image;
32 colormap(gray);
33

```

```

34 % - - - - - - - - - - - - - - - - - - - -
35 % - DELETE OVERLAPPING LABELS - -
36 % - - - - - - - - - - - - - - - - - - - -
37 while 1
38     kk = 0;
39     for kx = 2:m-1;
40         for ky = 2:n-1;
41             if X(kx,ky) ~= 0 && X(kx+1,ky) ~= 0 && X(kx,ky) ~= X(kx+1,ky)
42                 old = X(kx,ky); new = X(kx+1,ky); kk = 1; break;
43             end
44         end
45     end
46     if kk == 0
47         break;
48     end
49     for kx = 2:m-1;
50         for ky = 2:n-1;
51             if X(kx,ky) == old;
52                 X(kx,ky) = new;
53             end
54         end
55     end
56 end
57 toc
58 figure(5);
59 imagesc(X); axis image
60
61 % - - - - - - - - - - - - - - - - - - - -
62 % - - - OBJECT EXTRACTION - - - -
63 % - - - - - - - - - - - - - - - - - - - -
64
65 indexCounter = 0;
66 existingPixelObjects = [0];
67 PixelObjects = PixelObject_.empty(0, 0);
68 map = [];
69 for kx = 2:m-1;
70     for ky = 2:n-1;
71         tmp = X(kx, ky);
72         if tmp ~= 0;
73             ex = false;
74             for ent = existingPixelObjects;
75                 if ent == tmp;
76                     ex = true;
77                     break;
78                 end
79             end
80             if ex ~= 0;
81                 % The thing already exists!
82                 PixelObjects(map(tmp)).addPoint([kx, ky]);
83             else
84                 % The thing didn't exist!
85                 indexCounter = indexCounter + 1;
86                 existingPixelObjects(indexCounter) = tmp;
87                 % Here we need to add another entry to the map!

```

```
88         map(tmp) = indexCounter;
89         PixelObjects(indexCounter) = ...
90             PixelObject_(indexCounter);
91             PixelObjects(indexCounter).addPoint([kx, ky]);
92         end
93     end
94 end
95
96 % -----
97 % PRINT NMBR OF BIG PIXELOBJTS -
98 % -----
99
100 pixelThreshold = 1000;
101 nbrOfObjects = 0;
102
103 for ent = PixelObjects;
104     if ent.EntryIndex > pixelThreshold;
105         ent.EntryIndex
106         nbrOfObjects = nbrOfObjects + 1;
107     end
108 end
109 nbrOfObjects
```

付録C 画像処理プログラム

C.1 SpliceBot.m

```

1 % -----
2 % - - - Splice Bot Class - - -
3 % -----
4 classdef SpliceBot < handle
5     properties
6         CurrentRoundNmbr;
7         NmbrOfImages;
8         MaxFolderNmbr;
9         NmbrOfSplicedImages;
10    end
11    methods
12        %
13        % - - - Constructor - - -
14        %
15        function obj = SpliceBot()
16            obj.CurrentRoundNmbr = 1;
17            obj.GetNmbrOfFolders();
18        end
19        %
20        % - - - Start - - -
21        %
22        function Start(obj)
23            while obj.CurrentRoundNmbr <= obj.MaxFolderNmbr;
24                obj.GetNmbrOfFiles();
25                obj.AddImages();
26                obj.AddRemainingImages();
27                obj.GetNmbrOfSplicedImages();
28                obj.ConvertToGrayScale();
29                obj.ConvertToBWIImages();
30                obj.Next();
31            end
32        end
33        %
34        % - - - Next - - -
35        %
36        function Next(obj)
37            obj.CurrentRoundNmbr = obj.CurrentRoundNmbr + 1;
38        end
39        %
40        % - - - Add Images - - -

```

```
41 % - - - - -
42 function AddImages(obj)
43     path = 'Round%d/rawdata/AIA%d.bmp'
44     path_ = 'Round%d/splicedImages/splicedImage%d.bmp'
45     splicedFullImages = floor(obj.NmbrOfImages/50);
46
47     for p = 0 : splicedFullImages-1;
48         tmp1 = imread(sprintf(path, ...
49                         obj.CurrentRoundNmbr,p*50 +1), 'bmp');
50
51         % crop the black pixels!
52         tmp1(end,:,:)= [];
53         tmp1(:,end,:)= [];
54         tmp1(1,:,:)= [];
55         tmp1(:,1,:)= [];
56
57         tmp1 = flipud(tmp1);
58         tmp1 = rot90(tmp1);
59         tmp1 = flipud(tmp1);
60
61         for i = p*50 +2 : (p+1)*50;
62             tmp = imread(sprintf(path,obj.CurrentRoundNmbr,i));
63             % crop the black pixels!
64             tmp(end,:,:)= [];
65             tmp(:,end,:)= [];
66             tmp(1,:,:)= [];
67             tmp(:,1,:)= [];
68
69             tmp = flipud(tmp);
70             tmp = rot90(tmp);
71             tmp = flipud(tmp);
72
73             tmp1 = cat(2,tmp1,tmp);
74         end
75         disp(p+1);
76         imwrite(tmp1,sprintf(path_,obj.CurrentRoundNmbr,p+1));
77     end
78 % - - - - -
79 % - - - AddRemainingImages - - -
80 % - - - - -
81 function AddRemainingImages(obj)
82     path = 'Round%d/rawdata/AIA%d.bmp'
83     path_ = 'Round%d/splicedImages/splicedImage%d.bmp'
84     splicedFullImages = floor(obj.NmbrOfImages/50);
85     rest = mod(obj.NmbrOfImages, 50);
86     if rest == 0
87         return;
88     end
89
90     tmp1 = ...
91         imread(sprintf(path,obj.CurrentRoundNmbr,splicedFullImages*50 ...
92                         +1), 'bmp');
```

```
92         % crop the black pixels!
93         tmp1(end,:,:)= [];
94         tmp1(:,end,:)= [];
95         tmp1(1,:,:)= [];
96         tmp1(:,1,:)= [];
97
98         tmp1= flipud(tmp1);
99         tmp1= rot90(tmp1);
100        tmp1= flipud(tmp1);
101
102        %for i = 10252 : 10285;
103        for i = splicedFullImages*50 +2 : obj.NmbrOfImages;
104            tmp = imread(sprintf(path,obj.CurrentRoundNmbr,i));
105            % crop the black pixels!
106            tmp(end,:,:)= [];
107            tmp(:,end,:)= [];
108            tmp(1,:,:)= [];
109            tmp(:,1,:)= [];
110
111            tmp = flipud(tmp);
112            tmp = rot90(tmp);
113            tmp = flipud(tmp);
114
115            tmp1 = cat(2,tmp1,tmp);
116
117        end
118        disp(splicedFullImages+1);
119        imwrite(tmp1,sprintf(path_,obj.CurrentRoundNmbr, ...
120                           splicedFullImages + 1));
121
122        % - - - - - - - - - - - - - - - - - -
123        % - - CONVERT TO GRayscale - - -
124        % - - - - - - - - - - - - - - - - - -
125        function ConvertToGrayScale(obj)
126            path = 'Round%d/rawdata/AIA%d.bmp'
127            for p = 1 : obj.NmbrOfSplicedImages;
128                img = imread(sprintf(path, obj.CurrentRoundNmbr, ...
129                               p), 'bmp');
130
131                grayImg = img(:,:,1) + img(:,:,2) + img(:,:,3);
132                grayImg = double(grayImg);
133                grayImg = grayImg / max(grayImg(:));
134                imgNmbrOfPixels = length(grayImg(:));
135                p
136                imwrite(grayImg,sprintf(path, obj.CurrentRoundNmbr, ...
137                                         p));
138
139        end
140
141        % - - - - - - - - - - - - - - - - - -
142        % - - - - - - - - - - - - - - - - - -
143        % - - - - - - - - - - - - - - - - - -
144        function ConvertToBWIImages (obj)
145            path = 'Round%d/splicedImages/splicedImage%d.bmp'
146            path_ = 'Round%d/BWIImages/BWImage%d.bmp'
```

```
143         for p = 1 : obj.NmbrOfSplicedImages;
144             img = imread(sprintf(path, obj.CurrentRoundNmbr, ...
145                         p), 'bmp');
146             bwImg = false(size(img));
147             bwImg(img > 250) = true;
148             p
149             imwrite(bwImg,sprintf(path_, obj.CurrentRoundNmbr, p));
150         end
151         % - - - - - -
152         % -- Get Number Of folders -- -
153         % - - - - - -
154         function GetNmbrOfFolders(obj)
155             D = dir(['.', '\Round*']);
156             obj.MaxFolderNmbr = length(D([D.isdir]));
157         end
158         % - - - - - -
159         % -- Get Number Of files -- -
160         % - - - - - -
161         function GetNmbrOfFile (obj)
162             path = 'Round%d/rawdata'
163             D = dir([sprintf(path, obj.CurrentRoundNmbr), '\*.bmp']);
164             obj.NmbrOfImages = length(D(not([D.isdir])));
165         end
166         % - - - - - -
167         % - Get Number Of spliced Img -
168         % - - - - - -
169         function GetNmbrOfSplicedImages (obj)
170             path = 'Round%d/splicedImages'
171             D = dir([sprintf(path, obj.CurrentRoundNmbr), '\*.bmp']);
172             obj.NmbrOfSplicedImages = length(D(not([D.isdir])));
173         end
174     end
175 end
```

C.2 SigExBot.m

```
1 % -----
2 % - - - SIG EX Bot CLASS - - -
3 %
4 classdef SigExBot < handle
5 properties
6     IsOnTrack;
7     FirstCycle;
8     TrackFollowing;
9     StepSize;
10    StartStepSize;
11    CurrentX;
12    CurrentY;
13    CurrentRoundNmbr;
14    ImgRGB = {};
15    Img = {};
16    ImgWidth;
17    ImgHeight;
18    PixelThreshold; % 0 ~ 255
19    CurrentImgNmbr;
20    MaxImgNmbr;
21    MaxImgNmbrFirstImgSet;
22    MaxRoundNmbr;
23    SignalIndex;
24    ProcessedSignal = [];
25    Signal = [];
26    DebugSignal1 = [];
27    SignalWidthArr = [];
28    MeanSignalWidth;
29    ChangeInSigWidth;
30    StrangeThingCounter;
31    Debug;
32    AlgoStopHeight;
33    CorVal;
34    CurrentCorVal;
35    GapFlag;
36    GapFillingFlag;
37    GapEndedCounter;
38    GapSlope;
39    StartOfGapX;
40    StartOfGapY;
41    StartOfGapIndex;
42    StartOfGapImgNmbr;
43    EndOfGapImgNmbr;
44    EndOfGapIndex;
45    EndOfGapY;
46    ReturnIndex;
47    ReturnX;
48    ReturnY;
49    TryToRecover;
50    TryToRecoverCounter;
51 end
```

```
52     methods
53         % -----
54         % - - - - - Constructor - - - -
55         % -----
56         function obj = SigExBot()
57             obj.TrackFollowing = false;
58             obj.FirstCycle = false;
59             % When the Bot is created, it isn't on track
60             obj.IsOnTrack = false;
61             obj.StartStepSize = 2.5;
62             obj.CurrentX = 1;
63             obj.CurrentY = 2400;
64             obj.ImgWidth = 0;
65             obj.ImgHeight = 0;
66             obj.PixelThreshold = 240;
67             obj.CurrentImgNmbr = 1;
68             obj.CurrentRoundNmbr = 1;
69             obj.StrangeThingCounter = 0;
70             obj.SignalIndex = 0;
71             obj.Signal(300000) = 0;
72             obj.ProcessedSignal(300000) = 0;
73             % Debug is On by default
74             obj.Debug = true;
75             % Algorithm stops when higher
76             % than 2000px at 1st image
77             obj.AlgoStopHeight = 2000;
78             % 1486px correction after one full Round
79             obj.CorVal = 1486;
80             obj.CurrentCorVal = 0;
81             obj.GapEndedCounter = 0;
82             obj.GapFlag = false;
83             obj.GapFillingFlag = false;
84             obj.TryToRecover = false;
85             obj.TryToRecoverCounter = 0;
86             obj.GetNmbrOfFolders();
87             obj.GetNmbrOfFiles();
88             obj.LoadImagesIntoRAM();
89             obj.ChangeCurrentImage(obj.CurrentImgNmbr);
90             obj.SetCurrentStepSize();
91         end
92         % -----
93         % - - Get Number Of folders - -
94         % -----
95         function GetNmbrOfFolders(obj)
96             D = dir(['.', '\Round*']);
97             obj.MaxRoundNmbr = length(D([D.isdir]));
98         end
99         % -----
100        % - - Get Number Of files - -
101        % -----
102        function GetNmbrOfFiles (obj)
103            D = dir([sprintf('Round%d/splicedImages', ...
104                obj.CurrentRoundNmbr), '\*.bmp']);
105            obj.MaxImgNmbr = length(D(not([D.isdir])))
```

```
105         if ~obj.TrackFollowing;
106             obj.MaxImgNmbrFirstImgSet = obj.MaxImgNmbr;
107         end
108     end
109 % ----- %
110 % -- Load Images into RAM --
111 % -----
112 function LoadImagesIntoRAM(obj)
113     for p = 1 : obj.MaxImgNmbr;
114         tmp = ...
115             imread(sprintf('Round%d/splicedImages/splicedImage%d.bmp', ...
116                             obj.CurrentRoundNmbr, p), 'bmp');
117         obj.Img(p) = {tmp};
118         [m n r] = size(tmp);
119         tmpRGB = uint8(zeros(m,n,3));
120         tmpRGB(:,:,1) = uint8(tmp(:,:,1));
121         tmpRGB(:,:,2) = tmpRGB(:,:,1);
122         tmpRGB(:,:,3) = tmpRGB(:,:,1);
123         obj.ImgRGB(p) = {tmpRGB};
124     end
125 % -----
126 % -- Set Current StepSize --
127 % -----
128 function SetCurrentStepSize(obj)
129     obj.StepSize = obj.StartStepSize * obj.MaxImgNmbr / ...
130                 obj.MaxImgNmbrFirstImgSet;
131 % -----
132 % -- Go Back Images --
133 % -----
134 function GoBackImages(obj, nmbr)
135     tmp = obj.CurrentImgNmbr + nmbr;
136     if tmp < 1;
137         obj.CurrentImgNmbr = obj.MaxImgNmbr + tmp;
138     else
139         obj.CurrentImgNmbr = tmp;
140     end
141     if obj.Debug & obj.TrackFollowing;
142         disp(sprintf('Going Back one Image to: %d Image: ... ...
143                                         %d', obj.CurrentRoundNmbr, obj.CurrentImgNmbr));
144     end
145     tmpSize = size(obj.Img{obj.CurrentImgNmbr});
146     obj.ImgHeight = tmpSize(1);
147     obj.ImgWidth = tmpSize(2);
148 % -----
149 % -- Change Current Image --
150 % -----
151 function ChangeCurrentImage(obj, nmbr)
152     if obj.Debug & obj.TrackFollowing;
153         disp(sprintf('processing Round: %d Image: %d', ...
154                         obj.CurrentRoundNmbr, obj.CurrentImgNmbr));
155     end
```

```
154         tmpSize = size(obj.Img{nmbr});
155         obj.ImgHeight = tmpSize(1);
156         obj.ImgWidth = tmpSize(2);
157         if (nmbr == 1 & (obj.ImgHeight - obj.AlgoStopHeight) > ...
158             obj.CurrentY & obj.TrackFollowing) | ...
159             (obj.CurrentRoundNmbr == obj.MaxRoundNmbr & ...
160              (obj.ImgHeight - obj.AlgoStopHeight) > ...
161              obj.CurrentY);
162             % even if there is a gap which
163             % the program tries to fill, stop it!
164             obj.GapFlag = false;
165             obj.SaveMarkedImages();
166             obj.CurrentRoundNmbr = obj.CurrentRoundNmbr + 1;
167             obj.CurrentCorVal = obj.CurrentCorVal - obj.CorVal;
168             obj.CurrentY = obj.CurrentY + obj.CorVal;
169             obj.GetNnbrOfFiles();
170             obj.LoadImagesIntoRAM();
171             obj.SetCurrentStepSize();
172             if obj.CurrentRoundNmbr > obj.MaxRoundNmbr;
173                 obj.TrackFollowing = false;
174             end
175         end
176         % - - - - -
177         % - - - Next Step - - - -
178         % - - - - -
179         function NextStep(obj)
180             obj.CurrentX = obj.CurrentX + obj.StepSize;
181             obj.SignalIndex = obj.SignalIndex + 1;
182             obj.Signal(obj.SignalIndex) = obj.CurrentY + ...
183                 obj.CurrentCorVal;
184             obj.DebugSignal1(obj.SignalIndex) = obj.ChangeInSigWidth;
185             if round(obj.CurrentX) > obj.ImgWidth;
186                 obj.CurrentX = 1;
187                 obj.CurrentImgNmbr = obj.CurrentImgNmbr + 1;
188                 if obj.CurrentImgNmbr > obj.MaxImgNmbr;
189                     obj.CurrentImgNmbr = 1;
190                 end
191                 obj.ChangeCurrentImage(obj.CurrentImgNmbr)
192             end
193         end
194         % - - - - -
195         % - - - Follow Track - - - -
196         % - - - - -
197         function FollowTrack(obj)
198             while obj.TrackFollowing;
199                 obj.CenterOnCurrentPos();
200                 obj.LeaveBlackMark();
201                 obj.NextStep();
202                 if obj.FirstCycle;
203                     obj.FirstCycle = false;
204                 end
205             end
206         end
207     end
```

```

204      % -----
205      % ----- Start -----
206      % -----
207      function Start(obj)
208          obj.TrackFollowing = true;
209          obj.FirstCycle = true;
210          if not(obj.IsOnTrack);
211              if not(obj.SearchClosestTrack());
212                  'could not find closest Track!'
213                  return
214              else
215                  obj.FollowTrack();
216              end
217          else
218              obj.FollowTrack();
219          end
220      end
221      % -----
222      % ----- LeaveBlackMark -----
223      % -----
224      function LeaveBlackMark(obj)
225          if obj.Debug;
226              if obj.GapFillingFlag
227                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
228                      round(obj.CurrentX), 1) = 0;
229                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
230                      round(obj.CurrentX), 2) = 0;
231                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
232                      round(obj.CurrentX), 3) = 255;
233              elseif obj.GapFlag
234                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
235                      round(obj.CurrentX), 1) = 255;
236                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
237                      round(obj.CurrentX), 2) = 0;
238                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
239                      round(obj.CurrentX), 3) = 0;
240              else
241                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
242                      round(obj.CurrentX), 1) = 0;
243                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
244                      round(obj.CurrentX), 2) = 255;
245                  obj.ImgRGB{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
246                      round(obj.CurrentX), 3) = 0;
247              end
248          end
249      end
250      % -----
251      % ----- Save Marked Images -----
252      % -----
253      function SaveMarkedImages(obj)
254          obj
255          if obj.Debug;
256              disp('saving marked images');
257              for p = 1 : obj.MaxImgNmbr;

```

```
249         imwrite(obj.ImgRGB{p}, ...
250             sprintf('Round%d/markedImages/splicedImage%d.bmp', ...
251                 obj.CurrentRoundNmbr, p));
252     end
253 end
254 % - - - - - - - - - - - - - - -
255 % - - Get Centering Correction - -
256 % - - - - - - - - - - - - - - -
257 function val = GetCenterOfTrack(obj)
258     % How far up can we go?
259     % - - - - - - - - - - - - - -
260     incrementerUp = 0;
261     while(obj.Img{obj.CurrentImgNmbr}(round(obj.CurrentY) + ...
262         incrementerUp, round(obj.CurrentX)) >= ...
263             obj.PixelThreshold)
264         incrementerUp = incrementerUp - 1;
265     end
266     % How far down can we go?
267     % - - - - - - - - - - - - - -
268     incrementerDown = 0;
269     while(obj.Img{obj.CurrentImgNmbr}(round(obj.CurrentY) + ...
270         incrementerDown, round(obj.CurrentX)) >= ...
271             obj.PixelThreshold)
272         incrementerDown = incrementerDown + 1;
273     end
274
275     signalWidth = incrementerDown - incrementerUp;
276     obj.CalcMeanSignalWidth(signalWidth);
277
278     val = (incrementerUp + incrementerDown)/2;
279 end
280 % - - - - - - - - - - - - - -
281 % - - Center On Current Pos - - -
282 % - - - - - - - - - - - - - -
283 function CenterOnCurrentPos(obj)
284     if obj.GapFillingFlag == true;
285         obj.GapFilling();
286     else
287         if obj.Img{obj.CurrentImgNmbr}(round(obj.CurrentY), ...
288             round(obj.CurrentX)) >= obj.PixelThreshold;
289             obj.StrangeThingCounter = 0;
290             % How far up can we go?
291             % - - - - - - - - - - - - - -
292             incrementerUp = 0;
293             while(obj.Img{obj.CurrentImgNmbr}(round(obj.CurrentY) ...
294                 + incrementerUp, round(obj.CurrentX)) >= ...
295                     obj.PixelThreshold)
296                 incrementerUp = incrementerUp - 1;
297             end
298             % How far down can we go?
299             % - - - - - - - - - - - - - -
300             incrementerDown = 0;
301             while(obj.Img{obj.CurrentImgNmbr}(round(obj.CurrentY) ...
```

```

    + incrementeDown, round(obj.CurrentX)) >= ...
    obj.PixelThreshold)
    incrementeDown = incrementeDown + 1;
end

297     signalWidth = incrementeDown - incrementeUp;
298     obj.CalcMeanSignalWidth(signalWidth);

299     changeInY = (incrementeUp + incrementeDown)/2;

300     % make it so the max chaneInY is 1
301     if abs(changeInY) >= 1;
302         changeInY = changeInY/abs(changeInY);
303     end

306     obj.ResetTryToRecoverFlag();

308     % This is quite important:
309     % Here we check wether or not the signal
310     % width changed more than x (in this case 5)
311     % if this is the case, we declare
312     % the signal from here on as "gap".
313     % after getting back on track,
314     % the gap will be filled with a linear slope.
315     if obj.ChangeInSigWidth >= 5 & ~obj.TryToRecover;
316         if ~obj.GapFlag
317             % The code here runs only
318             % one cycle after gap was detected
319             obj.GapFlag = true;
320             obj.StartOfGapX = obj.CurrentX -4 * ...
321                 obj.StepSize;
322             obj.StartOfGapIndex = obj.SignalIndex - 4;
323             if obj.StartOfGapX < 1;
324                 obj.StartOfGapX = obj.CurrentX;
325                 obj.StartOfGapIndex = obj.SignalIndex;
326             end
327             obj.StartOfGapY = ...
328                 obj.Signal(obj.StartOfGapIndex) - ...
329                     obj.CurrentCorVal;
330             obj.StartOfGapImgNmbr = obj.CurrentImgNmbr;
331         end
332         obj.GapEndedCounter = 0;
333         % as long as the change in signal width
334         % is bigger than x we go straight ahead.
335         changeInY = 0;
336     elseif obj.GapFlag;
337         % If we get in here, we are inside a gap,
338         % but the signal width has normalized to a ...
339             degree where
340             % Trackfollowing is possible again.
341             obj.GapEndedCounter = obj.GapEndedCounter + 1;
342             if obj.GapEndedCounter > 30;
343                 obj.GapFlag = false;
344                 obj.GapEndedCounter = 0;

```

```
342         obj.EndOfGapIndex = obj.SignalIndex - 26;
343         obj.ReturnIndex = obj.SignalIndex;
344         obj.ReturnX = obj.CurrentX;
345         obj.ReturnY = obj.CurrentY;
346         obj.EndOfGapY = ...
347             obj.Signal(obj.EndOfGapIndex) - ...
348                 obj.CurrentCorVal;
349             obj.CalculateSlope();
350             obj.GapFillingFlag = true;
351             obj.SignalIndex = obj.StartOfGapIndex;
352             obj.CurrentX = obj.StartOfGapX;
353             obj.CurrentY = obj.StartOfGapY;
354             obj.EndOfGapImgNmbr = obj.CurrentImgNmbr;
355             if ~(obj.StartOfGapImgNmbr == ...
356                 obj.CurrentImgNmbr);
357                 % Start and End on different Images
358                 obj.GoBackImages(obj.StartOfGapImgNmbr ...
359                                 - obj.CurrentImgNmbr);
360             end
361         end
362     end
363     obj.CurrentY = obj.CurrentY + changeInY;
364 else
365     % just go straight. Somethings strange happened!
366     % (We are in here because there is no pixel
367     % bright enough to do the centering on)
368     obj.StrangeThingCounter = ...
369         obj.StrangeThingCounter + 1;
370     if obj.StrangeThingCounter > 600;
371         % give we are already trying to recover and
372         % there is once again no bright pixelcluster
373         % to follow.
374         if obj.TryToRecover;
375             obj.SaveMarkedImages();
376             disp('Something strange happened!');
377             disp(obj.CurrentImgNmbr);
378             obj.TrackFollowing = false;
379         end
380         % after 600 steps of going blindly
381         % straight forward: Try to recover!
382         % We try to recover by setting the current Y
383         % value to the value 750 steps earlier
384         % (then everything should be still fine)
385         % Then we connect the X,Y point 750 steps ago
386         % with the current X and current (new) Y.
387         % We use a straight line for this.
388         obj.StrangeThingCounter = 0;
389         disp('Try to recover!');
390         obj.GapFlag = false;
391         obj.GapFillingFlag = true;
392         obj.TryToRecover = true;
393         obj.EndOfGapIndex = obj.SignalIndex;
394         obj.EndOfGapImgNmbr = obj.CurrentImgNmbr;
395         obj.ReturnIndex = obj.SignalIndex;
```

```
391         obj.ReturnX = obj.CurrentX;
392         obj.CurrentY = obj.SignalIndex(obj.SignalIndex - ...
393                                         750) - obj.CurrentCorVal;
394         obj.ReturnY = obj.CurrentY;
395         obj.EndOfGapY = obj.CurrentY;
396         obj.SignalIndex = obj.SignalIndex - 750;
397         obj.CurrentX = obj.CurrentX - 750 * ...
398                                         obj.StepSize;
399         obj.StartOfGapX = obj.CurrentX;
400         obj.StartOfGapY = obj.CurrentY;
401         obj.StartOfGapIndex = obj.SignalIndex;
402         obj.StartOfGapImgNmbr = obj.CurrentImgNmbr;
403         obj.CalculateSlope();
404         if obj.CurrentX < 1;
405             obj.GoBackImages(-1);
406             obj.StartOfGapImgNmbr = obj.CurrentImgNmbr;
407             obj.CurrentX = obj.ImgWidth + obj.CurrentX;
408         end
409     end
410 end
411 % - - - - -
412 % - - Reset Try To Recover Flag -
413 % - - - - -
414 function ResetTryToRecoverFlag(obj)
415     if obj.TryToRecover;
416         obj.TryToRecoverCounter = obj.TryToRecoverCounter + 1;
417         if obj.TryToRecoverCounter > 500;
418             obj.TryToRecoverCounter = 0;
419             obj.TryToRecover = false;
420         end
421     end
422 end
423 % - - - - -
424 % - - - GapFilling - - - -
425 % - - - - -
426 function GapFilling(obj)
427     if obj.SignalIndex < obj.EndOfGapIndex;
428         % do the gap filling!
429         obj.CurrentY = obj.StartOfGapY + (obj.SignalIndex - ...
430                                         obj.StartOfGapIndex) * obj.GapSlope;
431     else
432         % switch back to normal behavior
433         obj.GapFillingFlag = false;
434         obj.SignalIndex = obj.ReturnIndex;
435         obj.CurrentX = obj.ReturnX;
436         obj.CurrentY = obj.ReturnY;
437         obj.CurrentImgNmbr = obj.EndOfGapImgNmbr;
438         obj.ChangeCurrentImage(obj.CurrentImgNmbr);
439     end
440 end
441 % - - - - -
442 % - - - Calculate Slope - - -
```

```
442 % -----
443 function CalculateSlope(obj)
444     length = obj.EndOfGapIndex - obj.StartOfGapIndex;
445     height = obj.EndOfGapY - obj.StartOfGapY;
446     obj.GapSlope = height / length;
447 end
448 % -----
449 % -- Calc Mean Signal Width --
450 %
451 function CalcMeanSignalWidth(obj, sigWidth)
452     if obj.FirstCycle;
453         obj.MeanSignalWidth = sigWidth;
454         obj.ChangeInSigWidth = 0;
455     else
456         diff = sigWidth - obj.MeanSignalWidth;
457         if abs(diff) <= 18
458             obj.MeanSignalWidth = obj.MeanSignalWidth + ...
459                 diff/100;
460         end
461         obj.ChangeInSigWidth = abs(diff);
462     end
463 %
464 % -- Search closest Track --
465 %
466 function ret = SearchClosestTrack(obj)
467     for Ypixel = obj.ImgHeight : -1 : 1;
468         if obj.Img{obj.CurrentImgNmbr}(Ypixel, 1) >= ...
469             obj.PixelThreshold
470             if obj.PixelThreshold <= ...
471                 sum(sum(obj.Img{obj.CurrentImgNmbr}(Ypixel-15:Ypixel-5, ...
472                     1:100)) / 1000);
473                 obj.IsOnTrack = true;
474                 obj.CurrentY = Ypixel;
475                 ret = true; % Track found!
476                 return;
477             end
478         end
479     end
480     ret = false; % No track found!
481 end
482 %
483 % -- Process Signal --
484 %
485 function ProcessSignal(obj)
486     for entry = obj.Signal;
487         entry
488     end
489 end
490 end
491 end
```

C.3 Needle.m

```
1 % -----
2 % ----- Needle CLASS -----
3 %
4 classdef Needle < handle
5     properties
6         Signal;
7         TrendCorrectedSignal;
8         FFTOutput;
9         SmoothingOutput;
10        NeedleSimOut;
11        SignalLength;
12        Yn;
13        X;
14        SamplingRate;
15        Drag;
16        SignalLengthInSecs;
17        xTime;
18        %NeedleSimualtion properties
19        Speed;
20        Mass;
21        Acceleration;
22        NeedleForce;
23        SpeedResForce
24        SpringResForce
25    end
26    methods
27        %
28        % ----- Constructor -----
29        %
30        function obj = Needle(signal)
31            obj.Signal = signal;
32            obj.SignalLength = length(obj.Signal);
33            obj.Yn = [];
34            obj.Init();
35        end
36        %
37        % ----- Init -----
38        %
39        function Init(obj)
40            obj.Drag = 0.0001;
41            obj.X = linspace(1, obj.SignalLength, obj.SignalLength);
42            obj.SamplingRate = 180000;
43            obj.SignalLengthInSecs = obj.SignalLength / ...
44                obj.SamplingRate;
45            obj.xTime = linspace(0, obj.SignalLengthInSecs, ...
46                obj.SignalLength);
47        end
48        %
49        % ----- Start -----
50        %
51        function Start(obj)
```

```

50         obj.TrendCorrectSignal();
51         obj.FFT();
52         obj.Crop();
53         obj.AdjustAmplitude();
54         obj.NeedleSimulation();
55         obj.Plot();
56         obj.SaveWAV();
57     end
58 % - - - - - - - - - - - - - - -
59 % - - - NeedleSimulation - - - -
60 % - - - - - - - - - - - - - - -
61 function NeedleSimulation(obj)
62     obj.Acceleration = 0;
63     obj.Speed = 0;
64     obj.Mass = 2;
65     % Here we make another array
66     % the same size as the input signal
67     obj.NeedleSimOut(length(obj.FFTOutput)) = 0;
68     obj.NeedleSimOut(1) = obj.FFTOutput(1);
69     for i = 2:length(obj.FFTOutput);
70         % compute all the forces
71         % for the current step
72         obj.NeedleForce = obj.FFTOutput(i) - ...
73             obj.NeedleSimOut(i-1);
74         obj.SpeedResForce = -obj.Speed*1.0;
75         obj.SpringResForce = -obj.NeedleSimOut(i-1) * 0.1;
76
77         % a = (F + SpeedRes + SpringRes)/m
78         obj.Acceleration = (obj.NeedleForce + ...
79             obj.SpeedResForce + obj.SpringResForce)/obj.Mass;
80
81         % 0.01 because we don't want to add
82         % the full acceleration every time.
83         obj.Speed = obj.Speed + obj.Acceleration * 0.8;
84         obj.NeedleSimOut(i) = obj.NeedleSimOut(i-1) + ...
85             obj.Speed*0.09;
86
87         % First thing: The Force which tries
88         % to move the needle towards the Signal!
89         % -SignalForce
90
91         % Then: All the Forces which work
92         % against the first Force!
93         % -SpeedResistanceForce (the greater
94         % the moving speed the greater this Force)
95
96         % -SpringForce (tries to move
97         % needle back to center position)
98     end
99 end
% - - - - - - - - - - - - - - -
% - - - TrendCorrectSignal - - -
% - - - - - - - - - - - - - - -
100 function TrendCorrectSignal(obj)

```

```

101         p = polyfit(obj.x, obj.Signal, 1);
102         obj.TrendCorrectedSignal = obj.Signal - polyval(p, obj.x);
103     end
104     % -----
105     % ----- SmoothingSignal -----
106     % -----
107     function SmoothingSignal(obj)
108         % Here we make another array the
109         % same size as the input signal
110         obj.Yn(length(obj.FFTOutput)) = 0;
111         obj.Yn(1) = obj.FFTOutput(1);
112         for i = 2:length(obj.FFTOutput);
113             diff = obj.FFTOutput(i) - obj.Yn(i-1);
114             obj.Yn(i) = obj.Yn(i-1) + obj.Drag * diff;
115         end
116         obj.SmoothingOutput = obj.FFTOutput - obj.Yn;
117     end
118     % -----
119     % ----- Plot -----
120     % -----
121     function Plot(obj)
122         figure('Name', 'FFTOutput + NeedleSimOut');
123         plot(obj.xTime, obj.FFTOutput);
124         hold on;
125         plot(obj.xTime, obj.NeedleSimOut);
126         %figure('Name', 'FFTOutput secs1');
127         %plot(obj.xTime, obj.FFTOutput);
128         %figure('Name', 'FFTOutput secs2');
129         %plot(obj.xTime, obj.FFTOutput);
130         %figure('Name', 'FFTOutput datapoints');
131         %plot(obj.FFTOutput);
132         %figure('Name', 'original Signal plot');
133         %plot(obj.Signal);
134     end
135     % -----
136     % ----- Adjust Amplitude -----
137     % -----
138     function AdjustAmplitude(obj)
139         obj.FFTOutput = obj.FFTOutput/ max(abs(obj.FFTOutput));
140     end
141     % -----
142     % ----- FFT -----
143     % -----
144     function FFT(obj)
145         %obj.TrendCorrectedSignal(1) = ...
146         %    obj.TrendCorrectedSignal(length(obj.TrendCorrectedSignal));
147         [obj.FFTOutput, f, y, y2] = fftf(obj.xTime, ...
148             obj.TrendCorrectedSignal, 15000, 100); %15000,100
149     end
150     % -----
151     % ----- Crop -----
152     % -----
153     function Crop(obj)
154         % Crop the first and last few

```

```

153         % entries of the FFTOutput
154         obj.FFTOutput(1:10000) = [];
155         obj.xTime(1:10000) = [];
156         len = length(obj.FFTOutput);
157         obj.FFTOutput(len-10000:len) = [];
158         obj.xTime(len-10000:len) = [];
159     end
160     % - - - - - - - - - - - - - - -
161     % - - - - - Save WAV - - - -
162     % - - - - - - - - - - - - - -
163     function SaveWAV(obj)
164         audiowrite('needleOutput.wav', obj.FFTOutput, ...
165                     obj.SamplingRate);
166         audiowrite('needleOutputSimuNeedle.wav', ...
167                     obj.NeedleSimOut, obj.SamplingRate);
168     end
169 end

```

C.4 fftf.m

```

1 function [X, f, y, y2] = fftf(t, x, varargin)
2 % fftf - fft filter;
3 % [X, f, y, y2] = fftf(t, x); with t the time vector and x the ...
4 % signal,
5 % displays the original signal, the Fourier transform (absolute ...
6 % values)
7 % and the reconstructed signal generated by the inverse transform ...
8 % ifft
9 % with a selected subset of the frequencies.
10 % By default, the frequencies in the filtered signal are cut at ...
11 % 1/8 the
12 % sampling frequency.
13 % The function returns X - reconstructed signal, f - vector of ...
14 % frequencies, y -
15 % full vector of amplitudes, y2 - the filtered vector of amplitudes.
16 % [X, f, y, y2] = fftf(t, x, cutoff); the user may set the cutoff
17 % frequency in units of Hertz.
18 % [X, f, y, y2] = fftf(t, x, cutoff, my_N); the user may select my_N
19 % amplitudes with highest absolute value to participate in the
20 % reconstruction.
21 % Examples
22 % (suppose you have t and x defined already, both 1-dimensional ...
23 % vectors of the same length.)
24 % fftf(t, x, 1e6); - reconstruct the signal with frequencies ...
25 % lower or
26 % equal to cutoff value of 1MHz.
27 % fftf(t, x, [], 20); - use only 20 biggest amplitudes for
28 % reconstruction.
29 % fftf(t, x, 1e6, 20); - select 20 biggest amplitudes within cutoff.
30 % Shmuel Ben-Ezra, Ultrashape ltd. August 2009
31
32 %% Verifying input

```

```
26 if ~any(size(t)==1),
27     disp('Unexpected vector size! - should be 1D vectors.')
28     return
29 end
30 if ~any(size(x)==1),
31     disp('Unexpected vector size! - should be 1D vectors.')
32     return
33 end
34 if length(t) ~=length(x),
35     disp('Unexpected vector size! - should be same length.')
36     return
37 end
38 %% Definitions
39 Fs=1/(t(2)-t(1)); %sampling freq
40 N=length(x);
41 Nfft=2^nextpow2(N);
42 f=Fs/2*linspace(0,1,1+Nfft/2); % create freqs vector
43 cutoff_freq=Fs/8;
44 cutoff_freqLow = 0;
45 my_freqs=[];
46 if nargin>2,
47     cutoff_freq=varargin{1};
48 end
49 if nargin>3,
50     cutoff_freqLow=varargin{2};
51 end
52 if nargin>4,
53     my_freqs=varargin{3};
54 end
55 %% main
56 y=fft(x,Nfft)/N; % perform fft transform
57 y2=filterfft(f, y, cutoff_freq, my_freqs, cutoff_freqLow); % filter ...
      amplitudes
58 %X=ifft(y2,'symmetric'); % the inverse transform. 'symmetric' is ...
      not recognized in older versions of matlab
59 X=ifft(y2); % inverse transform
60 X=X(1:N)/max(X);
61 ind1 = find(y2(1:1+Nfft/2)); % get the nonzero elements in y2
62 nfl = length(ind1); % count nonzero elements
63 %% display
64 figname = 'fftf - FFT at work';
65 ifig = findobj('type', 'figure', 'name', figname);
66 if isempty(ifig),
67     ifig = figure('name', figname); % on my machine: ..., ...
      'position', [360    120    600    800]);
68 end
69 figure(ifig);
70 % first plot
71 subplot(3,1,1)
72 plot(t*1e6,x)
73 xlabel('uSec')
74 axis tight
75 title('Original signal')
76 %second plot
```

```
77 subplot(3,1,2)
78 yplot=abs(y(1:1+Nfft/2));
79 yplot=yplot/max(yplot);
80 semilogy(f*1e-6, yplot, f(ind1)*1e-6, yplot(ind1), '.r');
81 xlabel('MHz')
82 title('Amplitudes')
83 legend('full spectrum', 'selected frequencies')
84 % third plot
85 subplot(3,1,3)
86 plot(t*1e6,X)
87 xlabel('uSec')
88 if isempty(cutoff_freq),
89     scutoff='No cutoff.';
90 else
91     scutoff=sprintf('Cutoff = %g [Mhz]', cutoff_freq/1e6);
92 end
93 stitle3=sprintf('Reconstructed signal with %d selected frequencies; ...
94 %s', nf1, scutoff);
95 title(stitle3)
96 axis tight
97 return
98
98 function y2=filterfft(f, y, cutoff, wins, cutoffLow)
99 nf=length(f);
100 ny=length(y);
101 if ~(ny/2+1 == nf),
102     disp('unexpected dimensions of input vectors!')
103     y2=-1;
104     return
105 end
106
107 % cutoff filter
108 y2=zeros(1,ny);
109 if ~isempty(cutoff) && (cutoffLow > 0);
110     ind1=find(f<=cutoff & f>=cutoffLow);
111     %display(ind1);
112     y2(ind1) = y(ind1); % insert required elements
113 else
114     y2=y;
115 end
116
117 % dominant freqs filter
118 if ~isempty(wins),
119     temp=abs(y2(1:nf));
120     y2=zeros(1,ny);
121     for k=1:wins, % number of freqs that I want
122         [tmax, tmaxi]=max(temp);
123         y2(tmaxi) = y(tmaxi); % insert required element
124         temp(tmaxi)=0; % eliminate candidate from list
125     end
126 end
127
128 % create a conjugate symmetric vector of amplitudes
129 for k=nf+1:ny,
```

```
130      y2(k) = conj(y2(mod(ny-k+1,ny)+1)); % formula from the help of ifft  
131 end  
132 return
```