Visual SLAM フレームワークの開発
― 簡単に手法を切り替えられるフレームワークの提案 ―

1. 背景

SLAM（Simultaneous Localization and Mapping）とは，センサーから得られる情報を用いて地図を作ることができる技術である．また，地図を作りながら，その地図の中で自分がどこにいるのかを推定することができる．SLAM には加速度センサーやジャイロセンサーなどのさまざまなセンサーが用いられる．大規模な地図を作る際は，位置情報を得るために GPS も使用される．ひとつひとつのセンサーには誤差が含まれるので，各種センサーの値をうまく統合することで，正確な位置を得る．

物体までの距離を測るための LiDAR と呼ばれるセンサーも非常によく用いられる．LiDAR とは，レーザー光を用いて周囲の物体までの距離を測定するデバイスである．周囲の物体との距離を測ることで地図を作成することができるうえ，それらの物体に対してロボットが相対的にどう動いているかを観測することで，ロボットの位置や姿勢を推定することも可能となる．ロボットにセンサーをたくさん搭載し，そこから得られる情報をうまく統合すれば正確な姿勢推定と地図の作成を行うことができる．しかし，これらのセンサーは高価であり，入手するだけでもかなりの資金が必要になる．なかでも LiDAR は特に高価で，ひとつあたり数十万～数千万円で販売されている．センサーがこれほど高価だと，SLAM を気軽に利用することはできない．

Visual SLAM は画像列から 3 次元地図とその地図内でのカメラの移動経路を推定する技術であり，センサーとして動画を撮影するカメラを用いる．カメラはほかのさまざまなセンサーと比べると安価であり，スマートフォンにも搭載されているように非常に広く普及している．広く普及しているカメラをセンサーをとすることで，SLAM のさまざまな応用が生まれ，巨大な市場を開拓することが期待できる．

Visual SLAM にはさまざまな手法が提案されており，アプリケーションに応じて手法を使い分ける必要がある．しかしながら，ソフトウェア開発環境が整備されていないため，開発者がアプリケーションに対して適切な手法を選ぶことが難しい．

2. 目的

本プロジェクトでは使いやすい Visual SLAM フレームワーク“Tadataka”の開発を目的とした．

3. 開発の内容

3.1. Visual SLAM フレームワーク“Tadataka”

本プロジェクトではプロジェクト期間中に Visual SLAM の実装には至らなかったものの，Visual SLAM の要素技術である Visual Odometry の手法を 2 種類実装した．

フレームワークには，現時点で特徴点ベースの Visual Odometry（図 1）と，画像の輝度を直接用いる手法である DVO（Dense Visual Odometry）（図 2）が含まれている．

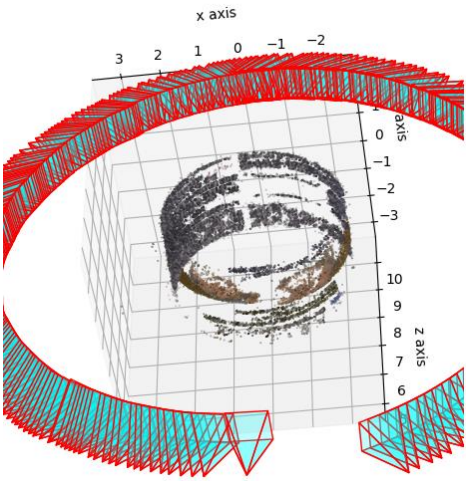

図 1. 特徴点ベースの Visual Odometry による缶詰の復元結果

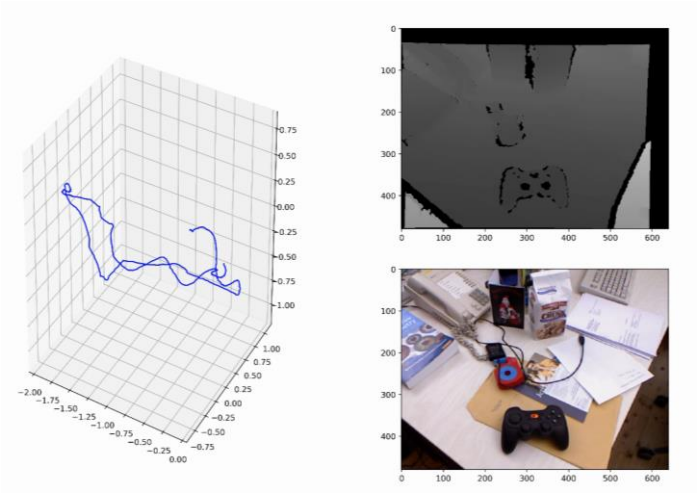

図 2. DVO によるカメラの移動量の推定結果

3.2. Bundle Adjustment のモジュール化

Bundle Adjustment のパッケージを作成した. Bundle Adjustment とは, 3 次元復元の処理の過程において蓄積していく誤差を修正することができる手法である(図 3). これは Visual SLAM 以外のさまざまな 3 次元復元手法にも適用できるため, 別パッケージとして切り離した.

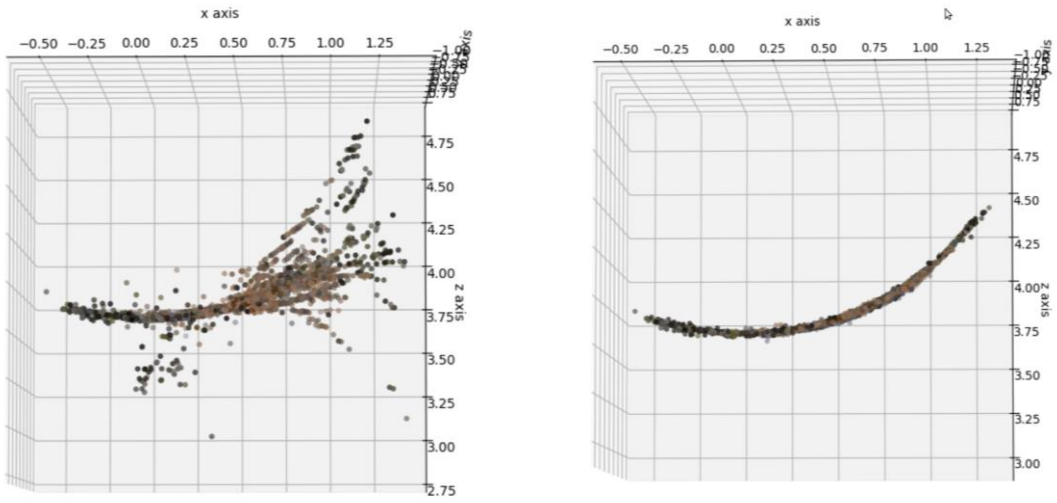

図 3. Bundle Adjustment を用いた誤差の修正結果
左は Bnudle Adjustment なしで Visual Odometry を実行したもの. 復元結果が破綻してしまっている. 右は Bundle Adjustment ありで Visual Odometry を実行したもの. 地図が破綻することなく, 正しく 3 次元構造を復元できている.

3.3. データセットの作成

Visual SLAM のデータセットは複雑なものが多く, 開発やデバッグに利用しづらいという問題がある. この問題を解消するため, 複雑なフレームを含まず, かつ復元結果がわかりやすいデータセットを作成した.

3.4. 知見の公開

Visual SLAM は基礎的な部分でさえ知見が公開されていないことが多いため，開発過程で得た知見をまとめ，Web ページ上で公開した．

4. 従来の技術（または機能）との相違

既存の Visual SLAM フレームワークは手法ごとに実装が独立しているため，別々の手法を切り替えることが難しかった．Tadataka を用いると，中身が全く異なる別々の手法をたった 2 行書き換えるだけで切り替えることができる．

4.1. 手法の切り替え

別々の手法を切り替える際は，コードを 2 行書き換えるだけでよい（図 4，図 5）．

```python
from tadataka.dataset import TumRgbdDataset
from tadataka.vo import DVO


dataset = TumRgbdDataset(
    "datasets/rgbd_dataset_freiburg1_desk",
    which_freiburg=1
)

vo = DVO()

for frame in dataset:
    pose = vo.estimate(frame)

```

図 4. DVO を動作させる場合

```python
from tadataka.dataset import TumRgbdDataset
from tadataka.vo import FeatureBasedVO


dataset = TumRgbdDataset(
    "datasets/rgbd_dataset_freiburg1_desk",
    which_freiburg=1
)

vo = FeatureBasedVO()

for frame in dataset:
    pose = vo.estimate(frame)

```

図 5. Feature Based VO を動作させる場合

4.2. データセットの切り替え

データセットを切り替える際は，該当部分を書き換えるだけでよい（図 6，図 7）．

5. 期待される効果

本プロジェクトで使いやすいフレームワークを開発したことで，Visual SLAM の研究開発と産業応用が促進される．また，単眼カメラという非常に広く普及しているセンサーを対象としたことで，Visual SLAM を利用した多様なアプリケーションが開発され，巨大な市場が生まれることが予想できる．

```python
from tadataka.dataset import TumRgbdDataset
from tadataka.vo import DVO


dataset = TumRgbdDataset(
    "datasets/rgbd_dataset_freiburg1_desk",
    which_freiburg=1
)

vo = DVO()

for frame in dataset:
    pose = vo.estimate(frame)

```

図 6. TUM RGB-D データセットを用いる場合

```python
from tadataka.dataset import NewTsukubaDataset
from tadataka.vo import DVO


dataset = NewTsukubaDataset(
    "datasets/NewTsukubaDataset",
)


vo = DVO()

for frame_l, frame_r in dataset:
    pose = vo.estimate(frame_l)

```

図 7. New Tsukuba Stereo Dataset を用いる場合

6. 普及(または活用)の見通し

Visual SLAM の研究成果は再現性に乏しい. Deep Learning では, フレームワークが発達したことにより, 再現性が高く, 応用もしやすい研究成果が数多く生まれた. Visual SLAM においても, 使いやすいフレームワークが数多く生まれ, 研究開発の基盤となり, 再現性のある研究がなされることを願う.

7. クリエータ名

石田 岳志

(参考)関連 URL

- Visual SLAM フレームワーク“Tadataka”:https://github.com/IshitaTakeshi/Tadataka
- Bundle Adjustment のモジュール化:https://github.com/IshitaTakeshi/SBA
- 知見の公開:https://ishitatakeshi.netlify.com