基礎輪講2週目

Kinectの話

斎藤英雄研究室

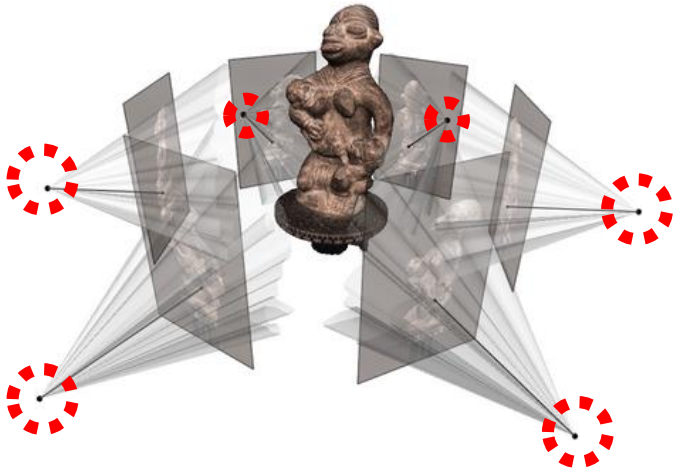
中山　祐介

3D Computer Vision

カメラから3次元の情報を取得

3次元再構築：2次元の画像から3次元形状の復元

多視点画像群†



3次元再構築結果†

3次元の情報を持つ画像ってないの？

距離画像

Depth（距離）の情報を持った画像

→各画素の部分に距離の値が入る.

距離画像

距離画像を取得する機器が必要

距離画像センサ

実世界の距離情報を取得できる.

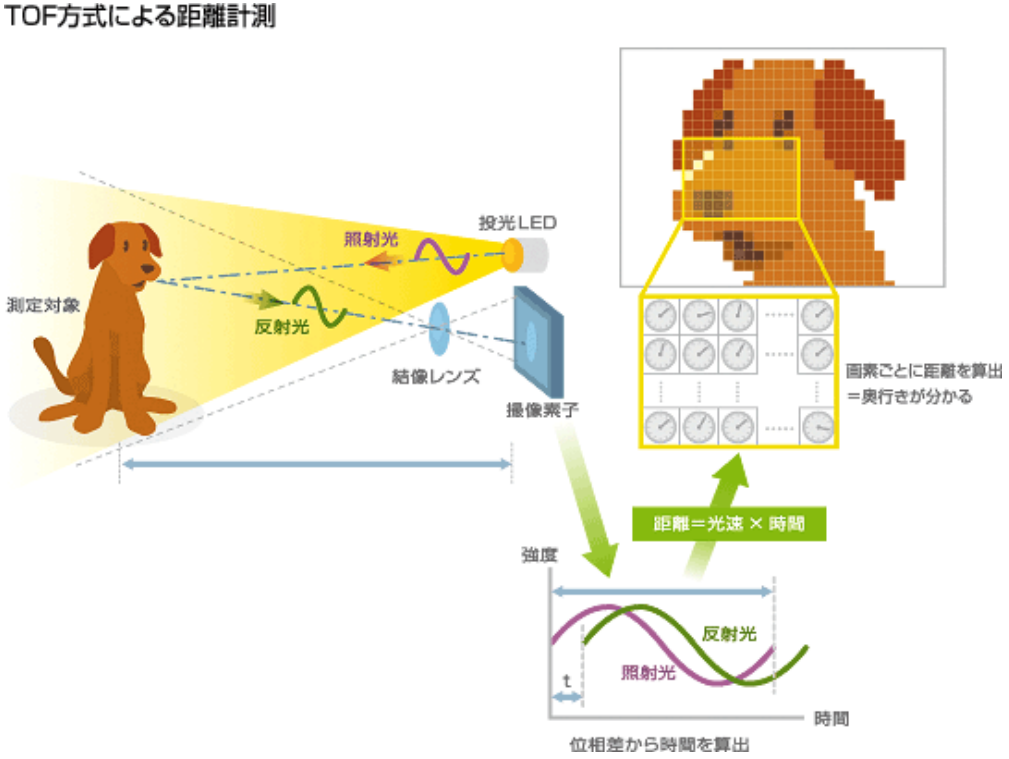
主に2つの方式

①TOF(Time of Flight)方式

②パターン照射(Projector-Camera)方式

TOF(Time of Flight)方式

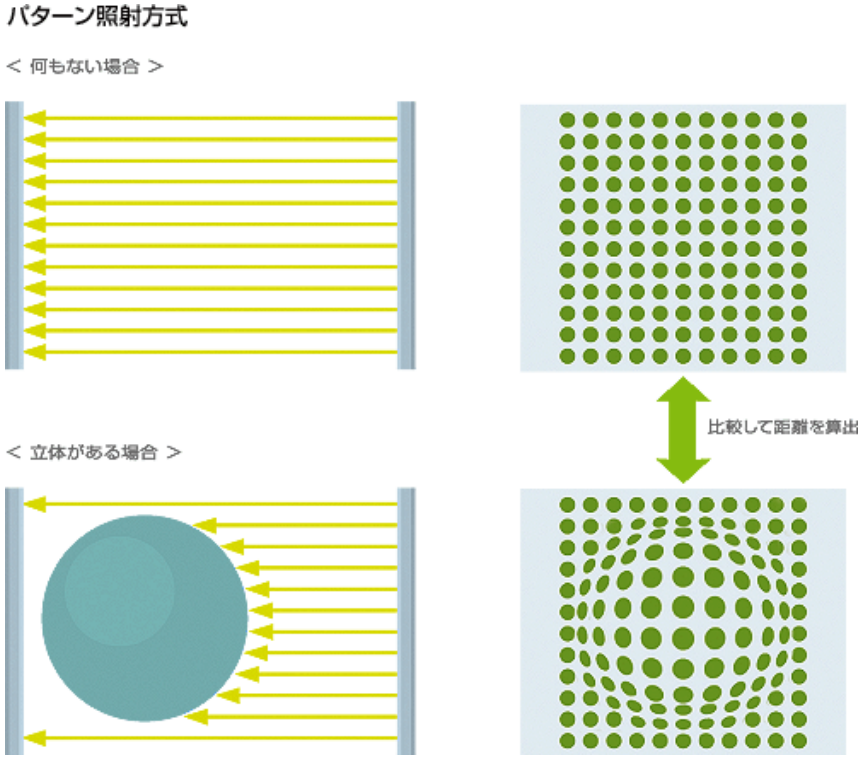
レーザーが対象まで往復するのにかかる時間(Time of Flight)から距離を計測



http://www.tdk.co.jp/techmag/knowledge/201102u/

パターン照射(Projector-Camera)方式

あらゆるパターンのレーザー光線を当て
反射する光のパターンのひずみで距離を測定



http://www.tdk.co.jp/techmag/knowledge/201102u/index2.htm

距離画像センサの問題点

→技術自体は古くからあるが非常に高価なものが多い.

距離画像センサの例

→もう少し安いセンサが欲しい！

コンシューマデプスセンサ

距離情報を取得する低価格なセンサ

→各製品2万～5万程度

→低価格なため広く普及し研究される.

→中でもKinectは数多くの研究に用いられている.

Kinect

もともとは，イスラエルのPrimesense社が製造しているデプスセンサ（Primesensor）

→Microsoft社に技術提供

→MicrosoftがXbox 360のゲームコントローラ（Kinect）として販売

→安価なデプスセンサとして利用可能なため広く普及

（USBでPCと接続可能→ハッキングしやすい！）

Kinectの仕様

- Depth解像度：640×480 px
- RGB解像度：1600×1200 px
- 最大60fps
- 撮影範囲：0.4~3.5m (near モード)

Kinectの原理

パターン照射方式を採用

Kinectの投影光

https://www.youtube.com/watch?v=nvvQJxgykcU

Kinectの種類

研究室には2つのKinectがあります！

 Kinect for XBOX360	・本来はXbox 360用 ・商用利用はNG ・開発時や個人用途に限りKinect for Windows SDK（Kinect用の開発キット）で利用可能 ・Nearモード使用不可 ・¥11,000くらい
 Kinect for windows	・windows PCに対応 ・商用利用OK ・Nearモード使用可 ・¥24,000くらい ・普通はこっちを使いたい

Kinectの用途

①距離情報の取得

→3次元点群処理

→Computer Vision的な使い方

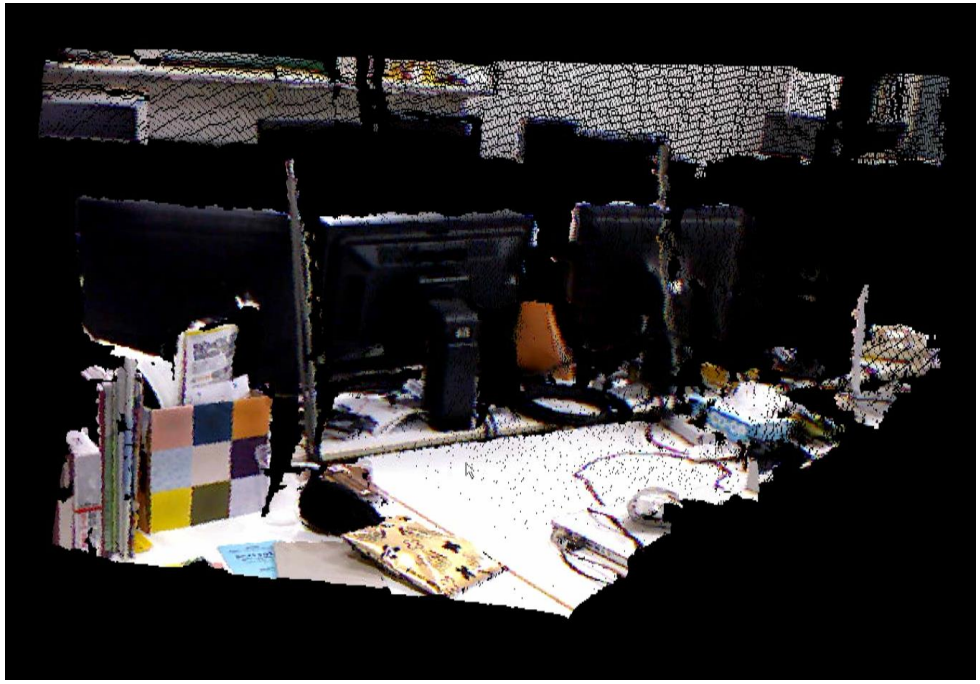
②人物姿勢のトラッキング

→NUI（Natural User Interface）等に利用

→Interaction的な使い方

距離情報の取得

Kinectにより3次元点群を取得

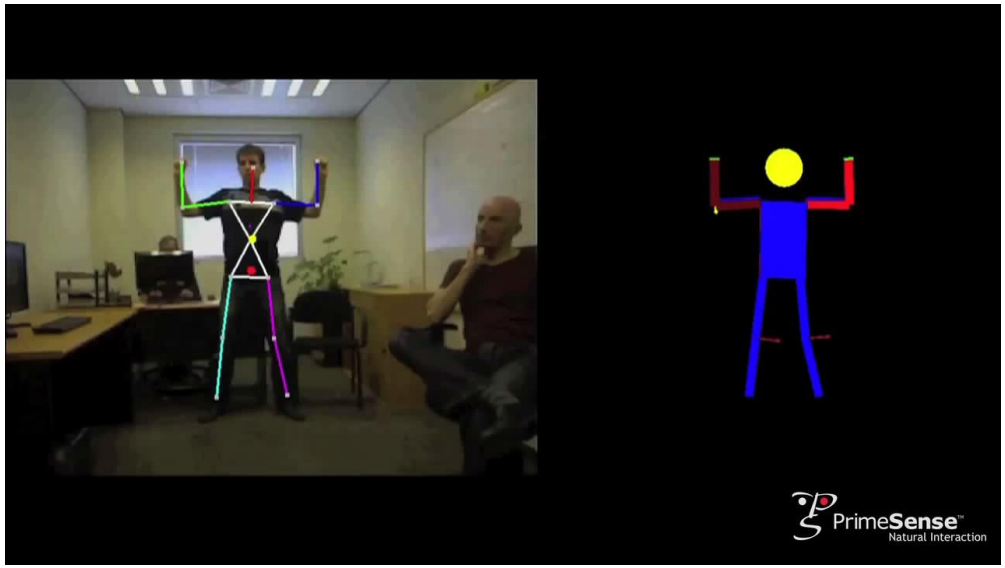


3次元点群の表示

→PCL（3次元点群処理ライブラリ）を使おう！

人物姿勢のトラッキング

体の各部位の推定→スケルトン（ボーン）のトラッキング



スケルトントラッキング

→様々なパターンの人物姿勢を機械学習し人を識別

→開発キットを使おう！

Kinectの開発キット

①OpenNI：PrimeSense社らが開発しているAPI群

→OpenNI（API），NITE（ジェスチャー認識ライブラリ）
Sensor（デバイスドライバ）から構成されている.

②Kinect for Windows SDK：Microsoft社が開発したSDK

→こっちが公式

※OpenNIとSDKは競合します！！

Kinectプログラミングのために

PCLとOpenNI，NITEをインストールしましょう.

PCL(http://pointclouds.org/downloads/windows.html)

→ Windows MSVC 2010 (64bit) PCL 1.6.0 All-In-One Installer を選ぶ

→ Do not add PCL to the system PATH を選択

→ OpenNIのインストールも自動的に始まる.

→ Sensor Kinectのインストールも自動的に始まる.

→ インストール終了後，環境変数のpathを

C:¥Program Files¥PCL 1.6.0¥binに通してPCを再起動

NITE：インストールファイルを配布

動作確認

①Kinectを接続して，

C:¥Program Files¥PCL 1.6.0¥bin内の

openni_voxel_grid_release.exeを起動し確認

②C:¥Program Files¥OpenNI¥Samples¥Bin64¥Release内の

NiUserTracker64.exeを起動し確認

課題

①KieorinWeek2Kinect1.zip

→KinectクラスのShowImage関数とSavePointcloud関数を作る.

②KieorinWeek2Kinect2.zip

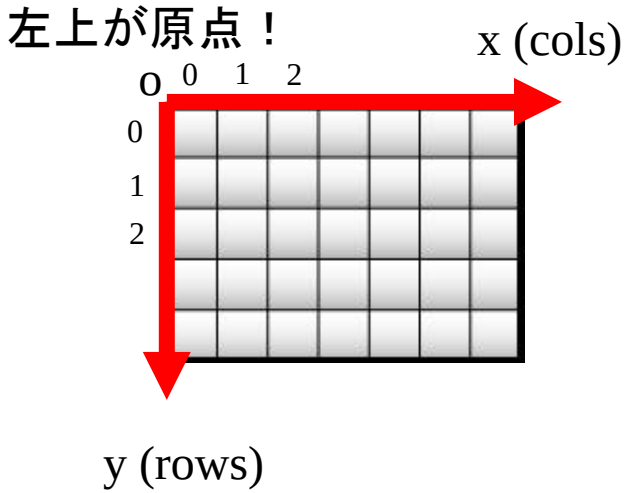
→実行して動作確認.（余力があればコードを読んでみる）

※プロパティシートを自分用に書き換えること！！

Kinectクラスのカラー画像

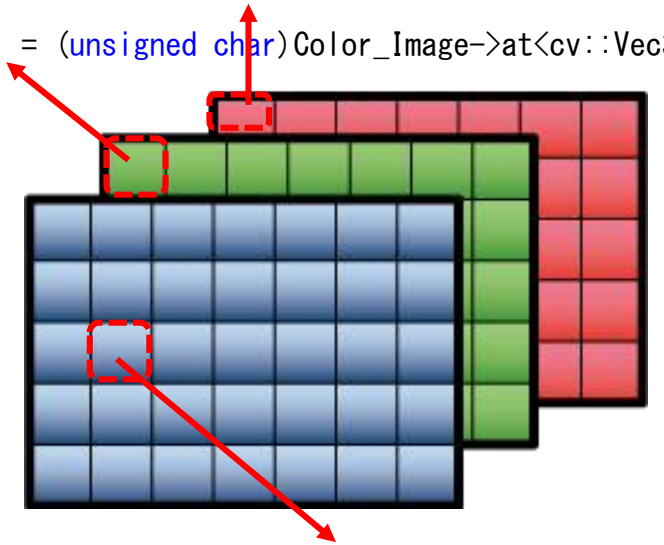
xn::ImageMetaData*　ImageMD（KinectからのRGB情報）

→　cv::Mat_<cv::Vec3b>*　Color_Image　に登録



Mat型の概要

```
(Rの明度値) = (unsigned char)Color_Image->at<cv::Vec3b>(0, 0).val[2];
(Gの明度値) = (unsigned char)Color_Image->at<cv::Vec3b>(0, 0).val[1];
```



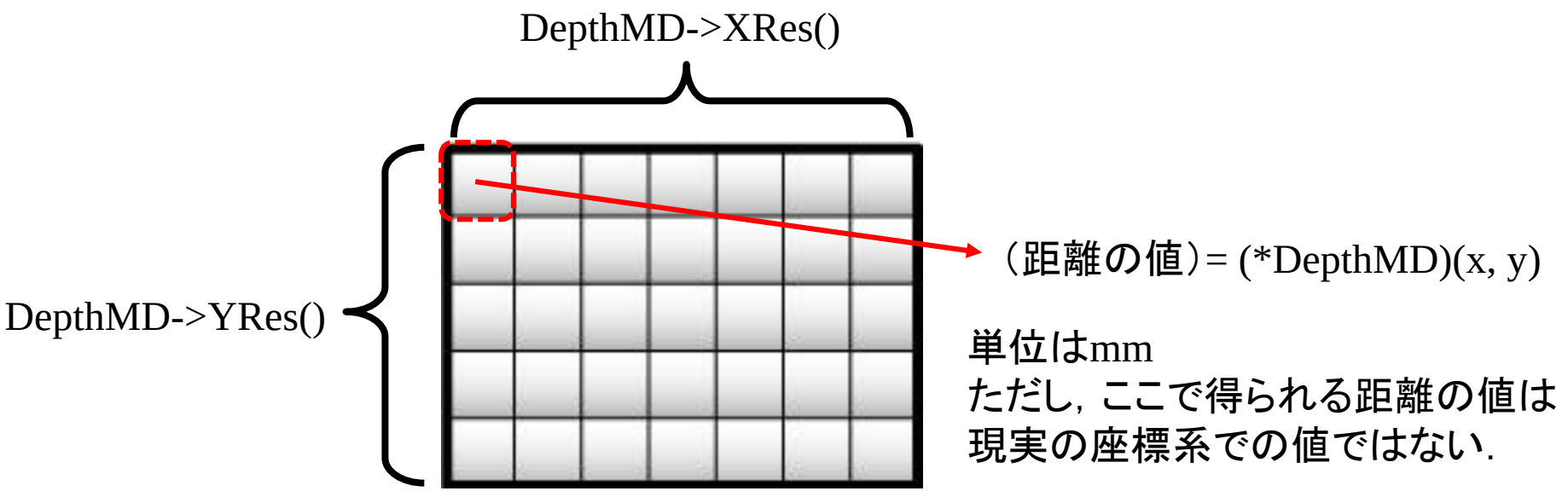
```
(Bの明度値) = (unsigned char)Color_Image->at<cv::Vec3b>(2, 1).val[0];
```

Color_Imageの概要

Kinectクラスの距離画像

xn::DepthMetaData* DepthMD （Kinectからの距離情報）

→ cv::Mat_<cv::Vec3b>* Depth_Image に登録



Depth_Imageの概要

カメラ座標系です！

Kinectの座標系変換

距離画像の値から現実の座標(~~世界座標系~~)に変換

変換用の3次元座標　XnPoint3D proj, real　を用意

proj(画像座標系+距離の値)

proj.X = (画像のx座標);
proj.Y = (画像のy座標);
proj.Z = (距離の値);

real(カメラ座標系)

real.X = (x座標[mm]);
real.Y = (y座標[mm]);
real.Z = (z座標[mm]);

KinectクラスのProjectToReal関数を使って変換