**前処理メモ**

1. **trimming.pyで****全体画像からカット画像を作成する**

図 2　カット画像

図 1　全体画像



**②正解画像を参考に，カット画像に対して石灰化領域のアノテーションを行う**

Easy,normal,hardは全体画像，CDなどはカット画像を使用

MATLABを起動

↓

上の「開く」を押して「shori.mlx」を開く

↓

セクションの実行で最初のセクションを実行

(imageFolder=fulfile("")にラベル付けしたいファイルが入ったディレクトリを指定)

↓

セッションの続きから始める時は「ラベル」

↓

「セッションを開く」

↓

「セッションを保存しますか」で「いいえ」

↓

前回保存した.matファイルを指定

セッションを新しく始める時は

「ラベル」

↓

「セッションの保存」名前を付けて保存で新しく.matファイルが作成される

**ラベル付けの方法**

左にあるROIラベルの「ラベル」をクリック

ラベル名を適当につける．例えば「label1」など(番号順にラベル付け)

その横のモードは「Pixel label」にする

色はカスタムでR:0 G:1 B:0にする

↓

作成すると左に「label1」と出てくるのでクリック

↓

上のバーの「ブラシ」をクリックし「ブラシサイズを」最小にする

ラベルの不透明度は最大にすると見やすい

↓

正解画像を見ながら石灰化領域を塗る

Ctrl+マウスホイールで画像の拡大，右上の手のマークで画像を移動できる

こまめに(3枚~4枚ごとに)「セッションの保存」から保存することをおすすめ

たまに保存時に失敗し,保存できなかったことがあったため

↓

塗ったところを消すときは「消去」で「ブラシサイズ」を調節　最大:大きく消せる　最小:小さく消せる

**ラベル付けが終わった後**

左上の「ラベル」をクリック

↓

上のバーの一番右の「エクスポート」をクリック

↓

「ファイルへ」

↓

「OK」を押すとgTruth.matとPixelLabelDataディレクトリができる

↓

PixelLabelDataディレクトリ内にpngファイルが保存される

このままだとラベルの輝度値が1なので見えない(図3)

↓

shori.mlxの2個目のセクションを実行

↓

imwrite(img,strcat(””,name))の””内のディレクトリに

輝度値を255倍した画像(図4)が保存される

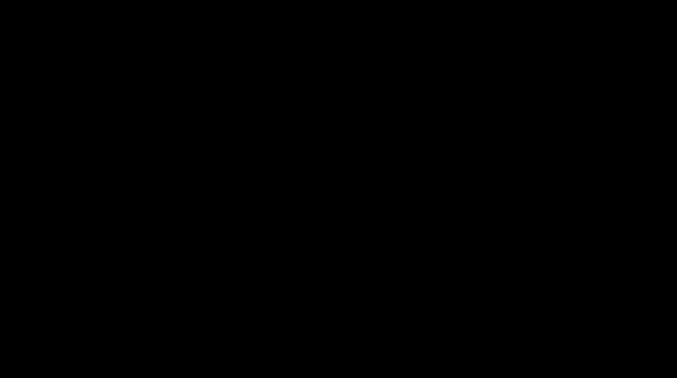
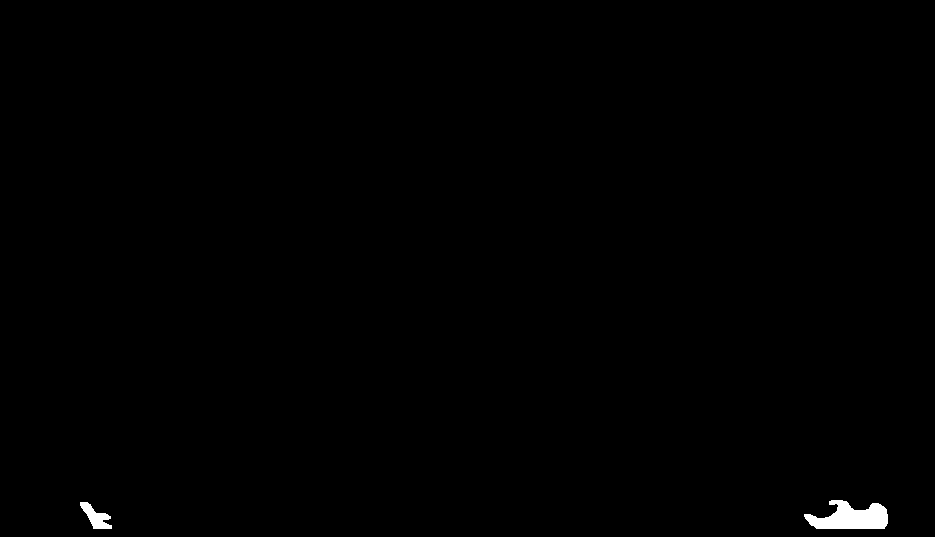


図 4　セクション2実行後

図 3　セクション2実行前

**③maeshori.pyで下1/4を切り出す前処理**

適当な名前のディレクトリを作成し(例えばeasyデータなど)

その中に「image」と「mask」の2つのディレクトリを作成

「image」ディレクトリ内にはカット画像

「mask」ディレクトリ内には白黒の2値ラベル画像(図4の画像)をまとめる

この時，画像のファイル名はimageとmaskで同じにする(後のプログラムのため)

hardから作成したものは「hard1」,CDから作成したものは「CD8\_0111841」のように

↓

maeshori.pyを実行

前処理1ディレクトリの「image」内にカット画像の下1/4の画像

「mask」内にラベル画像の下1/4の画像が保存される



図 5　カット画像の下1/4の画像

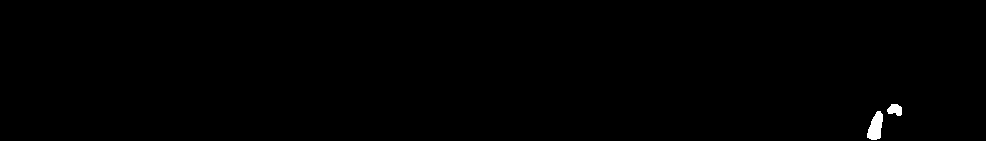


図 6　ラベル画像の下1/4の画像

**④maeshori2.pyで左右に分割を行う前処理**

前処理1の結果に対して実行する

前処理2ディレクトリの「image」に図5(カット画像下1/4)の左右分割後の画像

「mask」に図6(ラベル画像下1/4)の左右分割後の画像が保存される

img1 = cv2.resize(img1, (416, 416))  
img2 = cv2.resize(img2, (416, 416))  
mask1 = cv2.resize(mask1, (416, 416))   
mask2 = cv2.resize(mask2, (416, 416))

maeshori2.pyの上のコード部分をコメントアウトしない

→左右分割し，リサイズ済みの画像(図7)が得られる

コメントアウトする

→左右分割のみを行った画像(図8)が得られる



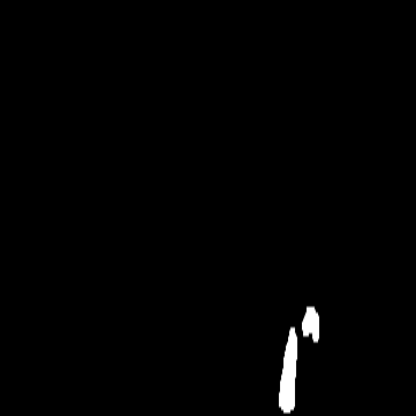
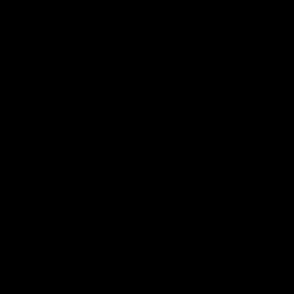


図 7　左右分割後の画像

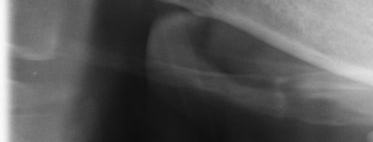


図 8

⑤**データセットの作成**

前処理が終わった画像を用いてデータセットを作成する

「images」に左右分割後の画像「masks」に左右分割後のマスク画像を全て入れる

split.pyを実行

TrainDataset(学習用)、TestDataset(テスト用)、ValDataset(検証用)にデータが分けられて，データセットが作成される

sekkai\_image\_toridasi.pyを実行すると

TrainDataset，TestDataset, ValDatasetから石灰化が存在しない画像を除いたsekkai\_TrainDataset, sekkai\_TestDataset, sekkai\_ValDatasetが作成される

**Memo**

1. maeshori(kenzyousya).pyは正解画像に石灰化領域がない時に

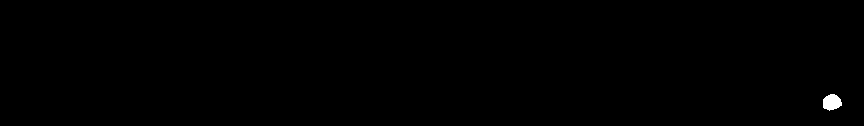
真っ黒のmask画像を作成するためのプログラム (CD12の健常者データに用いる)

~~2. maeshori3.pyは正解画像の塗りつぶし(図9上)からマスク画像(図9下)を~~

~~作成するためのプログラム~~(使わない)



↓



~~図 9　maeshori3.pyで作成される画像~~

3. annotation.pyはカット画像と白黒の2値ラベル画像から

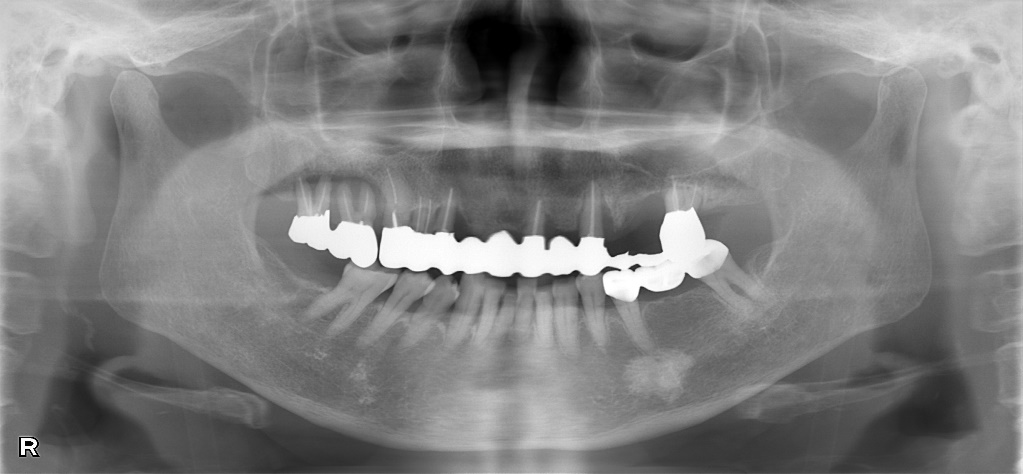
図10のような画像を作成するためのプログラム

歯科大の先生に確認してもらうとき時に正解画像と一緒に提出する



図 10　anootation.pyで作成される画像

4. R\_remove.pyは「2020年石灰化マーク」と「2021年石灰化マーク」の画像(図11上)から「R」を消去した画像(図11下)を作成するためのプログラム



↓



図 11　 R\_remove.pyで作成される画像

5. image\_rekka\_hantei.pyはR\_remove.pyを用いて「R」を消した画像について，

本当に「R」部分のみ消去できたかを判定するためのプログラム

(「R」の周りの画素値が劣化している可能性があるため)

6. green\_label\_to\_mask.pyはeasyとnormalの緑色の正解画像から，

もとのオリジナルの画像の大きさの白黒のマスク画像(図12)に変換するためのプログラム

これを実行すると，easy,normalについてhardとCDと同じ手順で

前処理(③以降)ができるようになる

imageとmaskで画像サイズが異なると，学習時にその画像が除外されてしまうため，画像サイズをそろえるために使用する



↓

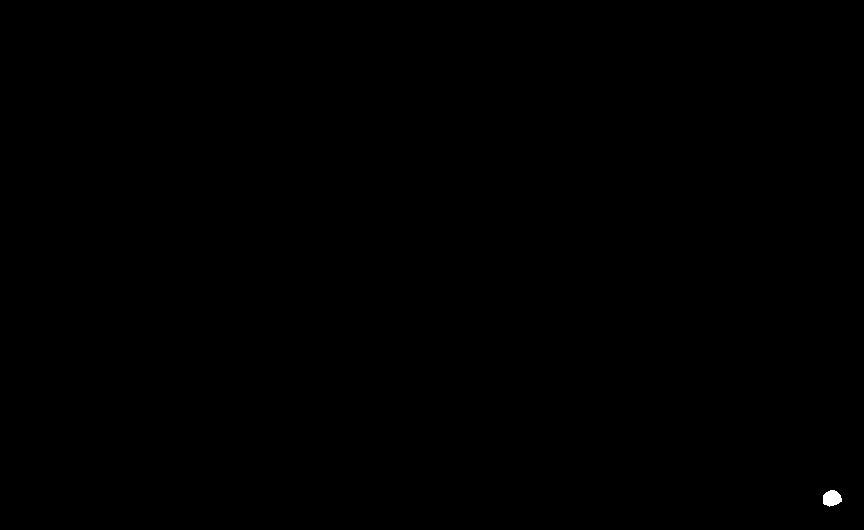


図 12 　green\_label\_to\_mask.pyで作成される画像