UAV空撮画像における3次元モデリング（SfM/MVS）ソフトウェアの形状特性比較

阪南大学　北川　悦司

自己紹介

- 北川悦司
  - 博士（情報学）
  - 阪南大学経営情報学部　教授
  - 23歳の時にITベンチャー企業を起業
    - 約12年間で様々な業務をこなす
  - 現在は個人事業主として企業と共同研究や受託開発などを行っている
  - 専門分野：IT全般
    - 写真測量，3次元点群処理，画像処理，SLAM
    - その他，様々なシステムやアプリを開発

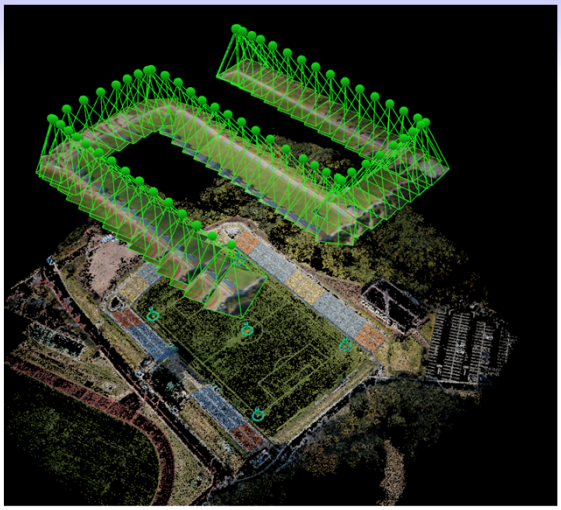
背景

- 地上分解能が小さい空中写真を容易に撮影できる手段としてUAV（Unmanned Aerial Vehicle）が注目
- UAV画像を用いた3次元計測にはSfM（Structure from Motion）とMVS（Multi-View Stereo)技術を用いた3次元モデリングソフトウェアがよく利用される
  - Pix4D，PhotoScan（2019年1月以降はMetashape）など

SfMとは

- Structure from Motionの略で，複数枚の写真からカメラの撮影位置と撮影角度を求める技術
  - つまり，写真測量のこと
  - 基本的な流れは，画像処理などで対応点（各写真の同じ場所）をもとめ，バンドル法などで写真測量する

SfMの画面キャプチャ（Pix4D）



MVSとは

- Multi-View Stereoの略で，SfMの計算結果（カメラの撮影位置と撮影角度）を用いて，点群を大量に生成する技術
  - 基本的な流れは，SfMの結果を用いて画像処理などで対応点を生成し，三角測量で点群を発生する
- Pix4DやPhotoScanは，「SfM/MVSソフトウェア」や「SfMソフトウェア」，「写真測量ソフトウェア」などと呼ばれる

研究の目的

- 3次元モデリングソフトウェア（SfM/MVS）はそれぞれに特徴があると言われているが，その特徴を明確にした報告は少ない
  - どのソフトウェアの精度が良いの？
  - ○○の場合，どれを使えばいいの？　など

そこで

- 近年よく利用されているPix4DとPhotoScanの2つのソフトウェアについて，比較実験を行い，特徴を明らかにすることを目的とする

研究内容

- 既存研究では，
  - 検証点における定点の評価のみで，3次元形状の比較はあまり行われていない
  - レンズ補正式の比較方法が間違っている

そこで

- 本研究では，
  - SfMにおける検証点の評価
  - レンズ補正式の違いの明確化
  - MVSにおける3次元形状の比較

実験データ

	場所	撮影高度	地上分解能	枚数	GCP 標定点	検証点
データ1	宮城県気仙沼市	約40m	約1.0cm	144	5点	7点
データ2	岡山県児島湾干拓地 海岸保全施設	約40m	約0.5cm	517	5点	17点
データ3	福岡県宗像市 グローバルアリーナ スタジアム	約150m	約2.5cm	64	5点	なし

データ1の画像

データ2の画像

データ3の画像

実験内容（SfM/MVSソフトウェア設定）

- 利用ソフトウェア
  - Pix4DmapperDesktopVer.4.1.24
  - PhotoScan ProfessionalVer.1.4.1.5925
- 処理フロー
  - SfM（カメラ位置算出→GCP設定→カメラ位置再計算）→MVS（3次元点群生成）
- 画像サイズ
  - カメラ位置算出時は1/1，点群生成時は1/2の設定
- GNSS（Exif）情報
  - PhotoScanの場合のみ精度が悪くなったため，GPS情報は削除して利用しなかった

SfM処理の比較

検証点における3次元座標の精度比較

高さ成分については，MVSの箇所で詳細説明

	場所	撮影高度	地上分解能	枚数	GCP 標定点	検証点
データ1	宮城県気仙沼市	約40m	約1.0cm	144	5点	7点
データ2	岡山県児島湾干拓地 海岸保全施設	約40m	約0.5cm	517	5点	17点

			水平成分（XY） RMSE	高さ成分(Z) RMSE	全成分 RMSE
標定点残差	データ1	PhotoScan	0.051	0.036	0.063
		Pix4D	0.027	0.033	0.043
	データ2	PhotoScan	0.027	0.010	0.029
		Pix4D	0.006	0.008	0.010
検証点残差	データ1	PhotoScan	0.054	0.053	0.075
		Pix4D	0.041	0.044	0.060
	データ2	PhotoScan	0.025	0.024	0.034
		Pix4D	0.006	0.033	0.033

今回のデータでは
Pix4Dの方が精度的には同等もしくは高精度

標定点（GCP）
Pix4Dの方が高精度

水平成分（検証点）
Pix4Dの方が高精度

高さ成分（検証点）
データ1ではPix4D
データ2ではPhotoScan
が高精度となった

レンズディストーション補正式の比較

- 国土地理院の研究報告では，
  - 日本測量協会のカメラ検定結果を固定値として比較
  - 精度が著しく低下することを指摘

- 写真測量の理論上このように著しく低下することは考えられない
  - 筆者らは測量協会と各ソフトウェアのレンズ補正モデルが異なると推測
  - マニュアルの調査や実験などでアルゴリズムを比較

レンズ補正モデルの比較結果

	補正モデル	変換方向	座標の単位	主点位置の表現方法
測量協会	Australisモデル	補正前→補正後	正規化なし	主点位置の座標
PhotoScan	Australisモデル	補正後→補正前	fで正規化	中心からのオフセット
Pix4D	Australisモデル	補正前→補正後	-fで正規化	主点位置の座標

- 測量協会 → Pix4Dの変換式
  - 線形の変換のため数学的に作成可能
- 測量協会 → PhotoScanの変換式
  - 非線形の変換のため数学的に不可能なので，別途アルゴリズムの検討が必要

測量協会のレンズ補正パラメータをSfMソフトに入力する場合，変換式で変換してから入力する必要がある

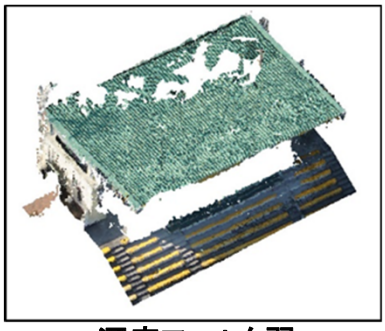
MVS処理の比較

PhotoScanの フィルタリング処理

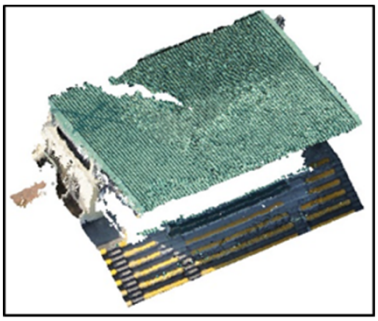
- 精度比較する上で把握しておかなければならない重要な項目
- 深度フィルタ（Depth filtering modes）
  - PhotoScanにのみ存在する処理
  - 3次元点群生成時に行っている
  - 強いフィルタリングを行うほど，信頼性の高い点だけが残る代わりに細部の点群が失われる可能性がある
  - デフォルトでは「強」が選択されているが，細部の点群が必要であれば「弱」を選ぶことが推奨されている．「無効」も設定できるが，ユーザマニュアルで推奨されていない．
    - 「無効」は精度がかなり悪かったので，「強」と「弱」でPix4Dと比較

深度フィルタの特性1

- 「弱」の方が，形状がおかしい点群が発生しやすい．
- 特に，細かい凹凸がある場所がおかしくなりやすく，見た目は「強」の方がかなり良い．



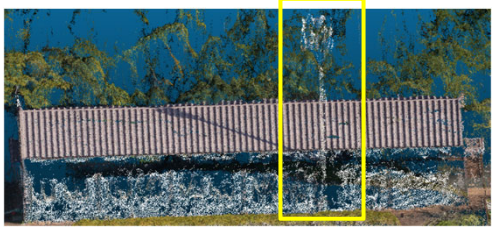
深度フィルタ弱



深度フィルタ強

深度フィルタの特性2

深度フィルタ強（照明が削除されている）

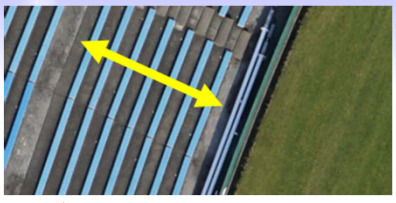


深度フィルタ弱（照明の点群が存在する）

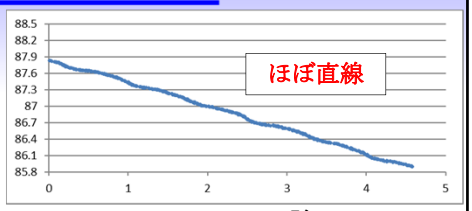
深度フィルタの特性2

- 点群が少ない構造物（特に平面になっていない電柱など）は深度フィルタ「強」で処理した場合，削除される可能性が高い．

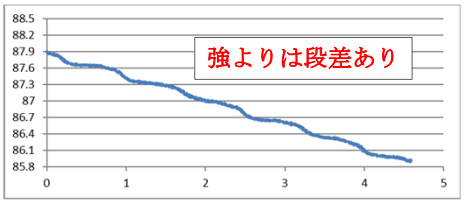
深度フィルタの特性3



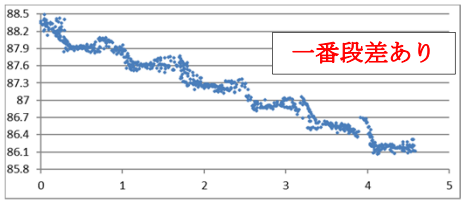
データ3（分解能：2.5cm）の
観客席画像（断面図作成部分）



PhotoScan強



PhotoScan弱



Pix4D

深度フィルタの特性3

- 「強」の方が，点群がより平滑化されるので見た目はよくなるが凹凸を若干表現できなくなっている．
  - 「強」，「弱」，「無効」のどれを選択しても，凹凸がかなり平滑化されてしまい変異抽出を行いにくい

Pix4DとPhotoScanの比較と考察

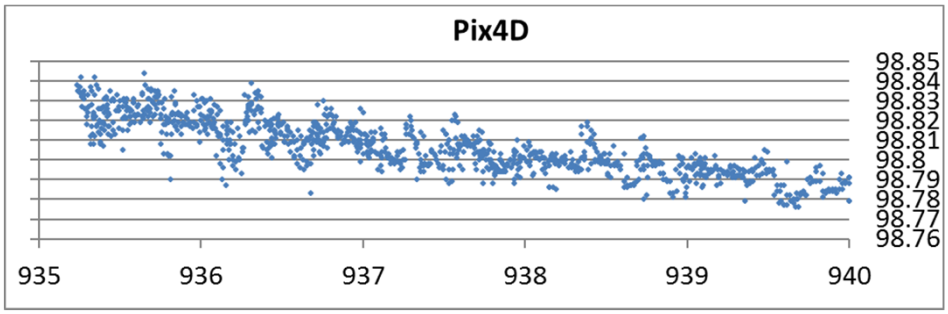
高さの揺れについて

点群の高さの揺れ

- データ2（地上分解能：約0.5cm）のコンクリート部分の点群の平面の揺れを調査

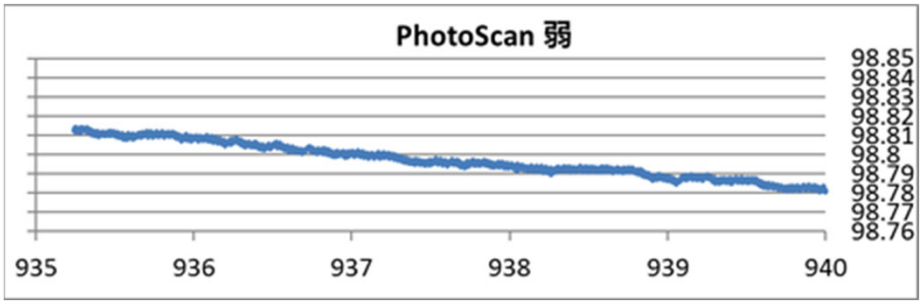
点群の高さの揺れ
～Pix4D～

- **Pix4D**
  - ▶ 約1cm（2ピクセル程度）の揺れが生じる

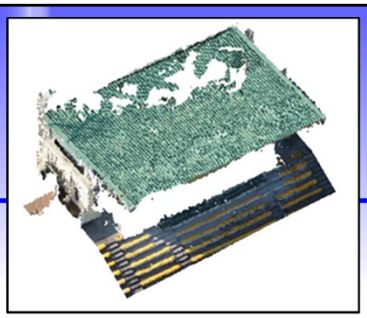


点群の高さの揺れ
～PhotoScan～

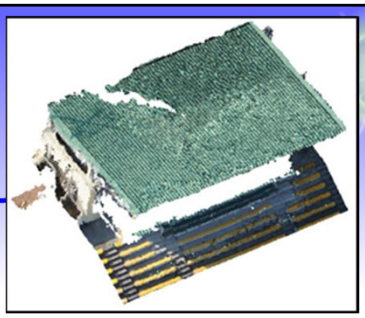
- **PhotoScan**
  - ▶ 凹凸が出やすい「弱」のフィルタでも揺れはない
  - ▶ 写真測量以外に平面形状の特性などを用いて点群を生成している可能性大



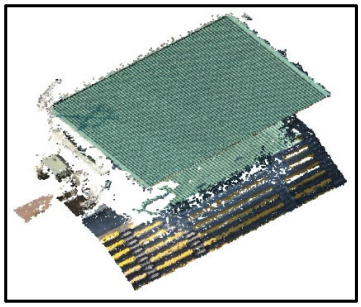
細かい凹凸の変異について



PhotoScan（深度フィルタ弱）

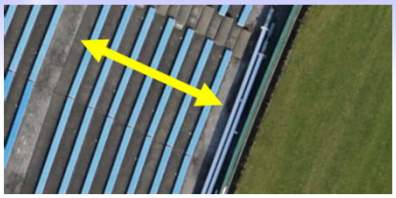


PhotoScan（深度フィルタ強）

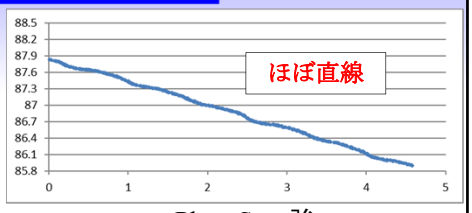


Pix4D

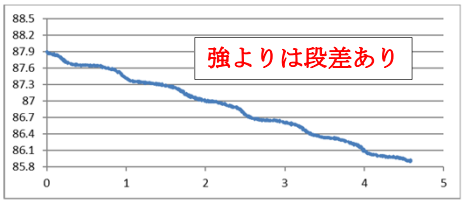
変異が平滑化される例



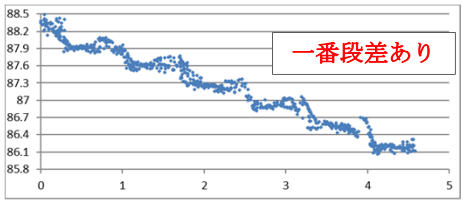
データ3（分解能：2.5cm）の
観客席画像（断面図作成部分）



PhotoScan強



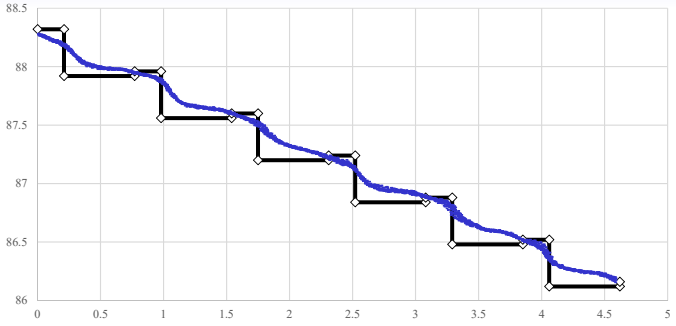
PhotoScan弱



Pix4D

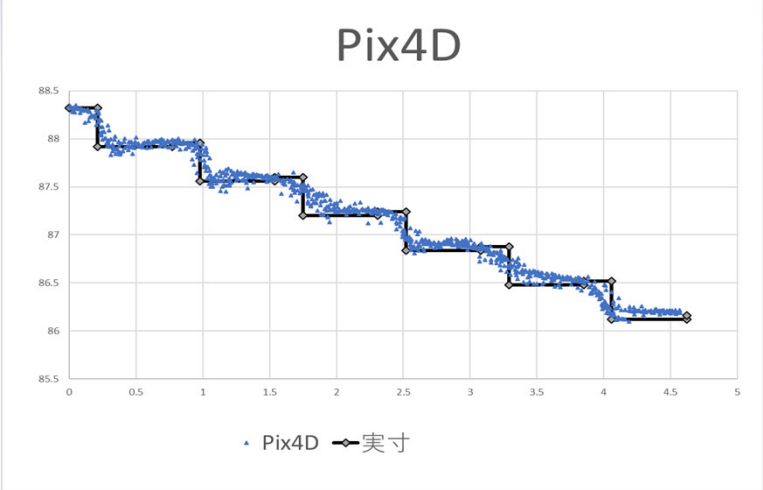
実寸との比較
（PhotoScan弱）

PhotoScan



PhotoScan ―◇―実寸

実寸との比較
（Pix4D）



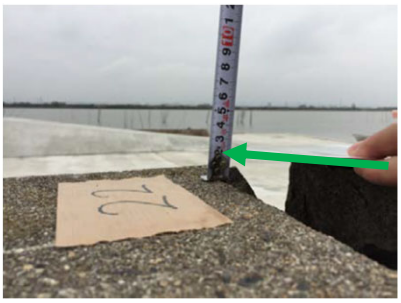
変異抽出
～細かい凹凸～

- **Pix4D**
  - 点群の高さ精度の揺れ（地上分解能×2ピクセル程度）以上の変異は抽出可能
  - 揺れより小さい変異を抽出したい場合は，Ransac法などで，揺れを取り除く必要がある
- **PhotoScan**
  - 上手く形状を表現できない場合がある
  - 見栄えは良くても平滑化されてしまうため，変異が抽出できない場合がある

その他の変異の抽出について

小さい変異抽出
～天端の段差～

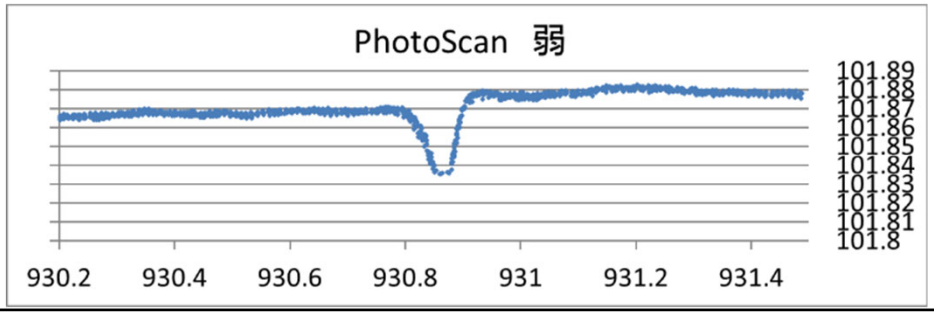
- データ2（地上分解能：約0.5cm）の堤防の段差2cmの部分について，検証した．



堤防の天端の段差（2cm）
※板が見にくいため画像上に線を描画

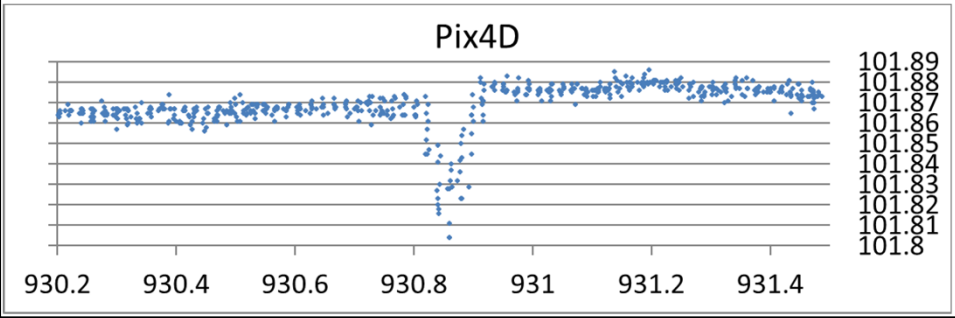
小さい変異抽出
～天端の段差　PhotoScan～

- 段差が出やすい深度フィルタ弱でも段差が1cmになっている
  - 上下とも地上分解能（0.5cm）分丸まっている
  - 地上分解能に近い数値の場合は，変異が消える可能性がある



小さい変異抽出
～天端の段差　Pix4D～

- 2cmの段差を表現できている点もあるが，上下1cmの揺れがあるため段差1cm～2cmと判断せざる負えない
  - 高精度に取得するには，Ransacなどで点群が多い箇所などを抽出して揺れを削除するなどの点群処理を適用する必要がある



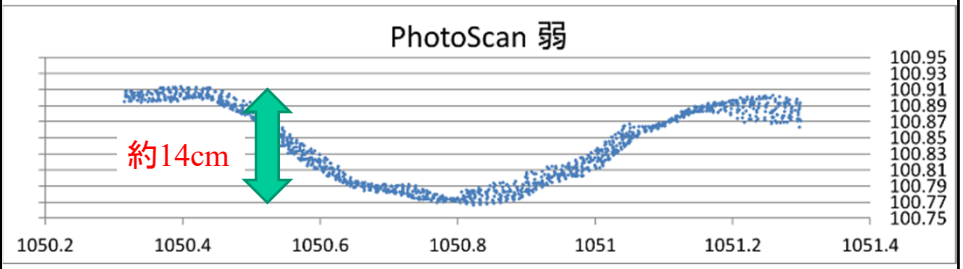
大きい変異抽出
～地面の窪み～

- データ2（地上分解能：約0.5cm）の地面の段差15cmの部分，幅70cmについて，検証した．

地面の窪み（深さ15cm，幅70cm）

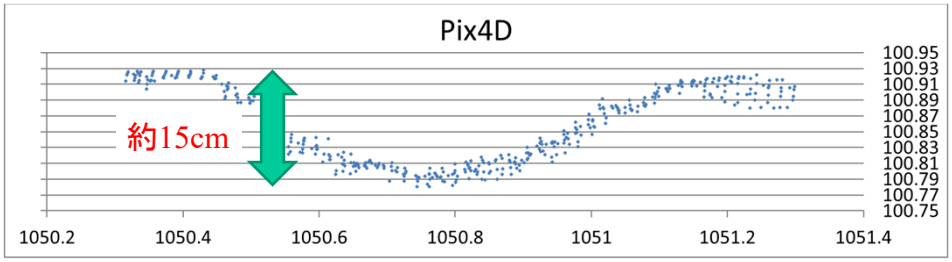
大きい変異抽出
～地面の窪み　PhotoScan～

- 段差は約14cmであり，小さい変異と同様に，地上分解能（0.5cm）×2の1cm丸まっている．
  - 地上分解能より，かなり大きい変異であれば抽出可能



変異抽出
～地面の窪み　Pix4D～

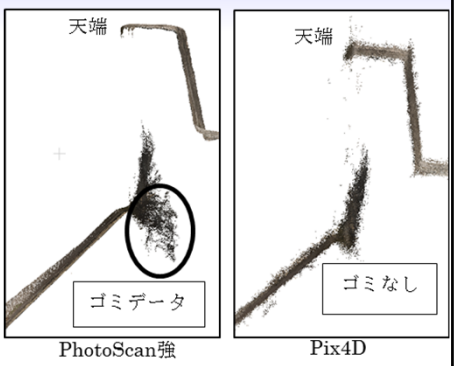
- 15cm±揺れ1cmとなっているため，小さい変異と同様に，より高精度を求めるのであれば，別途点群処理が必要になってくる．



点群の発生箇所について

点群の発生箇所
～点群のごみデータ～

- **PhotoScan**
  - 対応点マッチング以外に平面形状などを利用していると想定されるため，**写真に写っていない箇所にもゴミの点が多く発生する**.
- **Pix4D**
  - 1点1点対応点マッチングして写真測量をしていると想定されるため，**画像に写っていない部分に点が発生しにくい**.



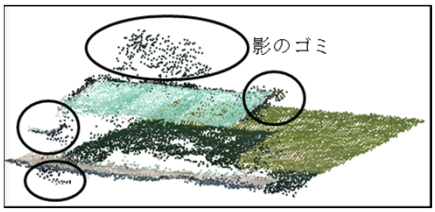
PhotoScan強　　Pix4D

点群の発生箇所（堤防）

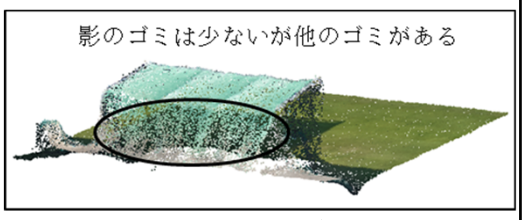
影について

影について

- **Pix4D**
  - 一般的な写真測量と同様に，撮影位置や角度によって形状が異なる影に若干弱いことがわかった
- **PhotoScan**
  - 平面の形状特性などを利用していると思われるため，影の影響は受けにくい．



Pix4D



PhotoScan強

まとめ（Pix4D）

- 1点，1点対応点取得し，写真測量で行っている
  - 高さ精度に揺れがあるため見栄えは若干よくないが，変異を抽出できる．
- 点群精度が安定している．
  - 平面精度は安定している
  - 揺れはあるものの，ほぼすべての形状で安定的に3次元点群を生成でき，間違った点が影や水，ガラス以外に発生しにくい
- 影の点群精度が悪くなる．

まとめ（PhotoScan）

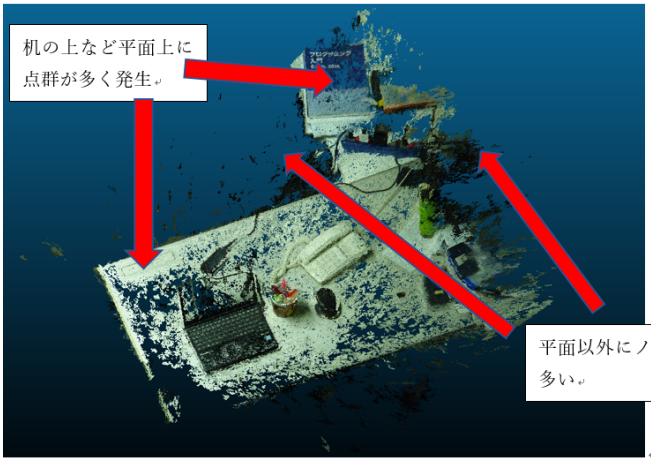
- 点群密度が多い
- 平面形状の特性を利用して点群を発生させている．フィルタリングも行っている．
  - 見栄えは良くCGなどに向いているが，変異が丸められ測量や変異抽出には形状によっては不向き
- 点群精度が悪くなる場所がある
  - 平面の延長上などに間違った点が発生する場合がある．細かい凹凸がある箇所が上手く計測できないことがある．など
- 平面以外の構造物が捨てられることがある．
  - 照明や電柱など

まとめ

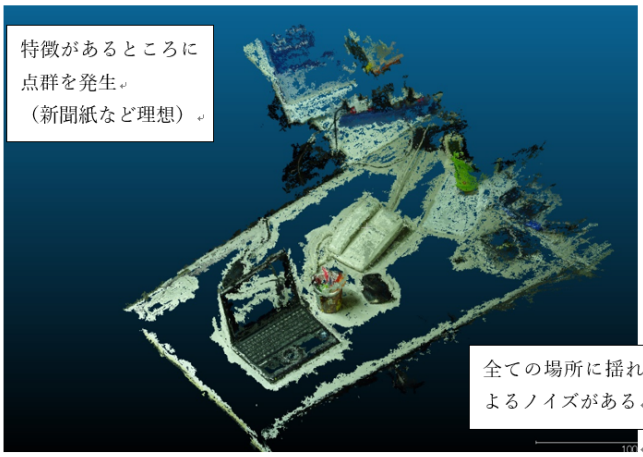
- **PhotoScan**
  - **詳細な形状が必要ないCGなど見栄え重視の場合に向いている**
- **Pix4D**
  - **見栄えはPhotoScanに劣るが，測量など詳細な形状が必要な場合に向いている**
    - **点群処理が別途必要な場合あり**

地上写真での実験

机（PhotoScan）



机（Pix4D）



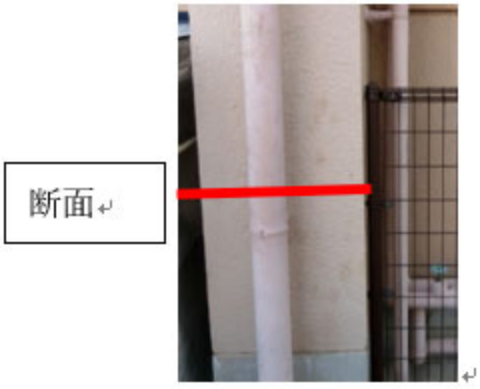
フィギュア（PhotoScan）

フィギュア（Pix4D）

屋外（PhotoScan）

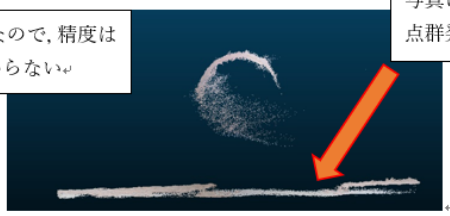
屋外（Pix4D）

パイプ



パイプ（PhotoScan）

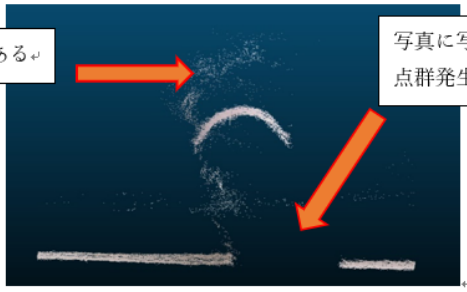
かなり近接なので，精度は
3つとも変わらない



写真に写っていない箇所に
点群発生している

パイプ（Pix4D）

ゴミデータがある



写真に写っていない箇所に
点群発生しない

地物の分類
（Classification）

Classificationとは

- 今までの点群は，座標と色情報のみであったが，そこに地物の分類（建物，道路など）を付与すること
  - 各点に分類番号が与えられる
- SfM/MVSソフトウェアに実装されている
  - Pix4Dは昨年から，PhotoScanはMetashapeに名前が変更された2019年1月から
  - Pix4D：地形，高い植生，建物，道路，車，人工物
  - Metashape：地形，植生，建物，道路，車

アルゴリズム（Pix4D）
～Metashapeもほぼ同じと思われる～

- 機械学習で行っている（ディープラーニングではない）
  - 学習方法は，GBT（Gradient Boosted Trees）
  - 幾何情報（X,Y,Z）と色情報（R,G,B）を入力データとしている
  - 幾何情報は固有値，固有ベクトルをベースに学習させるデータを作成している！！
  - 色情報は，RGBではなく，HSVで学習させる
  - マルチスケールで計算（徐々にダウンサンプリングしていく）したものを学習させていく．

学習パラメータ（Pix4D）
（論文より抜粋）

Covariance	Omnivariance	$(\lambda_1 \cdot \lambda_2 \cdot \lambda_3)^{\frac{1}{3}}$
	Eigenentropy	$-\sum_{i=1}^{3} \lambda_i \cdot \ln(\lambda_i)$
	Anisotropy	$(\lambda_1 - \lambda_3)/\lambda_1$
	Planarity	$(\lambda_2 - \lambda_3)/\lambda_1$
	Linearity	$(\lambda_1 - \lambda_2)/\lambda_1$
	Surface variation	λ_3
	Scatter	λ_3/λ_1
	Verticality	$1 - \lvert\langle [0,0,1], \mathbf{e}_3 \rangle\rvert$
Moments	1^{st} order, 1^{st} axis	$\sum_{\mathbf{p}\in\mathcal{S}_\mathbf{x}} \langle \mathbf{p} - \hat{\mathbf{p}}, \mathbf{e}_1 \rangle$
	1^{st} order, 2^{st} axis	$\sum_{\mathbf{p}\in\mathcal{S}_\mathbf{x}} \langle \mathbf{p} - \hat{\mathbf{p}}, \mathbf{e}_2 \rangle$
	2^{st} order, 1^{st} axis	$\sum_{\mathbf{p}\in\mathcal{S}_\mathbf{x}} \langle \mathbf{p} - \hat{\mathbf{p}}, \mathbf{e}_1 \rangle^2$
	2^{st} order, 2^{st} axis	$\sum_{\mathbf{p}\in\mathcal{S}_\mathbf{x}} \langle \mathbf{p} - \hat{\mathbf{p}}, \mathbf{e}_2 \rangle^2$
Height	Vertical Range	$z_{\max\{\mathcal{S}_\mathbf{x}\}} - z_{\min\{\mathcal{S}_\mathbf{x}\}}$
	Height below	$z_\mathbf{p} - z_{\min\{\mathcal{S}_\mathbf{x}\}}$
	Height above	$z_{\max\{\mathcal{S}_\mathbf{x}\}} - z_\mathbf{p}$
Color	Point color	$[H_\mathbf{x}, S_\mathbf{x}, V_\mathbf{x}]$
	Neighborhood colors	$\frac{1}{\lvert\mathcal{N}_\mathbf{x}(r)\rvert} \sum_{\mathbf{p}\in\mathcal{N}_\mathbf{x}(r)} [H,S,V]_\mathbf{p}$

各点において，
右の値を設定したマルチスケール単位で計算して，機械学習している

右の値数×マルチスケール数×点群数の値を入力データとして，機械学習している

SLAMについて

SLAMとは

- Simultaneous Localization And Mapping の略で，自己位置推定（撮影位置を求める）と，地図作成（3次元点群生成や平面抽出など）を同時にリアルタイムで行う技術
- 近年，スマホなどのタブレット型端末で実現できるため，非常に注目されている
  - Apple社の「ARKit」
  - Google社の「ARCore」 など

SLAMの種類

・カメラのみを利用する方法（VisualSLAM※1）
　メリット　：機種依存しない
　　安価である
　デメリット：精度が劣る
　手法　　：KUDANSLAM，PTAM，ARKit（Apple），ARCore（Google）など
　※1　VisualSLAM とは，カメラの映像から特徴を抽出して追跡することである．
　そのため，Tango なども VisualSLAM と呼ぶ可能性があるが，ここでは別とした．
・カメラ以外のセンサを利用する方法
　メリット　：高精度である
　デメリット：機種依存する
　　高価な場合がある．
　手法　　：Tango(Google)，HoloLens（Microsoft），LiderSLAM（レーザプロファイラ），など

SfM/MVSとSLAMの違い

- 基本的な概念は同じだが，リアルタイムかどうかが大きな違い！！
- SLAMはリアルタイムに行うために，計算を工夫している．
  - ▶9軸センサが重要！！
- 精度は，SfM/MVSソフトウェアの方が絶対に良い！！　※理論的に．

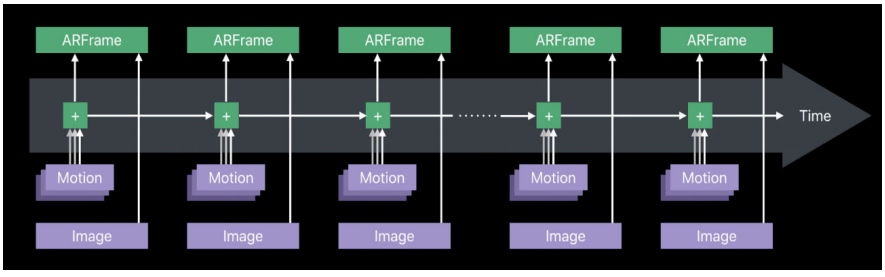
ARKitについて
～ARCoreもほぼ同じ～

ARKitは，Appleの製品のみで動作する
ARCoreは，多くの製品（Appleも含む）で動作する

画像等の引用先：
https://developer.apple.com/videos/play/wwdc2018/610/

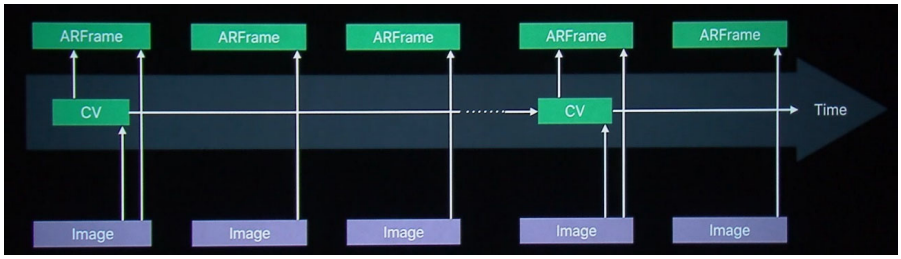
アルゴリズム（1）
～9軸センサを用いた自己位置推定～

- 9軸センサ（加速度，ジャイロ，地磁気）を用いた，全フレームにおけるカメラの位置と角度を算出する．
  - センサフュージョン（3つのセンサの結合）には，クォータニオン（4元数）を用いたMadgwickフィルタを用いるのが今は主流

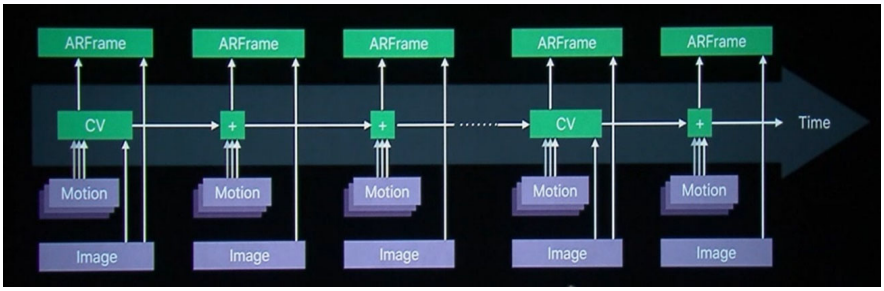


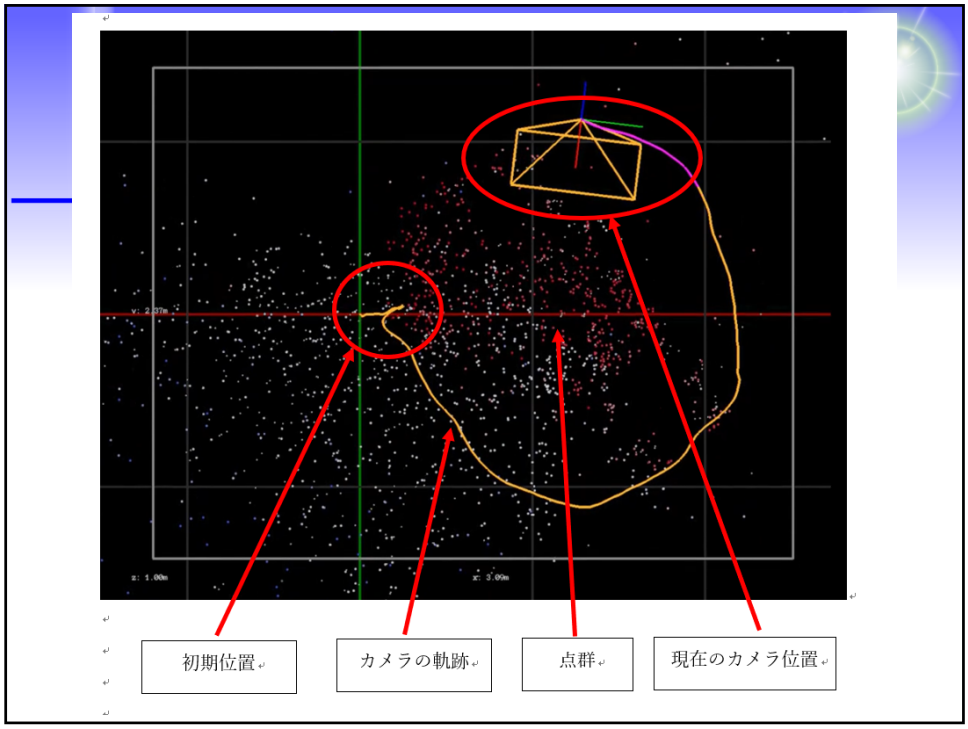
アルゴリズム（2）
～写真測量で自己位置を補正～

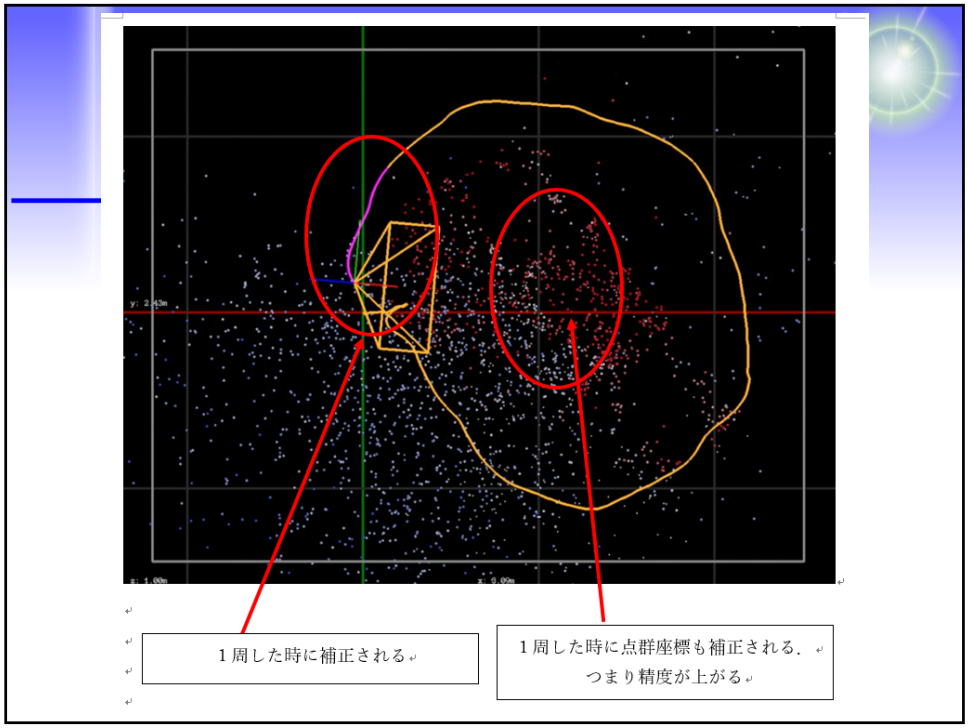
- 理想は写真測量で自己位置を計算したいが，リアルタイムではできない．そのため，リアルタイムは9軸センサに任せ，写真測量を補正に利用している．
- つまり，ARKitなどのSLAMで発生させた点群は，随時補正されるため，移動する．
  - 周回で撮影して，初めの位置に戻ったときなども補正される



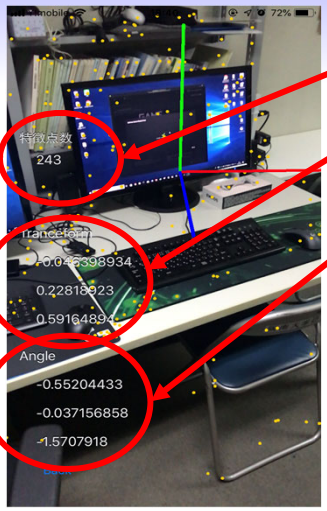
全体のフロー




初期位置
カメラの軌跡
点群
現在のカメラ位置


1周した時に補正される
1周した時に点群座標も補正される．
つまり精度が上がる

ARKitを用いた SLAMの実行結果



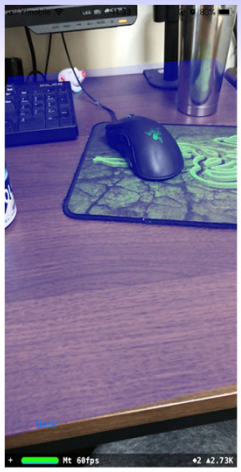
発生させた点群の数

カメラ位置

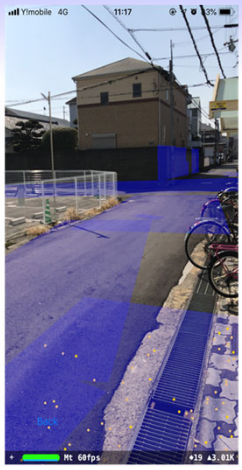
カメラの角度

上記の値などを毎フレーム簡単に取得できる！！
※補正されてそれぞれの値が移動する点に注意が必要

平面抽出もできる

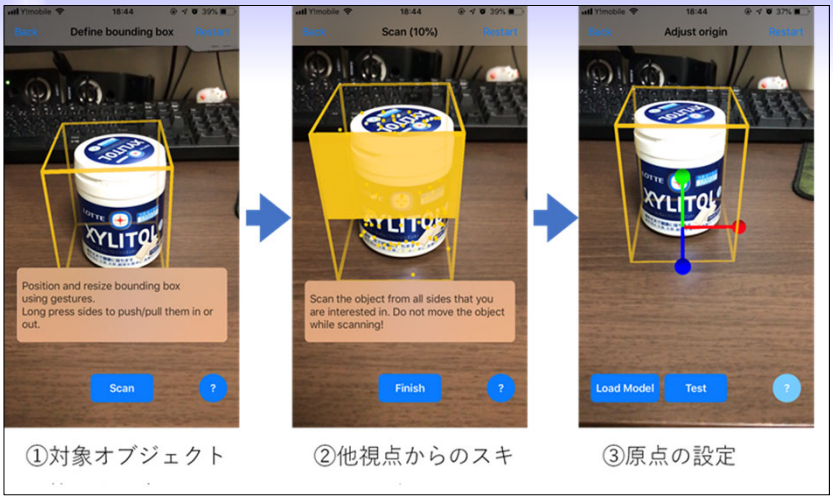


テーブルの平面検出



屋外での平面検出

物体検出もできる



①対象オブジェクト　②他視点からのスキ　③原点の設定

ご清聴ありがとうございました