RGBD 画像の三次元点群への変換

4 年 C 組　濵野 泰地
指導教員　藤野 智美

１．要約

近年注目されているドローンの自動操縦では、私たちが日常的に使用している 2D の地図ではなく、3D の地図が必要である。本研究では 3D センサを使用し、3D の地図を自動で作製することを目指した。第一段階として、色データに距離データを追加した RGBD データを取得することが可能な RGBD センサを利用し、そのデータをx, y, zの座標軸をもつ点の集合（三次元点群）へと変換するシステムを開発した。成果として、RGBD のデータをセンサを原点とした三次元点群へ変換し、視覚的にわかりやすいデータを作製することができた。

キーワード：デプスカメラ、RGBD データ、三次元点群

２．研究目的

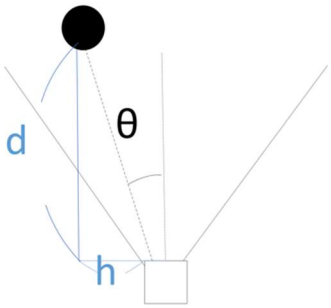
私は以前より、自動運転の技術について興味があった。その理由の 1 つに、自動運転が輸送に携わる労力や人件費の削減がある。その技術を近年注目されているドローンによる今の技術と組み合わせれば、より、円滑に輸送ができると考えた。しかし現状、ドローンの自動運転には様々な課題がある。特に大きな課題とされているのは、ドローンの詳細な座標の取得が難しい点である。ドローンの操縦においては、縦、横の座標に合わせて、高さを考慮しなければならないため、これらの情報を持った 3D の地図の普及が必須である。しかし、この 3D 地図を自動で活用するシステムは、未だ普及していない。本研究では、3D の地図を自動で作製するシステムの開発を目指し、第一段階としてセンサに映る対象物までの距離を測り、色データと組み合わせることのできる RGBD センサを使い、得られるデータを三次元点群へ変換するシステムを開発する。

３．研究手法

３．１ システム構築

今回は RGBD センサとしてインテル RealSense デプスカメラ D435 (図 1) を使用した。 RGBD データは色 (RGB)、と距離 (D) のデータを持っている点 (ピクセル) の集合体である。このデータから三次元点群を作製するために、1 ピクセルごとにセンサを原点とする座標データを取得していく。

図 1　デプスカメラ D435



□:カメラ
●:物体
θ:物体への角度
d:物体への縦の距離
H:物体への横の距離

図 2　測定のイメージ図

RGBD センサ（以下、センサ）は視野角を持っており、データのピクセルはその視野と連動している。よって、ある RGBD データの一つのピクセルにおける、そのデータの中心からの距離（L）
と、そのピクセルの現実世界でのセンサから見た角度（θ）は正比例し、最大値はセンサの視野角（A）の半分となる。この関係を用いて物体への角度θを求めると、以下の式で表される。

$$\theta = A \times L \times \frac{1}{2}$$

実際はx, y, zの三つの軸があるが、図 2 はセンサを上からみた図であるため、図 2 から見て奥行方向にもう一つ軸が存在することになる。そのもう一つの軸における物体への角度を求めなくてはならない。そのため、二回分この操作をする。次に、求められた角度と、距離データを利用して、カメラを原点とした時のピクセルの座標を求める。

$$h = tan\theta \times d$$

h: 物体への横の距離　　d: 物体への縦の距離

一連のデータ測定方法をまとめると、以下の流れとなる。
①RGBD 画像の一つのピクセルに着目し、そのピクセルに映る物体までの角度を計算する。
②①を用いて、その物体までの横の距離を求める。
その結果、カメラを原点としたその物体までの三次元の座標（x, y, z）が得られる。

３．２システムの改良と言語選択

システムのコードを書くにあたって研究初期は Python でコードを記述していたが、計算量がネイティブな Python で処理するには多すぎるため、そのまま利用することはできなかった。一方、センサからデータを受け取るためのモジュールは Python で使ったほうが利用しやすかったため、Python 以外の言語に置き換えることは利点が少ないと感じた。そこで、Python で書かれている、または Python から呼び出すことができるようなシステム処理を高速化する手法が必要だった。活用するシステムの検討過程を以下に示す。

（0）ネイティブな Python のみの利用

ネイティブな Python には行列計算できるものがリスト型しかなく、それは、動作速度や、使用感に問題があり、モジュールや外部ライブラリの使用を余儀なくされた。

（1）Numpy モジュールの利用

Numpy とは、Python の行列計算を、一部 C 言語を介して行うことで、より高速かつ円滑に処理を行うことを目的としたモジュールである。このモジュールの利用により、演算の高速化を図った。しかし、この Numpy モジュールには今回の研究を行うにあたって以下に示す 2 つの欠点があった。
①多次元配列においてすべての行、列の長さを統一しなければならない。座標演算でx, y, zの軸と三次元配列を用意する必要があったが、場所によって点の数に違いがあり、不都合が生じた。
②単純な計算速度の問題。Numpy は高速に演算することを目的としているが、あくまで、Python のモジュールであり、事前コンパイルがないため、コンパイル言語と比べると非常に遅く、ボトルネックとなった。

（2）Numba の利用

Numba とは Numpy によって書かれたソースコードを事前にコンパイルし、コンパイル言語のように動作させ、Numpy の高速化を目的としたコンパイラである。しか

し、①の問題に関しては、結論としてNumpy モジュールに属する Ndarray 型を使わなければならなかったため、解決しなかった。また、②の問題に関しては、大幅に改善はしたものの、やはり速度が足りず最終的な解決へとは至らなかった。

（3）Cython 言語の利用

Cython とは、Python とほぼ同じ文法や Python のモジュールを利用しつつ、Python 用モジュールへコンパイルすることを目的としたプログラミング言語である。内部の処理で C 言語に置き換えてコンパイルすることで、高速に動作する。Numpy や、Numba との大きな違いは、書かれたコードを直接 C 言語に変換し、より高速に動作できる点だ。この手法により、②の問題は解決したが、Cython で行列計算が可能なモジュールは Numpy しかなく、①の問題は解決できなかった。

（4）Pybind11 ライブラリの利用（採用）

Pybind11 とは、C++言語で書かれたコードを Python モジュールにコンパイルし、C++の型や、処理系をそのまま Python に組み込むことで高速化や、その他 C++ライブラリの導入を可能にしたライブラリである。C++には Vector や Eigen といった、行列計算ライブラリがあり、①の問題を解決できた。また、もちろんネイティブな C++で書かれているため、（1）～（3）のどれよりも高速に動作し、②も同時に解決した。

最終的に唯一①、②の問題を同時に解決できたのは（4）の Pybind11 のみだったため、これを利用することにした。

４．実験と結果

作製したシステムの精度を確かめるため、以下の工程の実験を行った。

①センサの前から少し離れたところに壁とペットボトルを置く（図 3）。

②①の状態でデータの取得を行う。

③②のデータを作製したシステムを用い、三次元点群に変換して描画する。

実験の結果として、図 4 のようなデータが得られた。実際は色のデータも取得できるが、今回は三次元点群の座標データに注目したいため、色データは除いている。図 3 と図 4 を比較すると、センサからの死角になっている部分は映っていないが、それ以外の部分においては実際のペットボトルや壁の距離感を再現できている。

図 3　計測した物体

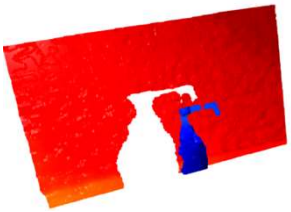


図 4　得られた三次元点群

５．今後の展望

本研究で開発したシステムは RGBD 画像を三次元点群へ変換することは達成できたが、映る範囲がセンサの視野に制限されるため、センサから死角になっている部分を表現することができない。また、定点からの観測にしか対応していないため、当初の目的である、3D 地図の自動作製には程遠い。そのため、今後は死角などの映っていない範囲を予測するような機能を搭載したり、ドローンなどにセンサを搭載して、移動しながらデータを取得するようなシステムの開発を行っていきたい。

６．謝辞

今回の研究を行うにあたり、顧問の藤野智美先生に多大なご指導を賜りました。この場を借りて御礼申し上げます。