クレジット：
Mathematics and Informatics Center メディアプログラミング入門 2020 山肩洋子

東京大学
THE UNIVERSITY OF TOKYO

CC BY NC ND

メディアプログラミング入門

第5回：画像の仕組みと画像処理

火５@本郷 2020年6月30日
情報理工学系研究科 数理・情報教育研究センター
准教授 山肩 洋子

第4回の課題：
宮沢賢治の小説を詳しく分析してみよう

課題１：課題１：*tf-idf*モデルの学習

- 宮沢賢治の全作品を使って、各作品を*1*文書としたときの*tf-idf*を学習
- *sklearn.feature_extraction.text*の*TfidfVectorizer*を利用
- *TfidfVectorizer*のパラメータは以下の通り
  - *max_features=1000 (*語彙サイズは最大*1000*個*)*
  - *max_df=5*　（*5*つ以下の小説までに現れる語彙）
  - *min_df=3*　（*3*つ以上の小説に現れる語彙）

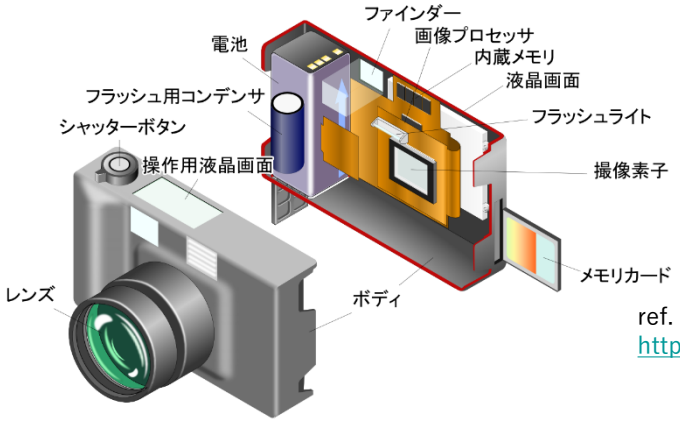
課題２：「風の又三郎」の重要語の抽出

- 小説「風の又三郎」の*tf-idf*値が大きい単語上位*10*単語とその*tf-idf*値
- 各行にカンマ区切りで単語と*tf-idf*値が並んだ形式で、*ex4-tfidf.txt*という名前のファイルに出力

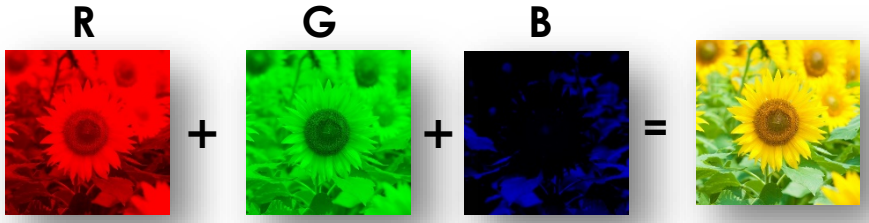
第5回 画像処理：画像を線画に変換しよう

講義内容：デジタルカメラの仕組みやカラー画像のデータ表現について学んだ後、簡単な画像処理を組み合わせて、カラー写真を線画に変換してみよう

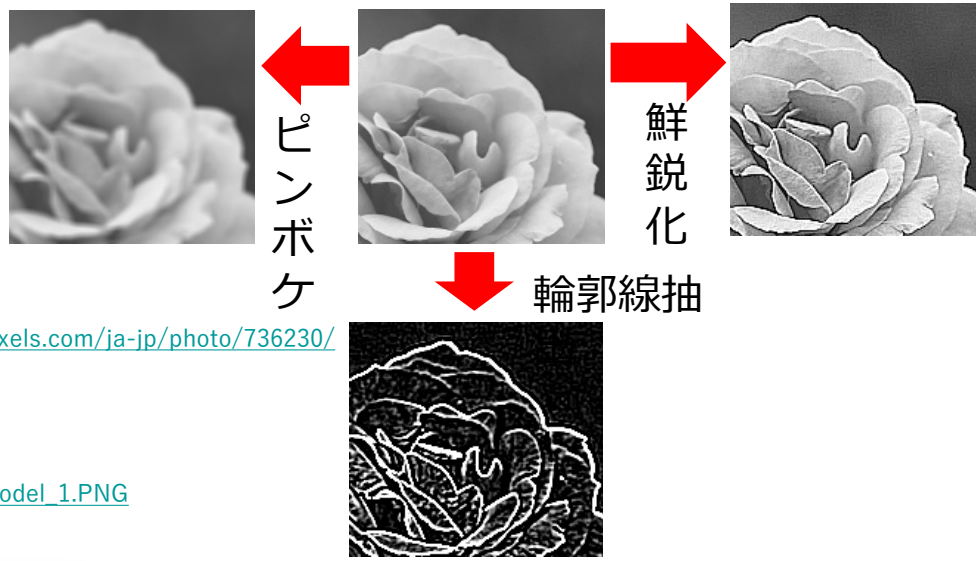
演習内容：カメラや画像の表現、画像の圧縮、色空間を理解したのち、画像処理ライブラリopenCVを使って2次元デジタルフィルタによる画像処理を学ぶ



デジタルカメラの仕組み
ref: https://commons.wikimedia.org/wiki/File:Digital_camera_cut_model_1.PNG
CC-BY-SA-3.0,2.5,2.0,1.0



デジタルカラー画像の表現



ref. (2020/4/4)
https://www.pexels.com/ja-jp/photo/736230/

2次元デジタルフィルタ
深層学習による画像認識の基礎技術
（畳み込み演算, Convolution)

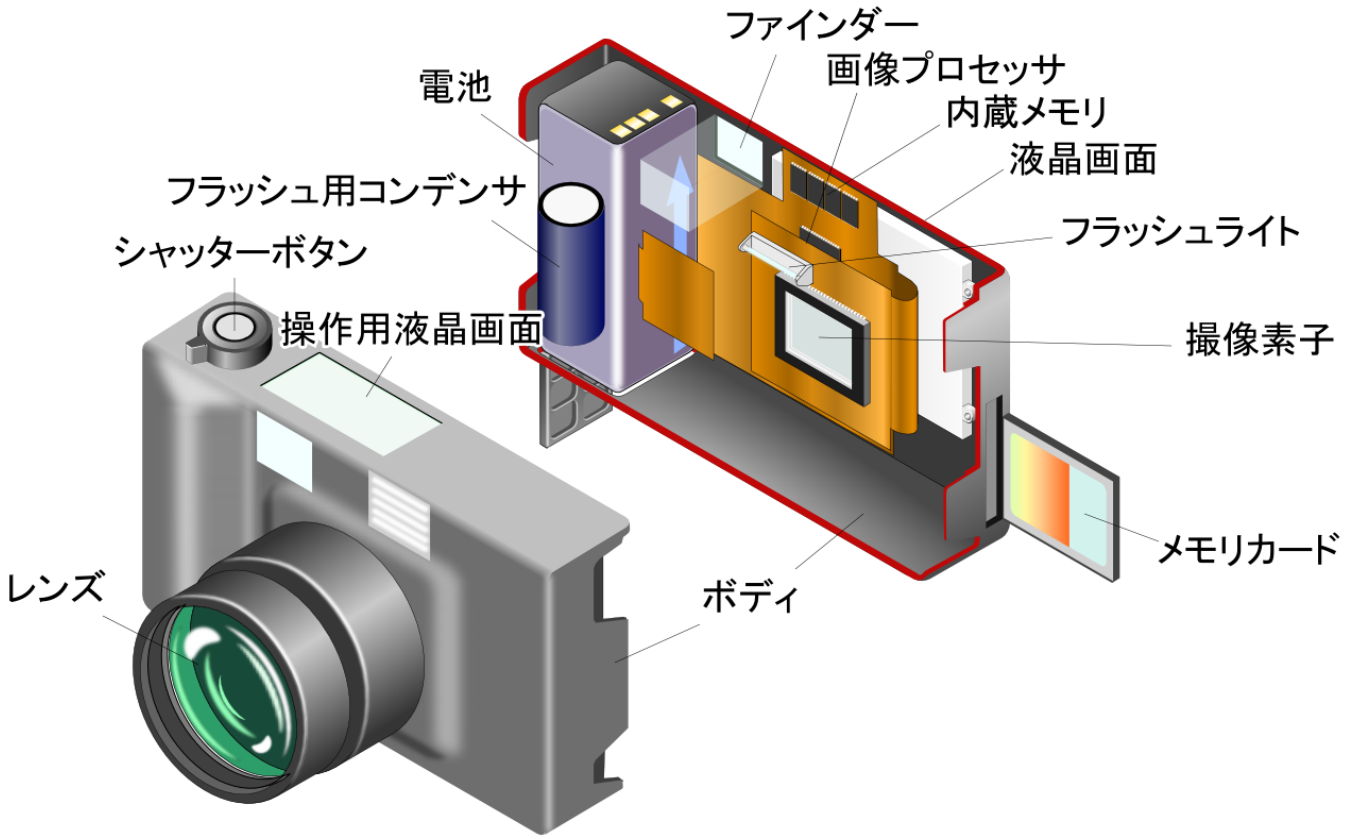
本日の学習内容

- 画像の表現と記録
  - デジタルカメラの仕組み
  - コンピュータにおける画像のデータ表現
  - 画像の保存と圧縮
- 色
  - 人間の色覚とディスプレイやプリンタにおける色表現
  - 色の直感的な近さを表現する色空間
  - カラーチェッカーを使った色恒常性の確保
- 二次元デジタルフィルタによる画像処理
  - 畳み込み演算（convolution)
  - 様々なフィルタ
    - ピンボケフィルタ
    - 鮮鋭化フィルタ
    - 輪郭線抽出フィルタ（エッジフィルタ）
    - ガウシアンフィルタ

画像の記録形式

ImageProcessing1.ipynb

デジタルカメラの仕組み



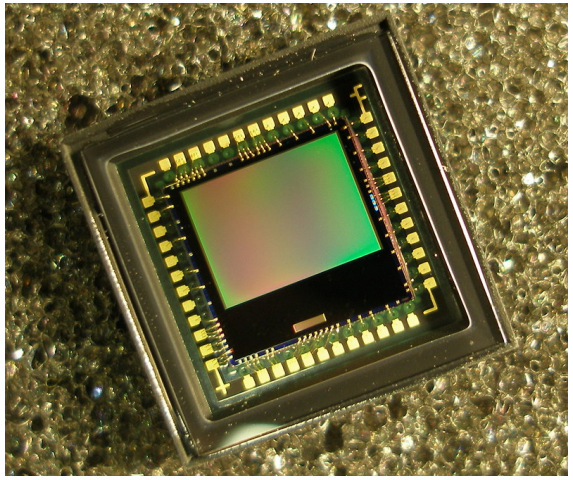
ref:千葉憲明著、『カメラの常識のウソ・マコト』、講談社、2004年6月20日第1刷発行、ISBN 4062574462, 森枝卓士著、『デジカメ時代の写真術』、NHK出版、2003年7月10日第1刷発行、ISBN 4140880740, 津軽海渡、木村誠恥著、『図解雑学 デジタルカメラ』、ナツメ社、2002年12月18日発行、ISBN 4816334092
ref. (2020/4/3): Wikimedia commons: File:Digital camera cut model 1.PNG (https://commons.wikimedia.org/wiki/File:Digital_camera_cut_model_1.PNG) CC BY-SA 3.0

撮像素子

- カメラの撮像素子とソーラーパネルの原理は同じ
  - 太陽光発電は晴れの日にたくさん発電、曇りや雨の日はあまり発電しない
  - 物質に光を当てると電子が飛び出る「光電効果」を利用
  - 半導体を使うと電子が電流として取り出せる

ref. (2020/4/3): Wikimedia commons: File:Solar panels on house roof.jpg CC BY-SA 3.0 https://commons.wikimedia.org/wiki/File:Solar_panels_on_house_roof.jpg

- 撮像素子は極小の正方形ソーラーパネルが並んだもの
  - 光が強く当たったところはたくさん電荷がたまる→明るい(白っぽい)
  - 光があまり当たらなかったところはあまり電荷がたまらない→暗い(黒っぽい)



ref. (2020/4/3): Wikimedia commons: File:Matrixw.jpg CC BY-SA 3.0 https://commons.wikimedia.org/wiki/File:Matrixw.jpg

アナログ画像とデジタル画像の違い

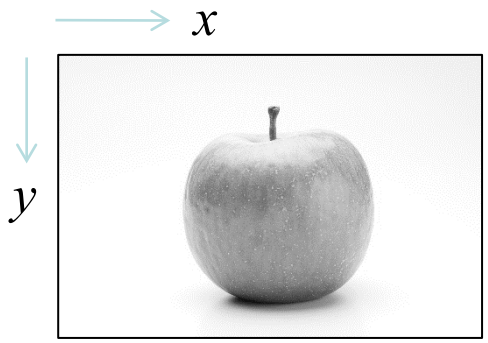
画像とは平面座標系で定義される2変数関数

アナログ画像の表現

$$z = f(x,y)$$

x, y, z は0以上の**実数**

- (x,y)は平面上の位置座標
- z は(x,y)における何らかの信号（光）の強さ

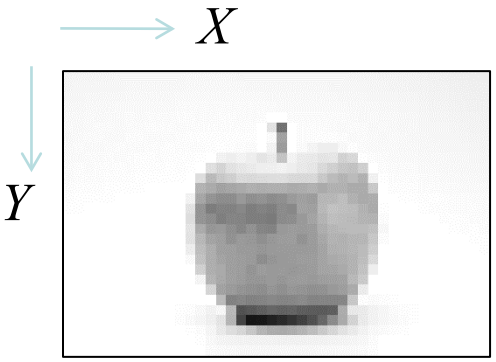


デジタル画像の表現

$$Z = F(X,Y)$$

X, Y, Z は0以上の**整数**

- (X,Y)は平面上の格子点
- Zは格子点(X,Y)における何らかの信号(光)の強さ



1つの格子点を**画素**または**ピクセル**(pixel)と呼ぶ
色の濃さ(Zの値の大きさ) を**輝度**または**濃淡値**と呼ぶ

デジタル画像は正方形タイルの集まり

横縦方向の刻みが細かい⇒**解像度が高い**

各ピクセルの階調の刻みが細かい
⇒**ビット深度が深い**

ref. (2020/4/3): Wikimedia commons: File:Red Apple.jpg CC BY 2.0 https://commons.wikimedia.org/wiki/File:Red_Apple.jpg

解像度の高低による画像の違い

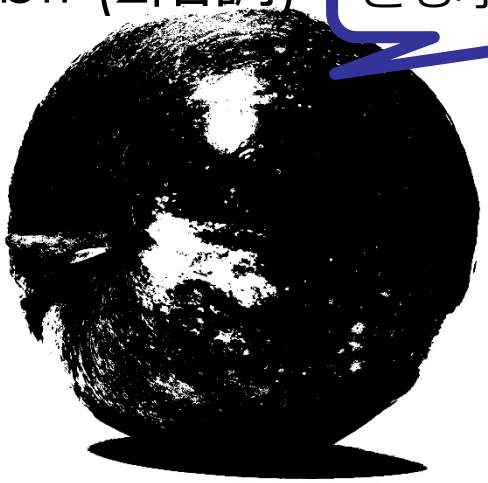
低解像度

高解像度

ビット深度の深さによる画像の違い

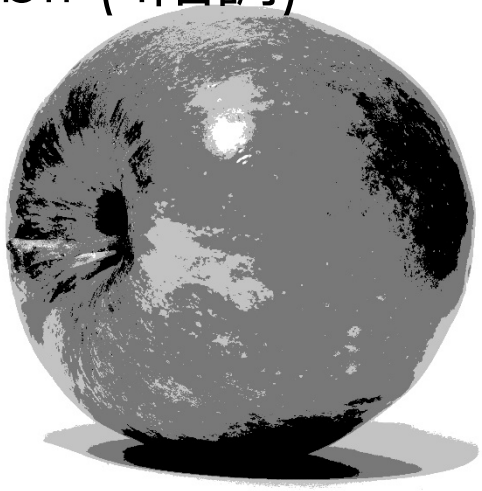
1bit (2階調)

二値画像 (binary image) とも呼ぶ

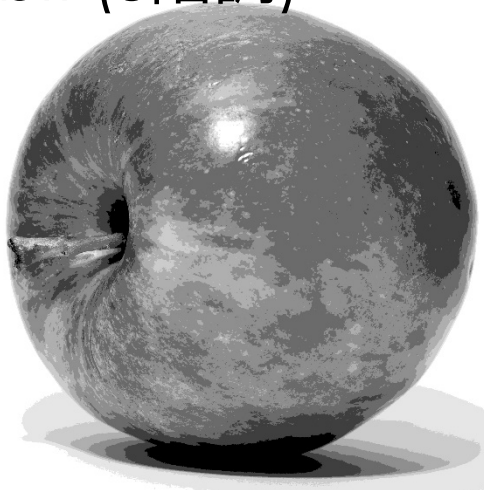


低ビット深度

2bit (4階調)



3bit (8階調)



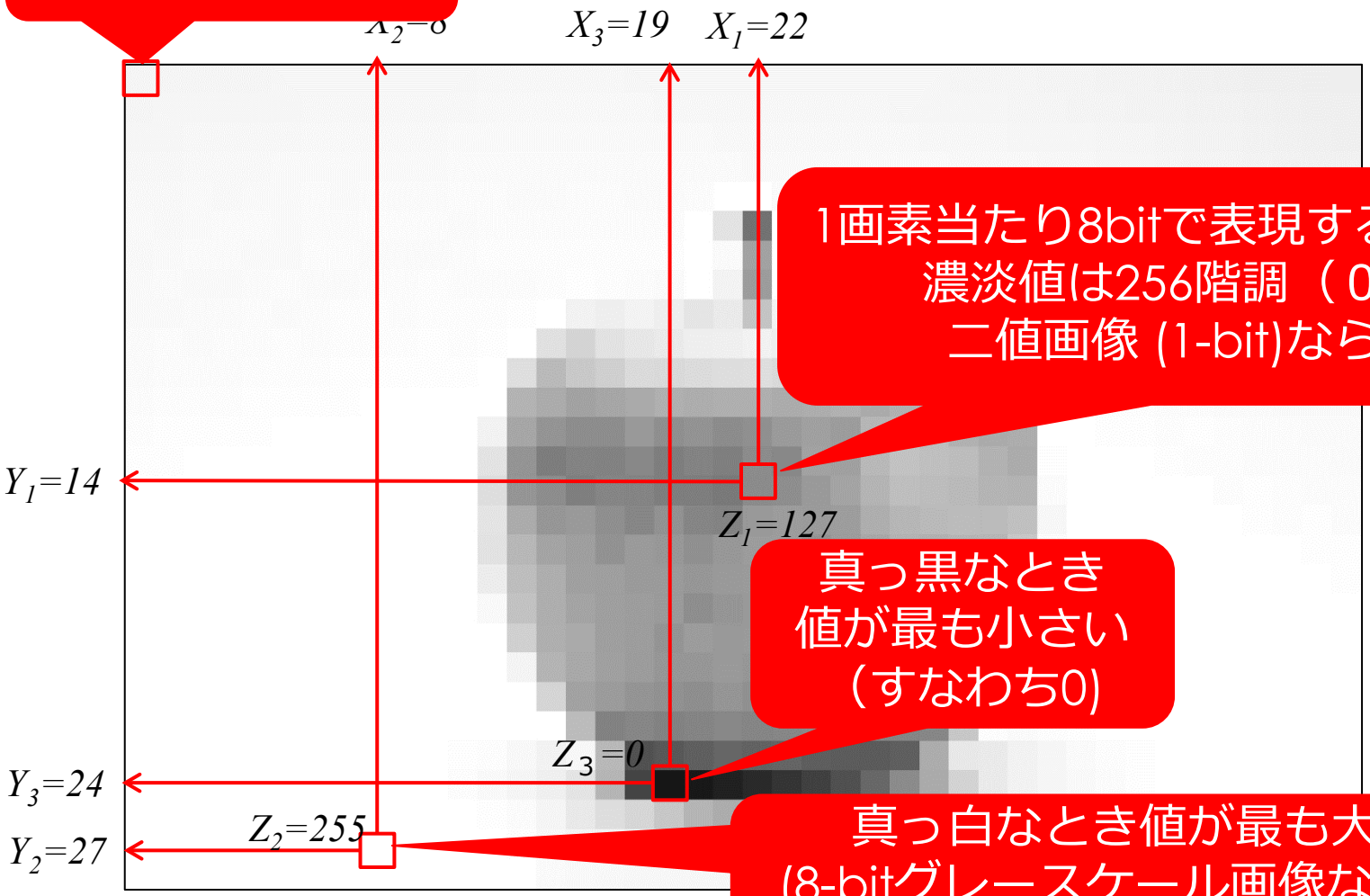
高ビット深度

8bit (256階調)：

一般的にグレースケール画像といえばこれ

座標と濃淡値の基準

原点(X=0, Y=0)は
左上が一般的



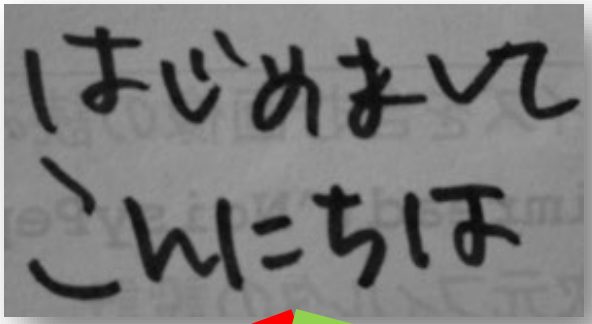
1画素当たり8bitで表現するとすると、
濃淡値は256階調（0～255)
二値画像 (1-bit)なら2階調

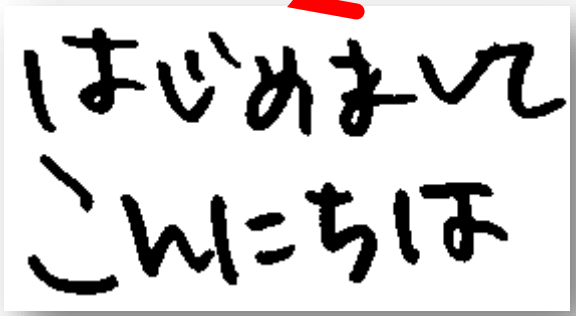
真っ黒なとき
値が最も小さい
（すなわち0)

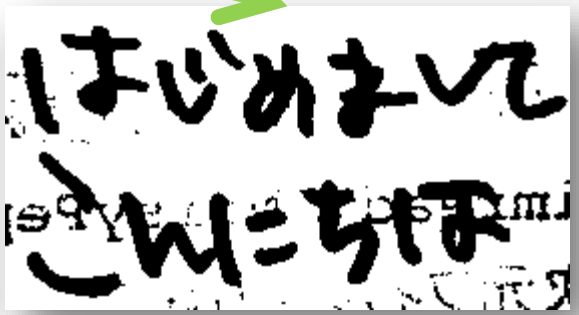
真っ白なとき値が最も大きい
(8-bitグレースケール画像なら255、
二値画像 (1-bit) なら1)

画像の二値画像 (Image Binarization)

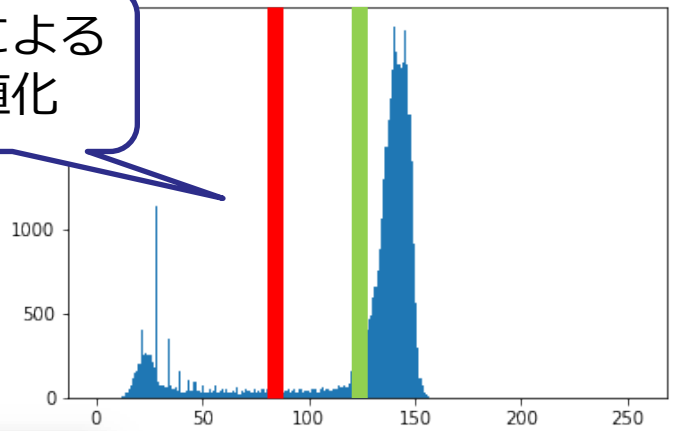
- 白と黒の二値しかない画像
  - 文書では文字が黒、背景が白
- 二値画像では、グレースケールのうち、中間の色を白か黒かに振り分けることによりノイズが除去される







閾値による二値化



画像のヒストグラム

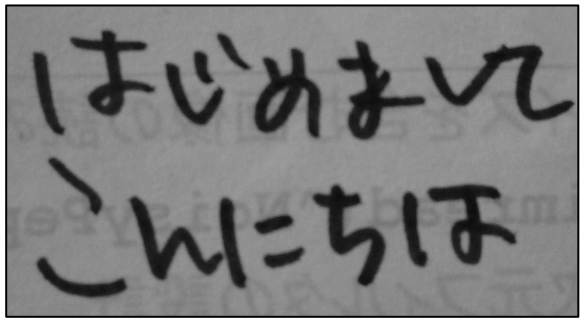
画像の二値化～裏紙に書いた文字の抜き出し～

- 二値画像（binary image)
  - 各画素の色（濃淡）を白（１）か黒（０）で表現

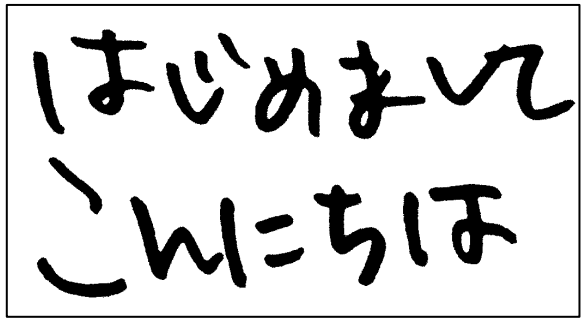
課題）裏紙に書いた絵をスキャンしたもの

→ 表に書いた文字を黒，それ以外を白にしたい

原画像



目的画像



画像の二値化～裏紙に書いた文字の抜き出し～

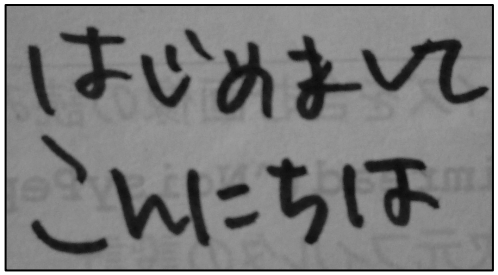
課題）裏紙に書いた絵をスキャンしたもの

→ 表に書いた文字を黒，それ以外を白にしたい

- 各画素がある値より大きい（明るい）ならば白，小さい（暗い）ならば黒にすればいい
- このとき，この「ある値」を閾（しきい）値（threshold)と呼ぶ

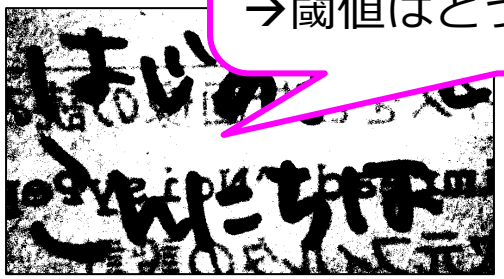
閾値が適切でないと表の文字だけが黒にならない！
→閾値はどう決めたらいい？

原画像

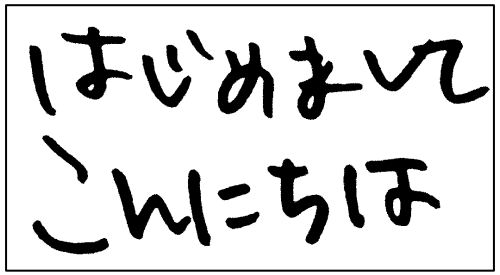


明度は0～256で表現

明度が100以上なら白
そうでないなら黒



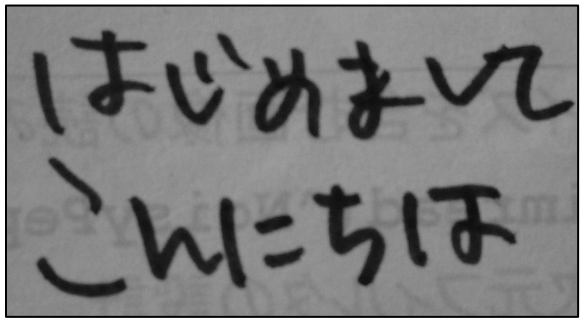
明度が50以上なら白
そうでないなら黒



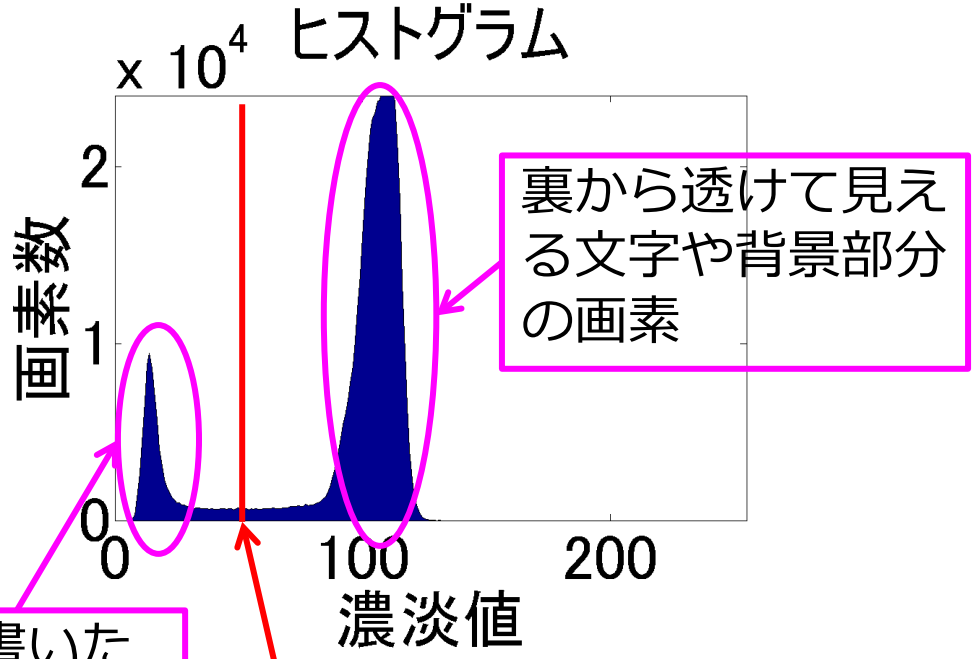
ヒストグラムを計算する

明度の各値の画素がいくつあるかでグラフ化

原画像



画像サイズ1050x570
→画素数は6.0×10^6



判別分析法（大津の二値化）

- 分散度（クラス間分散とクラス内分散の差）が最大となるようしきい値を自動で設定
  - しきい値tで二値化したとき，
    - しきい値より小さい側の画素数ω_1, 平均m_1, 分散σ_1
    - しきい値より大きい側の画素数ω_2, 平均m_2, 分散σ_2
    - 画像全体の画素数ω_t, 平均m_t, 分散σ_t

    とすると，

クラス内分散　$\sigma_w^2 = \dfrac{\omega_1\sigma_1^2 + \omega_2\sigma_2^2}{\omega_1 + \omega_2}$

クラス間分散

$$\sigma_b^2 = \frac{\omega_1(m_1 - m_t)^2 + \omega_2(m_2 - m_t)^2}{\omega_1 + \omega_2} = \frac{\omega_1\omega_2(m_1 - m_2)^2}{(\omega_1 + \omega_2)^2}$$

判別分析法（大津の二値化）

全分散値は $\sigma_t^2 = \sigma_b^2 + \sigma_w^2$ なので，クラス間分散とクラス内分散の比は次のようになる。

$$\frac{\sigma_b^2}{\sigma_w^2} = \frac{\sigma_b^2}{\sigma_t^2 - \sigma_b^2}$$

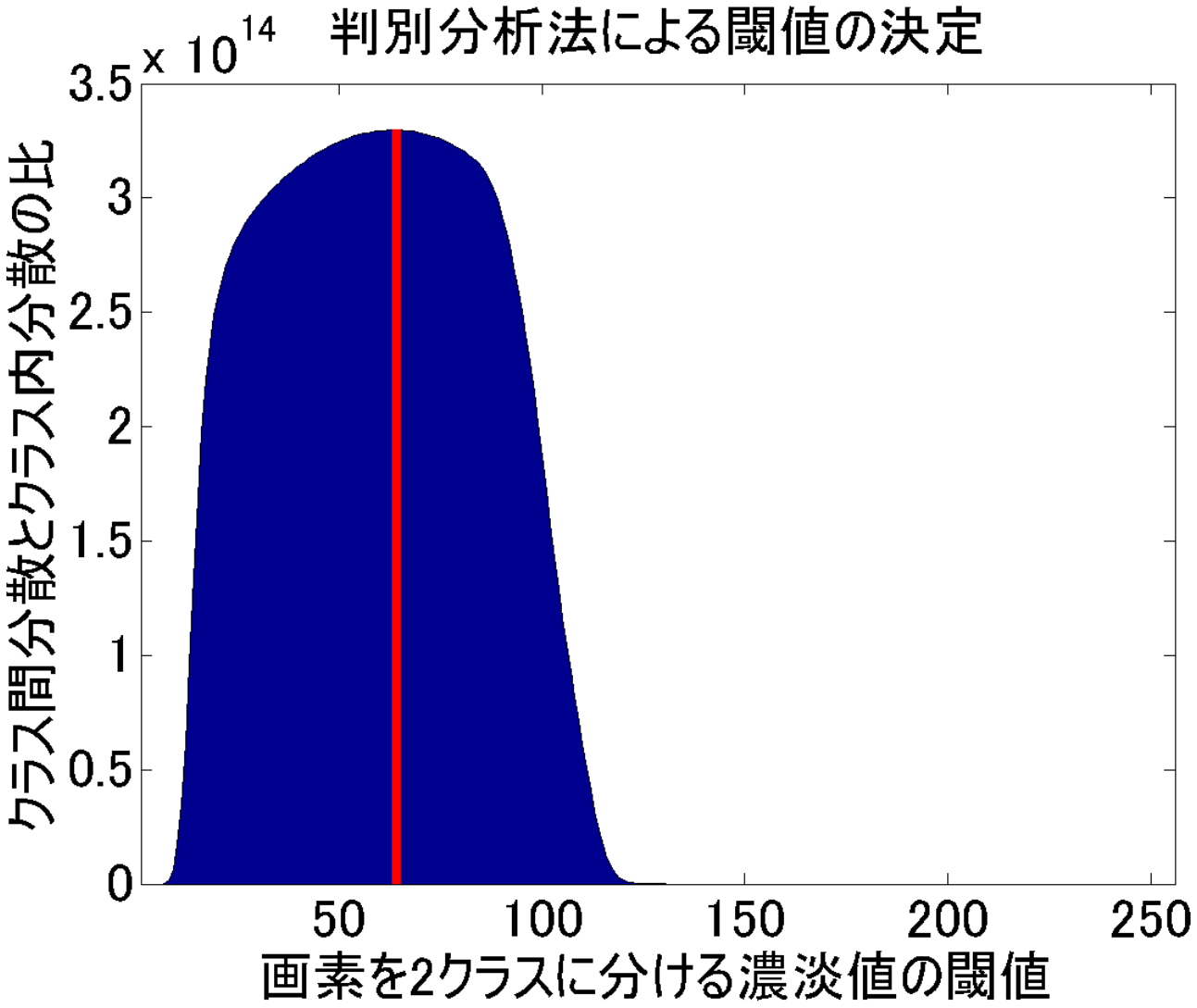
これが最大となるしきい値 t を求めればよい.

ここでσ_tはしきい値に関係なく一定なので，クラス間分散$\sigma_b{}^2$が最大となるしきい値を求めればよい.

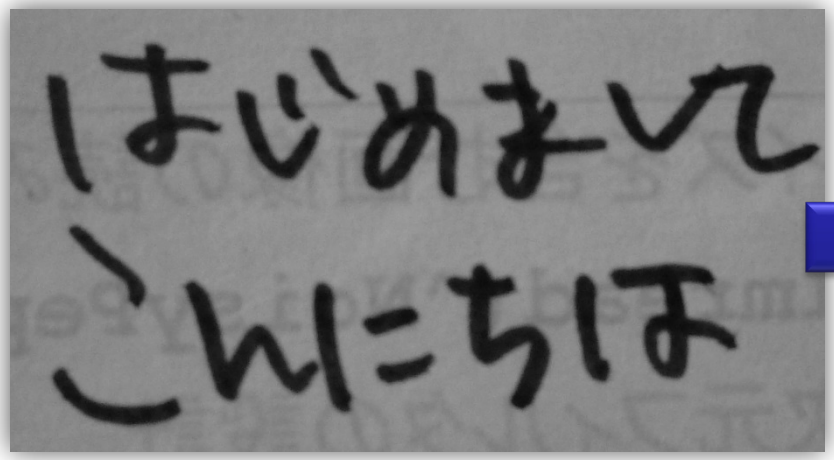
また、 $\sigma_b{}^2$の分母も一定なため，結局

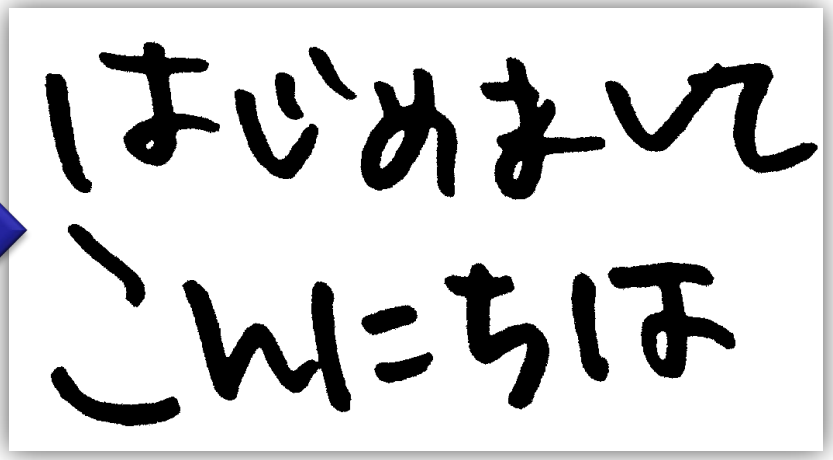
$\omega_1\omega_2(m_1 - m_2)^2$ が最大になる t をもとめればよい

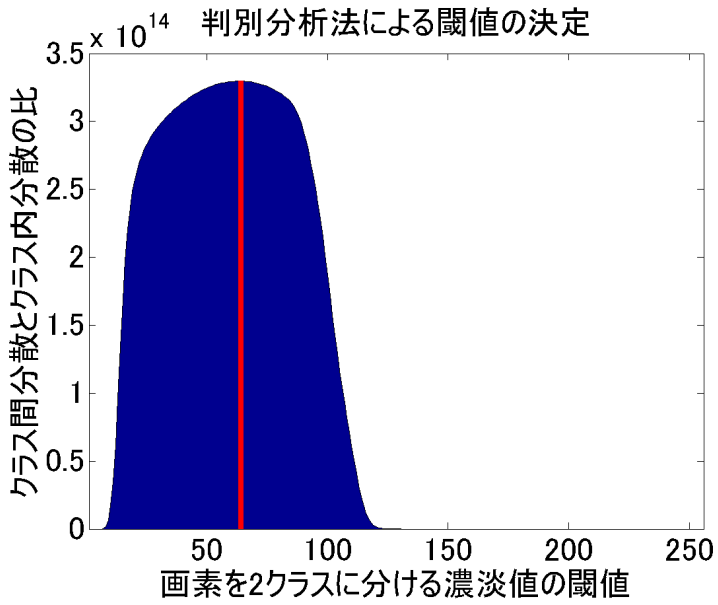
大津の二値化による閾値の探索

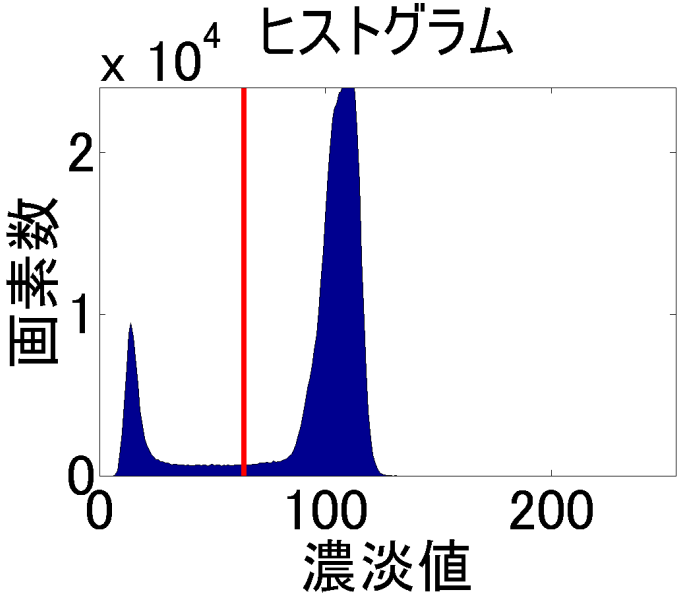


判別分析法の結果

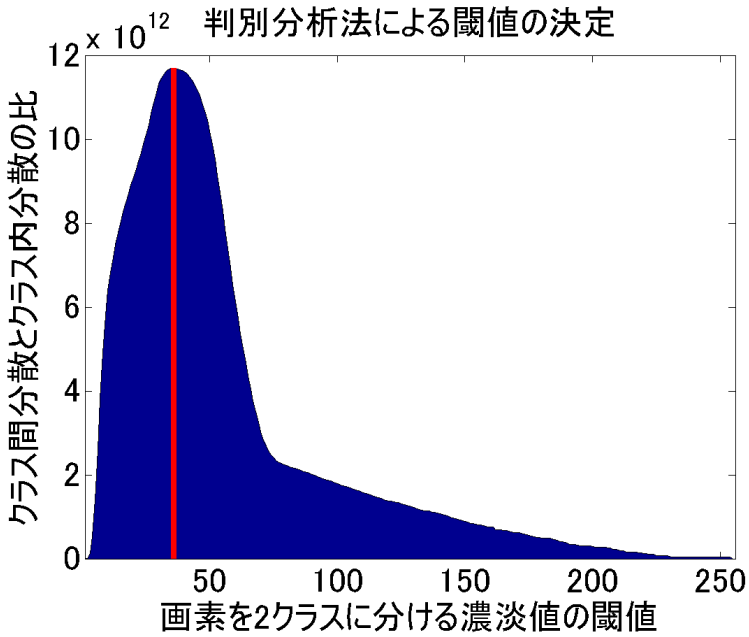

はじめまして
こんにちは


はじめまして
こんにちは

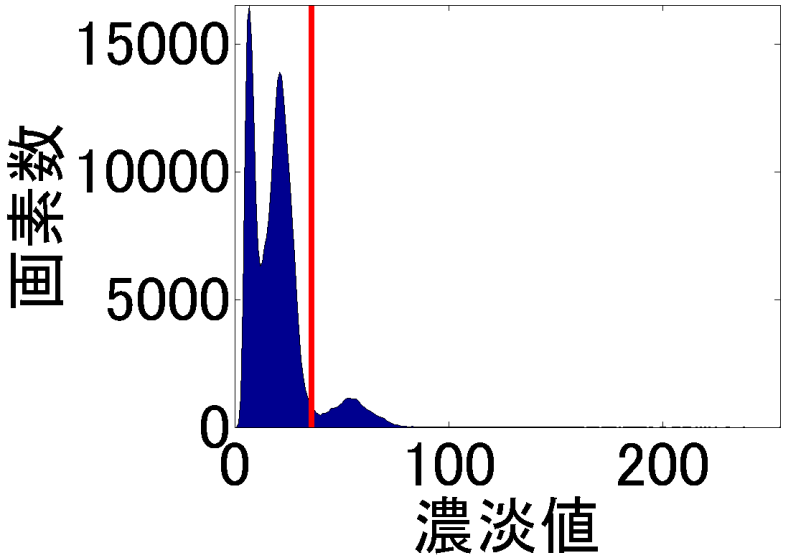

判別分析法による閾値の決定
x 10^14
クラス間分散とクラス内分散の比
3.5
3
2.5
2
1.5
1
0.5
0
50
100
150
200
250
画素を2クラスに分ける濃淡値の閾値


ヒストグラム
x 10^4
画素数
2
1
0
0
100
200
濃淡値

判別分析法の結果（写真の場合はうまくいかない）



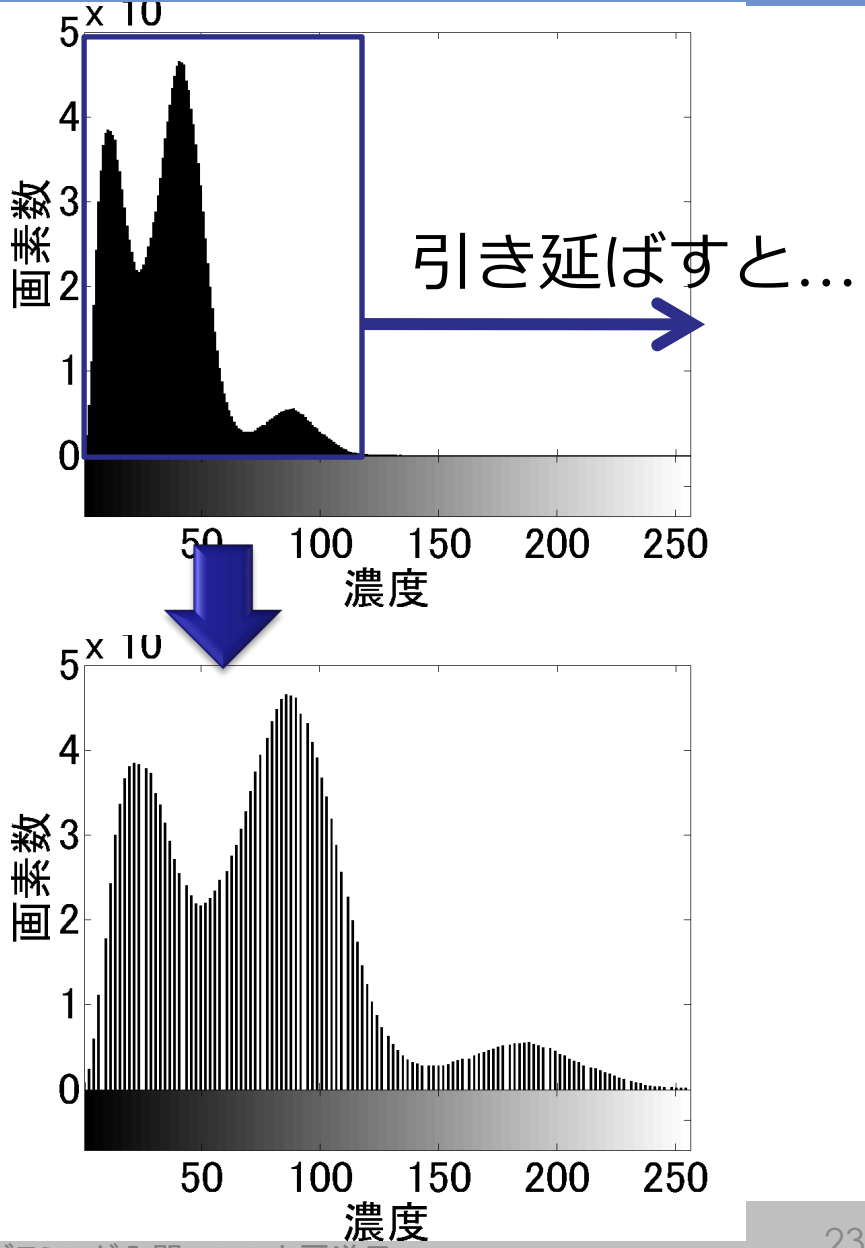
ヒストグラム

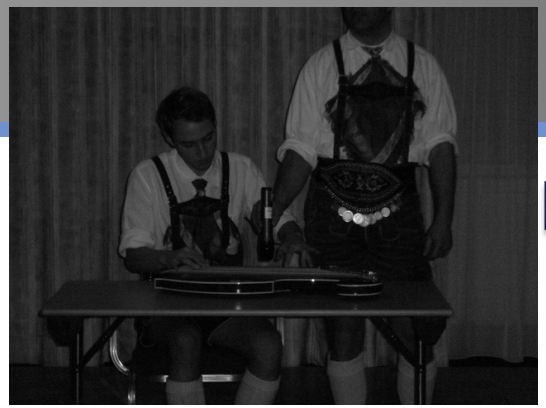


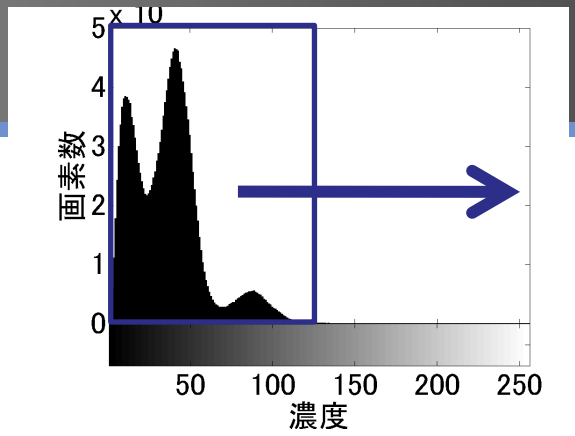
明度のストレッチ
～薄暗い画像を明るくしよう～

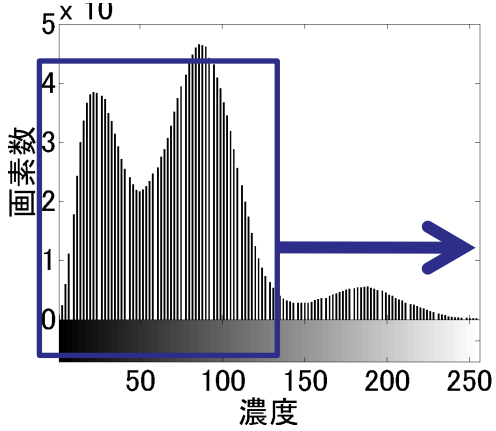
課題）暗い部屋で写真を撮った
→ 何が写っているのかさっぱり分からないので明暗をはっきりさせたい！

明度のストレッチ
～薄暗い画像を明るくしよう～

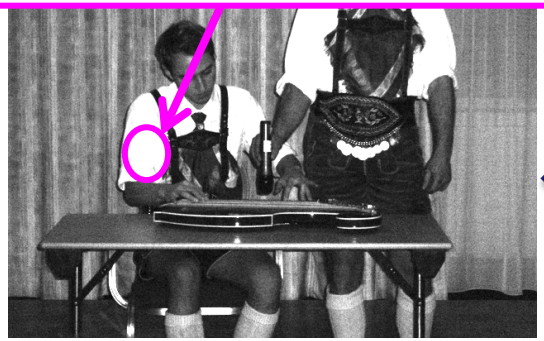


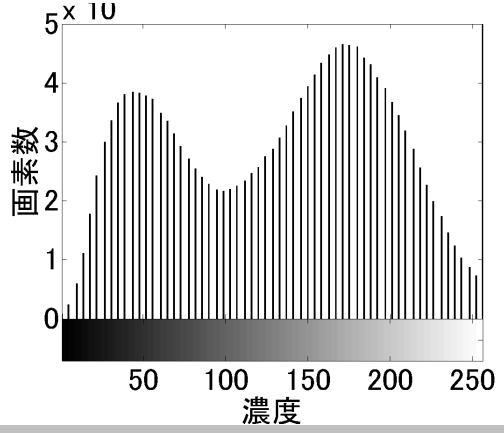






白い洋服の部分は白とびしている（明度が255に至ってそれ以上上げられない状態）





画像の保存形式

ImageProcessing2.ipynb

画像の保存形式

- 輝度値が画素分並んだ画像（例: bitmap）はサイズ大！
  - ノートPCのよくある解像度：1920 x 1200
  - 小さいと感じる640x480で1画素あたり8-bit×RGB3チャネルの画像で、1画素あたり24-bitフルカラーで記録すると、1枚当たり151MB → CD-ROMに4枚しか保存できない！
- よく使われる画像保存形式
  - PNG (Portable Network Graphics) フォーマット
  - JPEG (Joint Photographic Experts Group) フォーマット

どちらを選ぶべきか？

PNG (Portable Network Graphics) フォーマット

- 可逆圧縮（元の画像を復元できる）
- 圧縮レベルを上げると圧縮に時間がかかる
- **イラストのように同じ画素値が並んでいる画像だと高圧縮が期待できる**（逆に写真はあまり小さくならない）

写真

イラスト風

← 写真と違い
色がべた塗り

	圧縮レベル０		圧縮レベル９	
	時間	画像サイズ	時間	画像サイズ
イラスト風	0.065 s	3.2 MB	0.406 s	0.19 MB
写真	0.097 s	3.2 MB	3.226 s	1.5 MB

※Intel Core i7-8550U CPU @ 1.80GHz、Python+OpenCVで実行

短時間でサイズ激減！

時間がかかる割に画像サイズはあまり小さくならない

JPEG (Joint Photographic Experts Group) フォーマット

- 非可逆圧縮（元の画像を復元できない）
- 高速圧縮。人間の視覚で知覚されづらい情報を捨てることで圧縮
- **写真のようにテクスチャが複雑な画像の圧縮が得意**
- 圧縮率が高いと見た目にわかるひずみが現れる（ブロックノイズ）

低圧縮写真

ブロックノイズが現れる

高圧縮写真

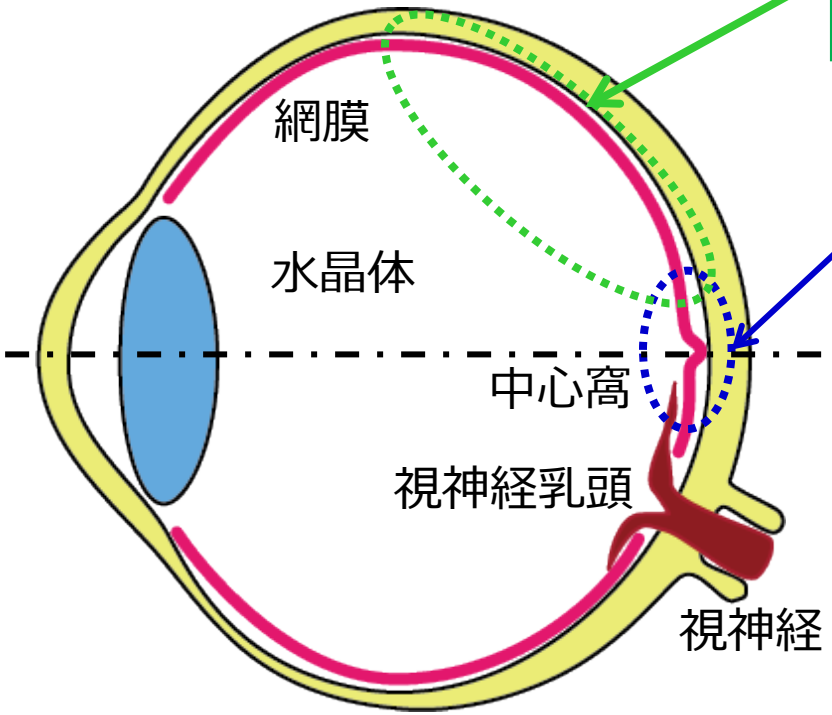
	クオリティ100		クオリティ10	
	時間	**画像サイズ**	**時間**	**画像サイズ**
イラスト風	0.067 s	340 KB	0.032 s	31 KB
写真	0.071 s	813 KB	0.033 s	41 KB

写真・イラスト風**ともに短時間でサイズ縮小が可能**

※Intel Core i7-8550U CPU @ 1.80GHz、Python+OpenCVで実行

色情報の獲得

光と色を知覚する細胞



桿体細胞：高感度（暗所で機能）
中心窩から離れた領域に広く分布

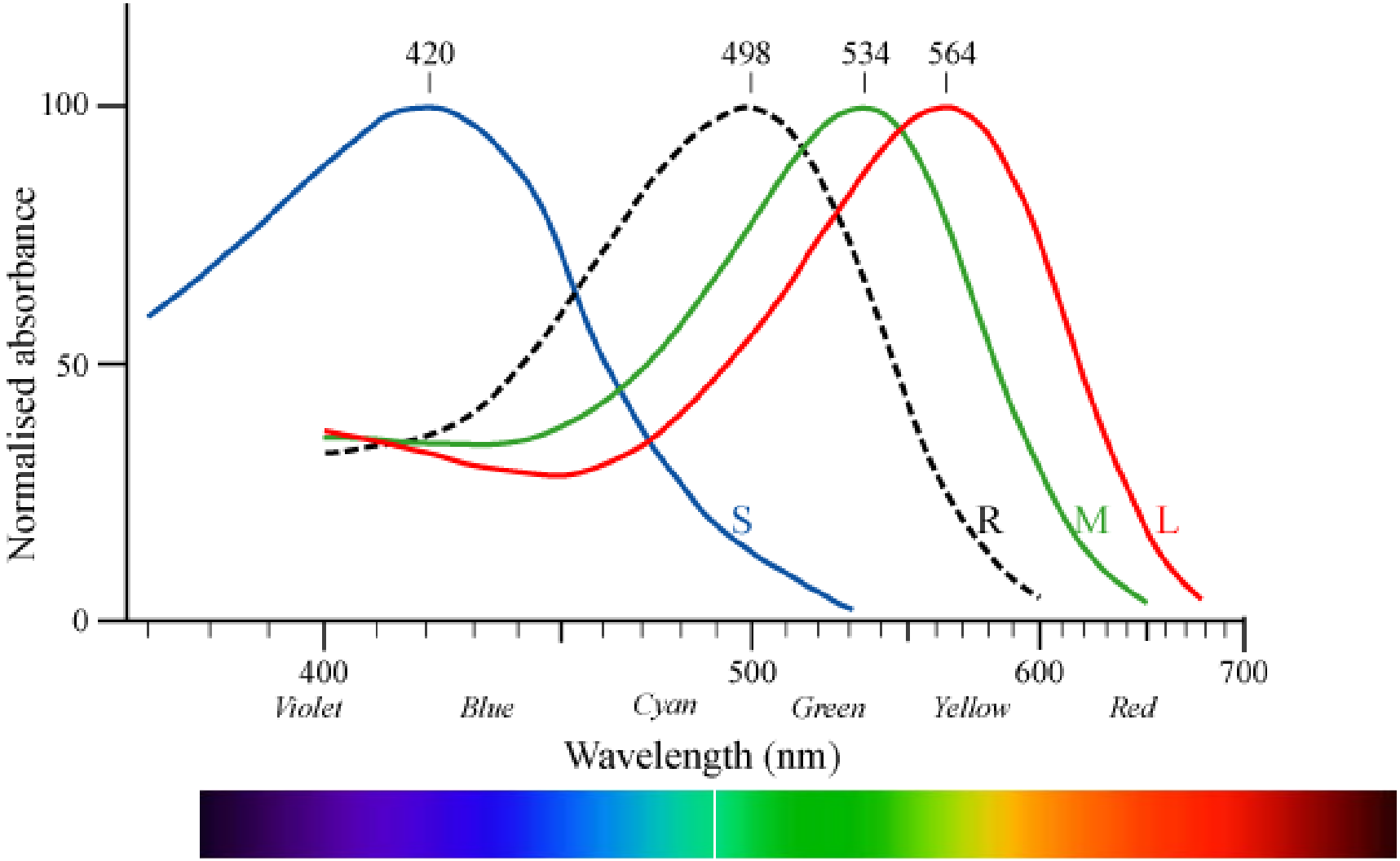
錐体細胞：
特定の波長の光に反応
（色覚）中心窩には錐体細胞のみ

- 3色型色覚の人は3種類の錐体細胞を持つ
- それぞれ、反応する光の波長の範囲が異なる

Ｌ錐体（570nm－黄周辺）
Ｍ錐体（535nm－緑周辺）
Ｓ錐体（445nm－青周辺）

知覚する色情報は３次元

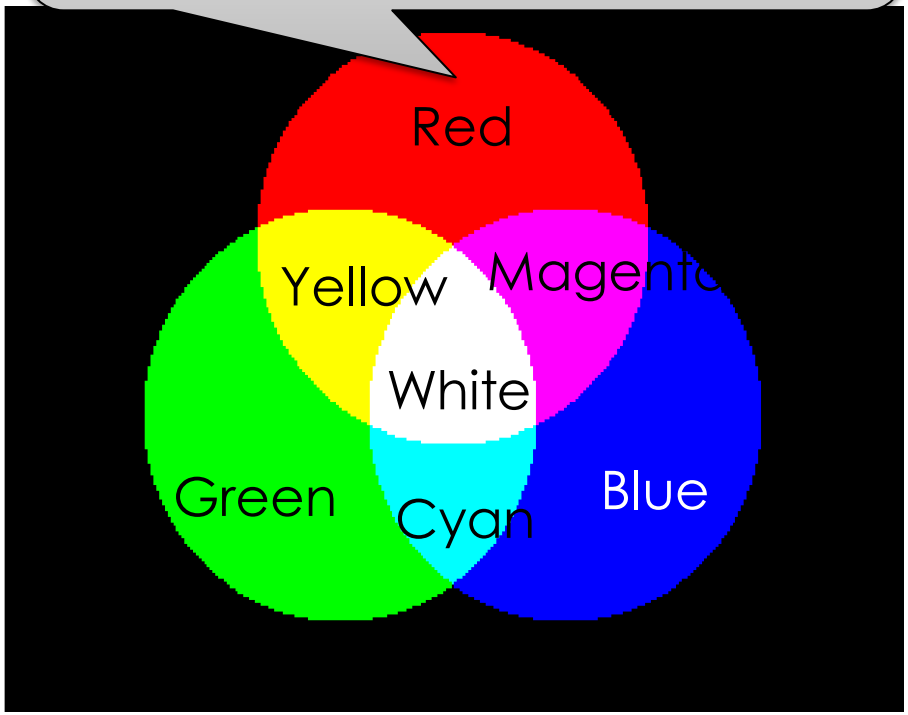
異なる波長の光に対する人間の錐体細胞の感度



光は色によって波長が異なる

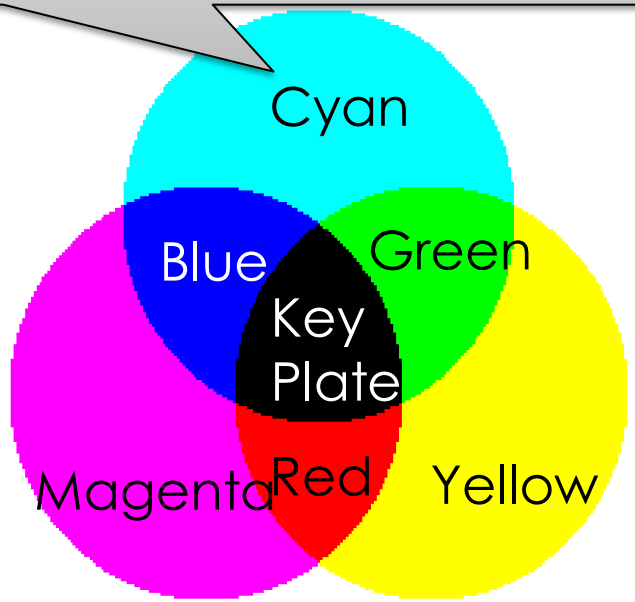
加色法と減色法

この色のカラーフィルム（赤青メガネの赤側）は青と緑の光を遮断し、赤色の光だけを透過する



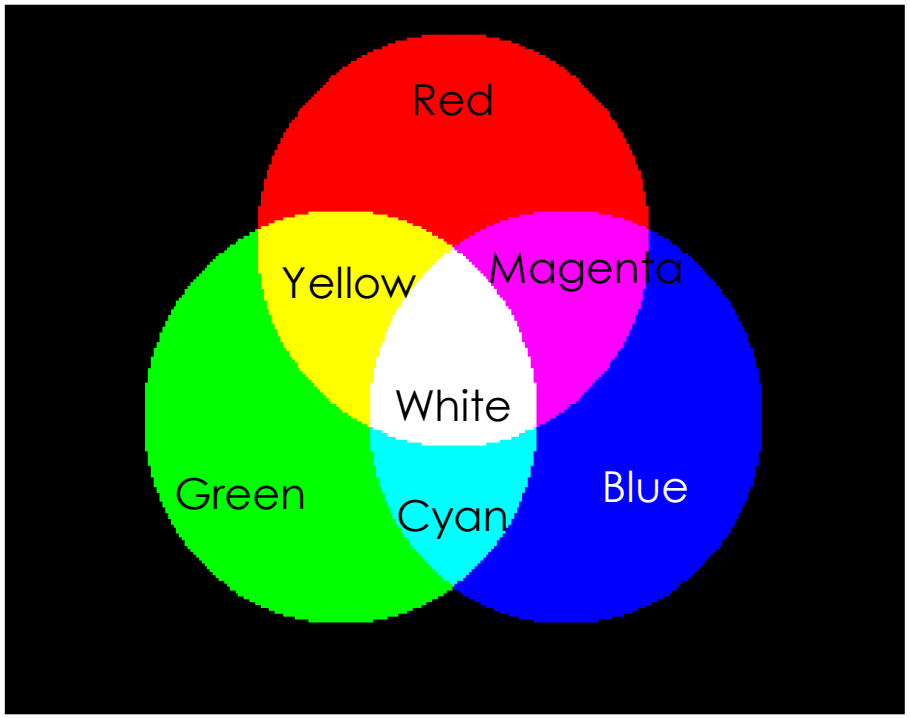
加色法
光の三原色。
すべて足すと白になる。
ディスプレイで使われる

この色の絵具は赤色の光を吸収して青と緑の光を反射する

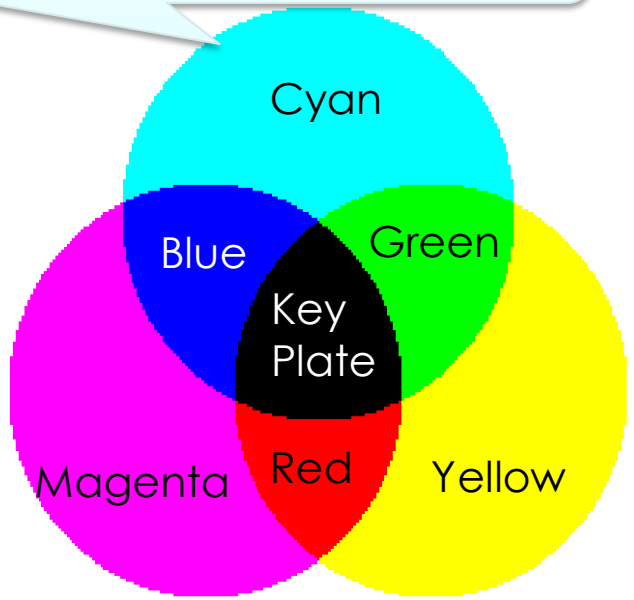


減色法
絵具の三原色。
すべて足すと黒になる。
プリンタで使われる

加色法と減色法



この色の絵具は赤色の光を吸収して青と緑の光を反射する



加色法

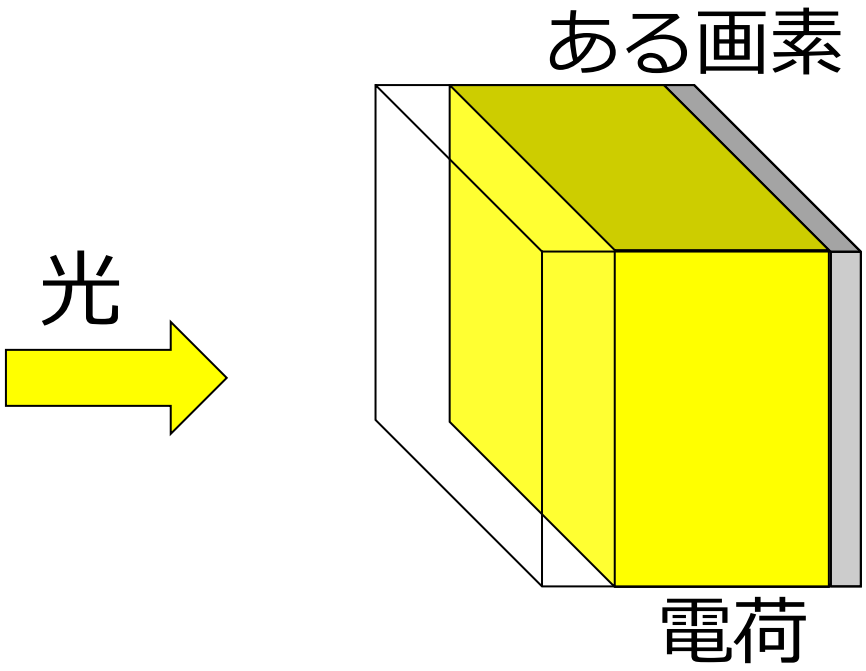
光の三原色。
すべて足すと白になる。
ディスプレイで使われる

減色法

絵具の三原色。
すべて足すと黒になる。
プリンタで使われる

この画面を赤青メガネ（本当は赤シアンメガネ）でのぞいてみよう！

撮像素子による「光の色」の計測



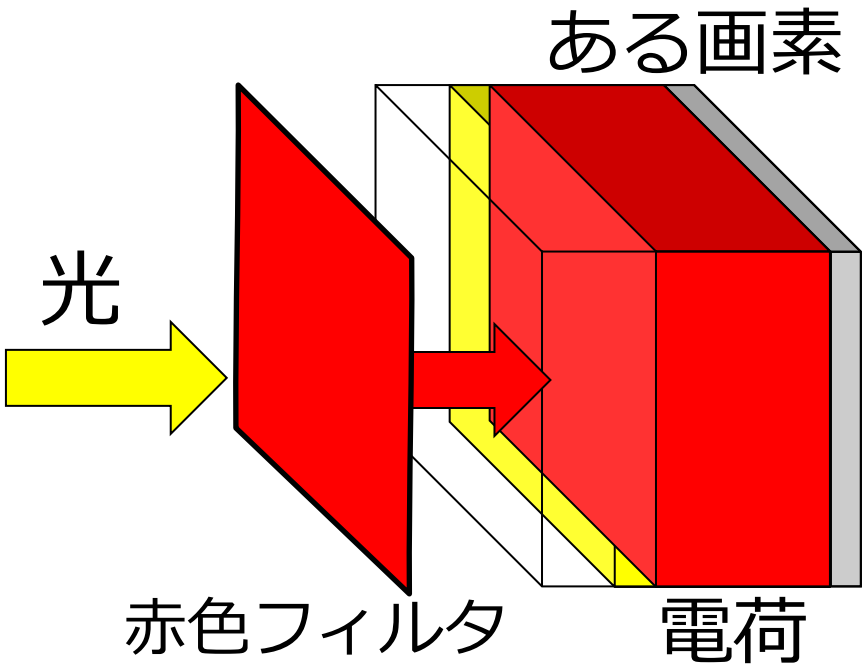
- 撮像素子は光の強さしか計測できない

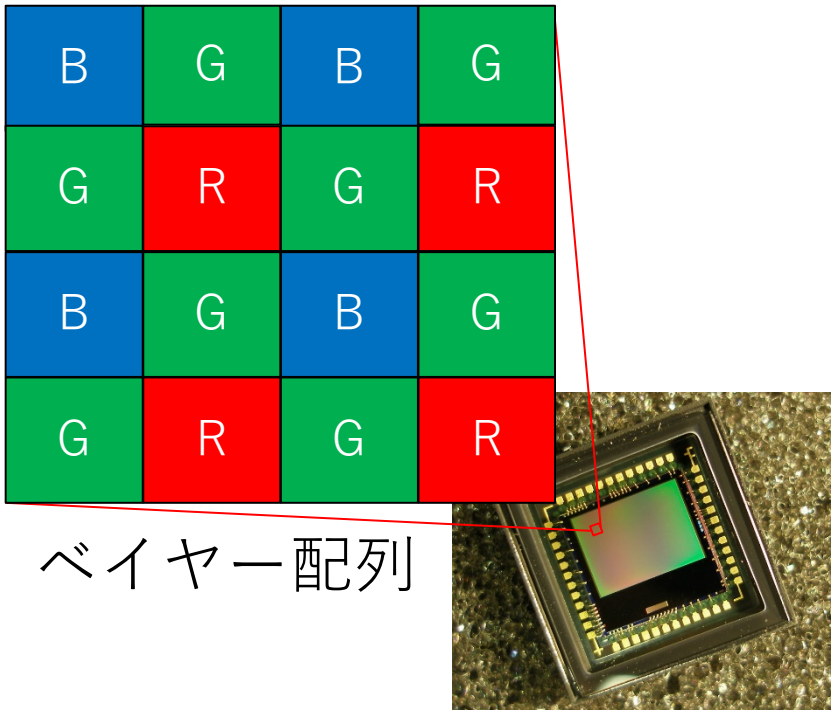
→撮像素子の各画素の前にカラーフィルタを設置

- 赤色フィルタは赤い波長の光のみを通過

→ 赤色の波長の光量を測定

撮像素子による「光の色」の計測





ref. (2020/4/3): Wikimedia commons: File:Matrixw.jpg
CC BY-SA 3.0
https://commons.wikimedia.org/wiki/File:Matrixw.jpg

- 撮像素子は光の強さしか計測できない

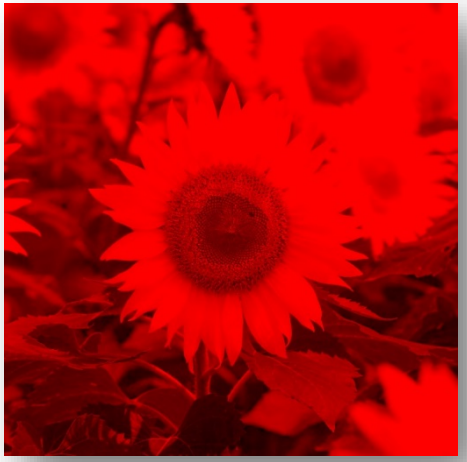
→撮像素子の各画素の前にカラーフィルタを設置

- 赤色フィルタは赤い波長の光のみを通過

→ 赤色の波長の光量を測定

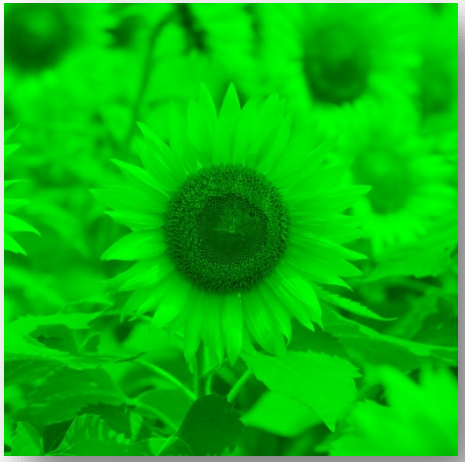
デジタル画像（24bitフルカラー）の表現

R 1画素あたり8bit (256階調)



+

G 1画素あたり8bit (256階調)



+

B 1画素あたり8bit (256階調)

=

1画素あたり24bit
（1677万色）
フルカラー

デジタルカラー画像の表現

原画像

R

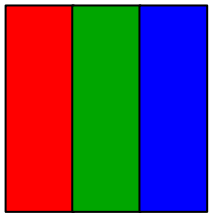
原画像を赤いフィルムを通してみるとRの濃淡画像と同じに見える！

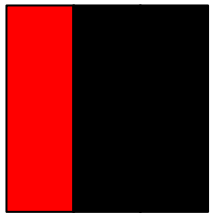
G

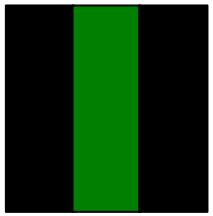
B

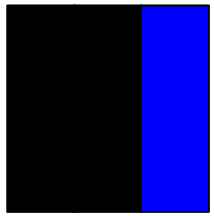
ディスプレイの色表現

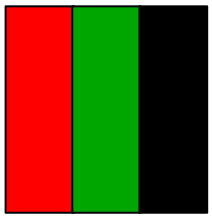
- 白色のバックライトの前に縦長の液晶シャッター（閉じるか開くかを制御．開くと光を透過）を並べ，さらにその前に赤・緑・青のカラーフィルムを並べる
- 液晶ディスプレイの1画素を拡大すると．．．

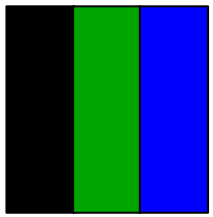

白色のとき

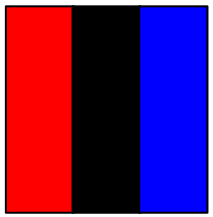

赤色のとき


緑色のとき


青色のとき


黄色のとき

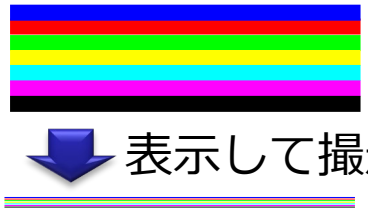

シアンのとき


マゼンタのとき

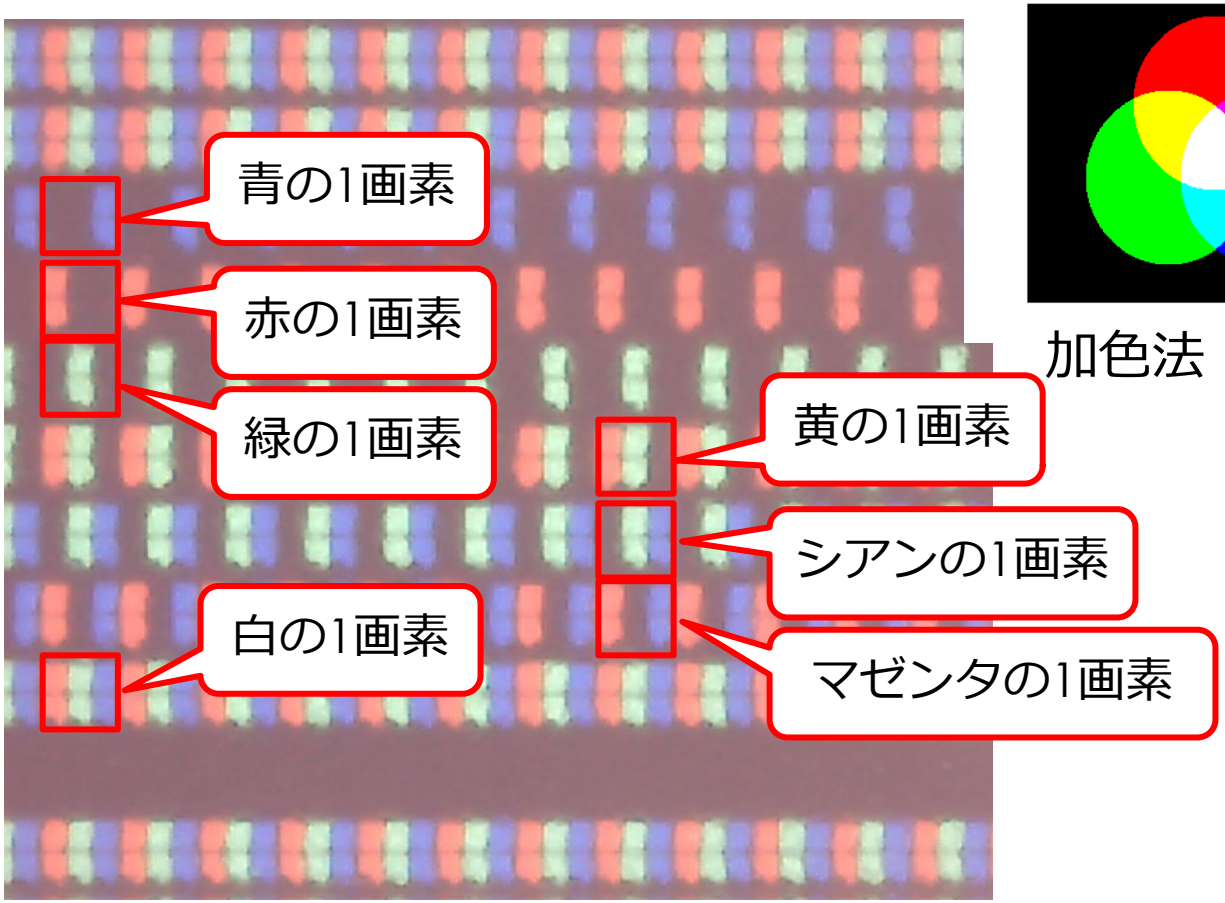

黒色のとき

ディスプレイをマイクロカメラで見てみよう

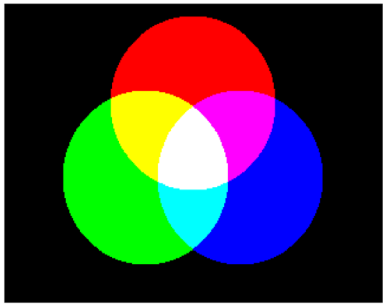
150倍マイクロカメラでディスプレイ表面を撮影



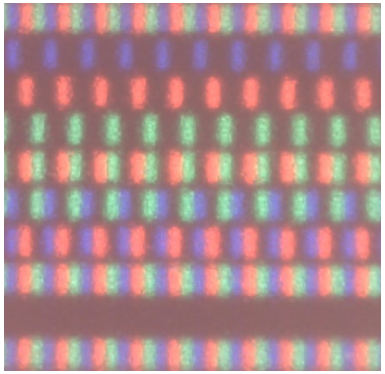
表示して撮影



Let’s note CF-SZ6

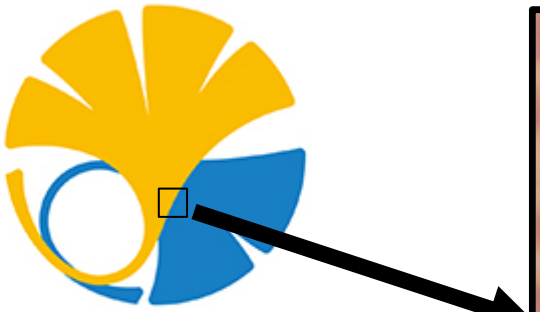


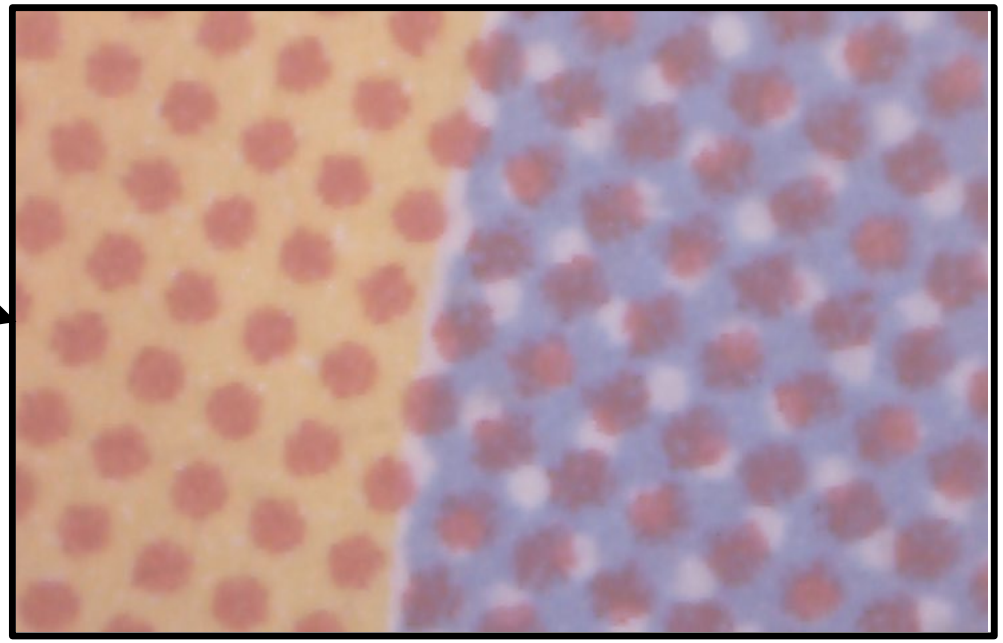
加色法（RGB）



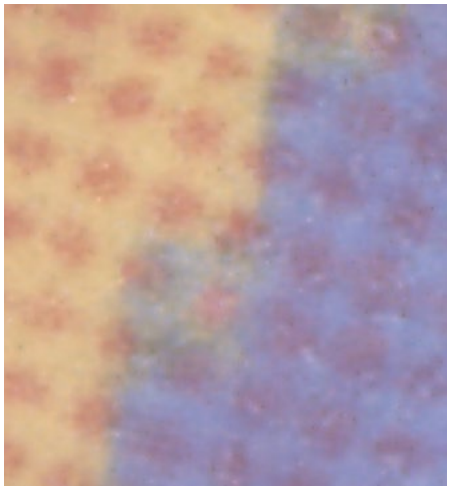
モニタ（PHILIPS HDR 600）

カラー印刷をマイクロカメラで見てみよう

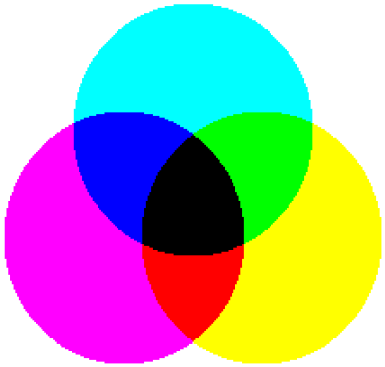




印刷会社で印刷されたパンフレット



プリンタで印刷したもの
Fuji Xerox ApeosPort-VI



減色法（CMYK）

色の直感的な近さを表現する色空間

RGB/CMYでは“色の直感的な近さ”を測りにくい

画像中で茶色の領域を切り出そうと思ったら・・・

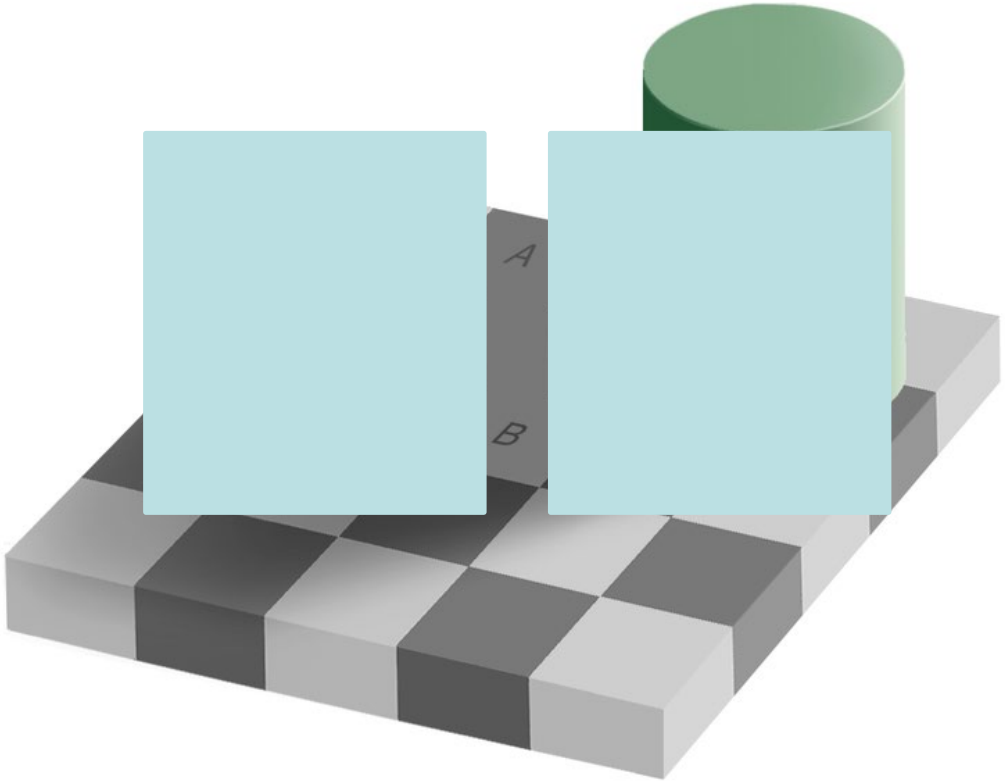
原画像

R画像

レンガと青空は直感的には別の色だが、RGBのRだけを
見ると区別がつかない（同程度にRの成分を持つ）

RGB/CMYでは“色の直感的な近さ”を表現できない

チェッカーシャドウ錯視



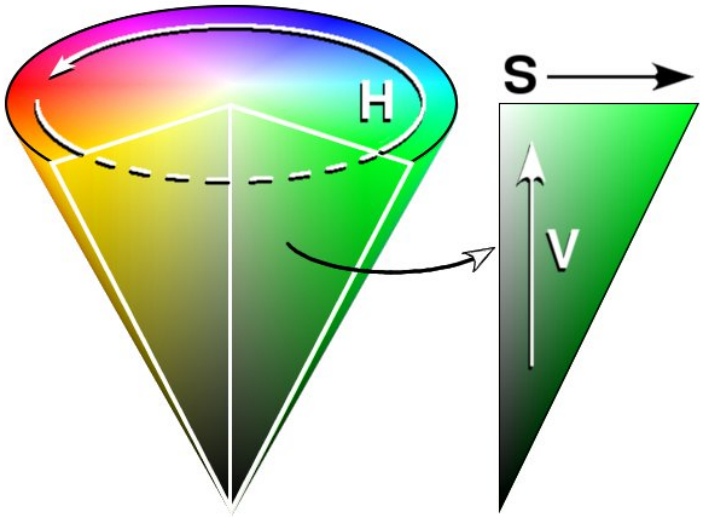
視覚は影による色変化を無視する機能がある

→ 影による色変化を無視できる色空間がいい

ref. (2020/6/1):
Wikimedia commons: Grey square optical illusion.PNG, Copyrighted free use, https://commons.wikimedia.org/wiki/File:Grey_square_optical_illusion.PNG
Wikimedia commons: Same color illusion proof2.png, Copyrighted free use, https://commons.wikimedia.org/wiki/File:Same_color_illusion_proof2.png

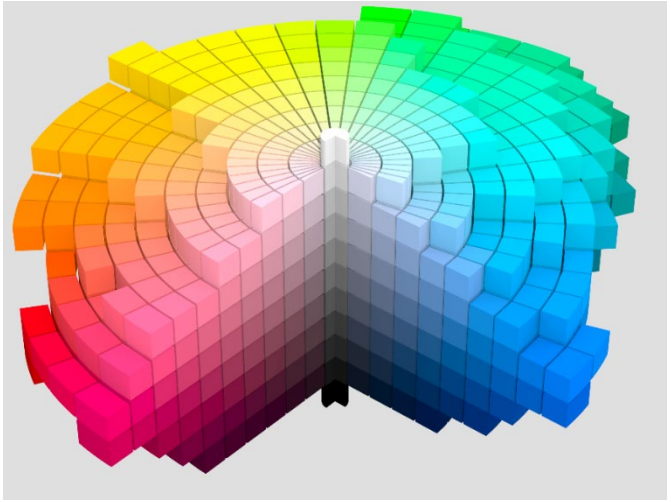
直感的な色の近さを表現する色空間：HSV表色系

- HSV: 色相 (Hue)、彩度 (Saturation)、明度 (Value)
- 美術でよく目にするマンセル表色系と似た表現
  - マンセル表色系は明度(V)最大のときは常に白
  - HSV表色系は明度(V)最大のとき色相による違いがある
- 色相 (Hue)は循環することに注意 (0~360度で表現した場合、赤色が0度付近または360度付近になるが、これらは連続している)



HSV表色系

ref. (2020/4/3): Wikimedia commons: File:HSV cone.jpg CC BY-SA 3.0
https://commons.wikimedia.org/wiki/File:HSV_cone.jpg



マンセル表色系

ref. (2020/4/3): Wikimedia commons: File:Munsell 1943 color solid cylindrical coordinates gray.png CC BY-SA 3.0
https://commons.wikimedia.org/wiki/File:Munsell_1943_color_solid_cylindrical_coordinates_gray.png

英語名称のいろいろ

- 色相
  - Hue
- 彩度
  - Saturation
  - Chroma
- 明度
  - Value
  - Intensity
  - Lightness

HSV
HSI
HSL
HVC
etc.

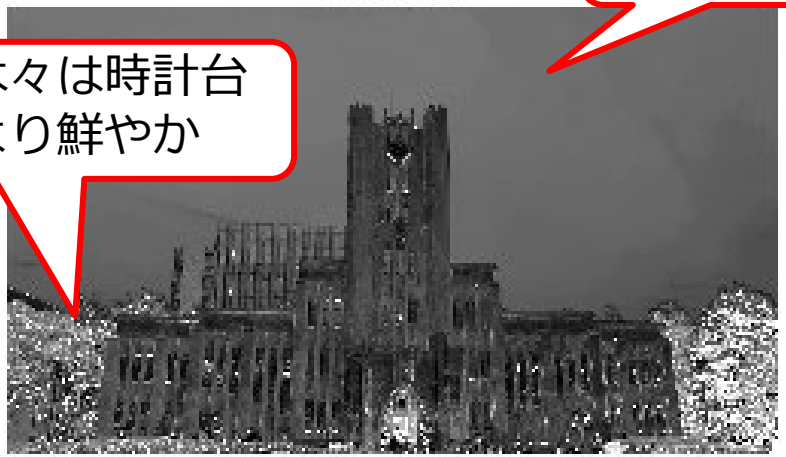
カラー画像をHSV空間に変換

original image

H image

時計台は赤

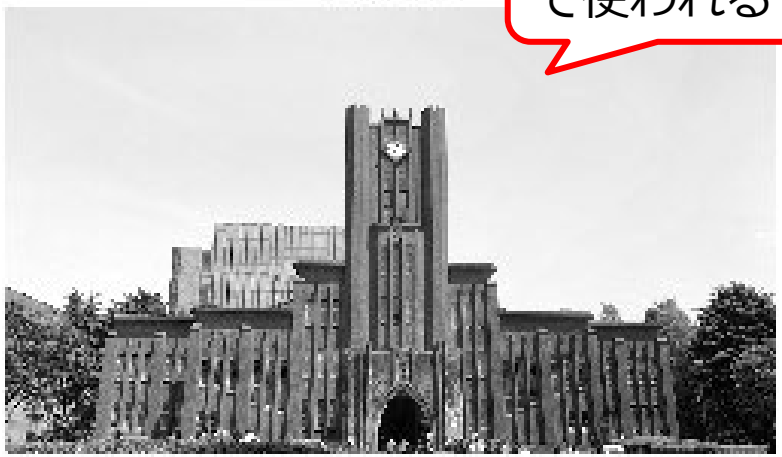
S image



明るい方が鮮やか

木々は時計台より鮮やか

V image



白黒写真として使われる

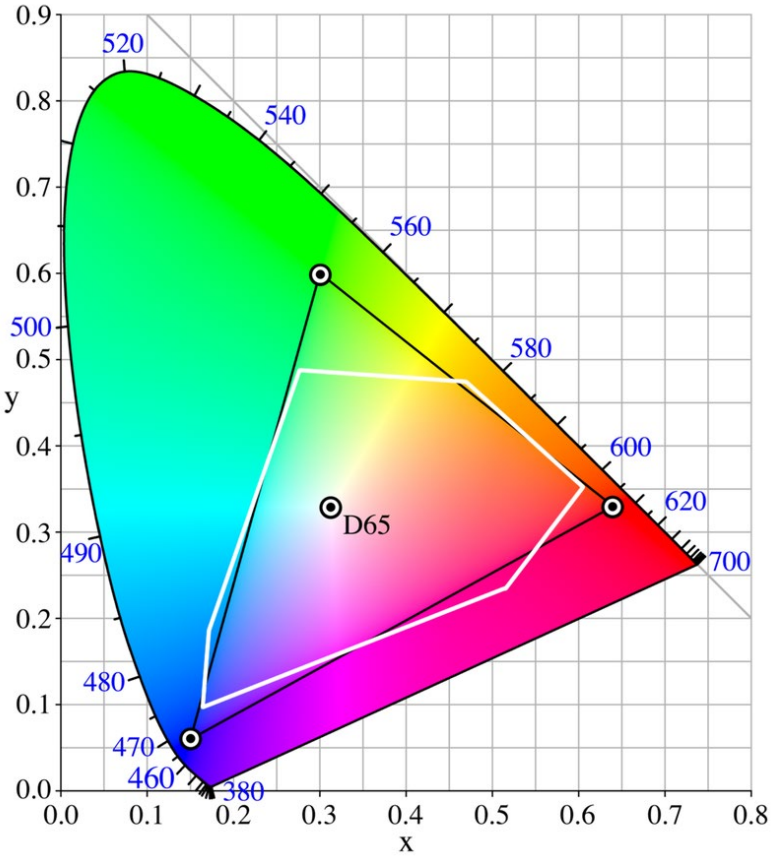
HSV空間における領域抽出

HSV色空間は色を指定することにより領域を抽出する場合によく使われる

色相（H)が30よりも小さい、あるいは160よりも大きく、彩度（S）が5よりも大きい領域を抽出後、穴を埋めた領域をマスクとして画像を抜き出したも

様々な色空間

- RGB表色系は視覚が認知できるすべての色を表現できない
  - 一般的なモニタの規格であるsRGBは右図の三角の領域
  - 可視光波長はおよそ400～800nm（右図の着色された領域）
  - RGBがマイナスの値を取れるとするならば表現可能だが、わかりづらい
- あらゆる可視光色が表現できる空間として、XYZ表色系がある
- その他、モニタ、印刷、デザイン、放送、デジタルシネマなどの目的や、様々な規格化団体によって制定された色空間がたくさんある



ref. (2020/4/4): Wikimedia commons: CIExy1931 sRGB CMY.png GFDL https://ja.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:CIExy1931_sRGB_CMY.png

色恒常性
(color constancy)

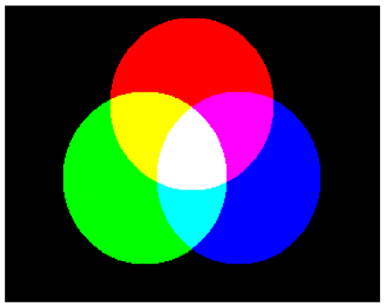
ネットショップで写真を見て買い物をしよう！

白熱球を当てて撮影

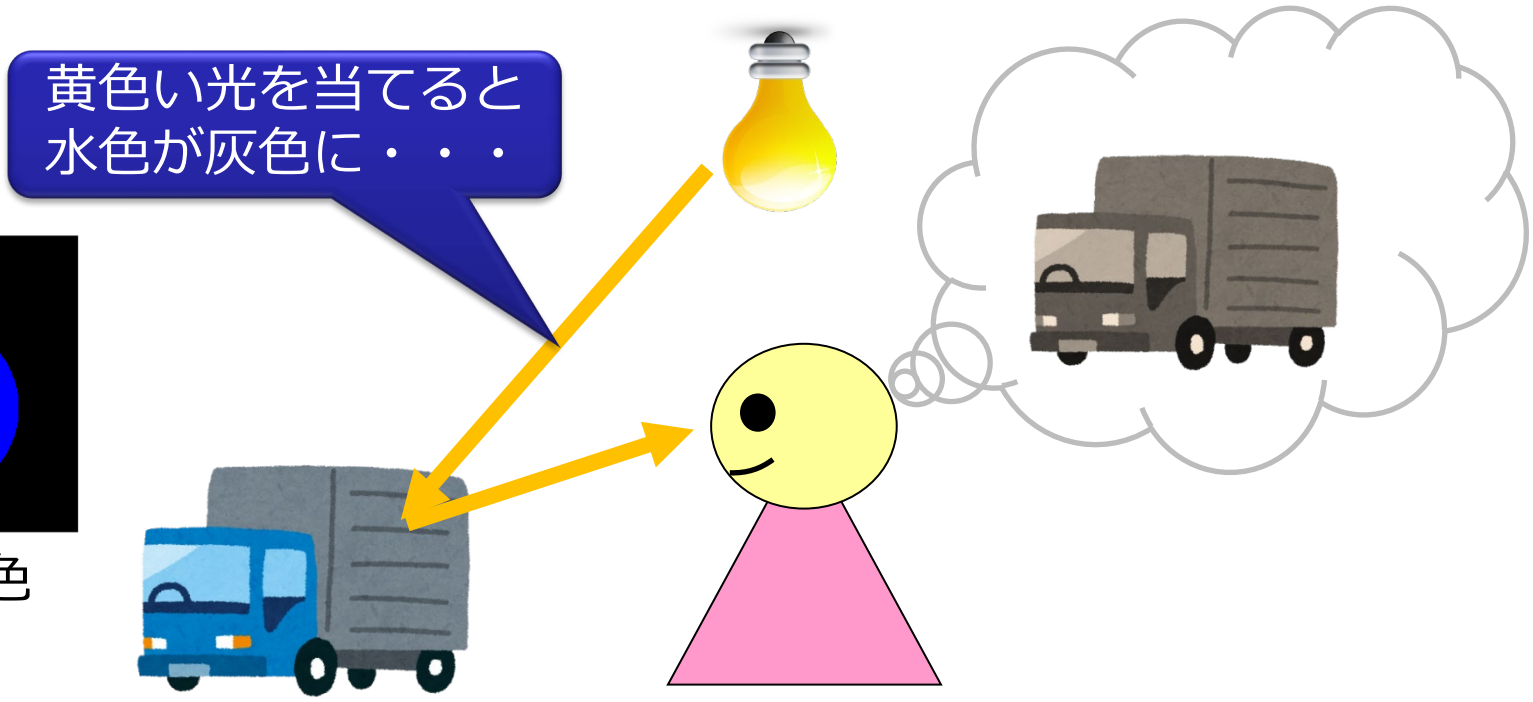
蛍光灯を当てて撮影

左の写真を見て商品を買った人が、届いた商品を
見てクレームを送ってくるかもしれません...

物体の色は物体表面で反射した光の色



光の三原色



照明の色によって同じ物体でも画像の色が変わる

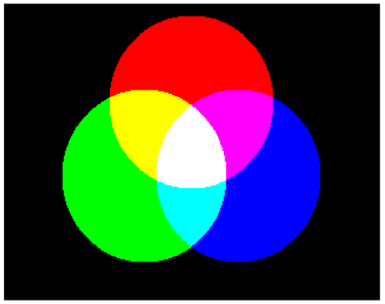
赤：ロウソクの光→白熱球→蛍光灯→太陽光(曇り)→太陽光(青空)：青

ref. (2020/4/3): Wikimedia commons: File:Color temperature.svg CC BY-SA 2.5 pl https://commons.wikimedia.org/wiki/File:Color_temperature.svg
ref. (2020/4/3): いらすとや「トラックのイラスト（車）」 https://www.irasutoya.com/2013/05/blog-post_6635.html

無色（白）とは何か？

いろいろな波長の光が万遍なく見えているはず！

- 人間の視覚は、見えている色をすべて足せば白になると考えて、そうなるように色覚を調整：ホワイトバランス
- 見えているものの色が偏ると、ホワイトバランスも偏る
- 赤いものばかり見続けて、急に白い意物体を見ると、シアン（赤の補色）に見える



手術着はなぜ青や緑なのか？

赤い血や内臓を見続けると、赤を白と知覚しようとすることで目がちらつくため、緑や青の手術着を切ることで補正する

ref. (2020/4/3): Wikimedia commons: File:Surgeons at Work.jpg,パブリック・ドメイン, https://commons.wikimedia.org/wiki/File:Surgeons_at_Work.jpg

どうしたら色の恒常性が担保できるのか？

- 標準色票（カラーチェッカー）：
  規格に従い無光沢単色で塗られたタイルで構成
- 商品を撮影するときと同じ照明環境で標準色票を撮影しておく
- 撮影した写真のうち、矩形の領域色が規格の数値(RGB)と一致するよう写真の色を補正
  (Photoshop等、写真編集ソフトには補正機能が内蔵)

ref. (2020/4/3): Wolfgang Lonien, 7e0_4053916-zuleikha-colorchecker CC BY-SA 2.0
https://www.flickr.com/photos/wjlonien/26165090302/

規格JIS Z 8721

No.	Name	Max255		
		R	G	B
1	Dark Skin	115	82	68
2	Light Skin	194	150	130
3	Blur Sky	98	122	157
4	Foliage	87	108	67
5	Blue Flower	133	128	177
6	Bluish Green	103	189	170
7	Orange	214	126	44
8	Purplish Blue	80	91	166
9	Moderate Red	193	90	99
10	Purple	94	60	108
11	Yellow Green	157	188	64
12	Orange Yellow	224	163	46
13	Blue	56	61	150
14	Green	70	148	73
15	Red	175	54	60
16	Yellow	231	199	31
17	Magenta	187	86	149
18	Cyan	8	133	161
19	White	243	243	242
20	Neutral 8	200	200	200
21	Neutral 6.5	160	160	160
22	Neutral 5	122	122	121
23	Neutral 3.5	85	85	85
24	Black	52	52	52

二次元デジタルフィルタによる画像処理

演習）ImageProcessing4.ipynb

画像のフィルタリング

ref. (2020/4/4)
https://www.pexels.com/ja-jp/photo/736230/

- デジタル画像を平面座標系において座標(x, y)が決まると、その座標における濃淡値 $f(x,y)$を返す関数と考える（ただし、x, y, zは0以上の整数）
- 入力画像 $f(x,y)$に対し、何らかのフィルタを適用して、出力画像 $g(x,y)$を生成することを考える

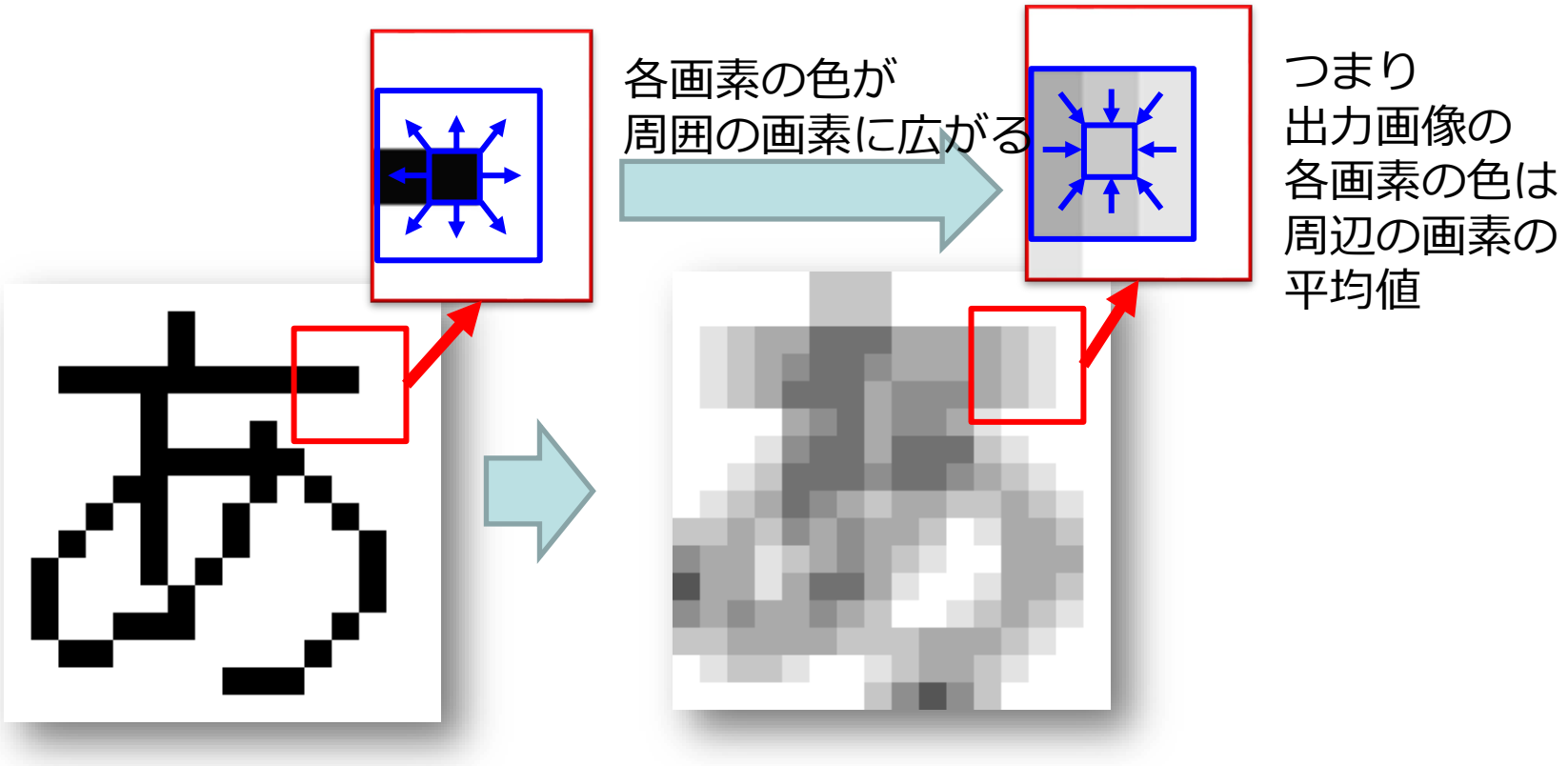
入力画像 $f(x,y)$

ケーススタディ：
画像を
ぼけさせる
フィルタとは？

出力画像 $g(x,y)$

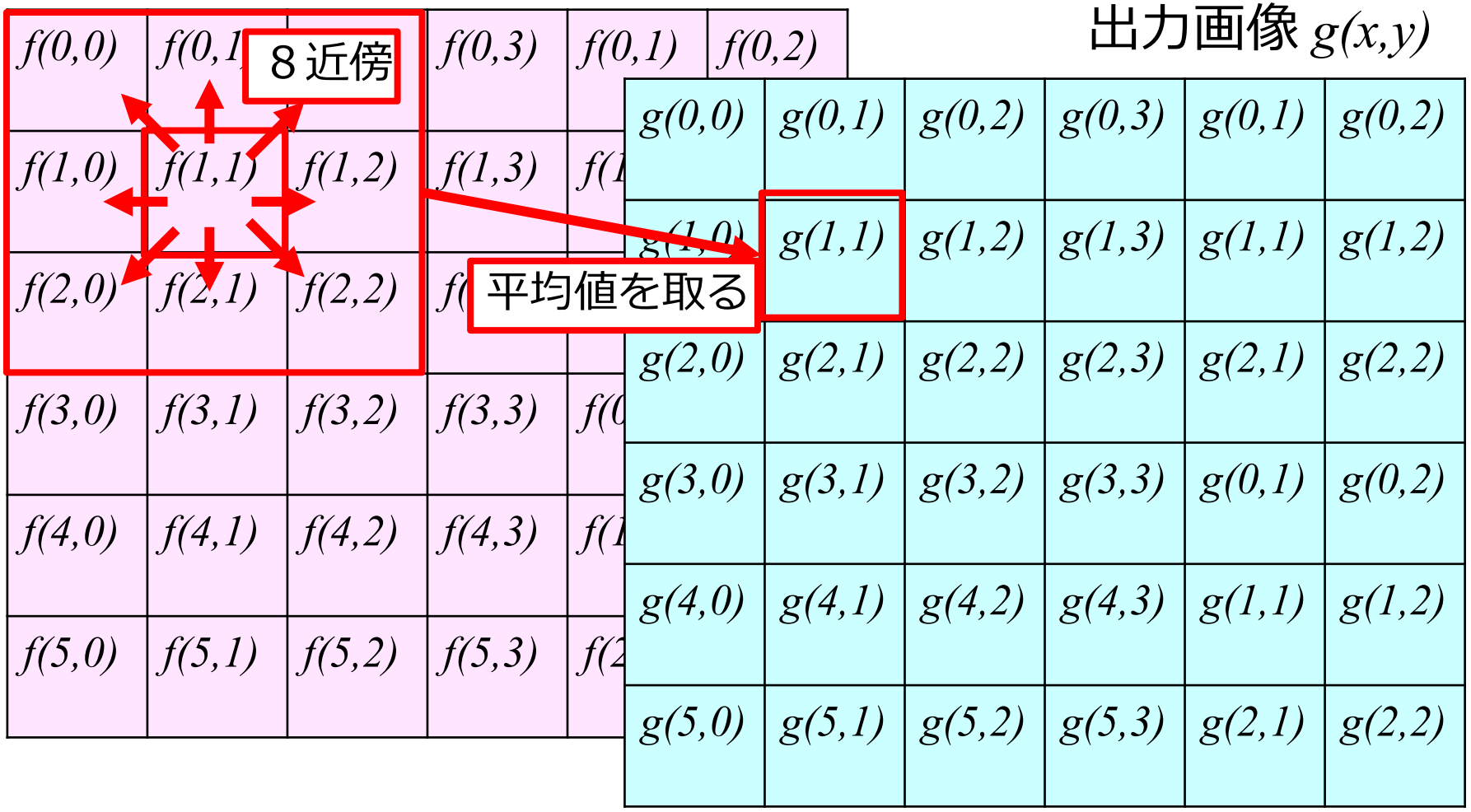
ケーススタディ：画像をぼけさせるには？

- 「画像がぼける」とは、各点の色が周囲に広がること
- 逆に言えば「ぼけた画像」の各点の色は、
  その周辺の点の色が少しずつ重なったもの
  →入力画像における同じ位置の画素に注目したとき、
  その画素を含む周辺の画素の画素値の平均値が出力画像の画素値



平均値フィルタ (Mean filter)

入力画像 $f(x,y)$



平均値フィルタ (Mean filter)

入力画像 $f(x,y)$

$f(0,0)$	$f(0,1)$	$f(0,2)$	$f(0,3)$	$f(0,1)$	$f(0,2)$
$f(1,0)$	$f(1,1)$	$f(1,2)$	$f(1,3)$	$f($	
$f(2,0)$	$f(2,1)$	$f(2,2)$	$f(2,3)$	$f($	
$f(3,0)$	$f(3,1)$	$f(3,2)$	$f(3,3)$	$f($	
$f(4,0)$	$f(4,1)$	$f(4,2)$	$f(4,3)$	$f($	
$f(5,0)$	$f(5,1)$	$f(5,2)$	$f(5,3)$	$f($	

出力画像 $g(x,y)$

$g(0,0)$	$g(0,1)$	$g(0,2)$	$g(0,3)$	$g(0,1)$	$g(0,2)$
$g(1,0)$	$g(1,1)$	$g(1,2)$	$g(1,3)$	$g(1,1)$	$g(1,2)$
$g(2,0)$	$g(2,1)$	$g(2,2)$	$g(2,3)$	$g(2,1)$	$g(2,2)$
$g(3,0)$	$g(3,1)$	$g(3,2)$	$g(3,3)$	$g(0,1)$	$g(0,2)$
$g(4,0)$	$g(4,1)$	$g(4,2)$	$g(4,3)$	$g(1,1)$	$g(1,2)$
$g(5,0)$	$g(5,1)$	$g(5,2)$	$g(5,3)$	$g(2,1)$	$g(2,2)$

平均を取る

平均値フィルタ (Mean filter)

入力画像 $f(x,y)$

$f(0,0)$	$f(0,1)$	$f(0,2)$	$f(0,3)$	$f(0,1)$	$f(0,2)$
$f(1,0)$	$f(1,1)$	$f(1,2)$	$f(1,3)$	$f(1$	
$f(2,0)$	$f(2,1)$	$f(2,2)$	$f(2,3)$	$f(2$	
$f(3,0)$	$f(3,1)$	$f(3,2)$	$f(3,3)$	$f(0$	
$f(4,0)$	$f(4,1)$	$f(4,2)$	$f(4,3)$	$f(1$	
$f(5,0)$	$f(5,1)$	$f(5,2)$	$f(5,3)$	$f(2$	

出力画像 $g(x,y)$

$g(0,0)$	$g(0,1)$	$g(0,2)$	$g(0,3)$	$g(0,1)$	$g(0,2)$
$g(1,0)$	$g(1,1)$	$g(1,2)$	$g(1,3)$	$g(1,1)$	$g(1,2)$
$g(2,0)$	$g(2,1)$	$g(2,2)$	$g(2,3)$	$g(2,1)$	$g(2,2)$
$g(3,0)$	$g(3,1)$	$g(3,2)$	$g(3,3)$	$g(0,1)$	$g(0,2)$
$g(4,0)$	$g(4,1)$	$g(4,2)$	$g(4,3)$	$g(1,1)$	$g(1,2)$
$g(5,0)$	$g(5,1)$	$g(5,2)$	$g(5,3)$	$g(2,1)$	$g(2,2)$

平均を取る

平均値フィルタ (Mean filter)

入力画像 $f(x,y)$

出力画像 $g(x,y)$

平均値フィルタ (Mean filter)

入力画像 $f(x,y)$

$f(0,0)$	$f(0,1)$	$f(0,2)$	$f(0,3)$	$f(0,1)$	$f(0,2)$
$f(1,0)$	$f(1,1)$	$f(1,2)$	$f(1,3)$	$f($	
$f(2,0)$	$f(2,1)$	$f(2,2)$	$f(2,3)$	$f($	
$f(3,0)$	$f(3,1)$	$f(3,2)$	$f($		
$f(4,0)$	$f(4,1)$	$f(4,2)$	$f(4,3)$	$f($	
$f(5,0)$	$f(5,1)$	$f(5,2)$	$f(5,3)$	$f($	

平均を取る

出力画像 $g(x,y)$

$g(0,0)$	$g(0,1)$	$g(0,2)$	$g(0,3)$	$g(0,1)$	$g(0,2)$
$g(1,0)$	$g(1,1)$	$g(1,2)$	$g(1,3)$	$g(1,1)$	$g(1,2)$
$g($	$g(2,1)$	$g(2,2)$	$g(2,3)$	$g(2,1)$	$g(2,2)$
$g(3,0)$	$g(3,1)$	$g(3,2)$	$g(3,3)$	$g(0,1)$	$g(0,2)$
$g(4,0)$	$g(4,1)$	$g(4,2)$	$g(4,3)$	$g(1,1)$	$g(1,2)$
$g(5,0)$	$g(5,1)$	$g(5,2)$	$g(5,3)$	$g(2,1)$	$g(2,2)$

平均値フィルタ (Mean filter)

入力画像 $f(x,y)$

$f(0,0)$	$f(0,1)$	$f(0,2)$	$f(0,3)$	$f(0,1)$	$f(0,2)$
$f(1,0)$	$f(1,1)$	$f(1,2)$	$f(1,3)$	$f($	
$f(2,0)$	$f(2,1)$	$f(2,2)$	$f(2,3)$	$f($	
$f(3,0)$	$f(3,1)$	$f(3,2)$	$f(3,3)$	$f($	
$f(4,0)$	$f(4,1)$	$f(4,2)$	$f(4,3)$	$f($	
$f(5,0)$	$f(5,1)$	$f(5,2)$	$f(5,3)$	$f($	

出力画像 $g(x,y)$

$g(0,0)$	$g(0,1)$	$g(0,2)$	$g(0,3)$	$g(0,1)$	$g(0,2)$
$g(1,0)$	$g(1,1)$	$g(1,2)$	$g(1,3)$	$g(1,1)$	$g(1,2)$
$g(2,0)$	$g(2,1)$	$g(2,2)$	$g(2,3)$	$g(2,1)$	$g(2,2)$
$g(3,0)$	$g(3,1)$	$g(3,2)$	$g(3,3)$	$g(0,1)$	$g(0,2)$
$g(4,0)$	$g(4,1)$	$g(4,2)$	$g(4,3)$	$g(1,1)$	$g(1,2)$
$g(5,0)$	$g(5,1)$	$g(5,2)$	$g(5,3)$	$g(2,1)$	$g(2,2)$

平均を取る

平均値フィルタ (Mean filter)

入力画像 $f(x,y)$

$f(0,0)$	$f(0,1)$	$f(0,2)$	$f(0,3)$	$f(0,1)$	$f(0,2)$
$f(1,0)$	$f(1,1)$	$f(1,2)$	$f(1,3)$	$f(1$	
$f(2,0)$	$f(2,1)$	$f(2,2)$	$f(2,3)$	$f(2$	
$f(3,0)$	$f(3,1)$	$f(3,2)$	$f(3,3)$	$f(0$	
$f(4,0)$	$f(4,1)$	$f(4,2)$	$f(4,3)$	$f(1$	
$f(5,0)$	$f(5,1)$	$f(5,2)$	$f(5,3)$	$f(2$	

出力画像 $g(x,y)$

$g(0,0)$	$g(0,1)$	$g(0,2)$	$g(0,3)$	$g(0,1)$	$g(0,2)$
$g(1,0)$	$g(1,1)$	$g(1,2)$	$g(1,3)$	$g(1,1)$	$g(1,2)$
$g(2,0)$	$g(2,1)$	$g(2,2)$	$g(2,3)$	$g(2,1)$	$g(2,2)$
$g(3,0)$	$g(3,1)$	$g(3,2)$	$g(3,3)$	$g(0,1)$	$g(0,2)$
$g(4,0)$	$g(4,1)$	$g(4,2)$	$g(4,3)$	$g(1,1)$	$g(1,2)$
$g(5,0)$	$g(5,1)$	$g(5,2)$	$g(5,3)$	$g(2,1)$	$g(2,2)$

平均を取る

平均値フィルタ (Mean filter)

入力画像 $f(x,y)$

$f(0,0)$	$f(0,1)$	$f(0,2)$	$f(0,3)$	$f(0,1)$	$f(0,2)$
$f(1,0)$	$f(1,1)$	$f(1,2)$	$f(1,3)$	$f(1$	
$f(2,0)$	$f(2,1)$	$f(2,2)$	$f(2,3)$	$f(2$	
$f(3,0)$	$f(3,1)$	$f(3,2)$	$f(3,3)$	$f(0$	
$f(4,0)$	$f(4,1)$	$f(4,2)$	$f(4,3)$	$f(1$	
$f(5,0)$	$f(5,1)$	$f(5,2)$	$f(5,3)$	$f(2$	

出力画像 $g(x,y)$

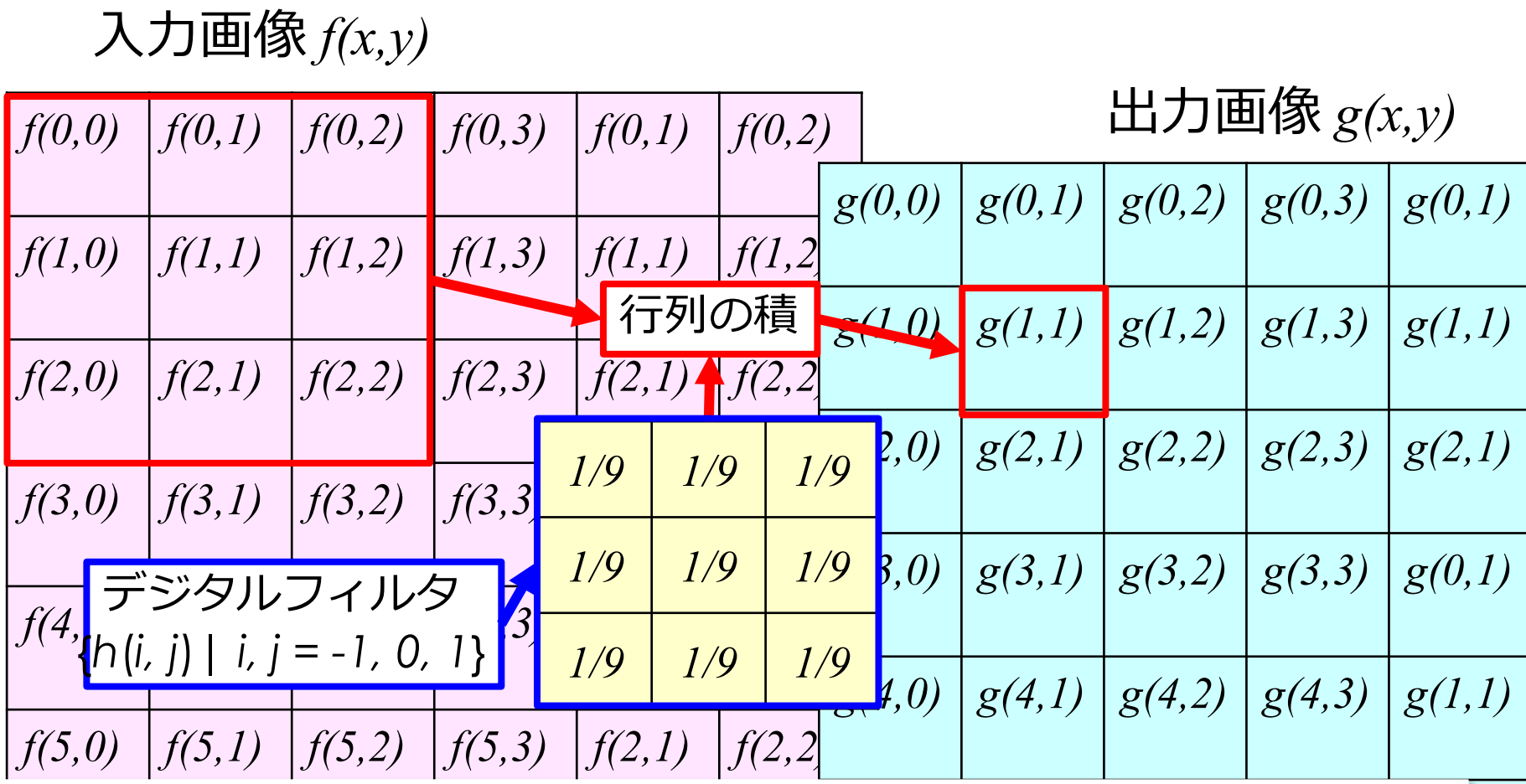
$g(0,0)$	$g(0,1)$	$g(0,2)$	$g(0,3)$	$g(0,1)$	$g(0,2)$
$g(1,0)$	$g(1,1)$	$g(1,2)$	$g(1,3)$	$g(1,1)$	$g(1,2)$
$g(2,0)$	$g(2,1)$	$g(2,2)$	$g(2,3)$	$g(2,1)$	$g(2,2)$
$g(3,0)$	$g(3,1)$	$g(3,2)$	$g(3,3)$	$g(0,1)$	$g(0,2)$
$g(4,0)$	$g(4,1)$	$g(4,2)$	$g(4,3)$	$g(1,1)$	$g(1,2)$

平均を取る

入力画像を $f(x, y)$
出力画像を $g(x, y)$ とすると、

$$g(n,m) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \frac{f(n+i, m+j)}{9}$$

平均値フィルタの拡張： 2次元デジタルフィルタ



iとjがそれぞれ$(-1, 0, 1)$のいずれかのとき$h(i, j)=1/9$とすると、

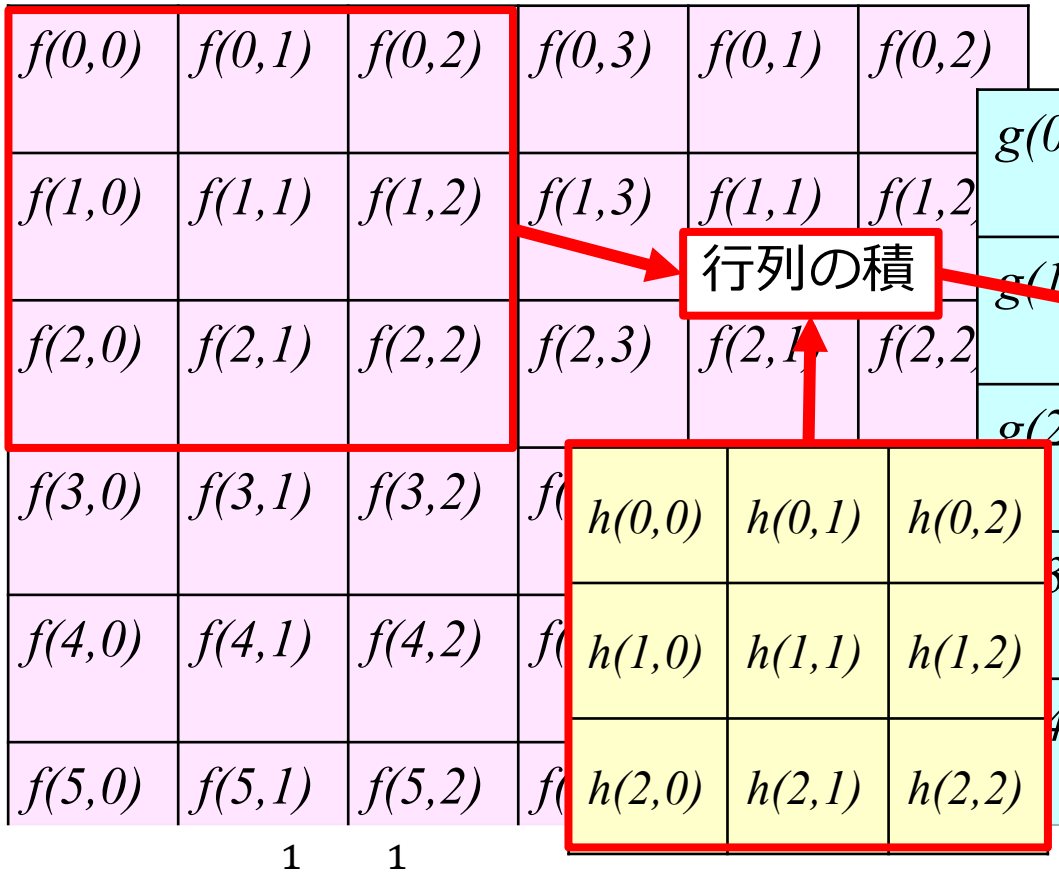
$$g(n,m) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j) f(n+i, m+j)$$

と書き換えられる

2次元デジタルフィルタの適用

入力画像 $f(x,y)$

出力画像 $g(x,y)$



$$g(n,m) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j) f(n+i, m+j)$$

2次元デジタルフィルタの適用

入力画像 $f(x,y)$

出力画像 $g(x,y)$

$f(0,0)$	$f(0,1)$	$f(0,2)$	$f(0,3)$	$f(0,1)$	$f(0,2)$
$f(1,0)$	$f(1,1)$	$f(1,2)$	$f(1,3)$	$f(1,1)$	$f(1,2$
$f(2,0)$	$f(2,1)$	$f(2,2)$	$f(2,3)$	$f(2,1)$	$f(2,2$
$f(3,0)$	$f(3,1)$	$f(3,2)$	$f(3,3)$	$f($	
$f(4,0)$	$f(4,1)$	$f(4,2)$	$f(4,3)$	$f($	
$f(5,0)$	$f(5,1)$	$f(5,2)$	$f(5,3)$	$f($	

行列の積

$g(0,0)$	$g(0,1)$	$g(0,2)$	$g(0,3)$	$g(0,1)$
	$g(1,1)$	$g(1,2)$	$g(1,3)$	$g(1,1)$
$g(2,0)$	$g(2,1)$	$g(2,2)$	$g(2,3)$	$g(2,1)$
	$,1)$	$g(3,2)$	$g(3,3)$	$g(0,1)$
	$,1)$	$g(4,2)$	$g(4,3)$	$g(1,1)$
				$1)$

$h(0,0)$	$h(0,1)$	$h(0,2)$
$h(1,0)$	$h(1,1)$	$h(1,2)$
$h(2,0)$	$h(2,1)$	$h(2,2)$

$$g(n,m) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j) f(n+i, m+j)$$

2次元デジタルフィルタの適用

入力画像 $f(x,y)$

$f(0,0)$	$f(0,1)$	$f(0,2)$	$f(0,3)$	$f(0,1)$	$f(0,2)$
$f(1,0)$	$f(1,1)$	$f(1,2)$	$f(1,3)$	$f(1,1)$	$f(1,2$
$f(2,0)$	$f(2,1)$	$f(2,2)$	$f(2,3)$	$f(2,1)$	$f(2,2$
$f(3,0)$	$f(3,1)$	$f(3,2)$	$f(3,3)$	$f(0,1)$	$f($
$f(4,0)$	$f(4,1)$	$f(4,2)$	$f(4,3)$	$f(1,1)$	$f($
$f(5,0)$	$f(5,1)$	$f(5,2)$	$f(5,3)$	$f(2,1)$	$f($

出力画像 $g(x,y)$

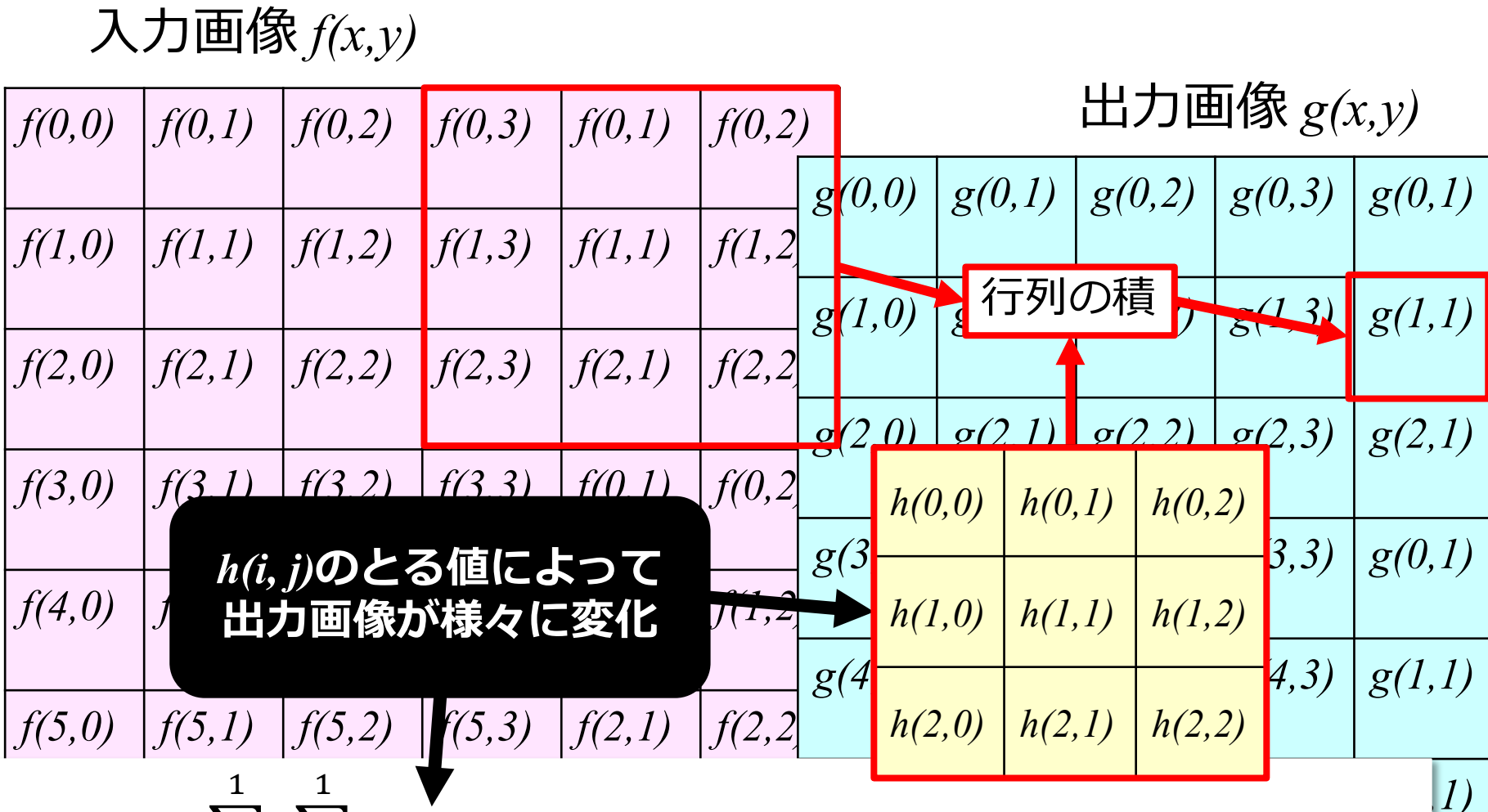
$g(0,0)$	$g(0,1)$	$g(0,2)$	$g(0,3)$	$g(0,1)$
g		$g(1,2)$	$g(1,3)$	$g(1,1)$
$g(2,0)$	$g(2,1)$	$g(2,2)$	$g(2,3)$	$g(2,1)$
		$,2)$	$g(3,3)$	$g(0,1)$
		$,2)$	$g(4,3)$	$g(1,1)$
				$1)$

行列の積

$h(0,0)$	$h(0,1)$	$h(0,2)$
$h(1,0)$	$h(1,1)$	$h(1,2)$
$h(2,0)$	$h(2,1)$	$h(2,2)$

$$g(n,m) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j) f(n+i, m+j)$$

2次元デジタルフィルタの適用



$$g(n,m) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} h(i,j) f(n+i, m+j)$$

→ h の f に対する畳み込み演算と呼ぶ

2次元デジタルフィルタにおける畳み込み演算

$h(0,0)$	$h(0,1)$	$h(0,2)$
$h(1,0)$	$h(1,1)$	$h(1,2)$
$h(2,0)$	$h(2,1)$	$h(2,2)$

3×3のカーネル

$$g(n,m) = \sum_i \sum_j h(i,j) f(n+i, m+j)$$

出力画像の明るさを変えたくない場合は、

$$\sum_i \sum_j h(i,j) = 1$$

- 一般的にフィルタのカーネルは奇数×奇数画素の正方行列
- それぞれの画素を中心とし、その画素とその周辺からなる画素集合とカーネルとの行列積を、新しい画像の画素値とする処理を「**畳み込み（convolution)**」と呼ぶ
- **畳み込み**はDeep Learningによる画像処理の基本的な処理（CNN=**Convolutional** Neural Network）
- フィルタによって出力画像が様々に変化

Question 1

入力画像にフィルタHをかけた画像はどれ？

1	0	0	1
1	0	0	1
1	0	1	1
1	0	1	1

入力画像

*

-1	-1	-1
-1	8	-1
-1	-1	-1

フィルタH

→

-	-	-	-
-	A	B	-
-	C	D	-
-	-	-	-

出力画像

-	-	-	-
-	-4	-4	-
-	-5	4	-
-	-	-	-

平均値フィルタ（Mean filter)

ref. (2020/4/4)
https://www.pexels.com/ja-jp/photo/736230/

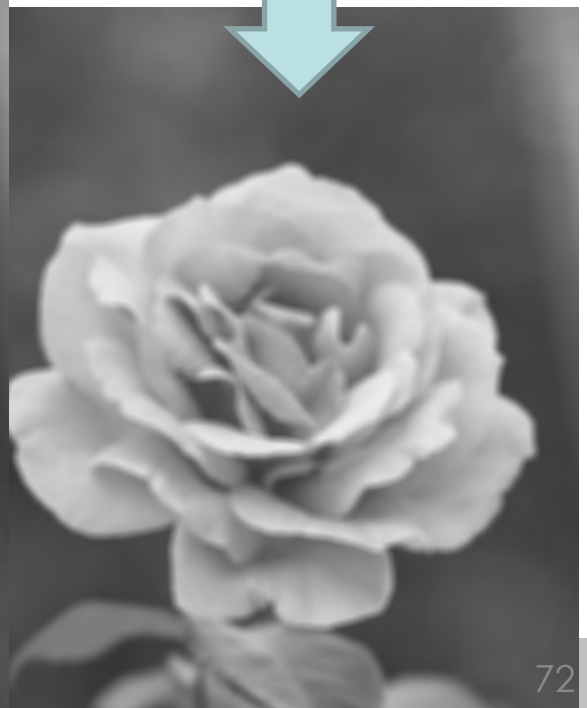
3x3 平均値フィルタ

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

5 x5 平均値フィルタ

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

入力画像(320x480)



鮮鋭化フィルタ

ref. (2020/4/4)
https://www.pexels.com/ja-jp/photo/736230/

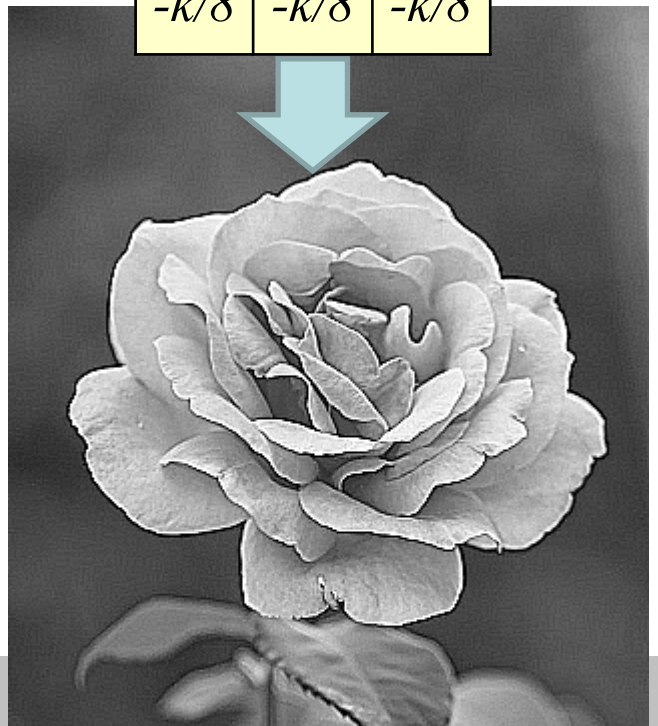
輪郭を際立たせ、鮮鋭にするフィルタ

- その画素と周辺の画素の値が違うとき、その画素の値を大きくする
- その画素と周辺の画素の値が同じ時、その画素の元の値をそのまま使う

3x3鮮鋭化フィルタ
k=5とすると
下図のようになる

$-k/8$	$-k/8$	$-k/8$
$-k/8$	$1+k$	$-k/8$
$-k/8$	$-k/8$	$-k/8$

入力画像 (320x480)



エッジ抽出：ラプラシアンフィルタ

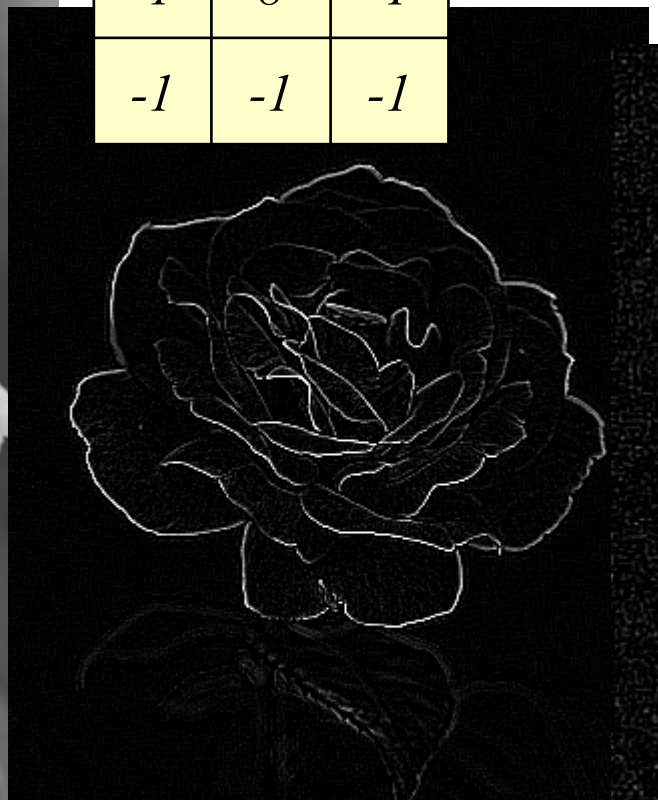
ref. (2020/4/4)
https://www.pexels.com/ja-jp/photo/736230/

その画素の値が周辺の画素と違うほど大きな値とすることによって、
周辺との輝度が大きく変化する輪郭線を取り出す

入力画像(320x480)

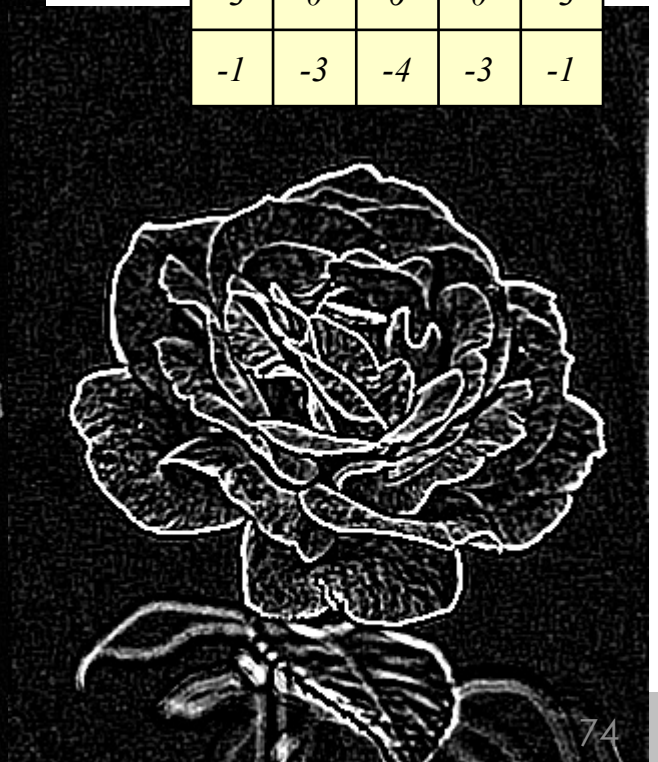
3x3 ラプラシアンフィルタ

-1	-1	-1
-1	8	-1
-1	-1	-1



5x5 ラプラシアンフィルタ

-1	-3	-4	-3	-1
-3	0	6	0	-3
-4	6	20	6	-4
-3	0	6	0	-3
-1	-3	-4	-3	-1



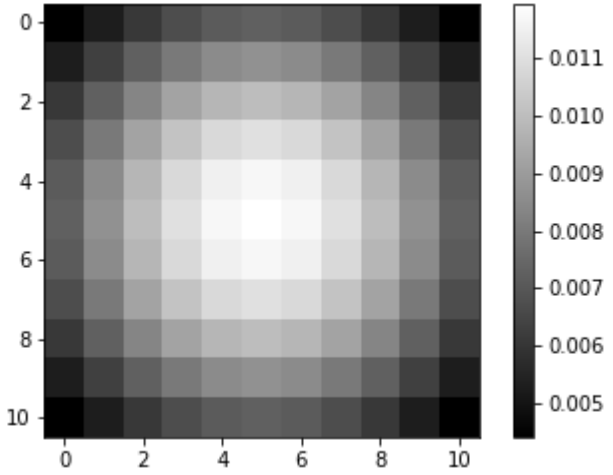
ガウシアンフィルタ

- 中心が最も値が大きく、
  中心から離れるほど値が小さくなる
- x軸方向、y軸方向それぞれの値の変化は
  ガウス関数に従う

$$h(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

- カメラのレンズによるボケを
  幾何学的に再現したモデルと考えられる

11x11のガウシアンフィルタ



サイズ3×3、 σ=2のフィルタ

0.102	0.115	0.102
0.115	0.131	0.115
0.102	0.115	0.102

サイズ５×５、 σ=5のフィルタ

0.0369	0.0392	0.0400	0.0392	0.0369
0.0392	0.0416	0.0424	0.0416	0.0392
0.0400	0.0424	0.0433	0.0424	0.0400
0.0392	0.0416	0.0424	0.0416	0.0392
0.0369	0.0392	0.0400	0.0392	0.0369

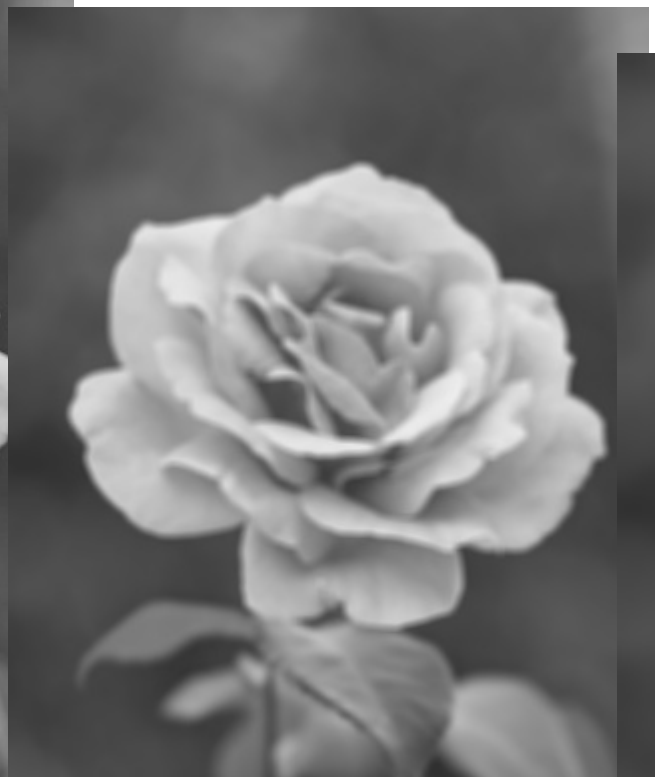
ガウシアンフィルタ

ref. (2020/4/4)
https://www.pexels.com/ja-jp/photo/736230/

その画素の値の影響を最も強く受け、そこから離れた位置にある画素ほど影響を受けなくなる

入力画像(320x480)

ガウシアンフィルタ
サイズ： 5x5
σ=5



ガウシアンフィルタ
サイズ： 11x11
σ=5

Question 2

ref. (2020/4/4)
https://www.pexels.com/ja-jp/photo/736230/

入力画像にフィルタHをかけた画像はどれ？

入力画像

画像B

画像A

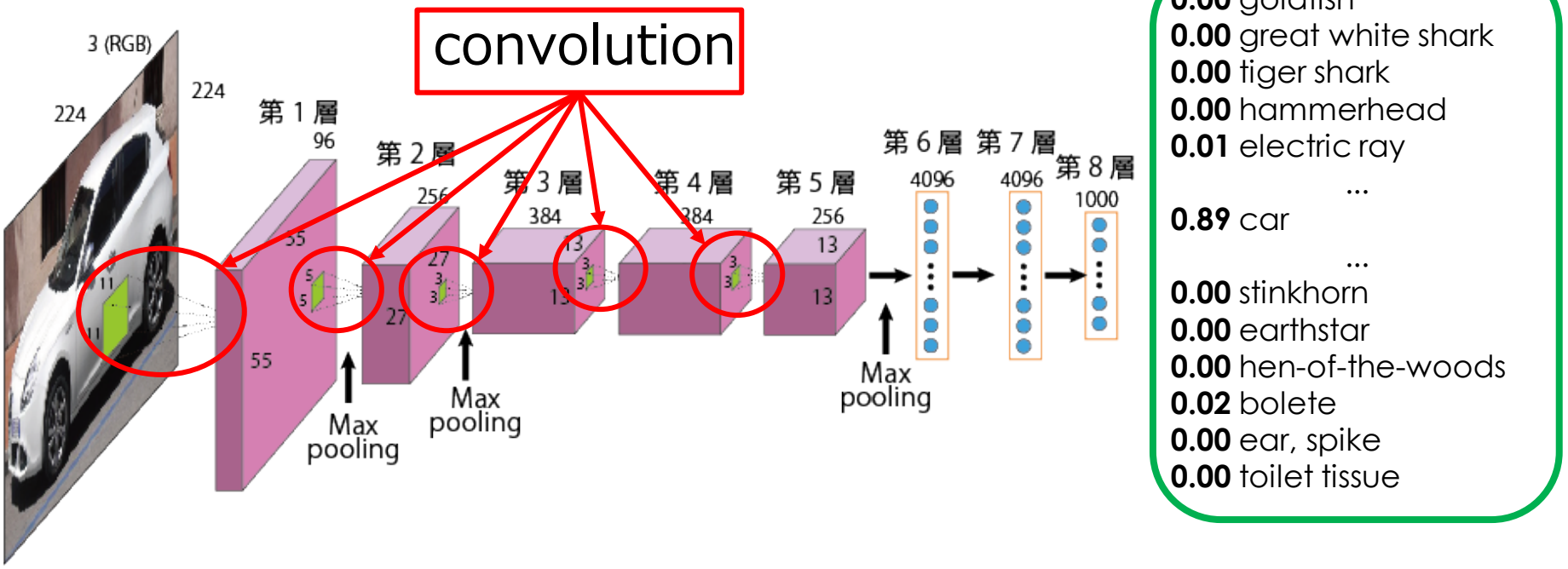
0	-1	0
-1	5	-1
0	-1	0

フィルタH

画像C

畳み込みニューラルネットワークにおける畳み込み演算

- 第1層から第5層までは、2次元デジタルフィルタで説明した**畳み込み演算 (convolution）**を行っている
- ただし、2次元デジタルフィルタではフィルタは人がデザインしていたのに対し、CNNではデータから学習により取得する

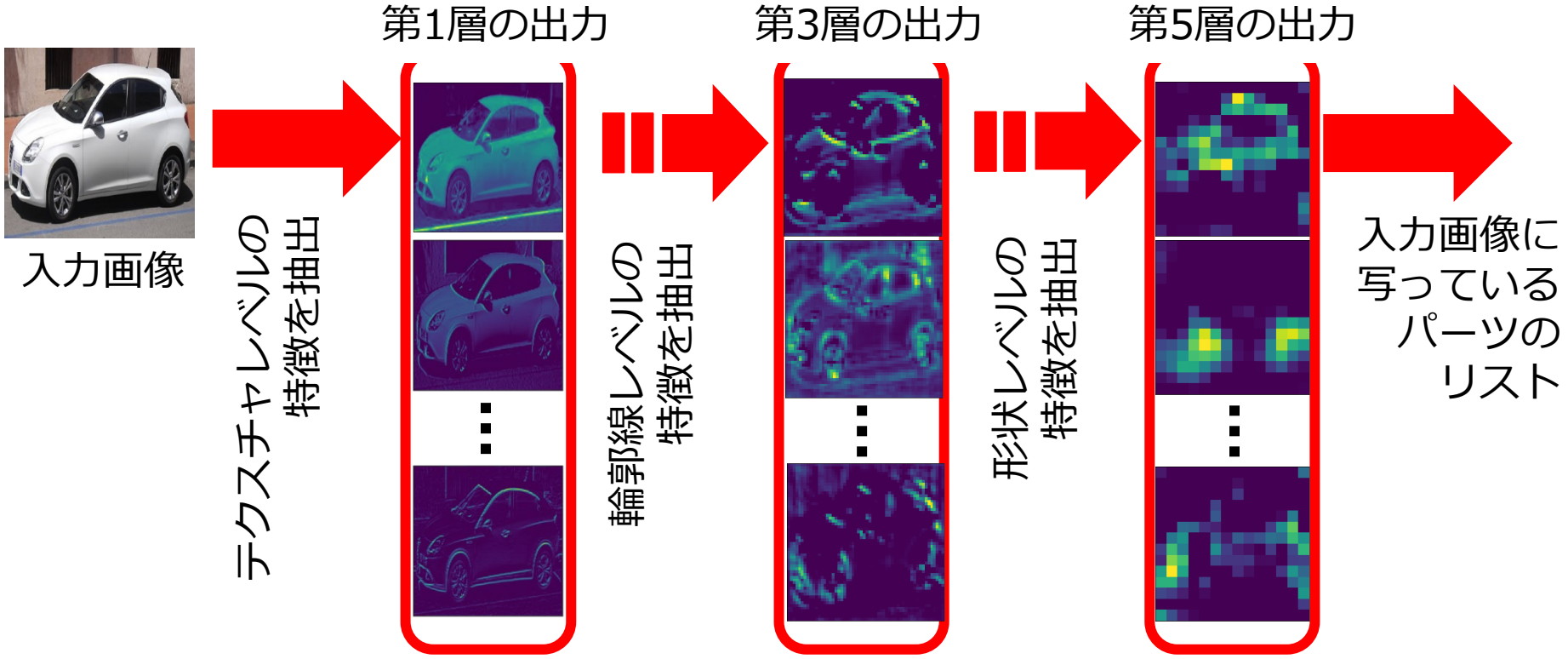


ref. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS2012

ref. (2020/4/6): Wikimedia commons: File:" 10 Alfa Romeo Giulietta white Derivate cut.JPG CC0
https://commons.wikimedia.org/wiki/Category:CC-Zero?uselang=ja#/media/File:%22_10_Alfa_Romeo_Giulietta_white_Derivate_cut.JPG

CNNにおける画像のフィルタリング

- 層を経るごとにより具体的な形状の特徴を取り出していく
- 第5層になると、「入力画像にどんなパーツが写りこんでいるか（実際には尤度分布）」がわかってくる
- 「入力画像に写っているパーツのセット」が「他のクラスに比べ、車にありがちなパーツのセット」であるならば「車」と判別



モデルはKeras VGG16 pretrainedを使用