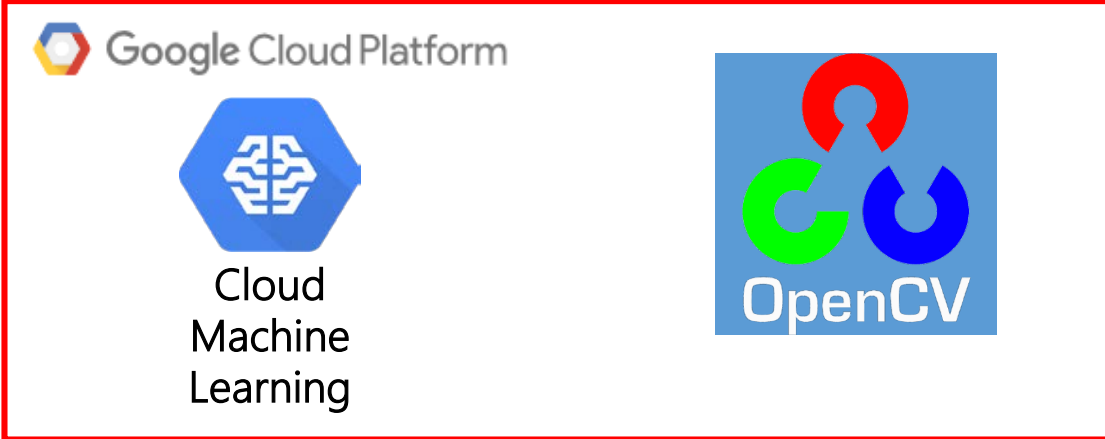



抜粋版


はじめてのクラウドAI開発キット2 ～グーグルのAIノウハウを最大活用、動画高速処理が可能に～ 開発編



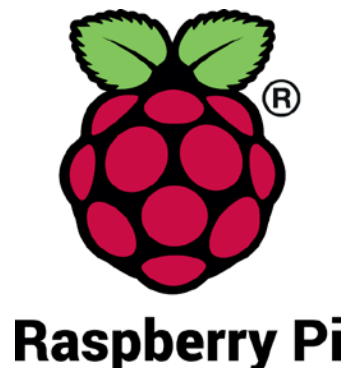
Google Cloud Platform



Cloud Machine Learning



OpenCV



スペクトラム・テクノロジー株式会社

<https://spectrum-tech.co.jp>

sales@spectrum-tech.co.jp

開発編 目次

抜粋版のためページと内容は一致しません

	ページ
• Raspberry Pi運用マニュアル	
1. RaspberryPiについて	<u>4</u>
2. Linux基本コマンド	<u>5</u>
3. RaspberryPi基本操作	<u>6</u>
4. 日常運用(ウイルススキャン、更新)	<u>7</u>
• クラウドAI公式ドキュメント	<u>9</u>
• GCP設定(Google cloud platform)	
1. アカウント登録	<u>10</u>
2. プロジェクト登録	<u>12</u>
3. API設定	<u>13</u>
4. APIキーの発行	<u>15</u>
5. サービスアカウントキーの発行	<u>16</u>
6. IAMと管理	<u>18</u>
• クラウドAI開発キット	
1. 開発キット全体像	<u>19</u>
2. AI活用例	<u>20</u>
3. 開発キットプログラム一覧	<u>22</u>
4. 使用方法	<u>28</u>

開発編 目次

抜粋版のためページと内容は一致しません

• クラウドAI開発キット

4. 使用方法

① Cloud vision:画像認識

- 1_OCR(文字認識)
- サービスアカウントキー設定
- API利用状況確認
- 2_face(顔検出)
- 3_label(ラベル検出)
- 4_web(web検出)
- 5_safe(不適切画像検出)
- 6_logo(ロゴ検出)
- 7_landmark(ランドマーク検出)
- 8_prop(プロパティ検出)
- 9_crop(切り出し検出)
- 複数画像同時処理

② Cloud speech:音声認識

- transcribe(短時間音声認識)
- Google cloud storageへのアップロード
- transcribe_async(非同期音声認識)
- transcribe_stream_mic(マイク音声認識)

③ Cloud Natural language:自然言語解析

- sentiment(感情分析)
- syntax(構文解析)
- entity(エンティティ分析)
- classify(カテゴリ分類):英語のみ

④ Cloud translate:翻訳

⑤ Cloud video:動画認識

• Opencv開発

1. Opencv

① 顔認識

② 顔検出

ページ

[28](#)

[33](#)

[35](#)

[36](#)

[40](#)

[43](#)

[46](#)

[49](#)

[52](#)

[55](#)

[58](#)

[60](#)

[64](#)

[65](#)

[68](#)

[69](#)

[72](#)

[74](#)

[75](#)

[76](#)

[77](#)

[79](#)

[83](#)

[84](#)

RaspberryPi運用マニュアル

1. Raspberry Piについて

既に全世界で1000万台以上販売された手のひらサイズのコンピュータです。
LinuxベースのRasbianOSで動作しております。

2. Linux基本コマンド

① システム関係

- 起動: 電源を入れると自動で起動します。
- 再起動: `$ reboot`
又は、`menu>shutdown>reboot`; 左上のメニューから
- 終了: `$ shutdown`
又は、`menu>shutdown>shutdown`; 左上のメニューから
- ログアウト `$ exit`
又は、`menu>shutdown>logout`; 左上のメニューから
- **日本語／英語の入力切替**: キーボードの CTL と `j` を同時に押します (コントロール: 左下と `j`)

RaspberryPi運用マニュアル

2. Linux基本コマンド

② ディレクトリ操作、コピー、移動、削除

pi@raspberrypi:~\$ **cd** /home/pi/Documents ディレクトリの切り替え
pi@raspberrypi:/home/pi/Documents\$ **ls** ファイルとディレクトリの表示(表示したら操作したいファイルを右クリックでコピーして操作します)

pi@raspberrypi:~\$ cp ファイル名 ディレクトリ	配下のディレクトリのファイルを別のディレクトリへコピー
pi@raspberrypi:~\$ mv ファイル名 ディレクトリ	配下のディレクトリのファイルを別のディレクトリへ移動
pi@raspberrypi:~\$ sudo rm ファイル名	ファイルの削除

便利な機能 **rm -help** コマンドのオプションが分からない場合は、ヘルプで問い合わせる。すべてのコマンド共通(マイナスを2個とhelp)

③ ユーザ権限、プロセス他

pi@raspberrypi:~\$ su -	スーパーユーザ(root)に切り替え、パスワードを入力
pi@raspberrypi:~\$ ps a	現状の動いているプロセスを表示
pi@raspberrypi:~\$ kill	特定のプロセスを強制終了
pi@raspberrypi:~\$ sudo apt-get install pkg	パッケージのインストールなどに使用
pi@raspberrypi:~\$ date	日付、時間の設定を行います。
pi@raspberrypi:~\$ sudo leafpad /etc/network/interfaces	インタフェースに記述している内容を変更します。Viよりも使いやすいです。

④ モジュール、usb、メモリ、HDDなどの表示

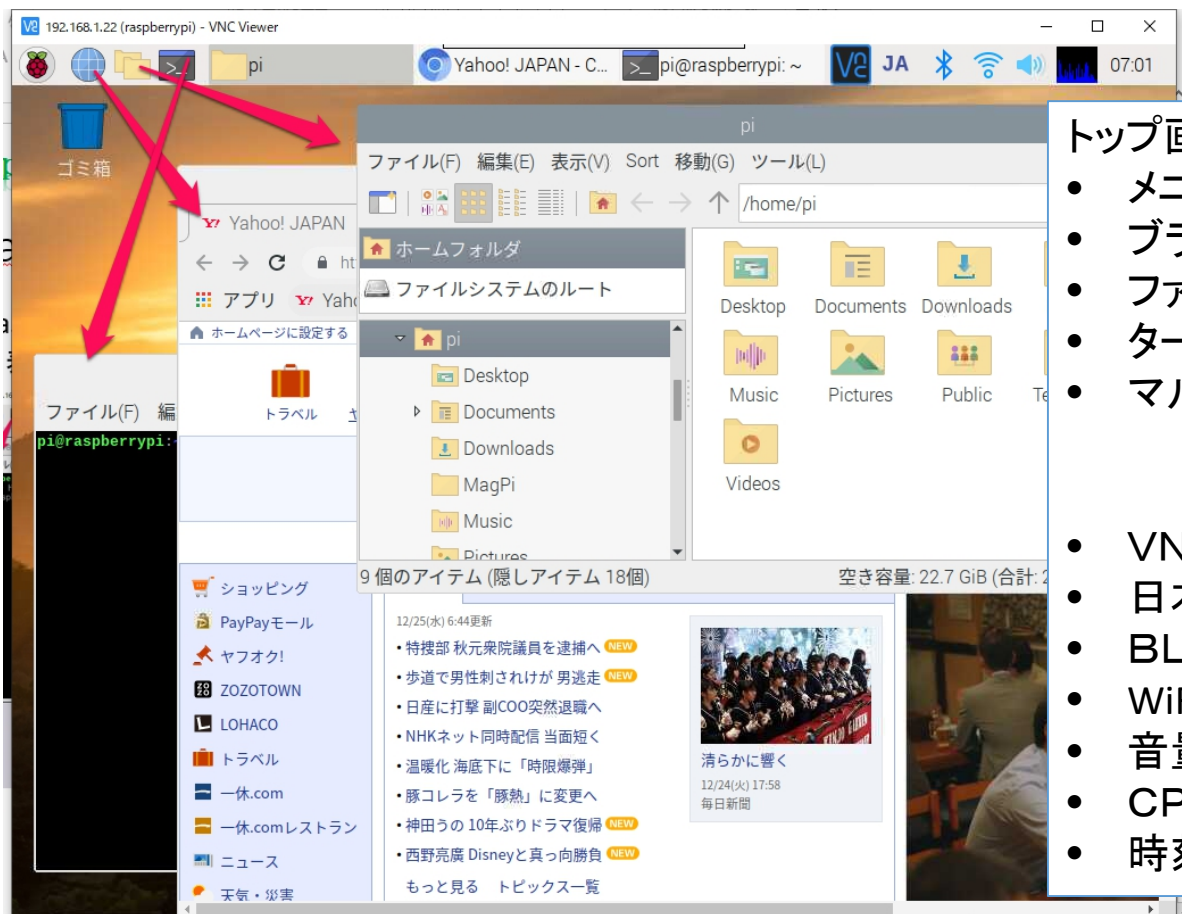
pi@raspberrypi:~\$ lsmod	linuxのモジュールリスト表示
pi@raspberrypi:~\$ lsusb	usbのデバイス表示
pi@raspberrypi:~\$ free -mt	メモリ使用状態表示
pi@raspberrypi:~\$ df	HDD(マイクロSD)の使用状態表示

RaspberryPi運用マニュアル

3. Raspberry Piの基本操作

① 表示画面と内容

デスクトップ上によく使うコマンド.txtがあります。
コピーして使ってください



- トップ画面(上段のタスクバーで選択)
- メニュー
 - ブラウザ
 - ファイルマネージャ
 - ターミナル
 - マルチ画面選択
-
- VNC
 - 日本語入力
 - BLE
 - WiFi
 - 音量
 - CPU使用率
 - 時刻

GCP設定 (Google cloud platform)

1. アカウント登録

- ・GCPにアクセスして、無料トライアルを申しこみ、アカウント登録を実施します

<https://console.cloud.google.com/>

無料トライアル終了後は、有料になります。各社で料金をチェックしてください。

The screenshot shows the Google Cloud Platform website. On the left, a navigation menu includes 'Google 選ぶ理由', 'サービス', 'ソリューション', 'ランチャー', '料金', and '導入事例'. The main content area features a video player and a '無料トライアル' (Free Trial) button, which is highlighted with a red box and an arrow pointing to the right. The right side of the image shows a modal window titled 'Google Cloud Platform の無料トライアル' with the following details:

- 国: 日本
- 受諾: はい いいえ
- Google Cloud Platform 無料トライアルの利用規約を讀んだうえで内容に同意します。続行するには「はい」を選択する必要があります
- はい いいえ
- 同意して続行

On the far right, a separate box lists terms and conditions for the trial:

- すべての Cloud Platform サービスへのアクセス
アプリや、ウェブサイト、サービスの構築と実行に必要な Firebase や Google Maps API などすべて使用できます。
- \$300 相当のクレジットを無料でご提供
ご登録いただく、Google Cloud Platform で今後 12 か月間ご利用いただける \$300 のクレジットを獲得できます。
- 無料トライアル期間が終了しても、自動的に請求されることはありません
ロボットによる登録ではないことを確認するため、クレジット カード番号を確認しております。有料アカウントに手動でアップグレードしない限り、課金されることはありません。

GCP設定 (Google cloud platform)

1.アカウント登録

- ・住所、会社名、氏名、電話、クレジットカード情報を入力して、申し込みます

Google Cloud Platform

Google Cloud Platform の無料トライアル Google

お客様情報

アカウントの種類 ⓘ

ビジネス

名前と住所 ⓘ

郵便番号 ⓘ

郵便番号を入力してください ⓘ

都道府県

市区郡

住所 1 行目

住所 2 行目

企業/組織名

企業/組織名

名前

メインの連絡先 ⓘ

お支払いタイプ

毎月の自動支払い

このサービスの費用を毎月 1 回定期的にお支払いいただけます。お支払い期限になると自動的に請求が行われます。

お支払い方法 ⓘ

カードの詳細

クレジット (デビット) カードの住所は上記と同じ

無料トライアルを開始

すべての Cloud Platform サービスへのアクセス

アプリや、ウェブサイト、サービスの構築と実行に必要な Firebase や Google Maps API などがすべて使用できます。

\$300 相当のクレジットを無料でご提供

ご登録いただくと、Google Cloud Platform で今後 12 か月間ご利用いただける \$300 のクレジットを獲得できます。

無料トライアル期間が終了しても、自動的に請求されることはありません

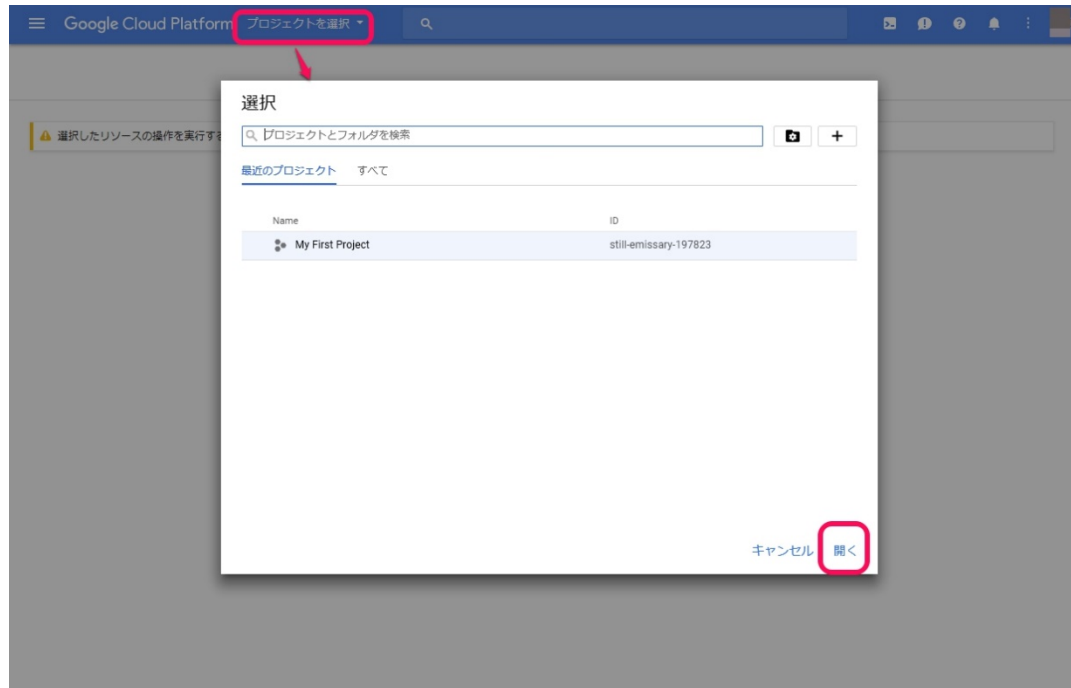
ロボットによる登録ではないことを確認するため、クレジットカード番号を確認しております。有料アカウントに手動でアップグレードしない限り、課金されることはありません。

プライバシーポリシー | よくある質問

GCP設定 (Google cloud platform)

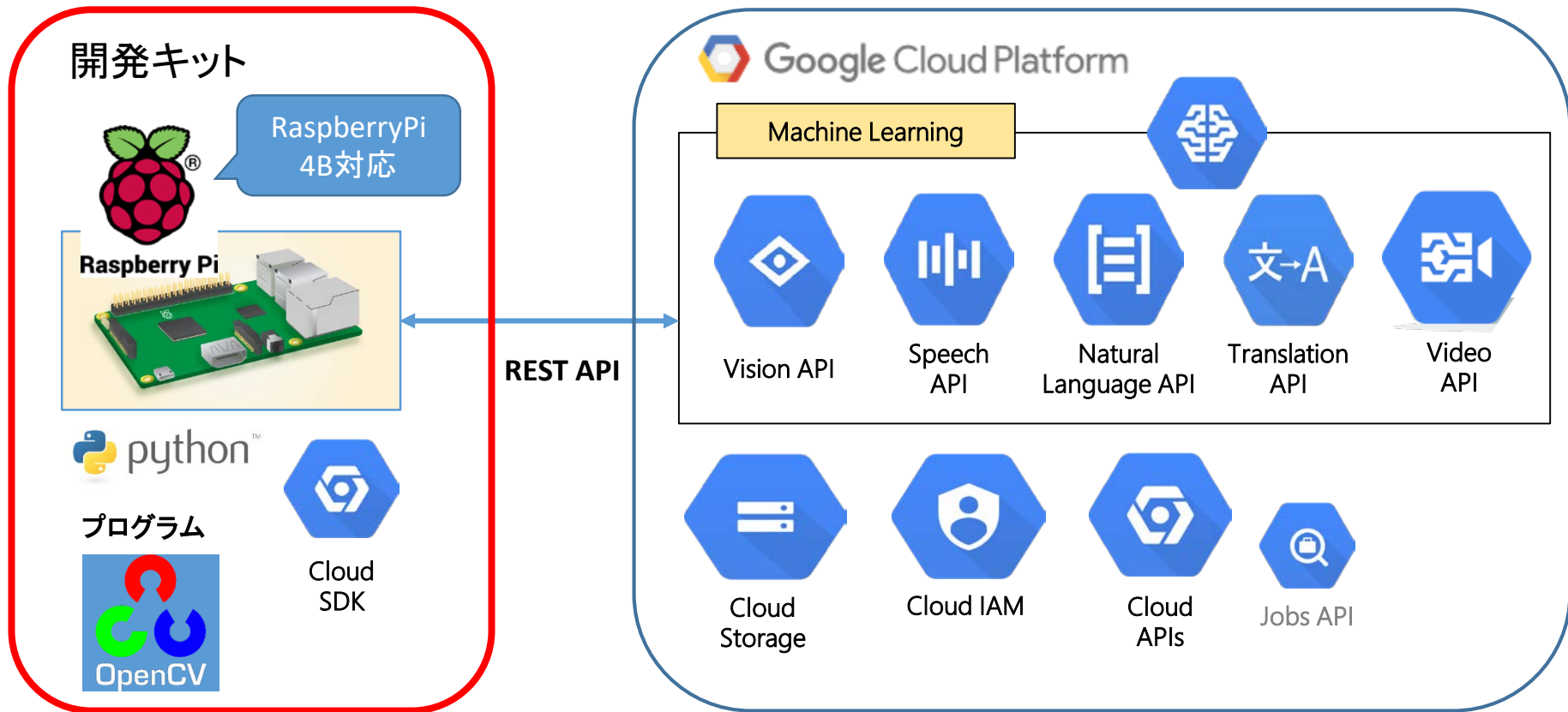
2.プロジェクト登録

- ・GCPのホームからプロジェクトを選択して、プロジェクト名を記入して、登録します。
- ・プロジェクト名とプロジェクトIDは作成後は、修正できません。



クラウドAI開発キット

1. 開発キット全体像



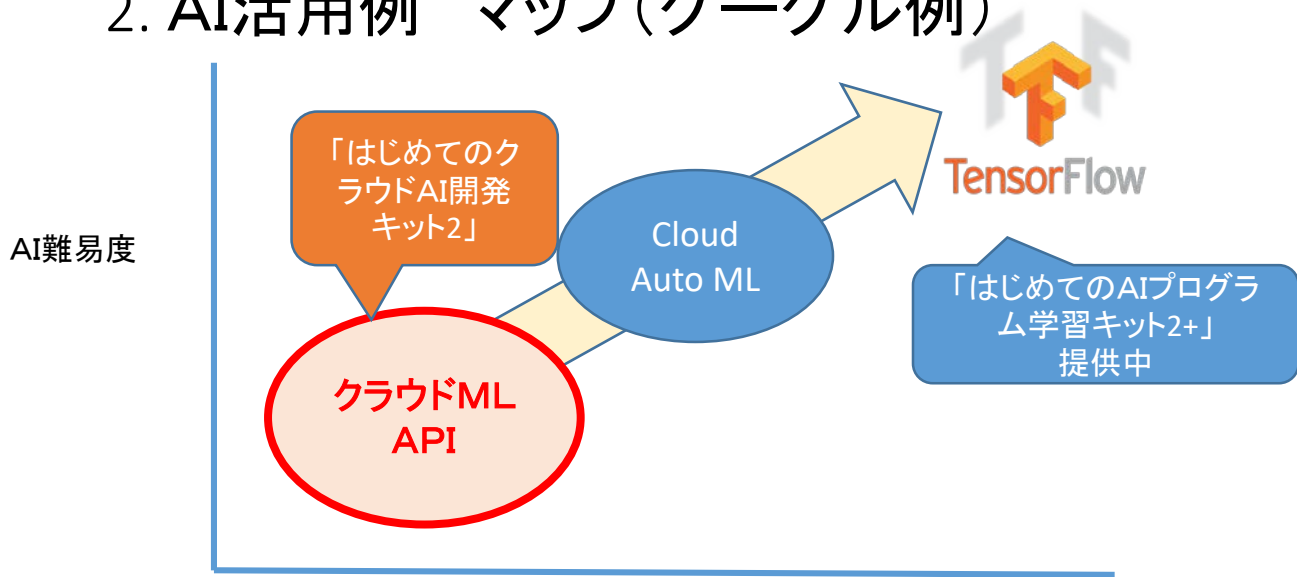
クラウドAI開発キット

2. AI活用例

カテゴリ	活用例	要素技術	
画像認識	<ul style="list-style-type: none"> ・OCRでの読み取り代替 ・写真のテキスト検出 ・顔検出による人数カウント ・不適切画像検出 ・類似デザイン検出 ・画像からWebサイト検出 ・画像へのインデックス付け 	Ocr Ocr Face Safe Web label	
音声認識	<ul style="list-style-type: none"> ・自動議事録作成 ・動画、音声ファイルからの文字起こし 	Transcribe_stream Transcribe_async	
自然言語解析	<ul style="list-style-type: none"> ・感情分析(音声認識との組合せ) ・エンティティ認識 	Sentiment+Transcribe_async entity	
自動翻訳	<ul style="list-style-type: none"> ・多言語翻訳 ・音声認識し、多言語で文字出力 	Snippet Translate+transcribe_async	
動画認識	<ul style="list-style-type: none"> ・不適切動画検出 ・ラベル検出 ・音声文字起こし 	Adult Label Transcribe_async	

クラウドAI開発キット

2. AI活用例 マップ(グーグル例)



一般向け

使用スキル

専門家向け

サービス名	特徴	考慮事項
クラウドML API	<ul style="list-style-type: none"> • グーグルのノウハウ(学習済モデル)を利用 • プログラム作成が簡単 • AI技術者が不要 	<ul style="list-style-type: none"> • カスタマイズが難しい
Cloud Auto ML	<ul style="list-style-type: none"> • クラウドAIとTensorflowの中間でカスタマイズが可能 • 高度なAI技術者が不要 	
Tensorflow	<ul style="list-style-type: none"> • 自由にモデルを作成可能 	<ul style="list-style-type: none"> • 学習データはユーザが準備 • AI技術者が必要

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
B: チャレンジ
C: 試験段階

クラウドAI名	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
Cloud vision api (画像認識)	OCR(文字認識)	画像内のテキストを検出、抽出できます。幅広い言語がサポートされており、言語の種類も自動で判別されます。	A	90%	手書き文字はC:10%位
	Face(顔検出)	画像に含まれる複数の人物の顔を検出できます。感情や帽子の着用といった主要な顔の属性についても識別されます。	A	90%	特定の人顔認識はできません
	Label(ラベル検出)	乗り物や動物など、画像に写っているさまざまなカテゴリの物体を検出できます。	A	90%	
	Web(ウェブ検出)	類似の画像をインターネットで検索できません	A	90%	
	Safe(不適切画像検出)	アダルトコンテンツや暴力的コンテンツなど、画像に含まれる不適切なコンテンツを検出できます	A	90%	
	Logo(ロゴ検出)	画像に含まれる一般的な商品ロゴを検出できます	C	10%	文字のないロゴは検出できない
	Landmark(ランドマーク検出)	画像に含まれる一般的な自然のランドマークや人工建造物を検出できます。	B	50%(画像次第)	
	Prop(画像属性検出)	画像のドミナントカラーや切り抜きのヒントなど、画像の一般的な属性を検出できます	B		
	Crop(画像切り抜き)	認識できる画像に切り抜きできます。画像の最小化	B		
	multipage	複数画像を処理できるプログラム			
	loadjson	Json形式のデータを読み出すプログラム			

http方式とライブラリ方式の二つ準備

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
B: チャレンジ
C: 試験段階

クラウドAI名	部品名	機能内容	実用度(主観)	信頼度(主観)	備考
Cloud speech api (音声認識、多言語対応)	Transcribe(同期音声認識)	ローカルに保存されている、短い音声(1分未満)を処理し、認識されたテキストを出力します。	A	70%	固有名詞が難しい
	Transcribe_async(非同期音声認識)	Google Cloud Storage に保存されている、1分より長い音声を認識し、テキストで出力します。 動画から音声テキストの検出も可能	A	70%	
	Transcribe_stream(ストリーミング音声認識)	ローカルのリアルタイムでストリーミング音声認識します			ローカルなので上記の部品で代用可能
	Transcribe_stream_mic(マイク入力ストリーミング音声認識)	マイク入力で、リアルタイムで音声認識を行い、テキストを出力します。 びっくりする能力。是非お試しください。 Googleドキュメントの音声入力と同じです。(*1)	A	70%	すごいです。
	pyaudio	マイクの試験用プログラム			

(*1)USBマイクは付いておりませんので、お客様で準備してください

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
 B: チャレンジ
 C: 試験段階

クラウドAI名	部品名	機能内容	実用度(主観)	信頼度(主観)	備考
Cloud Natural language api(自然言語処理、多言語)	Sentiment(感情分析)	テキストのブロック内で示されている全体的な感情を読み取ることができます。	B		
	Syntax(構文解析)	トークンと文の抽出、品詞(PoS)の特定、各文の係り受け解析ツリーの作成が可能です。	B		文字ずれ発生中:原因不明
	Entity(エンティティ認識)	エンティティとラベル(人、組織、場所、イベント、商品、メディアなど)を特定できます	A	90%	
	Classify(カテゴリ分類) 英語のみ	事前定義された 700 以上のカテゴリでドキュメントを分類できます	A	90%	英語のみ
Cloud translate api(翻訳、多言語)	Quickstart(テキスト翻訳)	グーグルの自動翻訳。多言語対応。プログラム上で言語を指定	A	70%	
	Snippet(テキスト翻訳、多言語)	グーグルの自動翻訳。多言語を翻訳時に指定。	A	70%	

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
 B: チャレンジ
 C: 試験段階

クラウドAI名	部品名	機能内容	実用度(主観)	信頼度(主観)	備考
Cloud video api beta (動画認識)	Label(ラベル認識)	「犬」、「花」、「車」などの動画内のエンティティを検出します	A	70%	
	adult(不適切動画検出)	アダルトコンテンツや暴力的コンテンツなど、画像に含まれる不適切なコンテンツを検出できます	A	70%	
	analyze(統合型分析)	Shot: 動画内のシーンの変更を検出します Adult: 不適切シーンの検出	B		
	Face(顔認識)	プログラムはありますが、エラー。プライバシー保護のため	C		エラー
	音声認識	Speech>transcribe_asyncにて、動画の音声のみ取り出して、文字起こしが可能	A	70%	便利

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
B: チャレンジ
C: 試験段階

クラウドAI名	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
Cloud video api (動画認識)	Label(ラベル認識)	「犬」、「花」、「車」などの動画内のエンティティを検出します	A	90%	
	adult(不適切動画検出)	アダルトコンテンツや暴力的コンテンツなど、画像に含まれる不適切なコンテンツを検出できます	A	90%	
	analyze(統合型分析)	Shot: 動画内のシーンの変更を検出します Adult: 不適切シーンの検出	B		
	音声認識	Speech>transcribe_asyncにて、動画の音声のみ取り出して、文字起こしが可能	A	80%	便利
	カメラ(リアルタイム)	リアルタイムで物体、顔のパーツを識別、高速で処理	A	90%	RaspberryPi4Bにより高速処理が可能

クラウドAI開発キット

3. 開発キットプログラム一覧

A: 実用可能
 B: チャレンジ
 C: 試験段階

区分名	プログラム名	機能内容	実用度 (当社評価)	信頼度 (当社評価)	備考
opencv	FaceDecton (顔認識)	顔の目、顔全体を認識します。	A	90%	
	FaceRecognition (顔検出)	顔登録: 4名まで顔を登録できます。20枚/人の顔を1分で登録します	A	90%	
		顔学習	A	90%	
		顔検出: 登録した顔を検出し、確率を表示します。	A	80%	

ローカルで動作します。

クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

- テキスト認識: ocr1.py (画像からテキスト抽出)
 - APIキーの設定
 - Windows PC>ネットワーク>RaspberryPi>Pi>Documents>cloud vision>1_ocrをクリック
 - Ocr1.pyをwindows PCのローカルにコピーして、**さくらエディタ**などでocr1.pyを開きます。
 - 17行のstr_api_key="xxxxx"にGCP設定>4.APIキー設定で取得したキー(15ページ)を貼り付けます
 - また、画像ファイル名は、7行目です。適宜書き換えて使用してください。
 - また、データ出力は、53行目になります。デフォルトは、data1.jsonです。
 - 修正が完了したら、再度raspberry Piにocr1.pyを上書きしてください。

さくらエディタ

<https://sourceforge.net/projects/sakura-editor/>

```
1 #!/usr/bin/python
2 #coding:utf-8
3 import base64
4 import json
5 from requests import Request, Session
6
7 path = "choushi3.png"
8
9 def recognize_captcha(str_image_path):
10     bin_captcha = open(str_image_path, 'rb').read()
11
12     str_encode_file = base64.b64encode(bin_captcha)
13     #str_encode_file = base64.b64encode(bin_captcha).decode("utf-8")
14
15     str_url = "https://vision.googleapis.com/v1/images/annotate?key="
16
17     str_api_key = "XXXXXXXXXXXXXXXXXXXX"
18
19     str_headers = {'Content-Type': 'application/json'}
20
21     str_json_data = {
22         'requests': [
23             {
24                 'image': {
25                     'content': str_encode_file
26                 },
27                 'features': [
28                     {
29                         'type': "TEXT_DETECTION",
30                         'maxResults': 10
31                     }
32                 ]
33             }
34         ]
35     }
36
37     print("begin request")
38     obj_session = Session()
39     obj_request = Request("POST",
40                           str url + str api key,
```


クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

- テキスト認識: ocr1.py(画像からテキスト抽出)
 - Raspberry piからocr1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata1.jsonの双方にでます。

pyrhon3で実施してください

コマンド

```
$ cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr
$ python3 ocr1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

The screenshot shows a Raspberry Pi terminal window with the following commands and output:

```
pi@raspberrypi: ~
└─$ su -
root@raspberrypi: ~
└─# cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# ls
choushi3.png data2.json handwriting1.jpg ocr1.py ocr3.py receipt1.jpg
data1.json data3.json loadjson.py ocr2.py ocr4.py
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# python ocr1.py
begin request
end request
{
  "responses": [
    {
      "textAnnotations": [
        {
          "locale": "en",
          "description": "1625\n",
          "boundingPoly": {
            "vertices": [
              {
                "x": 128,
                "y": 222
              },
              {
                "x": 170,
```

The Google Cloud Vision interface shows the 'Text' tab selected, displaying the detected text '1625' from an image of a runner. A red box highlights the 'Text' tab in the interface.

クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

- テキスト認識: ocr1.py(画像からテキスト抽出)
 - jsonデータを、loadjson.pyでテキストのみ取り出します。
 - 読み出すデータは5行目に定義。
 - 必要に応じて書き換えてください。

コマンド

```
$ cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr  
$ python3 loadjson.py
```

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
}  
  }  
    "blockType": "TEXT"  
  }  
}  
1,  
  }  
}  
1,  
  "text": "1625\n"  
}  
1  
}  
1625  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# ^C  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# ls  
choushi3.png data2.json handwriting1.jpg ocr1.py ocr3.py receipt1.jpg  
data1.json data3.json loadjson.py ocr2.py ocr4.py  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr# python loadjson.py  
1625  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/1_ocr#
```

テキスト出力結果

```
デスクトップ#murakami_leno#ビジネス#技術関係#deep_learning#gcp_ai#cloud_vision#1_ocr#loadjson.py - sakura 2.2.0.1  
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)  
1 #!/usr/bin/python  
2 #coding:utf-8  
3 import json  
4  
5 f = open('data1.json', 'r')  
6 json_str = json.load(f) #json file load  
7 str = json_str["data"] #str to dictionary  
8  
9 print(json_str["data"])  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
1行 1桁 CRLF 23 SJS REC 挿入
```

クラウドAI開発キット

4.使用方法

① Cloud vision>1_OCR(文字認識)

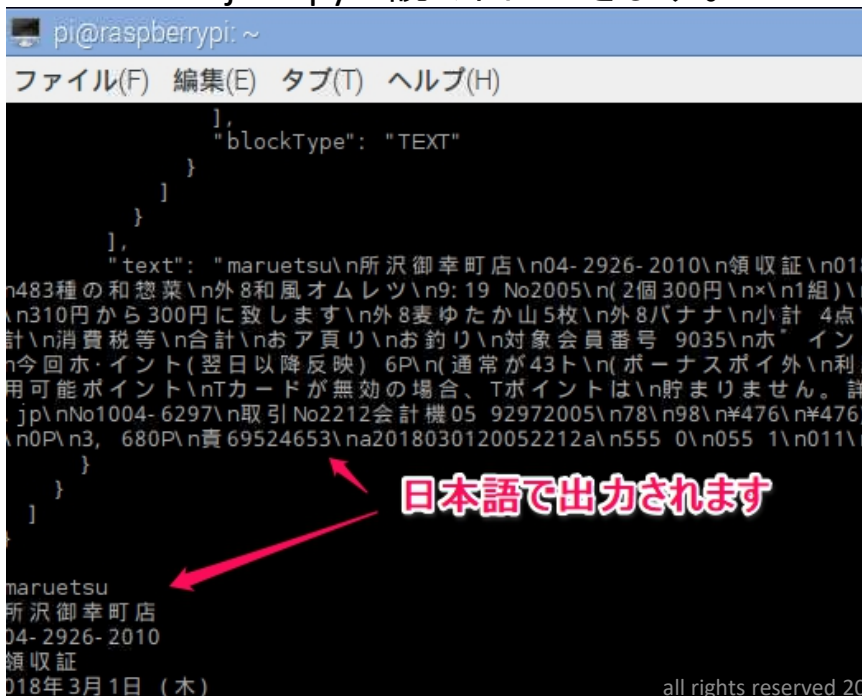
- テキスト認識: ocr2.py(文字読み取り)
 - 同様にApi キーを設定します。
 - Raspberry piからocr2.pyを実行します。
 - 出力は、piの画面上とファイルのdata2.jsonの双方にです。
 - loadjson.pyで読み出しできます。

コマンド

```
$ cd /home/pi/Documents/gcp_api/cloud_vision/1_ocr
$ python3 ocr2.py
$ python3 loadjson.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>



クラウドAI開発キット

4.使用方法

① Cloud vision>2_face(顔検出)

- face1.py (http使用)
 - Raspberry piからface1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata5.jsonの双方にでます。

コマンド

```
$ cd /home/pi/Documents/gcp_api/cloud_vision/2_face
$ python3 face1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
Documents/gcp_api/cloud_vision/2_face
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/2_face# ls
choushi3.png data5.json face1.py face2.py loadjson_f.py
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/2_face# python face1.py
begin request
end request

"responses": [
  {
    "faceAnnotations": [
      {
        "boundingPoly": {
          "vertices": [
            {
              "x": 125,
              "y": 112
            },
            {
              "x": 187,
              "y": 112
            },
            {
              "x": 187,
              "y": 184
            }
          ]
        }
      }
    ]
  }
]
```



顔認識は、できません。プライバシー保護

クラウドAI開発キット

4.使用方法

① Cloud vision>4_web(web検出)

- web1.py(http使用)
 - Raspberry piからweb1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata7.jsonの双方にでます。

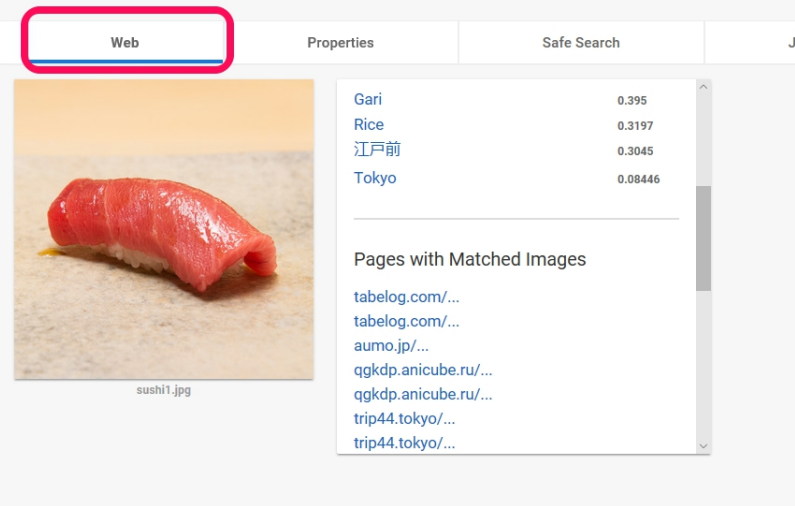
コマンド

```
$ cd /home/pi/Documents/gcp_api/cloud_vision/4_web
$ python3 web1.py
```

Webでの確認
<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
list1=if "webDetection" !f "pagesWithMatchingImages"]
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/4_web# python web1.py
begin request
end request

"responses": [
  {
    "webDetection": {
      "webEntities": [
        {
          "entityId": "/m/0h3rqf_",
          "score": 2.11,
          "description": "Sushi Saito"
        },
        {
          "entityId": "/m/07030",
          "score": 1.779,
          "description": "Sushi"
        },
        {
          "entityId": "/m/042ck",
          "score": 1.321,
          "description": "Japanese Cuisine"
        }
      ]
    }
  }
]
```



Web上で類似した画像、URLが出力されます

クラウドAI開発キット

4.使用方法

① Cloud vision>6_logo(ロゴ検出)

- logo1.py(http使用)
 - Raspberry piからlogo1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata9.jsonの双方にでます。

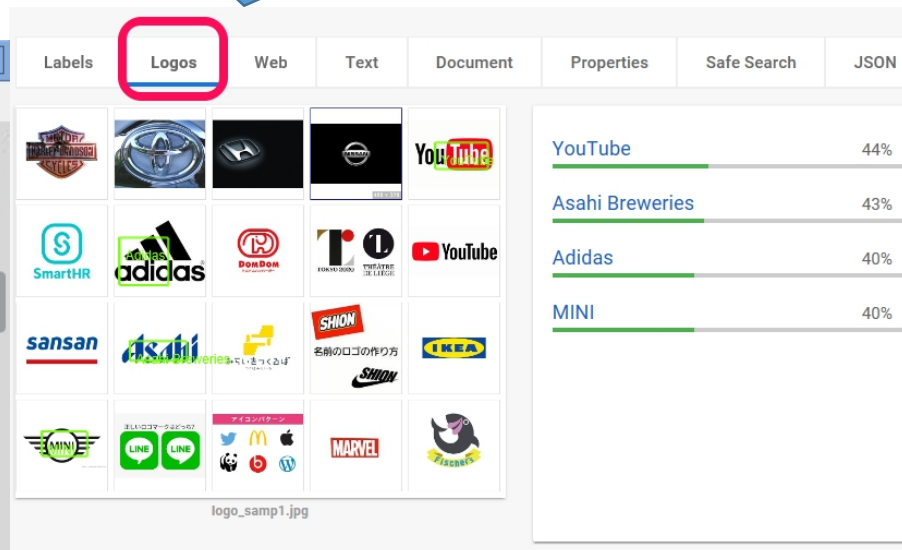
コマンド

```
$ cd /home/pi/Documents/gcp_api/cloud_vision/6_logo
$ python3 logo1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/6_logo# ls
data9.json loadjson_lo.py logo1.py logo2.py logo_samp1.jpg
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/6_logo# python logo1.py
begin request
end request
{
  "responses": [
    {
      "logoAnnotations": [
        {
          "mid": "/g/1dv8ql4m",
          "description": "YouTube",
          "score": 0.44070464,
          "boundingPoly": {
            "vertices": [
              {
                "x": 797,
                "y": 72
              },
              {
                "x": 897,
                "y": 72
              },
              {
                "x": 897,
                "y": 72
              },
              {
                "x": 797,
                "y": 72
              }
            ]
          }
        }
      ]
    }
  ]
}
```



ロゴ検出できます。テキストがない場合は検出できません

クラウドAI開発キット

4.使用方法

① Cloud vision>7_landmark(ランドマーク検出)

- landm1.py(http使用)
 - Raspberry piからlandm1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata11.jsonの双方にでます。

コマンド

```
$ cd
```

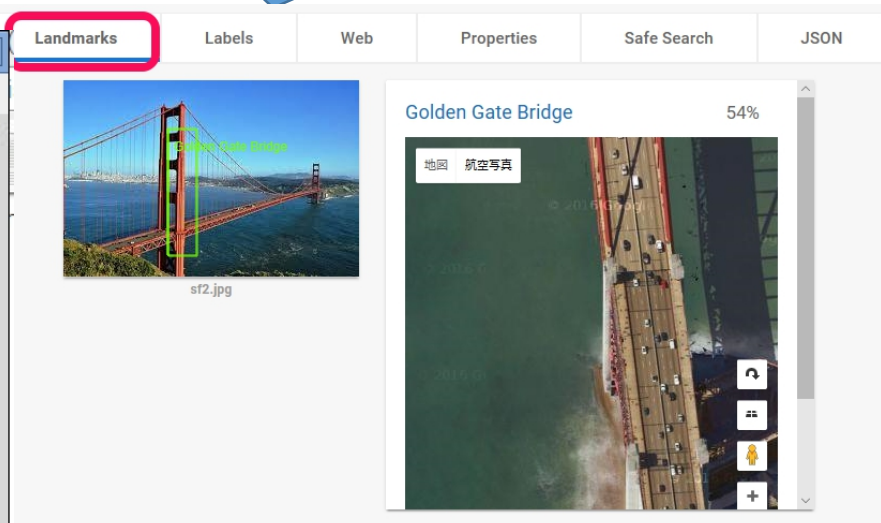
```
/home/pi/Documents/gcp_api/cloud_vision/7_landmark
```

```
$ python3 landm1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~  
ファイル(F) 編集(E) タブ(T) ヘルプ(H)  
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/7_landmark# python landm1.py  
begin request  
end request  
{  
  "responses": [  
    {  
      "landmarkAnnotations": [  
        {  
          "mid": "/m/035p3",  
          "description": "Golden Gate Bridge",  
          "score": 0.54256004,  
          "boundingPoly": {  
            "vertices": [  
              {  
                "x": 98,  
                "y": 46  
              },  
              {  
                "x": 125,  
                "y": 46  
              },  
              {  
                "x": 125,  
                "y": 125  
              },  
              {  
                "x": 98,  
                "y": 125  
              }  
            ]  
          }  
        }  
      ]  
    }  
  ]  
}
```



ランドマークが検出できます。

クラウドAI開発キット

4.使用方法

① Cloud vision>8_prop(プロパティ検出)

- Prop1.py(http使用)
 - Raspberry piからprop1.pyを実行します。
 - Api キーがあつてると、結果が出力されます。
 - 出力は、piの画面上とファイルのdata10.jsonの双方にでます。

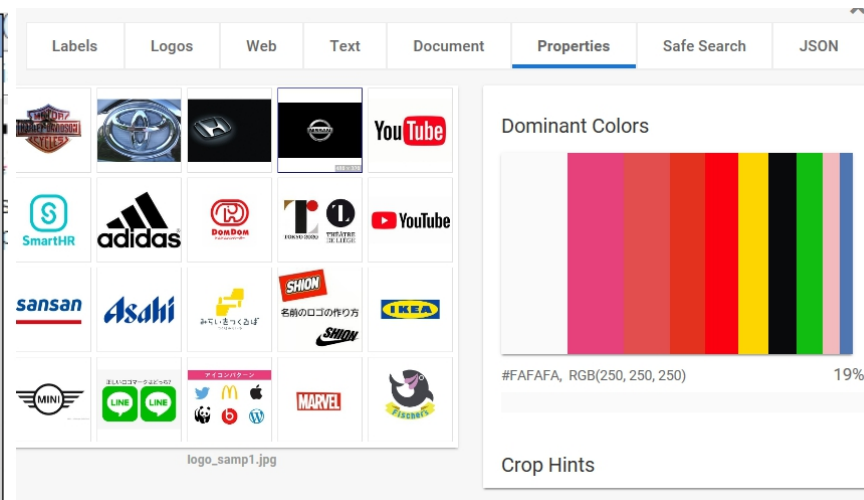
コマンド

```
$ cd
/home/pi/Documents/gcp_api/cloud_vision/8_prop
$ python3 prop1.py
```

Webでの確認

<https://cloud.google.com/vision/?authuser=1&hl=ja>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/8_prop# ls
data10.json loadjson_p.py logo_samp1.jpg prop1.py prop2.py
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/8_prop# python prop1.py
begin request
end request
{
  "responses": [
    {
      "imagePropertiesAnnotation": {
        "dominantColors": {
          "colors": [
            {
              "color": {
                "red": 250,
                "green": 250,
                "blue": 250
              },
              "score": 0.11050146,
              "pixelFraction": 0.7340543
            },
            {
              "color": {
                "red": 231,
                "green": 65,
```



画像の色などのプロパティ情報が検出できます。

クラウドAI開発キット

4.使用方法

① Cloud vision>9_crop(切り出し検出)

- Crop2.py(ライブラリ使用)
 - Raspberry piからpython3 crop2.py choushi3.jpg cropを実行します。
 - 画像ファイルは、crop2.pyの後に設定します。Jpgのみ有効です。
 - 出力は、output-crop.jpgです。

コマンド

```
$ cd  
/home/pi/Documents/gcp_api/cloud_vision/9_crop  
$ python3 crop2.py choushi3.jpg crop
```



output-crop.jpg

choushi3.jpg

クラウドAI開発キット

4.使用方法

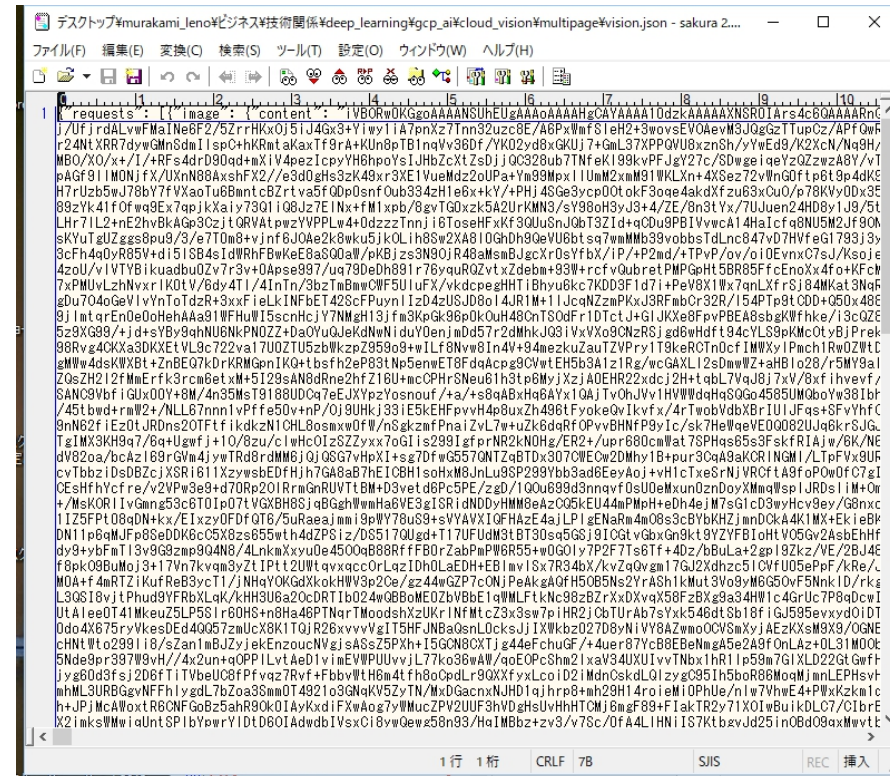
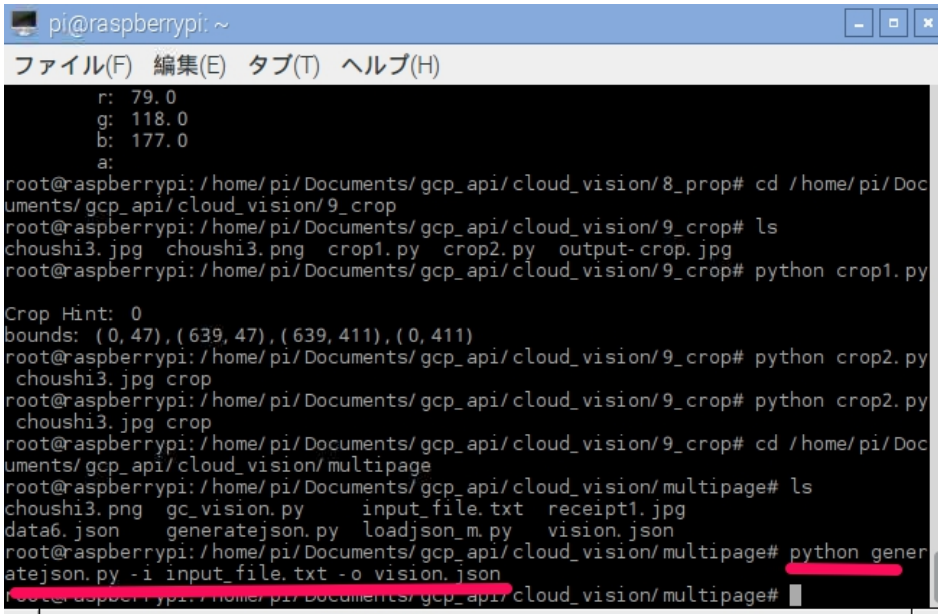
① Cloud vision>multipage (複数画像同時処理)

• Vision.json (cloud visionに送信するファイル作成)

- Generatejson.pyを使用して、送信ファイルを作成します。
- 入力ファイル名は、input_file.txt
- 出力ファイル名は、vision.json (任意に変えられます)

コマンド

```
$ cd /home/pi/Documents/gcp_api/cloud_vision/multipage $ python3 generatejson.py -i input_file.txt -o vision.json
```



vision.json

クラウドAI開発キット

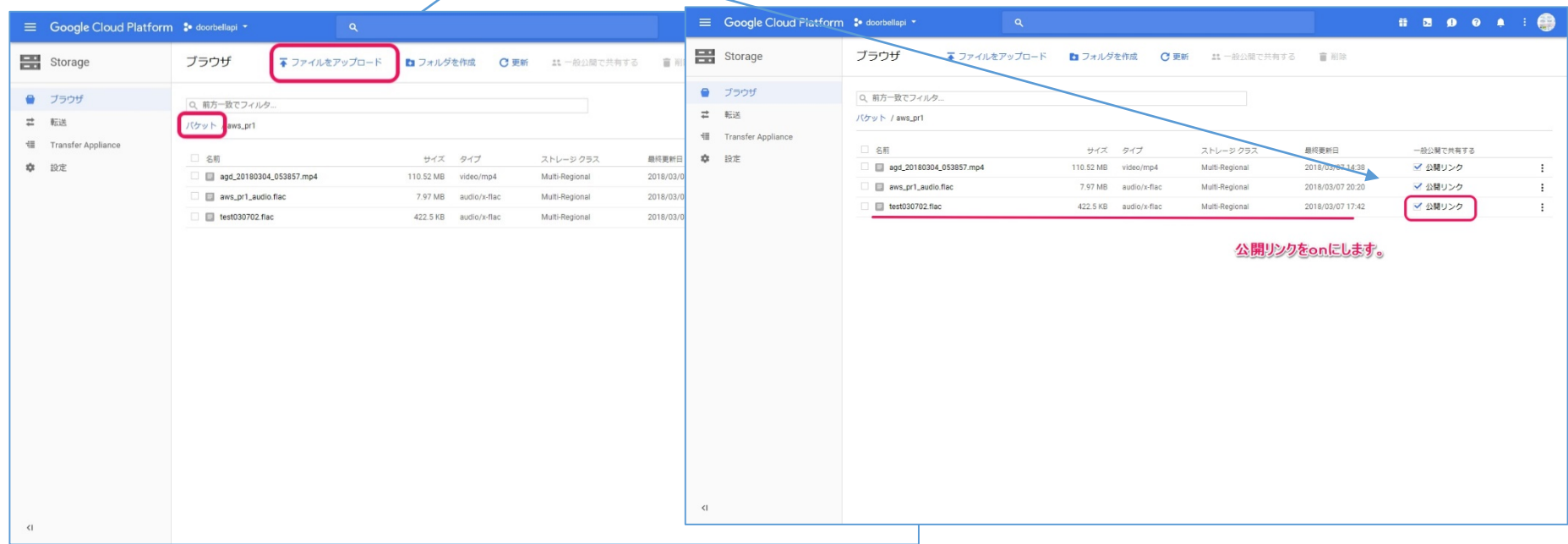
4.使用方法

② Cloud speech>transcribe (短時間音声認識)

- transcribe.py (google storage)
 - flacなどの音声ファイルをグーグルのストレージに保存して、で日本語テキストに変換
 - バケット>ファイルをアップロード
 - 公開リンクをONへ
 - リンク名は、gs://バケット名/test030702.flac

```
コマンド
$ cd
/home/pi/Documents/gcp_api/cloud_speech/
$ python3 transcribe.py gs://バケット名
/test030702.flac
```

音声ファイル変換サイト
<https://online-audio-converter.com/ja/>



クラウドAI開発キット

4.使用方法

② Cloud speech>transcribe (短時間音声認識)

- transcribe.py (google storage)
 - flacなどの音声ファイルをグーグルのストレージに保存して、で日本語テキストに変換
 - 音声ファイルを作成する場合は、1ch, 44.1KHzで行ってください。
 - コーデックを変更する場合は、76行の値を変更してください。Flacのみになります

```
コマンド
$ cd
/home/pi/Documents/gcp_api/cloud_speech/
$ python3 transcribe.py gs://バケット名
/test030702.flac
```

音声ファイル変換サイト
<https://online-audio-converter.com/ja/>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
#476)
#514
*6 0 4
0P
3, 680P
費 69524653
a2018030120052212a
555 0
055 1
011

root@raspberrypi: /home/pi/Documents/gcp_api/cloud_vision/multipage# cd /home/pi/
Documents/gcp_api/cloud_speech
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_speech# ls
output.wav      transcribe.py      transcribe_streaming_mic.py
pyaudio_test.py transcribe_async.py
test030701.wav  transcribe_streaming.py
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_speech# python transcribe.py t
est030701.wav
Transcript: 今日はテストですテストテスト村上雅彦
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_speech# python transcribe.py g
s://aws-pi-1/test030702.flac
Transcript: 今日はテストですテストテスト村上雅彦
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_speech#
```

```
デスクトップ#murakami_keno#ビジネス#技術関係#deep_learning#gcp_ai#cloud_speech#transcribe#transcribe.py - saku...
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)
63 +
64 # [START def_transcribe_gcs]+
65 def transcribe_gcs(gcs_uri):+
66     """Transcribes the audio file specified by the gcs_uri."""+
67     from google.cloud import speech+
68     from google.cloud.speech import enums+
69     from google.cloud.speech import types+
70     client = speech.SpeechClient()+
71 +
72     # [START migration_audio_config_gcs]+
73     audio = types.RecognitionAudio(uri=gcs_uri)+
74     config = types.RecognitionConfig(+
75         encoding=enums.RecognitionConfig.AudioEncoding.FLAC,+
76         sample_rate_hertz=44100,+
77         language_code='ja-JP')+
78     # [END migration_audio_config_gcs]+
79 +
80     response = client.recognize(config, audio)+
81     # Each result is for a consecutive portion of the audio. Iterate through+
82     # them to get the transcripts for the entire audio file.+
83     for result in response.results:+
84         # The first alternative is the most likely one for this portion.+
85         print(u'Transcript: {}'.format(result.alternatives[0].transcript))+
86     # [END def_transcribe_gcs]+
87 +
88 +
89 if __name__ == '__main__':+
90     parser = argparse.ArgumentParser(+
91         description=_doc,+
92         formatter_class=argparse.RawDescriptionHelpFormatter)+
93     parser.add_argument(+
94         'path', help='File or GCS path for audio file to be recognized')+
95     args = parser.parse_args()+
96     if args.path.startswith('gs://'):+
97         transcribe_gcs(args.path)+
98     else:+
99         transcribe_file(args.path)+
[EOF]
```


クラウドAI開発キット

4.使用方法

③ Cloud nl>sentiment(感情分析)

- senti1.py(全文解析)
 - sentiment1.pyを実行します。
 - 14行目にテキストのファイル名を設定していません。test_sentence.txtを添付しています。
 - スコア(ポジティブ、ニュートラル、ネガティブ)と感情の強度がでます。詳細は以下のURL参照。
 - <https://cloud.google.com/natural-language/docs/basics?authuser=1&hl=ja#interpreting-sentiment-analysis-values>

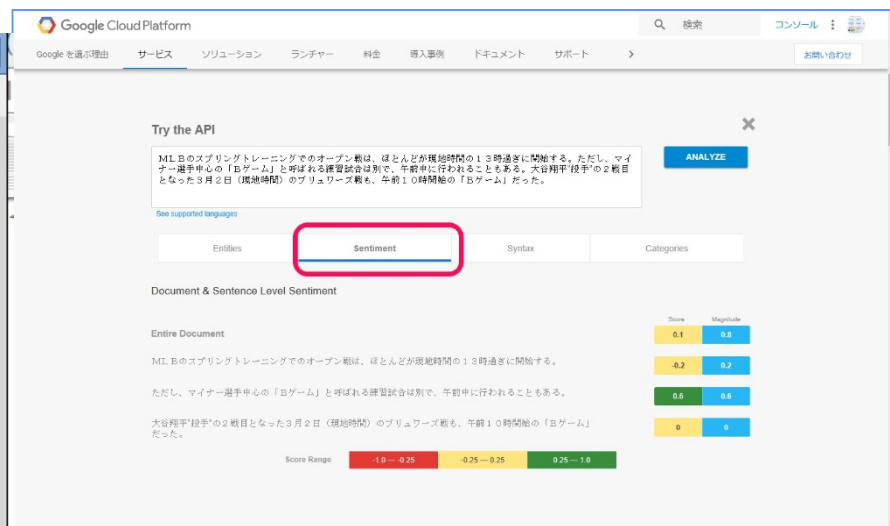
コマンド

```
$ cd
/home/pi/Documents/gcp_api/cloud_nl/
$ python3 senti1.py
```

Webでの確認

<https://cloud.google.com/natural-language/?authuser=1&hl=ja>

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
listen_print_loop(responses)
File "/usr/local/lib/python2.7/dist-packages/google/api_core/grpc_helpers.py", line 54, in error_remapped_callable
return callable(*args, **kwargs)
File "/usr/local/lib/python2.7/dist-packages/google/api_core/grpc_helpers.py", line 332, in _next
self._state.condition.wait()
File "/usr/lib/python2.7/threading.py", line 340, in wait
KeyboardInterrupt
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl# cd /home/pi/Documents/gcp_api/cloud_nl/
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl# ls
classify1.py hello_world.py senti2.py test_sentence.txt
entity1.py senti1.py syntax1.py test_sentence_en.txt
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl# python senti1.py
Score: 0.10000000149
Magnitude: 0.800000011921
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_nl#
```



クラウドAI開発キット

4.使用方法

⑤ Cloud video

- analyze.py (複数動画認識)

- analyze.pyを実行します。複数のコマンドがあります。Google storage上のファイルで動作します。

\$ python3 analyze.py labels gs://バケット名/ファイル名.mp4 ラベル認識

\$ python3 analyze.py explicit_content gs://バケット名/ファイル名.mp4 不適切動画認識

```
コマンド
$ cd
/home/pi/Documents/gcp_api/cloud_video/
$ python3 analyze.py explicit_content gs://
バケット名/ファイル名.mp4
```

```
pi@raspberrypi: ~
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
Time: 143.530829s
pornography: Very unlikely
root@raspberrypi: /home/pi/Documents/gcp_api/cloud_video# python analyze.py explicit_content gs://a.../.../7.mp4
Processing video for explicit content annotations:
Finished processing.
Time: 0.594403s
pornography: Very unlikely
Time: 1.719093s
pornography: Very unlikely
Time: 2.575706s
pornography: Very unlikely
Time: 3.616114s
pornography: Very unlikely
Time: 4.798984s
pornography: Very unlikely
Time: 5.694583s
pornography: Very unlikely
Time: 6.571351s
pornography: Very unlikely
Time: 7.46716s
pornography: Very unlikely
```

```
デスクトップ#murakami_jeno#ビジネス#技術関係#deep_learning#gcp_api/cloud_video#analyze#analyze.py - sakura 2.2.0.1
Copyright 2017 Google Inc. All Rights Reserved.
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
This application demonstrates face detection, label detection,
explicit content, and shot change detection using the Google Cloud API.
Usage Examples:
python analyze.py faces gs://demomaker/google_gmail.mp4+
python analyze.py labels gs://cloud-ml-sandbox/video/chicago.mp4+
python analyze.py labels_file resources/cat.mp4+
python analyze.py shots gs://demomaker/gbikes_dinosaur.mp4+
python analyze.py explicit_content gs://demomaker/gbikes_dinosaur.mp4+
利用例: 顔認証は動作しません
import argparse+
import io+
from google.cloud import videointelligence+
def analyze_explicit_content(path):+
    """ Detects explicit content from the GCS path to a video. """+
    video_client = videointelligence.VideoIntelligenceServiceClient()+
    features = [videointelligence.enums.Feature.EXPLICIT_CONTENT_DETECTION]+
```

クラウドAI開発キット

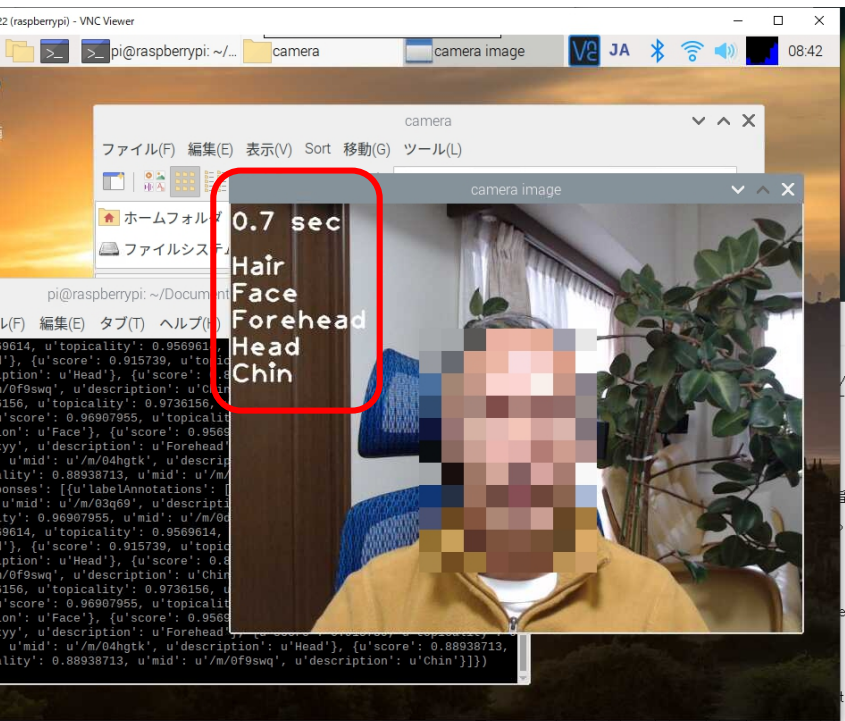
4.使用方法

⑤ Cloud video

- object_cam1.py (リアルタイム・カメラによる物体認識)
 - Webカメラで映ったラベルを検出します。
 - 16行目にAPIキー
 - RaspberryPi4Bで高速処理が可能

```
コマンド
$ cd
/home/pi/Documents/gcp_api/cloud_video/
$ python object_cam1.py
```

python3でエラーのためpythonで実施してください



```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Google Cloud Vision API: LABEL_DETECTION
import sys
import base64
import cv2
from requests import Request, Session
import json
import time
import threading

api_key = 'A1zaSyDpkR'

# 検出する類の数の最大数 (増やすとレスポンスが遅くなる)
max_results = 5

# END POINTS
DISCOVERY_URL = 'https://vision.googleapis.com/v1/images:annotate?key='

# cv画像と画像ファイルへのPathと検出最大数が引数
def googleAPI(img, max_results):
    # 通信不良等を考慮してTry/exceptしておく
    try:
        # カメラ画像をJPG画像へ変換
        retval, image = cv2.imencode('.jpg', img)

        # Headerとpayload
        str_headers = {'Content-Type': 'application/json'}
        batch_request = {'requests': [{'image': {'content': base64.b64encode(image)}, 'features': [{'type': 'LABEL_DETECTI

        # セッション作ってリクエストSend
        obj_session = Session()
        obj_request = Request("POST", DISCOVERY_URL + api_key, data=json.dumps(batch_request), headers=str_headers)
        obj_prepped = obj_session.prepare_request(obj_request)
        obj_response = obj_session.send(obj_prepped, verify=True, timeout=180)

        # Responseからjsonを抽出
        response_json = json.loads(obj_response.text)
```

Opencv開発

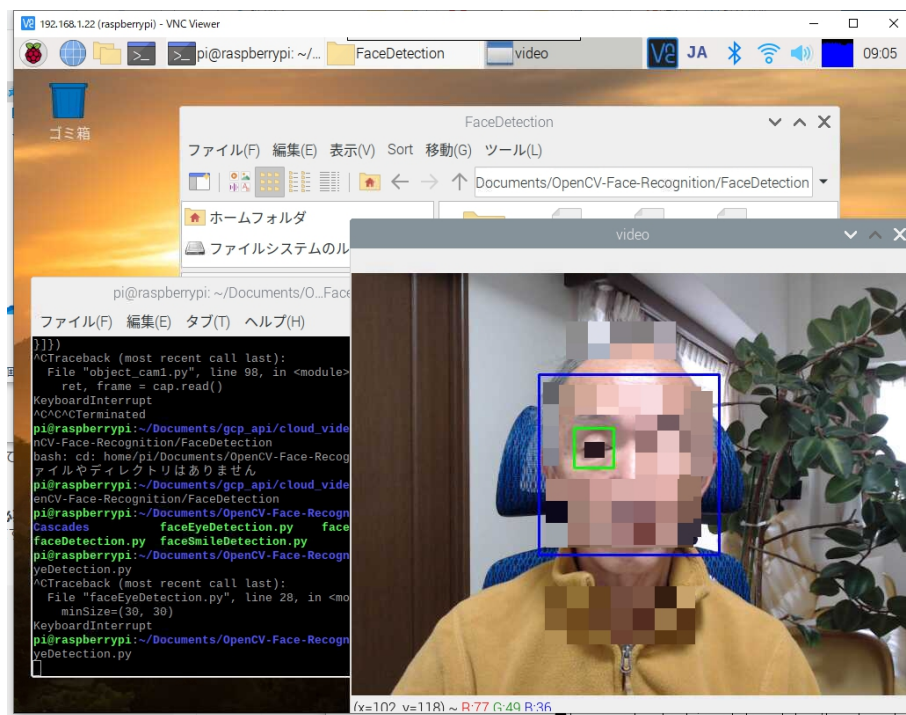
1. Opencv

①顔認識(カメラ)

コマンド

```
$ cd /home/pi/Documents/OpenCV-Face-Recognition/FaceDetection
```

- Opencvを使ったwebカメラでの顔認識です。
- <https://www.instructables.com/id/Real-time-Face-Recognition-an-End-to-end-Project/>
- python3 faceDetection.py
- python3 faceEyedetection.py



```

pi@raspberrypi: ~/Documents/O...FaceDetection
File "object_cam1.py", line 98, in <module>
    ret, frame = cap.read()
KeyboardInterrupt
^C^C^C^CTerminated
pi@raspberrypi: ~/Documents/gcp_api/cloud_video
nCV-Face-Recognition/FaceDetection
bash: cd: /home/pi/Documents/OpenCV-Face-Recognition/FaceDetection: Not a directory
pi@raspberrypi: ~/Documents/gcp_api/cloud_video
nCV-Face-Recognition/FaceDetection
pi@raspberrypi: ~/Documents/OpenCV-Face-Recognition/FaceDetection
python3 faceEyeDetection.py
python3 faceDetection.py
pi@raspberrypi: ~/Documents/OpenCV-Face-Recognition/FaceDetection
python3 faceEyeDetection.py
^C^C^C^CTerminated
pi@raspberrypi: ~/Documents/OpenCV-Face-Recognition/FaceDetection
python3 faceEyeDetection.py

```




Opencv開発

1. Opencv

②顔検出(カメラ)

コマンド
\$ cd /home/pi/Documents/OpenCV-Face-Recognition/FacialRecognition

- Opencvを使った、webカメラでの登録した顔を検出します。
- <https://www.instructables.com/id/Real-time-Face-Recognition-an-End-to-end-Project/>
- 顔の登録
- python3 01_face_dataset.py
 - Face id:0-3までを入力してください。
 - 自動で顔のスクランが始まります。数分で終了します。
- python3 02_face_training.py
- python3 03_face_recognition.py
 - カメラで認識します、名前は26行目で変更可能

```
1 Real-time Face Recognition -
2 Rea...=> Each face stored on dataset/ dir, should have a unique numeric integer ID as 1, 2, 3, etc
3 # Add original code by Anirban Kar: https://github.com/theodacus/Face-Recognition
4
5 Developed by Marcelo Rovali - MJRoBot.org @ 21Feb18
6
7 ...
8
9
10
11 import cv2
12 import numpy as np
13 import os
14
15 recognizer = cv2.face.LBPHFaceRecognizer_create()
16 recognizer.read("trainer/trainer.xml")
17 cascadePath = "haarcascade_frontalface_default.xml"
18 faceCascade = cv2.CascadeClassifier(cascadePath);
19
20 font = cv2.FONT_HERSHEY_SIMPLEX
21
22 # Initiate id counter
23 id = 0
24
25 # names related to ids: example => Marcelo; id=1, etc
26 names = ['None', 'Masa', 'Paula', 'Ilza', 'Z', 'M']
27
28 # Initialize and start realtime video capture
29 cam = cv2.VideoCapture(0)
30 cam.set(3, 640) # set video width
31 cam.set(4, 480) # set video height
32
33 # Define min window size to be recognized as a face
34 minW = 0.1*cam.get(3)
35 minH = 0.1*cam.get(4)
36
37 while True:
38
39     ret, img = cam.read()
40     #img = cv2.flip(img, -1) # Flip vertically
41
42     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
43
44     faces = faceCascade.detectMultiScale(
45         gray,
46         scaleFactor = 1.2
```

