```
1 // EEE 598 : DIVP Project 1: Stereo Correspondence using FAST+FREAK, AGAST
     +LUCID and Sift
 2 // Reference :https://docs.opencv.org
 3 // Authors : Chandana Srinivasa and Kiran Kumar (Group 1)
 5 #include<stdio.h>
 6 #include<iostream>
 7 #include <vector>
 8 #include"opencv2/core.hpp"
9 #include"opencv2/features2d.hpp"
10 #include"opencv2/xfeatures2d.hpp"
11 #include"opencv2/highgui.hpp"
12 #include"opencv2/imgproc.hpp"
#include "opencv2/calib3d.hpp"
14
15 using namespace cv;
16 using namespace std;
17
18 using namespace cv::xfeatures2d;
19 /** @function main */
20
21
22 // Function to find out the error value
23 float errFinder(Mat img_1_Grey, Mat GT_img, vector<KeyPoint> keypoints_1_fast, →
      vector<KeyPoint> keypoints_2_fast, vector< DMatch > matches_Fast_Freak)
24 {
25
       // img1: right img2: left (Reference)
26
27
       std::vector<int> pointIndexesLeft;
28
       std::vector<int> pointIndexesRight;
29
       for (std::vector⟨cv::DMatch⟩::const iterator it = matches Fast Freak.begin →
          (); it != matches_Fast_Freak.end(); ++it)
30
31
           // Get the indexes of the selected matched keypoints
32
33
           pointIndexesRight.push_back(it->queryIdx);
           pointIndexesLeft.push_back(it->trainIdx);
34
35
       }
36
       // Convert keypoints into Point2f
37
38
       std::vector<cv::Point2f> selPointsLeft, selPointsRight;
       cv::KeyPoint::convert(keypoints 1 fast, selPointsRight,
39
          pointIndexesRight);
40
       cv::KeyPoint::convert(keypoints_2_fast, selPointsLeft, pointIndexesLeft);
41
       int xright = 0, xleft = 0, yright = 0, yleft = 0;
42
43
       Mat diff(img_1_Grey.rows, img_1_Grey.cols, CV_8UC1);
       diff = 255; int k = 0; int new_xright = 0; float MSE = 0.0; float add_all >
44
          = 0.0; float count = 0; float err = 0;
45
       cout<<"Match count"<<selPointsRight.size() << endl;</pre>
       for (int i = 0; i < (int)selPointsRight.size(); i++)</pre>
46
47
           xright = (int)selPointsRight.at(i).x;
48
49
           yright = (int)selPointsRight.at(i).y;
50
           xleft = (int)selPointsLeft.at(i).x;
51
```

```
...2015\Projects\DIVP_Project_1\DIVP_Project_1\Source1.cpp
52
            yleft = (int)selPointsLeft.at(i).y;
53
54
55
56
            k = GT img.at<uchar>(xleft, yleft);// Disparity from Ground Truth
57
            new_xright = xleft + k;
58
59
            count += 1;
60
            //root mean square error calculation
61
            MSE = pow((new_xright - xright), 2) + pow((yleft - yright), 2);
62
            add all = add all + MSE;
63
            //diff.at<uchar>(yright,xright) = abs(xright - xleft);
64
65
        err = sqrt(add_all) / count;
66
67
        waitKey();
68
        return err;
69 }
70
71 int
         main(int argc, char** argv)
72 {
73
        try
74
        {
75
            // ... Contents of your main
76
77
        catch (cv::Exception & e)
78
        {
79
            cerr << e.what() << endl; // output exception message</pre>
80
        cout << " Output For Without Cross Check and Without Threshold " << endl;</pre>
81
82
83
               img 1 = imread("D:\\ASU Sem 1\\DIVP Project\\new data set\
84
        Mat
                                                                                     P
           \Flowerpots\\view1.png",CV BGR2GRAY);// reading input image 1
               img 2 = imread("D:\\ASU Sem 1\\DIVP Project\\new data set\
85
           \Flowerpots\\view5.png",CV_BGR2GRAY);// reading input image 2
        Mat GT_img = imread("D:\\ASU_Sem_1\\DIVP_Project\\new_data_set\\Flowerpots →
86
           \\disp5.png");//reading ground truth disparity values
87
        //cout << "GT_img"<< GT_img << endl;</pre>
88
89
        Mat img 1 D, img 2 D, img 1 Grey, img 2 Grey;
90
        pyrDown(img 1,img 1 D);
91
        pyrDown(img_2,img_2_D);
92
93
        cvtColor(img 1 D, img 1 Grey, CV BGR2GRAY);
        cvtColor(img_2_D, img_2_Grey, CV_BGR2GRAY);
94
95
96
        // -- Detect the keypoints using a FAST Detector
97
        Ptr <FastFeatureDetector> detector_fast = FastFeatureDetector::create();
98
        std::vector<KeyPoint> keypoints_1_fast, keypoints_2_fast;
        detector_fast->detect(img_1_Grey, keypoints_1_fast);
99
100
        detector_fast->detect(img_2_Grey, keypoints_2_fast);
101
```

cout << " [INFO] key-point 1 size: Fast " << keypoints_1_fast.size() <<</pre>

102

103

endl;

```
...2015\Projects\DIVP_Project_1\DIVP_Project_1\Source1.cpp
104
         cout << " [INFO] key-point 2 size: Fast" << keypoints 2 fast.size() <<</pre>
           end1;
105
106
         // -- Draw keypoints
107
               img keypoints 1 fast;
108
               img keypoints 2 fast;
         drawKeypoints(img_1_Grey, keypoints_1_fast, img_keypoints_1_fast,
109
           Scalar::all(-1), DrawMatchesFlags::DEFAULT);
110
         drawKeypoints(img_2_Grey, keypoints_2_fast, img_keypoints_2_fast,
           Scalar::all(-1), DrawMatchesFlags::DEFAULT);
111
112
         // -- Show detected (drawn) keypoints
         //imshow("Keypoints 1", img keypoints 1 fast);
113
114
         imwrite("D:\\ASU_Sem_1\\DIVP_Project\\Output_images\
                                                                                     P
           \img_keypoints_1_fast.png", img_keypoints_1_fast);
115
         //imshow("Keypoints 2", img_keypoints_2_fast);
116
         imwrite("D:\\ASU_Sem_1\\DIVP_Project\\Output_images\
117
                                                                                     P
           \img_keypoints_2_fast.png", img_keypoints_2_fast);
118
119
        Mat desc1 fast, desc2 fast;
120
121
         //FREAK Descriptor
122
         Ptr<FREAK> freak = FREAK::create(true,true,22.0f,4);
         freak->compute(img_1_Grey, keypoints_1_fast, desc1_fast);
123
         freak->compute(img_2_Grey, keypoints_2_fast, desc2_fast);
124
125
126
127
         // finding the matched points using BFMatcher
         BFMatcher matcher(NORM_L2);
128
129
         //BFMatcher matcher(NORM L2,true); //For crosscheck
130
         std::vector< DMatch > matches Fast Freak;
131
         std::vector< DMatch > matches fast freak;
132
         matcher.match(desc1 fast, desc2 fast, matches fast freak);
133
        Mat img matches fast;
         // ----- For Thresholding -----
134
135
136
         //DMatch f; float mini fast = 200;
137
         //for (int k = 0; k < matches_fast_freak.size(); k++)</pre>
138
         //{
             //f = matches fast freak[k];
139
             //if (f.distance < 3 * mini_fast)</pre>
140
141
                 //matches Fast Freak.push back(matches fast freak[k]);
142
         //}
         matches Fast Freak = matches fast freak; // Remove this line for
143
           thresholding
144
145
         //draw the best matches
146
         drawMatches(img_1_D, keypoints_1_fast, img_2_D, keypoints_2_fast,
           matches_Fast_Freak, img_matches_fast);
147
         //-- Show detected matches
148
149
         //imshow("FAST+FREAK", img_matches_fast);
150
         imwrite("D:\\ASU_Sem_1\\DIVP_Project\\Output_images\
                                                                                     P
           \img_matches_fast_wo_cross_wo_thresh.png", img_matches_fast);
151
```

```
...2015\Projects\DIVP_Project_1\DIVP_Project_1\Source1.cpp
152
153
         // Sift keypoints detection
154
         Ptr <SIFT> detector = SIFT::create();
         std::vector<KeyPoint> keypoints_1_Sift, keypoints_2_Sift;
155
156
157
158
         detector->detect(img_1_Grey, keypoints_1_Sift);
159
         detector->detect(img_2_Grey, keypoints_2_Sift);
160
         // -- Draw keypoints
161
         Mat
              img_keypoints_1_Sift;
162
        Mat
               img keypoints 2 Sift;
163
         cout << " [INFO] key-point 1 size: Sift " << keypoints_1_Sift.size() <<</pre>
164
           endl:
165
         //KeyPointsFilter::retainBest(keypoints 1 fast, 500);
166
         cout << " [INFO] key-point 2 size: Sift" << keypoints 2 Sift.size() <<</pre>
167
           endl;
168
169
170
         Mat desc1 Sift, desc2 Sift;
         drawKeypoints(img 1 D, keypoints 1 Sift, img keypoints 1 Sift, Scalar::all →
171
           (-1), DrawMatchesFlags::DEFAULT);
         drawKeypoints(img_2_D, keypoints_2_Sift, img_keypoints_2_Sift, Scalar::all →
172
           (-1), DrawMatchesFlags::DEFAULT);
         // -- Show detected (drawn) keypoints
173
174
         //imshow("Keypoints 1 SIFT", img keypoints 1 Sift);
175
         imwrite("D:\\ASU Sem 1\\DIVP Project\\Output images\
                                                                                      P
           \img_keypoints_1_Sift.png", img_keypoints_1_Sift);
176
177
         //imshow("Keypoints 2 SIFT", img keypoints 2 Sift);
178
         imwrite("D:\\ASU Sem 1\\DIVP Project\\Output images\
                                                                                     P
```

```
\img_keypoints_2_Sift.png", img_keypoints_2_Sift);
179
         //Sift descriptor
180
         detector->compute(img_1_D, keypoints_1_Sift,desc1_Sift);
181
         detector->compute(img_2_D, keypoints_2_Sift,desc2_Sift);
182
183
184
185
         // Matches
186
         std::vector< DMatch > matches sift;
         std::vector< DMatch > matches Sift;
187
         //DMatch A; float mini = 30;
188
189
         matcher.match(desc1 Sift, desc2 Sift, matches sift);
190
         // ----- For Thresholding -----
191
192
         //for (int k = 0; k < matches_sift.size(); k++)</pre>
193
         //{
194
             //A = matches sift[k];
195
             //if (A.distance < 3 * mini)</pre>
196
             //matches_Sift.push_back(matches_sift[k]);
197
198
         matches Sift = matches sift; // Remove this line for thresholding
199
200
         Mat img matches Sift;
201
         drawMatches(img 1 D, keypoints 1 Sift, img 2 D, keypoints 2 Sift,
```

```
...2015\Projects\DIVP_Project_1\DIVP_Project_1\Source1.cpp
           matches Sift, img matches Sift);
202
203
         //-- Show detected matches
204
         //imshow("SIFT", img_matches_Sift);
205
         imwrite("D:\\ASU Sem 1\\DIVP Project\\Output images\
                                                                                     P
           \img_matches_Sift_wo_cross_wo_thresh.png", img_matches_Sift);
206
207
         //AGAST Keypoints
208
         Ptr <AgastFeatureDetector> detector agast = AgastFeatureDetector::create →
           ();
209
         std::vector<KeyPoint> keypoints_1_Agast, keypoints_2_Agast;
210
         detector agast->detect(img 1 Grey, keypoints 1 Agast);
211
         detector_agast->detect(img_2_Grey, keypoints_2_Agast);
212
         cout << " [INFO] key-point 1 size: Agast " << keypoints_1_Agast.size() << >
213
           endl;
214
         //KeyPointsFilter::retainBest(keypoints 1 fast, 500);
215
216
         cout << " [INFO] key-point 2 size: Agast" << keypoints_2_Agast.size() << →
           endl;
217
218
219
         // -- Draw keypoints
220
         Mat
               img keypoints 1 Agast;
221
               img_keypoints_2_Agast;
         Mat
         Mat desc1_Lucid, desc2_Lucid;
222
223
         drawKeypoints(img 1 D, keypoints 1 Agast, img keypoints 1 Agast,
           Scalar::all(-1), DrawMatchesFlags::DEFAULT);
224
         drawKeypoints(img_2_D, keypoints_2_Agast, img_keypoints_2_Agast,
           Scalar::all(-1), DrawMatchesFlags::DEFAULT);
225
226
         // -- Show detected (drawn) keypoints
227
         //imshow("Keypoints 1 AGAST", img_keypoints_1_Agast);
228
         imwrite("D:\\ASU Sem 1\\DIVP Project\\Output images\
                                                                                     P
           \img_keypoints_1_Agast.png", img_keypoints_1_Agast);
229
         //imshow("Keypoints 2 AGAST", img_keypoints_2_Agast);
         imwrite("D:\\ASU_Sem_1\\DIVP_Project\\Output_images\
230
                                                                                     P
           \img_keypoints_2_Agast.png", img_keypoints_2_Agast);
231
232
         //Detectors using LUCID
233
         Ptr<LUCID> lucid = LUCID::create();
234
         lucid->compute(img 1 D, keypoints 1 Agast, desc1 Lucid);
235
         lucid->compute(img_2_D, keypoints_2_Agast, desc2_Lucid);
236
237
         // Matches
238
         std::vector< DMatch > matches Agast Lucid;
239
         std::vector< DMatch > matches_agast_lucid;
         DMatch a; float mini_lucid = 10;
240
241
         matcher.match(desc1_Lucid, desc2_Lucid, matches_agast_lucid);
242
         // ----- For Thresholding -----
243
         //for (int k = 0; k < matches sift.size(); k++)</pre>
244
245
         //{
246
             //a = matches_agast_lucid[k];
247
             //if (a.distance < 3 * mini_lucid)</pre>
248
             //matches Agast Lucid.push back(matches agast lucid[k]);
```

```
...2015\Projects\DIVP_Project_1\DIVP_Project_1\Source1.cpp
249
         //}
250
         matches Agast Lucid = matches agast lucid; // Remove this line for
                                                                                      P
           thresholding
251
252
253
         Mat img_matches_Agast_Lucid;
254
         drawMatches(img_1_Grey, keypoints_1_Agast, img_2_Grey, keypoints_2_Agast, >>
           matches_Agast_Lucid, img_matches_Agast_Lucid);
255
         //-- Show detected matches
256
         //imshow("AGAST+LUCID", img_matches_Agast_Lucid);
         imwrite("D:\\ASU Sem 1\\DIVP Project\\Output images\
257
                                                                                      P
           \img matches Agast Lucid wo cross wo thresh.png",
                                                                                      P
           img matches Agast Lucid);
258
259
260
         // Finding the error : Calling the errFinder function
261
262
         float err_fast_freak, err_Sift, err_Agast_Lucid;
263
         err_fast_freak = errFinder(img_1_Grey, GT_img, keypoints_1_fast,
                                                                                      ₽
           keypoints_2_fast, matches_Fast_Freak);
         cout << "err value fast freak" << err fast freak << endl;</pre>
264
         err_Sift = errFinder(img_1_Grey, GT_img, keypoints_1_Sift,
265
           keypoints 2 Sift, matches Sift);
266
         cout << "err value Sift" << err_Sift << endl;</pre>
267
         err_Agast_Lucid = errFinder(img_1_Grey, GT_img, keypoints_1_Agast,
                                                                                      P
           keypoints_2_Agast, matches_Agast_Lucid);
268
         cout << "err value Agast Lucid" << err_Agast_Lucid << endl;</pre>
269
270
         waitKey(0);
271
         waitKey();
272
         return 0;
273 }
274
275
276
277
         //Mat Disparity(994, 1440, CV_8U, diff);
         //imshow("Disparity", diff);
278
279
280
```