画像処理

第6回
2値画像処理・複数画像の利用

2値化処理

濃淡（グレースケール）画像やカラー画像から，
適当な条件を元に2値画像を生成する処理．

||

白または黒のみの画素を持つ画像

グレースケールとは違い灰色は存在しない

カラー画像

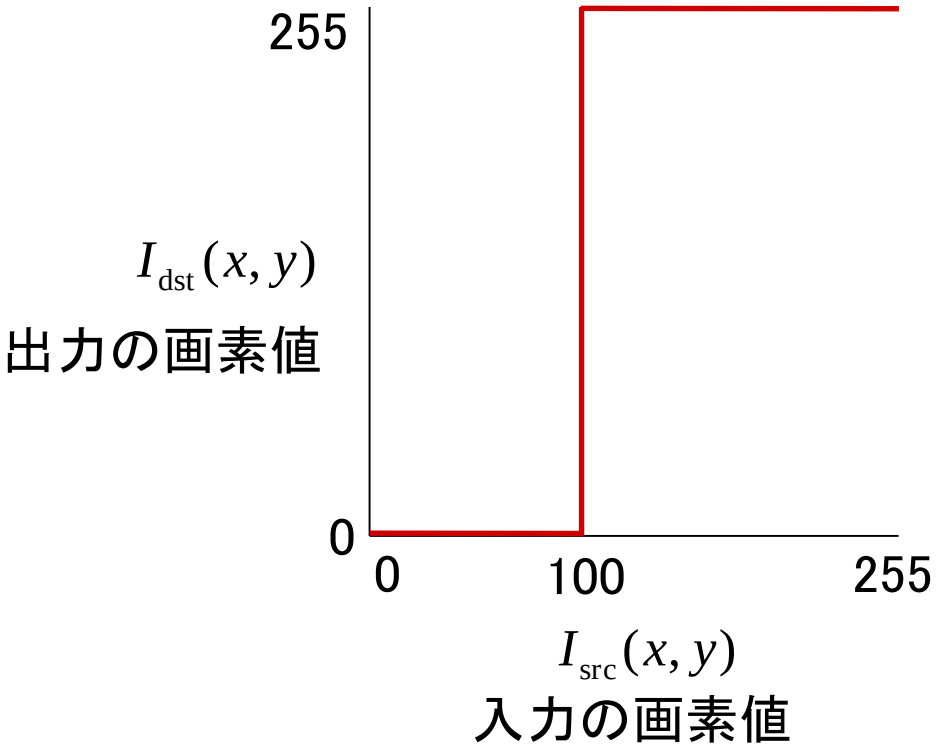
グレースケール画像

2値画像

二値化処理

二値化処理の単純な方法として，閾値処理がある．
（thresholding）

$$I_{\mathrm{dst}}(x,y)=\begin{cases}1 & (I_{\mathrm{src}}(x,y) > \text{閾値}) \\ 0 & (I_{\mathrm{src}}(x,y) \leq \text{閾値})\end{cases}$$



演習：閾値処理による二値化

06-01_threshold.py

入力画像

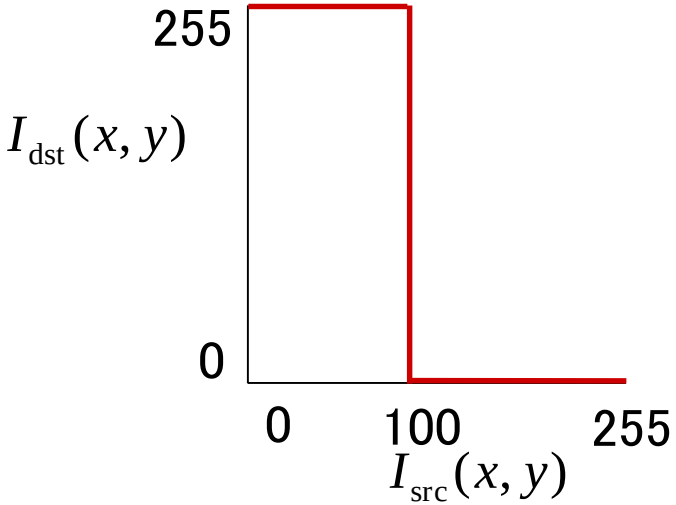
出力された
二値画像

（閾値＝100）

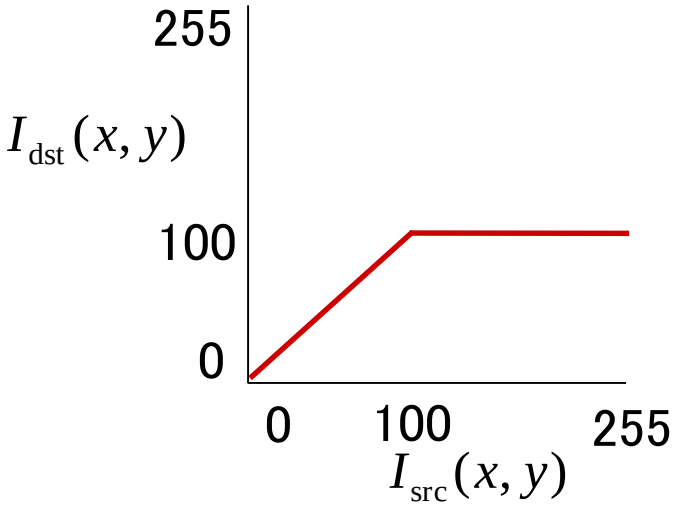
（閾値＝200）

プログラムの閾値を変化させ，
その振舞いを確認してみよう．

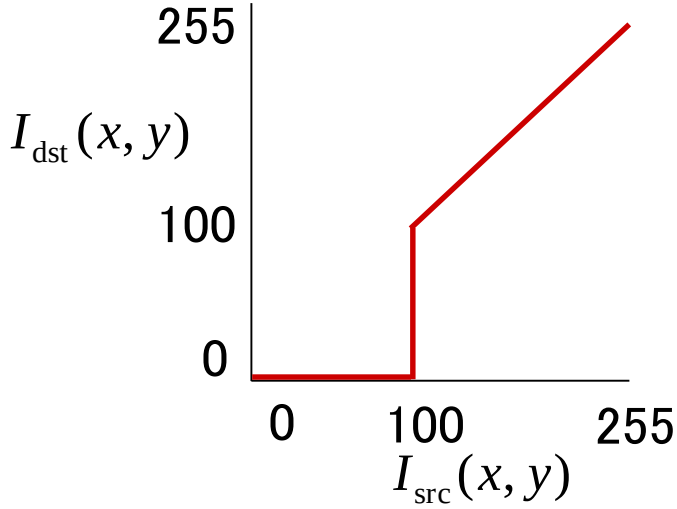
様々なトーンカーブ



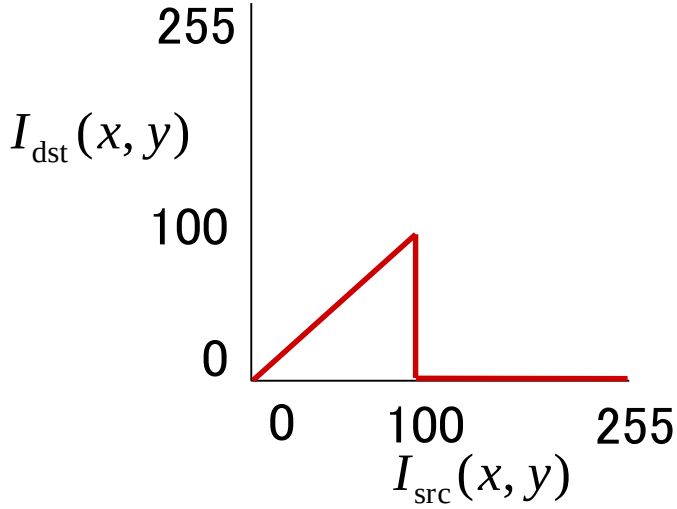
THRESH_BINARY_INV



THRESH_TRUNC



THRESH_TOZERO



THRESH_TOZERO_INV

演習：様々なトーンカーブ

06-01_threshold.py

入力画像

出力された二値画像
（THRESH_TOZERO）
（閾値＝100）

[課題6-1]

06-01_threshold.py のトーンカーブや閾値を変化させ，
それぞれのトーンカーブの特性について報告せよ．

※wordファイルに画像を貼り，そのときの閾値を示し，考察を含めること．

マスク処理

マスク処理（masking）

不必要とする部分を完全に消去し，必要な領域のみを抽出する処理で，クロマキー合成（chroma keying）などに用いられる．

||

ある色を透過色に設定し，画像の必要な部分のみ抽出し，複数の画像を合成する．

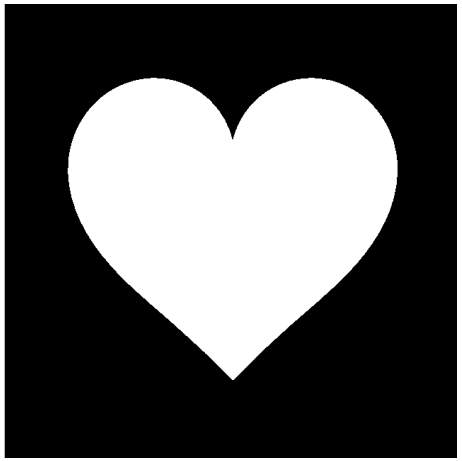
http://www.isp.co.jp/products/robuskey/movie/clip_alpha.html

演習：マスク処理

06-02_mask.py

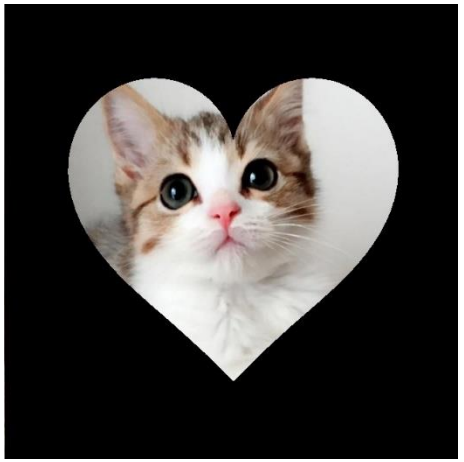
マスク画像の白部分のみを入力
画像から抜き出した結果になる

元画像



マスク画像

入力画像



出力画像

（THRESH_TOZERO）
（閾値＝100）

[練習]

マスク画像を自分で作成し，入力画像にマスクをかけてみよう.

（注意）入力画像サイズは同じにすること

膨張・収縮処理

自然な風景などを撮影した画像は濃淡値が一様でなく，ムラがあるため，閾値処理による2値化処理を行った場合，得られる2値画像には多くのノイズが入ってしまう．

（例）

- 多くの微小な穴が空く
- 複数に分断されて連続性が失われる
- 不必要な部分が微小な独立点として残る

このような二値画像に対しては，膨張・収縮処理を施すことで，ノイズ除去することが多い．

膨張・収縮処理

膨張処理（dilation）

図形を外側に1画素分広げる処理.

⇨ 微小な孔を塞ぐことが可能.

収縮処理　（erosion）

図形を内側に1画素分狭める処理.

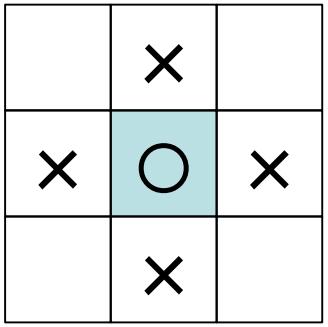
⇨ 独立点や突起を除去できる.

入力画像

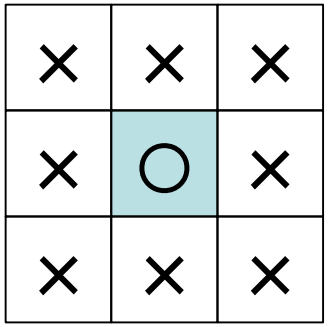
膨張処理後の結果

収縮処理後の結果

膨張・収縮処理



4近傍



8近傍

注目画素（中央の○）と近傍（×）

膨張処理

注目画素またはその近傍に白画素があれば，
注目画素を白にする．

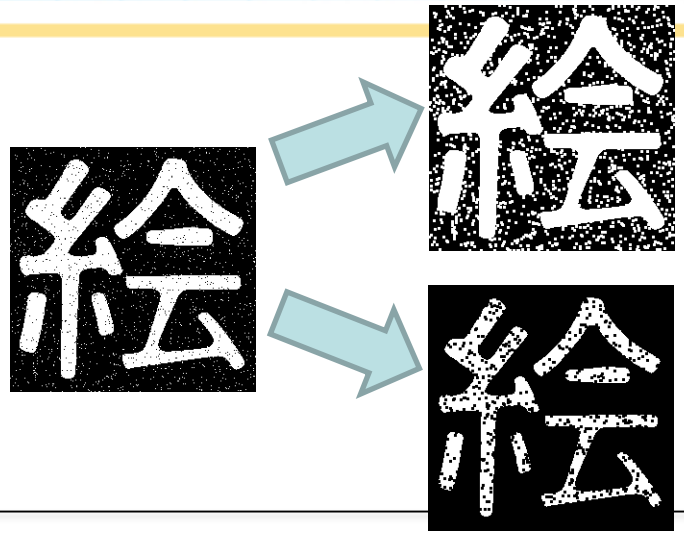
収縮処理

注目画素またはその近傍に黒画素があれば，
注目画素を黒にする．

演習：膨張・収縮処理

06-03_dilation.py
（膨張処理）

06-04_erosion.py
（収縮処理）



```
# 8近傍
neiborhood8 = np.array([[1, 1, 1],
                        [1, 1, 1],
                        [1, 1, 1]], np.uint8)

# 膨張
img_dilation = cv2.dilate(img_src, neiborhood8, iterations=1)
# 収縮
img_erosion = cv2.erode(img_src, neiborhood8, iterations=1)
```

オープニング・クロージング

膨張・縮小処理だけでは，一方のノイズが除去できるものの，
もう一方のノイズが増大してしまうこともある．

（解決策）

- **クロージング**（closing）
  膨張をn回実施した後，収縮をn回実施する．
  小さな穴をふさぎ，分断された連結要素を接続することができる

- **オープニング**（opening）
  収縮をn回実施した後，膨張をn回実施する．
  小さなノイズを取り除くことができる

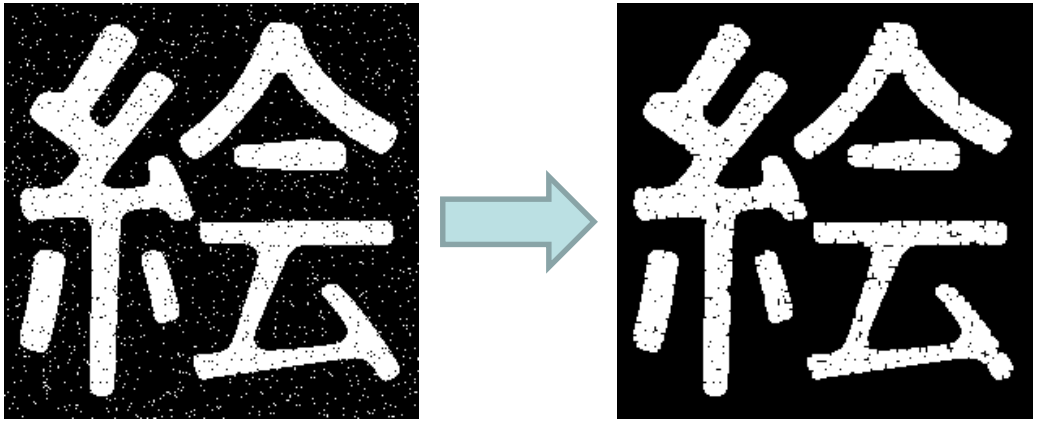
（注意点）
処理回数nを大きくすることで、より大きな穴をふさいだり，よりおおきなノイズを取り除いたりできるが，次第に元の形状が失われてしまう欠点もある．

課題：オープニング・クロージング

[課題6-2]

膨張・収縮のプログラムを参考にしながら，
オープニングまたはクロージングのプログラムを作成し，
e-noise.png のノイズを除去せよ．

（ソースコードと画像ファイルをwordに貼って提出すること）



できる限りノイズを
取り除く！

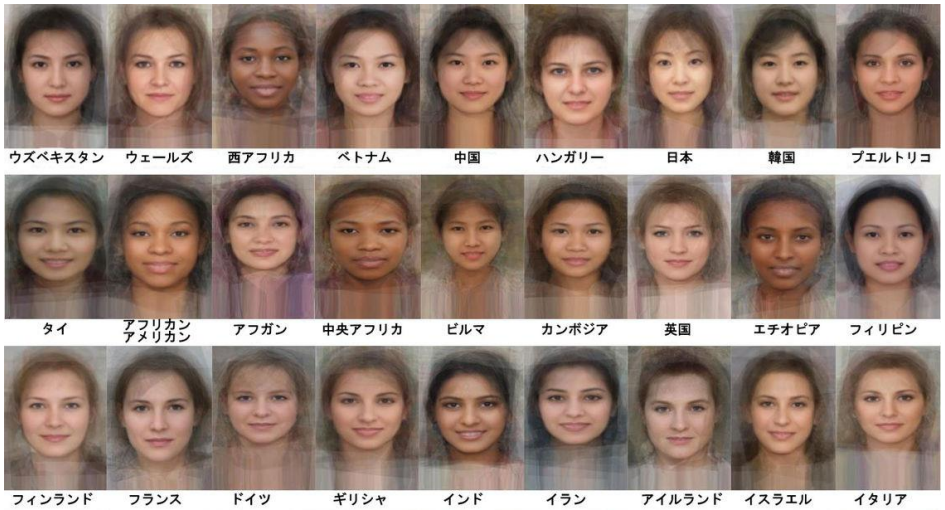
複数画像の利用

画像間演算

2枚またはそれ以上の画像を入力して，それぞれの画像の同じ位置にある画素ごとに，ある決められた演算を行い，出力値を決定する処理.

- 四則演算などの算術演算
- 論理積，論理和などの論理演算

が用いられる



世界各国別平均的な女性の顔

アルファブレンディング

2つの入力画像の平均値を計算して出力する.

2枚の入力の画素値を $I_1(x,y)$, $I_2(x,y)$ とすると,

2枚の入力画像の平均画像の画素値 $I_G(x,y)$ は

$$I_G(x,y)=\frac{I_1(x,y)+I_2(x,y)}{2}$$

単に平均値を計算するのではなく, 重み付き平均値を取り,

$$I_G(x,y)=\alpha\cdot I_1(x,y)+(1-\alpha)\cdot I_2(x,y)$$

としたものを, アルファブレンディングと呼ぶ.

演習：アルファブレンディング

06-05_blending.py

```
img_dst = img_apple * 0.5 + img_orange * 0.5
```

または

```
img_dst = cv2.addWeighted(img_apple, 0.5, img_orange, 0.5, 0.0)
```

課題：アルファブレンディング

[課題6-3]

αの値を徐々に変化させることにより，
下記のような図を作成せよ．

（ヒント）

- 処理しやすいように，入力画像の横サイズは256ピクセルにしてある．
- for文の中で，まずα（0～1）を求めてみよう．

背景差分

異なる2つの時刻で撮影された2枚の画像の差を観察すれば，
画像内で発生している変化情報を得ることができる．

背景画像

観測画像

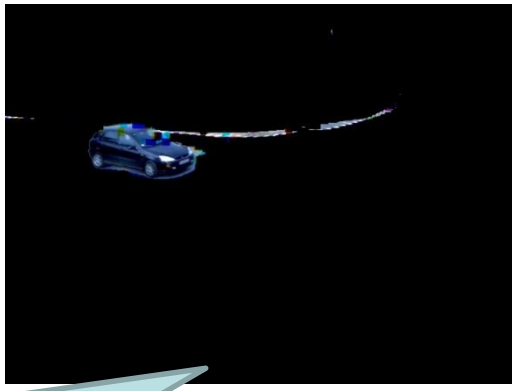
差分画像

演習：背景差分

06-06_bg_subtraction.py

背景画像
(00000050.jpg)

観測画像
(00000065.jpg)



背景差分処理により
車が抽出されることを確認

[練習]
背景画像(00000050.jpg)と観測画像(00000150.jpg)との
差分も調べてみよう.

演習：背景差分

動画＝連続した静止画
（フレーム）

06-07_video.py

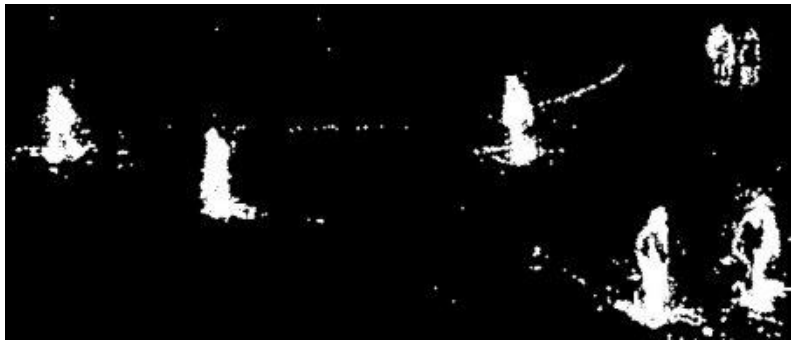
動画の読み込み

```
cap = cv2.VideoCapture('vtest.avi')

while(cap.isOpened()):
    # フレームを取得
    ret, frame = cap.read()

    cv2.imshow('Frame', frame)

    # 100ミリ待つ, 'q'で終了
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break
```

演習：背景差分

06-08_bg_subtraction.py

統計的な背景の推定法と画素単位でのベイス推定に基づく
領域分割を組み合わせたアルゴリズムによる背景差分



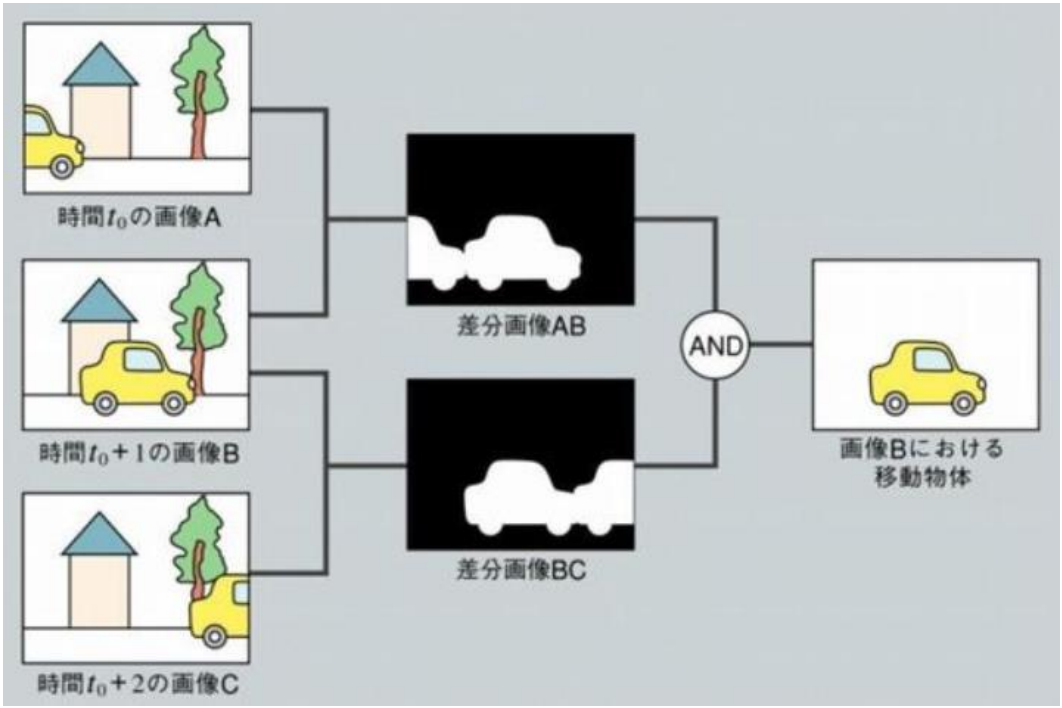
背景のモデル構築に最初の数フレーム(デフォルトで120)を使う
※ 表示されるまでに多少時間がかかる

（参考）http://lang.sist.chukyo-u.ac.jp/classes/OpenCV/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html

フレーム間差分

移動物体がないという理想的な背景画像が得られないことがある.

連続する3枚の画像間の差分を利用し,
移動物体領域を抽出できる.



今後の予定 （変更の可能性あり）

- 第7回　画像認識の概要
- 第8回　顔検出
- 第9回　画像のマッチング
- 第10回　一般画像認識
- 第11回　演習
- 第12回　ディープラーニング（1）
- 第13回　ディープラーニング（2）
- 第14回　発表（1）
- 第15回　発表（2）