

データサイエンス概論I & II データサイエンス総論I & II

非構造化データ解析

九州大学 数理・データサイエンス教育研究センター

【再掲】構造化データと非構造化データ

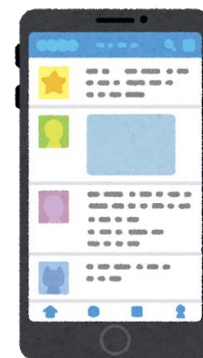
● 構造化データ

- 簡単に言えば、表形式のデータを構造化データ
- 例えば、「横に月・縦に都道府県」を並べた表を作り、それを「ある月のある県での平均降水量」で埋めたとすれば、それは構造化データ

	A	E	C	D	B	F	G	H	I	J	K	L	M	N	O
1	1-8 降水量 (平年値) (昭和56年～平成22年)														
2															
3	観測地点	降水量 (mm)													
4		年計	1月	2月	3月	4月	5月	6月	7月	8月	9月	10月	11月	12月	
6	札幌	1,107	114	94	78	57	53	47	81	124	135	109	104	112	112
7	青森	1,300	145	111	70	63	81	76	117	123	123	104	138	151	
44	高知	2,548	59	106	190	244	292	346	328	283	350	166	125	58	
45	福岡	1,612	68	72	113	117	143	255	278	172	178	74	65	60	
46	佐賀	1,870	57	78	129	156	198	339	339	197	180	76	76	48	
47	長崎	1,858	64	86	132	151	179	315	314	195	189	86	86	61	
48	熊本	1,986	60	83	138	146	196	405	401	174	170	79	81	54	
49	大分	1,645	45	65	112	129	150	274	253	172	220	121	69	34	
50	宮崎	2,509	64	91	182	213	239	429	309	290	355	182	95	60	
51	鹿児島	2,266	78	112	180	205	221	452	319	223	211	102	92	71	
52	那覇	2,041	107	120	161	166	232	247	141	241	261	153	110	103	
55	資料 気象庁「2010年平年値」														

● 非構造化データ

- 文章，画像，音がその代表例
- 「表形式」にはならないので「非構造化データ」と呼ばれる
- スマートフォンやパソコンで日々読んだり見たり聞いたりしているが、これらもデータ

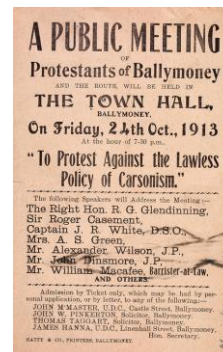


今回の話

画像データ

皆さんの身の回りにある画像データ

- カメラ画像
- 文字，文書，記号，標識，ナンバープレート
- 顔，指紋，虹彩，耳，唇，掌の静脈
- CT・MRI・X線などの医用画像



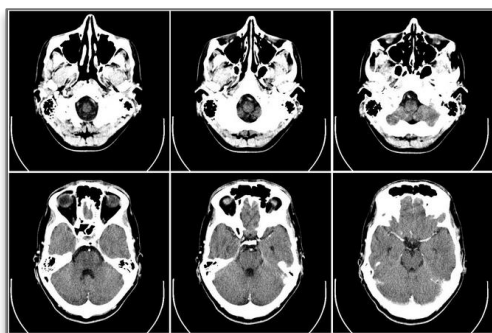
commons@flickr



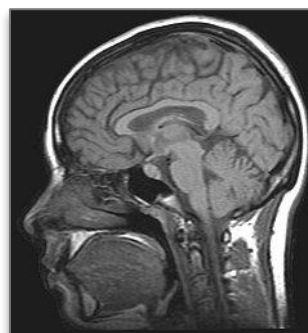
IAM face dataset



@wikipedia



CT画像@wikipedia

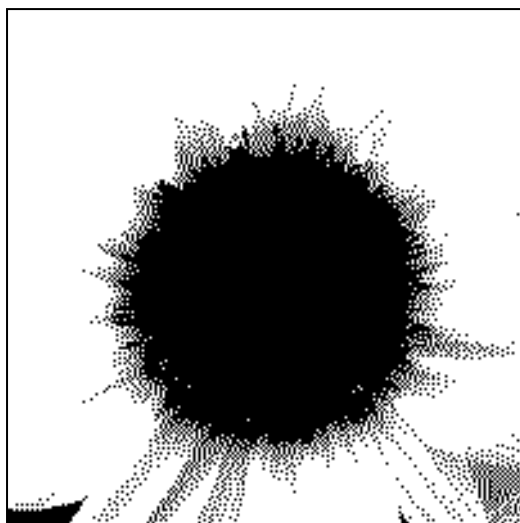
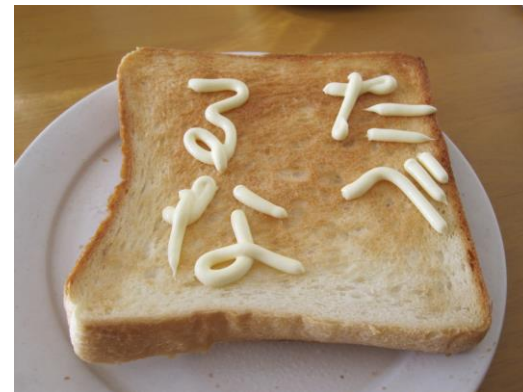
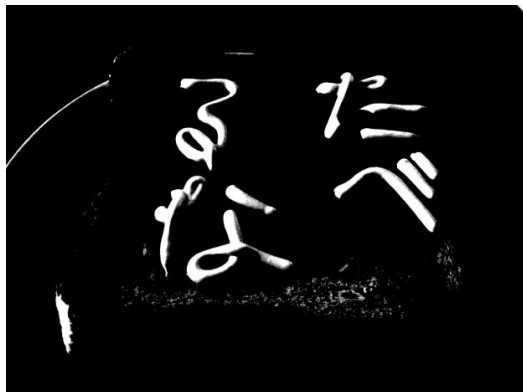


MRI画像@wikipedia



X線画像@wikipedia

2値画像, 濃淡画像, カラー画像



カラー画像



=



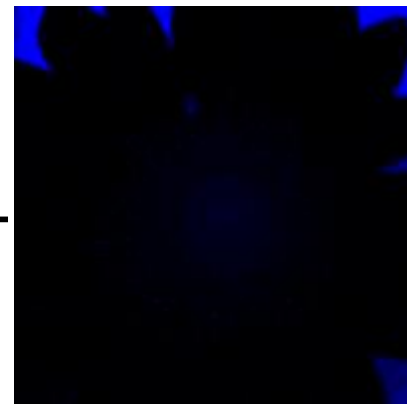
R成分

+



G成分

+



B成分

色って何？→付録

動画像 = 静止画像の時系列



フレーム
(静止画像)

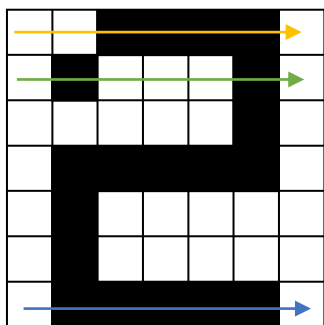


時間

画像データはどのように扱われる？ (1/3)

【再掲】画像は高次元ベクトル

● 2値画像の例

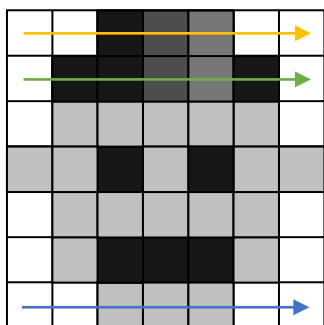


7×7画素

$$\Rightarrow (1, 1, 0, 0, 0, 0, 1, 1, 0, \dots, 0, 1)$$

49次元ベクトル

● グレースケール画像の例



7×7画素

$$\Rightarrow (255, 245, 10, 35, 92, 231, 254, \dots, 249)$$

49次元ベクトル

画像データはどのように扱われる？(2/3)

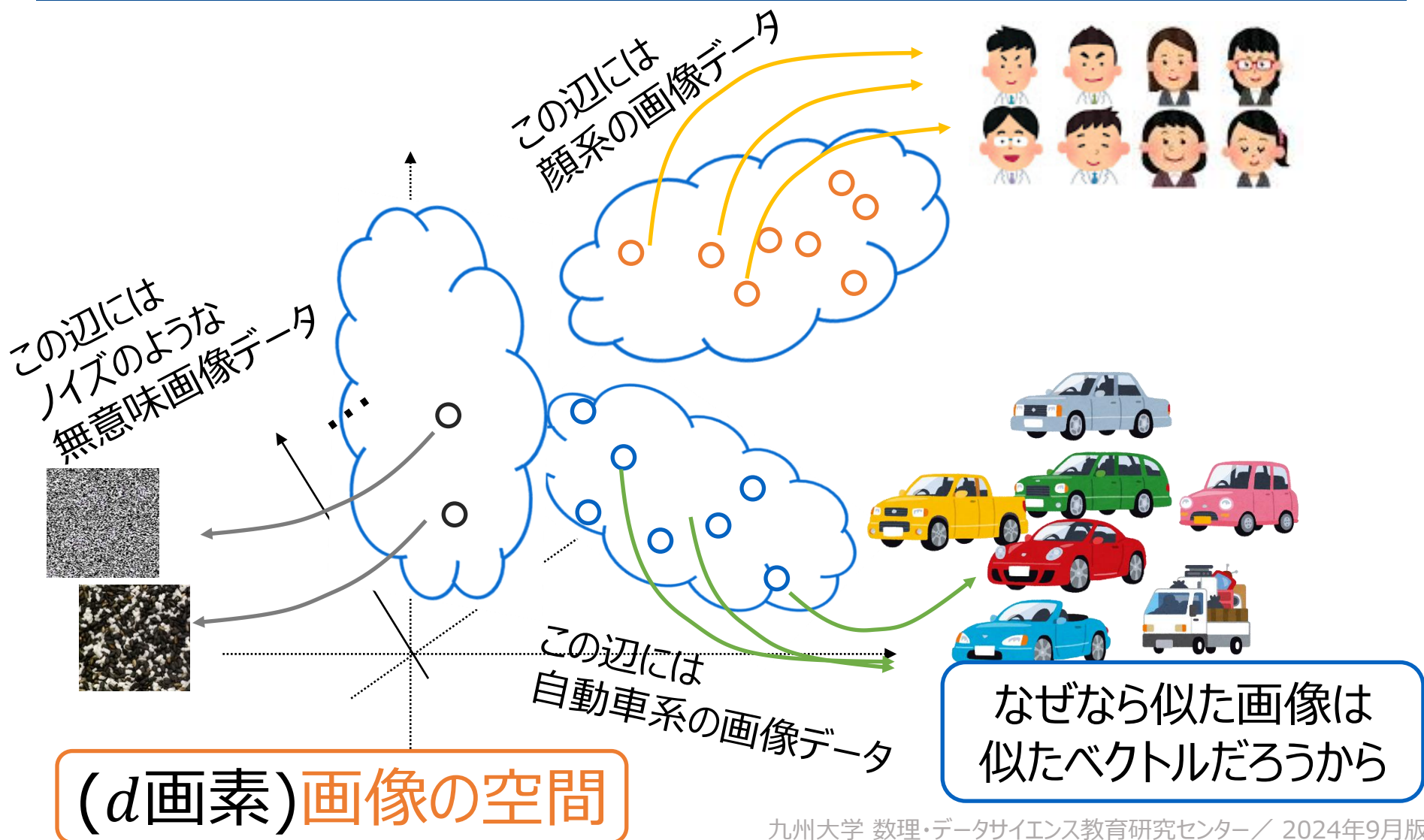
【再掲】画像は高次元ベクトル



- 皆さんのスマホ・デジカメ・コンピュータは、いつも超高次元ベクトルを扱っている
 - シャッター押した瞬間に1200万次元ベクトルが一つ生まれている

画像データはどのように扱われる？ (3/3)

【再掲】画像は高次元ベクトル



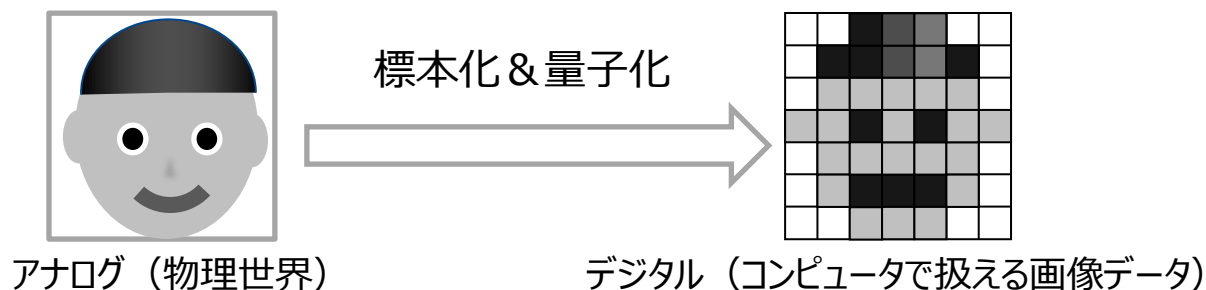
どうやって画像がベクトルに？

アナログ-デジタル変換

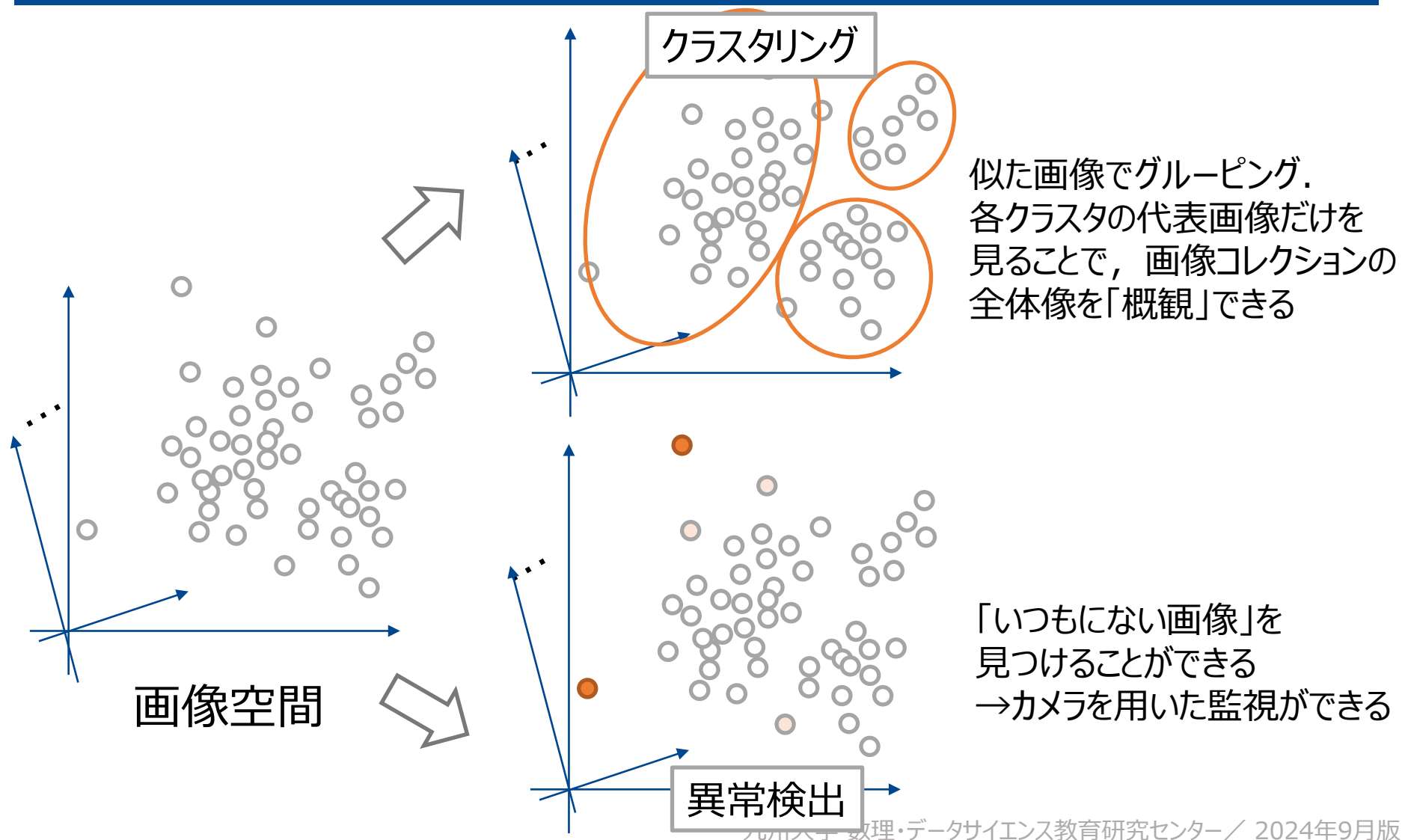
- そもそも目に見えている物理世界に「画素」はない
- カメラで写して初めて「画素」ができ、ベクトルになる



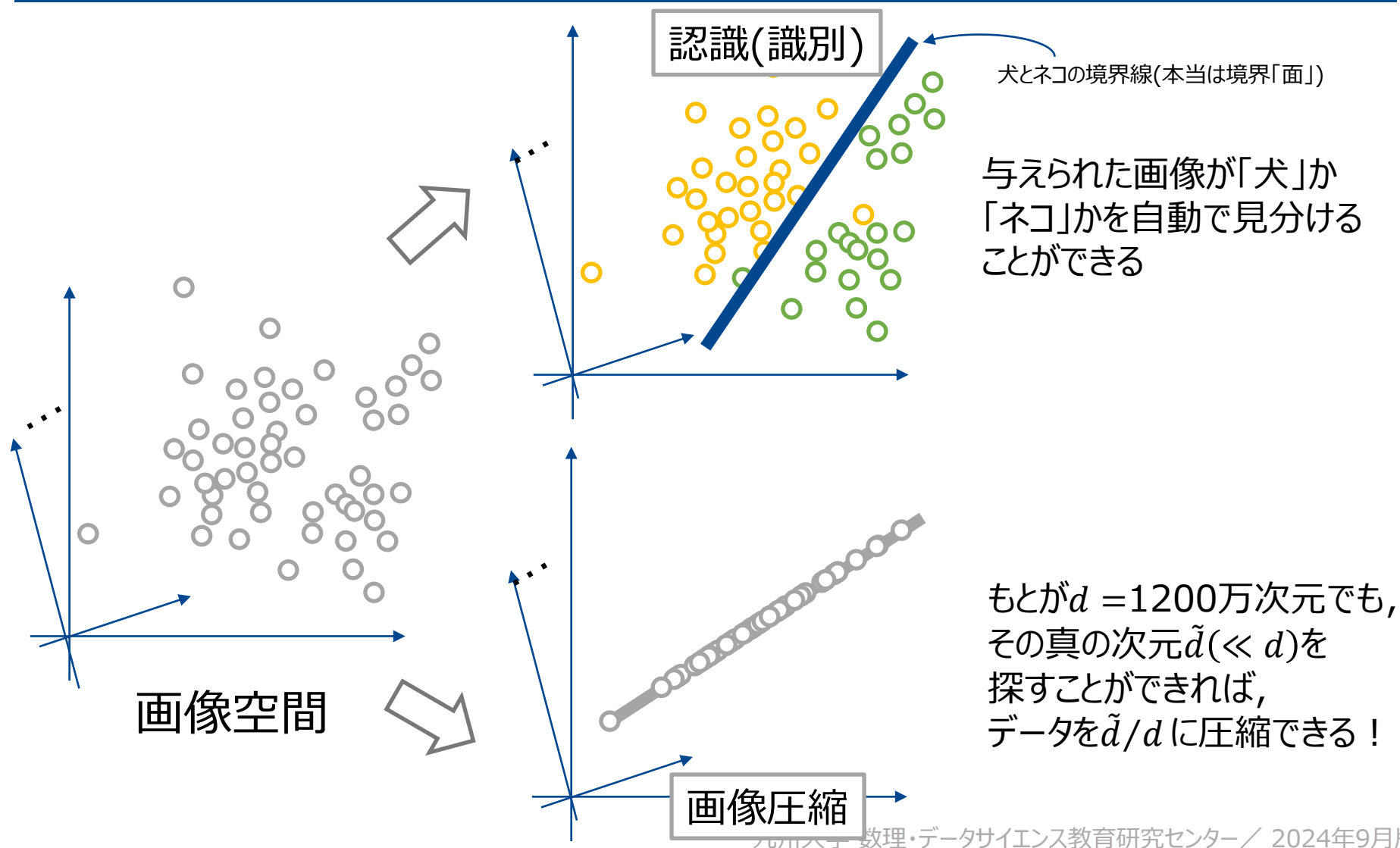
- カメラがやっているのが「アナログ-デジタル変換」(AD変換)
 - 標本化：一定間隔で画像の色・明るさを採集→【付録】
 - 量子化：各画素値をキリのよい数字(有限桁数)にする →【付録】



集めた画像は「ベクトルの集合」、 それをつかって、いろいろなことができます

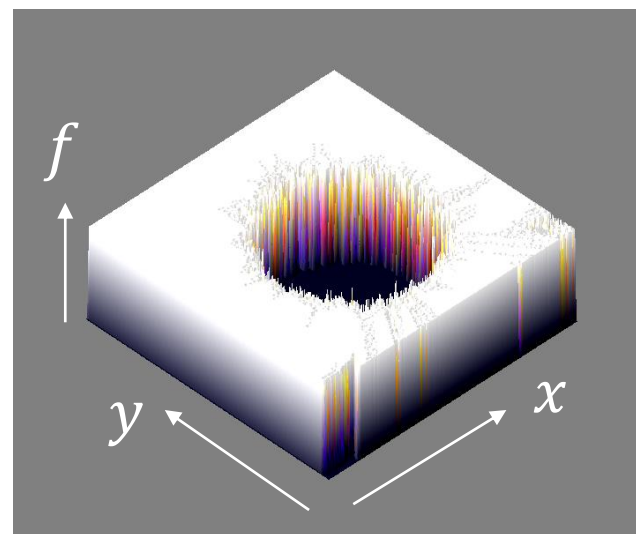
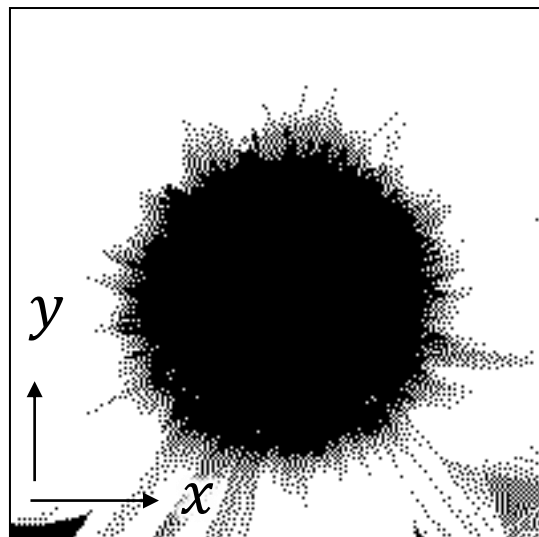


集めた画像は「ベクトルの集合」、 それをつかって、いろいろなことができます

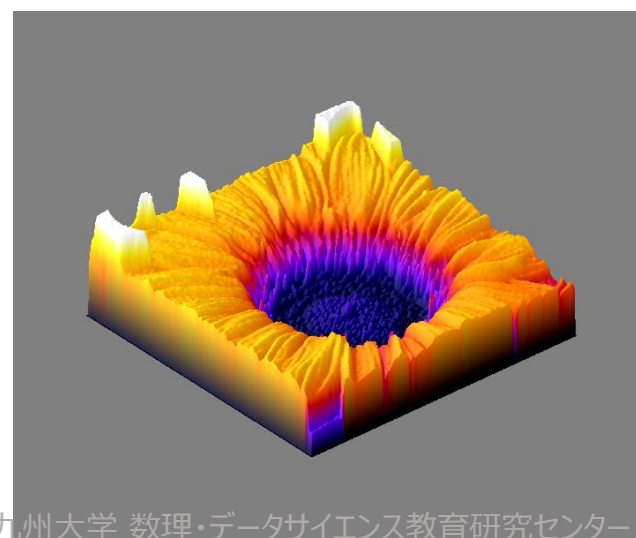


2次元信号 $f(x, y)$ として画像を見ることが出来る

二値画像



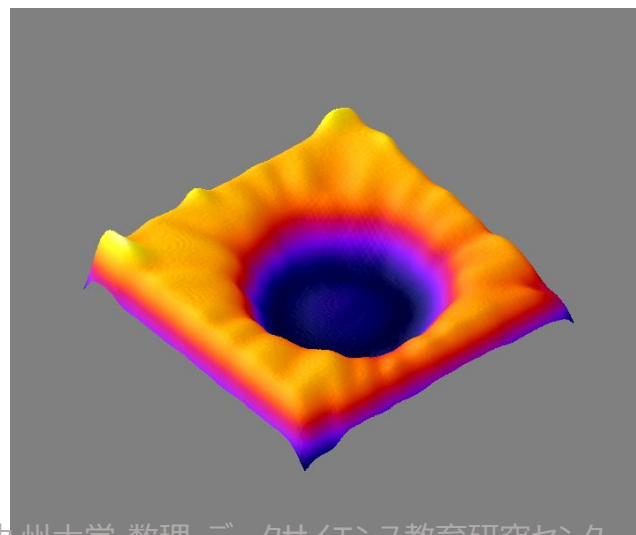
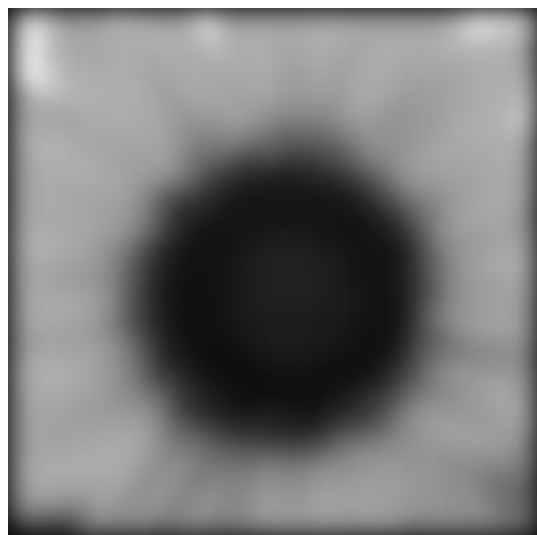
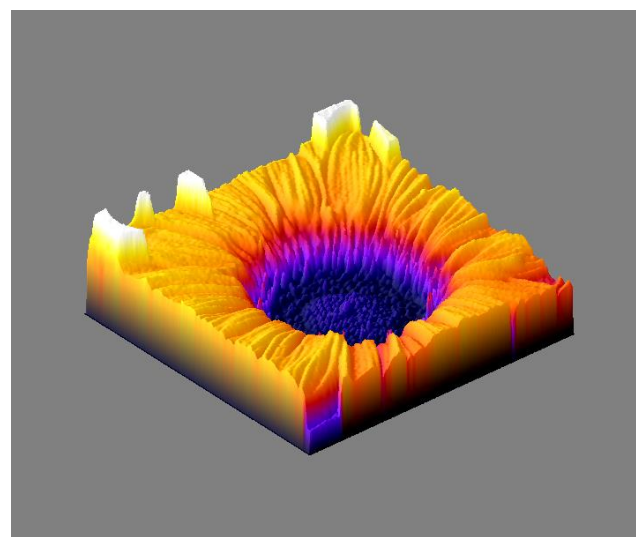
濃淡画像



画像データに関する様々な分析課題

- 画像処理
 - 画像の色や明るさを補正し、より見やすくする
 - 例：フィルタリング→【付録】
- 画像認識
 - そこに写っているものが何かを認識する（→「パターン認識と ニューラルネットワーク」）
- コンピュータビジョン
 - 複数枚の画像を組み合わせて3D立体視（ステレオ）
 - 画像中で動くものを追跡する（トラッキング）
- 画像生成
 - 新しい画像を作成する(Stable diffusionなど)
- 特徴抽出
 - 画像中に写っているものの個数をカウントする・面積を測る, など
 - 例：画像の二値化→【付録】
- 画像圧縮
 - 画像をよりコンパクトに表現する

画像処理の例： フィルタリング【付録】による画像の平滑化（ぼかし）



ぼかす

画像認識

ザクとした原理は「パターン認識とニューラルネットワーク」の項で

そこに写っているものは何かを当てる



正解→

GT: coucal

1: coucal

2: indigo bunting

3: lorikeet

4: walking stick

5: custard apple

認識結果
(上位5位)



GT: komondor

1: komondor

2: patio

3: llama

4: mobile home

5: Old English sheepdog



GT: yellow lady's slipper

1: yellow lady's slipper

2: slug

3: hen-of-the-woods

4: stinkhorn

5: coral fungus

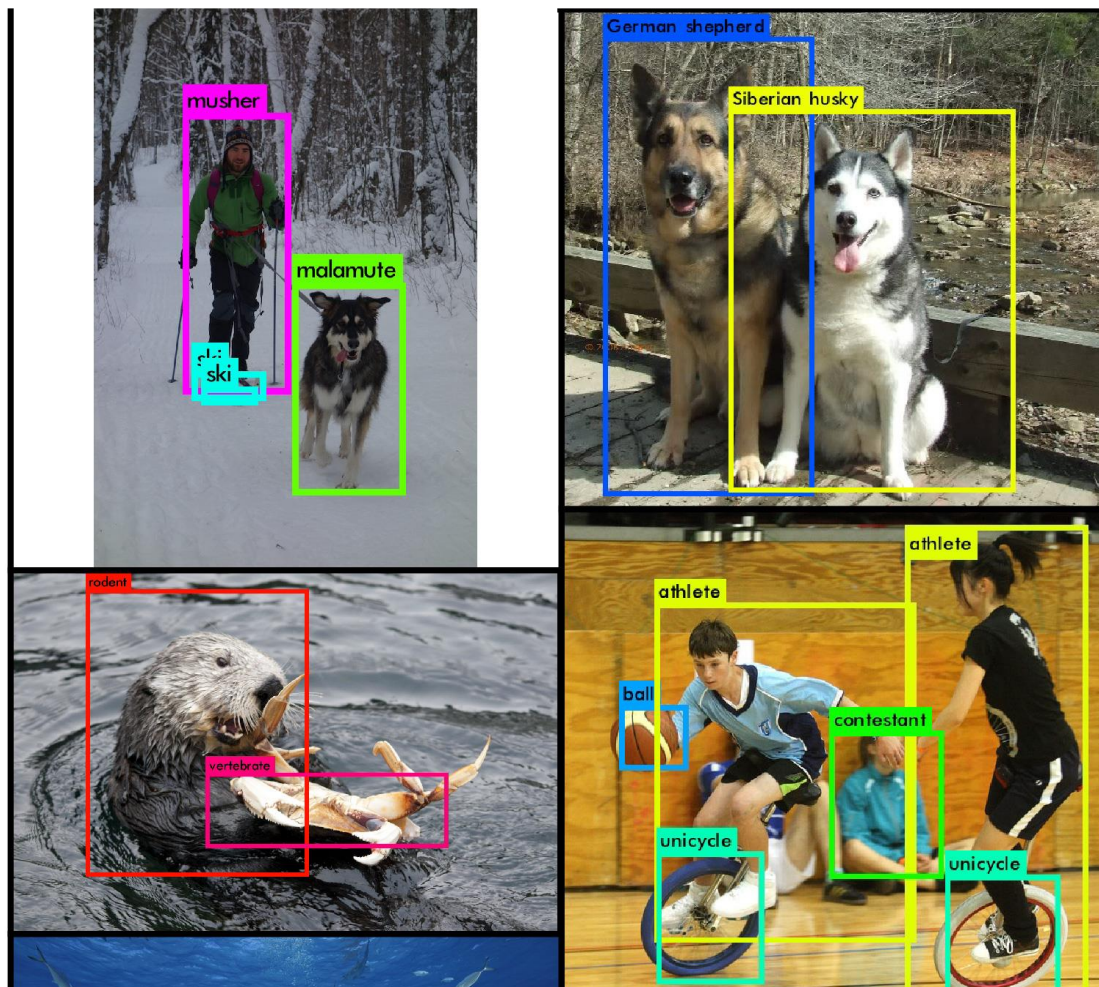
特定条件下ではあるものの、
人間の画像認識能力を超える
可能性があることが示された



[He+, ICCV2015]

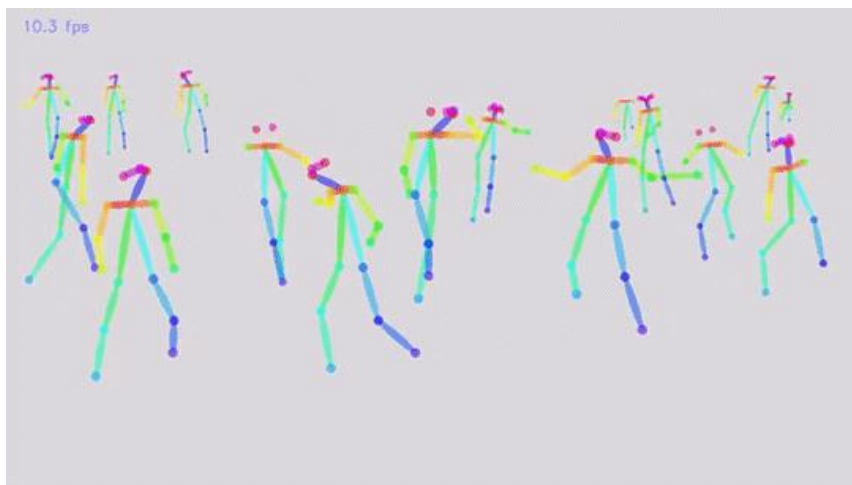
物体検出

9000種類の物体検出



[Redmon&Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR2017]

人体骨格抽出 OpenPose

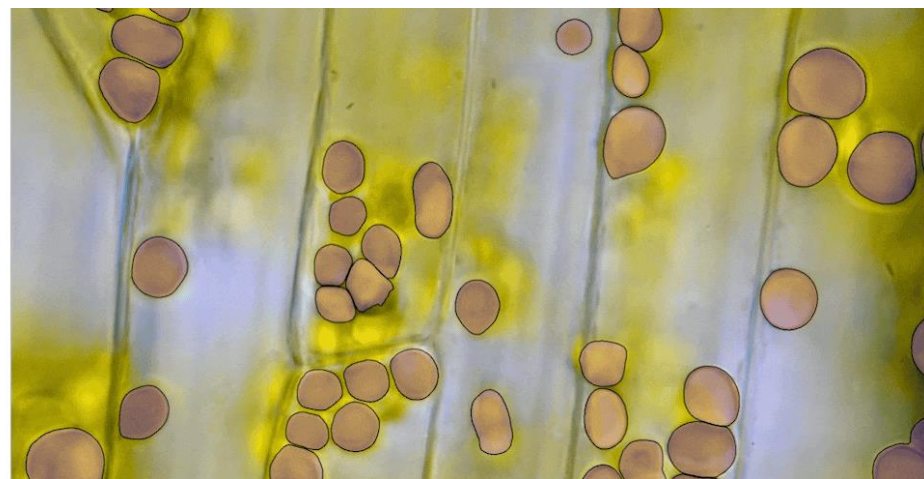
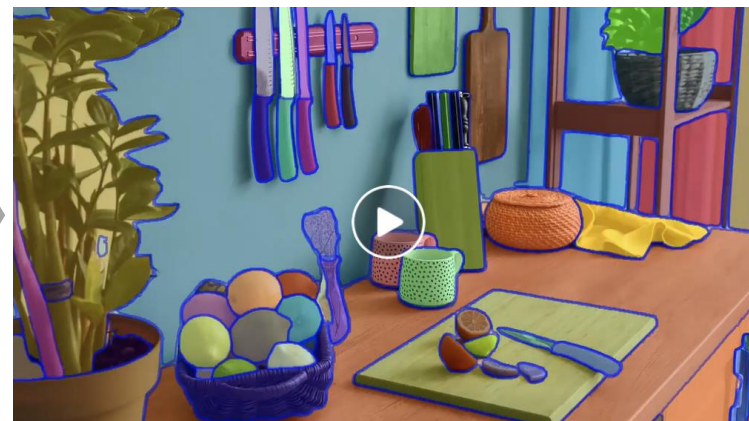
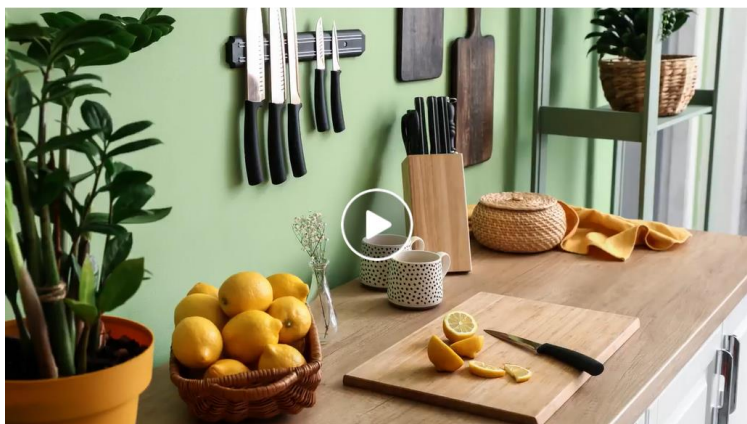


バイオ系の方は、DeepLabCutも要チェック

<https://github.com/CMU-Perceptual-Computing-Lab/openpose>

九州大学 数理・データサイエンス教育研究センター / 2024年9月版

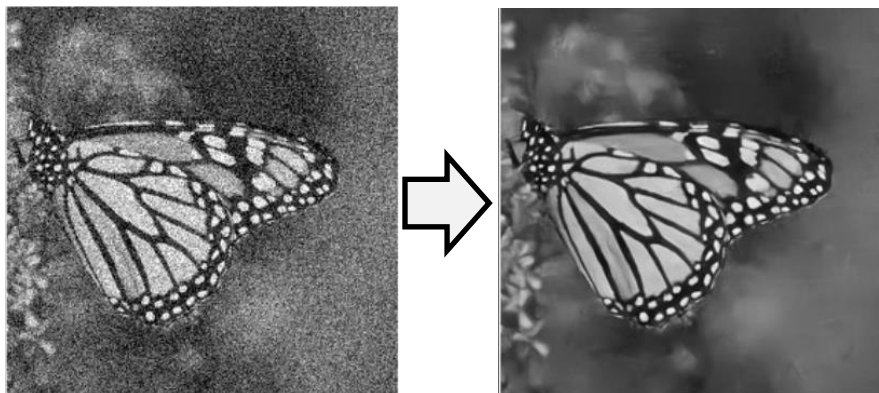
画像の領域分割 Segment Anything Model



<https://ai.facebook.com/blog/segment-anything-foundation-model-image-segmentation/>

様々な画像変換

- 高度なノイズ除去



[Mao+, NIPS2016]

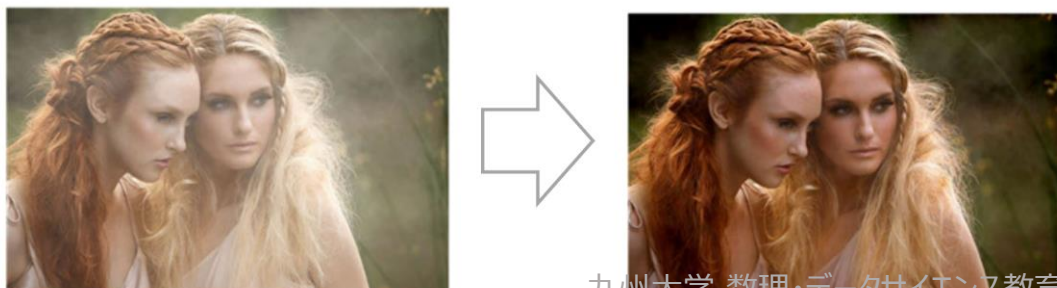
- 超解像

- 低解像度画像の高解像度化



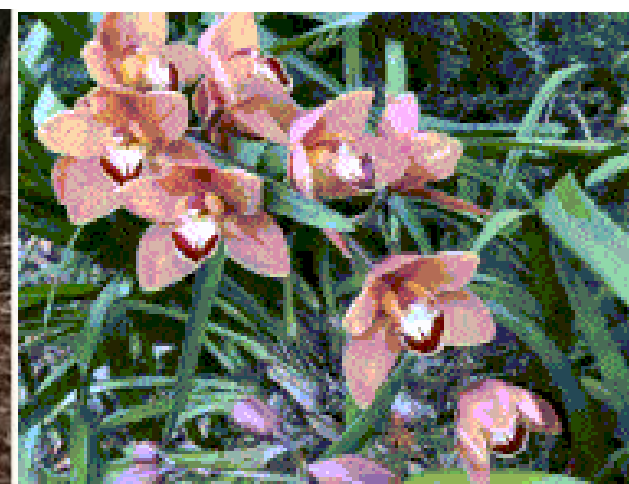
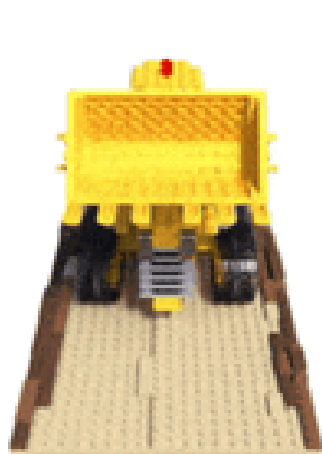
<https://towardsdatascience.com/image-super-resolution-an-overview-of-the-current-state-of-research-94294a77ed5a>

- 霧の除去



[Cai+, IEEE Trans. IP 2016]

複数静止画からの3次元再構成 NeRF



[Mildenhall+, ECCV2020]

生成AIについては「人工知能入門」の項でも、また説明します

画像生成 敵対的生成ネットワーク(GAN)

GANによる合成画像(要するに「存在しない人」)



学習に用いたデータの中で一番似ているもの

[Karras+, "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR2018]

画像生成 拡散モデル

拡散プロセス



拡散モデルによる生成画像の例



<https://stability.ai/news/stable-diffusion-sdxl-1-announcement>

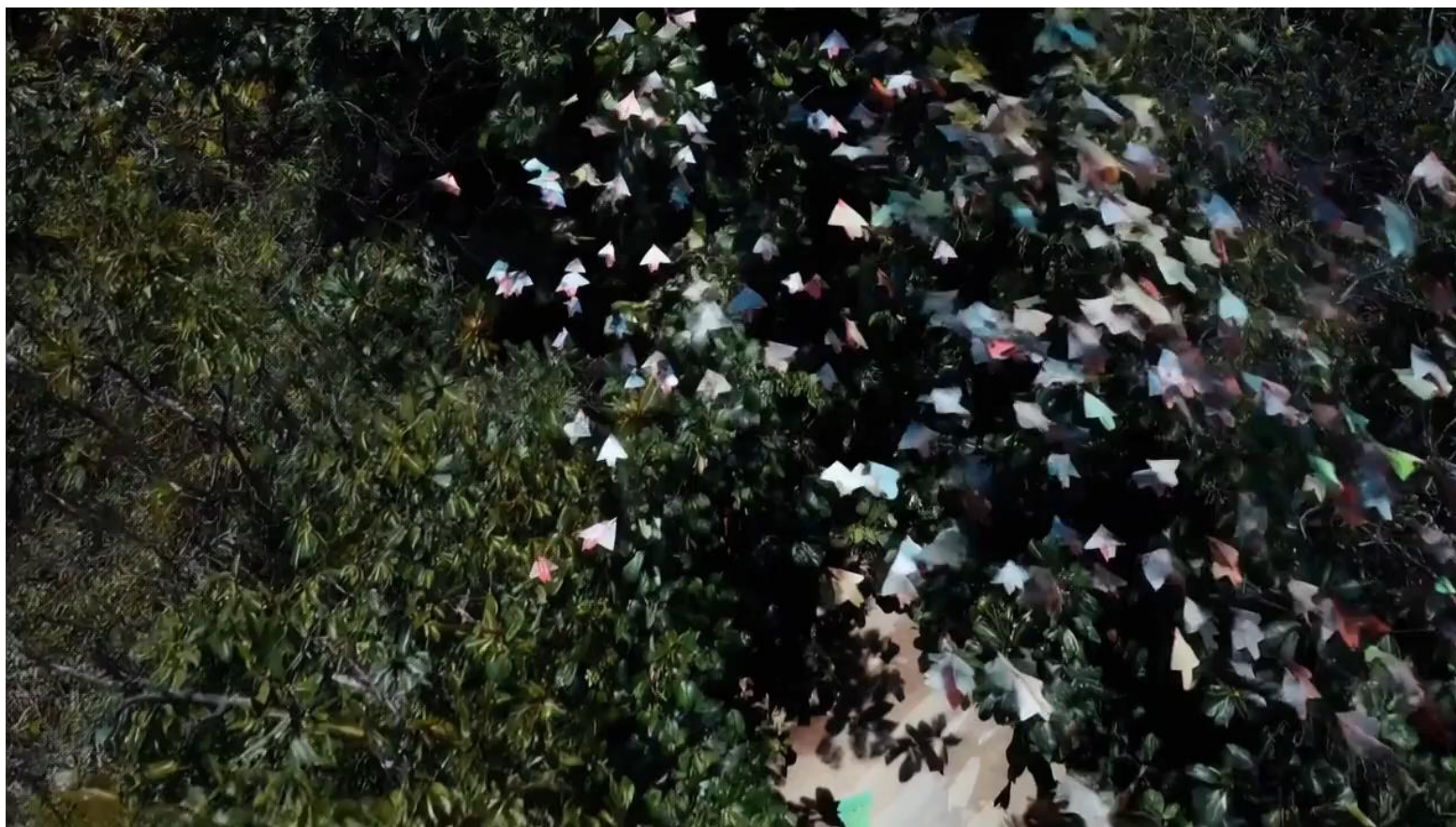


Kreis+, "Latent Diffusion Models," NeurIPS2023 tutorial

九州大学 数理・データサイエンス教育研究センター / 2024年9月版

プロンプトからの動画生成 Sora (OpenAI) 拡散モデルを利用

- Prompt: A flock of paper airplanes flutters through a dense jungle, weaving around trees as if they were migrating birds.



<https://openai.com/index/sora/>

音声／音楽データ

周波数との深い関係

音もデータ

- コンピュータにとっては「音」もデータ
 - 時々刻々と変わる音の波の高さを数値化すればデータに



- 三種類の「音」
 - 人間の声である音声



- 音楽



- 雑音や騒音, 生活音などの一般的な環境音



音声データの分析



- 音声認識

- 何と喋っているかを理解する（Siriやスマートスピーカ）



- 話者認識

- 誰がしゃべっているかを推定する：オレオレ詐欺のように人間でも時々難しい



- 感情認識

- 声の様子から感情を推定する
- 声質変換，音声強調（より聞きやすい音声にする）



- 音声合成

- 文-音声変換：例えば「おはよう」という文字列を読み上げる
- 音声翻訳：ある言語の音声を，別の言語に翻訳して発話
- 歌声合成：ボーカロイドでおなじみ



komtmt.files.wordpress.com/2012/12/img_miku_web.jpg

音楽データの分析



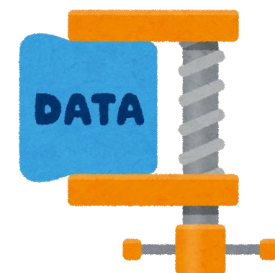
- 楽曲分析
 - 音楽のコード進行やメロディーライン, リズム等を推定
- 自動作曲および作曲支援
 - メロディーを入れるとコード進行を自動でつける
- 音楽合成
 - プロンプトで, 音楽データを自動生成
- 音響分析
 - 例えば, ボーカルだけを取り出したり, 消したりする
- 音楽認識
 - 街中で流れている音楽や鼻歌が, どの楽曲であるかを推定
- 楽曲推薦
 - 特定の楽曲と似た印象の楽曲を見つけて提示する



OpenAI Jukebox

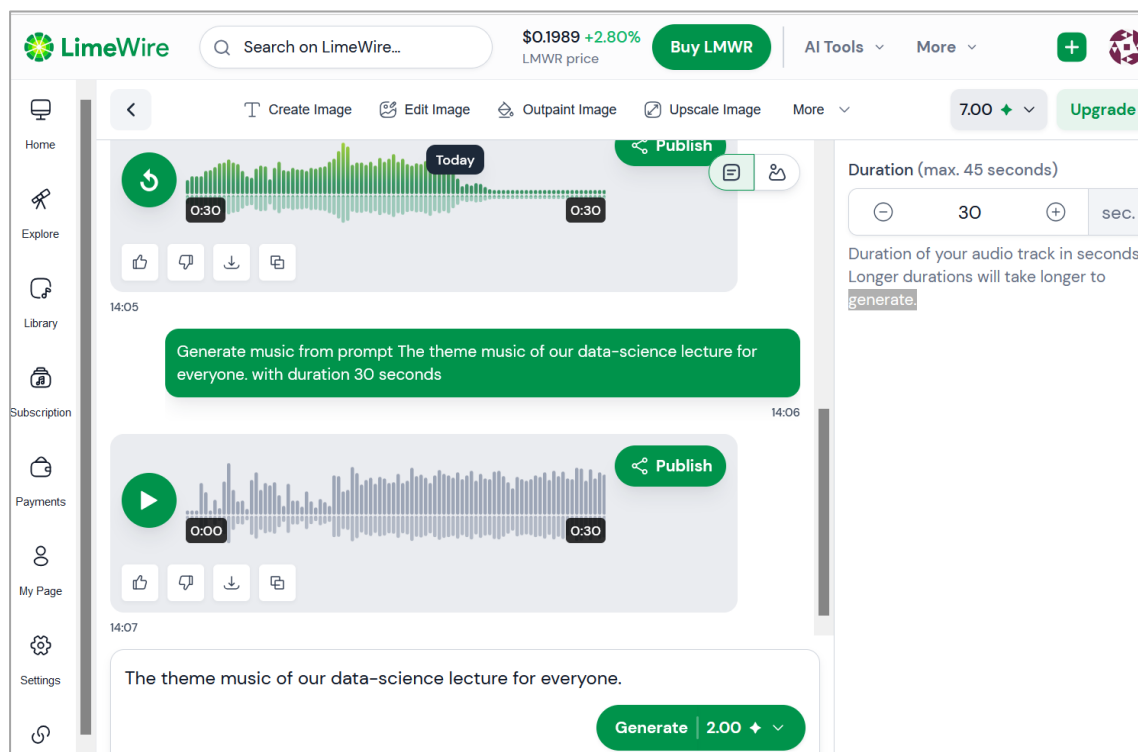


Meta MusicGen



MusicGenで作った音楽

- プロンプト : "The theme music of our data-science lecture for everyone." (2024年9月作成)



<https://musicgen.com/>

環境音データの分析

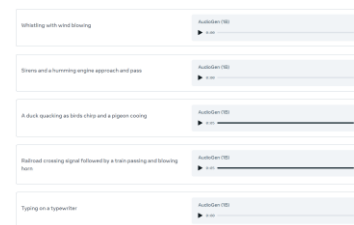
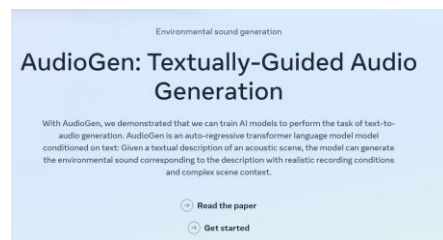
- 環境音認識

- 街中騒音や生活音が何の音かを推定する



- 音響合成

- 様々な音を合成する



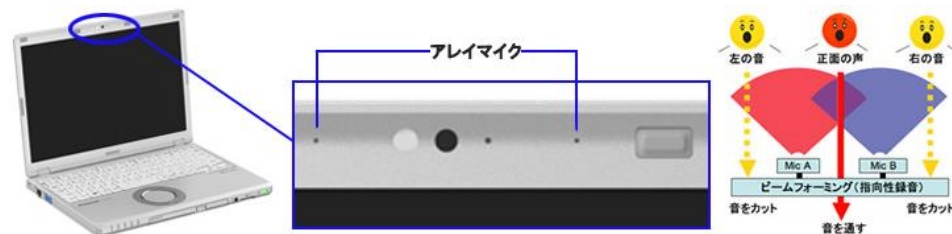
<https://audiocraft.metademolab.com/audiogen.html>

- 音源分離

- 複数の音が一度に鳴っているときに、音源別に音を分ける

- 音源同定

- どこから鳴っている音かを推定



<https://panasonic.biz/cns/pc/prod/note/sz5y/basic2.html>

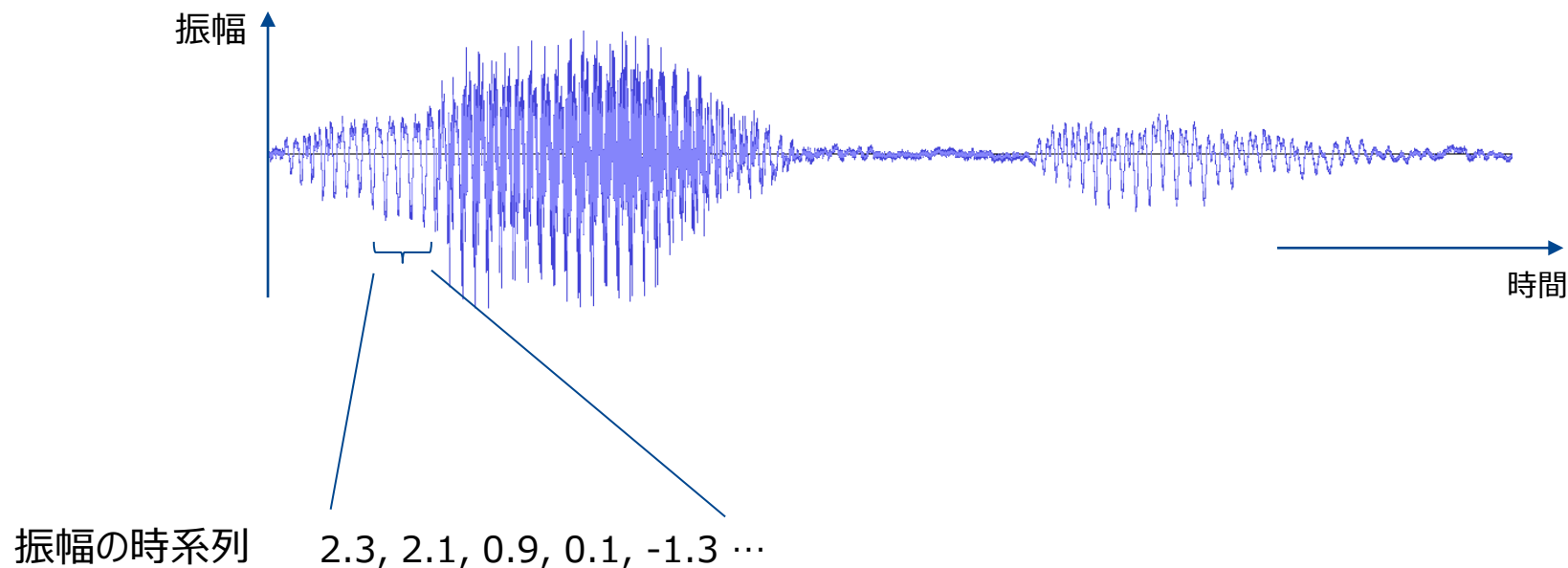
- 異常音検出

- いつもと違う音に気づく。例えばエンジン音の異常など

音のデータはどのように扱われる？ (1/2)

波形のまま = 振幅の時系列データ

- 音声波形の例

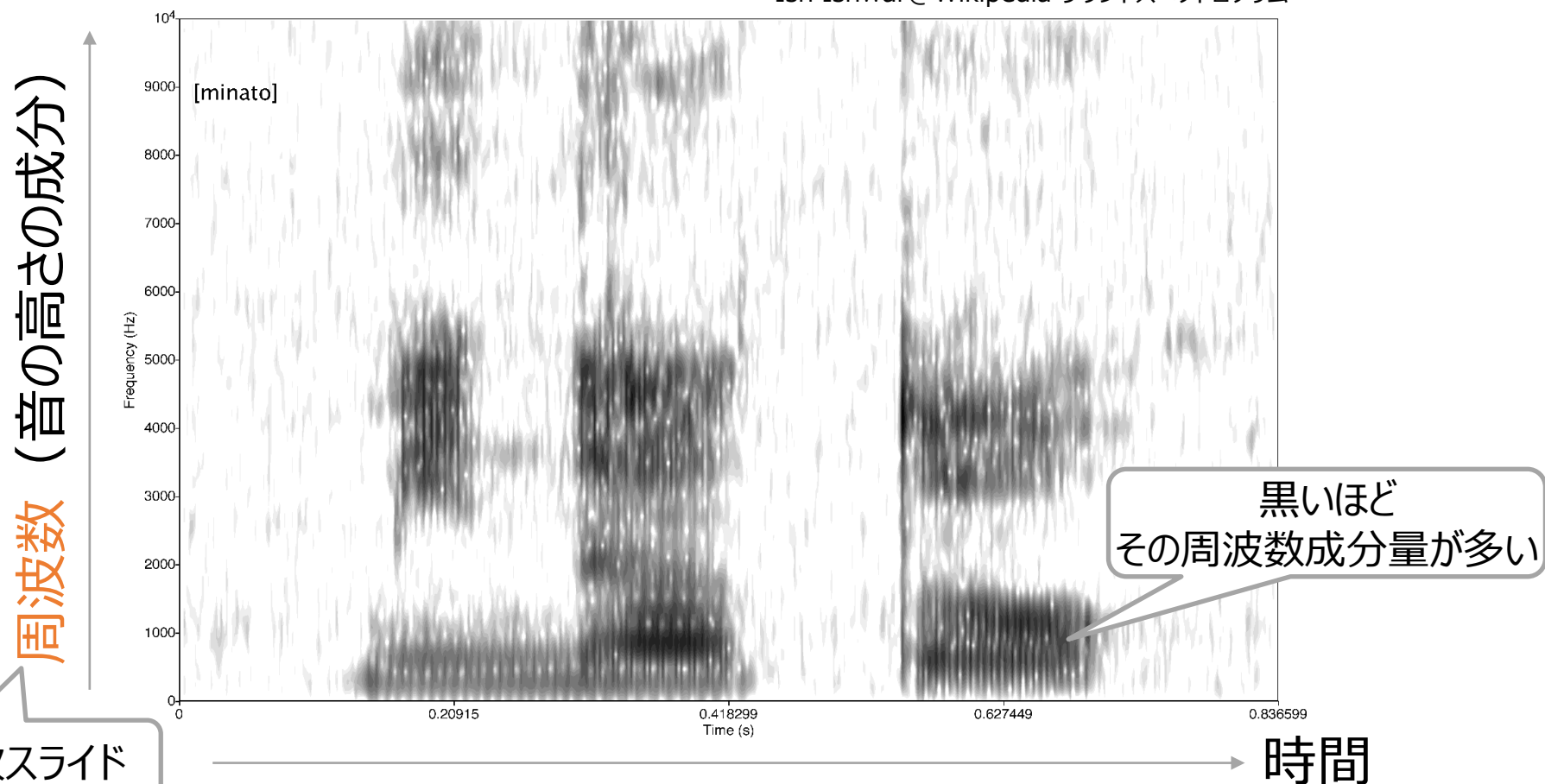


音のデータはどのように扱われる？ (2/2)

サウンドスペクトログラム

- 女性が「みなと」と発声

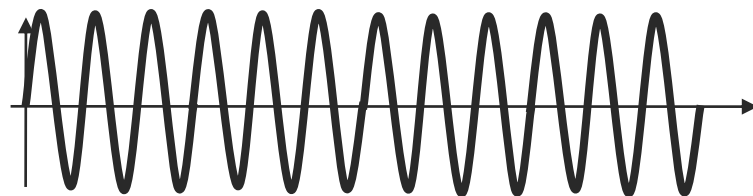
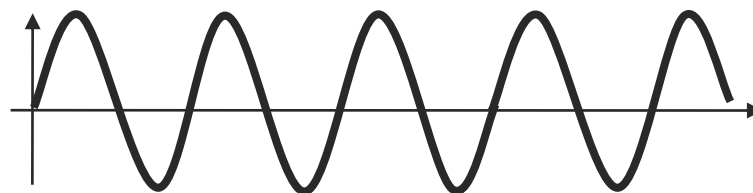
Ish Ishwar@Wikipedia サウンドスペクトログラム



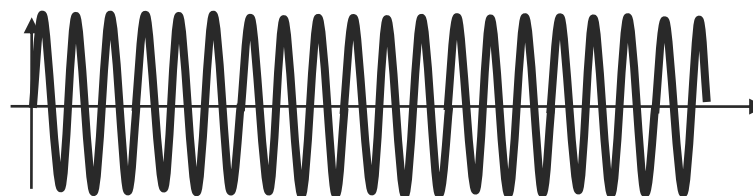
周波数とは？

繰り返し周期の逆数

周波数低い
(低音)

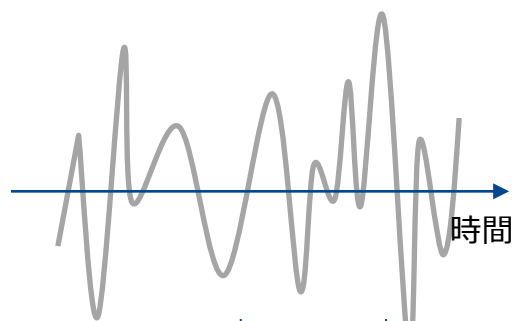


周波数高い
(高音)

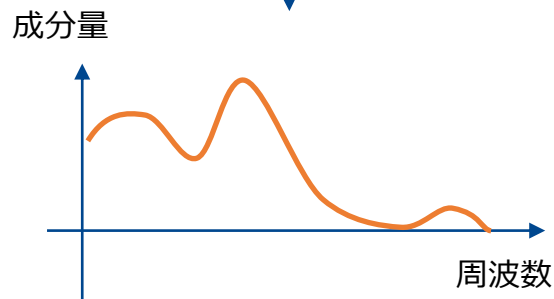


サウンドスペクトログラムはどうやって求める？

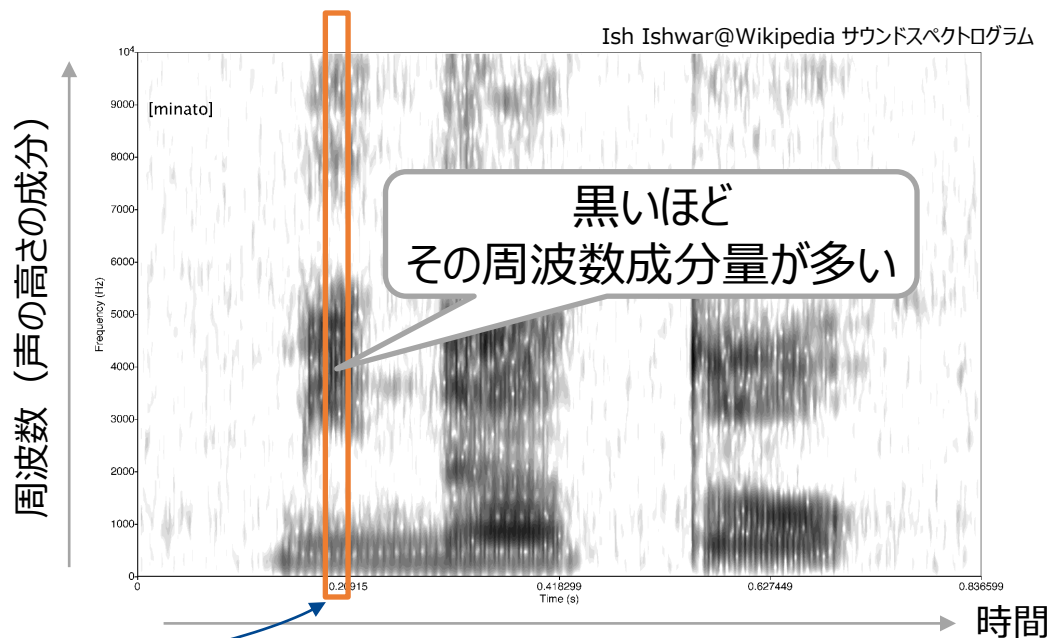
→ 各時刻で「周波数スペクトル」を求める



周波数分析
(フーリエ変換(後述&付録))

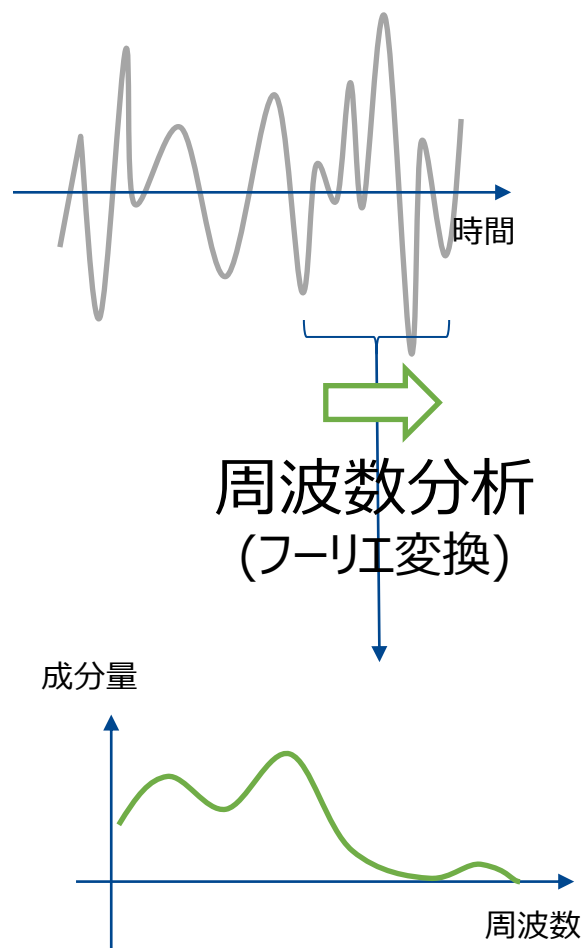


周波数スペクトル
(各周波数成分がどれぐらい入ってるか)



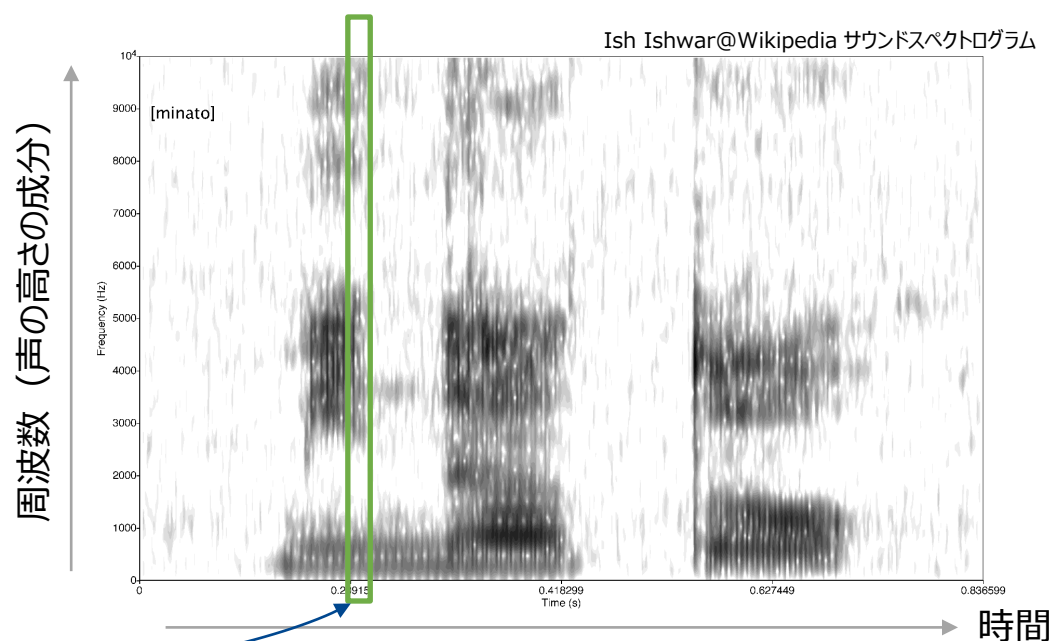
サウンドスペクトログラムはどうやって求める？

→ 各時刻で「周波数スペクトル」を求める



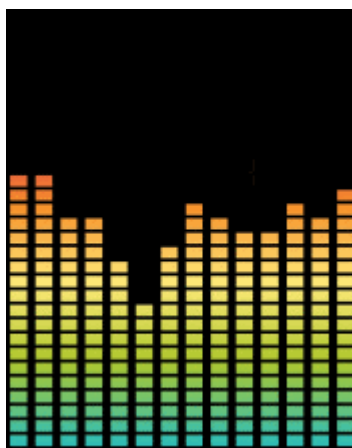
周波数スペクトル
(各周波数成分がどれぐらい入ってるか)

これを時刻ごとに作っていけば完成



こんなところにも周波数スペクトル

- 音楽に合わせて動く, こういう↓ものを見たことないですか？
 - これも周波数スペクトルの一種



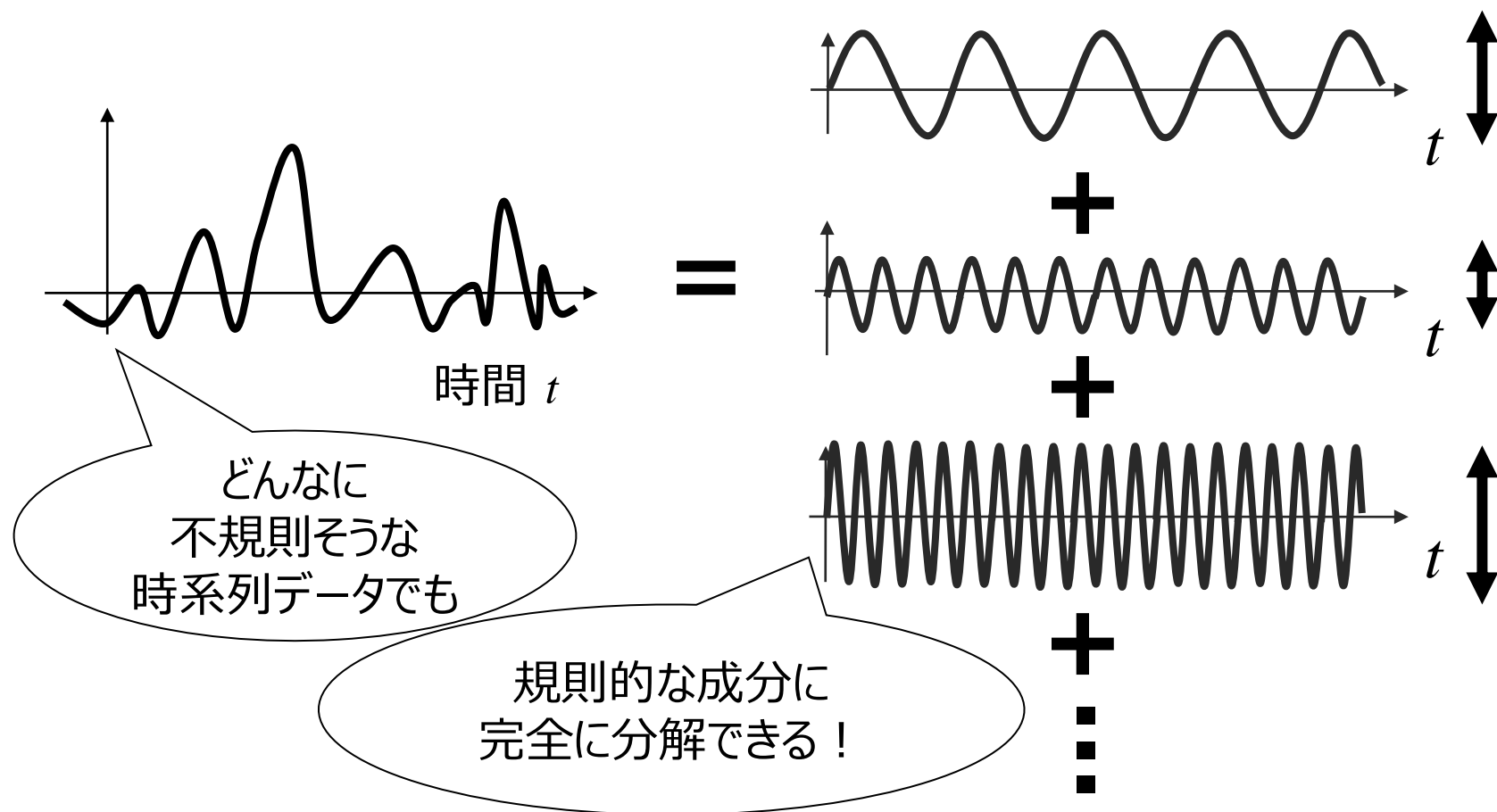
<https://phoneky.com/gif-animations/?id=s2s106913>

低い周波数
(低音)

高い周波数
(高音)

- その波形を作る「レシピ」なのです
 - 「レシピ？ 確かベクトルの分析のところでも聞いたような」と思い出してくれると幸い

驚きの事実：時系列データは、 周波数の異なるsin波・cos波の和に分解可能！



思い出そう...

材料セット

$e_1, \dots, e_i, \dots, e_d$



$$\alpha_i = x \cdot e_i$$



カレー x

$$\sum \alpha_i e_i$$

レシピ(分析結果)



$\rightarrow \alpha_1$ グラム

\vdots



$\rightarrow \alpha_i$ グラム

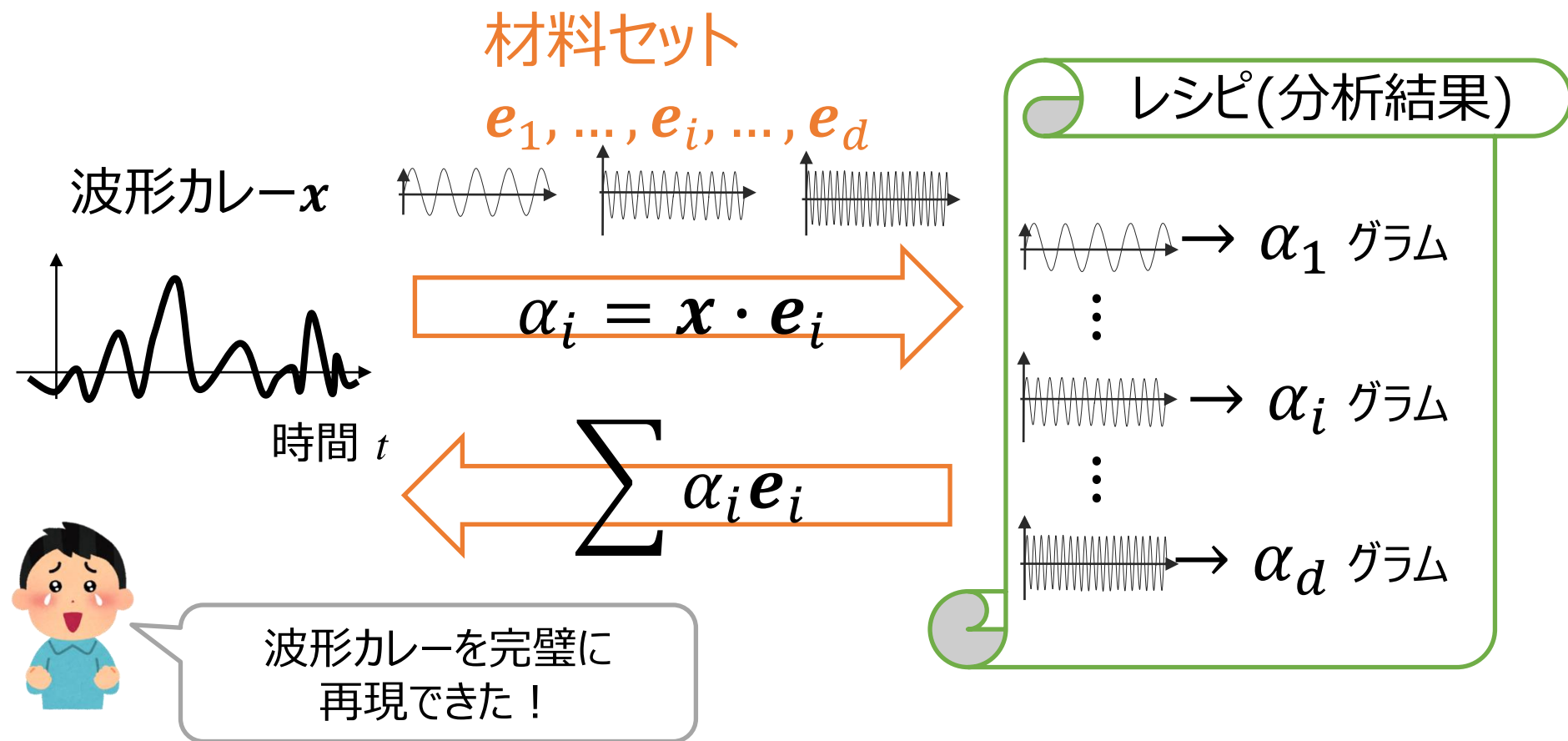
\vdots



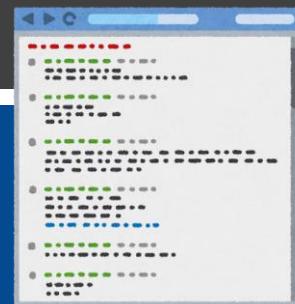
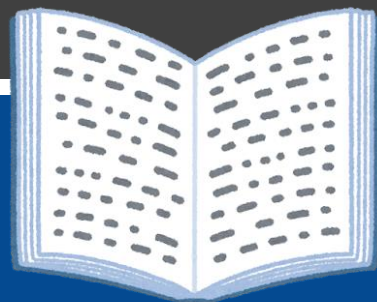
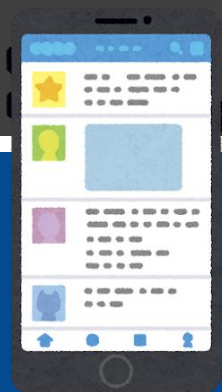
$\rightarrow \alpha_d$ グラム

あの味を完璧に再現できた！

実は「ほぼ」同じ話です：
これはフーリエ変換^{【付録】}と呼ばれ、工学で頻繁に登場



自然言語データ



言語データ

- 言語データ = 文字列で表されるデータ
 - 「あ」や「A」などの文字を並べると単語ができる
 - 単語を並べると「勉強，大変ですね」というような文ができる
- SNS やネットでは，言語データが日々大量に生まれている
- 自然言語処理(natural language processing)
 - 言語データをコンピュータによって分析する技術

自然言語処理の例： 頻出単語と翻訳

● 頻出単語

- 言語データの中で最も多く出てきた単語を見つける処理
- 例えば
 - 本日「twitterで最もつぶやかれた単語」 → その日のトレンド
 - ある本の中で比較的多く出てくる単語 → その本の内容の概要
 - 普通はあまり使われてない言葉だけど、その文書では何回か出てくる → キーワード

● 翻訳

- 日英翻訳でお世話になってる人も多いのでは？
- 昔は「文法解析」に基づく翻訳
- いまは対訳データをAIに学習させる場合が主流
 - “I love you” ⇔ “私はあなたが好きです” というようなペアを大量に準備
 - AIに、それらペアに潜む傾向を理解させる



自然言語処理, その他の例

- 検索

- ある文に似た意味の文をネットなどから探す

- 要約

- 長い文章を, 内容を損なわない程度に短くする

- 対話

- チャットボットのように会話したり, 質問に回答する
- その延長線上に「文章自動生成」も

- 校正

- よりよい文章になるような書き換えを提案する
- 文法チェック

- トピック分析

- その文章が, 例えばスポーツ関係なのか, 政治関係なのかを分析する



自然言語処理，さらに進んだ例：

● 意味解析

- 各単語や文を，その意味に応じて数値ベクトル化
- king - man + woman = queen のような，言葉の計算が可能に

● パラフレーズ解析

- 「私はお腹がすいた」と「何か食べるものが欲しい」は同じ意味か否か
- 言葉としては違うが意味は同じという，難しい問題

● センチメント解析

- 文章に潜む感情を推測する課題
- 「こんなに汚れが取れる洗剤，見たことがない」という文章は肯定的か否定的？
- 「ない」という否定語で終わってるが，内容は肯定的

● 文章生成

- 小説のような文章や，ジョーク（ダジャレ），俳句を自動的に作らせる
- 最近では…

難しいけど
面白い技術！



文章生成AI : ChatGPT@OpenAI, Bing@Microsoft, Gemini@Google (1/2)

SE

あなたは情報系の研究者です。データサイエンスが苦手な人に講義をする際、どの辺に気を付ければよいでしょうか？

2023.9.1実施



データサイエンスを苦手な人に講義する際、以下の点に気を付けることが大切です。これらのアプローチを取ることで、学習者が理解しやすく、興味を持ちやすい環境を提供できます。

1. 丁寧な基礎教育:

データサイエンスは幅広いトピックから成り立っていますが、最初に基本的な概念をしっかりと教えることが重要です。統計学、プログラミング、データの前処理などの基礎を丁寧に説明し、学習者が基本的なスキルを身につけられるようサポートしましょう。

2. 目的を明確に伝える:

学習者に対して、データサイエンスを学ぶ目的やその価値を明確に伝えることが重要です。実世界の問題を解決する手段としてデータサイエンスがどれだけ役立つかを示し、モチベーションを高めましょう。

3. 実践的なプロジェクトを組み込む:

理論だけでなく、実践的なプロジェクトを積極的に取り入れましょう。実際のデータを扱い、問題を解決する過程を通じて、学習者がスキルを実践で活用できるようになります。

4. インタラクティブな学習を促進:

講義をインタラクティブにし、学習者が質問をしやすい環境を提供しましょう。ディスカッションやデモンストラ

文章生成AI : ChatGPT@OpenAI, Bing@Microsoft, Gemini@Google (2/2)

2023.9.1実施

判定もできる

SE

二値化の画像処理のソースコードをopenCVとpythonを使って記述してみてください

プログラミングもできる



OpenCVとPythonを使用して画像を二値化するソースコードを以下に示します。このコードでは、OpenCVの`cv2.threshold()`関数を使用して、指定の閾値を超えるピクセルを白、それ以下のピクセルを黒に変換します。

```
python
import cv2

# 画像を読み込む
image = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE) # 画像ファイル名を適切に

# 二値化の閾値を設定
threshold_value = 128 # 任意の閾値を設定（例：128）

# 画像を二値化する
ret, binary_image = cv2.threshold(image, threshold_value, 255, cv2.THRESH_BINARY)

# 二値化された画像を表示
cv2.imshow('Binary Image', binary_image)

# キー入力待ち（任意のキーを押すとウィンドウを閉じる）
cv2.waitKey(0)
cv2.destroyAllWindows()

# 二値化された画像を保存（任意のファイル名を指定）
cv2.imwrite('binary_output.jpg', binary_image)
```

このコードでは、`input.jpg`という名前の画像ファイルを読み込み、指定の閾値（例では128）を使用して二値化します。結果の二値化画像はウィンドウに表示され、`binary_output.jpg`という名前で保存されます。閾値を調整することで、二値化の効果を調整できます。必要に応じて画像ファイル名を適切に変更してください。

SE

User

あなたは救急救命士です。次の状況の危険度を、10段階で答えてください。
「包丁でちょっと指先を切った」
「プールにおぼれて意識不明になっている」



次の状況の危険度を10段階で評価します：

- 「包丁でちょっと指先を切った」
 - 危険度: 2/10
 - 指先を包丁で軽く切った場合、通常は出血が少量であり、一般的には緊急性が低いです。応急処置で出血を止め、傷口を清潔に保つことが大切ですが、深刻な問題ではありません。
- 「プールにおぼれて意識不明になっている」
 - 危険度: 10/10
 - プールでおぼれて意識不明になる状況は非常に危険です。水中での意識喪失は窒息や酸素不足による脳損傷のリスクが高まります。直ちに救助を行い、救急医療の支援を呼ぶ必要があります。時間の経過が重要で、速やかな対応が命を救うことに繋がります。

SE

ウサギと亀の恋愛短編小説をギャル語で作ってみてください。

物語生成もできる



もちろん、ギャル語でウサギと亀の恋愛短編小説を作成します。こちらですね！

ウサギちゃんは、ガーリーガールなの。ピンクのリボンがトレードマークで、ぶるぶるのモテモテボディなのよ。いつもファッションに気を使って、キュートなコーディネートが大好きで、友達とカフェでチャイラテを楽しむのが日課。

ある日、お散歩中に会ったのが、亀くん。亀くんはクールで、ゆったりとした性格のカリスマ男子。サングラスをかけて、スローペースで歩く姿が格好良くて、ウサギちゃんも一目惚れしちゃった。でも、ウサギちゃんのスピードについてはいけないうさぎは、最初は距離を感じていたんだよ。

つまんないの後略

3版

「生成型AIによる生成物」と「人間が作ったもの」は もやは見破りAIでは見抜けない

ChatGPT生成文を正しく「生成物」と見抜けた割合
(= #「生成物と判定」/ #生成物)

人間が作った文章を正しく
「人間による」と見抜けた割合

TABLE I: Summary of analyzed papers

Approach	Published in	Target Model				Publicly Available	Free/Paid	ChatGPT detc. Capability (TPR%)	Human-text detc. Capability (TNR%)
		Grover	GPT-2	GPT-3	ChatGPT*				
Kumarage et al. [21]	2023		✓			✓	Free	23.3	94.7
Bleumink et al. [6]	2023			✓	✓	✓	Paid	13.4	95.4
ZeroGPT [40]	2023				✓	✓	Paid	45.7	92.2
OpenAI Classifier [28]	2023				✓	✓	Free	31.9	91.8
Mitchell et al. [25]	2023		✓			✓	Free	18.1	80.0
GPTZero [29]	2023		✓	✓	✓	✓	Paid	27.3	93.5
Hugging Face [13]	2023				✓	✓	Free	10.7	62.9
Guo et al. [18]	2023				✓	✓	Free	47.3	98.0
Perplexity (PPL) [17]	2023				✓	✓	Free	44.4	98.3
Writefull GPT [36]	2023			✓	✓	✓	Paid	21.6	99.3
Copyleaks [10]	2023			✓	✓	✓	Paid	22.9	92.1
Cotton et al. [8]	2023			✓	✓	×	-	-	-
Khalil et al. [20]	2023				✓	×	-	-	-
Mitrovic et al. [26]	2023		✓		✓	×	-	-	-
Content at Scale [3]	2022		✓	✓	✓	✓	Paid	38.4	79.8
Originality.ai [1]	2022			✓	✓	×	Paid	7.6	95.0
Writer AI Detector [37]	2022			✓	✓	✓	Paid	6.9	94.5
Draft and Goal [12]	2022			✓	✓	✓	Free	23.7	91.1
Gao et al. [15]	2022				✓	×	-	-	-
Fröhling et al. [14]	2021	✓	✓	✓		✓	Free	27.8	89.2
Kushnareva et al. [22]	2021	✓	✓			✓	Free	25.1	96.3
Solaiman et al. [33]	2019		✓			✓	Free	7.2	96.4
Gehrmann et al. [16]	2019		✓			✓	Free	32.0	98.4
Zellers et al. [39]	2019	✓				✓	Free	43.1	91.3

*GPT 3.5 and above.

生成AI≠検索. なので、次のような質問をすると…

SE

九大伊都キャンパス回りの美味しいラーメン屋さんを教えてください。

2023.9.1実施



もちろん、九大伊都キャンパス周辺の美味しいラーメン屋さんをいくつか紹介します。ただし、私の知識は2021年9月までの情報に基づいているため、最新の情報や新しい店舗については確認が必要です。以下は、九大伊都キャンパス周辺で評判の良いラーメン屋さんの一部です：



1. まる吉ラーメン

- 住所: 福岡市東区香椎駅前1-10-17
- 人気のあるラーメンチェーンで、濃厚な豚骨スープが自慢です。

2. 麺屋 いっとく

- 住所: 福岡市東区香椎駅前1-10-21
- こちらも香椎駅近くにあり、家系ラーメンが評判です。

3. らぁ麺 鶴見

- 住所: 福岡市東区鶴見2-4-15
- 鶴見駅近くに位置し、独自のスープと麺が楽しめます。

4. らーめん 鶴見屋

- 住所: 福岡市東区鶴見2-2-26
- 鶴見駅周辺にある、ラーメン愛好者に愛されるお店です。

5. 博多一風堂

- 住所: 福岡市東区香椎芥屋団地1-1
- 博多ラーメンの有名なチェーンで、濃厚なスープが特徴です。

•なぜか福岡市東区の店

- 伊都キャンパスは東区と数十キロ離れている

•住所滅茶苦茶

- 1.2.の番地にあるのはマンション
- 3.~5.鶴見や香椎芥屋団地という町はない

•5.以外は福岡市には存在しない

これらのラーメン店は、九大伊都キャンパス周辺にある美味しいラーメンを楽しむための選択肢の一部です。訪れる前に、営業時間や定休日などの最新情報を確認することをお勧めします。また、地元の人々からの口コミや評判も参考にして、自分の好みに合ったお店を見つけるのも良いアイデアです。

ところが、最近では検索結果も利用可能な生成AIも

2024.9実施

九大伊都キャンパス周りの美味しいラーメン屋さんを教えてください。



4 件のサイトを検索しました ▾

九大伊都キャンパス周辺には、いくつか美味しいラーメン屋があります。以下のおすすめ店をご紹介します。

1. まる勝

九州大学伊都キャンパス近くの人気店で、純粋な豚骨ラーメンが楽しめます。キクラゲラーメンなど、トッピングも豊富で、学生からも評判が高いお店です(ウォーカープラス (Walkerplus) (Ramen Database))。 ← 根拠リンクの提示

2. ラーメン大吉

こちらも同じく豚骨スープが自慢のラーメン店で、濃厚でクリーミーなスープが特徴です。キャンパスからのアクセスも良く、九大生に人気があります(ウォーカープラス (Walkerplus))。 ← 根拠リンクの提示

3. ラーメン く〜た 伊都店

「く〜た」は、九大学研都市駅近くに位置し、定番の豚骨ラーメンが楽しめるお店です。クリーミーなスープと、しっかりした麺の組み合わせが人気です(Ramen Database)。 ← 根拠リンクの提示

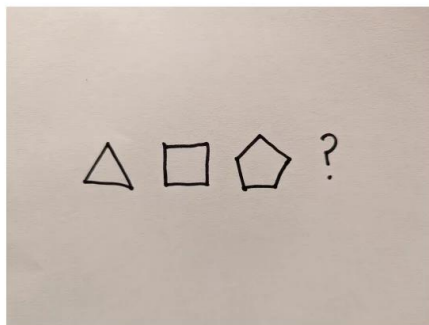
いずれもキャンパスからアクセスしやすい場所にあるので、ぜひ訪れてみてください！



画像も含めた「マルチモーダルな」対話も出現し始めた

● Google Gemini (2023.12)

Prompt



Look at this sequence of three shapes. What shape should come as the fourth shape? Explain your reasoning with detailed descriptions of the first shapes.

Model Response

The fourth shape should be a hexagon. The first shape is a triangle, the second shape is a square, and the third shape is a pentagon. The number of sides in each shape is increasing by one

画像を含む
質問への回答

手書き答案
採点

Prompt

Please create a blog post about a trip to New York, where a dog and his owner had lots of fun. Include a few pictures of the dog posing happily at different landmarks.

Model Response

Hi, I'm Cooper. I'm a 2 year old Goldendoodle who loves to travel. Last month, my family and I went to New York City. It was my first time in the Big Apple, and I had a blast!

We stayed in a hotel near Times Square. It was so much fun to see all the bright lights and people. We went to see a Broadway show, and I even got to meet some of the actors.

One of my favorite parts of New York City was Central Park. It was so big and beautiful. I loved running around and playing fetch. We also went to the Statue of Liberty and the Empire State Building. I was so excited to see all the famous landmarks.

I had so much fun in New York City. It's definitely one of my favorite places to travel. I can't wait to go back someday!

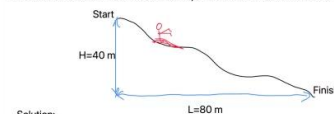
画像生成



Prompt

Here's a solution to a Physics problem by a student.

23. A skier slides down the frictionless slope as shown. What is the skier's speed at the bottom?



Solution:

The conservation of energy means that the energy of the skier is same at the beginning and at the end.
Start: $E = m g L$
End: $E = \frac{1}{2} m v^2$
 $\Rightarrow m g L = \frac{1}{2} m v^2$
 $\Rightarrow v = \sqrt{2 g L} = \sqrt{2 \times 9.81 \times 80} = 39.6 \text{ m.s}^{-1}$

Try to reason about the question step by step.

Did the student get the correct answer? If the solution is wrong, please explain what is wrong and solve the problem.

Make sure to use LaTeX for math and round off the final answer to two decimal places.

Response

The student did not get the correct answer. The student made a mistake in the calculation of the potential energy at the start of the slope. The potential energy at the start of the slope is given by mgh , where m is the mass of the skier, g is the acceleration due to gravity, and h is the height of the slope. The student used the length of the slope, L , instead of the height, H , in the calculation of the potential energy.

The correct solution is:

The conservation of energy means that the total energy of the skier is the same at the beginning and at the end of the slope.

At the start of the slope, the skier has only potential energy, which is given by mgh .

At the end of the slope, the skier has only kinetic energy, which is given by $\frac{1}{2} m v^2$

Therefore, we can write:

$$mgh = \frac{1}{2} m v^2$$

Canceling out the mass m on both sides, we get:

$$gh = \frac{1}{2} v^2$$

Solving for v , we get:

$$v = \sqrt{2gh}$$

Substituting the given values, we get:

$$v = \sqrt{2 \times 9.81 \times 40} = 28.01 \text{ m/s}$$

Therefore, the skier's speed at the bottom of the slope is 28.01 m/s.

<https://geminigoogle.cc/>

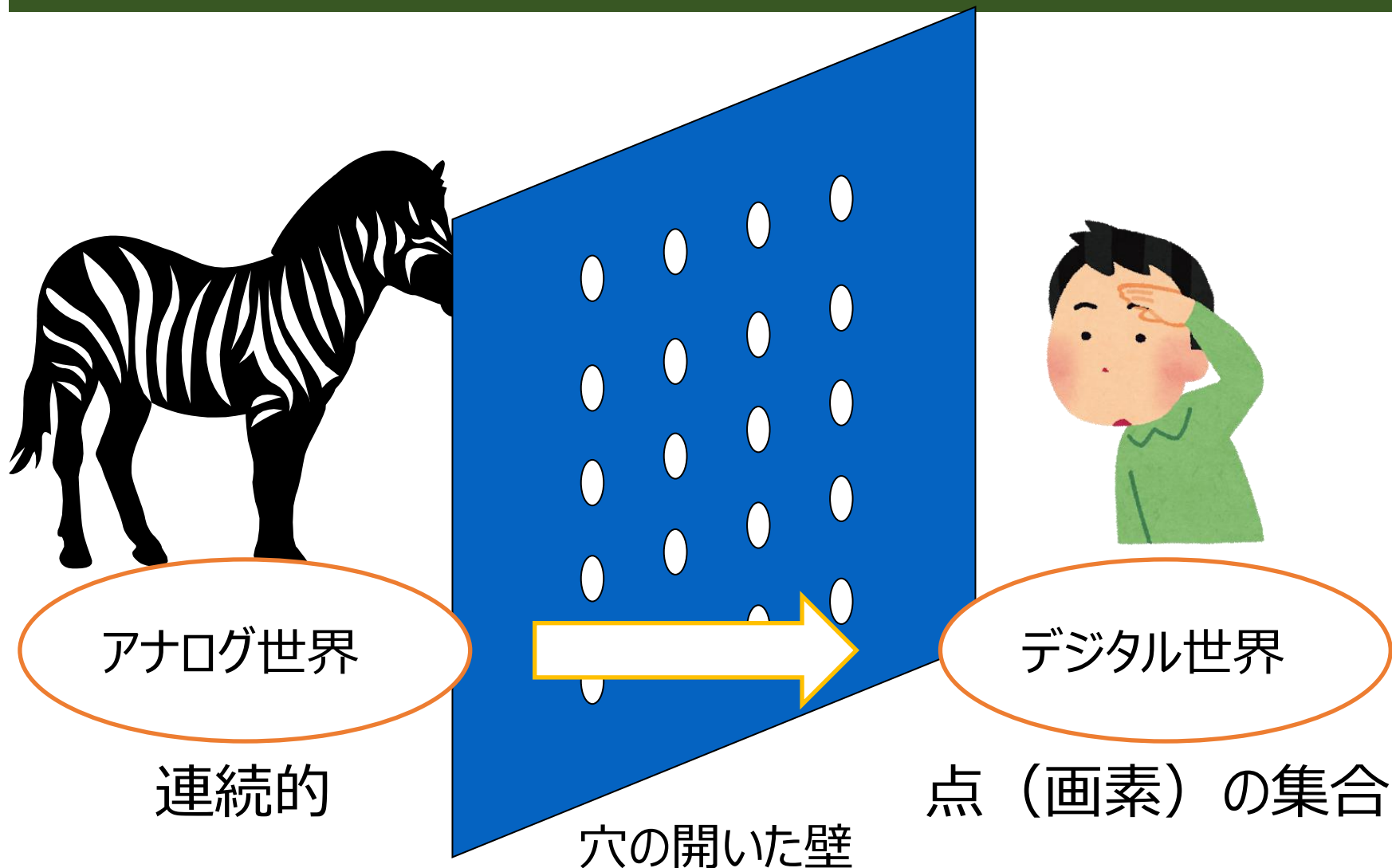
【付録】

コンピュータに画像が保存されるまでの処理

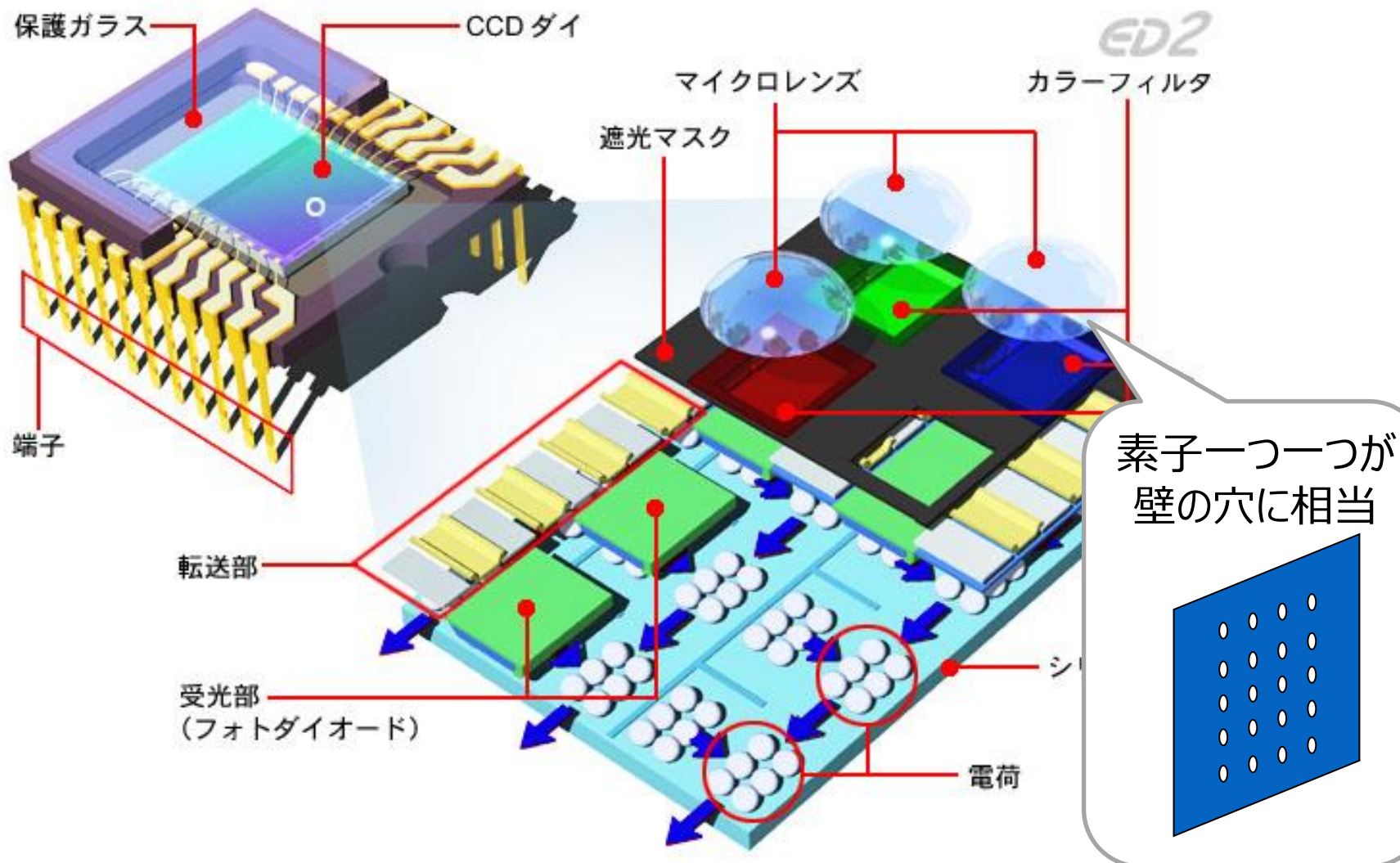
①標本化

いわゆる「アナログ→デジタル変換」と画像圧縮
その①

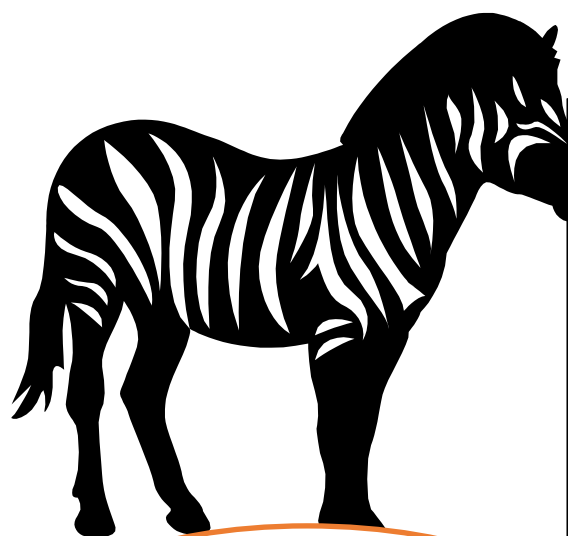
標本化(sampling)： デジタル画像生成の第一歩



標本化の方法： 有限個の撮像素子のそれぞれが光強度をセンシング

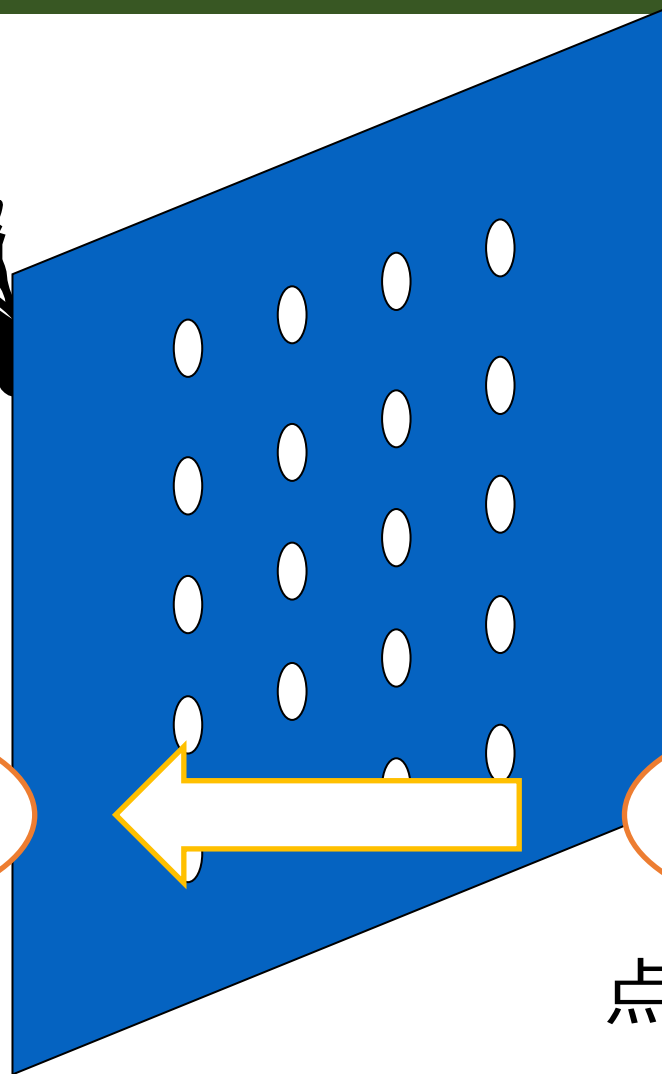


驚くべき事実： 逆もできる！



アナログ世界

連続的



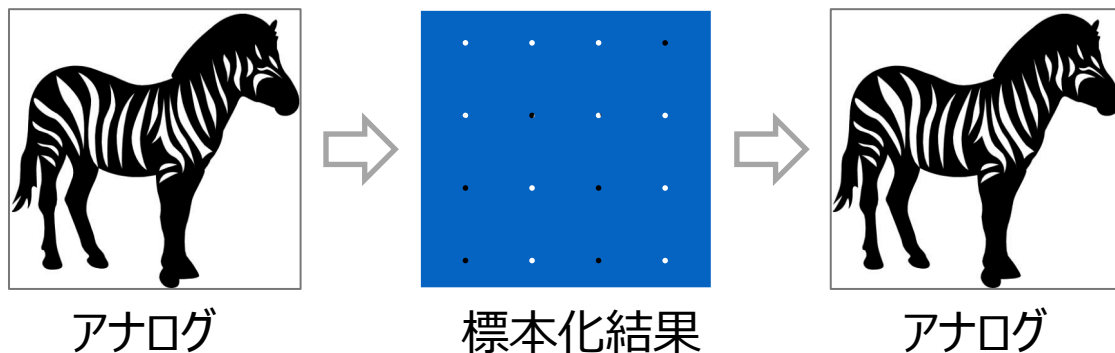
デジタル世界

点（画素）の集合

驚くべき事実！ 「標本化定理」Sampling theorem



- ある条件を満たせば、のぞき穴からの情報だけで、壁の向こうが完全に復元できる！
- デジタル画像からアナログ世界が戻せる！



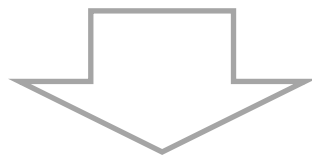
- そんなバカな！ 穴と穴の間の見えてない部分がもともとどうだったかとか、そんなのわかるわけがない！
- 「ある条件を満たせば」のところが怪しい...



標本化定理

条件

連続データの最高周波数が f_{\max} のとき,
周期 $T < 1/(2 f_{\max})$ 毎に標本化すれば,

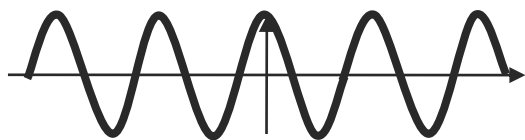
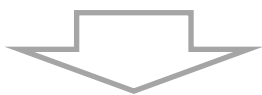
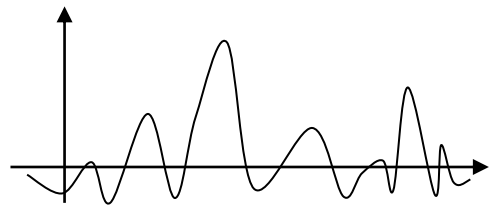


標本化結果（デジタル）から
連続データ（アナログ）を再現可能

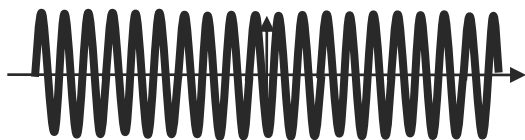
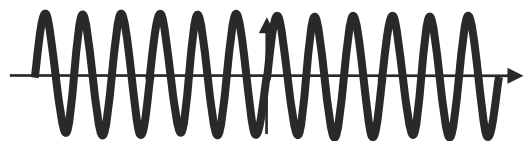
- 周波数とは？

余談：画像にも「周波数」成分があります

時系列データ(1次元信号)

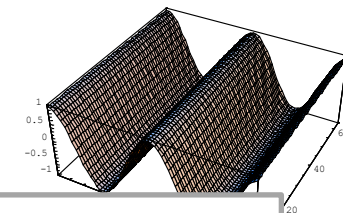
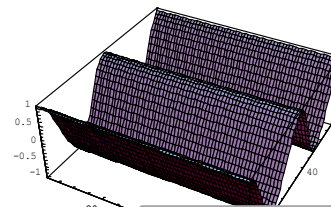
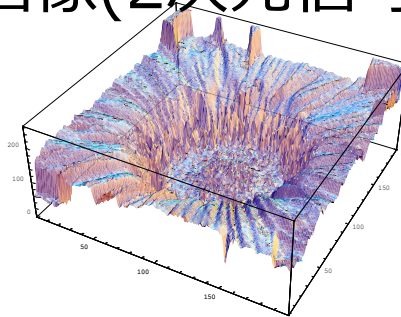


低い周波数

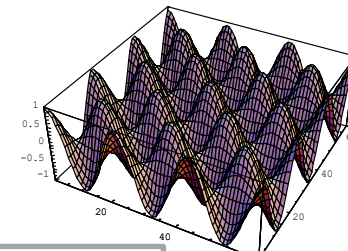
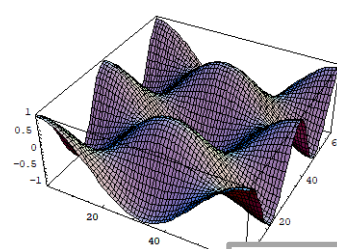


高い周波数

画像(2次元信号)



緩やかな明るさ変化を表現

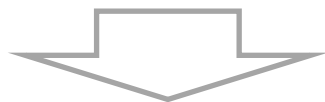


細かい模様を表現

話は戻って，標本化定理

条件

連続データの最高周波数が f_{\max} のとき，
周期 $T < 1/(2 f_{\max})$ 毎に標本化すれば，



標本値（デジタル）から
連続データ（アナログ）を再現可能

話は戻って、標本

壁の向こうの世界に含まれる
最も細かい変化

条件

連続データの**最高周波数**が f_{\max} のとき、
周期 $T < 1/(2 f_{\max})$ 毎に標本化すれば、

細かければ細かいほど
短い周期で標本化すれば

標本値（デジタル）から
連続データ（アナログ）を再現可能

標本化定理の「タネアカシ」

1. 最高周波数がわかっている
2. その最高周波数に応じた周期で標本化している

標本化定理～最も極端な場合で理解する

- 最高周波数 f_{\max} が0であること（＝壁の向こうは変化なし＝一定色の世界が広がっている）がわかっているならば、
- 周期は無量大でOK
 - =標本点(穴)は1つでOK

条件

連続データの最高周波数が f_{\max} のとき、
周期 $T < 1/(2 f_{\max})$ 毎に標本化

たった一つの
穴だけど

$f_{\max}=0$ と知っているので

壁の向こうは
この色 1 色の世界が
広がっていることがわかる

アナログ(物理世界)

標本化結果(1点のみ)

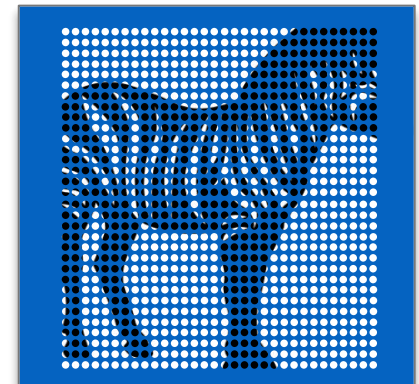
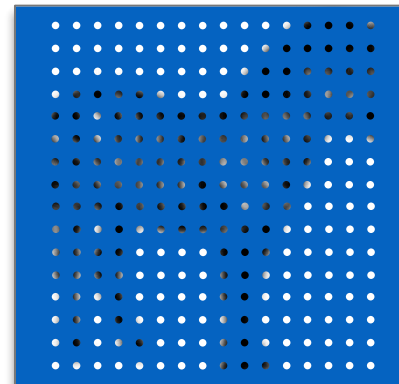
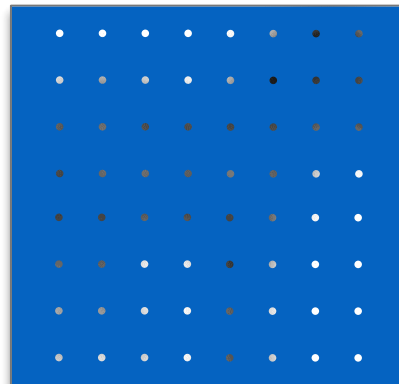
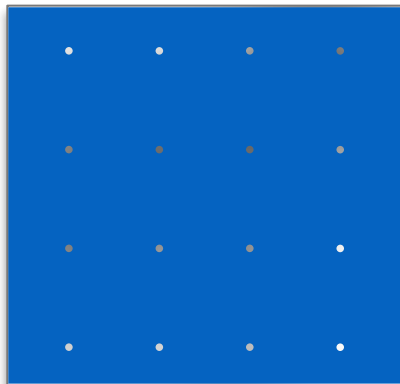
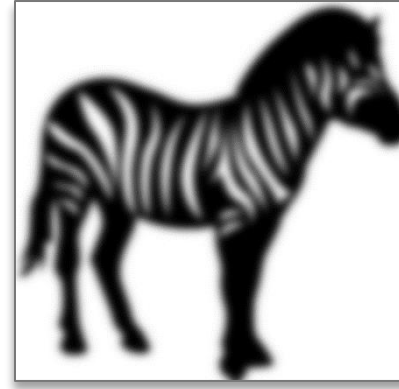
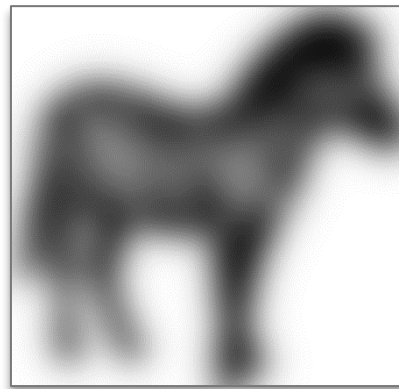
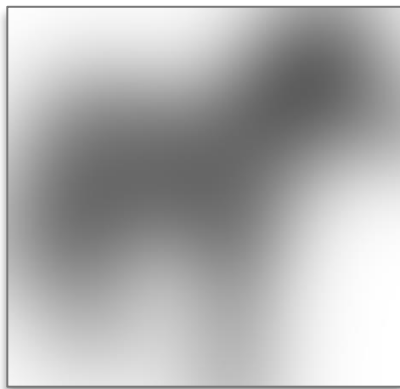
完璧な復元結果

標本化定理～もっと一般的な場合も理解する

条件の基準になるもの
(画像の細かさ = 周波数)

最高周波数が低い

最高周波数が高い

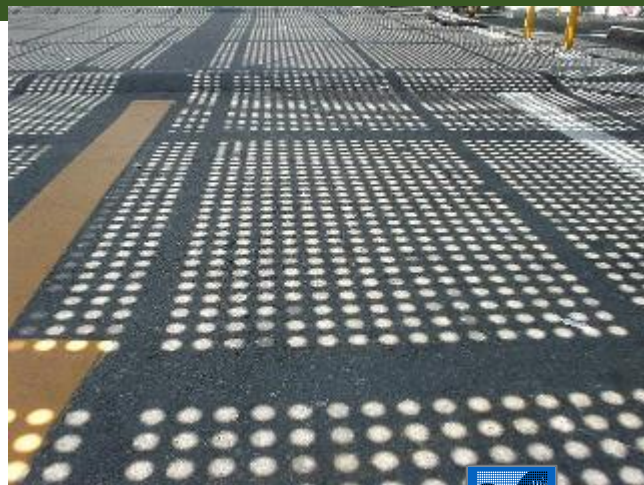


間隔大

条件を満たすサンプリング間隔

間隔小

標本化周期による画像の変化と、 条件を満たさない場合に起こる「エイリアシング」



640x480



320x240



同じ範囲を
写しているのに、
画素数が多い
(=高解像度)ほど
短い標本化周期

ん!?



ん!?

160x120



んん!?



80x60



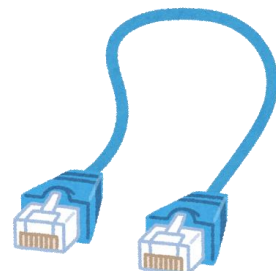
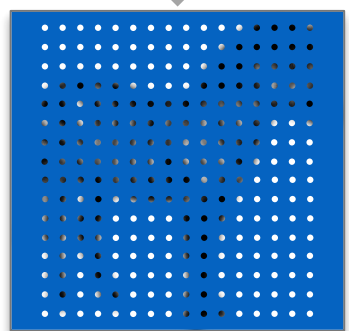
画像の最高周波数に比べ、標本化周期が
長すぎた結果、生じてしまった偽の模様(エイリアシング)

標本化されたデータを介して，画像を見ている！

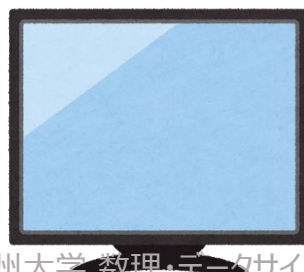
アナログ



標本化結果を
送受信・蓄積

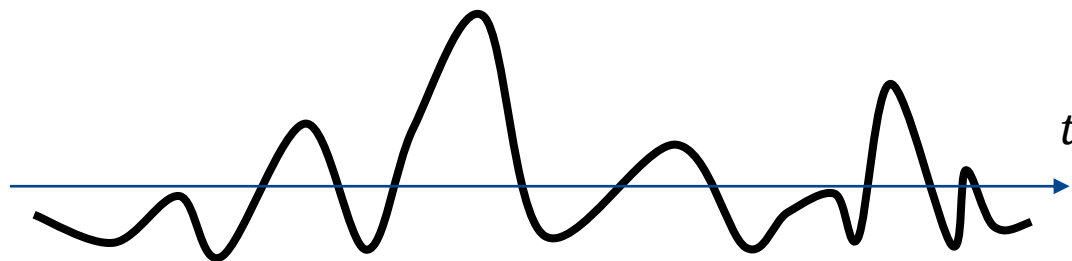


復元結果
(アナログ)

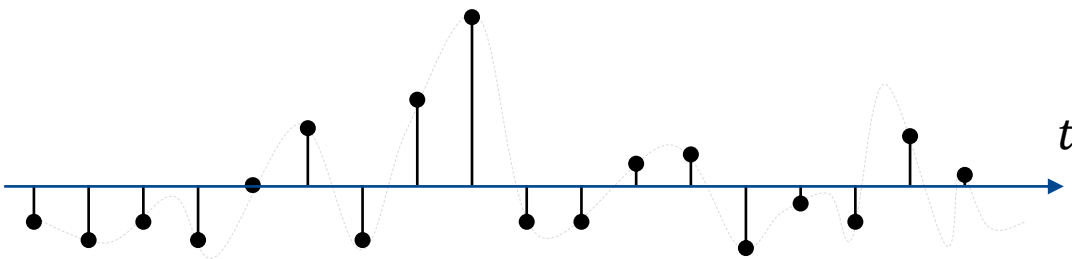


音声のような時系列データも同様！ そのおかげでデジタル機器やCDで音楽が聴ける

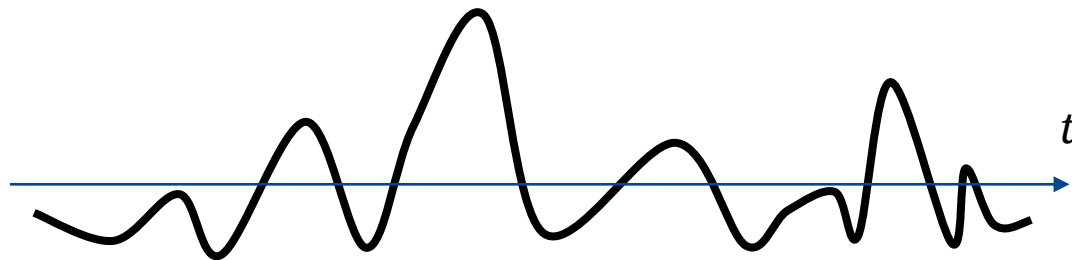
アナログ
 $x(t)$



標本化結果を
送受信・蓄積



復元結果
(アナログ)

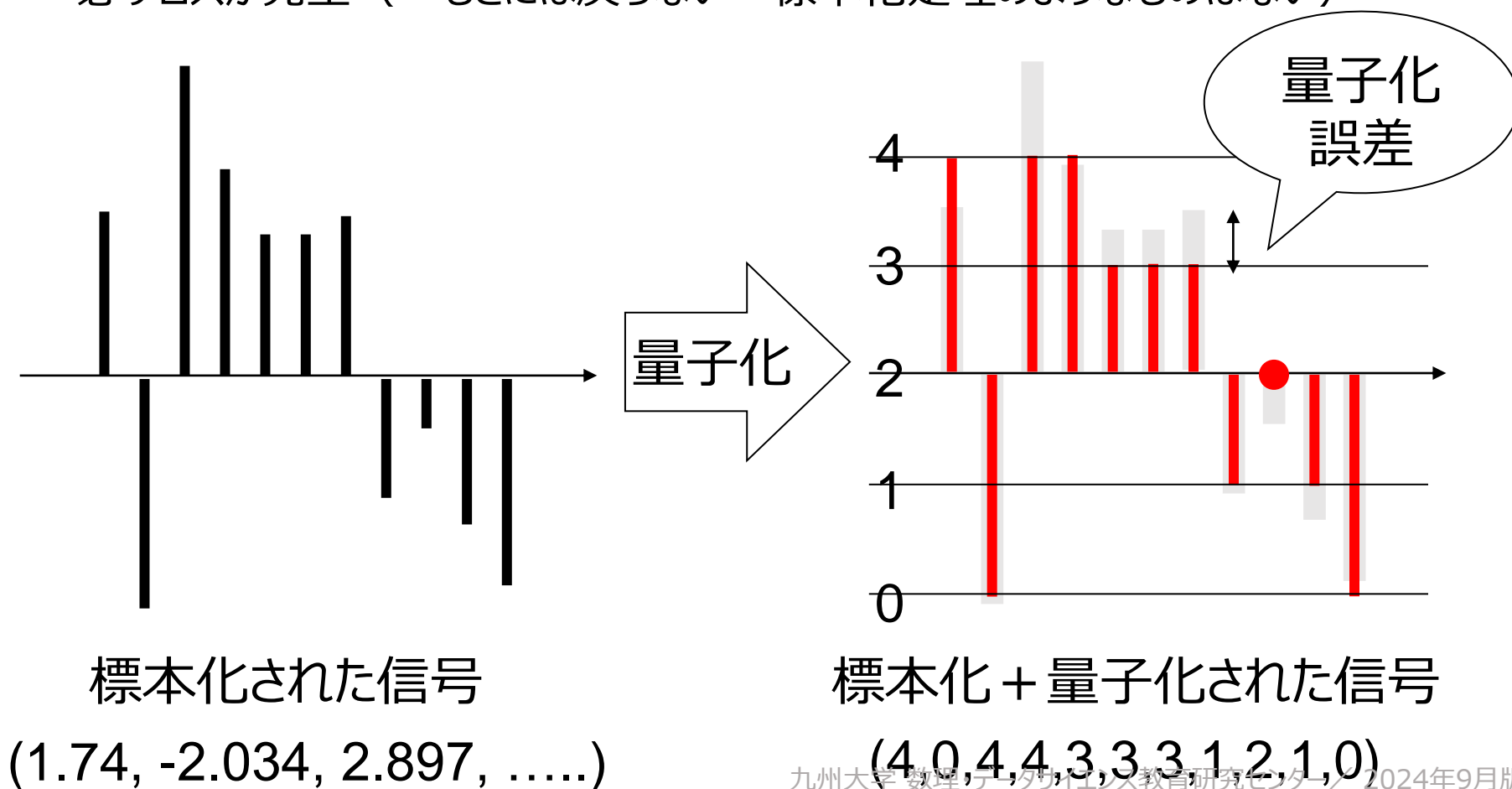


【付録】 コンピュータに画像が保存されるまでの処理 ②量子化

いわゆる「アナログ→デジタル変換」と画像圧縮
その②

量子化(quantization)の概念

- 連続量 → (四捨五入的处理) → キリのよい数字に (有限桁数に)
 - 必ずロスが発生 (=もとには戻らない = 標本化定理のようなものはない)



量子化レベルによる画像の変化

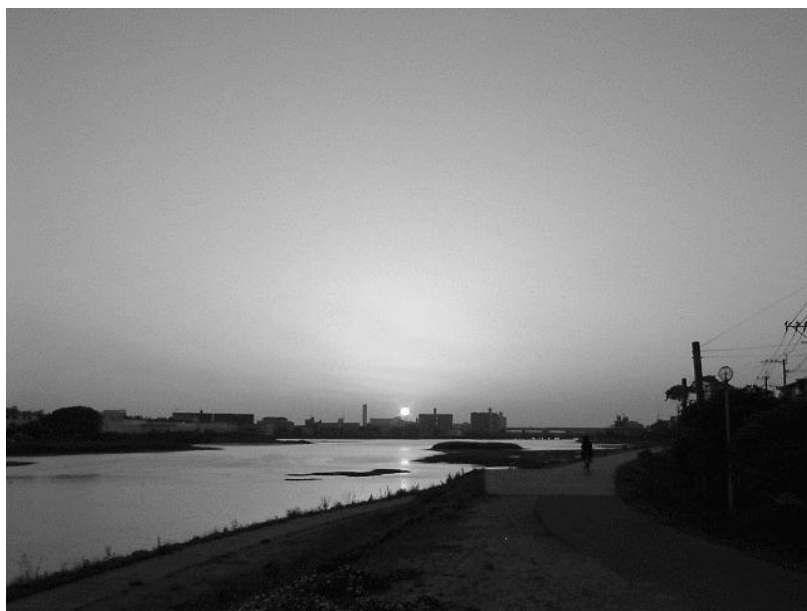
256 (8bit)



64 (6bit)



32 (5bit)



16 (4bit)



量子化レベルによる画像の変化(つづき)

8 (3bit)



4 (2bit)



2 (1bit)



量子化レベルによる画像の変化

RGB各256レベル
(8bit×3=24bit)



64 (6bit×3)



32 (5bit×3)



16 (4bit×3)



量子化レベルによる画像の変化(つづき)

RGB各8レベル
(3bit×3)



2 (1bit×3)



4 (2bit×3)



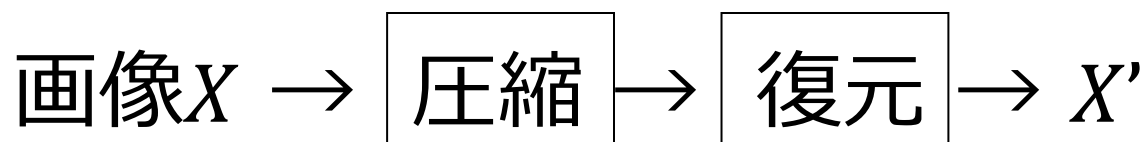
【付録】

コンピュータに画像が保存されるまでの処理

③画像圧縮

いわゆる「アナログ→デジタル変換」と画像圧縮
その③

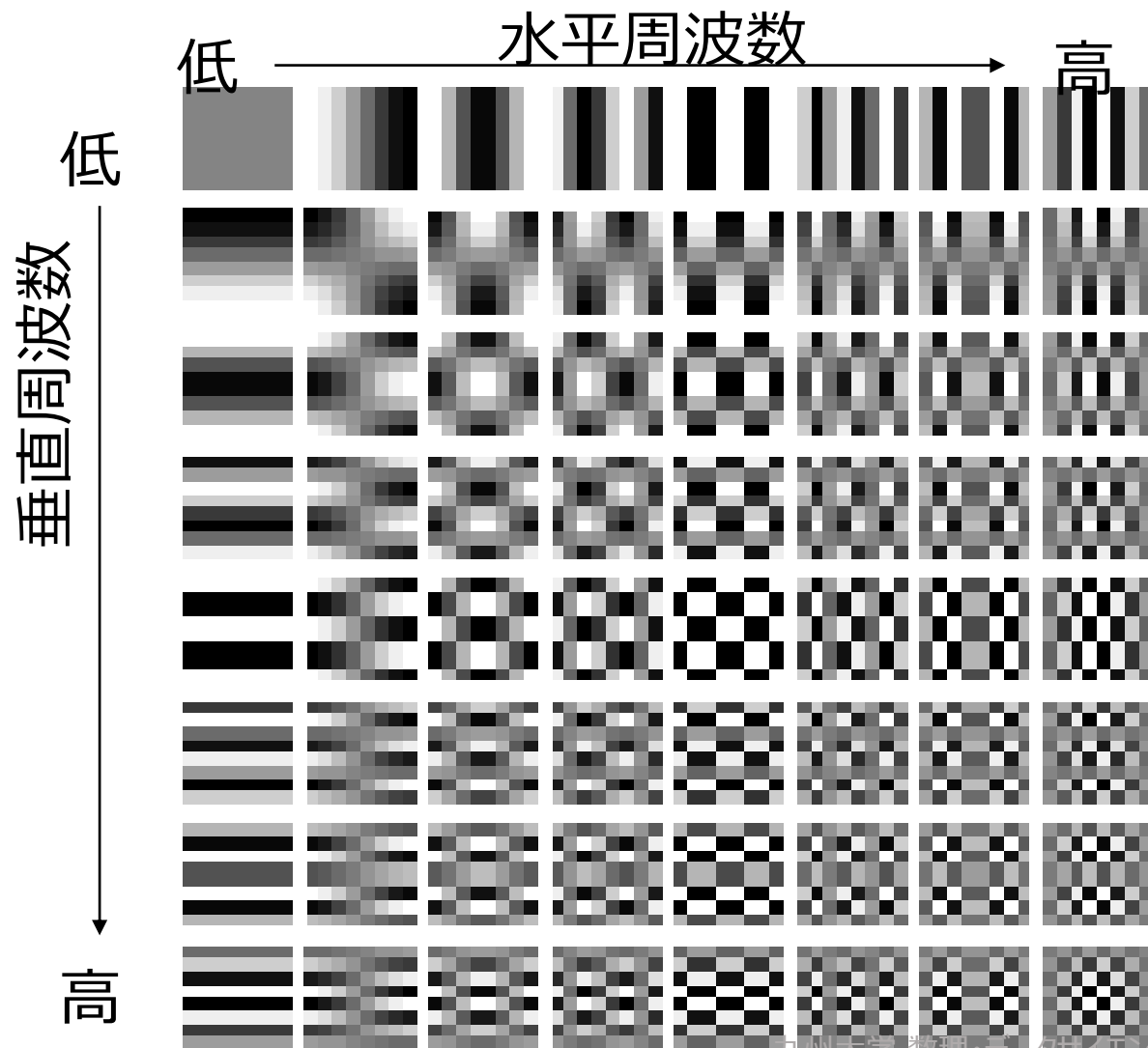
画像圧縮～種類と応用先



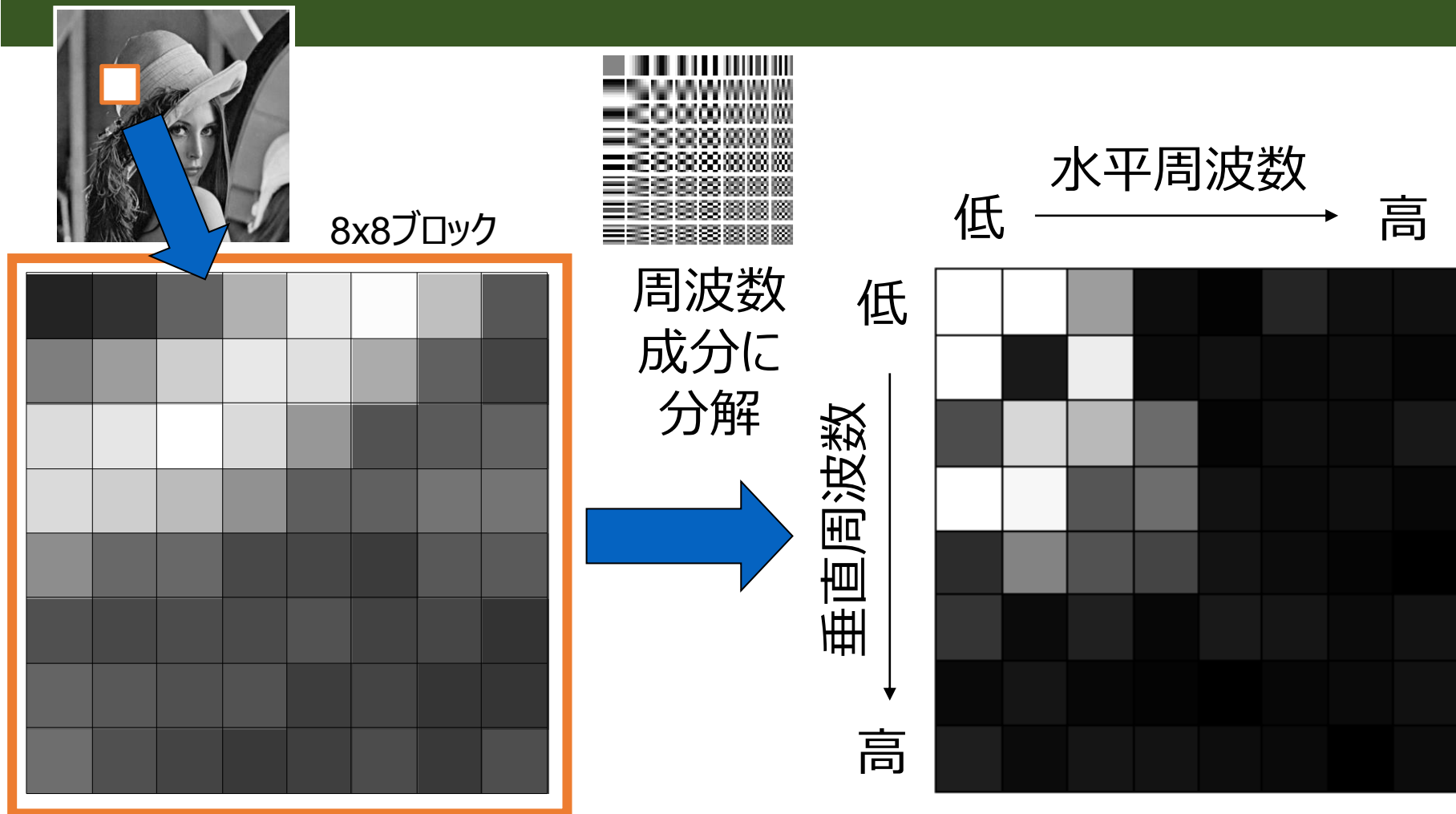
- **可逆**符号化 $X = X'$
 - 医用画像, 文化財 (古文書) 保存, リモートセンシング画像
 - 圧縮率 $1/2 \sim 1/3$ 程度
- **非可逆**符号化 $X \neq X'$ but $X \doteq X'$
 - 通信用画像(Web上の画像, デジカメ画像), 動画
 - 圧縮率 $1/10 \sim 1/50 \sim \dots$

普通のJPEGは
こちら

画像は各周波数に分解できる

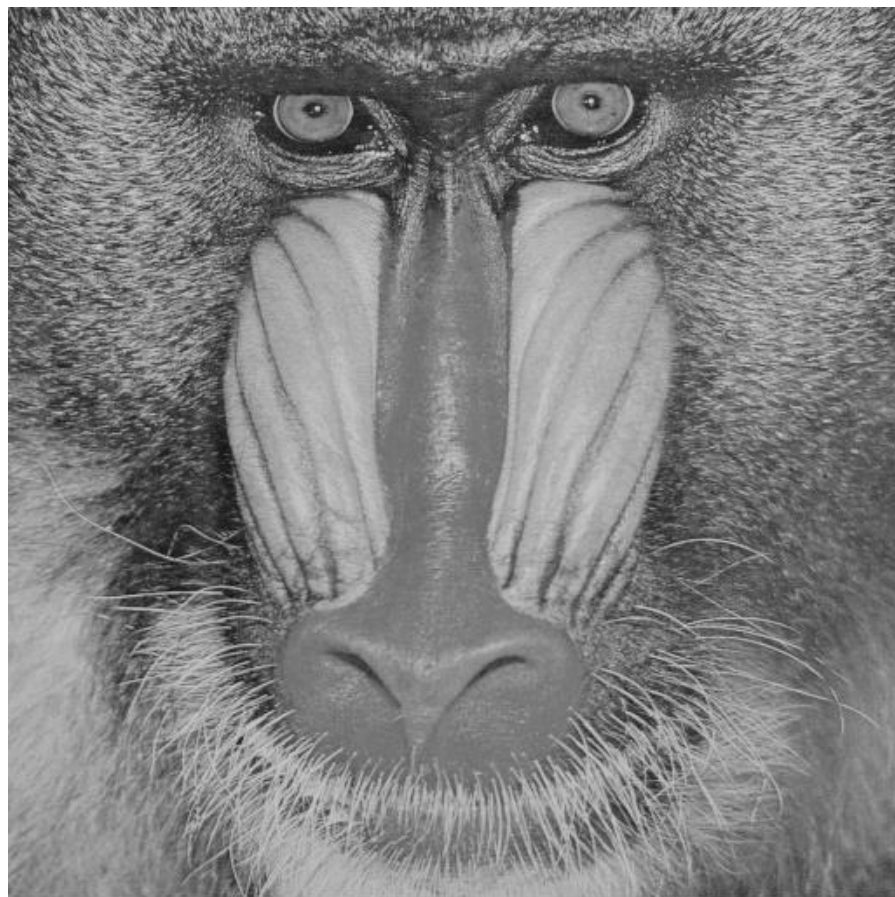
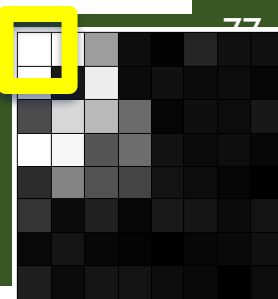


JPEGの原理:小ブロック毎に周波数分解

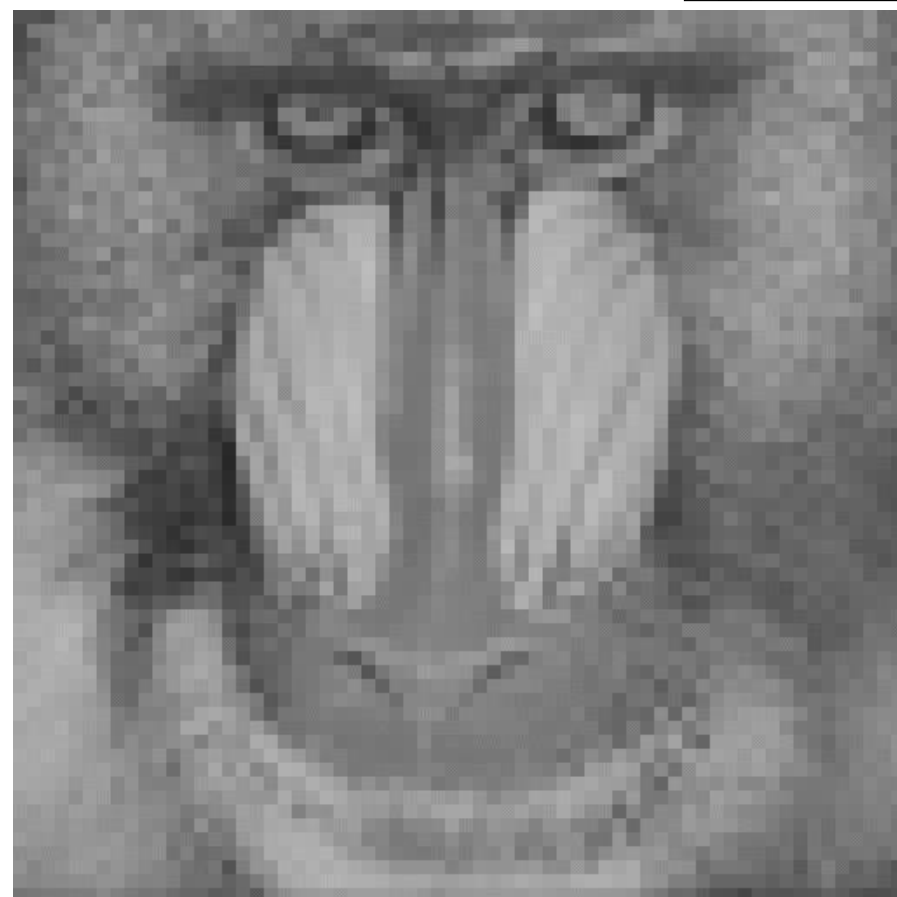


値の大きな成分は低周波数域に集中する (energy compaction)

最も低い周波数成分だけで表現

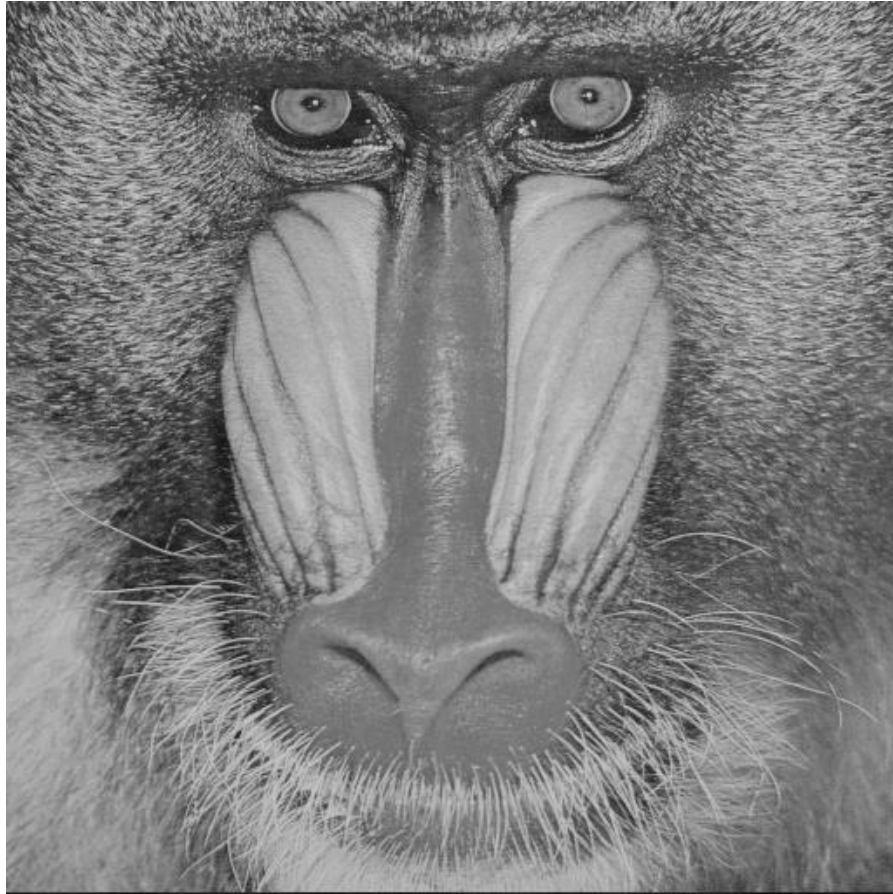
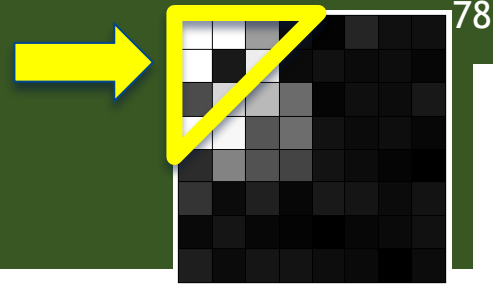


原画像

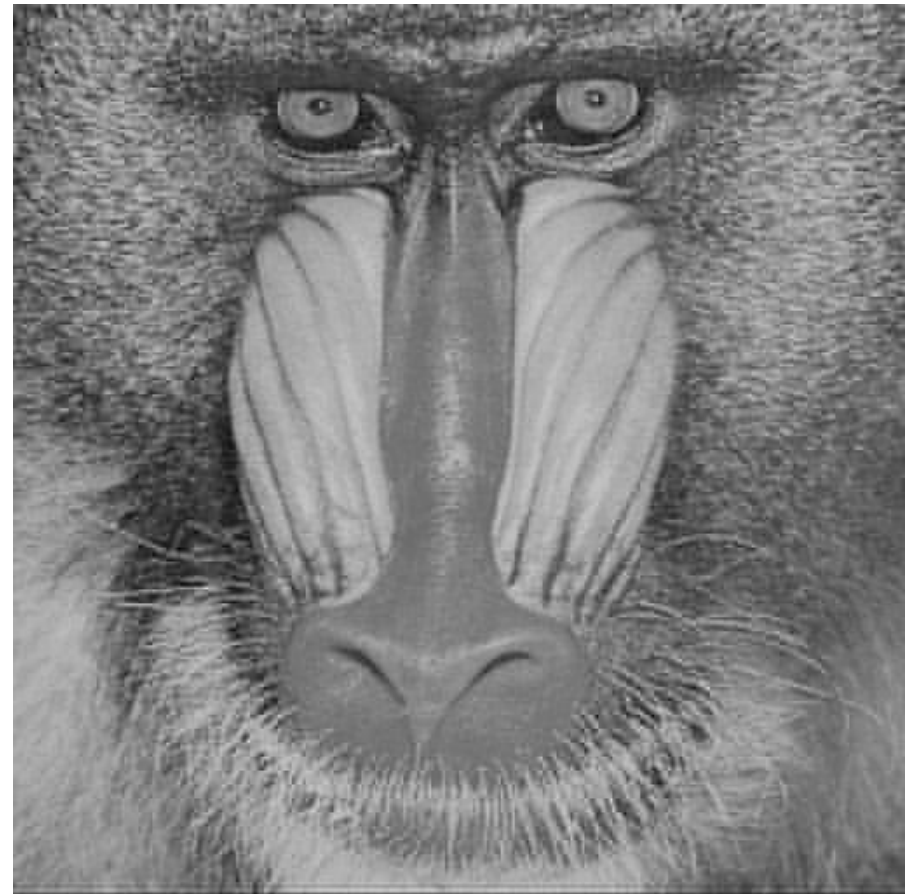


圧縮後

もう少しだけ使って表現

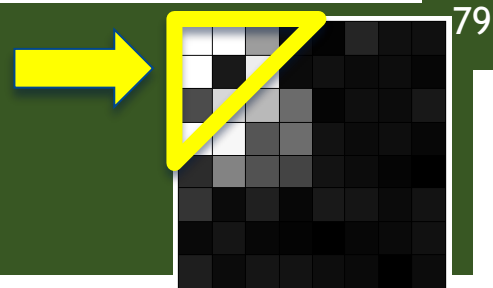


原画像

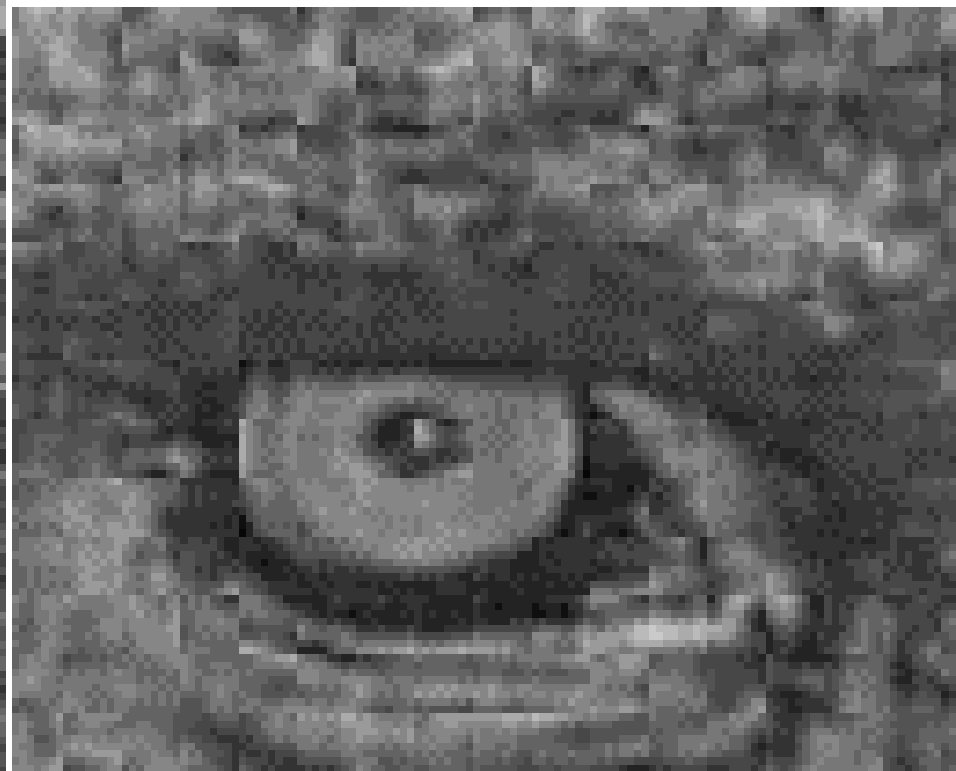


圧縮後

拡大版



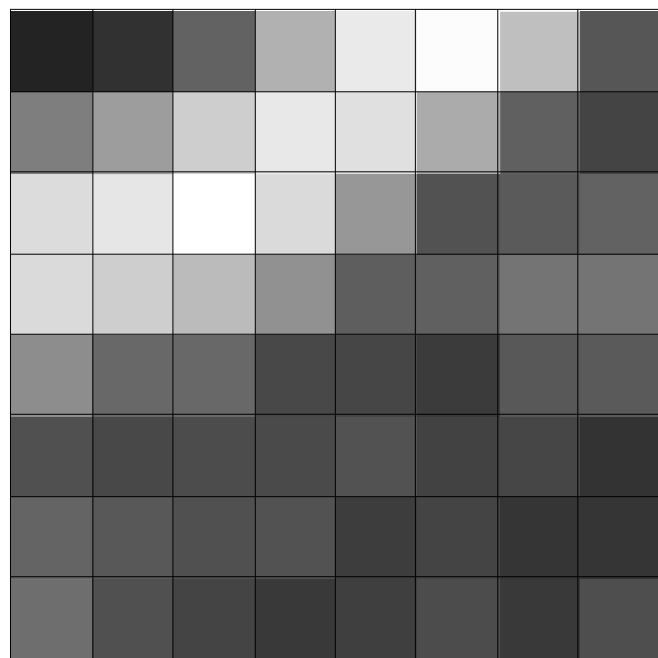
原画像



圧縮後

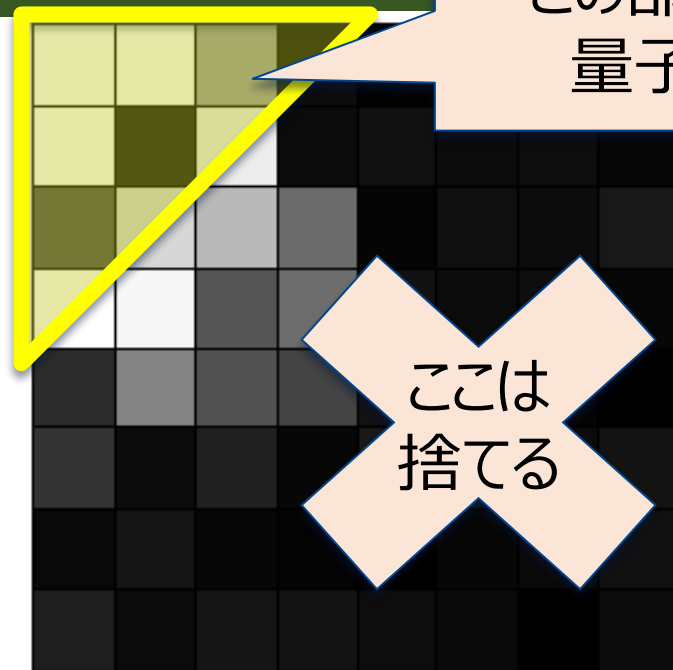
ブロック歪み(後述)が良く見えます

なぜ圧縮できる？



(濃淡画像) こちらを
送らずに

圧縮
復元



さらに
この部分を
量子化

ここは
捨てる

(周波数成分)
こちらを送る

しかしブロック歪には要注意！
ブロック歪が画像処理に悪影響することもある！

以前やった，以下の話と全く同じ！

カレーの例で説明すると(1/2)：
プロの味でなくても，ご家庭の味でOKなケース！

材料セット

$e_1, \dots, e_i, \dots, e_j, \dots, e_d$

$$\alpha_i = x \cdot e_i$$

$$\sum \alpha_i e_i$$

は除く

レシピ(分析結果)

→ α_1 グラム

→ α_i グラム

→ α_j グラム

→ α_d グラム

高周波数
成分が
相当する

常にほとんどゼロ
(プロの隠し味)

同じじゃないけど，そこそこ旨い

ブロック歪み(block-noise)

32kbyte



10kbyte



3kbyte



歪み小
(圧縮率低)

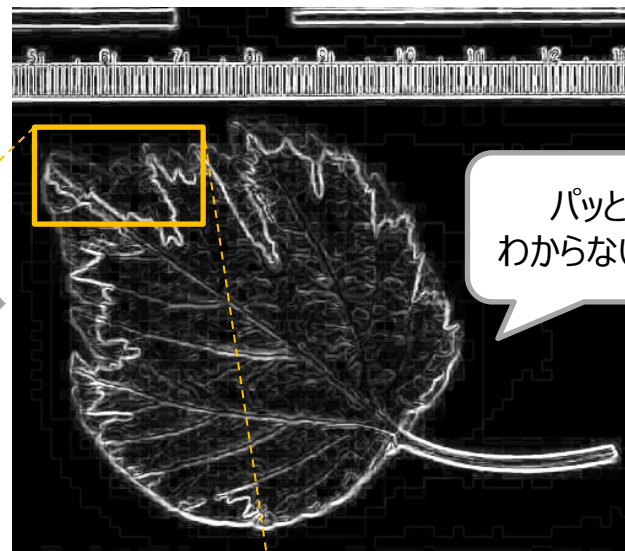
歪み大
(圧縮率高)

ブロック歪み(block-noise)の画像処理への影響

歪みあり(圧縮率高/10kbyte)

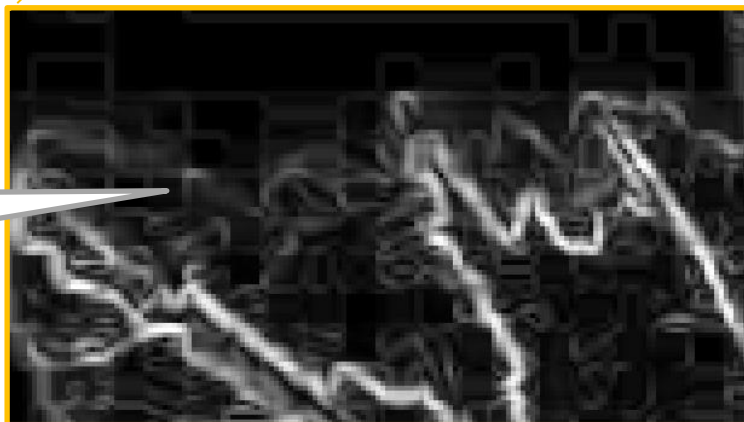


エッジ抽出結果

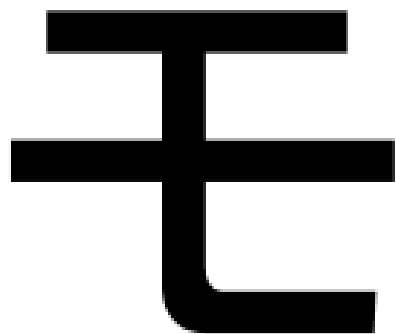


パッと見
わからないが...

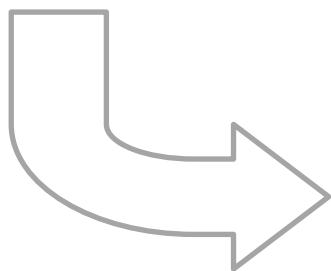
ブロック状の凸凹が...
これで形状解析したりすると
変なデータが出るかも...



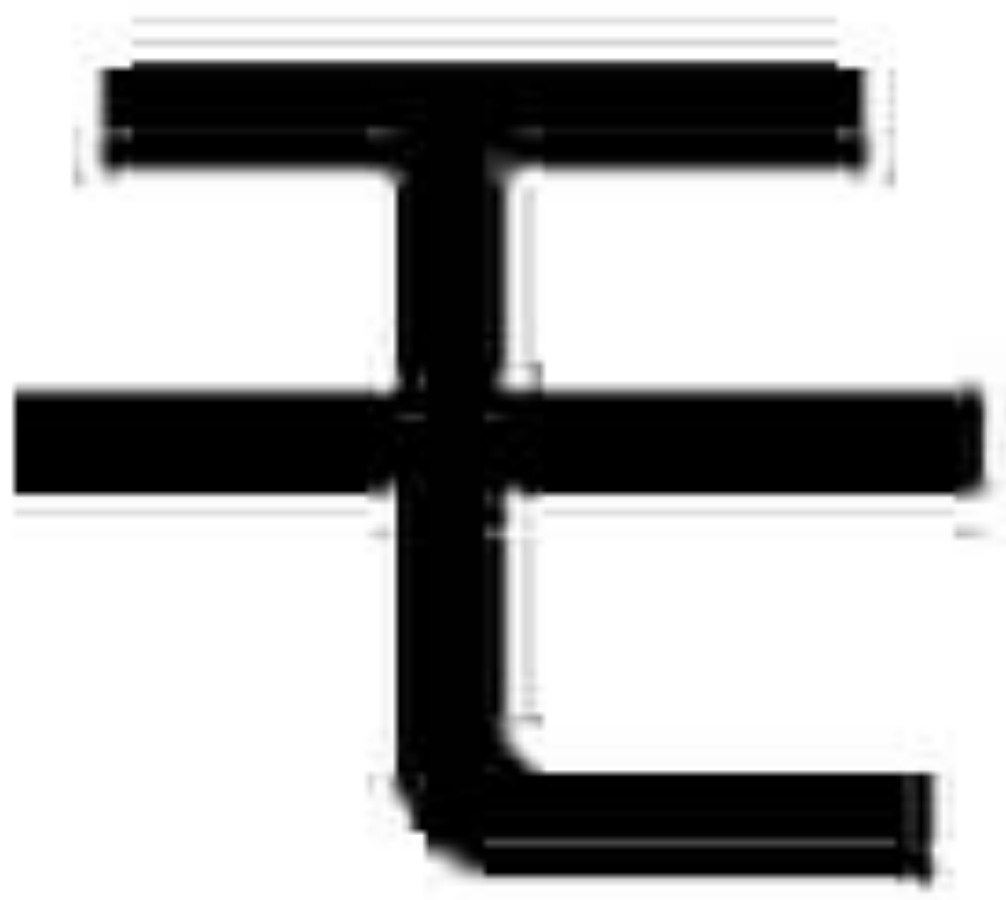
モスキートノイズ(mosquito noise)



原画像

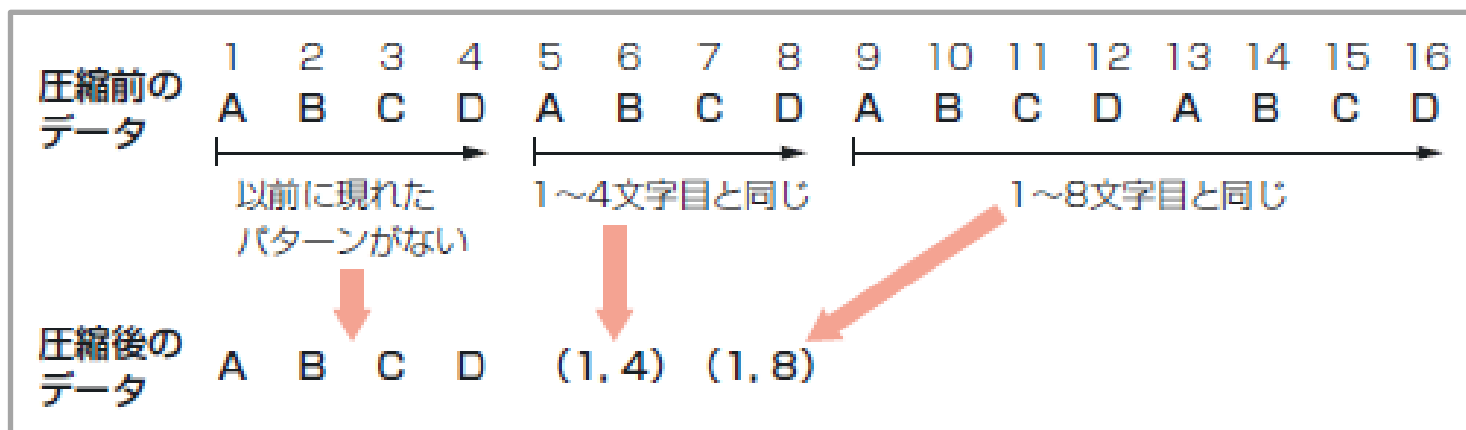


JPEG圧縮後



可逆圧縮 PNG

- 汎用圧縮方式 (LZ77圧縮)が中心技術
 - 似たパターンが繰り返し出現することに着目した手法



<http://itpro.nikkeibp.co.jp/article/COLUMN/20061012/250605/?P=7>

- GZIP等でも利用
- 画像特有の性質を積極的に活用した手法ではない
- 色は24ビットRGB
 - RGB各8ビット=JPEGと同じ

【付録】 色の話

なぜ、R(赤), G(緑), B(青)の組で黄色が表現できるのか？

コンピュータにおいて、カラー画像は
3つの単色画像の重ね合わせで表現される



=



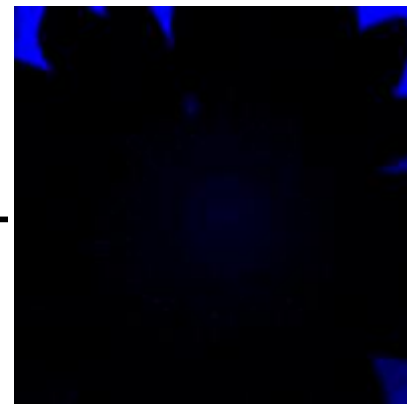
R成分

+



G成分

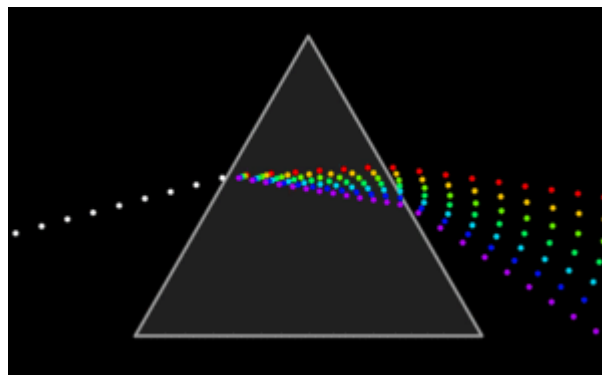
+



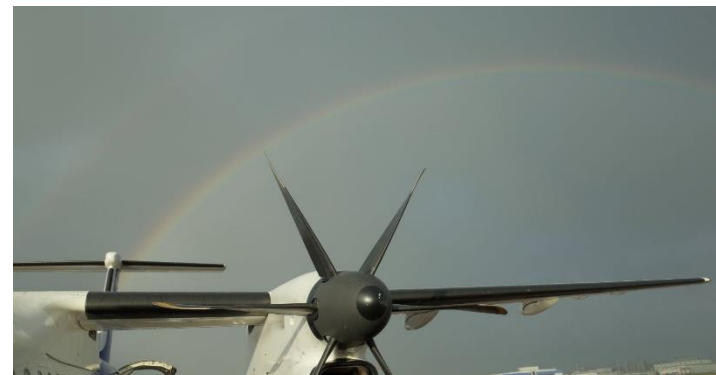
B成分

ところで、色は光の波長と関係する。
波長が変われば色が変わる

「可視光」

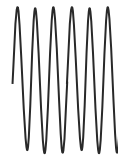
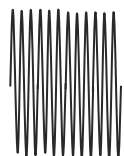


wikipedia



波長 = 380nm

750nm



あれ？ 色って R + G + B では？？

要するに、人間による色の「知覚」はいい加減

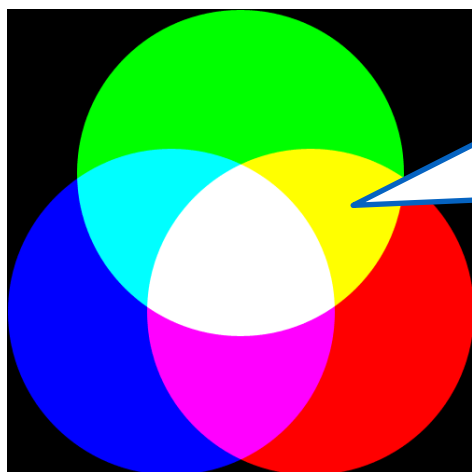


単一波による黄色

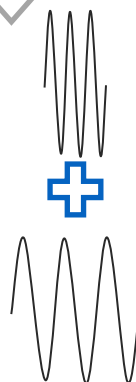


wikipedia

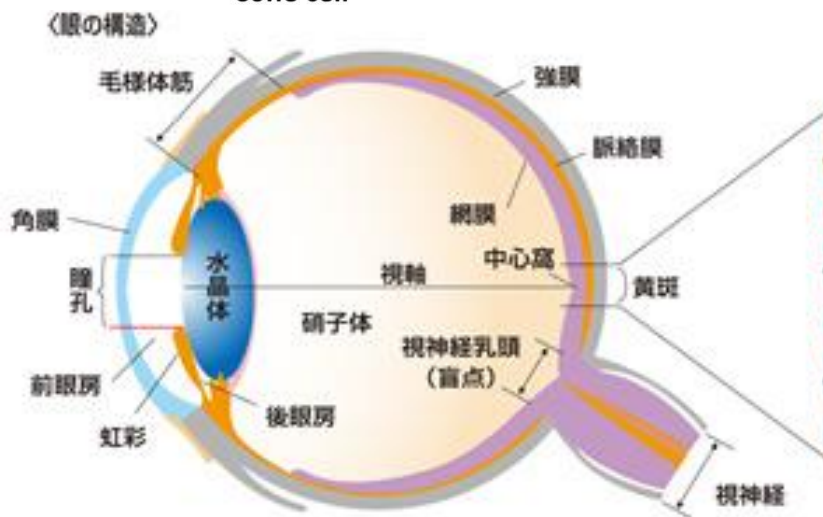
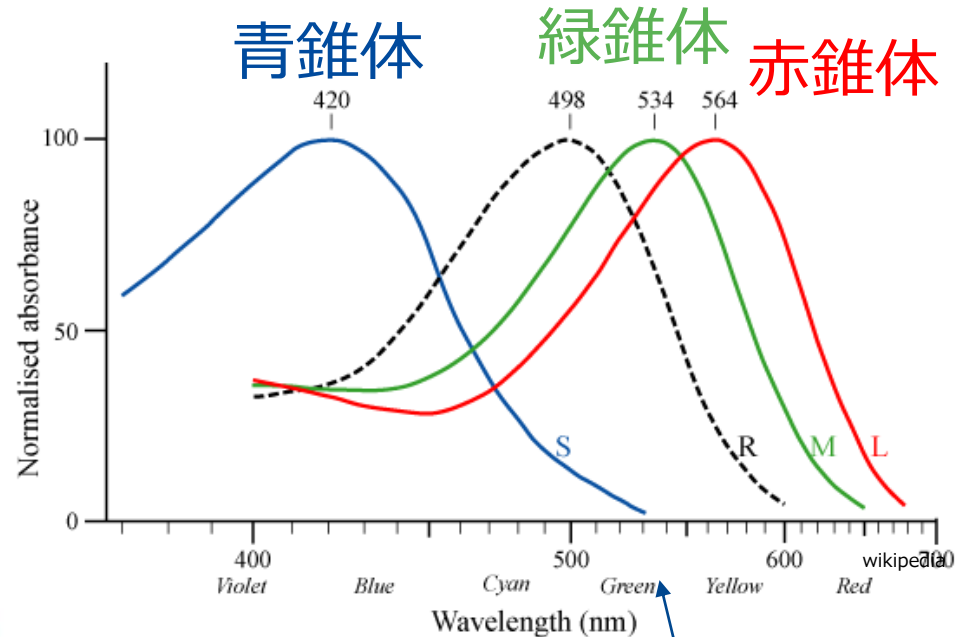
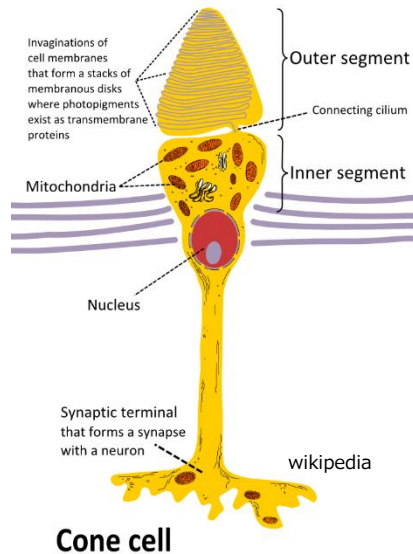
(波形としては全然違うのに)人間はこれらを区別できない！



RとGを
混色して
できる
黄色



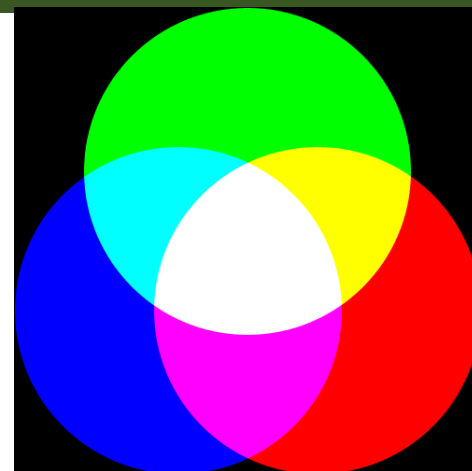
区別できないのは人間側のセンサ(視細胞)が RGBの3色担当しかないから



これら3タイプの視細胞の
反応の組み合わせで色を知覚

見る人間が3センサしかないので、コンピュータでも 3色のブレンドでOK：RGB表色系

- R (赤、700nm)
- G (緑、546.1nm)
- B (青、435.8nm)



- この3原色の混ぜ具合によって、色を表現

$$\text{色} = \alpha R + \beta G + \gamma B$$



$$(\alpha, \beta, \gamma)$$

の3つの数字で色を表す

他の表色系： マンセル表色系（HSV表色系）

● 3つの値

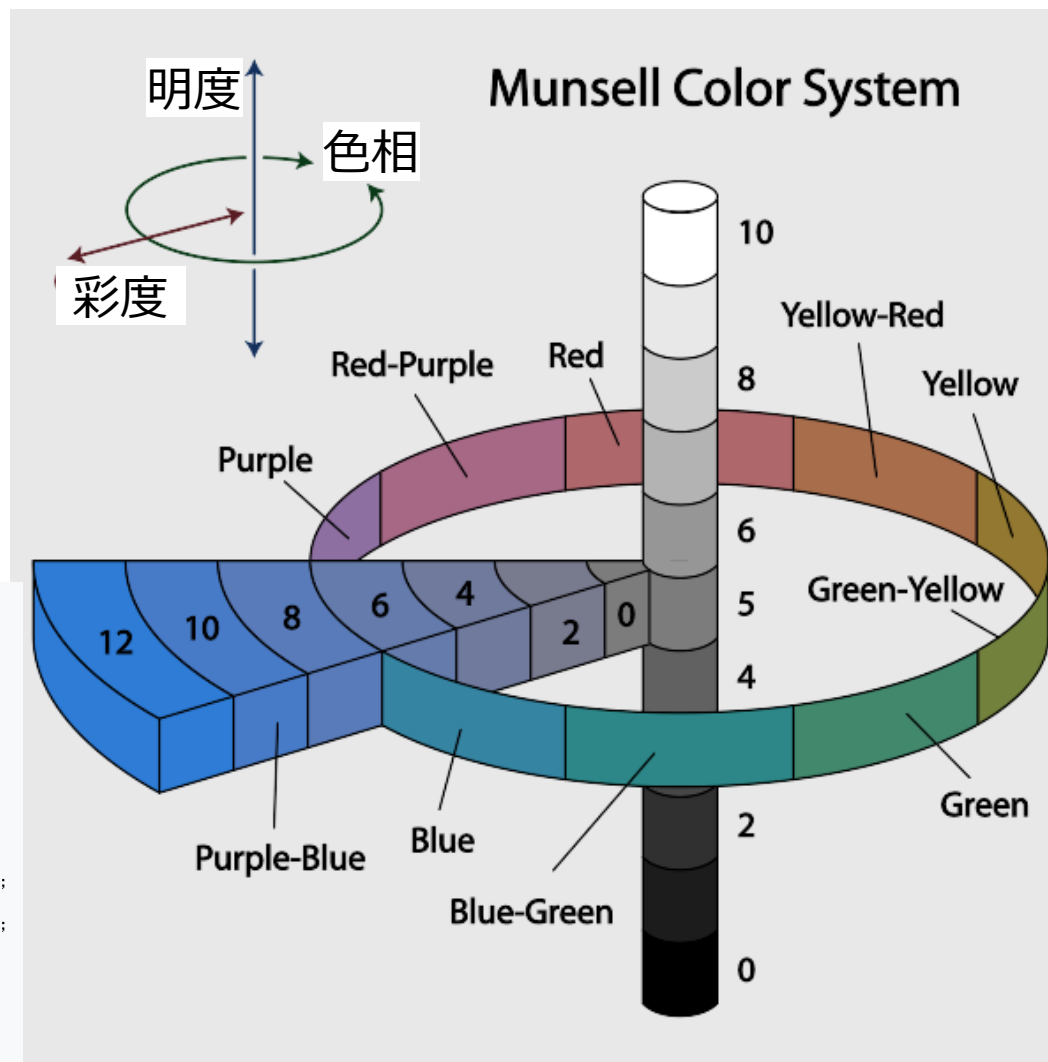
大雑把に言うと

- 色相(Hue, 色の波長)
- 彩度(Saturation, 鮮やかさ)
- 明度(Value, 明るさ)

で色を表現

● R,G,Bの値から H,S,Vの値を 計算可能

```
// (float r, float g, float b)
float max = r > g ? r : g;
max = max > b ? max : b;
float min = r < g ? r : g;
min = min < b ? min : b;
float h = max - min;
if (h > 0.0f) {
    if (max == r) {
        h = (g - b) / h;
        if (h < 0.0f) {
            h += 6.0f;
        }
    } else if (max == g) {
        h = 2.0f + (b - r) / h;
    } else {
        h = 4.0f + (r - g) / h;
    }
}
h /= 6.0f;
float s = (max - min);
if (max != 0.0f) {
    s /= max;
}
float v = max;
```



【付録】 具体的な画像データ分析の例： 画像の2値化

濃淡画像を“わざわざ”白黒画像にする？
でも利用者は非常に多い，基本的画像解析技術。
こんなシンプルな話でも，今でも研究されているから驚き
(要するに完璧な方法はまだ存在しない)

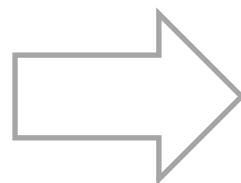
画像の2値化

領域境界が
ハッキリする

- 濃淡画像を白と黒の2階調に変換



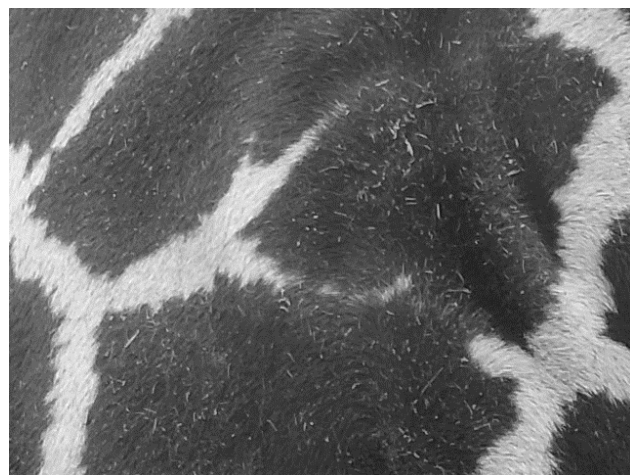
原画像（濃淡画像）



2値画像（白黒画像）

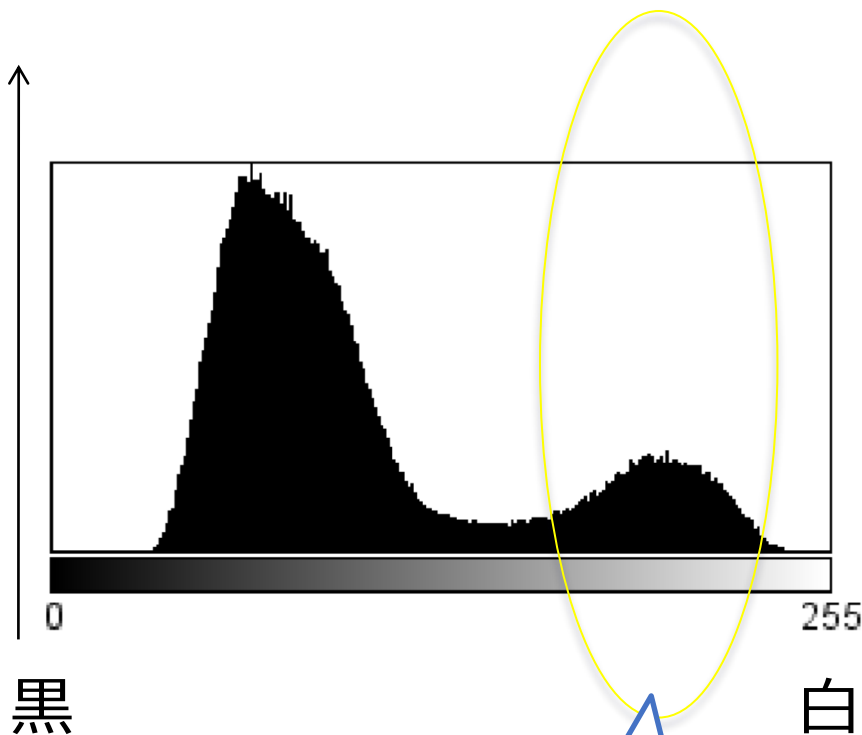
- 利用目的の例
 - 前景（物体）と背景の分離，各種成分抽出（連結成分解析，後述）
 - ノイズ除去
 - 画像の高速処理，画像の圧縮蓄積

2値化の手がかり：ヒストグラム（頻度）



キリン(?)の表皮
(濃淡画像)

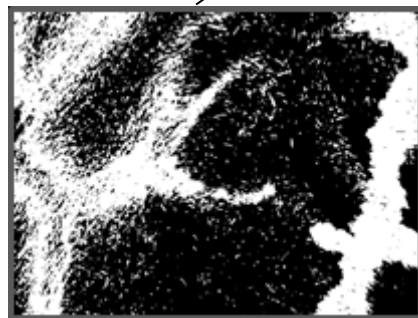
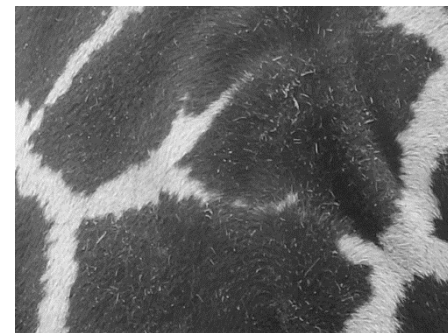
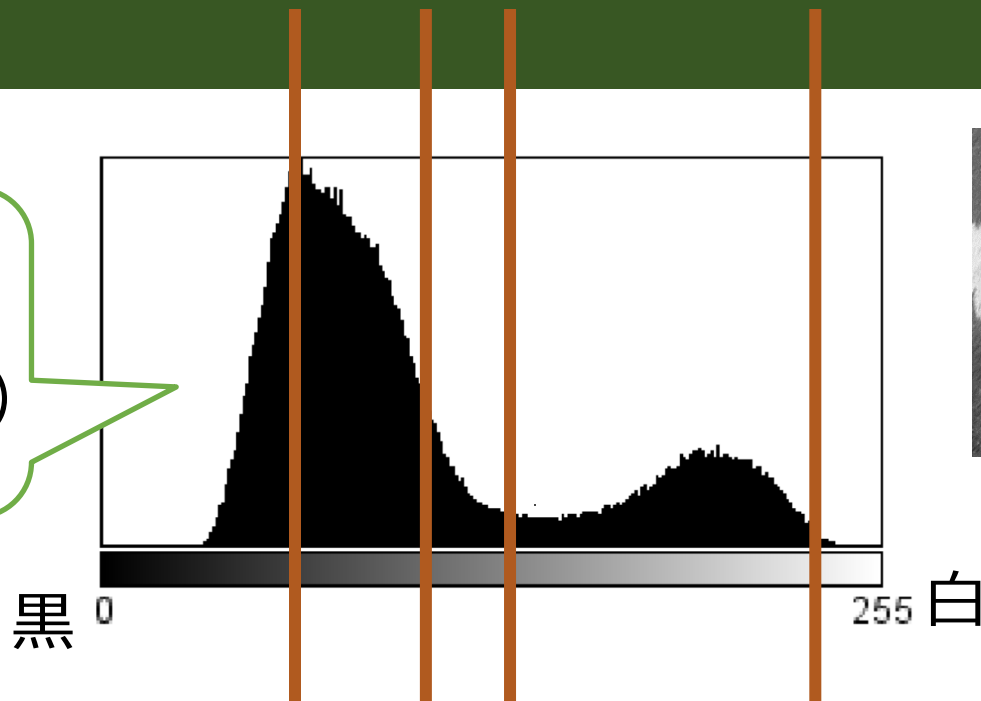
画素数



どうもこの辺が
「白線」部分？

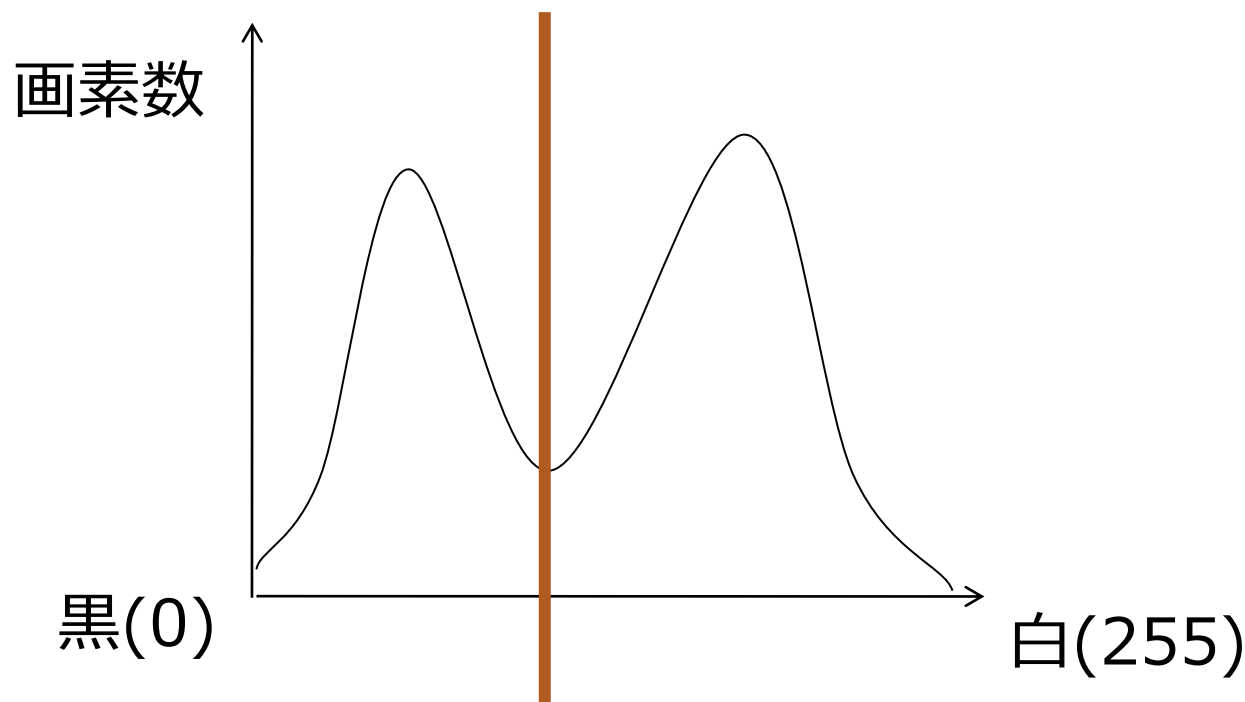
2値化の手がかり：ヒストグラム

どこに
「しきい値」
(threshold)
を設定？



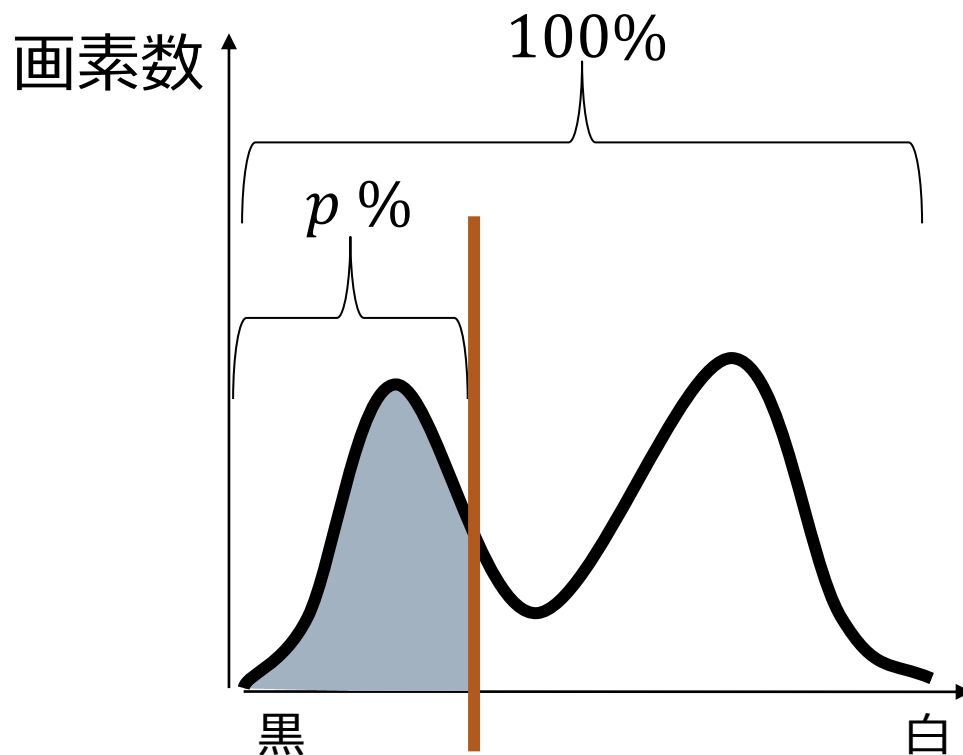
2値化のための「しきい値」決定法 (1/3) : モード法

- ヒストグラムの谷 = しきい値



2値化のための「しきい値」決定法 (2/3) : pタイル(percentile)法

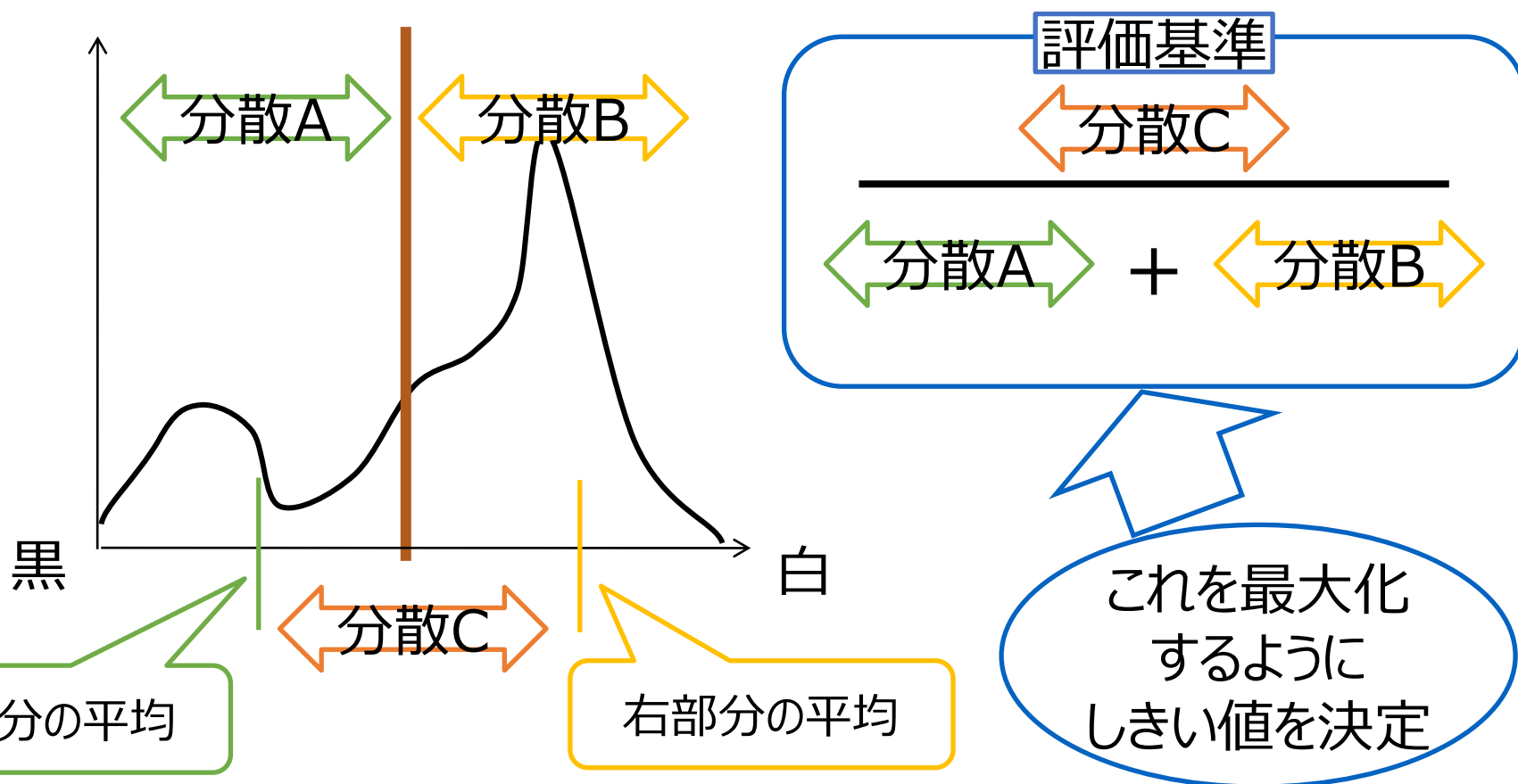
- 「全画素の p %が前景」が既知なら使える方法
- 全体的に明るさが変わってもOK!



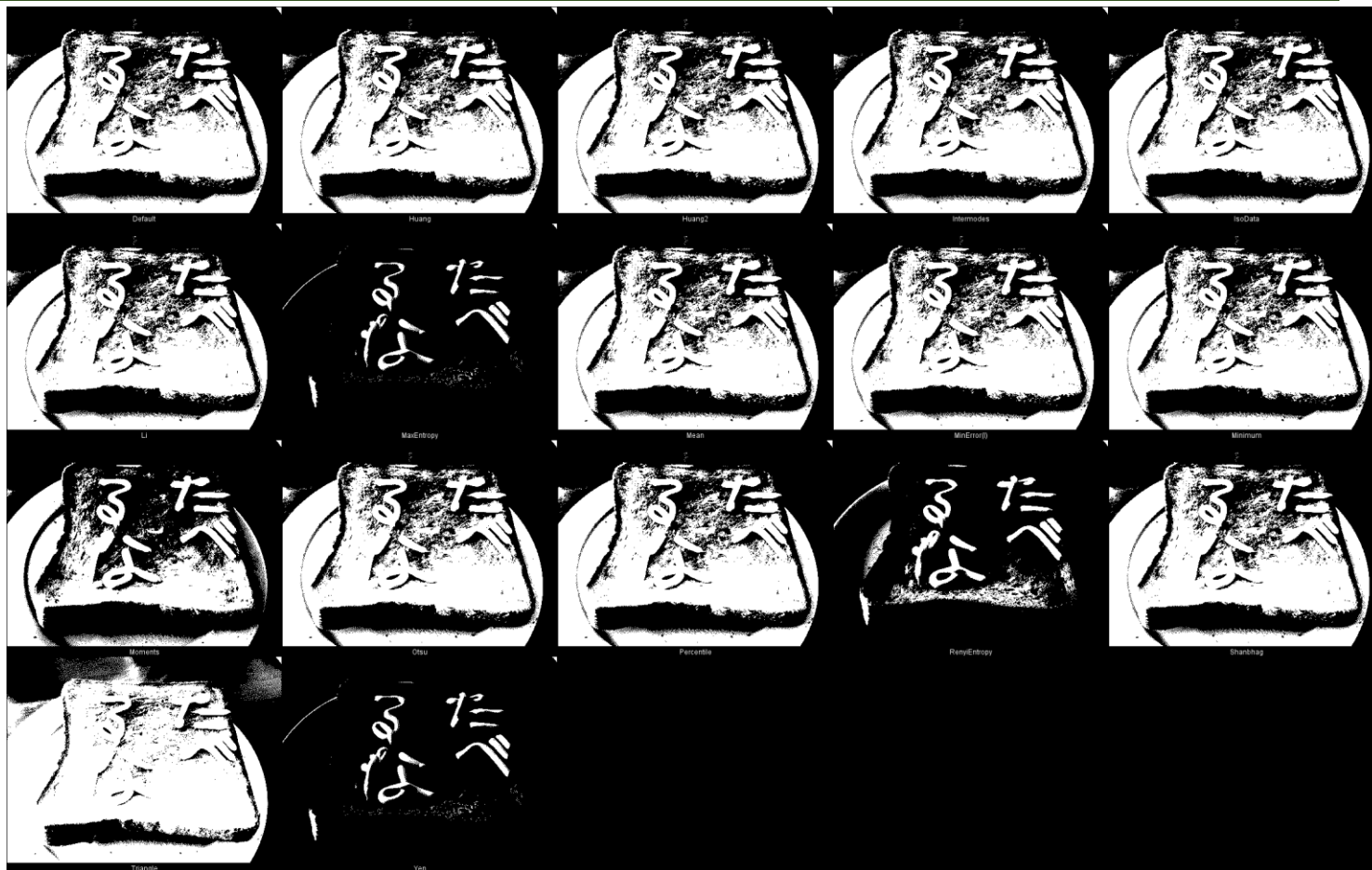
※前景 = 黒の場合

2値化のための「しきい値」決定法 (3/3) : 大津の二値化

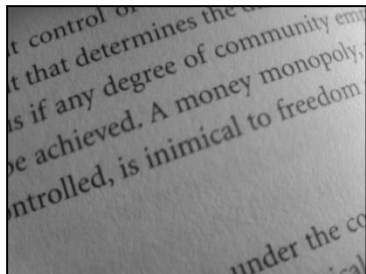
- 最適しきい値を決定



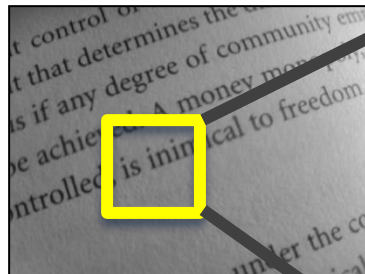
ImageJ/Fiji(画像処理の無料ソフト)を使えば 同時に色々試せる



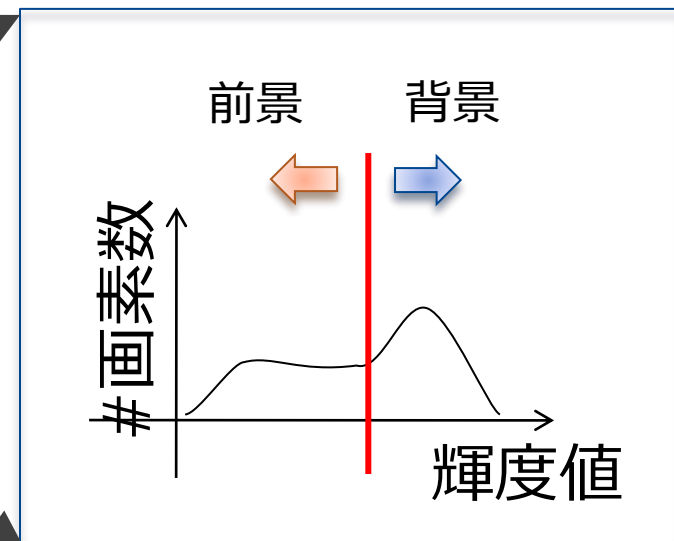
より進んだ二値化：局所二値化



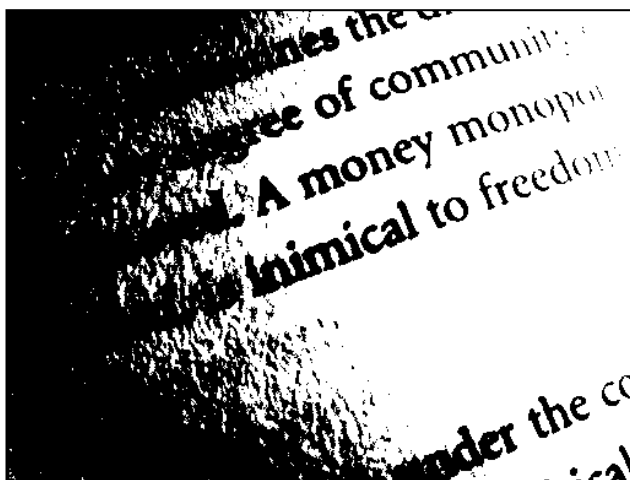
原画像



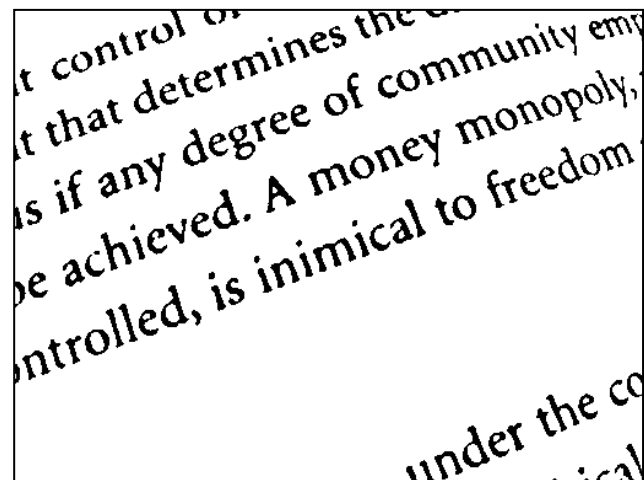
局所領域



全体で二値化



局所二値化



ImageJ/Fiji(画像処理の無料ソフト)を使えば 局所しきい値系でも同時に色々試せる



Bernsen



Contrast



Mean



Median



MidGray



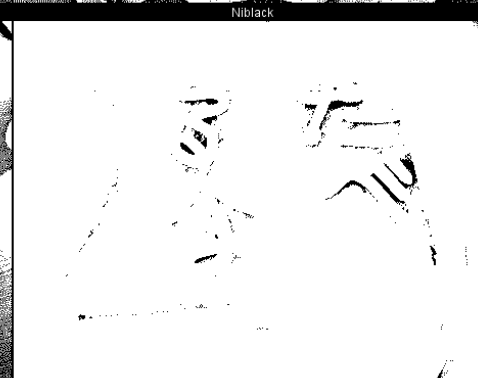
Niblack



Otsu



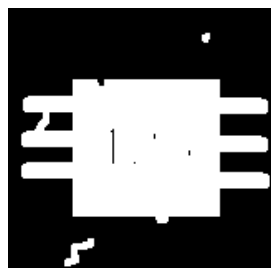
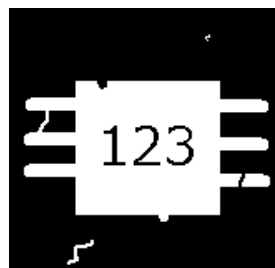
Phansalkar



Sauvola



2値化後によくやる処理： 膨張・収縮によるノイズ除去

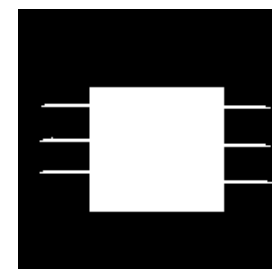
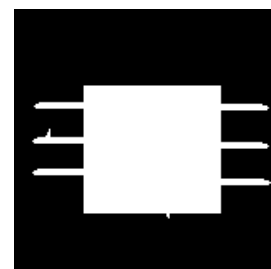
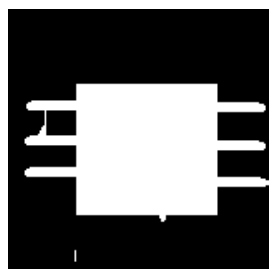
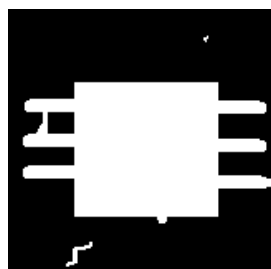
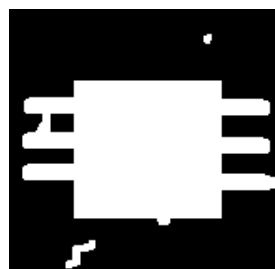


膨張

膨張

膨張

収縮



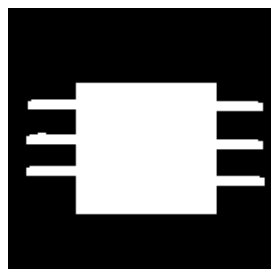
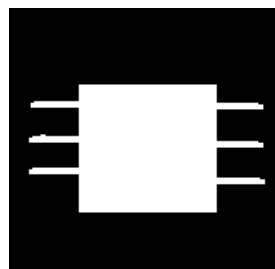
収縮

収縮

収縮

収縮

収縮



膨張

膨張

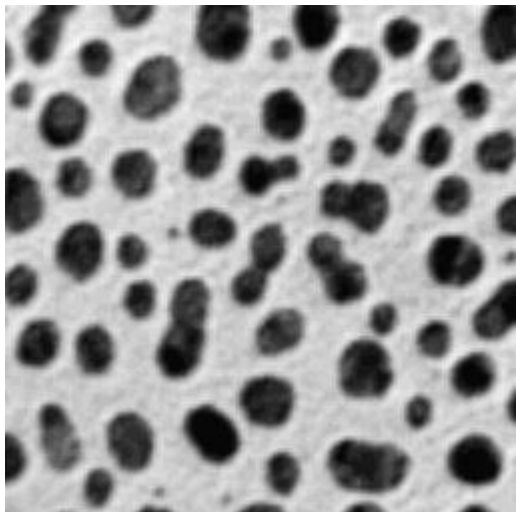


<http://imaging-solution.blog107.fc2.com/blog-entry-101.html>

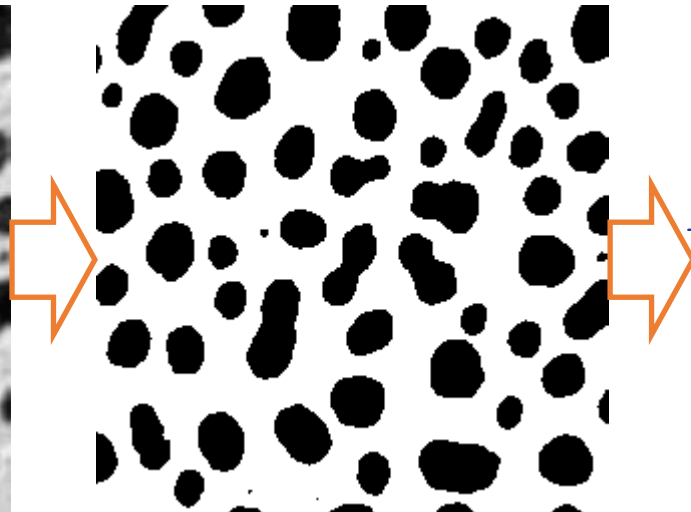
小さなギャップを埋められる
小さな孤立点を潰せる

連結成分解析: 粒状物体の定量化に便利

- 連結成分 = 黒(or 白)画素の塊
- 全体での数や, 各成分の面積や形が定量化できるようになる!

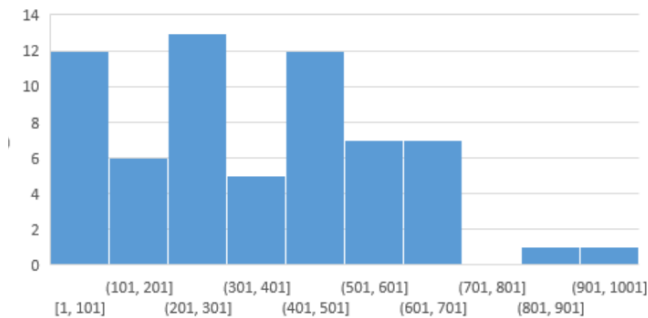


Fiji sample image "Blobs"



64個
総黒画素面積 22243
平均面積 347.547

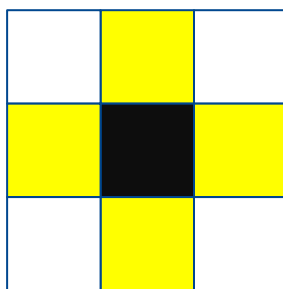
個数



面積

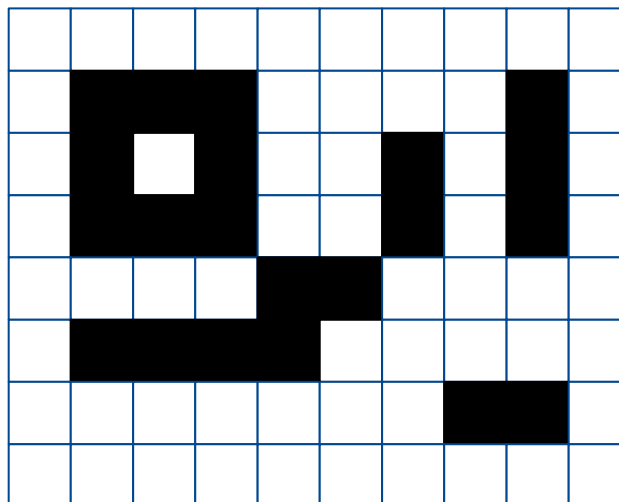
連結成分解析: 4連結と8連結～どこまでを「隣」とみるか？

- 4 連結での「隣」

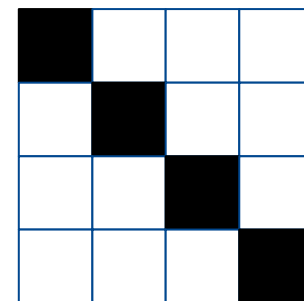
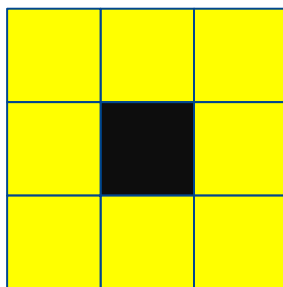


- 4連結か, 8連結かで変わる

(黒の) 連結成分数は何個？



- 8連結での「隣」



(白の) 連結成分数はいくつ？

【付録】 具体的な画像データ分析の例： 画像のフィルタリング

画像をぼかしたり，シャープにしたり，ノイズを除去したり

フィルタリング

何かしらの
計算結果

画素(x, y)

これを(x, y)を変えながら
全ての画素について実施

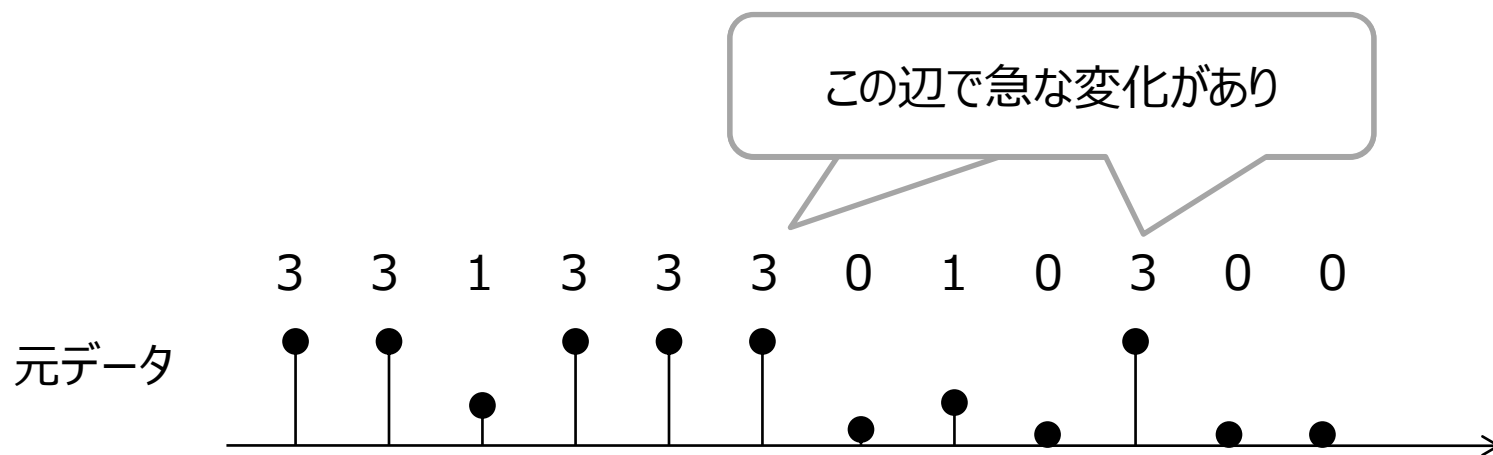
この
画素値に

原画像

フィルタリング結果

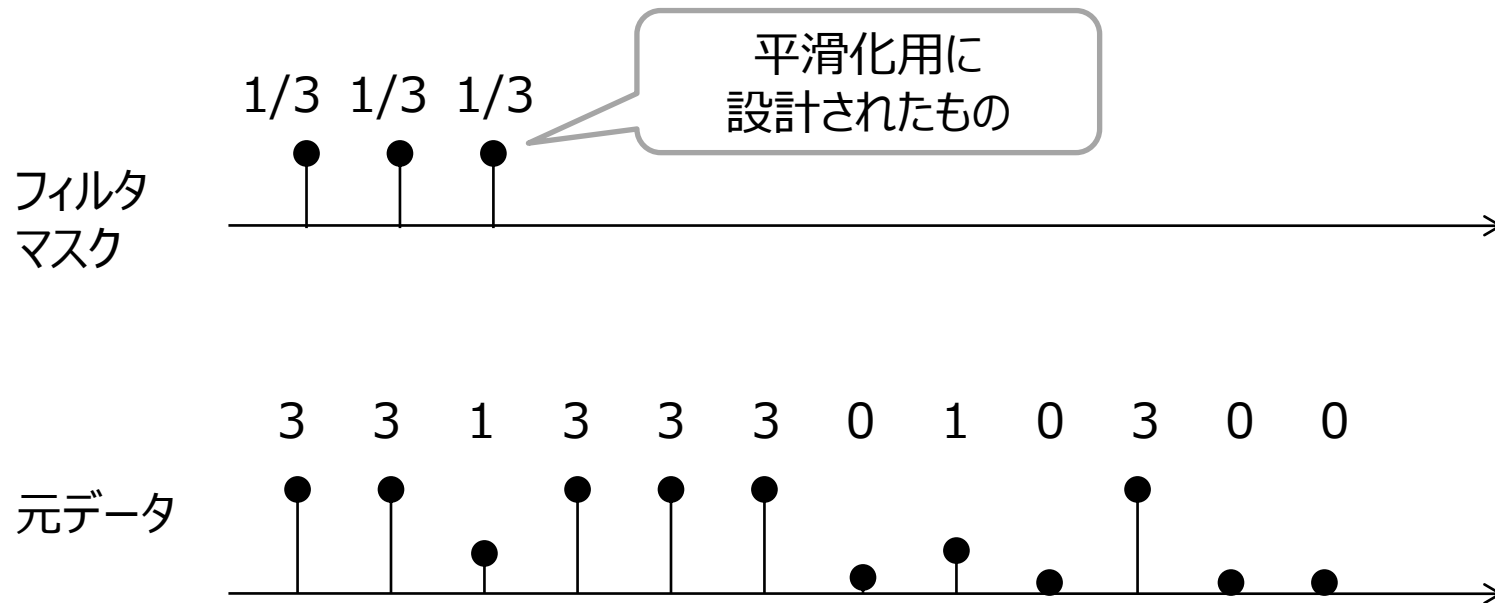
最も基本的なフィルタ：平滑化 (1/2)

まずは1次元で考えてみる



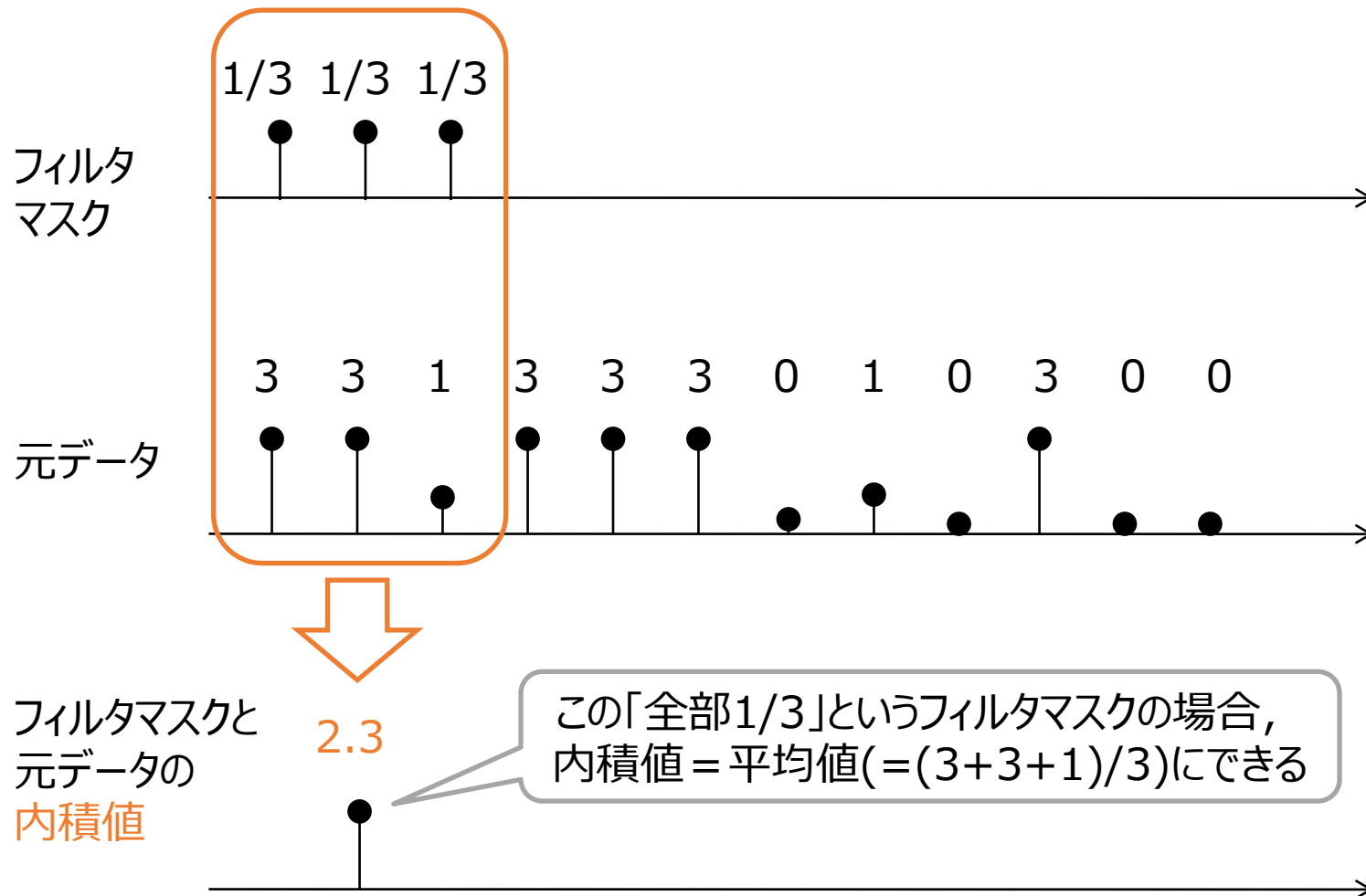
最も基本的なフィルタ：平滑化 (1/2)

まずは1次元で考えてみる



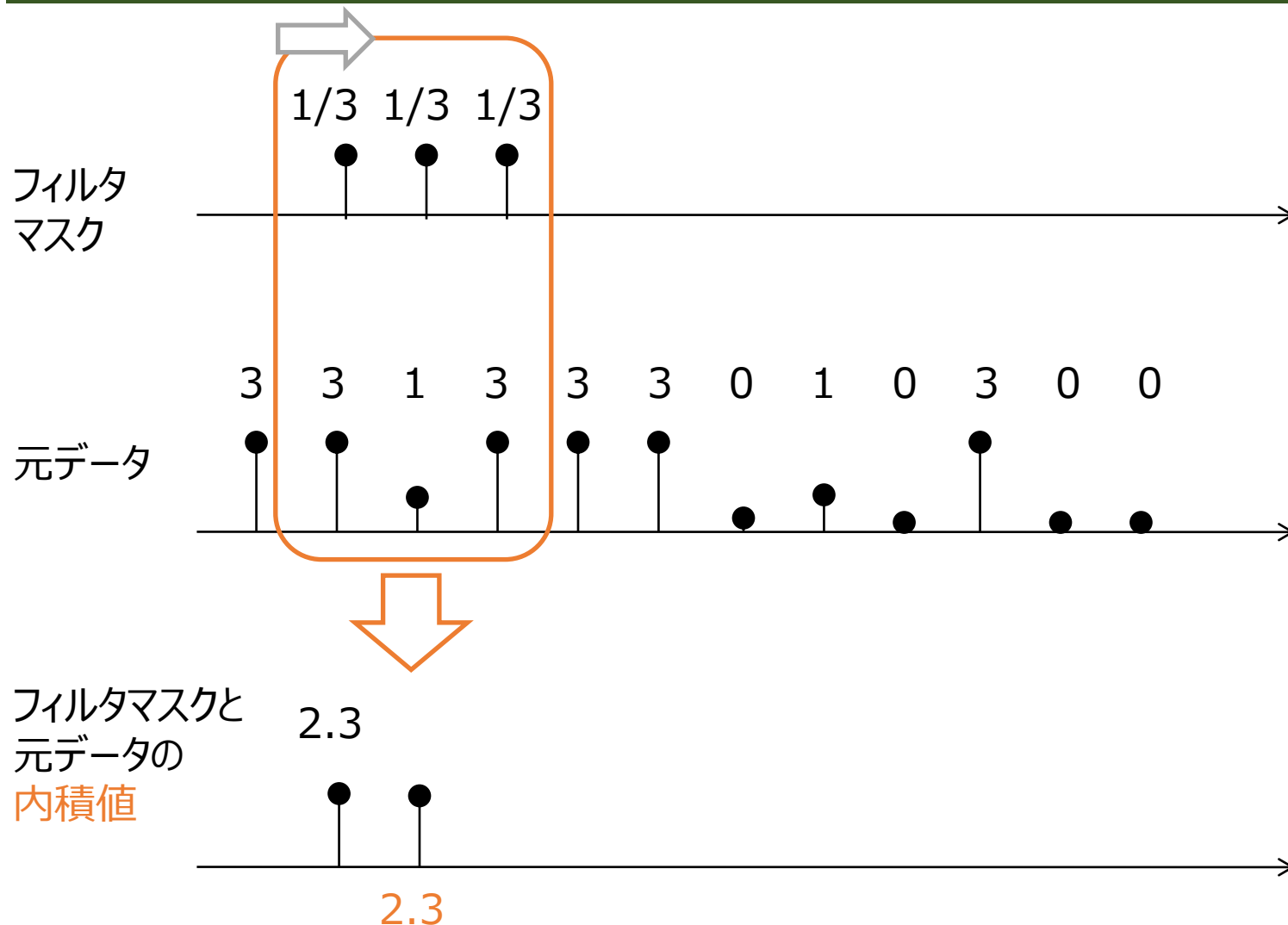
最も基本的なフィルタ：平滑化 (1/2)

まずは1次元で考えてみる



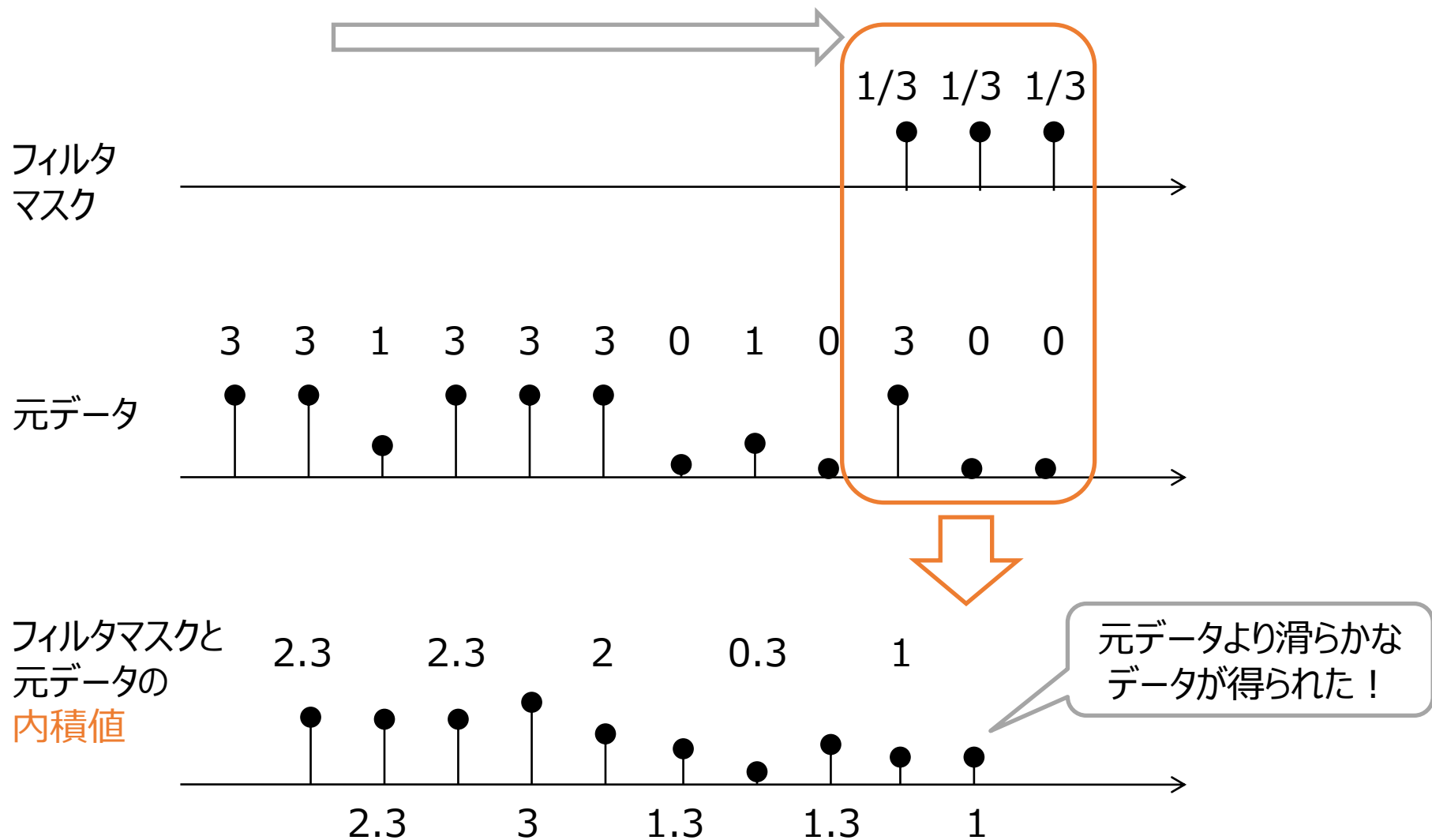
最も基本的なフィルタ：平滑化 (1/2)

まずは1次元で考えてみる



最も基本的なフィルタ：平滑化 (1/2)

まずは1次元で考えてみる



最も基本的なフィルタ：平滑化 (2/2)

画像の場合

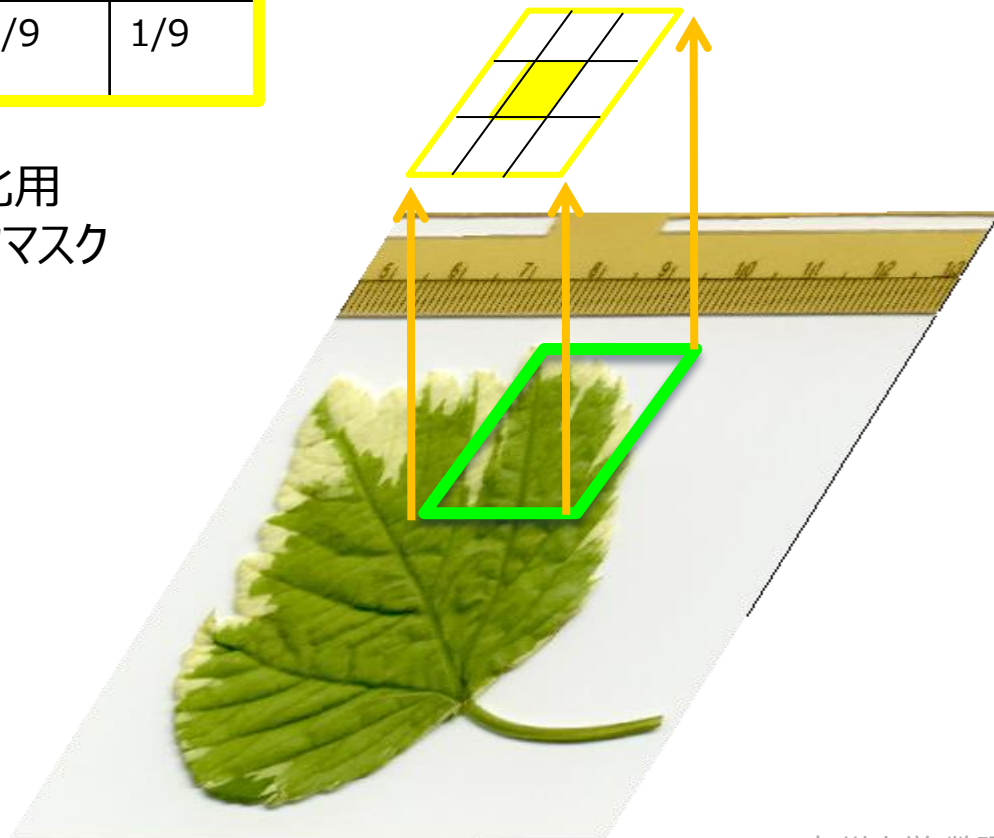


最も基本的なフィルタ：平滑化 (2/2)

画像の場合も同様に，ずらしながら内積

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

平滑化用
フィルタマスク
(3x3)



マスク

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

✕ 内積

元データ

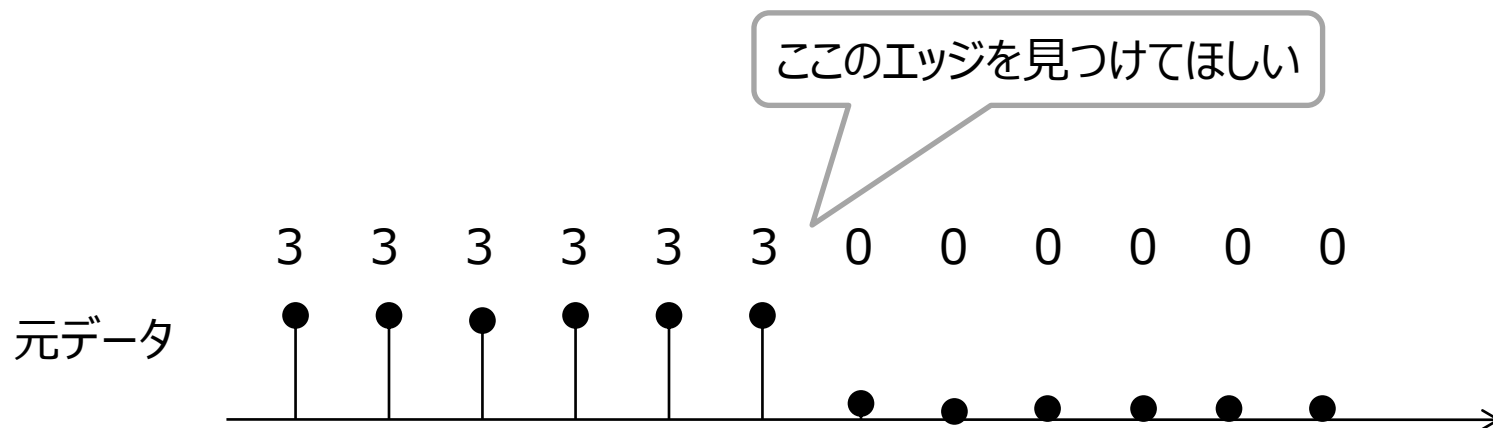
10	10	10
10	20	10
10	10	10

平滑化後

10	10	10
10	11	10
10	10	10

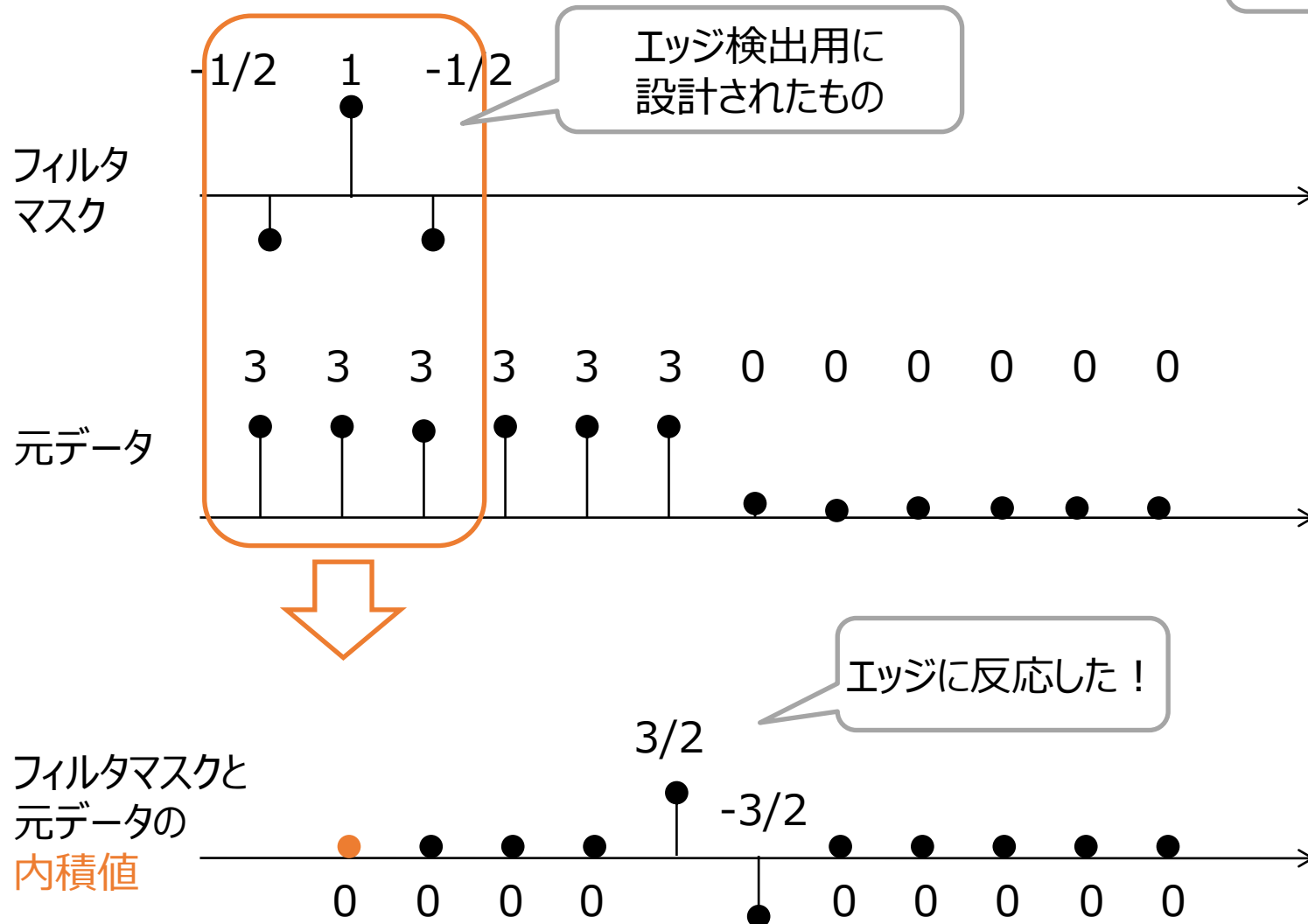
基本的なフィルタ：エッジ検出 まずは1次元で考えてみる

エッジ＝色や明るさが
急に変わる箇所



基本的なフィルタ：エッジ検出 まずは1次元で考えてみる

エッジ＝色や明るさが
急に変わる箇所

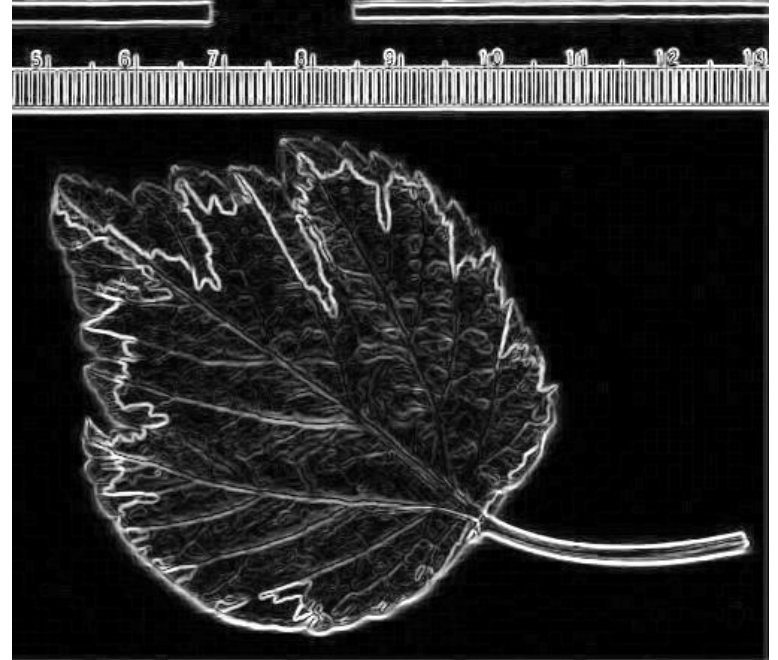


画像でのエッジ検出例

- 横方向と 縦方向で エッジを出して, それらを合成

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1



非線形フィルタ：メディアンフィルタ

- 各局所領域の「中央値」を採る処理
 - 中央値 = 濃淡値を大小順に並べ、その中で真ん中の値
 - 内積型ではないフィルタの代表例

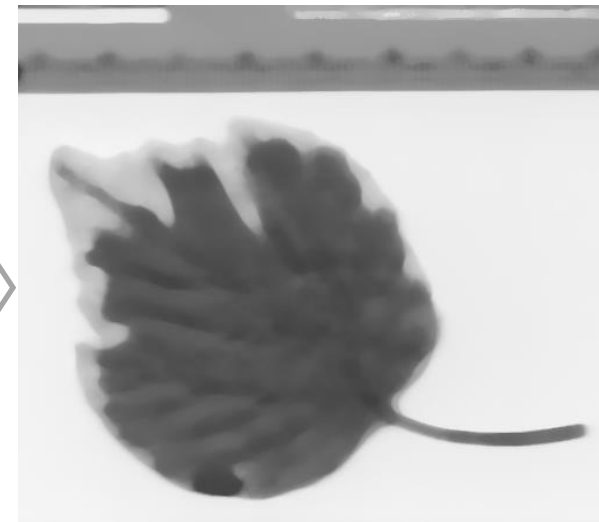
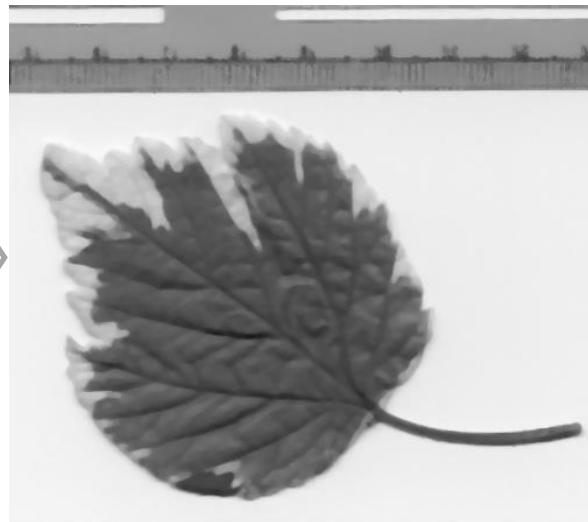
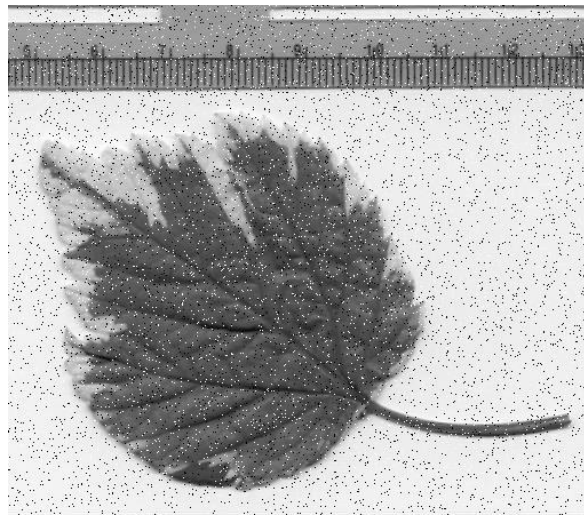
10	12	9
200	8	14
12	11	13

⇒

	12	

フィルタサイズ = 3×3

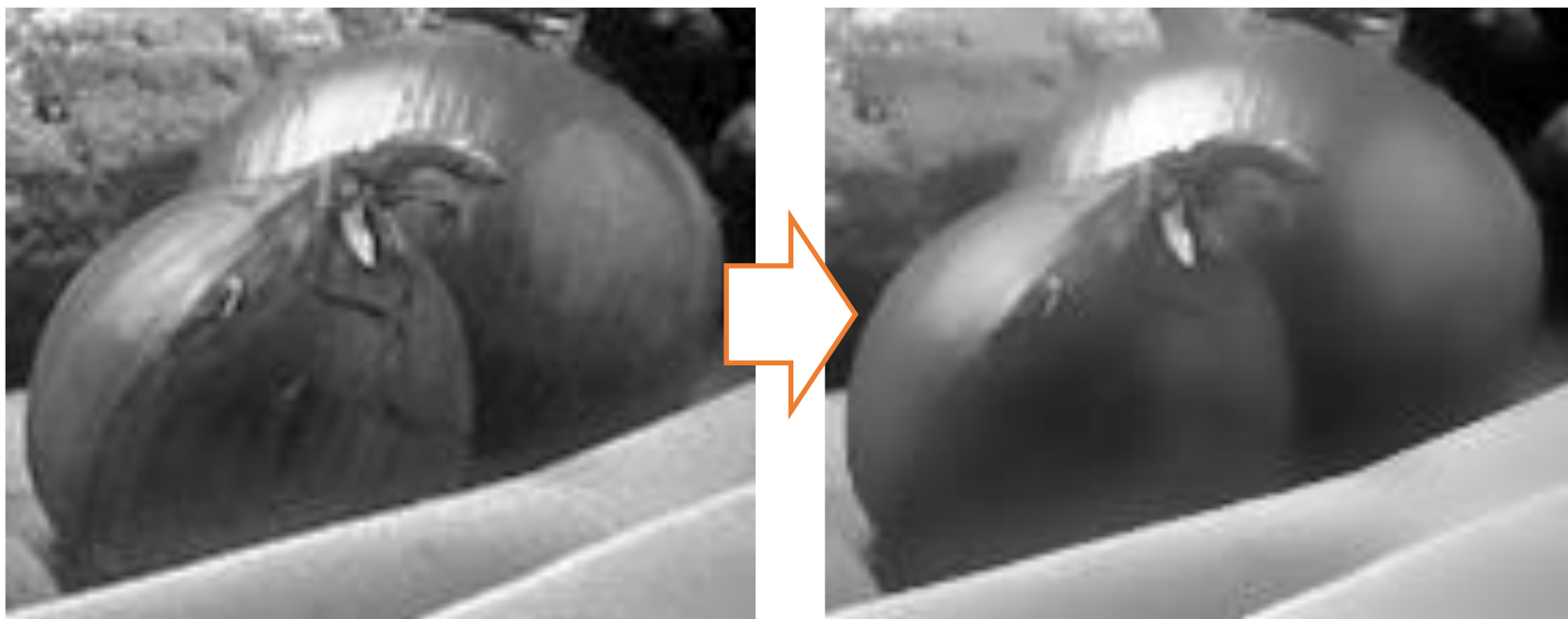
10×10



http://www.mvision.co.jp/WebHelpIM/_RESOURCE/Filter_Mvc_Median.html

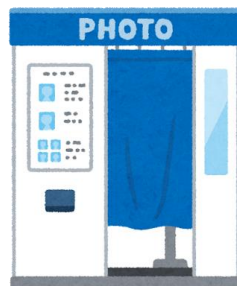
非線形フィルタ：バイラテラルフィルタ

- エッジは保存しながら細かい変化を平滑化



[Tomasi+, ICCV1998]

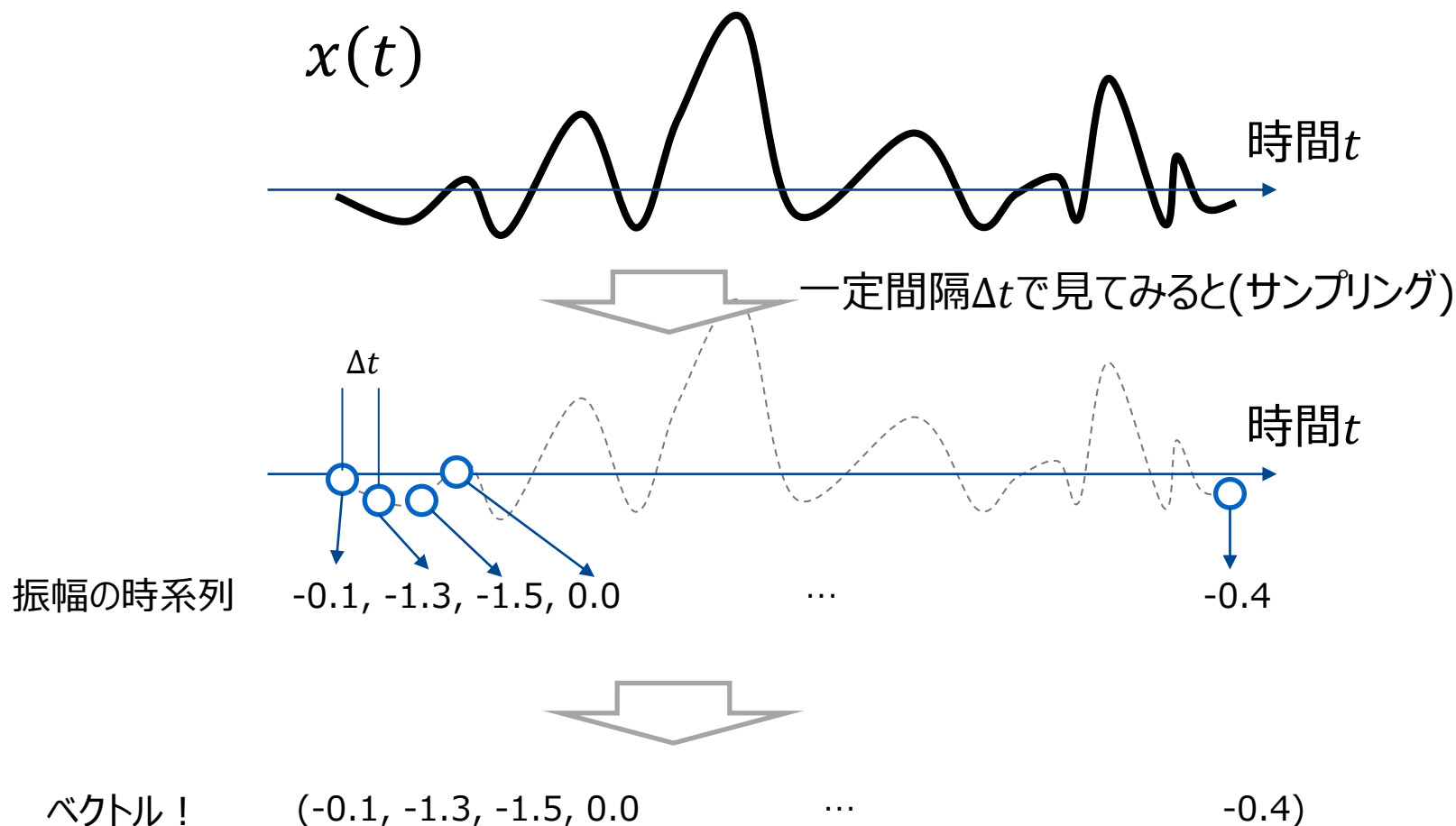
- 小じわが見えにくく？



【付録】 フーリエ変換

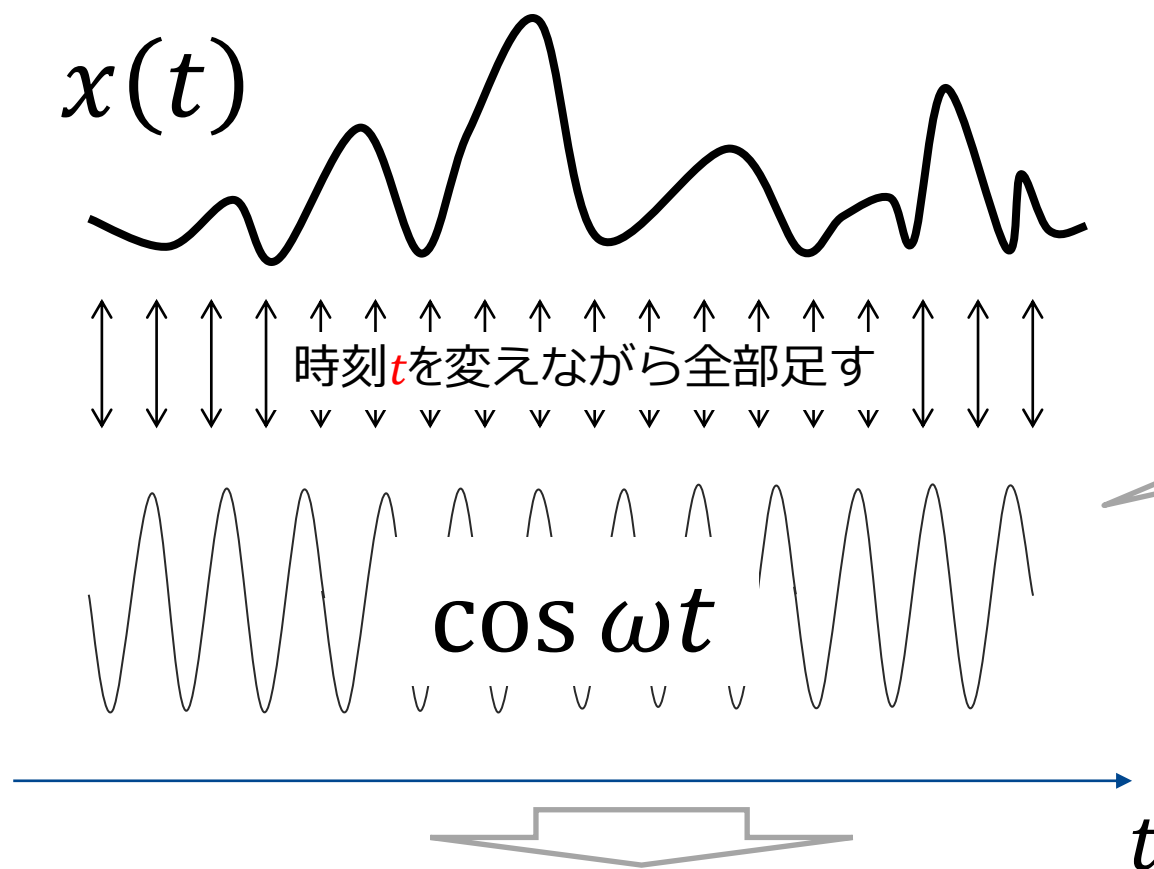
工学系ではよく使われます。
式は難しそうに見えますが、内積によるベクトルの分解と合成を
覚えていれば、似たような同じ話です。

波形を(無限次元)ベクトルとみる



Δt をどんどん小さくすると \circ の数はどんどん増える \rightarrow ベクトルは無限次元に

特定周波数成分の抽出 =ベクトルの分解と同じく「内積」



同じ時刻同士で
掛け合わせたものを
足し合わせる！

$$\int x(t) \cos \omega t \, dt$$

100とか1000とか
有限次元ならΣだが、
無限次元なので積分になる

$x(t)$ 中の $\cos \omega t$ 成分量 (ω で規定される周波数 $f (= 2\pi\omega)$ の成分量) が求まる！

フーリエ変換

$\int dt \leftrightarrow$ 時刻 t を変えながら全部足す

$$X(\omega) = \int x(t)(\cos \omega t - j \sin \omega t) dt$$

$$= \int x(t) \cos \omega t dt - j \int x(t) \sin \omega t dt$$



$x(t)$ と $\cos \omega t$ の内積



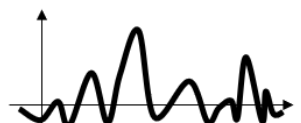
$x(t)$ と $\sin \omega t$ の内積

フーリエ変換により，入力信号中の 各周波数の成分量が求まっているわけです

材料セット

$e_1, \dots, e_i, \dots, e_d$

波形カレ- x



時間 t

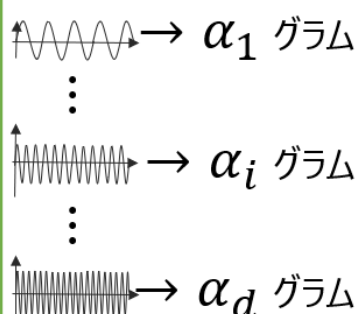
$$\alpha_i = x \cdot e_i$$

\sum



波形カレ-を完璧に
再現できた！

レシピ(分析結果)



$X(\omega_1)$

\vdots

$X(\omega_i)$

\vdots

$X(\omega_d)$

周波数
スペクトルと
呼ぶ

この内積部分が先ほどの

$$X(\omega) = \int x(t)(\cos \omega t - j \sin \omega t) dt$$

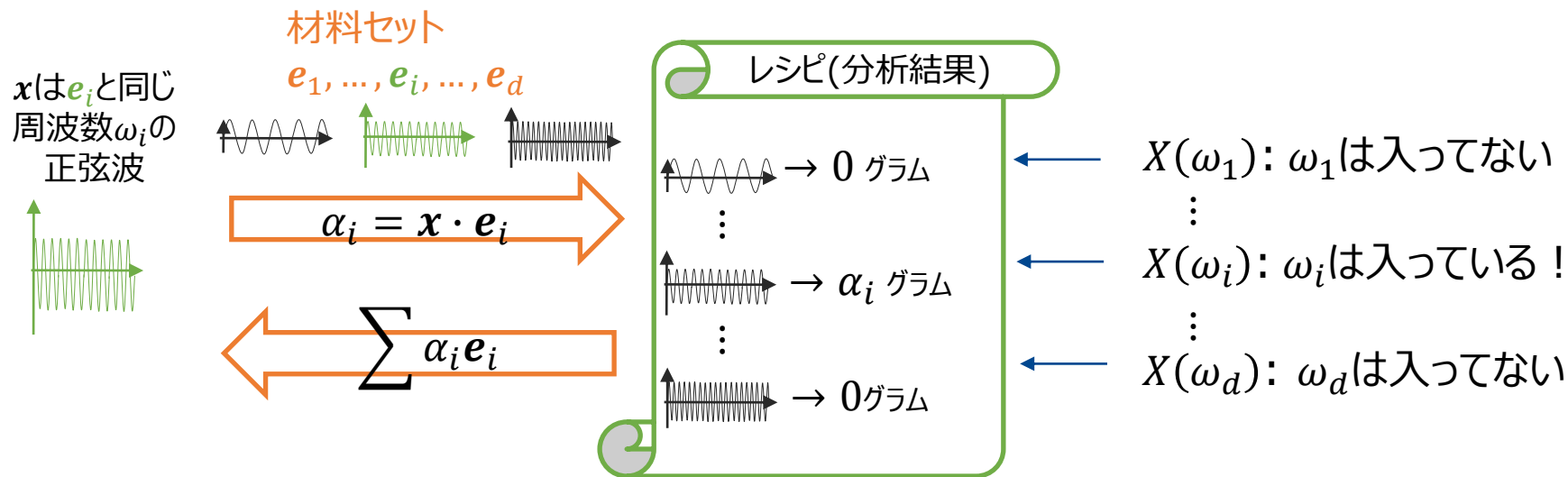
に相当

「基底 e_i はどれか」が気になる人は

$$X(\omega_i) = \int x(t) \underbrace{(\cos \omega_i t - j \sin \omega_i t)}_{e_i \text{に相当}} dt$$

とすればわかりやすい？

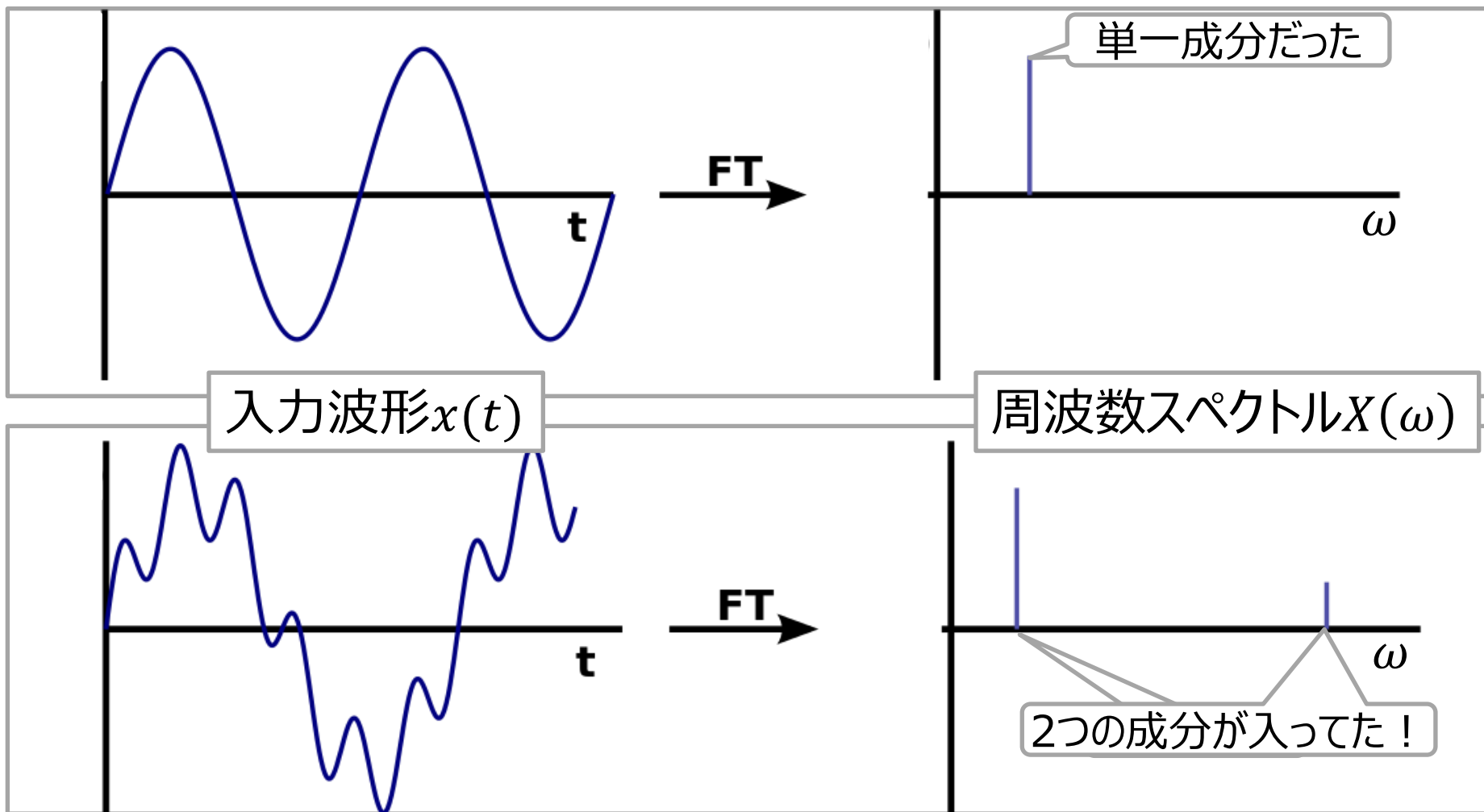
「むずかしいな」と感じたら，シンプルな例で (1/2)



水しか入っていないカレー(?)なら
水成分のみが非ゼロ(水以外はゼロ)



「むずかしいな」と感じたら，シンプルな例で (2/2)

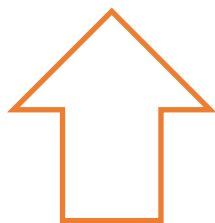


フーリエ逆変換

$\int d\omega \leftrightarrow$ 材料 ω を変えながら全部足す



$$x(t) = \int X(\omega)(\cos \omega t + j \sin \omega t) d\omega$$



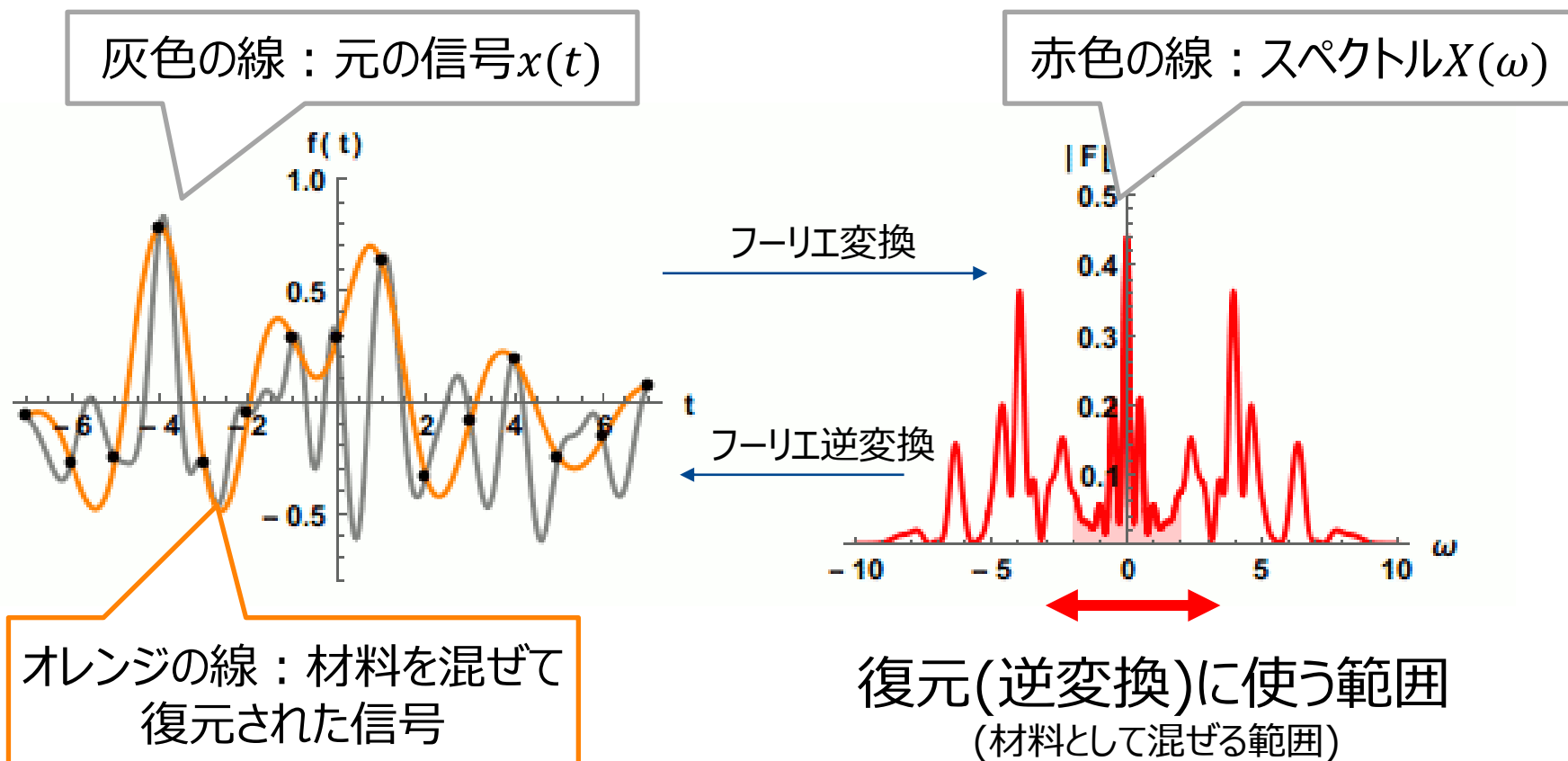
分量(レシピ)



各材料


動画でみるフーリエ変換・逆変換

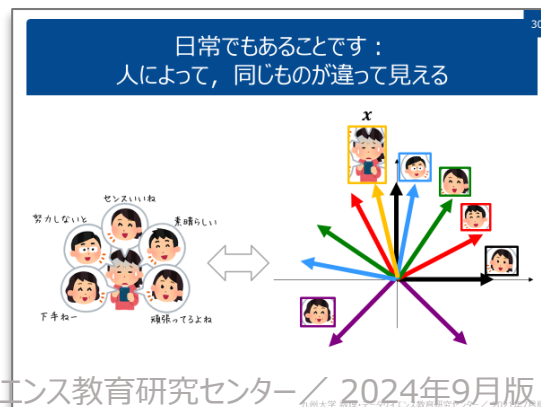
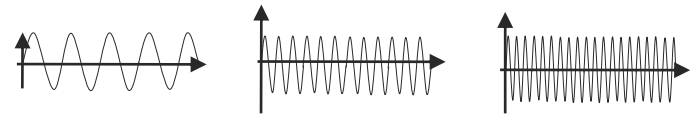
https://commons.wikimedia.org/wiki/File:Nyquist_sampling.gif



全部の材料をレシピ通り混ぜたら、元通り(灰色 = オレンジ色) になる点に注目！
さらに「最初は大雑把だが、徐々に詳細部も似てくる」点にも注目！

フーリエ基底

- \cos/\sin 関数に分解して戻る
 $= \cos/\sin$ 関数は（関数の）完備正規直交基底
- フーリエ基底とも呼ばれます
- （異常な関数を除けば）「**どんな関数でも \cos/\sin 成分に分解できて、合成すればもとに戻る**」というのは、（しつこいですが）美しい
 - 「異常とは何か」が気になる人は「2乗可積分関数」について調べてみよう
- 他にも色々な完備正規直交基底が存在
 - この話と同じ！ 
 - 同じ信号でも、基底を変えれば違って見える



【付録】 自然言語処理に関する余談

単語のデータ化 (1/2)

- 一番基本は「1-hotベクトル」表現

語彙数が N なら N 次元ベクトル

$$x_{\text{apple}} = (0, 0, 0, 1, 0, 0, 0, 0, 0, \dots, 0)$$

↑
apple

$$x_{\text{orange}} = (0, 0, 0, 0, \dots, 0, 0, 1, 0, \dots, 0)$$

↑
orange

- これだと「 x_{apple} と x_{orange} の違い（距離）」は
「 x_{apple} と x_{car} の違い（距離）」と同じ
- それでいいのか??

単語のデータ化 (1/2)

- これを最近は「単語埋め込み」により意味ベクトル化する

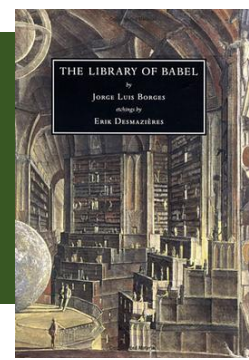
1-hotを意味に変換する関数 f

この f はたくさんの文章例と
機械学習で創り出す

$$X_{\text{apple}} = f(x_{\text{apple}})$$

- $X_{\text{apple}} \sim X_{\text{orange}}$ (似てるものは似た特徴ベクトルに！)
- 基本アイデア (分散仮説) = 文章の同じ位置に入りうる単語は似たベクトルに
- Ex. “I like [] for breakfast.” のように, appleやorangeを入れることができる文章が多い → appleとorangeは似た意味ベクトルに
- $X_{\text{king}} - X_{\text{man}} + X_{\text{woman}} = X_{\text{queen}}$ なんて計算もできる

超余談：バベルの図書館 (架空の図書館)



- 1冊410ページで構成される本を所蔵
 - さらにどの本も1ページに40行，1行に80文字→131.2万文字/冊
- すべての文字の並びの本が所蔵されている

131.2万文字

1冊目: AAAA AA
 2冊目: AAAA AB
 ⋮
 X冊目: E AP QE
 ⋮
 Y冊目: YOU ARE REALLY
 ⋮
 Z冊目: ZZZZ ZZ

“A”-“Z” 26文字種
 + 空白だと
 1.8×10^{1877949} 冊

ちなみに全宇宙の
 原子数は 10^{80}

- (ほとんどが無意味な文字列で書かれた本だが) よく探すと、**まだ**
読んだことない名作も、**昔だれかが書いた日記**も、すべて所蔵