機能設計書

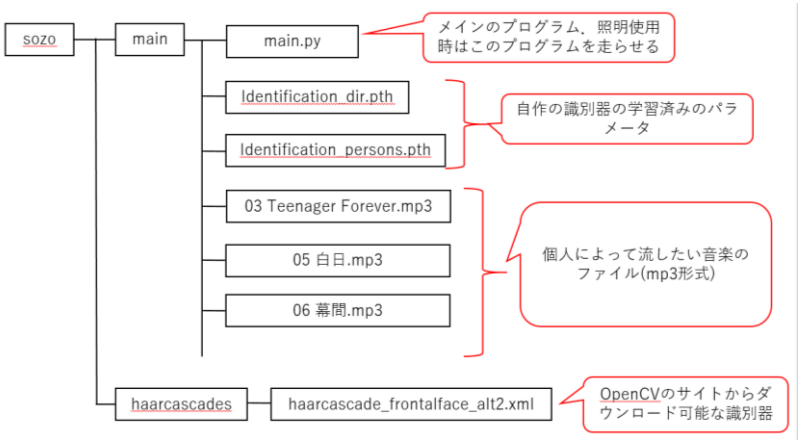
作成者：学籍番号 20073　氏名　所京太朗
確認者：学籍番号 20089　氏名　前川汰輝

班名：　A 班　　商品名；ドラゴンライト

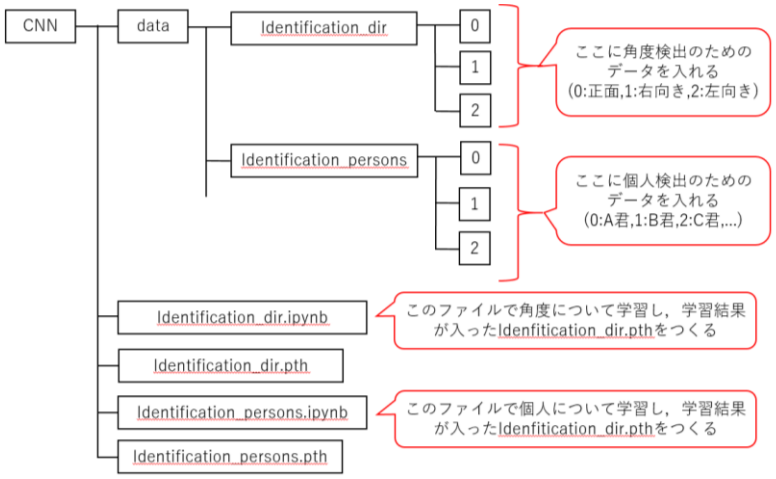
担当した機能要素が複数あるため，以下の順番で記載する．

1. 照明の自動点灯
2. 照明の方向調整
3. 人物による BGM 変更

照明を動かすのに必要なファイルを以下のように配置する．main フォルダは”https://drive.google.com/drive/folders/1xKx4tVXwfh_yu-bHNu-xraHEYZ9SnZMK?usp=sharing”に公開する．ただし，権利の問題に対処するために BGM のファイルは公開しない．



自作した識別器を学習するプログラムを含むファイルは”https://drive.google.com/drive/folders/1xkf4GT3o2n6IO5qkFX1oE2P0jMMypeeG?usp=sharing”に公開する．ただし，データは班員の顔写真となっており，公開するのははばかれるのでデータを入れるファイルは空にしてある．



1. 機能要素名（ブロック名）：照明の自動点灯（ソフトウエア）

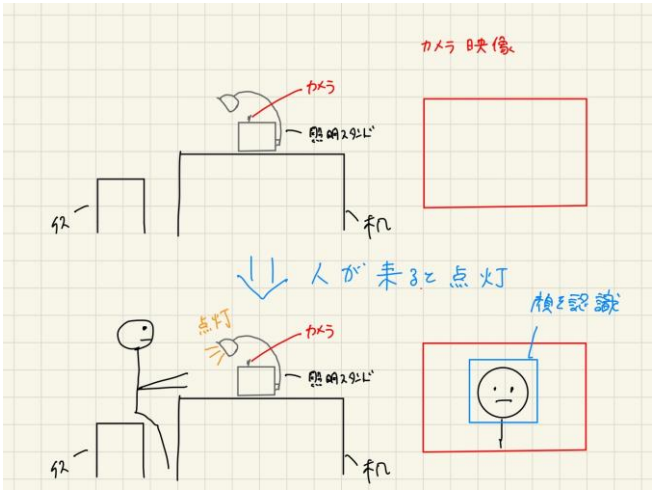
(1) 狙いの主機能・性能

「作業中でも操作可能な照明スタンド」というコンセプトを達成するために，照明の点灯・消灯を自動で行えるようにする．照明の点灯・消灯は，机で作業をしているか否かによって変更したいため，以下の2点を狙いの主機能とする．

(1)人が照明スタンドの前に座ると点灯させ，前にいる間は点灯をさせ続ける．

(2)人が照明スタンドの前からいなくなると消灯させ，いない間は消灯させ続ける．

(2) 要図（図で説明：手書きの図を貼付可、商品全体のどの部位か、どの動作（ソフトウエア）かわかるように）



照明自動点灯のイメージ図

カメラで照明スタンド前方の画像を取得し，人の顔が検出されれば明かりをつけ，検出されなければ明かりを消す．

(3) 要求される働き

要求される働きは(1)で示した二点である．

①人が照明スタンドの前に座ると点灯させ，前にいる間は点灯をさせ続ける．

②人が照明スタンドの前からいなくなると消灯させ，いない間は消灯させ続ける．

(4) 保証すべき項目とレベル、その理由

(3)で示した2点を保証すべき項目とし，そのレベルと理由を示す．

①人が照明スタンドの前に座ると点灯させ，前にいる間は点灯をさせ続ける．

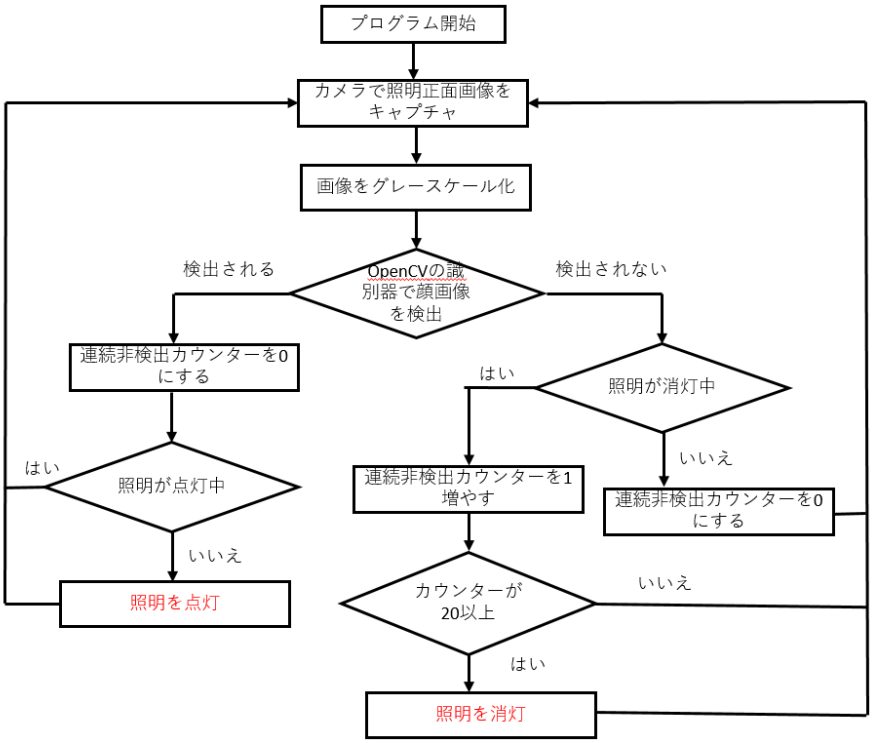
着席と同時に作業を始めたいため，人が映ったらできるだけ早く照明を点灯させる，また，作業中に顔の誤検出により照明がついたり消えたりするのはストレスがたまるので，顔の誤検出による誤作動は起こらないようにする．

②人が照明スタンドの前からいなくなると消灯させ，いない間は消灯させ続ける．

人がいなくなった場合は電力消費を抑えるために消灯させる．ただし，物を取るためにちょっと体勢を崩し顔が検出されなくなっただけで消灯されては不便であるため，一定期間顔が検出されない場合は消灯するようにする．一定期間は約 3 秒と設定したが，変更は容易であるため説明書などを使いユーザーが任意で設定できるようにすることができる．

(5) 設計根拠（含：設計式と設計値、安全率、フローチャート、実験結果、引用技術方式、従来例など）

main.py ファイルにこの機能を達成するプログラムを書いた．プログラムの流れは以下である．



照明自動点灯を実現するためのフローチャート

顔検出の識別器は OpenCV に用意されている顔検出をするカスケード識別器”haarcascade_frontalface_alt2.xml”を用いた(参考[1]より取得できる)．カスケード識別器は複数の識別器で構成されており，それぞれの識別器で判断基準を緩くしているため，非検出の場合は高速に判定される．そのため，リアルタイムの識別に用いるのに適している．(参考[2])

ライトの制御は gpiod モジュールによって行った．main.py の 211-220 行目で LED_onoff(x)という関数を定義し，ライトの on/off をこの関数によって変更できるようにした．(x=0 とすると消灯，x=1 とすると点灯)．

誤検出による誤作動を防ぐために，複数枚フレームの判定結果によって照明を消灯するべきかどうかを判定する．具体的には，連続して 20 フレームで顔が検出されなかった場合に消灯させる．この方法で，どれほどの誤検出が防げるかを考える．仮に 10%の割合で人がいるのにいないと判定する誤検出が起こ

るとする．その場合，誤って消灯してしまう確率は 1.0×10^{-18} %であり，ほとんど起こらないと考えられる．

(参考)

[1] https://github.com/opencv/opencv/tree/master/data/haarcascades
[2] https://algorithm.joho.info/programming/python/opencv-haar-cascade-face-detection-py/

(6) 確認すべき項目（試作で確認すべき項目と判定基準及び測定方法）

まず，人がいなくなってから何秒で消灯するかを計測した．設定としては上で述べたように 20 フレームと設定したが，これが実際に何秒なのかは他のプログラムの動作時間によって変わってくる．計測すると，平均 2.8 秒であり，目標としていた 3 秒に近い結果が得られた．

次に誤検出による誤動作が起こる頻度を計測する．起こりうる誤動作として，人がいないのに点灯する誤動作と，人がいるのに消灯する誤動作が考えられる．後者は上で述べたように理論上誤る確率は低く，実際に 1 分間測定した結果誤動作は 0 回であった．前者については，5 分間，顔が映らないようにして様々な正面映像を映し，何回誤動作が起こるかという実験を行った．結果 1 回だけ誤動作が起きた．

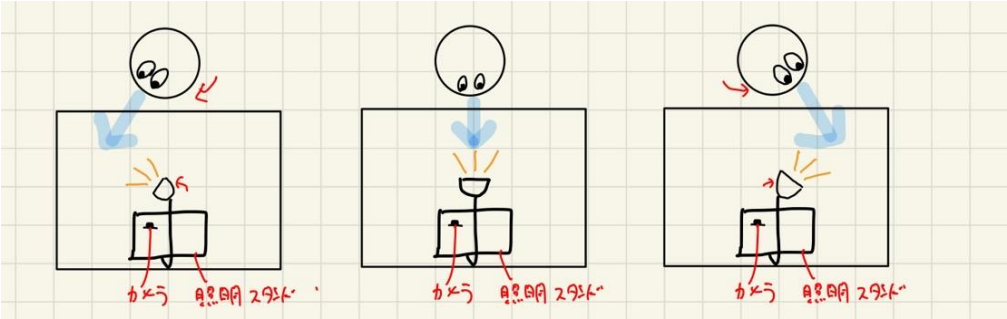
(次ページに続く)

2. 機能要素名（ブロック名）：照明の方向調整（ソフトウエア）

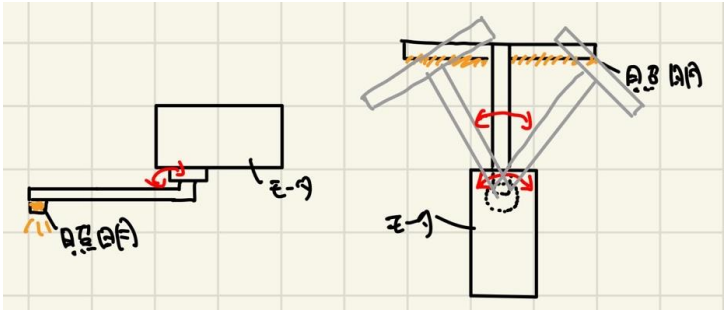
(1) 狙いの主機能・性能

「作業中でも操作可能な照明スタンド」というコンセプトを達成するために，照明の方向調整を自動で行えるようにする．照明で照らしたい場所は視線の先であるので，顔の向きによってライトの照らす方向を変更する．照明の変更可能角度は正面，右，左の三方向とする．

(2) 要図（図で説明：手書きの図を貼付可、商品全体のどの部位か、どの動作（ソフトウエア）かわかるように）



照明の方向調整のイメージ図



モーターの動きと照明の動きの関係

カメラで照明スタンド前方の画像を取得し，人の顔がどの方向に向いているかを判定し，モーターを制御することで照明を人が向いている方向に向ける．モーターは LEGO モーターを使用し，Raspberry-Pi Build Hut の portA に直接つなぐ．

(3) 要求される働き

要求される働きは，精度よく人がどの方向を向いているかを推定し，その推定結果に対してライトを以下のように動かすことである．

①正面を向いているときは，正面を照らす
②右を向いているときは，右側を照らす
③左を向いているときは，左側を照らす

④人がいないときは，正面に向ける

精度よく人が向いている向きを推定するには，OpenCV などで用意されている既存の識別器では無理だと判断し，精度の良い識別器を自分たちで用意することにした．

(4) 保証すべき項目とレベル、その理由

向いている方向を照らしてくれなかったり，動作が不安定であったりするとユーザーは不便さを感じるので，方向推定は精度良く行う必要があると考える．そのため，方向推定は 95%以上の精度を達成するべきである．

照らしてほしい位置を照らせるようにモーターの回転角度を調整するべきである．

長時間の使用をしてもモーターの角度にずれが生じないように，モーターの角度や速さを調整するべきである．

(5) 設計根拠（含：設計式と設計値、安全率、フローチャート、実験結果、引用技術方式、従来例など）

まず，正面，右向き，左向きについて定義した．どの位置から照明の向きが変わってほしいかを話し合い，利用者と照明スタンドの直線上を 0 度として 50 度以上顔を傾けた場合を右/左向きであると定義した．

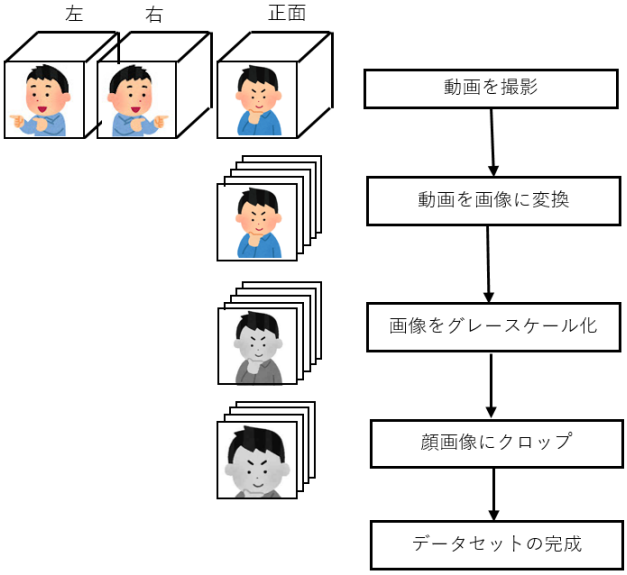
精度よく人が向いている向きを推定するには，OpenCV などで用意されている既存の識別器では無理だと判断し，識別器を自分たちで構築することにした．そのため，(i)データセットの作成，(ii)モデル構築と学習，(iii)構築したモデルを利用したモーター制御の流れの順で説明する．

(i)データセットの作成

データは班員 5 人の画像を用いた．

まず，今回は正面，右，左の三つに分類したいので，その 3 パターンに分けて各 30 秒の動画を撮影する．その際に，範囲内できるだけ顔を動かしたり，表情を変えたりして様々なデータを取れるようにした．

撮影した動画から，1 秒当たり 5 枚の画像に変換する．30 秒の動画から 150 枚の画像が得られる．それをグレースケール変換し，ライト制御で用いた OpemCV の識別器を使って顔画像をクロップする．それらを合わせたものをデータセットとした．



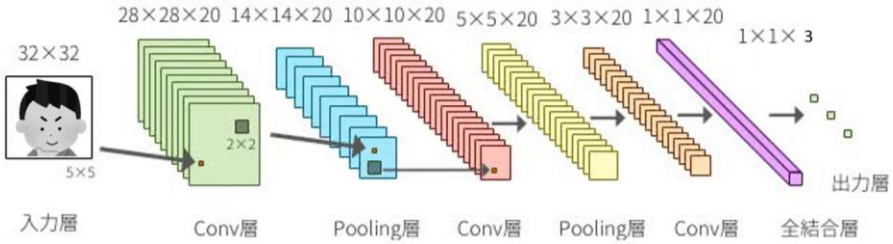
データセット作成の流れ

理想的には 150 枚×3(パターン)×5(人)=2250 枚のデータ画像が得られるはずであったが，OpenCV で顔を検出できない画像が多かったため，約 1500 枚のデータセットになった．

(ii)モデルの構築と学習

モデルの構築と学習について説明する．これらは 1 ページ目で挙げた CNN フィルダの Identification_dir.ipynb ファイルの内容である．この学習は Raspberry-Pi では重たいと感じたので，google colaboratory 上で GPU を使用して行った．

精度が求められているので，pytorch を使い，深層学習によりモデルを構築することにした．いろいろな構造を試した結果, 精度とモデルの軽さを両立するために以下の構造にすることにした. これは ResNet を参考にした(参考[1]).



方向推定のための識別器の構造

データセットは，正面向きの画像が右向きと左向きに比べ多かったのでアンダーサンプリングを行い，合計 1053 枚のデータを用いた．このうち 400 枚を検証データとした．また，重くならないように，データを 32×32 に変換した．

学習結果は，訓練データでも検証データでも accuracy が 98%を超えることができた．

学習したパラメータを Identification_dir.pth として保存し，これを Raspberry-Pi に送ることでこの学習結果を利用できる．

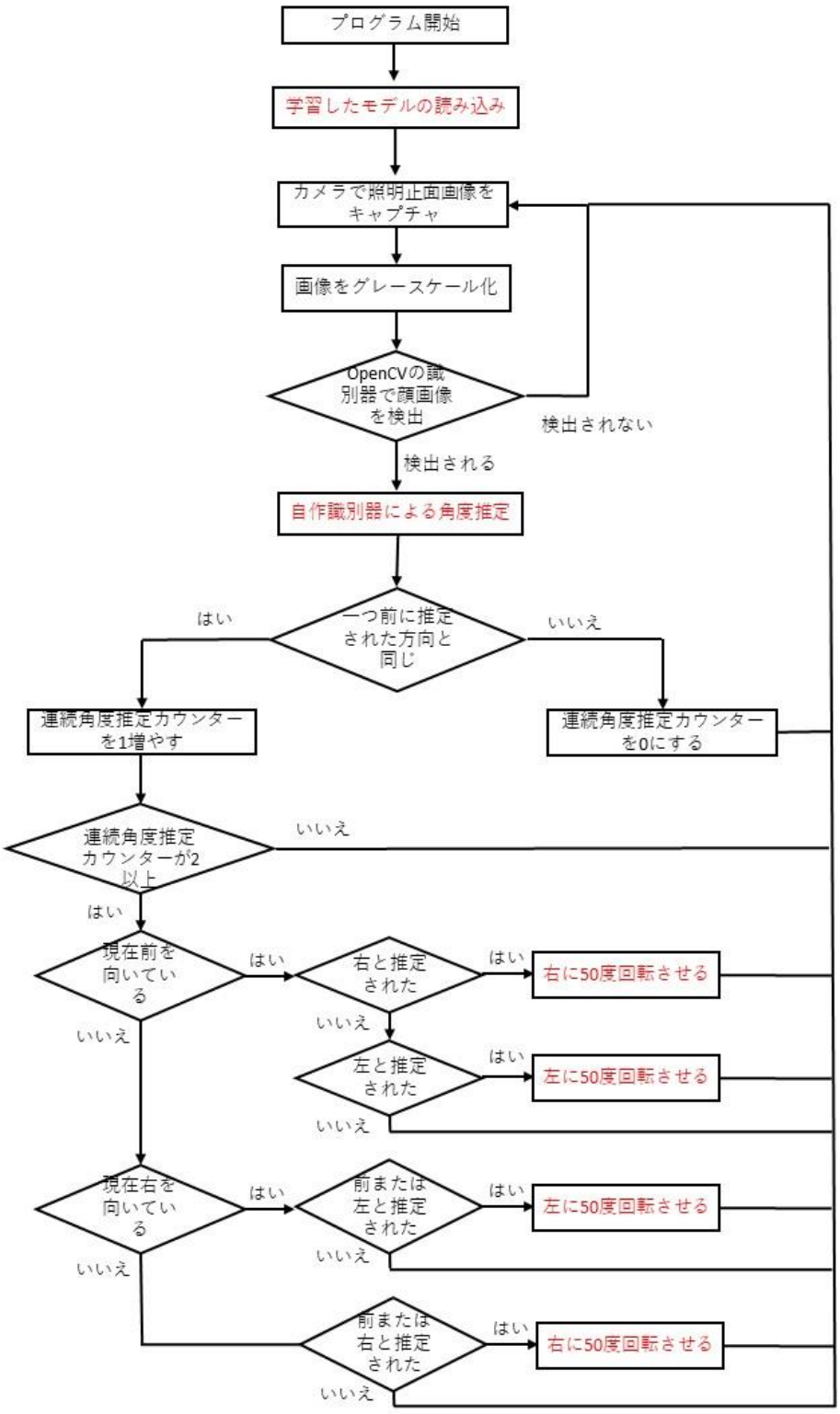
(iii) 構築したモデルを利用したモーター制御の流れ

main.py ファイルにこの機能を達成するプログラムを書いた．プログラムの流れを次ページに示す．

まず，(ii)で得られた学習結果を用いて，識別器を構築する．その後，カメラから得られた画像をデータセットと同様の変換を行い，識別器に入れ，方向推定結果を得る．その結果から以下のフローチャートに従ってモーターを制御する．

モーターの制御は buildhat パッケージの Motor モジュールによって行った．main.py の 224-228 行目で Motor (degrees)という関数を定義し，モーターの角度をこの関数によって変更できるようにした．(degrees =50 とすると照明は右に 50 度傾く)．

誤検出による誤作動を防ぐために，複数枚フレームの判定結果によって照明の動かす方向を決定する．しかし，照明の on/off の時と違い，そもそも識別器の accuracy が 98%を超えているので，2 フレームの結果を反映するだけで誤った方向に向いてしまう確率は 0.04%になる．よって 2 フレーム連続して同じ推定結果が得られた時モーターを動かすようにした．



照明方向変更のフローチャート

(参考)

[1] https://deepage.net/deep_learning/2016/11/30/resnet.html

(6) 確認すべき項目（試作で確認すべき項目と判定基準及び測定方法）

連続してモーターを動かすことで，実際のモーターの角度と理論値がずれてこないかを確認するために，右左を 10 往復させ，最終的に 0 度に戻ってくるかを試した．結果はずれが 5 度以下であり，十分な精度であると言えた．

顔の角度を変えてから何秒でライトの向きが変わるかを測定した．測定結果は約 1.5 秒であり，ある程度反応は速く，しかしちょっとした動作の変化には反応しない動作時間だといえる．

次に誤検出による誤動作が起こる頻度を計測する．上で述べたように識別器の accuracy が非常に高く，実際に使用してもほとんど誤作動は観察できなかった．

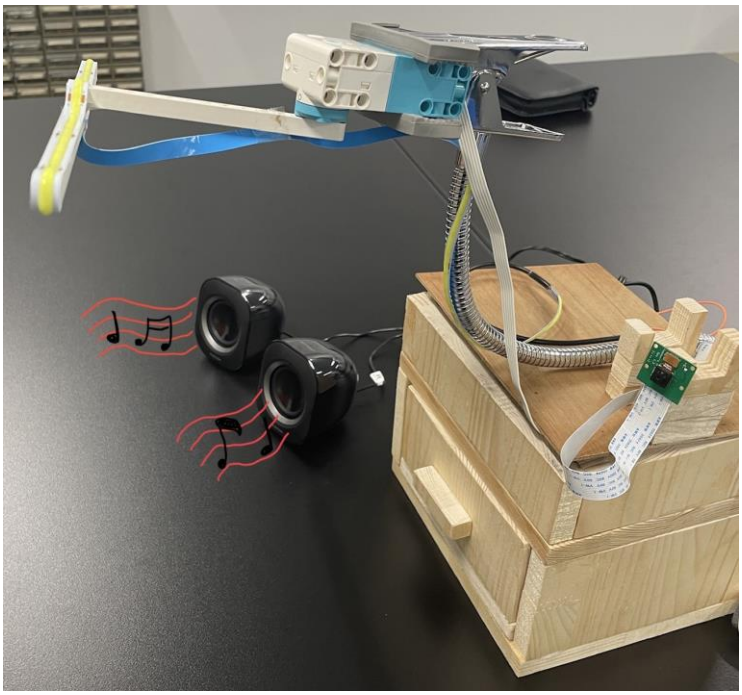
(次ページに続く)

2. 機能要素名（ブロック名）：人物による BGM 変更（ソフトウエア）

(1) 狙いの主機能・性能

設定をしなくても理想的な勉強環境を自動で作るために，座っている人に合わせて BGM を流す機能をつける．人に合わせた BGM とは，事前にその人が設定して置いた音楽のことである．この機能を付けることで，学校などの公共の場所や兄弟間などで共有しても，個人に合った勉強環境をつくることができる．

(2) 要図（図で説明：手書きの図を貼付可、商品全体のどの部位か、どの動作（ソフトウエア）かわかるように）



照明スタンドとスピーカー

カメラで照明スタンド前方の画像を取得し，だれが前にいるのかを判定し，その人が事前に設定していた音楽を流す．スピーカーは用意されていたものを使い，イヤホンジャックと USB を Raspberry-Pi に直接つないだ．

(3) 要求される働き

要求される働きは，精度よく使用者が誰かを推定し，その推定結果によって流す音楽を切り替える．

利用者が誰なのかを判定するためには，利用者のデータによって学習した識別器が必要になってくる．そのため，事前に利用者になりえる人の画像(動画)を撮影し，また利用者がそれぞれ流したい音楽を設定しておく必要がある．

(4) 保証すべき項目とレベル、その理由

設定した音楽が流れなかったり，音楽が途切れ途切れになったりするとユーザーは不便さを感じるので，個人推定は精度良く行う必要があると考える．そのため，個人推定は 95%以上の精度を達成するべきである．

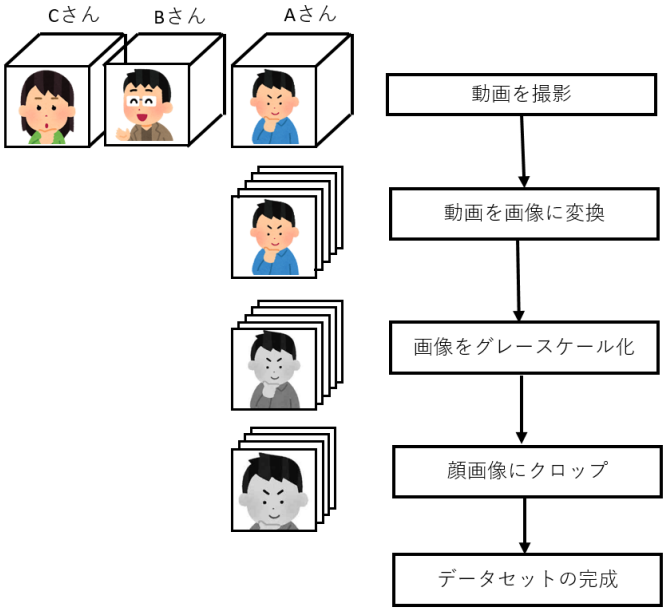
(5) 設計根拠（含：設計式と設計値、安全率、フローチャート、実験結果、引用技術方式、従来例など）

ユーザーに特化した機能を付けるために，識別器を自分たちで構築することにした．そのため，(i)データセットの作成，(ii)モデル構築と学習，(iii)構築したモデルを利用したモーター制御の流れの順で説明する．

(i)データセットの作成

データは班員 5 人の画像を用いた(5 人を班物できるようにした). 1 人当たり 90 秒の動画を撮影する．その際に，範囲内できるだけ顔を動かしたり，表情を変えたりして様々なデータを取れるようにした．

撮影した動画から，1 秒当たり 5 枚の画像に変換する．90 秒の動画から 450 枚の画像が得られる．それをグレースケール変換し，ライト制御で用いた OpemCV の識別器を使って顔画像をクロップする．それらを合わせたものをデータセットとした．



データセット作成の流れ

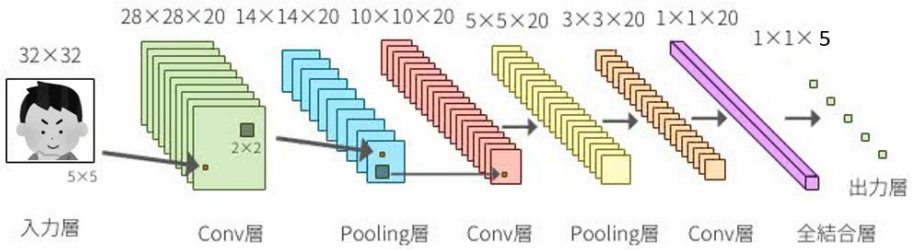
理想的には 450 枚×5(人)=2250 枚のデータ画像が得られるはずであったが，OpenCV で顔を検出できない画像が多かったため，約 1500 枚のデータセットになった．

実際に使い場合には，音楽を設定する際に音楽を指定するのと同時にその人の画像を取り，そのたびに学習しなおす必要がある．

(ii)モデルの構築と学習

モデルの構築と学習について説明する．これらは 1 ページ目で挙げた CNN フィルダの Identification_persons.ipynb ファイルの内容である. この学習は Raspberry-Pi では重たいと感じたので, google colaboratory 上で GPU を使用して行った.

精度が求められているので，pytorch を使い，深層学習によりモデルを構築することにした．いろいろな構造を試した結果, 精度とモデルの軽さを両立するために以下の構造にすることにした. これは ResNet を参考にした(参考[1]).



個人推定のための識別器の構造

データセットは, 人によるデータ数のばらつきはあったがそれほど問題になる差ではないと考え, 合計 1517 枚のデータを用いた．このうち 500 枚を検証データとした．また，重くならないように，データを 32×32 に変換した.

学習結果は，訓練データでも検証データでも accuracy が 97%を超えることができた.

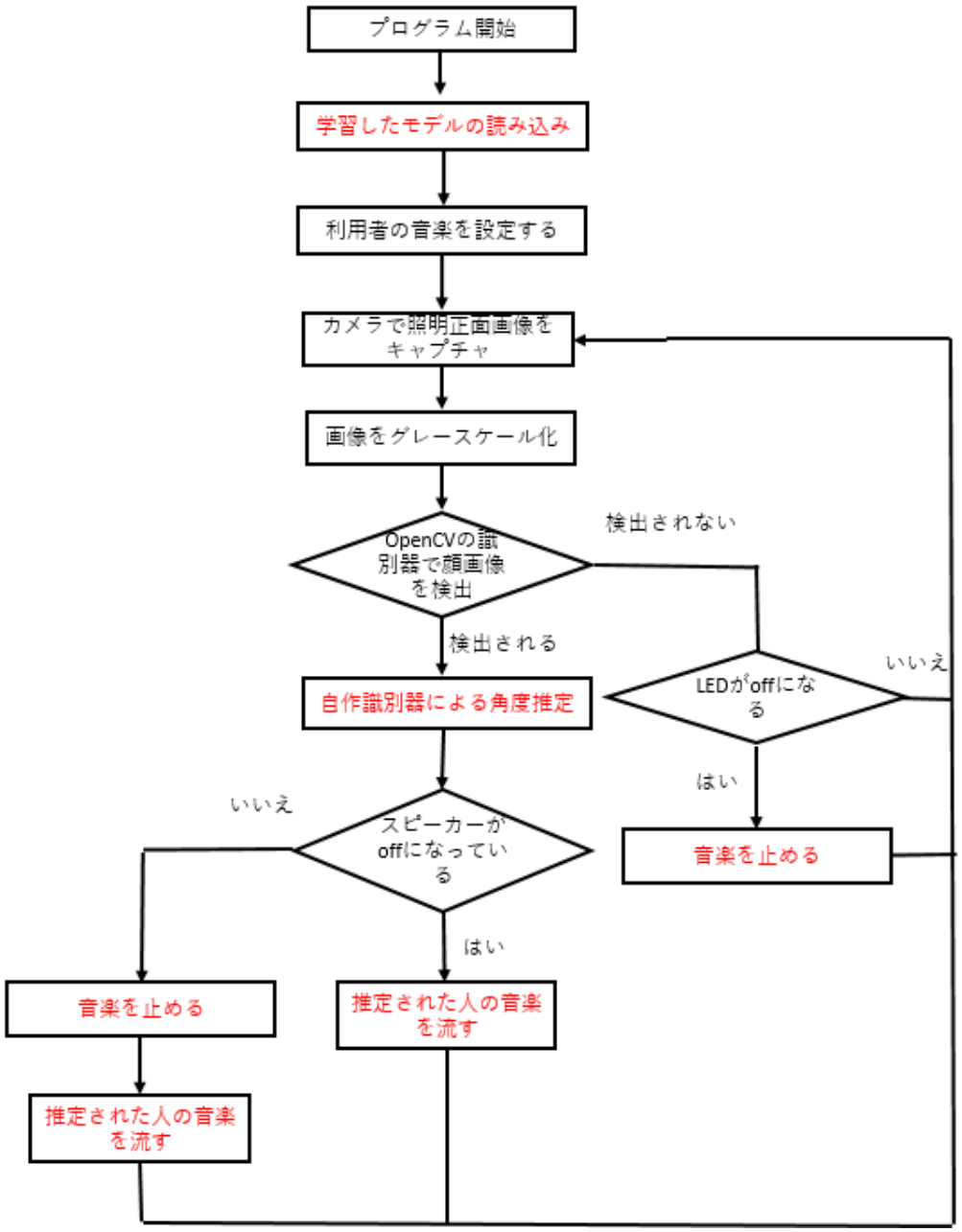
学習したパラメータを Identification_persons.pth として保存し，これを Raspberry-Pi に送ることでこの学習結果を利用できる.

(iii) 構築したモデルを利用したスピーカー制御の流れ

main.py ファイルにこの機能を達成するプログラムを書いた．プログラムの流れを次ページに示す.

まず，(ii)で得られた学習結果を用いて，識別器を構築する．その後，カメラから得られた画像をデータセットと同様の変換を行い，識別器に入れ，個人推定結果を得る．その結果から以下のフローチャートに従ってスピーカーを制御する.

スピーカーの制御は pygame.mixer モジュールによって行った(参考[2]). スピーカーを on にするタイミングは個人推定ができたときで，off にするときはライトを消灯するのと同じタイミングで行う.



BGM 変更のフローチャート

(参考)

[1] https://deepage.net/deep_learning/2016/11/30/resnet.html

[2] https://shizenkarasuzon.hatenablog.com/entry/2019/02/24/090652

(6) 確認すべき項目（試作で確認すべき項目と判定基準及び測定方法）

誤検出による誤動作が起こる頻度を計測する．上で述べたように識別器の accuracy が非常に高く，実際に使用してもほとんど誤作動は観察できなかった．