

# 物体認識サーバを作りながら コード生成能力を試す

中西 克典

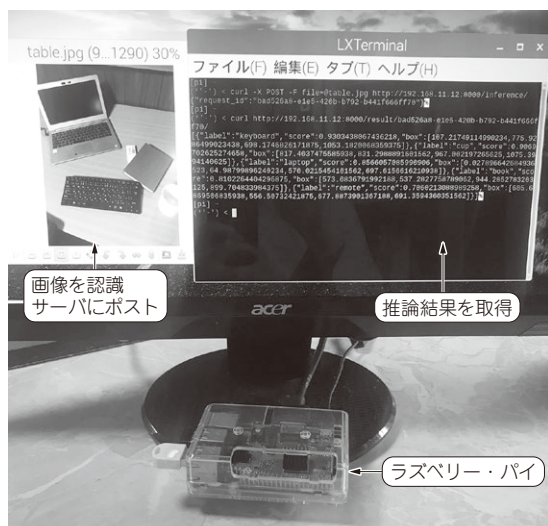


図1 ラズベリー・パイから認識サーバにアクセスしている様子

AIによる物体認識機能付きサーバ(以降、認識サーバ)を作ります。ラズベリー・パイ(以下、ラズパイ)などで収集したカメラ画像を認識サーバへポストすることで、認識サーバに搭載された強力な計算資源で物体検出の推論を実行するイメージです(図1)。複数の非力なエッジ・デバイスを用いるIoTシステムを手軽に実現できる典型的な方式です。

## ● 課題設定の意図

ChatGPTは有名なライブラリの使い方を熟知しています。そのため、API部分やデータベース部分のコード生成でChatGPTの威力が体験できると考え、サーバをテーマにしました。サーバの処理内容は、筆者が慣れた物体検出を選びました。物体検出モデルは大量にありますが、せっかくなのでSOTA(≒ベスト・スコア)相当のモデルを用いることにしました。

当然、ChatGPTはそのような最新手法は学習していません。このモデル選択は、ChatGPTが詳しくないツールを使わなければならない場面でのChatGPTとのやり取りを試す機会にもなるとも考えました。

表1 動作確認環境

項目	値
CPU	Ryzen 7 3700X (AMD)
GPU	GeForce RTX 3080(エヌビディア) (メモリ10GBバイト)
OS	Ubuntu 22.04
Docker	24.04, build 3713ee1
NVIDIA ドライバ	525.125.06
NVIDIA Container Toolkit	1.13.3

## 認識サーバ作りを試すための環境

### ● ソースはGitHubに

再現環境/サポートのため、本章のソースコードとChatGPTとのやり取りの全文を、

<https://github.com/kurusugawa-computer/interface-2023-11-chatgpt>

に公開しています(ソースコード・リポジトリと呼ぶ)。認識サーバの動作環境にはDockerを用います。app/launch.sh(詳細は後述)でDockerコンテナを起動し、このコンテナ内で本文中のコマンドを使って認識サーバを実行するという使い方を想定します。ただし、認識サーバの動作には適切なGPU環境が必要です。

### ● ソースコード・リポジトリの構造

本章に関係するソースコード・リポジトリのディレクトリは次の通りです。

- app…認識サーバのソースコードです。サーバ・ディレクトリと呼びます
- app/images…認識サーバの動作確認に使った画像があります
- docker…認識サーバ用Dockerイメージの元になるファイルがあります
- prompts/実践編…本章でのChatGPTとのやり取りの全文があります。ログ・ディレクトリと呼びます