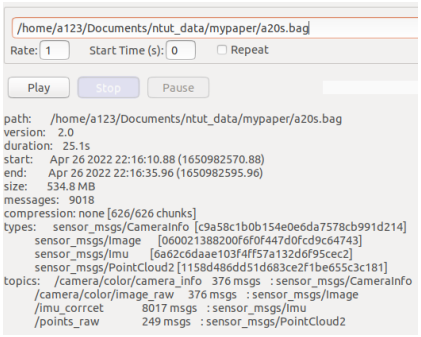
論文程式交接_達源

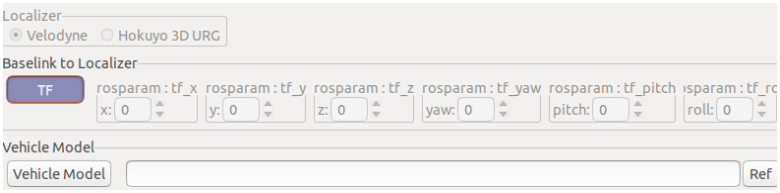
專案名稱：mypaper
GITHUB 網址：https://github.com/t108368123/mypaper
功能簡介：用於 2D-3D 外部參數校正

使用方法：

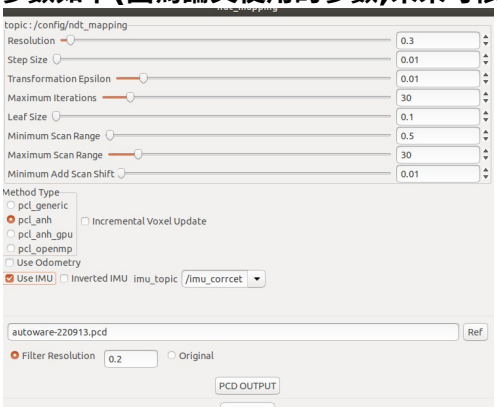
----------雙棋盤格校正（使用 a20s.bag 當範例）----------

1.用 autoware 播放 bag,TF 都設為 0

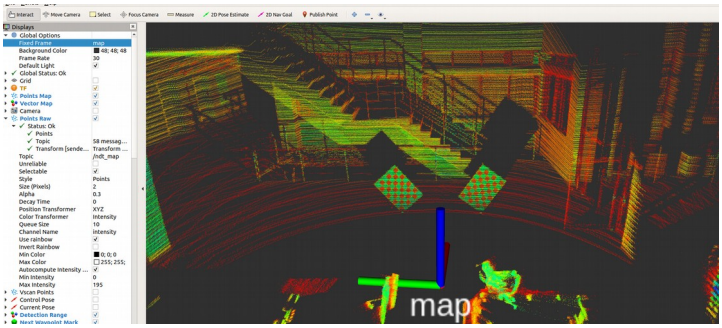




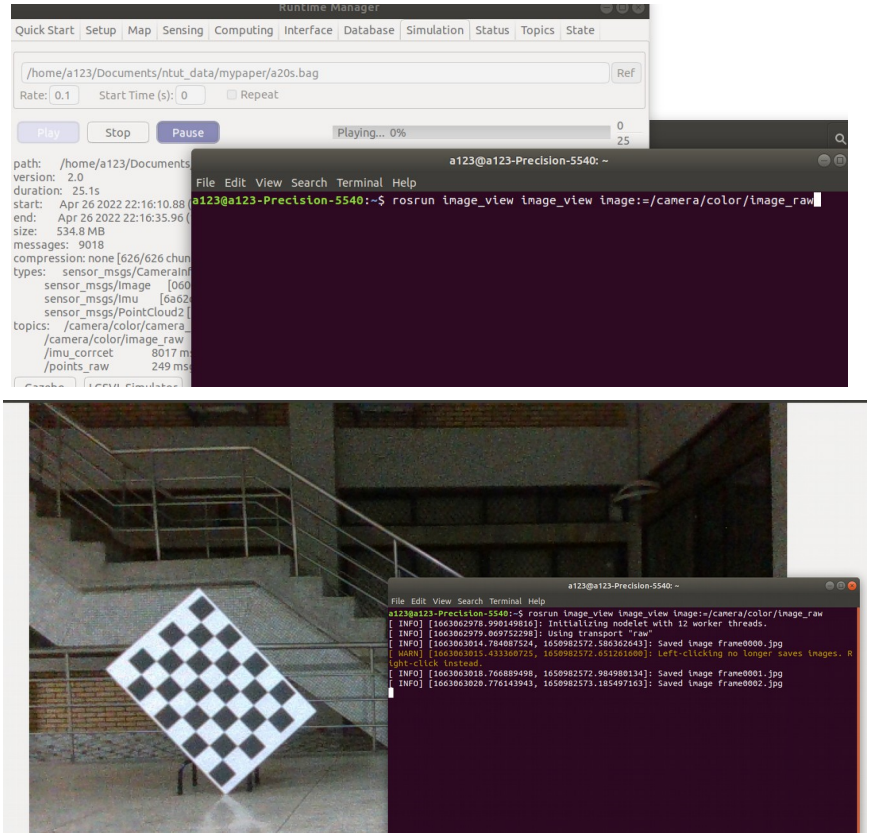
2.開啟 ndt mapping,設定參數如下(圖為論文使用的參數,未來可依情況調整)



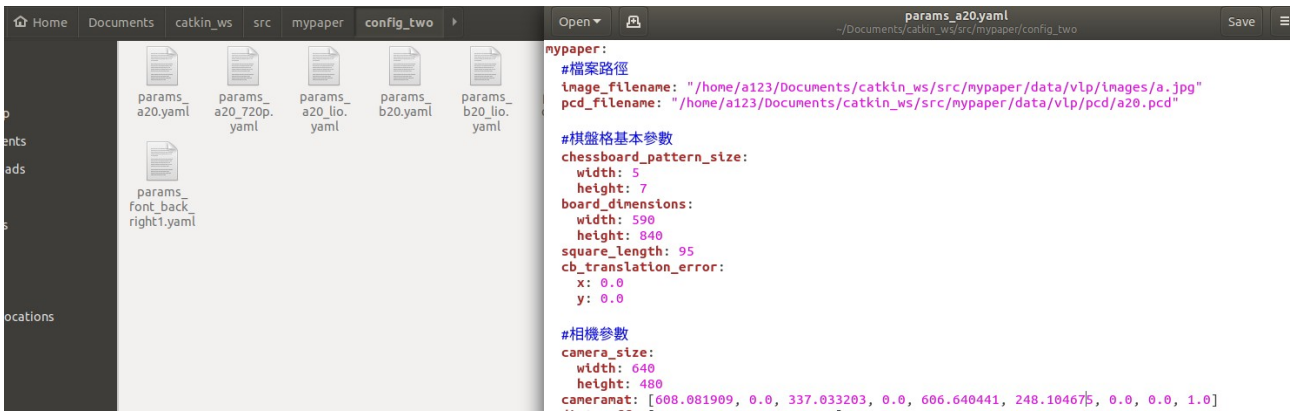
3.out pcd,輸出時要設為無降採樣(Original)



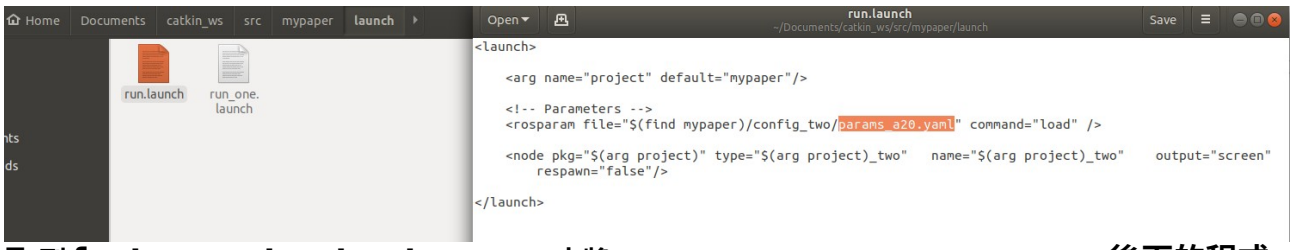
4.將影像儲存為 jpg
指令 rosrun image_view image_view image:=/camera/color/image_raw
接著在圖像上按右鍵即可儲存 jpg(bag file 要邊播放才能存檔,所以建議 bag 播放速度調 0.1 避免時間差過大)



5.到 config_two 內修改 jpg 和 pcd 的路徑(建議每次不同資料校正都創建一個新的 yaml)



6.到 run.launch 內修改為要使用的 yaml

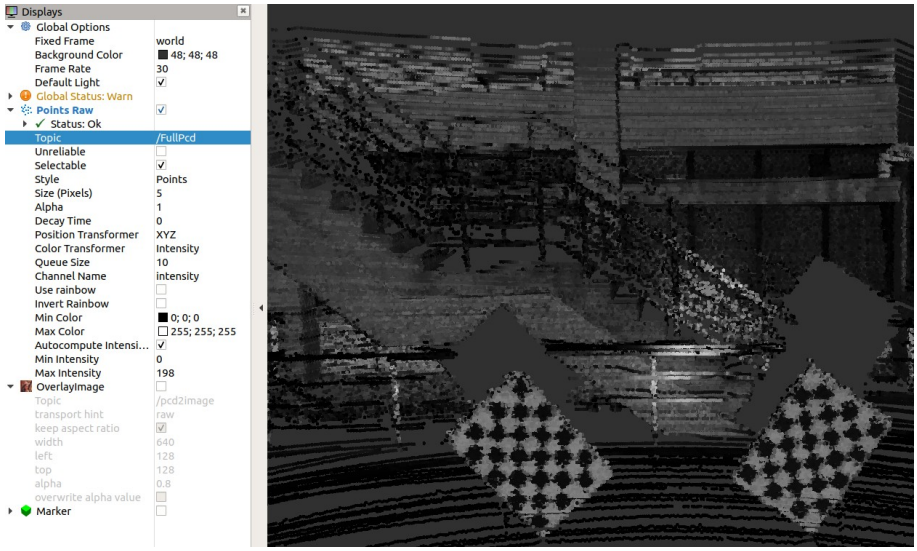


7.到 feature_extractor_two.cpp 中將 passthrough(full_pcd, passthrough_pcd);後面的程式先註解掉，並執行 roslaunch mypaper run.launch，rviz 選擇 FullPcd 即可看到完整點雲(建議 rviz 先選擇 topic 為 FullPcd 再執行程式,因為 fullpcd 只會在一開始載入 1 次而已)

feature_extractor_two.cpp

home > a123 > Documents > catkin_ws > src > mypaper > src > feature_extractor_two.cpp

```
1190  #else
1191      pcl::PointCloud<POINT_TYPE>::Ptr passthrough_pcd(new pcl::PointCloud<POINT_TYPE>);
1192      pcl::PointCloud<POINT_TYPE>::Ptr ransac_pcd(new pcl::PointCloud<POINT_TYPE>);
1193      pcl::PointCloud<POINT_TYPE>::Ptr convexhull_pcd(new pcl::PointCloud<POINT_TYPE>);
1194      std::vector<cv::Point3d> cross_point;
1195
1196      std::cout << "run" << std::endl;
1197      if(!loadpcd_done)
1198        loadpcd_done = read_pcd(full_pcd);
1199
1200      passthrough(full_pcd, passthrough_pcd);
1201      //ransac(passthrough_pcd, ransac_pcd);
1202      /*pca(passthrough_pcd, ransac_pcd);
1203      bound(ransac_pcd, convexhull_pcd);
1204      if (!savefile_done)
1205      {
1206        save_point(convexhull_pcd);
1207        savefile_done = true;
1208      }
1209      hough3dlines(line_arr);
1210      cross_point = find_crossline(line_arr);
1211      auto [board_image_pixels, in_rMat, in_tMat] = findImgChessboardCorners();
1212
1213      cv::Mat extrinsic_matrix = output_extrinsic(in_rMat, in_tMat);
1214      makeImageFromPointCloud(convexhull_pcd, cross_point, in_rMat, in_tMat, extrinsic_matrix);
1215      reprojection_error(extrinsic_matrix, cross_point, board_image_pixels);*/
1216  #endif
1217
```



8.手動將 pcd 背景切除

使用 rviz 上方的 Publish Point 功能,去選擇棋盤格的 4 個角點

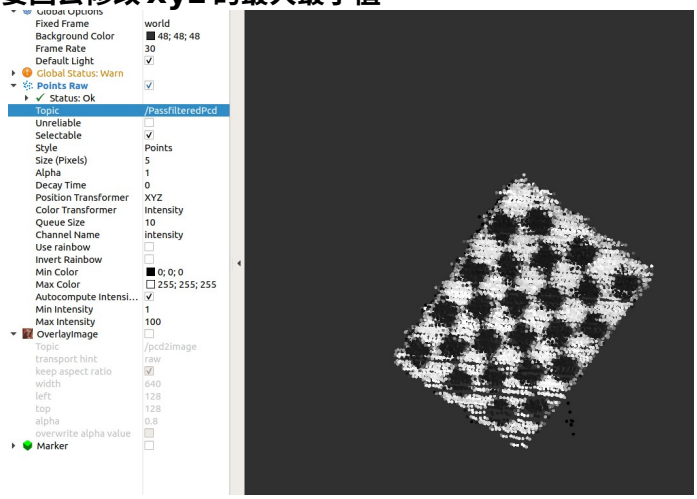
再用 topic echo /clicked_point 顯示出來

```
a123@a123-Precision-5540: ~
File  Edit  View  Search  Terminal  Help
a123@a123-Precision-5540:~$ rostopic echo /clicked_point
header:
  seq: 0
  stamp:
    secs: 1663064395
    nsecs: 267262915
  frame_id: "world"
point:
  x: 3.59194087982
  y: 1.2498139143
  z: 0.325959920883
---
header:
  seq: 1
  stamp:
    secs: 1663064401
    nsecs: 996951147
  frame_id: "world"
point:
  x: 3.52435064316
  y: 1.68377709389
  z: -0.078736782074
---
header:
  seq: 2
  stamp:
    secs: 1663064406
    nsecs: 763341832
  frame_id: "world"
point:
  x: 3.57075023651
  y: 0.683711409569
  z: -0.26623415947
---
header:
  seq: 3
  stamp:
    secs: 1663064411
    nsecs:  36717724
  frame_id: "world"
point:
  x: 3.50136184692
  y: 1.09525418282
  z: -0.649348974228
---
```

將 x,y,z 的最大最小值貼到 yaml 裡,有時可能切太多或太少,數值需要依實際情況做增減

```
Open                                   params_a20.yaml
                                       ~/Documents/catkin_ws/src/mypaper/
cameramat: [608.081909, 0.0, 337.033203, 0.0, 606.640441, 2
distcoeff: [0.0, 0.0, 0.0, 0.0, 0.0]

#棋盤格在icp的初始預測矩陣,切割閥值,算出的3D3D轉換矩陣
counterclockwise:
  x_min: 3.475
  x_max: 3.595
  y_min: 0.68
  y_max: 1.675
  z_min: -0.635
  z_max: 0.35
  icp_init_rotmat: [0.0, 0.7071067, -0.7071067, 0.0, -0.707
  icp_init_tranmat: [-1.0, 0.5, 3.8]
  icp_rotmat: [0.990, -0.043, 0.133, 0.042, 0.999, 0.012, -
  icp_tranmat: [0.003499, 0.219854, -0.264196]
  r_2d: [1,0,3,2]
  r_3d: [0,1,2,3]
clockwise:
```

重新執行程式,將 rviz 的 topic 選為 PassfilteredPcd,並觀察去背的效果,如果棋盤格有明顯被切割到或抓到地板點雲,要回去修改 xyz 的最大最小值



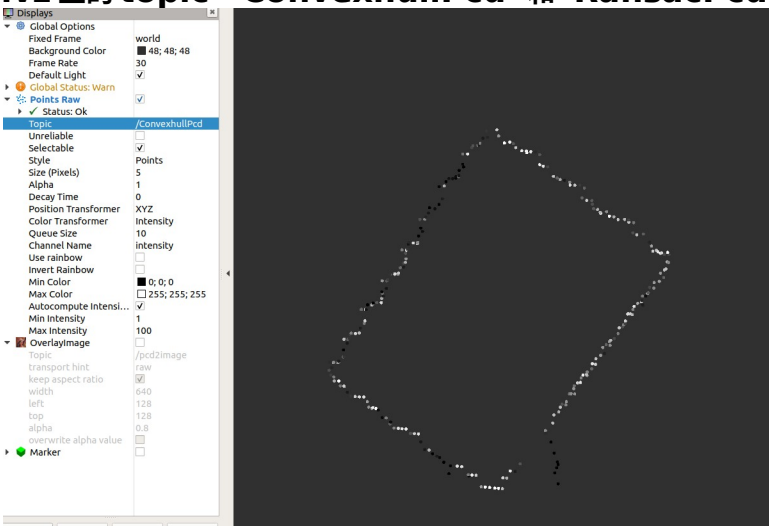
9.將剛程式碼註解的後半段復原

feature_extractor_two.cpp

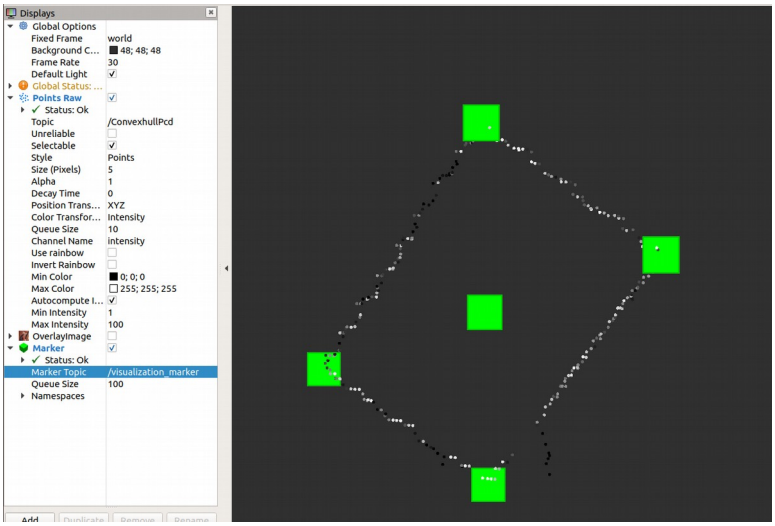
home > a123 > Documents > catkin_ws > src > mypaper > src > feature_extractor_two.cpp

```
1187    /*makeImageFromPointCloud(convexhull_pcd_all, cross_point, in_rMat, in_tMat, (extrinsic_matrix
1188    reprojection_error((extrinsic_matrix+extrinsic_matrix2)/2, cross_point, board_image_pixels);*/
1189
1190  #else
1191    pcl::PointCloud<POINT_TYPE>::Ptr passthrough_pcd(new pcl::PointCloud<POINT_TYPE>);
1192    pcl::PointCloud<POINT_TYPE>::Ptr ransac_pcd(new pcl::PointCloud<POINT_TYPE>);
1193    pcl::PointCloud<POINT_TYPE>::Ptr convexhull_pcd(new pcl::PointCloud<POINT_TYPE>);
1194    std::vector<cv::Point3d> cross_point;
1195
1196    std::cout << "run" << std::endl;
1197    if(!loadpcd_done)
1198      loadpcd_done = read_pcd(full_pcd);
1199
1200    passthrough(full_pcd, passthrough_pcd);
1201    //ransac(passthrough_pcd, ransac_pcd);
1202    pca(passthrough_pcd, ransac_pcd);
1203    bound(ransac_pcd, convexhull_pcd);
1204    if (!savefile_done)
1205    {
1206      save_point(convexhull_pcd);
1207      savefile_done = true;
1208    }
1209    hough3dlines(line_arr);
1210    cross_point = find_crossline(line_arr);
1211    auto [board_image_pixels, in_rMat, in_tMat] = findImgChessboardCorners();
1212
1213    cv::Mat extrinsic_matrix = output_extrinsic(in_rMat, in_tMat);
1214    makeImageFromPointCloud(convexhull_pcd, cross_point, in_rMat, in_tMat, extrinsic_matrix);
1215    reprojection_error(extrinsic_matrix, cross_point, board_image_pixels);
1216  #endif
1217
```

重新執行程式,觀察 rivz 上的 topic “ConvexhullPcd”和”RansacPcd”有沒有正常



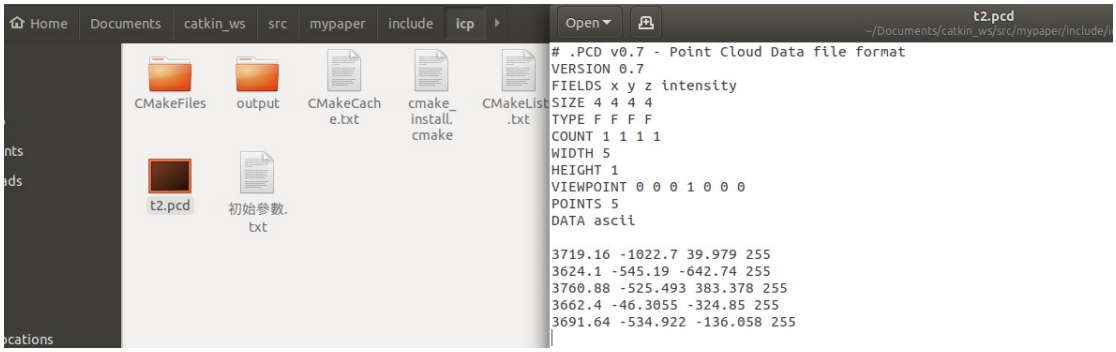
觀察 rviz 上的 visualization_marker 有沒有對應到點雲的特徵點



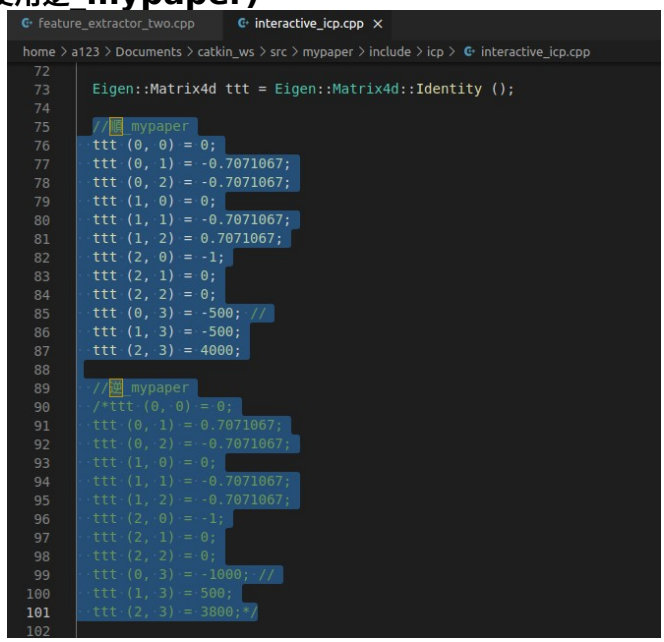
10.以上步驟都正常的話,到 terminal 找到 5 個特徵點的座標,如下圖位置

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

imagepoint_x:364.224; imagepoint_y:269.406
reprojection_error_mean:1.37882
run
^C[mypaper_two-1] killing on exit
npoints=79, a=(3.682545,-0.838782,-0.222979), b=(0.113358,-0.569451,0.814171)
npoints=101, a=(3.708359,-0.269922,0.005650), b=(0.114406,-0.556703,0.822796)
npoints=53, a=(3.739232,-0.783486,0.205194), b=(0.068885,0.820874,0.566940)
npoints=44, a=(3.644520,-0.279242,-0.473278), b=(0.064620,0.841578,0.536255)
crossline:
3719.16 -1022.7 39.979
crossline:
3624.1 -545.19 -642.74
crossline:
3760.88 -525.493 383.378
crossline:
3662.4 -46.3055 -324.85
0,1_length:83.8545
0,2_length:60.5706
0,3_length:104.387
1,2_length:103.538
1,3_length:59.2796
2,3_length:86.0759
center:
3691.64 -534.922 -136.058
2D corners:[393.1790466308594, 258.6477355957031;
 401.6368103027344, 245.6674194335938;
 410.5241394042969, 232.5394134521484;
 419.1221313476562, 219.6205902099609;
```

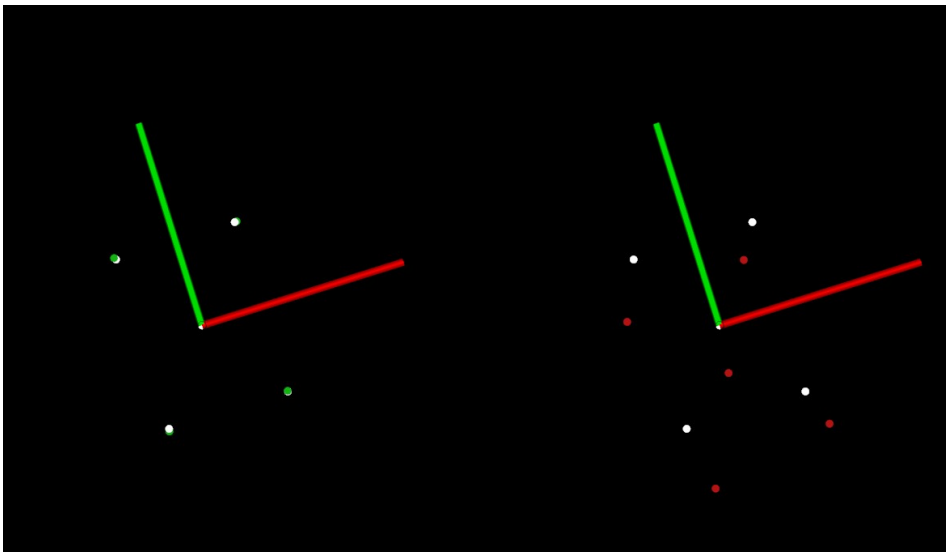
將 5 個特徵的座標貼到以下路徑的 t2.pcd 中,最後的 255 是 intensity



修改 interactive_icp.cpp 的程式,修改 icp 的初始轉換矩陣(如果棋盤格為順時針擺放使用順_mypaper,逆時針則使用逆_mypaper)



重新 make 後,執行./interactive_icp t1.pcd t2.pcd 10
觀察 icp 匹配效果,圖右邊為 icp 匹配前,圖左邊為 icp 匹配後 ,如果兩個棋盤格初始座標差太多 icp 會失敗,需至上一步修改初始轉換矩陣,讓 icp 可以成功匹配



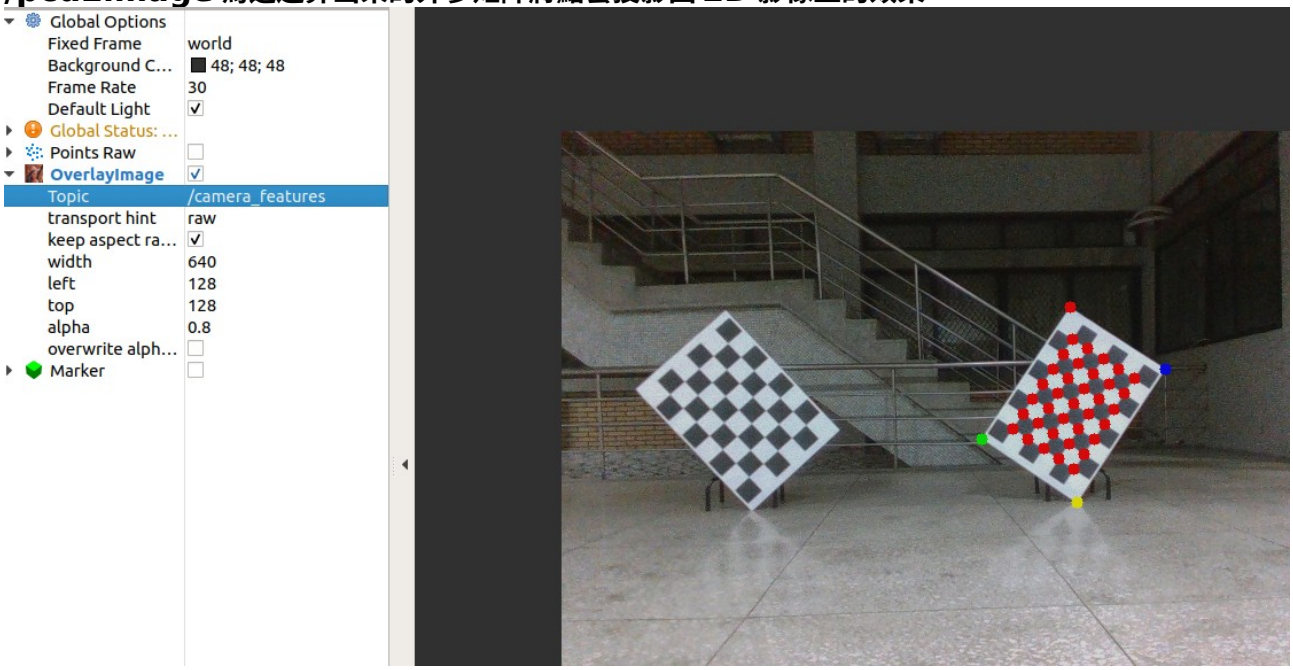
icp 計算的旋轉矩陣和平移矩陣可從 terminal 找到

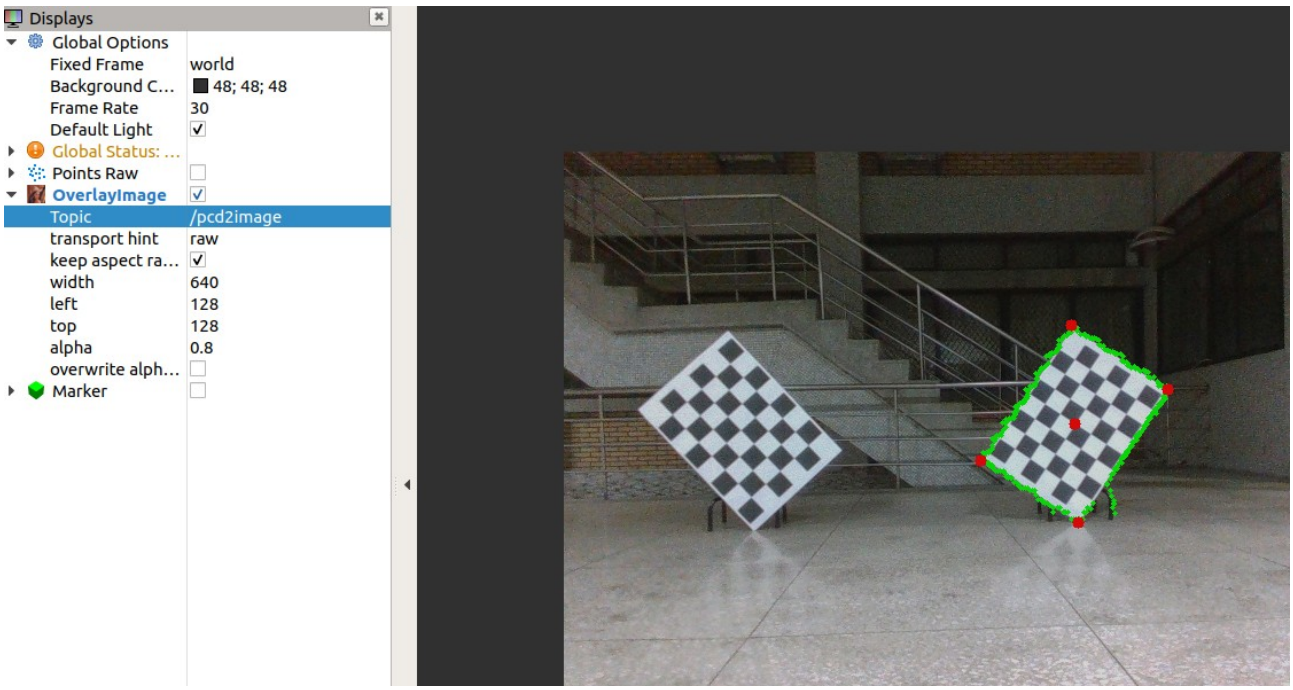
```
a123@a123-Precision-5540: ~/Documents/catkin_ws/src/mypaper/include/icp
File  Edit  View  Search  Terminal  Help
' is deprecated: This map is deprecated and is kept only to prevent breaking existing user c
ode. Starting from PCL 1.8.0 model sample size is a protected member of the SampleConsensusM
odel class [-Wdeprecated-declarations]
   SAC_SAMPLE_SIZE (sample_size_pairs, sample_size_pairs + sizeof (sample_size_pairs) / size
of (SampleSizeModel));
   ^~~~~~~~~~~~~~~
/usr/include/pcl-1.8/pcl/sample_consensus/model_types.h:99:3: note: declared here
[100%] Linking CXX executable interactive_icp
[100%] Built target interactive_icp
a123@a123-Precision-5540:~/Documents/catkin_ws/src/mypaper/include/icp$ ./interactive_icp t1
.pcd t2.pcd 10

Loaded file t1.pcd (5 points) in 0 ms

Applied 10 ICP iteration(s) in 0 ms

ICP has converged, score is 72.0773

ICP transformation 10 : cloud_icp -> cloud_in
Rotation matrix :
0.979093,0.191908,0.067441,-0.183516,0.976373,-0.114091,-0.087742,0.099329,0.991179
Translation vector :
0.046043,0.243305,-0.286235
```

11.將 icp 的初始轉換矩陣和計算後的轉換矩陣貼到 yaml 裡

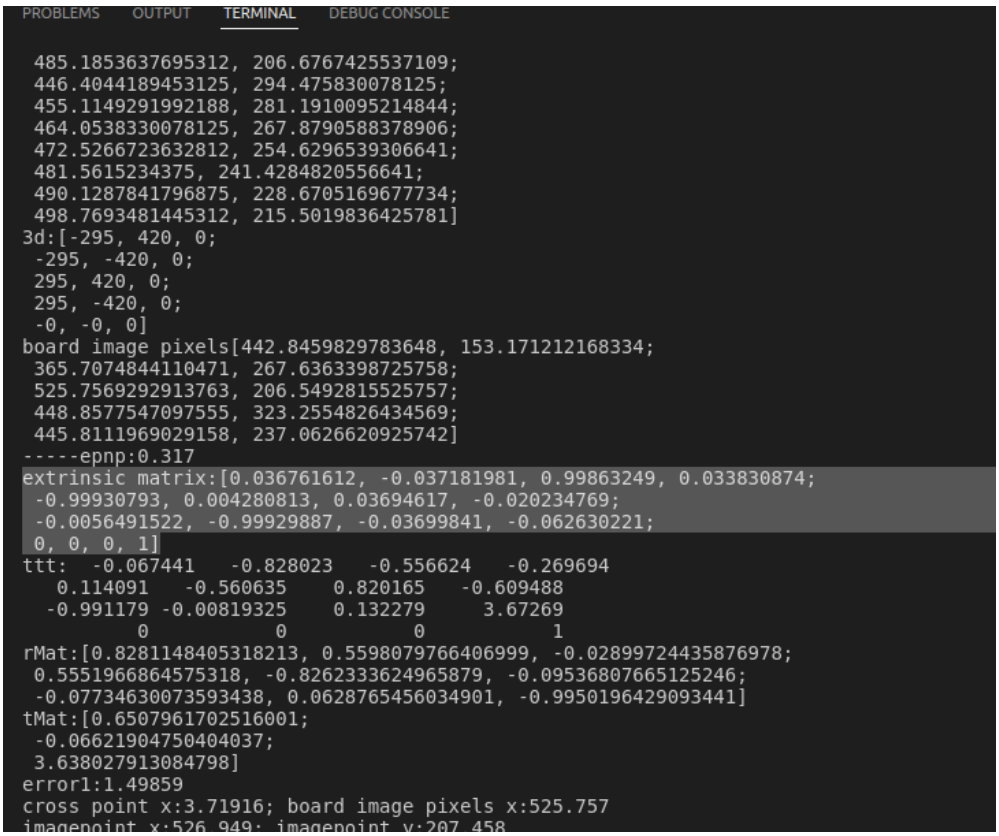
```
Open   params_a20.yaml
~/Documents/catkin_ws/src/mypaper/config_two   Save

  height: 480
cameramat: [608.081909, 0.0, 337.033203, 0.0, 606.640441, 248.104675, 0.0, 0.0, 1.0]
distcoeff: [0.0, 0.0, 0.0, 0.0, 0.0]

#棋盤格在icp的初始預測矩陣,切割閥值,算出的3D3D轉換矩陣
counterclockwise:
  x_min: 3.475
  x_max: 3.595
  y_min: 0.68
  y_max: 1.675
  z_min: -0.635
  z_max: 0.35
  icp_init_rotmat: [0.0, 0.7071067, -0.7071067, 0.0, -0.7071067, -0.7071067, -1.0, 0.0, 0.0]
  icp_init_tranmat: [-1.0, 0.5, 3.8]
  icp_rotmat: [0.990, -0.043, 0.133, 0.042, 0.999, 0.012, -0.134, -0.007, 0.991]
  icp_tranmat: [0.003499, 0.219854, -0.264196]
  r_2d: [1,0,3,2]
  r_3d: [0,1,2,3]
clockwise:
  x_min: 3.6
  x_max: 3.8
  y_min: -1.15
  y_max: 0.0
  z_min: -0.63
  z_max: 0.38
  icp_init_rotmat: [0.0, -0.7071067, -0.7071067, 0.0, -0.7071067, 0.7071067, -1.0, 0.0, 0.0]
  icp_init_tranmat: [-0.5, -0.5, 4.0]
  icp_rotmat: [0.979093,0.191908,0.067441,-0.183516,0.976373,-0.114091,-0.087742,0.099329,0.991179]
  icp_tranmat: [0.046043,0.243305,-0.286235]
  r_2d: [2,3,0,1]
```

12.重新執行程式,rviz 中的/camera_features 可看到 2D 影像找到的特徵點
/pcd2image 為透過算出來的外參矩陣將點雲投影回 2D 影像上的效果





13.可從 terminal 中找到外參矩陣



重投影誤差：

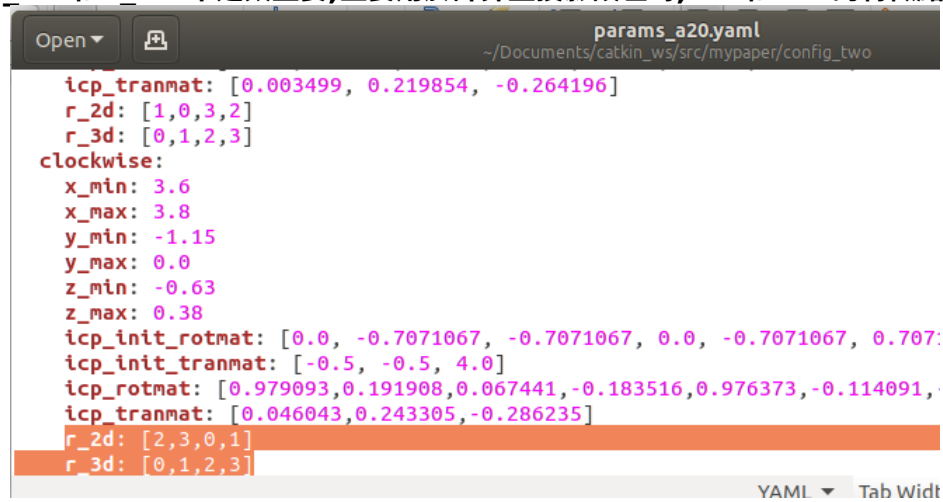
```
-----epnp:0.597
extrinsic matrix:[0.036761612, -0.037181981, 0.99863249, 0.033830874;
 -0.99930793, 0.004280813, 0.03694617, -0.020234769;
 -0.0056491522, -0.99929887, -0.03699841, -0.062630221;
 0, 0, 0, 1]
ttt:  -0.067441   -0.828023   -0.556624   -0.269694
   0.114091   -0.560635    0.820165   -0.609488
  -0.991179 -0.00819325    0.132279     3.67269
          0           0           0           1
rMat:[0.8281148405318213, 0.5598079766406999, -0.02899724435876978;
 0.5551966864575318, -0.8262333624965879, -0.09536807665125246;
 -0.07734630073593438, 0.0628765456034901, -0.9950196429093441]
tMat:[0.6507961702516001;
 -0.06621904750404037;
 3.638027913084798]
error1:1.49859
cross point x:3.71916; board image pixels x:525.757
imagepoint x:526.949; imagepoint y:207.458
error2:0.0792821
cross point x:3.6241; board image pixels x:448.858
imagepoint x:448.881; imagepoint y:323.18
error3:1.6011
cross point x:3.76088; board image pixels x:442.846
imagepoint x:442.493; imagepoint y:151.609
error4:2.41428
cross point x:3.6624; board image pixels x:365.707
imagepoint x:363.973; imagepoint y:269.316
reprojection error mean:1.39831
run
npoints=79, a=(3.682545,-0.838782,-0.222979), b=(0.113358,-0.569451,0.814171)
npoints=101, a=(3.708359,-0.269922,0.005650), b=(0.114406,-0.556703,0.822796)
npoints=53, a=(3.739232,-0.783486,0.205194), b=(0.068885,0.820874,0.566940)
npoints=44, a=(3.644520,-0.279242,-0.473278), b=(0.064620,0.841578,0.536255)
```

棋盤格邊長(兩個特別大的值是對角線長度)：

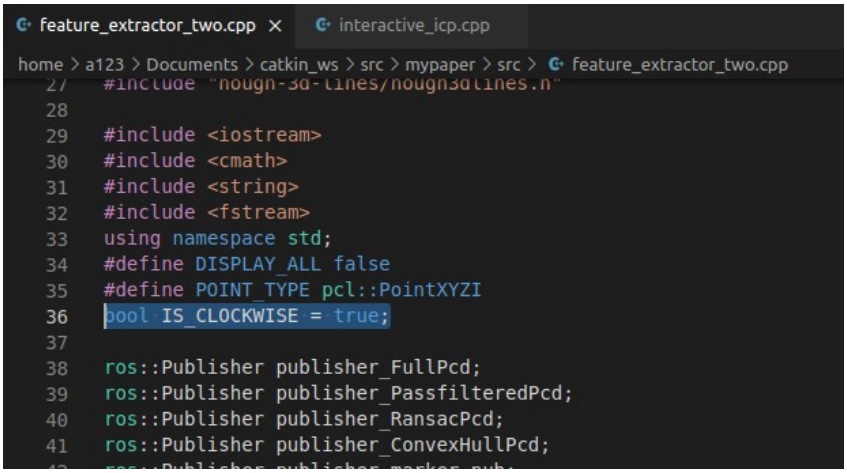
```
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

npoints=101, a=(3.708359,-0.269922,0.005650), b=(0.114406,
npoints=53, a=(3.739232,-0.783486,0.205194), b=(0.068885,0
npoints=44, a=(3.644520,-0.279242,-0.473278), b=(0.064620,
crossline:
3719.16 -1022.7 39.979
crossline:
3624.1 -545.19 -642.74
crossline:
3760.88 -525.493 383.378
crossline:
3662.4 -46.3055 -324.85
0,1 length:83.8545
0,2 length:60.5706
0,3 length:104.387
1,2 length:103.538
1,3 length:59.2796
2,3 length:86.0759
center:
3691.64 -534.922 -136.058
2D corners:[393.1790466308594, 258.6477355957031;
 401.6368103027344, 245.6674194335938;
 410.5241394042969, 232.5394134521484;
 419.1221313476562, 219.6205902099609;
```

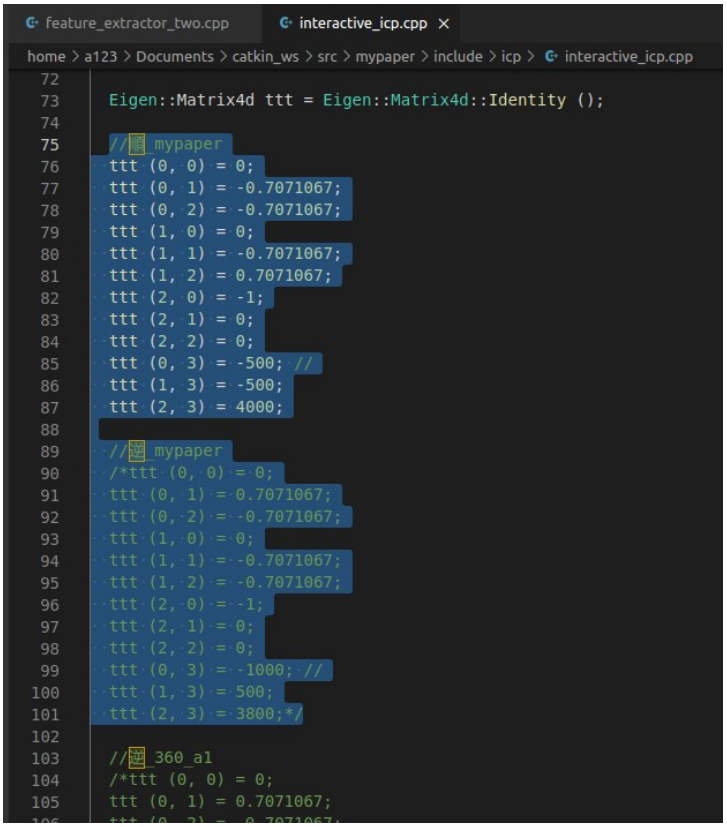
14.其他補充說明

yaml 內的 r_2d 和 r_3d 不是太重要,主要用於計算重投影誤差時,2D 和 3D 的特徵點對應用

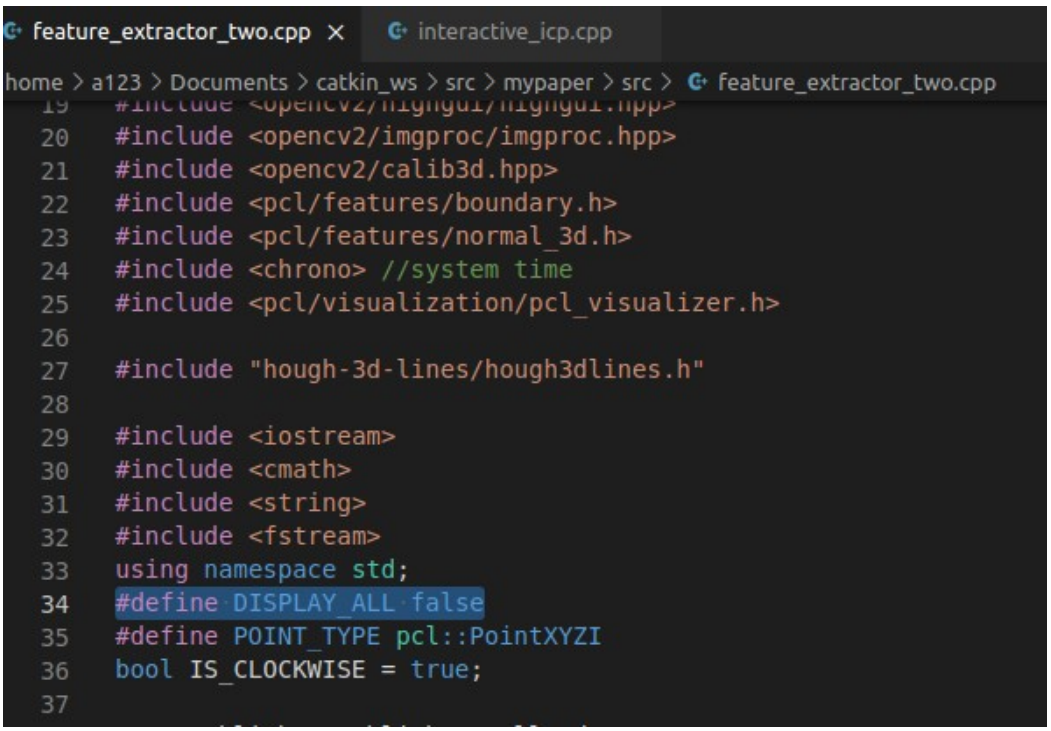


因為論文中用兩個棋盤格,所以 yaml 內分成順時針 clockwise 和逆時針 counterclockwise 擺放棋盤格,程式內也要依當前是順時針還是逆時針做修改,所以 1~14 的步驟要做兩次

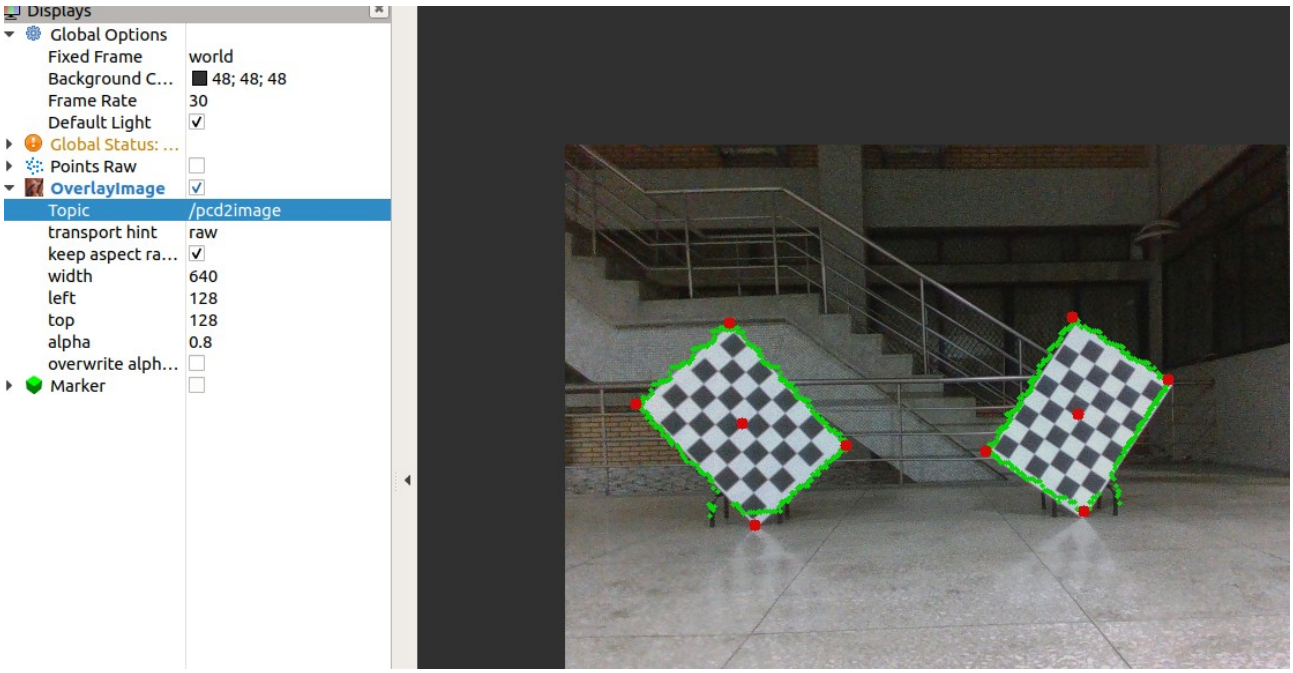




正常情況左右棋盤格校正時,DISPLAY_ALL 設為 false



DISPLAY_ALL 設為 true 時,可以同時顯示兩個棋盤格的校正結果



此部份用來修改 terminal 顯示的重投影誤差(基於左棋盤格 or 基於右棋盤格 or 基於兩個棋盤格)

feature_extractor_two.cpp × interactive_icp.cpp

home > a123 > Documents > catkin_ws > src > mypaper > src > feature_extractor_two.cpp

```
1176        cross_point.push_back(cross_point2[i]);
1177      }
1178      for(int i=0;i<board_image_pixels2.size();i++)
1179      {
1180        board_image_pixels.push_back(board_image_pixels2[i]);
1181      }
1182      makeImageFromPointCloud(convexhull_pcd_all, cross_point, in_rMat2, in_tMat2, extrinsic_matrix2);
1183      reprojection_error(extrinsic_matrix2, cross_point, board_image_pixels);
1184      IS_CLOCKWISE = true;
1185      /*makeImageFromPointCloud(convexhull_pcd_all, cross_point, in_rMat, in_tMat, extrinsic_matrix);
1186      reprojection_error(extrinsic_matrix, cross_point, board_image_pixels);*/
1187      /*makeImageFromPointCloud(convexhull_pcd_all, cross_point, in_rMat, in_tMat, (extrinsic_matrix+extrinsic_matrix2)/2);
1188      reprojection_error((extrinsic_matrix+extrinsic_matrix2)/2, cross_point, board_image_pixels);*/
1189
1190  #else
1191      pcl::PointCloud<POINT_TYPE>::Ptr passthrough_pcd(new pcl::PointCloud<POINT_TYPE>);
1192      pcl::PointCloud<POINT_TYPE>::Ptr ransac_pcd(new pcl::PointCloud<POINT_TYPE>);
1193      pcl::PointCloud<POINT_TYPE>::Ptr convexhull_pcd(new pcl::PointCloud<POINT_TYPE>);
1194      std::vector<cv::Point3d> cross_point;
```

----------單棋盤格校正（使用 a1.pcd 和 a1.jpg 當範例）----------

1.流程基本上跟雙棋盤格校正一樣,bag file 先用 ndt mapping 存出 pcd 和擷取影像 jpg,只是沒有寫 yaml 檔,參數要直接在 feature_extractor.cpp 內改

2.跑 launch 時要改成 roslaunch mypaper run_one.launch

3.設定 jpg 和 pcd 的路徑,去除背景時的 xyz 最大最小值

feature_extractor.cpp

home > a123 > Documents > catkin_ws > src > mypaper > src > feature_extractor.cpp

```
#include <iostream>
#include <cmath>
#include <string>
#include <fstream>
using namespace std;

//360 a1
#define image_filename "/home/a123/Documents/catkin_ws/src/mypaper/data/fusion360/a1.jpg"
#define pcd_filename "/home/a123/Documents/catkin_ws/src/mypaper/data/fusion360/a1.pcd"
#define X_MIN 4.151742
#define X_MAX 4.2686
#define Y_MIN -0.7875
#define Y_MAX 0.1531
#define Z_MIN -0.5896
#define Z_MAX 0.401703

ros::Publisher publisher_FullPcd;
```

4.相機內參和棋盤格基本資訊

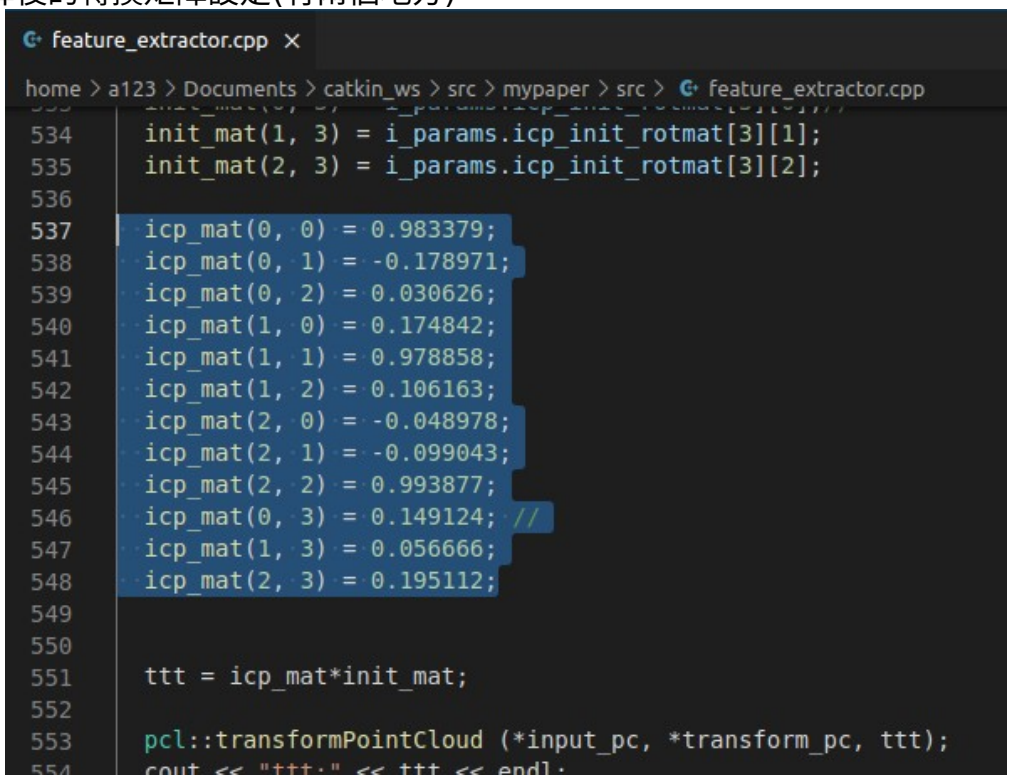
feature_extractor.cpp

home > a123 > Documents > catkin_ws > src > mypaper > src > feature_extractor.cpp

```
void get_params()
{
  i_params.chessboard_pattern_size.width = 5;
  i_params.chessboard_pattern_size.height = 7;

  i_params.board_dimensions.width = 590;
  i_params.board_dimensions.height = 840;
  i_params.square_length = 95;
  i_params.cb_translation_error.x = 0;
  i_params.cb_translation_error.y = 0;
  i_params.cameramat = cv::Mat::zeros(3, 3, CV_64F);
  i_params.cameramat.at<double>(0, 0) = 1164.6233338297;
  i_params.cameramat.at<double>(0, 2) = 950.1242940800;
  i_params.cameramat.at<double>(1, 1) = 1161.1018211652;
  i_params.cameramat.at<double>(1, 2) = 538.5516554830;
  i_params.cameramat.at<double>(2, 2) = 1;
  i_params.distcoeff = cv::Mat::eye(1, 4, CV_64F);
  i_params.distcoeff.at<double>(0) = 0.107542;
  i_params.distcoeff.at<double>(1) = -0.160397;
  i_params.distcoeff.at<double>(2) = 0.000789;
  i_params.distcoeff.at<double>(3) = -0.004844;

  //順時針45度
  /*i_params.icp_init_rotmat[0][0] = 0;
```

5.icp 的初始轉換矩陣設定

6.ICP 計算後的轉換矩陣設定(有兩個地方)



feature_extractor.cpp

home > a123 > Documents > catkin_ws > src > mypaper > src > feature_extractor.cpp

```
621   init_mat.at<float>(1,3) = i_params.icp_init_rotmat[3][1];
622   init_mat.at<float>(2,3) = i_params.icp_init_rotmat[3][2];
623
624   icp_mat.at<float>(0,0) = 0.983379;
625   icp_mat.at<float>(0,1) = -0.178971;
626   icp_mat.at<float>(0,2) = 0.030626;
627   icp_mat.at<float>(1,0) = 0.174842;
628   icp_mat.at<float>(1,1) = 0.978858;
629   icp_mat.at<float>(1,2) = 0.106163;
630   icp_mat.at<float>(2,0) = -0.048978;
631   icp_mat.at<float>(2,1) = -0.099043;
632   icp_mat.at<float>(2,2) = 0.993877;
633   icp_mat.at<float>(0,3) = 0.149124; //
634   icp_mat.at<float>(1,3) = 0.056666;
635   icp_mat.at<float>(2,3) = 0.195112;
636
637   extrinsic_matrix = (r_Mat*icp_mat*init_mat).t();
638   extrinsic_matrix.at<float>(0,3) = -(extrinsic_matrix.at<float>(3,2)
639   extrinsic_matrix.at<float>(1,3) = extrinsic_matrix.at<float>(3,0) +
640   extrinsic_matrix.at<float>(2,3) = extrinsic_matrix.at<float>(3,1) +
641   extrinsic_matrix.at<float>(3,0) = 0;
```

7.一樣可從 rviz 看 3D 特徵點效果和轉換結果
terminal 可找到外參矩陣

----------論文校正用到的 bag pcd jpg 已上傳 NAS----------