画像による 3 次元計測
実験手順書

1. 目的

- 画像計測の実際に触れ，コンピュータビジョンの要素技術を理解することを目標とする．映像センサおよび距離センサから画像を取得し，3 次元空間中の物体の形状等を計測する．
- 実験の目的を達成する方法を考え，グループのメンバーと協力して計画的に実施する姿勢を養う．
- 適切な報告書の書き方を学ぶ．

2. スケジュール

3 人程度で 1 グループを構成し，共同で実験する．レポート作成に備え，メモやデータ保存など，こまめに記録しながら実験を進めること．進捗を☑チェックしていこう．

- 1 日目：
  - ☐ 画像による 3 次元計測の予習事項を確認する．
  - ☐ 実験装置，センサの利用方法を確認する．
  - ☐ 基本課題開始．レポートのための記録をとる．
  - ☐ 各自，LACS の掲示板に作業記録を投稿する．
- 2 日目：
  - ☐ 基本課題を完了させる．
  - ☐ 発展課題の実施計画を立てる．
  - ☐ 各自，LACS の掲示板に作業記録を投稿する．
- 3 日目：
  - ☐ 各自，基本課題までのレポートを提出し，担当教員または TA の添削を受ける．
  - ☐ 発展課題の実施計画書を提出し，担当教員の許可を得た後，計画を実施する．
  - ☐ LACS の掲示板に発展課題のプログラムを投稿する．
  - ☐ LACS の掲示板に作業記録を投稿する．
- 4 日目：
  - ☐ 各自，最終版のレポートを持参し，試問を受け，提出する．
  - ☐ 各自，最終版のレポートの PDF ファイルを LACS に提出する．

実験 4 日目に最終版のレポートを提出できるように，主な実験の作業は 3 日目までに終了し，必要なデータや資料をそろえること．

3.　実験装置

3.1 ハードウェア

- RealSense

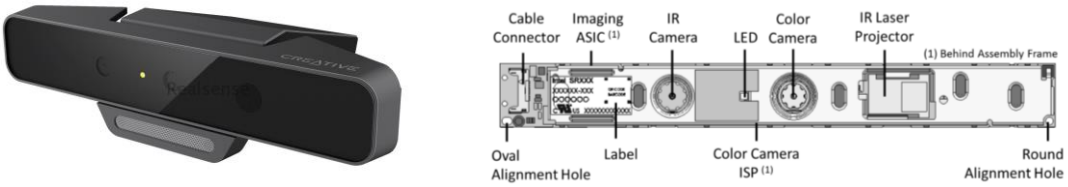

図 1　RealSense SR300. 左 : 外観, 右 : 内蔵のセンサの一部 (製品データシートから抜粋).

RealSense（リアルセンス）は，ジェスチャー等による PC の入力装置として Intel から提供されている 3 次元計測デバイスと開発キットである．RealSense デバイスは赤外ライトのプロジェクタと赤外カメラを備えており，能動的ステレオ法で 3 次元物体の形状を計測する．可視光カメラとマイクも搭載しており，PC に USB 接続して計測データを取得できる．

- PC 一式

図 2　PC 一式．左：PC（Dell Mobile Precision 3510），右：起動用 USB メモリ（128GB）.

- PC（Dell Mobile Precision 3510，Core i7-6820HQ 2.7GHz，8GB，15.6 型 1,920×1,080）
- ワイヤレスマウス
- 起動用 USB メモリ（SanDisk SDCZ88-128G Extreme Pro，Ubuntu 16.04LTS）

3.2 ソフトウェア

起動用 USB メモリには Ubuntu 16.04LTS がインストールされている．RealSense センサを接続して実験できるように，Cross Platform API（librealsense），OpenCV，MeshLab 等がインストール済みである．

- Intel RealSense Cross Platform API (librealsense)
  （https://github.com/IntelRealSense/librealsense）
  Linux，Windows，Mac OS X で RealSense デバイスを使用可能にするドライバとライブラリ. RealSense デバイスからカラー画像, 深度画像, カメラパラメータ等を取得する機能を提供する．Intel RealSense SDK とは異なり，コンピュータビジョンの機能やジェスチャー認識等のアプリケーションは含まれていない.

- OpenCV（http://opencv.org/）
  オープンソースのコンピュータビジョン向けライブラリ．画像に関する入出力，画像処理，画像構造解析，モーション解析，パターン認識，機械学習等，多くの関数を提供する．インテル（’98～’08），Willow Garage（’09～’12）が開発し，現在は Itseez（http://itseez.com/）が管理している．

- MeshLab（http://www.meshlab.net/）
  3 次元の点群（ポイントクラウド）を可視化するソフトウェア．GUI による点群の編集や, 表面再構成（surface reconstruction）も可能である．

- 実験用のプログラムおよび資料

`/home/user/Experiment/`	以下のディレクトリにファイルがある.
`*.cpp`	C++ソースコード（ 基本課題(2), (3), (4) ）
`Sample_librealsense/*`	librealsense のサンプルプログラム（ 基本課題(1) ）
`Document/*`	資料や説明書.

3.3 その他の器具

- 巻尺，計測対象物体等．壊さないように，丁寧に扱ってください.

4. 実験手順

下記に従って「基本課題」と「発展課題」に取り組む．実行・確認したことはすべてレポートで報告する．レポートに書かれていない事項は実行・確認しなかったものとして評価するので注意せよ．済んだ実験には☑チェックを記入しよう．

実行【グループ】と記載されている課題はグループで共同して行ってよい．それ以外は必ず各自で実行し，グループのメンバーで確認し合うこと.

4.1 起動と終了【グループ】

- ワイヤレスマウスの USB レシーバ，起動用 USB メモリを PC に挿入する.
- 電源を接続し，Ubuntu を起動する．もし Windows が起動してしまったら，何もせ

ずに終了し，起動しなおすこと．

- ログインしたら，RealSense デバイスを USB 接続し，課題に取り組む．
  ユーザ名：user　　パスワード：password
- 課題の作業が終わったら，画面右上の終了ボタンからシャットダウンする．

4.2　基本課題

(1) librealsense のサンプルプログラムの実行と考察　　（～14:30）

Intel RealSense Cross Platform API (librealsense) Developer's Guide
https://software.intel.com/sites/products/realsense/camera/developer_guide.html
の Samples 以降の解説を読みながら，以下のサンプルプログラムを実行する．実行結果の報告だけでなく，RealSense から得られるデータの種類や仕様，性質について調査・考察する．

☐ `cpp-capture`

- ➢Q１　何のカメラが搭載され，どのような画像が得られるか．
- ➢Q２　赤外（IR）画像は，どこが明るく・暗くなる性質があるか．理由と共に性質を 3 つ以上挙げよ．
- ➢Q３　深度画像には画素の欠損がある．どこがなぜ欠損しやすいか．
- ➢Q４　カラー画像と深度画像は視点と視野が異なる．どのように違うか．

ヒント：　Alt + PrtSc で実行画面をキャプチャし，レポートで図説するとよい．

ヒント：　実行中に 'c' や 'd' を押してみる（ソースコード 75～76 行目）．

☐ `cpp-config-ui`

- ➢Q５　右上の "Start Capture" を押し，MOTION RANGE と CONFIDENCE THRESHOLD のスライドを動かすと，何がどう変わるか．

ヒント：　'motion vs range tradeoff'（Camera Specifications - GitHub）

(2) カラー画像と深度画像の表示，保存　　（14:30～16:00）

Intel RealSense Cross Platform API (librealsense) Documentation: OpenCV Tutorial
https://github.com/IntelRealSense/librealsense/blob/master/doc/stepbystep/getting_started_with_openCV.md
に掲載されているソースコードを利用する．

☐ Displaying Color Frame のソースコードを `BGR_sample.cpp` として保存する．

- ソースコードを読み，RealSense デバイスからカラー画像と深度画像を取得・表示する方法，および OpenCV を用いて画像を表示する方法を理解する．
- コンパイル，実行し，動作を確認する．

```
$ g++ -std=c++11 BGR_sample.cpp -lrealsense -lopencv_core
-lopencv_highgui -o BGR
```

☐ `BGR_sample.cpp` に以下の変更を施して `BGR_sample_save.cpp` を作成する．

- `imshow` の行を `imwrite("color00.png", color);` に変更する．

- waitKey(0); を削除する.
- コンパイル方法：

```
$ g++ -std=c++11 BGR_sample_save.cpp -lrealsense -lopencv_core
-lopencv_highgui -lopencv_imgcodecs -o BGR_save
```

□ BGR_sample.cpp に以下の変更を施して BGR_sample_show.cpp を作成する.

- for 文を while(waitKey(1) != 'q') に変更し，imshow まで{}で囲む.
- nameWindow と waitKey の行を削除する.
- コンパイル方法：

```
$ g++ -std=c++11 BGR_sample_show.cpp -lrealsense -lopencv_core
-lopencv_highgui -o BGR_show
```

□ BGR_sample_show.cpp から以下のように BGR_Depth_show.cpp を作成する.

- dev->start() の前に，下記のように深度画像ストリームを設定する.

```
dev->enable_stream(rs::stream::depth, 640, 480,
                                    rs::format::z16, 30);
```

- 深度画像の画素値をメートル単位に換算するための定数を取得する.

```
float scale = dev->get_depth_scale();
```

- while 文中の適当な箇所に下記を追加する.

```
Mat depth(Size(640, 480), CV_16UC1,
   (void*)dev->get_frame_data(rs::stream::depth)); //(*1)
Mat gray_depth;
depth.convertTo(gray_depth, CV_8UC1, 255*scale, 0);
imshow("Depth Image", gray_depth);
```

- コンパイル方法：

```
$ g++ -std=c++11 BGR_Depth_show.cpp -lrealsense -lopencv_core
-lopencv_highgui -lopencv_imgproc -o BGR_Depth_show
```

➢ Q６　librealsense や OpenCV で提供されている関数を main 関数内で使っている．重要な関数の機能を調べ，main 関数の動作の流れをレポートで解説せよ.

➢ Q７　深度画像 depth は，画面表示できる濃淡画像 gray_depth へどのように変換されているか．gray_depth の画素値と深度の関係を説明せよ.

➢ Q８　(*1)の行で rs::stream::depth_aligned_to_color に変更すると，どのような深度画像が得られるか.

(3) 画像の点に対応する 3 次元空間の点の計算（逆透視投影）　（16:00～17:30）

□ ソースコード clickXYZ.cpp を読み，逆透視変換する関数 Zuv_to_XY を完成させる．また，コンパイル，実行し，動作を説明する.


図 2 多面体.

- □ 机や床など, 同一平面上にある異なる 3 点の座標を計測し記録する．また，この 3 点から，平面の法線の向きを表す単位ベクトル（単位法線ベクトル）を算出する．
- □ 図 2 のような適当な多面体の異なる 2 平面の法線をそれぞれ求め，面のなす角を求める．また，求めた値が適切であるか評価する．

ヒント：どんな計測も反復試行し，平均やばらつき等で統計的に評価すべきである．また，誤差は対象物の姿勢やセンサとの位置関係に依存しうる．

- ➢Q 9　「正確度（accuracy)」と「精度（precision)」とは何か.
- ➢Q10　誤差の原因は何か.

(4) ポイントクラウドの取得　　(13:00～15:00)

- □ ソースコード `PointCloud.cpp` を読み，逆透視変換する関数 `Zuv_to_XY` を完成させる．
- □ `PointCloud.cpp` をコンパイル，実行し，生成されたポイントクラウドファイル `xyzrgb00pc.ply` をエディタで開いて内容を確認する．
- □ MeshLab で `xyzrgb00pc.ply` を開いてポイントクラウドを可視化する．様々な視点からポイントクラウドを観察し，点の分布や欠落の様子を観察する．
- □ MeshLab の機能を使って 2 点間の距離を測ることができる．図 2 や図 3 のような適当な物体について大きさを計測し，実物の大きさと計測値を比較した「正確度」，計測値のばらつき「精度」を調べる．また，その傾向や原因を考察する．


図 3 適当な物体.

ヒント：プログラム中で深度画像の画素値をメートル[m]の値に換算している．

- ➢Q11　机や床などの平面を計測した点は厳密には平面状に分布していない．誤差の主な原因は何か．また，誤差を低減する技法を調べよ．

4.3　発展課題：平面の検出と応用

RealSense を用いて机や床などの平面を検出する．また，その応用として，物体の一部と平面（例えば適当な物体の頭頂と机）の間の距離を計測する．本課題の目標は，性能の高い方法をプログラミングすることではない．考案した方法の単純さと合理性，結果の再現性を示すことであり，次の(a)～(d)の観点からレポートを評価する．

(a) 実験前に実験計画書を所定の様式で作成していること
(b) 平面検出と距離計測の方法を適切に記述できていること
(c) 想定している設定や環境（適切に検出・計測できる条件）を明記していること
(d) 検出・計測の正確さの程度について具体的な予想と根拠，実験的な評価方法が示されていること

(1) 計画書の作成【グループ】　　(15:00～17:30)

□ 別紙の様式を用い，必要に応じて図や式などを使って記述する．

i) 平面を検出する計算手順（アルゴリズム）を記述する．アルゴリズムは以下の仕様を満たすものとする．

入力：深度画像 D，カラー画像 C（RealSense から毎秒 30 枚取得できる）
出力：平面の領域が着色されているカラー画像 A（例：図 4 左）
（平面に該当する画素の画素値を加工した画像）

これ以外のものを入出力に追加・工夫してもよい（例えばマウスによる入力や，平面の代表点，向き，誤差の目安等，平面の特定に役立つ何らかの情報）．アルゴリズムを読めば誰もが同一の仕組みをプログラミングできるように記述すること．記号や変数，式で具体的に記述した擬似コードを書くとよい．

ii) 平面と物体の一部の間の距離を計算する手順を記述する．
例 1：平面から最も遠い（または近い）物体の一点までの距離を測る（図 4 右）．
例 2：色と深度で特定した人体の一点と平面の距離を測る．

iii) 検出・計測の正確さの程度を具体的に示すための方法を書く．どのような設定で実行し，結果として何を記録すればよいか．

ヒント：(i)や(ii)では，平面の方程式や，点と平面の距離を計算する方法を思い出そう．

ヒント：計測して得た「数値」については正確度・精度を評価すべき．また，「検出」の誤りは，誤検出（false positive detection）と検出漏れ（false negative detection）の 2 種類があり，これらを評価すべき．

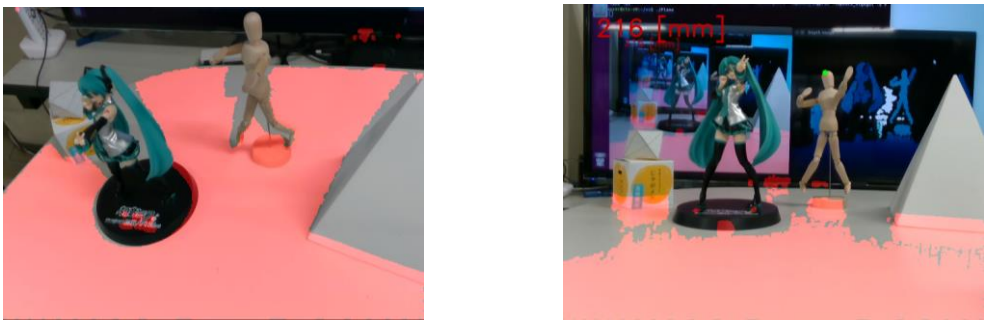

図 4　平面の検出と距離の測定への応用．左：平面を検出するアルゴリズム(i)の出力画像 A の例．右：平面から最も遠い物体の点（緑で示された点）までの距離[mm]を画像の左上に表示している．

(2) 実施　　(13:00～17:30)

□ 作成した計画書に従って実験を実施する．

- アルゴリズムを実装・実行し，結果を記録せよ．入力の深度画像 D とカラー画像 C，出力のカラー画像 A も保存すること．
- 結果を考察せよ．また，実験実施中に気がついたことを報告せよ．

5. その他

- わからないこと，知らないことがあったら放置せず，積極的に調べたり，教員や TA に相談しよう．
- 実験室では無線 LAN からインターネットを利用できる．実験中の調査やデータの保存等に活用してよい．
- 実験 4 日目終了後，起動用 USB メモリに保存したファイルやデータは消去される．必要であれば，インターネットや私物の USB メモリ等を利用して保存すること．

付録：本実験で便利な OpenCV の使用例

詳細は OpenCV のリファレンスマニュアル等を参考にすること．

① **`Z = D.at<unsigned short>(i, j);`**
D は unsigned short 型の要素を持つ OpenCV の行列型である．この例では，D の第 i,j 要素を Z に代入している（0≦i＜480，0≦j＜640）．

② **`A.at<Vec3b>(i, j) = Vec3b(127, 0, 0);`**
行列型のカラー画像 A の第 i,j 要素に暗い青（(青,緑,赤)=(127,0,0)）を代入する（0≦i＜480，0≦j＜640）．

③ **`Mat A(Size(640, 480), CV_8UC, Scalar(0,0,255));`**
縦 480×横 640 のカラー画像 A を行列型で確保する．この例では，全画素が赤（(青,緑,赤)=(0,0,255)）に初期化される．

④ **`imwrite("filename.png", A);`**
行列 A が PNG 形式の画像として保存される．

⑤ **`circle(A, Point(j, i), 8, Scalar(255,0,0), -1);`**
`// circle(画像，円の中心座標，半径，色，線太さ);`
行列型のカラー画像 A に円を描き込む．この例では，円の中心は画素 (i, j)（0≦i＜480，0≦j＜640），半径は 8，色は青，線の太さは-1（塗りつぶし）である．

⑥ **`minMaxLoc(D, &minD, &maxD, &minLoc, &maxLoc, M);`**
行列 D の要素の最小値（double minD），最大値（double maxD），最小値の要素の位置（Point minLoc），最大値の要素の位置（Point maxLoc）を求める．ただし，行列 M の要素が非ゼロとなっている D の要素についてのみ調べる（M は省略可）．例えば，行列 D の最小値は minD であり，その位置は minLoc.y 行 minLoc.x 列である．

⑦
```
Point3f pa, pb, pc, pd;
pa.x = 0.1;   pa.y = 0.2;   pa.z = 0.3;
pb.x = 0.8;   pb.y = 0.5;   pb.z = 1.2;
pc = pb - pa;    pd = pb + ba;
```
3 次元ベクトルの成分に代入し，和と差を計算できる．Point3f 型は，float 型のメンバ x, y, z を持つ．その他，Vec3f 型等，ベクトルの演算が定義されている型もある．リファレンスマニュアルの基本構造体を参照のこと．

⑧
```
putText(A, "abc", Point(300,400), FONT_HERSHEY_SIMPLEX,
                                4.0, Scalar(0,255,0), 12, CV_AA);
```
// putText(画像，テキスト，位置（左下），字種，スケール，色，線太さ，種類);行列型のカラー画像 A の位置(300,400)に緑色の文字列 "abc" を描き込む．