ライダーデータによる歩行者の統計的検出

Statistical detection of pedestrians from LiDAR data

鎌倉研究室
17N7100029J　森田 和希

1　研究背景

近年，画像認識等を用いての歩行者の検出が盛んだが，プライバシーや倫理的観点からの問題が必ず付いて回る．そこで人物個人などのプライバシーに考慮した点から，LiDAR によって得られた 3D 点群データを用いて物体の認識や，歩行者の検出が行われている．そのアルゴリズムは様々なものがある．しかし，その中でも，Deep Learning を用いて歩行者の検出を行なっているものは少ない．そこで，本研究では，VLP-16 による 3D 点群データを白黒画像に見立て，畳み込みニューラルネットワーク (CNN) によって歩行者の検出を行う．また，現在 LiDAR は規格がいくつかあるが，精度の高い規格となると必然的にデータ量が多くなる．しかし，アルゴリズムはどれも複雑かつ計算量が多く，リアルタイムでの計測に不向きであると思われる．そこで，Deep Learning の際に扱った LiDAR よりも精度が高く，データ量も多い LiDAR「HDL-64」を用いてより簡潔な方法を提案する．

2　LiDAR およびニューラルネットワーク

2.1　LiDAR

本研究で用いた LiDAR は Velodyne 社の「VLP-16」と「HDL-64」を使用した．LiDAR とは，レーザー光を発し，物体に反射して戻ってきた時間を用いてその物体と本体との距離を計測する機械のことである．VLP-16 は 16 個のセンサーを搭載しており，水平方向 360 度，垂直視野 30 度の範囲で 1 秒間に 15 フレーム，約 300,000 データを取得することが可能である．また，半径 100m 以内のそれぞれの点群の誤差は ±3cm 以内で取得することができる．HDL-64 は 64 個のセンサーを搭載しており，垂直視野 26.8 度，水平方向 360 度の範囲のデータを取得できる．1 秒間に最大 1,333,000 ポイントの点群を，半径 120m までを誤差 ±2cm で以内で取得することが可能である．

2.2　畳み込みニューラルネットワーク

畳み込みニューラルネットワーク (Convolutional Neural Network) は，通常のニューラルネットワークにおける中間層の部分に，畳み込み層とプーリング層が含まれる特殊なニューラルネットワークであり，主に画像認識に使用されている．通常のニューラルネットワークはすべてのユニットが結合されているが，畳み込みニューラルネットワークでは畳み込み層とプーリング層によって特定のユニットのみが結合する仕組みとなっている (図 1).

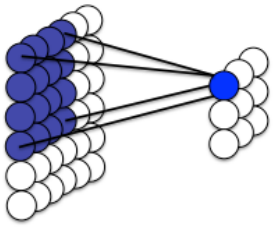

図 1: 畳み込み層とプーリング層の結合イメージ

2.2.1　畳み込み層

畳み込み層は，上記した通り特定のユニットのみが結合しており，この重みの形をフィルタと呼ぶ．通常，フィルタは入力画像よりも小さいサイズで構成されており，このフィルタに入力を通すことによって，小さいサイズの画像を生成 (畳み込み) する．

入力画像のサイズを $W * W$，画像のインデックスを $i, j(i, j = 0, ..., W - 1)$ とし，画素値を h_{ij} と表す．フィルタのサイズを $H * H$，フィルタのインデックスを $p, q(p, q = 0, ..., H - 1)$ とし，画素値を w_{pq} と定義すると，畳み込みは，

$$u_{ij} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{i-p,i-q} h_{pq} \tag{1}$$

というように画像とフィルタ間の積和計算として定義される．

以上で説明した畳み込みはグレースケールの画像の場合である．実際にはカラー画像のような多チャネルの画像を使用するケースが多く，この場合は複数のフィルタを並行して演算をする．チャネル数が K の画像では，各画素は K 個の値をもつ (RGB の画像では $K = 3$)．これらの画像の画素数は $W \times W \times K$ と現わせる．これらをモデル化すると図 2 のようになる．

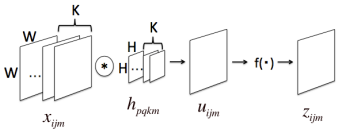

図 2: モデルイメージ

ここで，m は各フィルタの数 $(m = 0, ..., K-1)$，$f(\cdot)$ は活性化関数，z_{ijm} は出力画像を表す．

プーリング層では，畳み込み層で抽出された特徴の位置感度を若干低下させることで，対象とする特徴量の画像内での位置が変化した場合でもプーリング層の出力が普遍になるようにする．

2.2.2 プーリング層

プーリング層は通常，畳み込み層の直後に設置される．

プーリング層では，畳み込み層で抽出された特徴の位置感度を若干低下させることで，対象とする特徴量の画像内での位置が変化した場合でもプーリング層の出力が普遍になるようにする．

プーリング層では入力画像での画素 (i, j) を中心とする $H \times H$ の正方領域をとり，この中に含まれる画素値を使って 1 つの画素値 u_{ijk} を求める．方法はいくつかあり，$H \times H$ の中にある最大値を選ぶ最大プーリング (2)，$H \times H$ の平均値を選ぶ平均値プーリング (3) などがある．

$$u_{ij} = \max_{p,q \in P_{ij}} z_{pq} \tag{2}$$

$$u_{ij} = \frac{1}{H^2} \sum_{p,q \in P_{ij}} z_{pq} \tag{3}$$

3 研究手法

3.1 立体物判定

3D 点群データから床の点群を取り除き，高さ情報を抜き取った XY 平面状のグリッドマップに投影し，同じグリッド内にある点群を一つのグループとする．本研究ではグリッドサイズを $0.35m$ とする．グループごとに，z 軸での分散を計算しする．立体物である場合，床（天井）のみの場合よりも分散が大きくなることから，ある一定の閾値を設定し，その数字より大きい分散の値をとったグループを立体物と判定する．本研究では閾値は 0.05 とする．

また，HDL-64 のデータを扱う際には，グリッドサイズを 0.35 に設定した後，さらに 0.5 と増やし立体物判定を行う．

3.2 クラスタリング

立体物判定と同様にグリッドマップに投影し，グリッドに点が存在していれば 1，存在していなければ 0 として二値画像を生成する．生成された二値画像に対してラベリング処理を行い，クラスタリングを行う．ラベリング処理とは，同じ値が連続している部分を個別に検出する手法である．

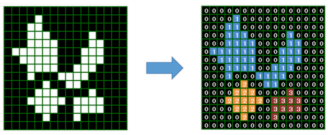

図 3: ラベリング処理の例

3.3 ラベルづけ

立体物判定，クラスタリングを終えたデータに対し，どのグループが歩行者であるかを，点群をプロットし視覚的に，歩行者であるグループには 1，そうでないものには 0 と手作業でラベルづけしていく．

3.4 データの加工

LiDAR によって得られたデータは，各点ごとの情報が入った行列となっている．そのままでは CNN に当てはめることが難しいため，それを一枚の白黒画像とみなすようにデータを整理する．

1 フレーム間に約 20000 個のデータを取得する．このデータからから一枚の白黒画像とみなすために，16×8000 の行列に変換する．3 次元座標から，

$$\sqrt{x^2+y^2+z^2}$$

により，LiDAR との距離を求め，距離を各ポイントに当てはめていく．そうすることで，表 1 のような 16×8000 の行列を一枚の画像とみなす．

表 1: LiDAR から取得するデータセット

$\phi \setminus \theta$	1	2	$\cdots$	J
1	r_{11}	r_{12}	$\cdots$	r_{1J}
2	r_{21}	r_{22}	$\cdots$	r_{2J}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
16	r_{161}	r_{162}	$\cdots$	r_{16J}

しかし，このままでは計算量が膨大なため，横角度 30 度の 8 分割し，16×1000 のデータを作成する (図 4)．歩行者が含まれているデータと含まれていないの 2 種類のデータセットを計 896 個用意し学習を行う．

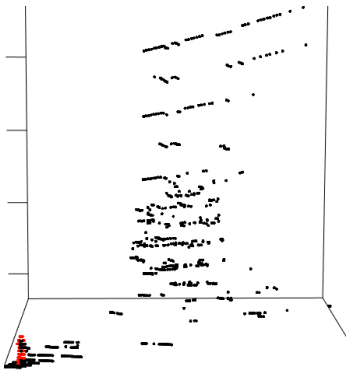

図 4: 16 分割したデータに歩行者をラベルづけした図

3.5　ニューラルネットワークの構造

今回設計したニューラルネットワークの構造を図 5 に示す．尚，本研究で用いた関数は，畳み込み層が soft-max，プーリング層では MAX プーリングを使用する．

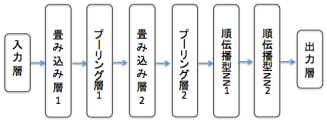

図 5: ニューラルネットワークの構図

3.6　2 フレーム間の点群の差

HDL-64 はセンサーの数が VLP-16 に対して 4 倍かつセンサー 1 つでの取れるデータ量も多いため，1 フレームの点群データ数も 4 倍強の 130000 ポイントとなる．しかし，1 秒差ごとでフレームを 1 つずつ抜き出した場合，LiDAR 本体が固定されている場合，動いていないものがほとんどなため，点群ごとの差を取ることで大半のデータを除去することができる．しかし，各フレームごとに点群数に差があるため，単純な行列計算によって求めることができない．そこで，HDL-64 の誤差 (約 $\pm 2cm$) を考慮したイプシロンボールを各点群に対し作成し，その中に比較先のフレームに点群が存在しているかを検証し，存在していた場合削除していく．

4　結果と考察

約 900 個のデータセットを使用しての学習によって作成されたニューラルネットワークでは，全ての数字が 0 となってしまったため，歩行者が写っているデータ 400 個のみで学習し直した．ニューラルネットワークによって歩行者として検出された点を青にプロットしたものが図 6 である．

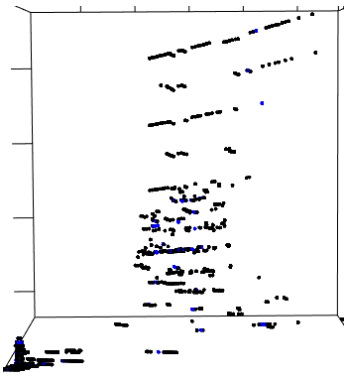

図 6: 学習結果

図 6 のように，人以外の部分に点が散らばるように検出されてしまい，うまく歩行者を検出することはできなかった．

差分をとったデータをプロットしたのが図 7 である．なお，ここではフレーム 140 のデータをもとに比較している．

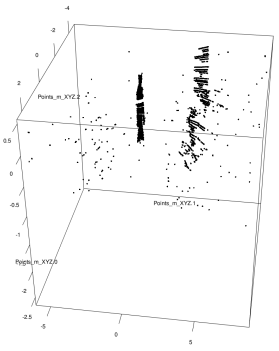

図 7: 差分をとったデータ

グリッド幅を 0.35，0.5 という大きさで繰り返し立体物判定を行う．0.5 とした理由は人の幅がおおよそ 50cm 以内に収まるからである．立体物判定を行い，各グリッドに含まれている点群の数が最も多かったグリッドに含まれている点群を抽出し，それを拡大したものが図 8 である．

図 8: 歩行者のみの点群

5　まとめ

本研究では，VLP-16 の点群データを用いての DeepLearning を通じた歩行者の解析と，HDL-64 での点群データに対しての歩行者の抽出を行なった．

VLP-16 では，XY 平面に対してグリッドを作成し，各グリッド内での点群に対して，Z 軸の分散を用いた立体物判定を行い，立体物とされた点群に対しラベリングによるクラスタリングを行なった．その後，視覚的にクラスタリングされた点群に対し歩行者かどうかのラベリングを行なった．以上のようなデータをさらに加工し，白黒画像のような形にした後，DeepLearning によって学習を行なったが，思うように歩行者を検出をすることはできなかった．その要因として，教師データとなる点群に対する歩行者へのラベルづけの荒さや，データ量の少なさが考えられる．より精度の高いデータを作成することで解決策が見えると考えられる．

HDL-64 のデータを用いての歩行者の検出は，1 フレームに対するデータ数のが多いため 2 つのフレームによる差分を用いてデータ数の軽量化を測った．差分は基盤となるフレームの点群に対しそれぞれにイプシロンボールを作成し，比較先の点群に同じ点があるかを判定した．軽量化されたデータに対し立体物判定を用いて，各グリッドに存在する点群の数を最も多いものを抽出することで，過去の方法に比べ，余計な点群が混じることなく，歩行者のみを検出することができた．本研究では，Deep Learning による検出までは行うことはできなかったが，軽量化されたデータを扱うことで Deep Learning による歩行者の検出へと近づくことができた．

参考文献

[1] Galai, Bence, and Csaba Benedek. (2017). *Gait Recognition with Compact Lidar Sensors.*

[2] 斎藤 康毅. (2016). ゼロから作る *Deep Leaning – Python* で学ぶディープラーニングの理論と実装, オライリー・ジャパンオーム社.

[3] Nick McClure. (2017). *TensorFlow* 機械学習クックブック *Python* ベースの活用レシピ *60+*, インプレスコミュニケーションズ.

[4] 城殿清澄, 渡邉章弘, 内藤貴志 & 三浦純. (2011). 高解像度レーザレーダによる歩行者識別, 日本ロボット学会誌, 29(10), 963-970.

[5] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks.*, In Advances in neural information processing systems (pp. 1097-1105).

[6] 魚住剛弘, & 菅沼直樹. (2011). 自動車の自律型自動運転のための全方位レーザー を用いた障害物検出., In 自動制御連合講演会講演論文集 第 54 回自動制御連合講 演会 (pp. 156-156). 自動制御連合講演会.