F-015ドローンを用いた牡蠣筏メンテナンスシステムのための自律飛行試験

機械工学科　1201709 奥迫　樹
機械工学科　1201719 清水　一世
（指導教員　吉川　祐樹）

１．緒言

近年ドローンの開発は進み世界的に普及し始めた.「最後の産業革命」と呼ばれるほど様々な業界から注目されており,空撮・物流・農業などで用いられている.ドローンは遠隔操作や自立飛行により,人が足を運ぶよりも簡単かつ安全に離れた物に接近できるため,対象物の観察における手間やリスクを解消するためのツールとして使用され始めている.

広島の牡蠣養殖業の方々は,海上数キロの範囲に複数の列に並んで浮かぶ牡蠣筏(写真 1)を船で回り,養殖作業や牡蠣筏メンテナンスを行っている.この業務において,船での移動による手間や燃料代が負担となっている.また,豪雨や高波など悪天候の後には，牡蠣筏の漂流や破損がないか確認する必要があり，悪天候が続く場合には危険を伴う.そこでドローンが自律飛行しながら各筏の状態を自動で撮影できれば，牡蠣養殖業の業務において時間や燃料代の削減,及び安全性の向上が期待できる.

これまで我々の研究室では,MATLAB を用いて画像認識を行い，MATLAB のアドオンである Simulink でドローンの自律飛行を制御するアルゴリズムを提案する研究を行ってきたが,飛行試験は Simulink のシミュレーション機能による疑似的なものであった．市販のドローンには,MATLAB と連携して実際に飛行試験を行える実機は存在するが,連携に必要な機材の入手や連携が難しい.故に,実際にドローンを用いた飛行試験が行えなかったのである.また,MATLAB と連携できるドローン実機も,現在も国内における製品サポートが無い.この研究の将来的な目標は,開発した支援システムを牡蠣養殖業の方々への普及・実用する事であるため,これらの問題点は致命的な問題であると考える.一方,現在国内で最も普及しているプログラミング可能なドローンは Tello であり,国内でのサポートも充実している.また,使用する言語は Python である.この言語は機械学習が可能であり,ドローンとの連携など様々な機能が無料で提供されている.言語を理解する難易度も比較的低く,これらのポテンシャルから国内において言語としての注目度が高いため,将来的にも更なるサポートの向上が期待できる.以上の理由から,Python 及び Tello を用いてシステム開発をする価値は大きいと考えた.

よって本研究では,プログラミング言語 Python を用いて牡蠣筏の機械学習を行い,PC と Tello を連携して,図 1 のように画像認識による牡蠣筏の自動追跡と写真回収を行い,帰還する支援システムを開発する.更に,実際に Tello を用いて開発したシステムの飛行試験を行い,その挙動からシステムを評価するまでを目標とする.

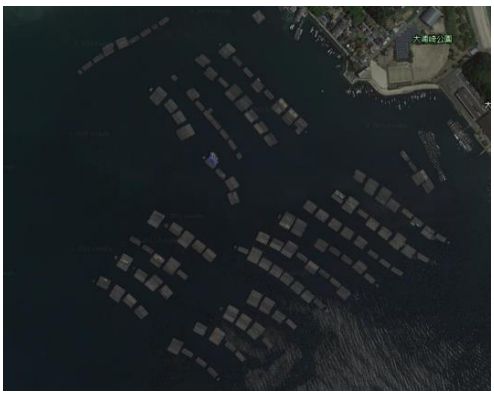

写真 1 海に並んで浮かぶ牡蠣筏

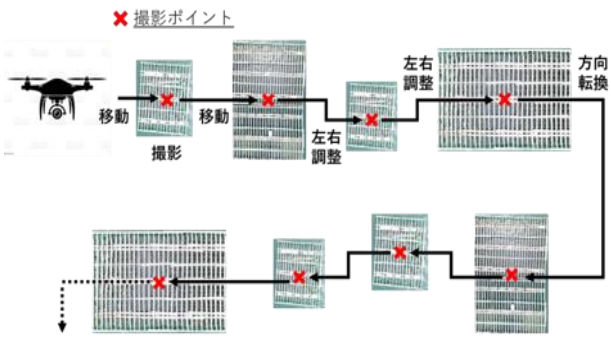

図 1 筏の自動追跡と写真回収のプロセス

２．開発方法

先述した目標を実現するためには,要求される課題が 3 つある.

(1)機械学習による牡蠣筏の画像認識
(2)ドローンのフライト制御システム
(3)牡蠣筏の画像を提供する web アプリケーション

これらの課題の開発方法を踏まえながら論述する.

2.1 機械学習による牡蠣筏の画像認識

(1)で述べた機械学習とは，大量の画像をコンピューターに読み込ませ，取り入れられたアルゴリズムに基づいて分析を行い学習させる手法である．Python での機械学習は OpenCV を用いて行う.OpenCV とは，画像処理，画像認識および機械学習等の機能を持つライブラリであり,Python，C/C++，Java，MATLAB など，様々なプログラミング言語で広く用いられている．本研究では機械学習の際に正解データを与えた状態で学習させる,教師あり学習と呼ばれる手法を用いる.画像の中から対象物を識別する画像認識において,対象物は特定の画素値の羅列であるため,明確に正解データを学習させる事が可能な教師あり学習は最適であるとされている．教師あ

り学習のメリットとしては,人間が正解となるデータを与えるため学習精度が高く,学習速度も早いことが挙げられる.学習に用いられる正解データが多いほど学習精度は高くなる傾向にある． 反対にデメリットとしては,正解が存在しない分野には利用できないことや,正解となるデータの質が学習精度に影響を与えることである.本研究では,正解データには簡易的な筏の画像を使用する.(図 2)また,機械学習には大量の画像を用意しなければならない.本研究では,OpenCV に備わっている数枚の画像データを数千枚の画像データに複製する createsample というコマンドと，用意した大量の正解画像と不正解画像をもとに学習を実行する.traincascade というコマンドを利用して大量の正解画像を用意する.
createsample は，画像を複製する際に元の画像をランダムに形状変化させることで，画像の特徴量の抽出をより効率的にしている．（図 3）このプログラムを用いることで大量の画像データの作成の手間を省くことができる．このプログラムを用いて，新たに作成した 3 枚の画像データを 2400 枚に複製し，学習用の正解画像データとして用いて機械学習を行う.

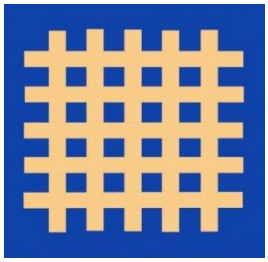

図 2 簡易的な筏の画像

本研究では,機械学習により筏の Haar-Like 特徴量を抽出する．Haar-Like 特徴量とは，複数の局所領域の明暗差から求めた特徴量である．例えば，図 4 のように頬の辺りは目元と比べて明るい色となる．このような明暗差による特徴を抽出したものが Haar-Like 特徴量と呼ばれる数値である．この Haar-Like 特徴量を用いた画像識別機を，カスケード型識別機という．カスケード型識別機はアルゴリズムがシンプルかつ処理時間も高速であるため，カメラ映像をリアルタイムで識別する場面で広く使われている.

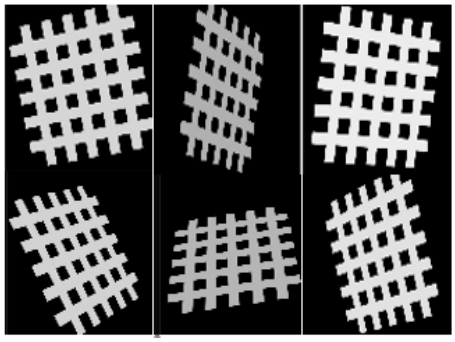

図 3 createsample による画像データの形状変化

図 4 局所領域の明暗差

2.2 ドローンのフライト制御システム

本研究では Python との互換性がある Rize Tech 社製のドローン,Tello を用いて開発を行う.Tello の電源を起動させると特有の簡易 Wi-Fi が飛ぶため,使用したい PC のインターネット設定で選択すると Tello と PC を連携させることができる.(2)で述べた制御システムの課題については，Python の OS に含まれるソケット通信を用いて,ドローンカメラの映像を取り込んだり,上昇などの操縦に関する信号の送受信を行うことでフライト制御を行う．また (3)で述べた画像を提供する web アプリケーションの課題については,フライト制御システムと同一のプログラムに牡蠣筏の写真を特定のフォルダに格納するコードを設けることで疑似的に再現している.

まず全体の概要として,ドローンのフライト制御システムの簡易的なフローチャートを図 5 に示す.

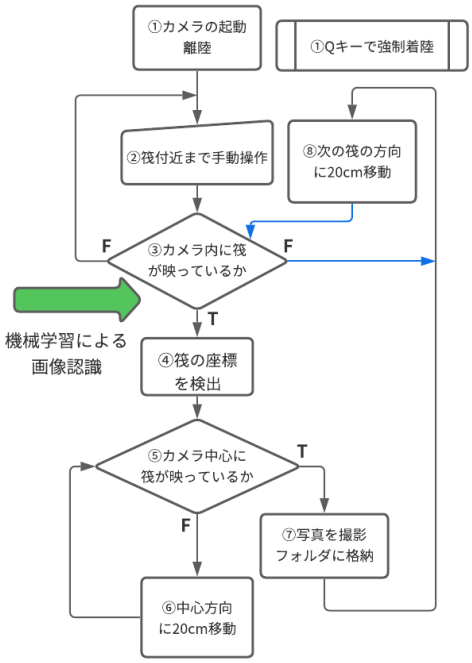

図 5 フライト制御システムのフローチャート

以下ではフライト制御システムの構築内容について図内の番号順に論述する.

①では最初にドローンのカメラの起動を行う.先述した通り,ソケット通信を用いてドローンカメラの映像を取り込んで PC 画面に表示させる.同時に動

作中のコマンド,ドローンの充電残量,経過時間,システムの状態を常時管理し,カメラ映像画面の左上に表示させる.(写真 2)

写真 2 ソケット通信による画面表示

次に,離陸や強制着陸をするためのコマンド設定を行う.Tello には操縦用の関数が用意されており,PC で操縦しやすいようにするため, 以下に示すプログラムのように if 文を用いてコマンドのキー割り当てを行う.また,このプログラムでは While ループを複数使用しているため,ドローンの強制着陸用に Q キーに break 機能を設ける.

```
# Q キーで着陸して終了
            if key == ord('q'):
                land()
                break
```

②では筏付近までドローンを手動操作する.このプログラムでは最初にカメラ内に筏が映ってから自動操縦を始めるため,手動操作による筏への接近が必要である.2 つ目以降の筏までの移動は自動で横移動して行うことができる.

③,④は先述した機械学習を用いた画像認識を行い,カメラ映像内の筏を検出するプロセスである.ドローンのカメラ映像は 1 コマずつのフレームで読み込まれる.この際 Tello から送られてくるカメラ映像のフレーム数はデフォルトで 30fps である.fps とは 1 秒間の映像が何枚の画像で構成されているかを示すの単位であり,fps が大きいほど滑らかな映像になるが,データ量も比例して多くなるため,端末に負荷がかかり映像の処理に不具合が生じやすくなる.このプログラムの実行においては,画像認識の処理によって端末に負荷がかかる.その分カメラ映像の読み込みの処理は軽くする必要があるため,読み込んだ本来のフレームを 3 枚ごとに間引いた画像を新たなフレームとし,OpenCV の関数でグレースケール変換した状態で使用する.画像認識には,プログラムと同じパス上に事前に用意しておいたカスケード型識別機を呼び出す.このカスケード型識別機は,detectMultiScale という OpenCV の関数を用いてフレームを取り込むと,フレームの中から筏を探索する.筏を識別出来るとその範囲を赤枠で囲み,フレームの左下の端を原点とした正方向の x 軸(横軸)と y 軸(縦軸)とした座標値を検出する.筏を識別できなければ最初の筏の場合は②の手動操作を繰り返し,2 枚目以降の筏の識別をする場合は⑧の処理を繰り返して自動で次の筏を探す.

⑤,⑥は筏がカメラの中心に映るように追跡するプロセスである.風の強い海上で筏の写真を空撮するためには,カメラの中心で筏を捉える事が重要になる.しかし④までのプロセスでは,フレーム内の端に筏が映り込んだだけの状態であるため,④で検出した座標値を用いてカメラの中心に映るようにドローンの位置を修正する必要がある.このプログラムでは,④で筏と識別した範囲が持つ 4 つの角のうち,対角である 2 つの角の座標を足して 2 で割ることで,筏の中心の座標(=face_center)を求める.次に,フレーム全体の x 座標(=flame_width)と y 座標(=flame_height)をそれぞれ 2 で割って,フレーム自体の中心の座標を求める.筏の中心が,フレーム自体の中心から左右上下 40 ピクセルの範囲内に入るまでドローンが位置調整を繰り返して,最終的にカメラの中心に筏が映るプログラムを, 以下に示すように作成した.

```
if face_center_x >= 0:
    if face_center_x < frame_width / 2 - 40:
                print("adjustment")
                left()
                keep = 0
    elif face_center_x > frame_width / 2 + 40:
                print("adjustment")
                right()
                keep = 0
    elif face_center_y < frame_height / 2 - 40:
                print("adjustment")
                up()
                keep = 0
    elif face_center_y > frame_height / 2 + 40:
                print("adjustment")
                down()
                keep = 0
    else:
                print("center")
                keep += 1
```

⑦は筏がカメラの中心に映った時に写真を撮影し,所定のフォルダに格納するプロセスである.⑤,⑥で示したプログラムの中に,keep と名付けたカウンターを設けた.筏がカメラの中心に映っている状態(=else)の間,フレームを読み込むごとに keep が 1 カウントされる.カウントが 100 になった時を,確実に筏がカメラの中心に映ったと見なす.この時に,OpenCV の関数を用いて現在読み込んでいる 1 枚のフレームをリサイズして,プログラム内で指定されたパスにあるフォルダへ格納することで,筏の画像を利用者に提供する.

⑧は,③と合わせて次の筏まで移動するプロセスである.基本的に牡蠣筏の群は列になって浮かんでいるため,隣の筏までの移動は直線的な移動だけで可能であると考える.また,列が多少ずれていても,カメラ映像内に筏が映れば③以降のプロセスにより自動的にカメラ中心で撮影することが可能である.したがってこのプログラムでは,③に新たなループを設けてカメラ映像内に筏が映るまで次の筏の方向に直線移動だけを繰り返すようになっている.

３．結果と考察

3.1 結果

先述した方法でドローンの飛行試験を行った.試験時の PC 及びドローンの挙動を大きく 3 項目に分け,結果として評価する.

(1) PC とドローンの連携に関しての結果

プログラム実行後,ドローンカメラの起動や,キーボードによるコマンドの送信は正常に実行された.しかし,筏の認識中に Q キーを入力して強制着陸コマンドを送信しても,実際に着陸動作を始めるまでに数秒の時間差が生じる事があった.通常飛行時は Q キーを入力すると 0.5 秒以内には着陸動作を始めるため,何らかの異常があったと考える.

(2) 筏の画像認識に関しての結果

カメラ内に筏が映ると,赤枠で囲まれて表示された.(写真 3)カメラ内に筏が映っている間は常に赤枠が表示されていたため,筏の画像認識は正常であり,機械学習の精度も十分な高さであったと言える.しかし,カメラ内に筏全体が映りきらないと認識できなかった.

(3) 筏の自動追跡及び写真回収システムに関しての結果

今回の飛行試験では,筏の左側からスタートし，筏がカメラに映るまでは手動操作によって近づけた.写真 3 に示すカメラ起動後 43 秒の時点で,筏がカメラの右側に映り,筏を認識して赤枠が表示されたため,手動入力によるドローンの操縦を止めた.この時,筏の中心がフレームの中心より右側にあるのをプログラムが検知し,ドローンが自動で右に移動を始めた.写真 4 に示す 46 秒の時点では,筏がフレームの中心に映り,先述した keep カウンターが 100 カウントされるまでこの場所を維持した.100 カウントし終わると,次の筏に移動するため,右側に自動で移動し始めた.写真 5 に示す 61 秒の時点では,筏が左側に映っているが,新しく右側に筏を認識するまで keep カウンターを維持している間は,フレーム中心への位置調整のプログラムが動かないようにしているため,認識し終わった筏の情報には干渉されず,次の筏まで直線的な移動を続けた.これら一連の動作はプログラムの通りであるため,自動追跡については正常であった.また,keep カウンターが 100 になった時のフレーム画像が,写真 6 に示すように指定したフォルダ内に格納されていたため,写真回収システムについても正常であった.

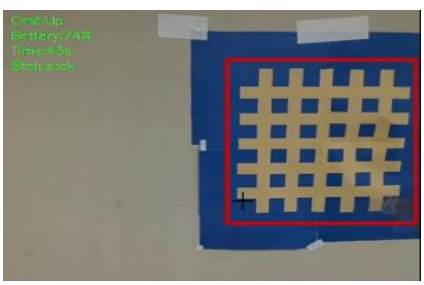

写真 3 カメラ映像(43 秒時点)

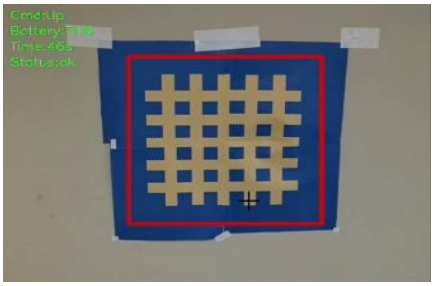

写真 4 カメラ映像(46 秒時点)

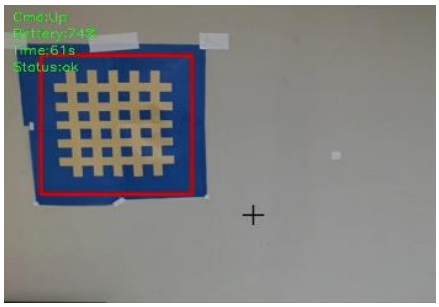

写真 5 カメラ映像(61 秒時点)

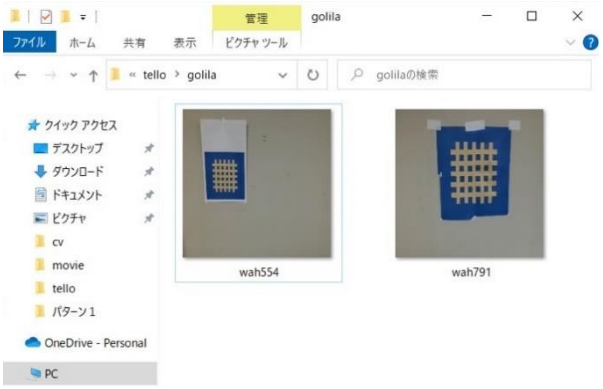

写真 6 写真回収システムのフォルダ

3.2 考察

結果より,牡蠣筏の画像認識が正常に行われ,ドローンとの連携や自動制御,牡蠣筏の写真回収システムも,先述したフライト制御システムのフローチャートを満たしているため,飛行試験としては成功であると考える.

しかし,飛行試験で生じた具体的な欠点や改善点が 3 つある.

1 つ目は,筏の認識中において強制着陸コマンドの実行が遅延したことである.強制着陸コマンドは,飛行に危険が生じた時にも使用するため,入力に対する応答の速さが求められる.このプログラムではドローンの飛行が while ループ内にて実行されており,このループの break を Q キー入力時に設けることで,強制着陸コマンドを成り立たせている.つまり,強制着陸コマンドである Q キーの入力と,実際の着陸動作の間に直接的なプログラムの干渉は無い.この事から,原因はプログラムではなく端末の性能に関係していると考えた.このプログラムの構造は,筏を検知すると自動制御を行うための if 関数内に入り,

その中では筏を検知した座標や範囲の計算,赤枠の表示を行ったり,自動制御用の if 関数,elif 関数の条件分岐が 5 つ並列している.筏の認識中は,これらの処理が重くなり PC に負荷がかかる事で,コマンドの送信が少し遅れる可能性が否定できないため,原因として有力であると考える.将来的には海上にドローンを飛ばすため,ドローンに Raspberry Pi のような小型 PC を搭載する事が考えられるため,重い負荷がかからないようにアルゴリズムの軽量化が必要であると考える.

2 つ目は,カメラ内に筏全体が映りきらないと認識できないことである.原因は 1 枚の画像を複製して教師あり学習させたことにより, 形状のバラツキが小さく学習精度が高すぎる事で, 学習させた形と全く同じ物しか認識できない事が原因であると考える. これはシステムの不具合ではないが, 実際の牡蠣筏が想定よりも流されていたり, 原型がないほど大きく破損していた場合に, 問題が発生している筏を認識できないという事である. この問題はシステムの目的である牡蠣筏のメンテナンスに対して重大な欠陥であると考える. 問題が発生している筏こそ認識する必要があるため, 機械学習と画像認識の方法を工夫し, 柔軟性のある判別が必要である. 今後の研究として 2 種類の方針を考えている. 1 つ目は様々な形状の筏や, 部分的に映る筏の画像を用意し, それぞれで機械学習させて複数個の識別機を作成することだ. 筏の映像を全ての識別機に通すことで様々なパターンの筏を識別できるようにして, 前述の問題を解決しようと考えている.

2 つ目は, 別の機械学習を行うことである. 本研究で用いた教師あり学習以外の手法には, 教師なし学習というものがある. この手法では, 前者のように正解データを与えない代わりに, データの中から共通点を見つけ, 特徴量として扱う. その特徴量を用いて判別するため, 正常な筏も異常な筏もある程度認識可能であると考える. デメリットとして精度が低いことが挙げられるが, 海上での色は海の青, 筏の茶色, 太陽の反射光の白, の 3 つが主であるため, ご検出の可能性は小さいと考える. また現在の教師あり学習の精度の高さも貴重であるため, 回収した大量の筏の異常検知システムを設けた場合に, そこで精度の良さを活かせると考えている. 場合や目的に応じた機械学習法の使い分けが重要である.

3 つ目は,現在のシステムでは全ての筏の追跡ができないことである.図 6 のように牡蠣筏はある程度 1 列に並び,何列も並列して設置されている.このシステムでは事前に決めた一方向にしか進めないため, 順次進む方向を指定しなければならない.また 1 つの列を追跡し終わると次の列まで移動しなければならない.これらの問題を解決するためには,牡蠣筏の列 1 つずつに始点と終点の座標を設け,その位置情報をもとにドローンを制御することが必要である. 今後の研究では GPS 情報をシステムに組み込もうと考えている.図 7 のように列の始点と終点の牡蠣筏に GPS 送信器を設置し,それらを追跡する形で牡蠣筏の追跡を行う.そのうえで本システムの位置調整機能を有効に使用できると考えている.

図 6 牡蠣筏の並び方

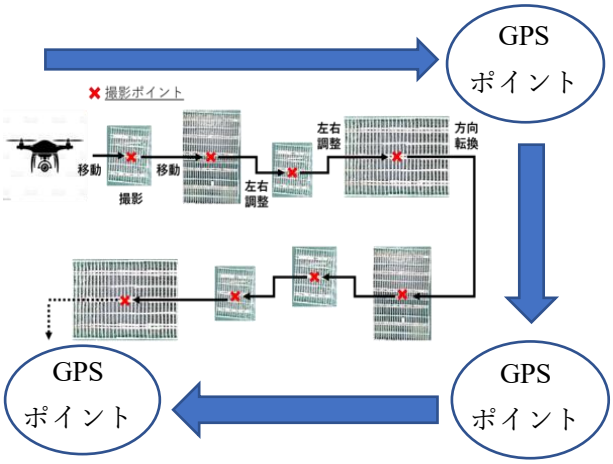

図 7 GPS による追跡

４．結言

先述した本研究の目標において,支援システムを開発する部分は,Python のライブラリである OpenCV により,カスケード型識別機を作成する事,カメラに映っている筏の座標を検出し,追跡するようにプログラムを組み,カメラ中心に筏が映ると写真をフォルダに格納する事によって達成した.実際に Tello を用いてシステムの飛行試験・評価を行う部分は, ソケット通信を用いて,Tello のカメラ映像と,動作状態を常に読み込み表示した事で,可視化による評価を実現して達成した.

また,今後は筏の自動追跡には教師なし学習を取り入れる事で,筏に異常が生じていた際も判別し,自動追跡・写真回収ができるようにする.また,現状のシステムに筏の状態を判別する機能を追加して,システムとしての利便性を向上させる事を目標とする.正常な筏と異常な筏を判別できる機械学習には現在の教師あり学習を用いて,大量の筏の画像の中から異常な筏を発見し易くさせる事で,メンテナンス業務の更なる効率化を目指す.この目標には,機械学習の手法を現在の明暗差だけではなく,輪郭,色彩の面からも学習させる必要があると考える..

５．参考文献

1) 機械学習と深層学習の違いとは？メリットや課題を挙げながら解説.
https://www.dsk-cloud.com/blog/difference-machine-learning-and-deep-learning

2) Haar-like 特徴量を用いたカスケード分類器による前方車両の識別.
http://www.ail.cs.gunma-u.ac.jp/ailwiki
3) OpenCV で物体検出器を作成する⑤ ～ createsamples～
https://www.pro-s.co.jp/blog/system/opencv/6397
4) OpenCV で物体検出器を作成する⑥ ～ traincascade～
https://www.pro-s.co.jp/blog/system/opencv/6471
5)教師あり学習とは？
https://business.ntt-east.co.jp/content/cloudsolution/column-161.html
6)教師なし学習とは？
https://business.ntt-east.co.jp/content/cloudsolution/column-162.html

６．謝辞

本論文の作成にあたり, 適切な助言を賜り, 丁寧に指導して下さった吉川祐樹先生に感謝致します. また, 共同研究者である, 清水一世さんには, 機械学習の手法や分類機の作成など, 研究の核となる部分の調査を熱心にご協力いただきました. ここに感謝致します.