

金子邦彦研究室

金子邦彦



答えを覚える → 答えを作る

答えを丸暗記…うーん。頑張る！



自分で答えを作ると…楽しい！



実行体験

自分のパソコンで実行体験。演習。ホームページで詳細説明。

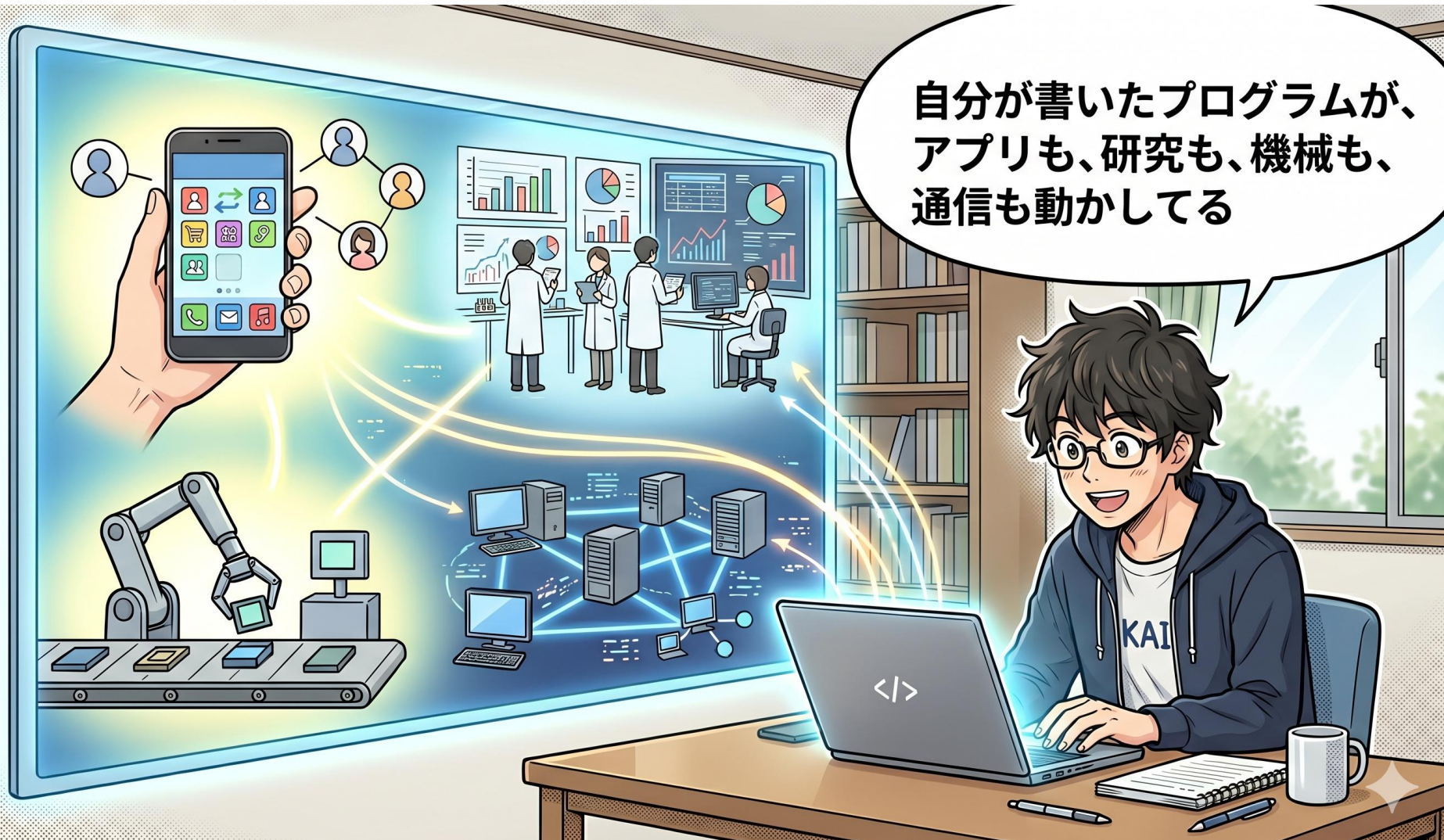
満足感

『動いた！』の瞬間を大切にしよう。

研究についての認識

研究は身近なパソコンと自分のアイデアから始めることができる。

Step 1: プログラミングは研究の 道具 —— 触って確かめよう

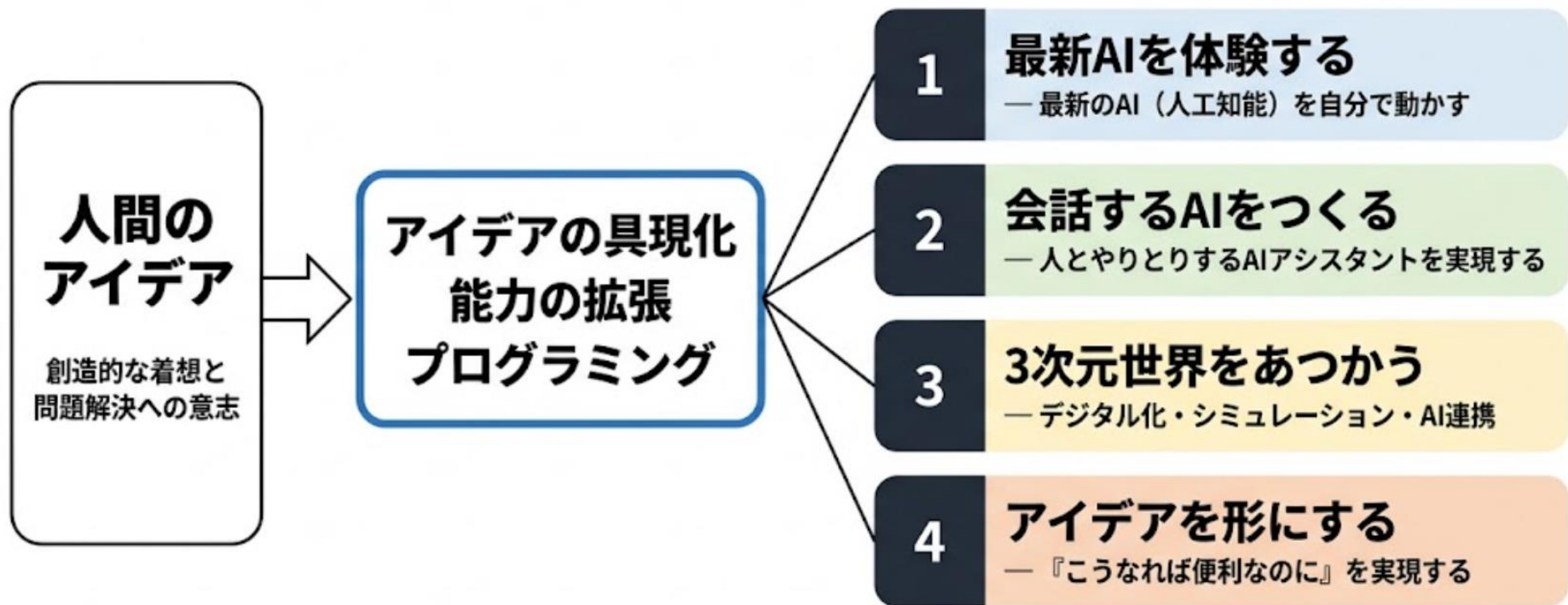


プログラムとは — 命令を書いた手順の集まり



だから複雑な作業も**自動化・効率化**できる

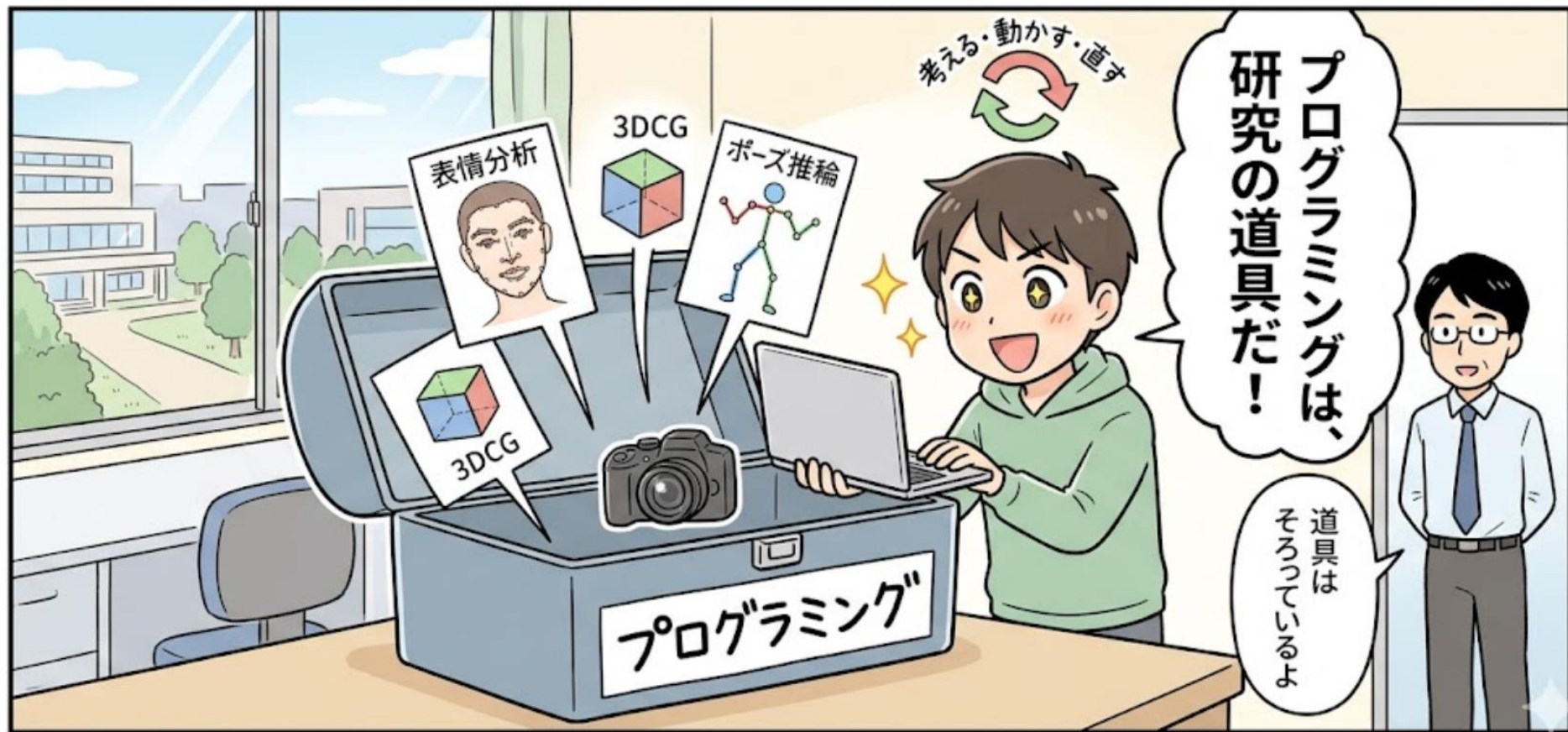
プログラミング観 - 人間の力を増幅する

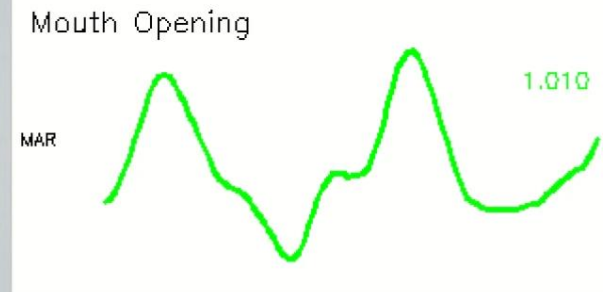
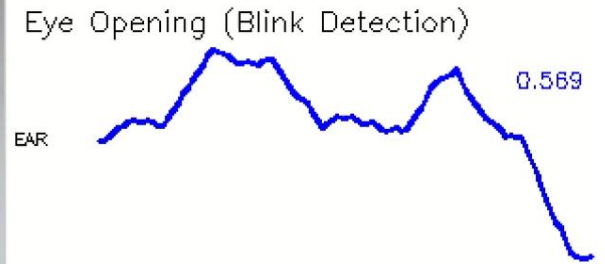


誤解：単なる『覚えて、問題を解くだけの勉強』

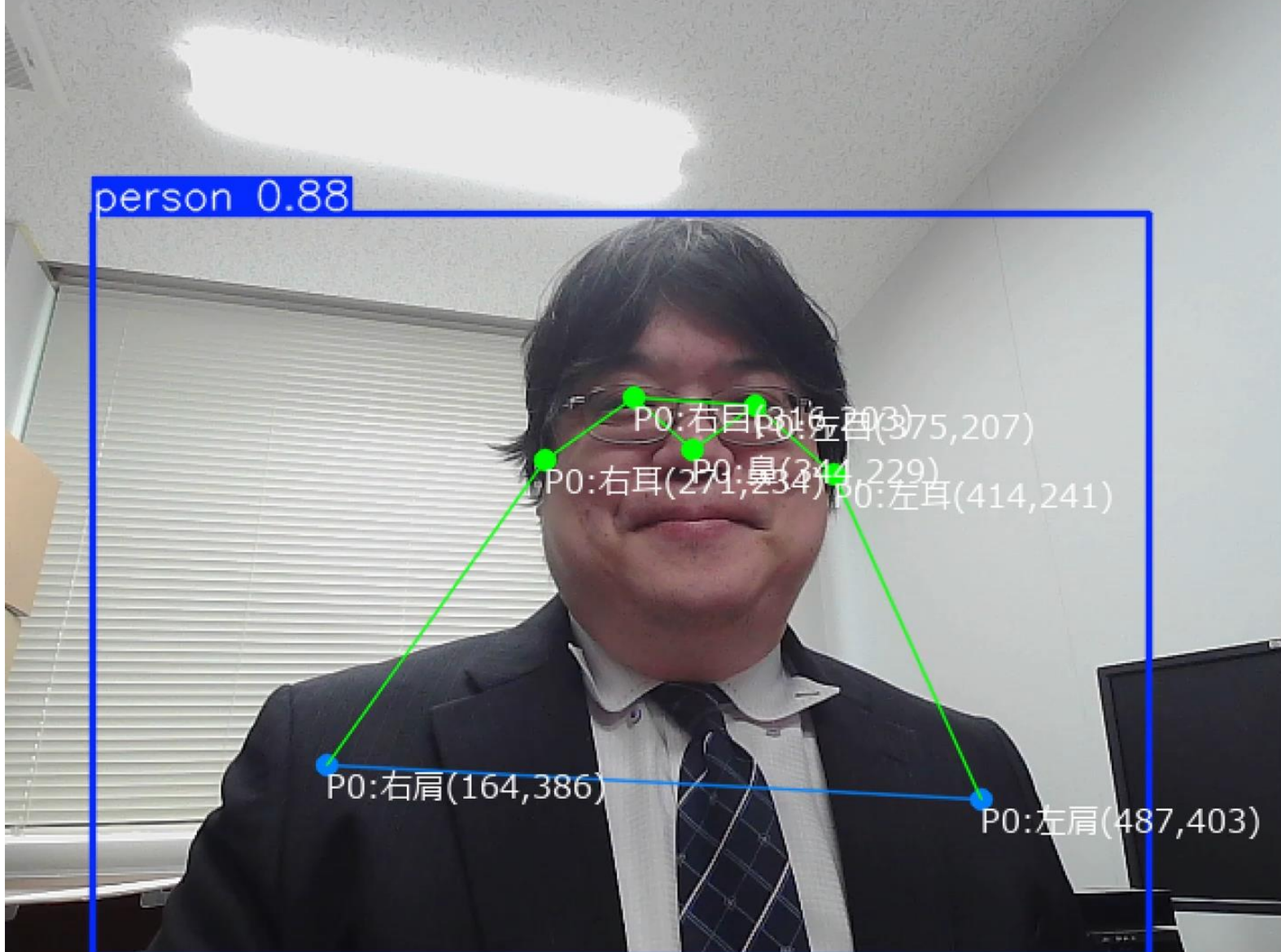
正解：『自分のアイデアを形にする』 創造的な作業

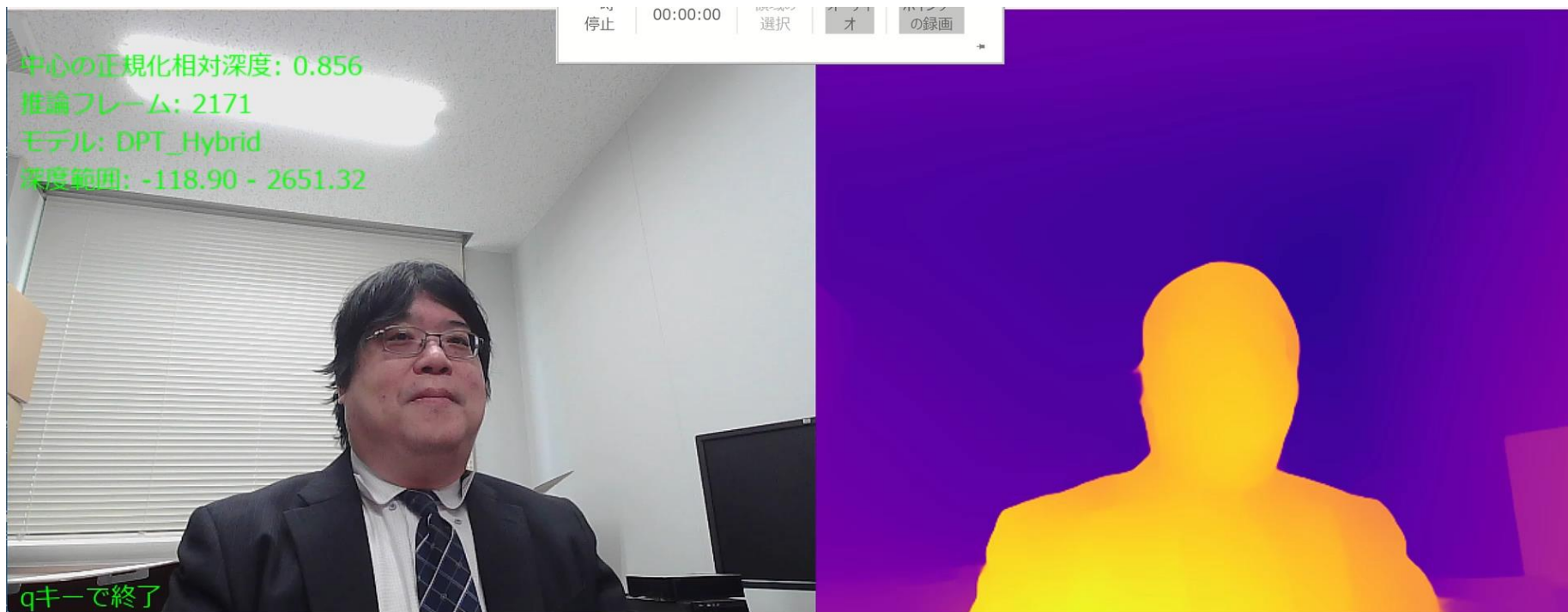
プログラミングは研究の道具





基本プログラムの例1：表情分析（目の開き、口の開き、目の動き）





基本プログラムの例 3 : 画像の 3 次元化、奥行方向の距離（深さ）推定

その他 100 以上の基本プログラムを研究室WEBページで公開

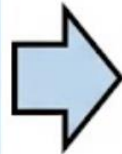
<https://www.kkaneko.jp/ai/labo/index.html>

<https://www.kkaneko.jp/cc/aitasks/mediapipe.html> その他

プログラミングが研究につながる

自分で工夫する

最新技術でも物足りないことがある。
自分で設定・調整したり、機能を追加したりして工夫する



改善を繰り返す

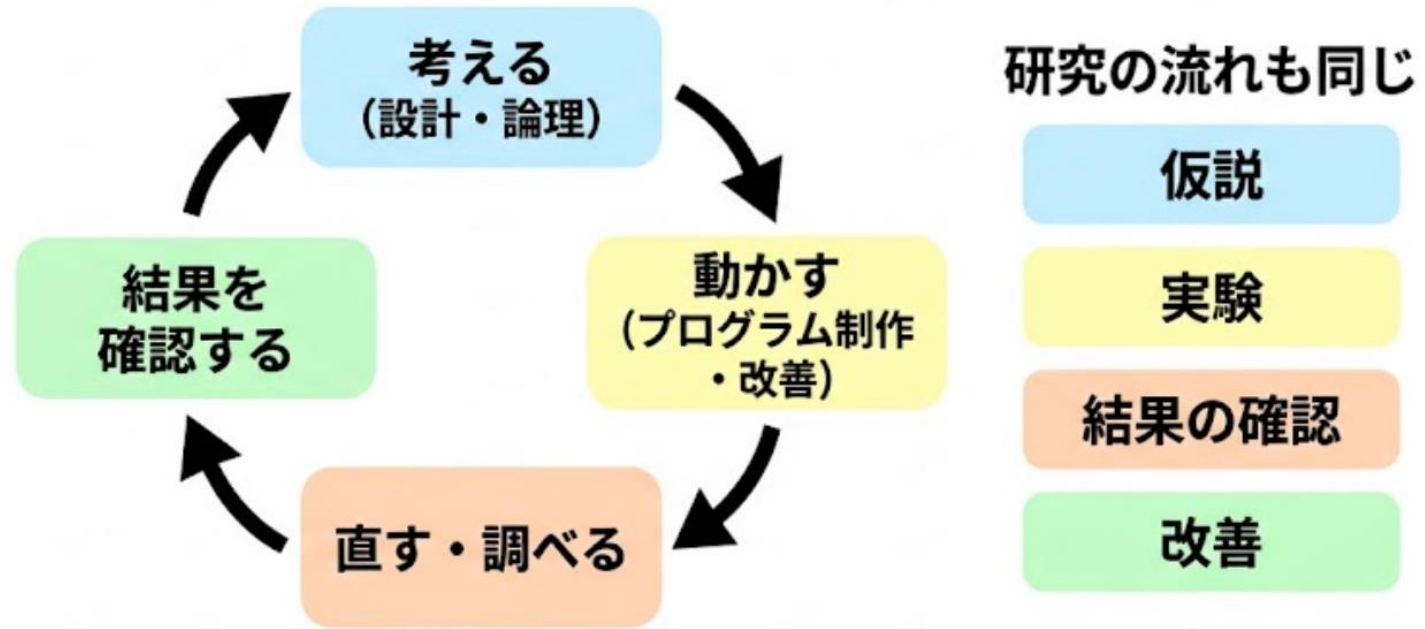
アイデアをプログラムとして実装し、実行結果を確認しながら改善を繰り返す。
これが学びや研究につながる



世界とつながる

インターネット上には最新研究のソースコードが公開・共有されている。公開コードの検証や改善も研究活動になり得る

研究は試行錯誤の繰り返し



プログラミングの試行錯誤＝研究のプロセス

公開コードの検証や改善も研究活動になる (出典明記などのマナーを守る)

最新AI、3次元CGプログラムが用意されている

金子研究室では、Windowsで動く100以上のAIプログラム例をソースコード付きで提供

既存技術を確認する

- 用意されたプログラムをそのまま動かす

設定を変えて実験する

- 閾値やモデルを変えて結果を比べる

アイデアを実現する

- 改造して新しい使い方を試す



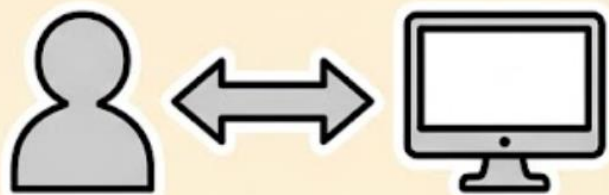
100以上のAIプログラム例

<https://www.kkaneko.jp/ai/lab/>

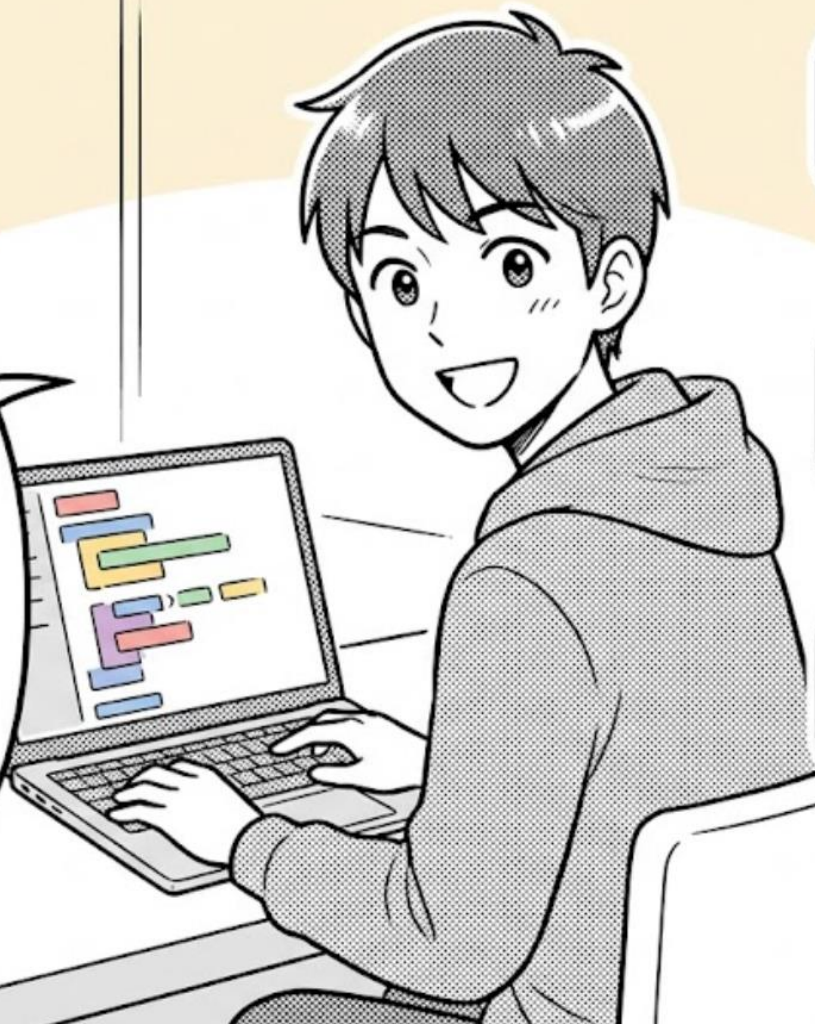


MediaPipeで15種類のAIタスク
(物体検出・姿勢推定・手や顔の認識・3D可視化など)

<https://www.kkaneko.jp/cc/aitasks/mediapipe.html>



ソースコードは
人とコンピュータの
“共通のことば”。
読めて、書けて、
あとから直せる。
だから扱いやすい!



書く



実行



結果



犬/猫画像認識AI

画像を入力すると、**AIが確率付きで認識結果を返す**

1枚目：犬 23%、猫 77%

2枚目：犬 70%、猫 30%

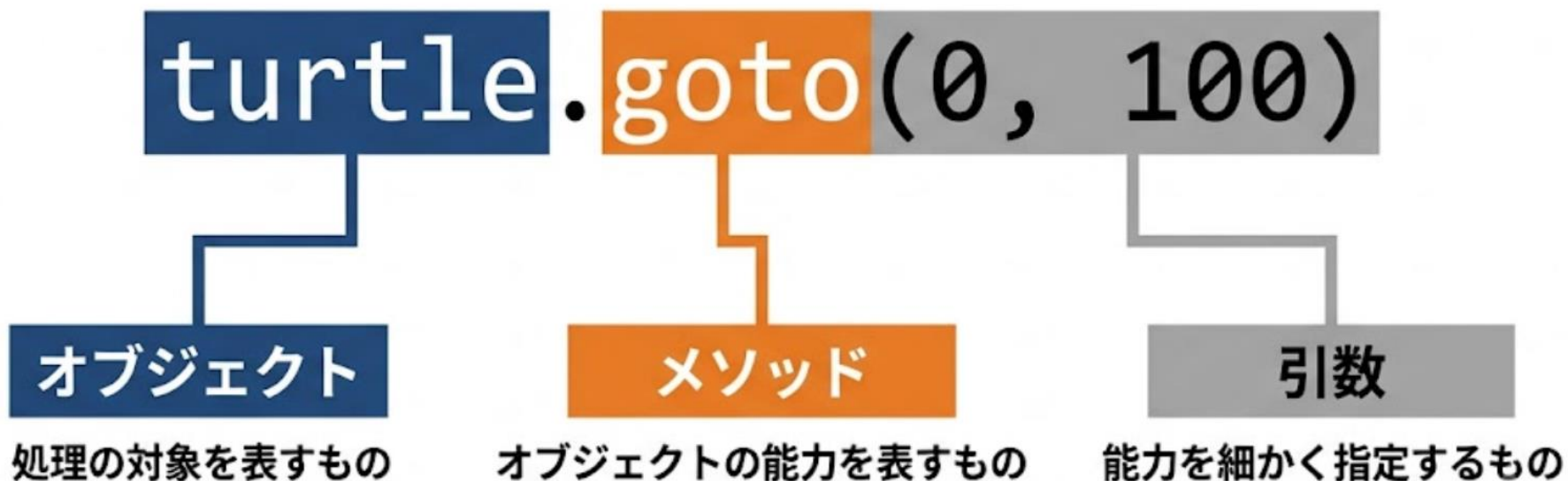


AIの特性：細かな差異を見逃さない（人間との違い）

16

オブジェクト・メソッド・引数

処理を表す三点セット



AI の学習と実行のプログラム

AI処理は6段階。どれも『三点セット』で書かれている

オブジェクト・メソッド・引数さえ分かれば、AIのコードも読める

1. 学習データの読み込みと前処理

```
images = preprocess_image(path)
```

2. モデルの構築

```
tf.keras.models.Sequential([...])
```

3. モデルのコンパイル

```
model.compile(optimizer='adam')
```

4. 学習

```
model.fit(images, labels,  
          epochs=5)
```

5. モデルの検証

```
model.evaluate(test_images,  
              test_labels)
```

6. タスクの実行

```
model.predict(test_image)
```

赤=オブジェクト / オレンジ=メソッド / () = 引数

複雑に見えても、オブジェクト・メソッド・引数の形



**turtle.goto(0,100) も model.fit(...) も、構造はまったく同じ。
だからAIのコードも読める**

AI の学習と実行のプログラム

ソースコード前半 — 赤=オブジェクト、オレンジ=メソッド

```
# 1. 画像の読み込みと前処理 赤=オブジェクト／オレンジ=メソッド
def preprocess_image(image_path):
    image = Image.open(image_path).resize((150, 150))
    image = np.array(image) / 255.0
    return image

# データセットの画像とラベルの設定
import os
os.chdir("c:/coco2017")
image_paths = ["dog/000000017029.jpg", "dog/000000022192.jpg", "dog/000000022877.jpg"]
labels = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
# 0: 犬, 1: 猫
images = [preprocess_image(path) for path in image_paths]
labels = np.array(labels)

# 2. モデルの構築
import tensorflow as tf
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])
```

AI の学習と実行のプログラム・ソースコード

②

ソースコード後半 — 赤=オブジェクト、オレンジ=メソッド

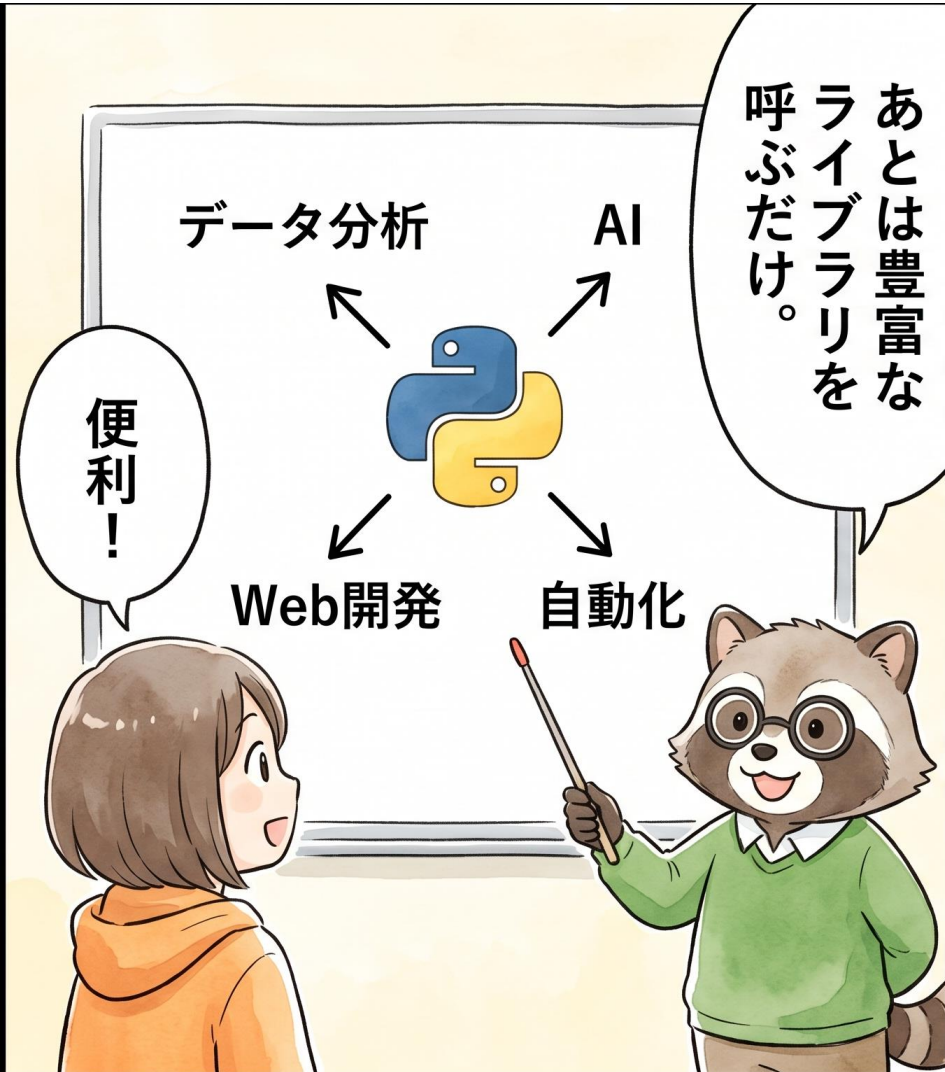
```
# 3. モデルのコンパイル
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 4. 学習
model.fit(np.array(images), labels, epochs=5)

# 5. モデルの評価
test_images = [preprocess_image(path) for path in ["dogcat/000000001675.jpg", "dogcat/000000007386.jpg"]]
test_labels = np.array([0, 1]) # テスト用のラベル
test_loss, test_accuracy = model.evaluate(np.array(test_images), test_labels)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)

# 6. タスクの実行 (学習済みモデルを使用して新たな画像に対して予測を行う)
import matplotlib.pyplot as plt
for test_image_path in ["dogcat/000000001675.jpg", "dogcat/000000007386.jpg"]:
    test_image = preprocess_image(test_image_path)
    test_image = np.expand_dims(test_image, axis=0) # 予測のために次元を追加
    predictions = model.predict(test_image)
    image = plt.imread(test_image_path)
    plt.imshow(image)
    plt.title(test_image_path)
    plt.show()
    print(test_image_path)
    print(predictions)
    predicted_label = np.argmax(predictions[0])
    if predicted_label == 0:
        print("分類結果:犬")
    else:
        print("分類結果:猫")
```

これだけ。AI学習のコードも数十行で書ける



Python の代表的なライブラリ



NumPy

数値計算・配列を扱う

1	2	3
4	5	6
7	8	9

配列

大量の数値をまとめて
高速に計算

外部ライブラリ

Matplotlib

データをグラフにする



`plt.plot(x, y, 'o')`
: 'o' は丸いマーカー

折れ線・散布図など
データを目で見える形に

外部ライブラリ

random

予測できない数値を作る



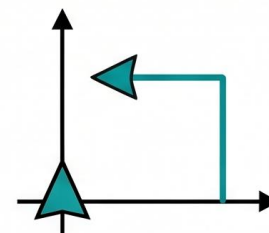
予測不可

シミュレーション/
ゲーム / 暗号化

標準ライブラリ

turtle

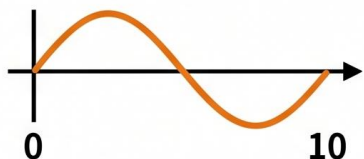
図形を描いて動かす



原点からスタート、
角度は度(°)で指定

標準ライブラリ

組み合わせるとさらに強力



`x = np.linspace(0, 10, 100)` : 0~10を100点に分割 / 点が多いほど滑らか
NumPyで数値を作り、Matplotlibでグラフにする — ライブラリは連携できる

import って何? —— 使える機能を増やす 1 行

標準機能

print()
if 文
for 文
リスト

Python をインストールした時点で、
import なしですぐ使える機能

import numpy

import matplotlib

この 1 行で、新しい機能が
使えるようになる

import turtle

ライブラリ = 特定の機能をまとめた
もの。import で読み込んで使う

NumPy

数値計算・配列を扱う

外部ライブラリ (自分で追加してから import で読み込む)

Matplotlib

データをグラフにする

外部ライブラリ (自分で追加してから import で読み込む)

turtle

図形を描いて動かす

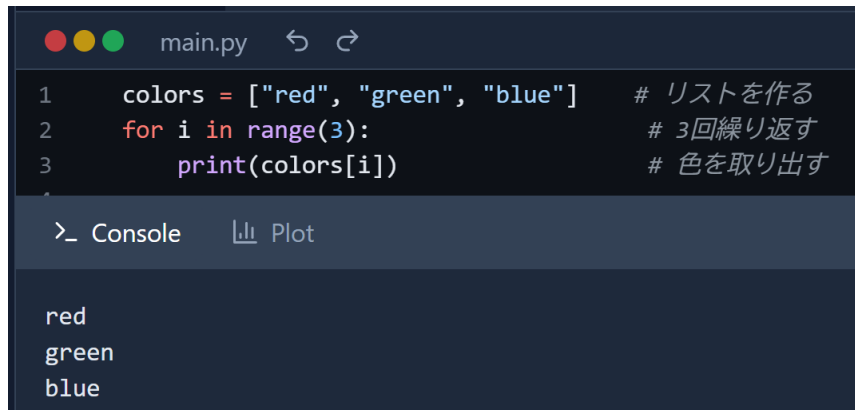
標準ライブラリ (最初から入っているが、import で読み込む)

標準機能だけでは足りない処理は、ライブラリを import で読み込んで補う

forループとリスト

複数のデータを一度に扱いたい場合

```
colors = ["red", "green", "blue"] # リストを作る
for i in range(3):                # 3回繰り返す
    print(colors[i])              # 色を取り出す
```



```
main.py ↶ ↷
1 colors = ["red", "green", "blue"] # リストを作る
2 for i in range(3):                # 3回繰り返す
3     print(colors[i])              # 色を取り出す
.
>_ Console | Plot
red
green
blue
```

- 少しずつ慣れる。感覚をつかむ。何ができるか知っておく
- **colors[i]** の使い方は、**演習②で生きる**

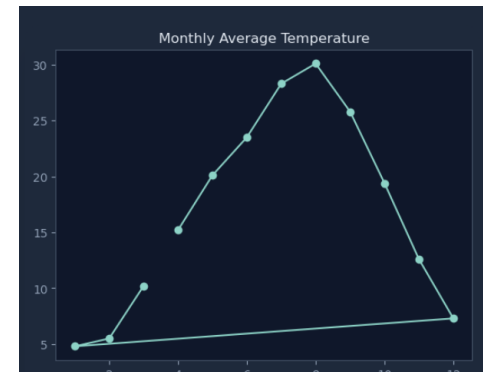
演習① グラフ描画

概要

- matplotlibの必要部分をインポート、**気温データを折れ線グラフで描画する。数値を変えるとグラフの形が変わることを確認する。**

書き方

- **import** でモジュールをインポート
- **plt.plot(...)** でデータをグラフにする
- **plt.show()** で結果を表示する



演習① グラフ描画

1. <https://pythoncompiler.io/> にアクセス
2. 下のコードを入れ、実行ボタンを押し、**グラフが表示されることを確認**する
3. `temperature` リストの数値を変えて再実行し、**グラフの変化を観察**する
4. うまく動いたら、隣の人とグラフを見せ合ってみよう

「変えて再実行」 —— これが小さな研究

```
import matplotlib.pyplot as plt
months = [4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3]
temperature = [15.2, 20.1, 23.5, 28.3, 30.1, 25.8, 19.4, 12.6, 7.3, 4.8, 5.5, 10.2]
plt.plot(months, temperature, marker='o')
plt.title('Monthly Average Temperature')
plt.show()
```

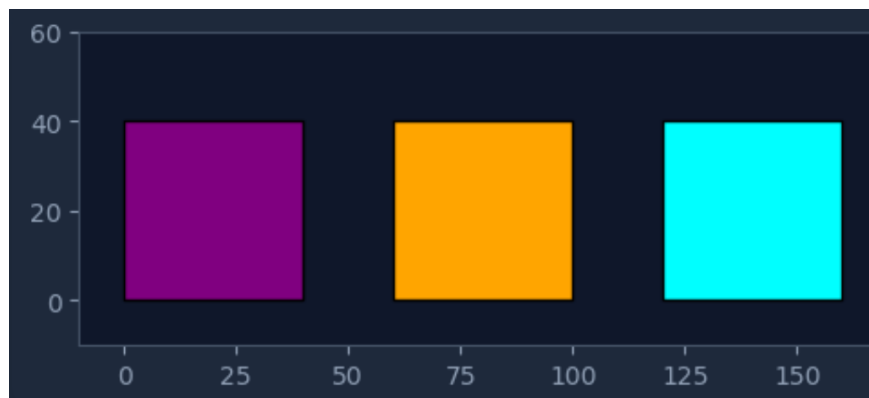
演習② 図形描画

概要

リスト, forループ, 色指定を組み合わせることで, 色の異なる複数の正方形を並べて描画する.

学ぶこと

- **for i in range(3):** による繰り返し処理
- **colors[i]** によるリストからの値の取り出し



演習② 図形描画

1. <https://pythoncompiler.io/> にアクセス
2. 下のコードを入れ、実行ボタンを押し、3色の正方形が並んで表示されることを確認する
3. 60, 0, 40, 40（8行目）の数値を変えてみて、変化を見る
4. 4つ目の色と正方形を追加してみる
5. うまく動いたら、隣の人と結果を見せ合ってみよう

「変えて再実行」 —— これが小さな研究

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches

fig, ax = plt.subplots()
colors = ["purple", "orange", "cyan"] # 色のリスト
for i in range(3): # 3回繰り返す
    ax.add_patch(patches.Rectangle(
        (i * 60, 0), 40, 40, # 位置(x,y), 幅, 高さ
        facecolor=colors[i], # 塗りつぶし色
        edgecolor="black" # 枠線の色
    ))

ax.set_xlim(-10, 190)
ax.set_ylim(-10, 60)
ax.set_aspect("equal") # 縦横比を揃える
plt.show()
```



Step 2: 君の学びを加速させる！

教員から専門知識を、仲間から
刺激を、AIからヒントを

研究室のゼミ

研究室のゼミ —— 仲間と作戦を練る場

- 「今週はこれを試した」「ここで詰まっている」を、**普段の会話のように気軽に共有する**
- 他の人の話から「それは面白そうだ」と、**新しいヒントを得られる**

一人の探究



批判的思考&実践



一人の探究
+
仲間との対話

仲間との対話



対話から発見

AIアシスタントとプログラミング

AIアシスタントの例 — Colabは登録無料

やりたいことを言葉で伝えれば、AIがコードを書くのを手伝ってくれる

コードを書く場所

```
▶ import plotly.graph_objects as go
import numpy as np

np.random.seed(42)
n = 100
x = np.random.randn(n)
y = np.random.randn(n)
z = np.random.randn(n)

fig = go.Figure(data=[go.Scatter3d(x=x,
                                   y=y, z=z, mode='markers')])
fig.show()
```

↔
相談しながら
コードを完成

AIアシスタント (Gemini)

こんにちは。ご用件をお聞かせください

Pythonライブラリのインストール方法

Googleドライブからデータを読み込む

シンプルなMLモデルの例

プログラミングが苦手でも安心。コードは自力で解くものではなく、自分の道具。
『動いた』を一つずつ積み重ねよう

AIアシスタントと研究

研究でAIにできること



研究やプログラミングの
家庭教師

「このコードって何してるの？」



アイデアを広げる
ブレスト相手

「グラフをもっと見やすくしたい！」



困ったときの相談相手

「matplotlibで他にできることは？」



研究アシスタント

「研究テーマの提案・具体化・
ツール調査・アイデア検証」

Google ColabやCopilotなどのAIが、研究のあらゆる場面で相棒になる



自分で試す × AIに相談する × チームで対話する — 研究も仕事も、この3つで前に進む

この進め方は、将来の仕事でこそ役立つ

演習③ AI支援プログラミング (Google Colab)

1. **Google Colab**で**新規ノートブック**を作成し, **AIアシスタント**へ依頼し**物体検出プログラム**を生成・実行。→ 動作確認後は、**ミニ研究・探求**に展開
2. **Googleアカウント**で**ログイン**必要
3. **ログイン画面表示が乱れる場合はシークレットモード**でColabを開き直す
4. AIは完璧ではない (余計な機能追加、最新情報欠落、自分のミスに気付かない)。**エラー発生時はAIに修正依頼を繰り返す**。必要に応じ**具体的ヒント**を人間から与える





[] ▶️ コーディングを開始するか、AI で生成します。



Gemini



こんにちは、邦彦さん

ご用件をお聞かせください。

Python ライブラリをインストールする方法

Google ドライブからデータを読み込む

シンプルな ML モデルのトレーニングの例

何を構築しますか？

+ 🗨️

Gemini 2.5 Flash ▶️



Step 3: 学びの展望



先輩たちの卒業研究

発想 ⇒ 行動 ⇒ 確認

新発見や新アイデアを探求しながら研究を推進



まずは試してみることから開始している

まずは『試してみよう』——教材はもう揃っている

研究に必要なのは高度な技術ではなく、試してみる姿勢。先輩たちも同じ場所から始めた

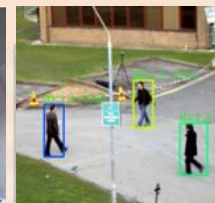
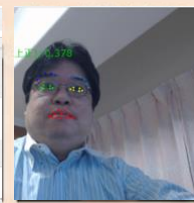
- 基本プログラムと説明は、
教員とAIから提供される
- そこから自分の工夫で発展させる
(AIに指示してもよい。主人公は人間)
- 先輩たちも同じところからスタートしている

研究に必要なのは

『試してみよう』の姿勢

100以上のAIアプリが無料で使える

金子研究室HP内のコーナー <https://www.kkaneko.jp/ai/labo/>



Windowsパソコンで実行可能

演習 4 . Colabによる AI 画像認識体験

画像認識AIを、自分の手で動かしてみる

一から学習させなくても、写真を入れるだけで犬種まで当ててくれる

Labrador_retriever (46.9%)



--- AI Image Recognition Results ---

1. Labrador_retriever: 46.9%
2. Eskimo_dog: 5.4%
3. Ibizan_hound: 2.4%
4. golden_retriever: 2.3%
5. basenji: 1.4%

体験内容

- 学習済みモデルの利用 — 自分で一から学習させなくても、すぐに画像認識を体験できる
- 上位5候補を確率付きで返す — AIは「断定」ではなく「可能性」で答える

「犬種を『細胞画像・製品の傷・植物の病気・表情』に置き換えれば、医療診断や品質検査など本物の研究になる。簡単に見える土台こそ、応用の出発点。」

▶ Google Colab で今すぐ試す

<https://colab.research.google.com/drive/1scoDefaCCes1BUq7WHtDMxohYmAEmoC6?usp=sharing>

演習 4 . Colabによる AI 画像認識体験

手順

1. https://colab.research.google.com/drive/1scoDefaCCEs1BUq7WHtDMxo_hYmAEmoC6?usp=sharing にアクセス
ログイン画面表示が乱れる場合はシークレットモードでColabを開き直す
2. セルを上から順に実行する
3. AIの結果（上位5カテゴリと確率）が表示されることを確認する

```
import os
import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import (
    MobileNetV2, preprocess_input, decode_predictions)
import numpy as np
import matplotlib.pyplot as plt

# --- サンプル画像のダウンロード (TensorFlow公式のラブラドル画像) ---
img_path = tf.keras.utils.get_file(
    'YellowLabradorLooking_new.jpg',
    'https://storage.googleapis.com/download.tensorflow.org/example_images/YellowLabradorLooking_new.jpg'
)

# --- モデルの読み込み ---
model = MobileNetV2(weights='imagenet')

# --- 画像の読み込みと前処理 ---
img = tf.keras.utils.load_img(img_path, target_size=(224, 224))
x = tf.keras.utils.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# --- 推論 ---
preds = model.predict(x, verbose=0)
results = decode_predictions(preds, top=5)[0]

# --- 結果の表示 ---
top_label, top_score = results[0][1], results[0][2]

plt.figure(figsize=(6, 6))
plt.imshow(img)
plt.axis('off')
plt.title(f'{top_label} ({top_score*100:.1f}%)', fontsize=14)
plt.show()

print("--- AI Image Recognition Results ---")
for rank, (imagenet_id, label, score) in enumerate(results, 1):
    print(f" {rank}. {label}: {score*100:.1f}%")
```

Labrador_retriever (46.9%)

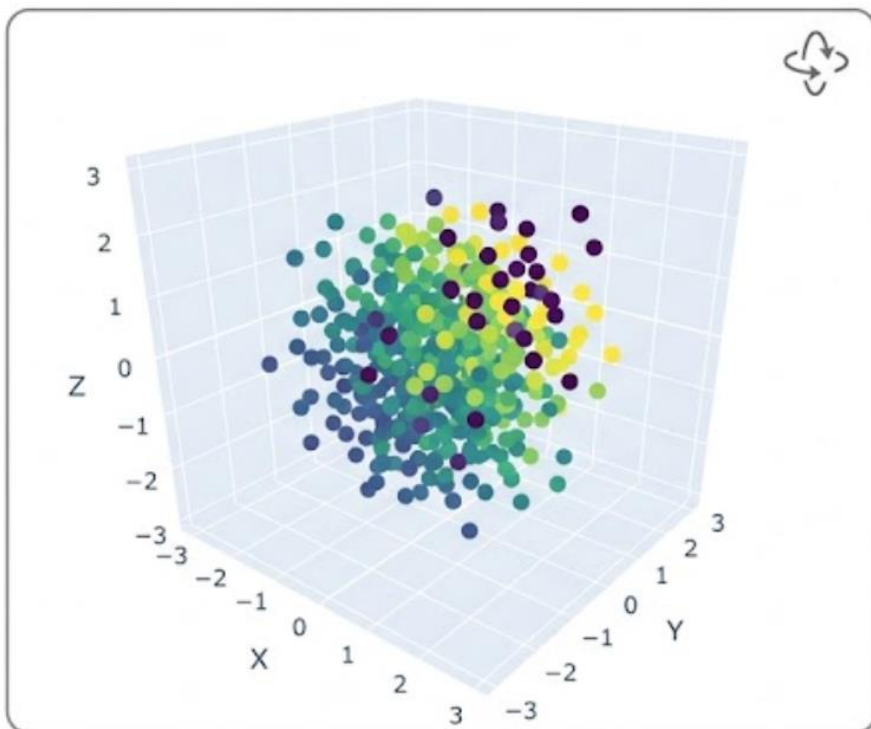


```
--- AI Image Recognition Results ---
1. Labrador_retriever: 46.9%
2. Eskimo_dog: 5.4%
3. Ibizan_hound: 2.4%
4. golden_retriever: 2.3%
5. basenji: 1.4%
```

演習 5. Colab による 3次元処理

3次元グラフを、自分の手で回してみる

マウスで回転・拡大できる。『3次元デジタル世界』への入口となる演習



体験内容

- plotlyによるインタラクティブな3次元グラフ
— マウスで回転・拡大して立体を観察できる
- 「3次元デジタル世界」への入口
— BlenderやゲームエンジンとAIの連携にもつながる

点を『分子構造・地形・人体・建築物』に置き換えれば、3D設計やシミュレーション、VR/メタバース研究になる。簡単に見える土台こそ、応用の出発点。

▶ Google Colab で今すぐ試す

<https://colab.research.google.com/drive/1lEXtElnxa4lVM6EgpOUUpGCWgAFibV1l9?usp=sharing>

演習 5 . Colab による 3次元処理

手順

1. <https://colab.research.google.com/drive/1EXtEInxa4IVM6EgpOUpGCWgAFibV1I9?usp=sharing> にアクセス
ログイン画面表示が乱れる場合はシークレットモードでColabを開き直す
2. セルを上から順に実行する
3. 3次元散布図が表示されたら、マウスでドラッグして回転させてみる
4. データの数 `n` を変えたり、`colorscale` を `Plasma` や `Cividis` に変えて再実行する
5. 自分なりの工夫を振り返る

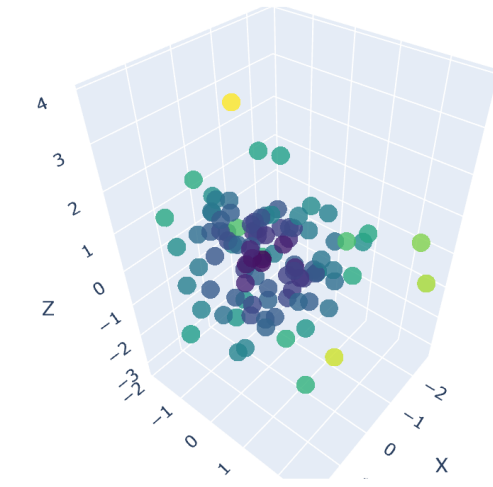
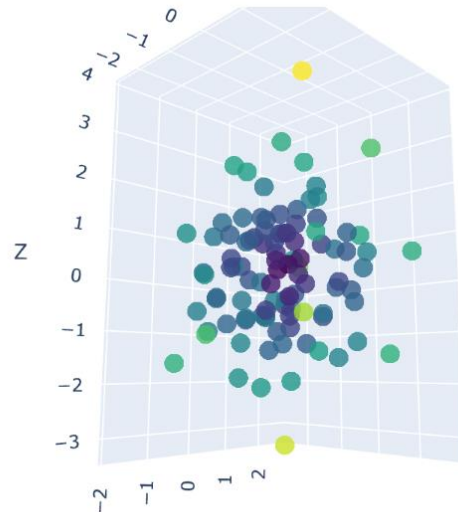
```
import plotly.graph_objects as go
import numpy as np

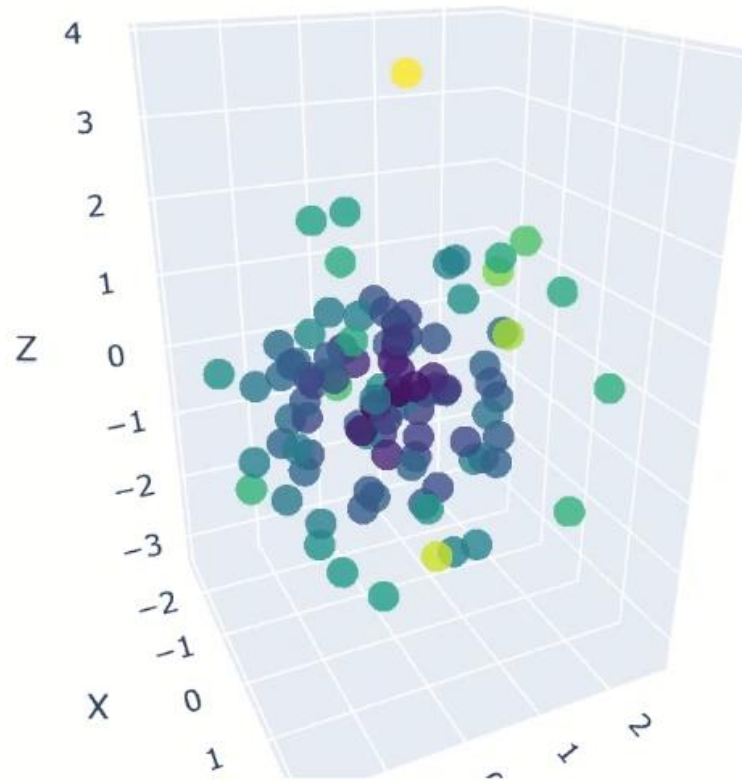
np.random.seed(42)
n = 100
x = np.random.randn(n)
y = np.random.randn(n)
z = np.random.randn(n)
colors = np.sqrt(x**2 + y**2 + z**2)

fig = go.Figure(data=[go.Scatter3d(
    x=x, y=y, z=z,
    mode='markers',
    marker=dict(size=5, color=colors, colorscale='Viridis', opacity=0.8)
)])

fig.update_layout(
    title='3D Scatter Plot',
    scene=dict(xaxis_title='X', yaxis_title='Y', zaxis_title='Z')
)

fig.show()
```





金子研究室・研究領域①

最新の AI による社会課題解決

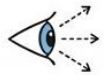
技術例



• 写真の中から人や車を見つける



• 人の体の動きを3次元で捉える



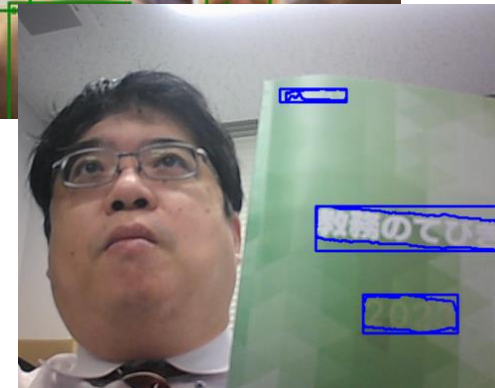
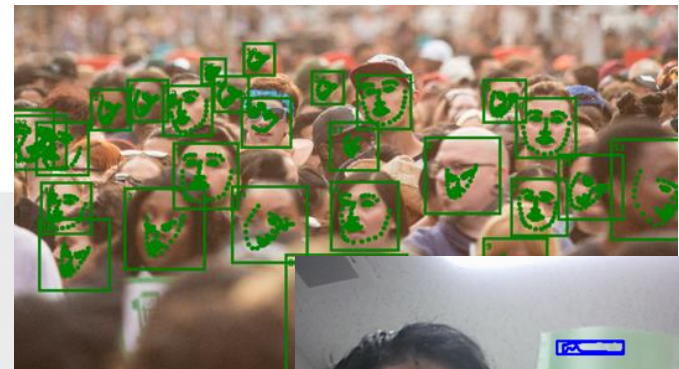
• 人がどこを見ているかを推定する



• 言葉で指定した物体を見つける



• 画像を領域に分ける



研究テーマ例 (参考)

- 視線推定技術を使って「授業中の学生の注目度を測る」
- 物体検出を使って「駐車場の空き状況をリアルタイムに把握する」

金子研究室・研究領域②

手元のPCで動くAI アシスタント

私はファンタジーの世界を舞台に冒険を始めることにした。そこは魔法と神秘的な力が満ち溢れる世界であり、古き魔法学院、神秘の森、危険なダンジョン、そして強大なドラゴン族が存在する。この世界では、勇者たちが魔法の剣や杖を手し、邪悪な魔術師や凶暴なモンスターと戦っている。私のために、魅力的なファンタジー冒険物語を創作してください。私は、若き魔法使いの弟子、勇敢な騎士、あるいは特殊な能力を持つ冒険者など、どのような役割でも構いません。

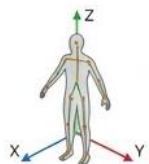
遙か古代の魔法が眠る「エメラルドの森」の入り口に立つあなたは、ある謎めいた指輪を見つける。その石は微かな魔力を発しており、学院の年長者たちはこれが失われた古代技術の鍵に関わるものだと警告していた。今夜は満月の夜。森を抜ける小川沿いの古びた道すがら、ふと、獣の唸りではなく、金属くすんだ重い足音が聞こえてくる。それは、深淵にいる危険なダンジョンの調査を目的とした、あるいは何かを追う邪悪な魔術師の手先かもしれない。指輪の魔力が脈打ち、あなたの心には「進むべき道」への強い衝動が灯る。周りには魔法学院の静けさと、未知なる闇の囁きが入り混じった空気が漂っている。

その指輪をそっと握りしめ、森の奥へと深く潜入する。周囲の環境を探り、この謎の足音の正体を慎重に調査する。即座に武器を取り、遭遇したかもしれない事態に備え、精神を集中させる。

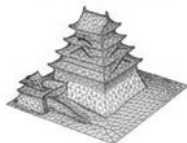
金子研究室・研究領域③

3次元世界を扱う：3次元構築、AI連携、シミュレーション

技術例



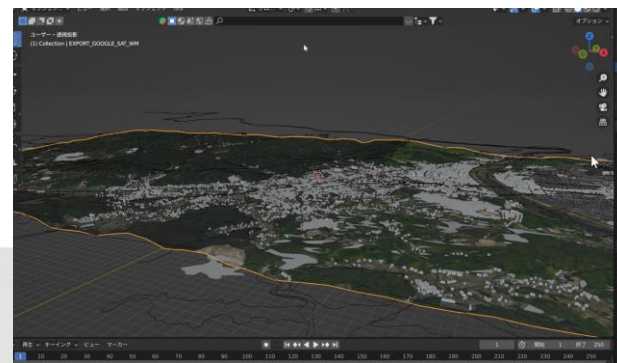
- 人体3次元姿勢推定（研究領域①の姿勢推定技術を、3次元空間再現と組み合わせて活用）



- 3次元の福山市の再現



- 画像からの3次元空間作成



研究テーマ例（参考）

- 防災計画、観光案内、仮想キャンパスツアー
- スマートフォンで撮影した写真から「自分の部屋の3次元モデル」を作成
- 福山市の観光スポットを3次元で再現した「仮想観光ツアー」

