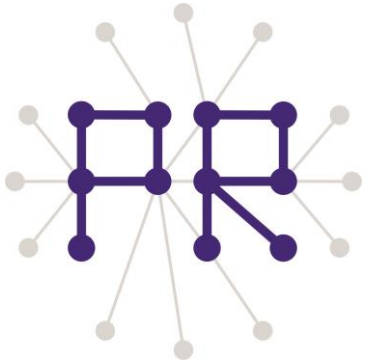
Multimedia Retrieval Exercise Course

8 Extracting and Matching Local Features

Kimiaki Shirahama, D.E.

Research Group for Pattern Recognition
Institute for Vision and Graphics
University of Siegen, Germany

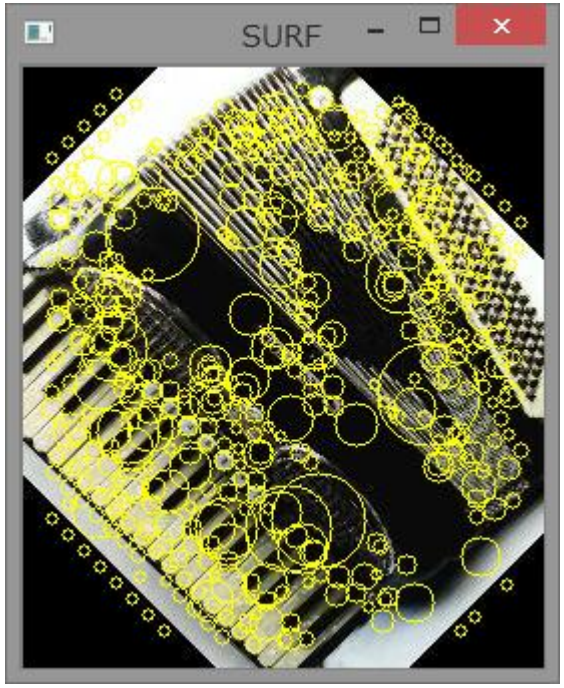


Overview of Today's Lesson

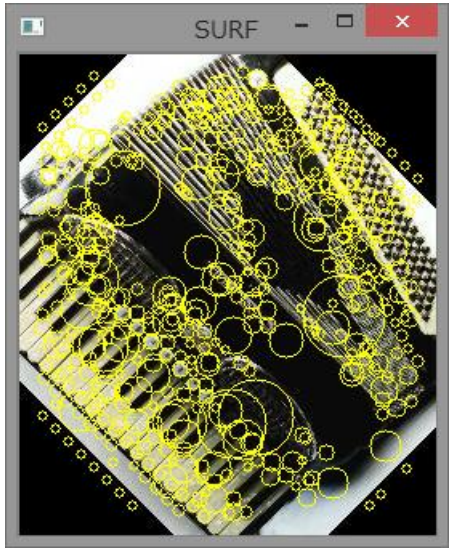
- Structure of SURF features extracted by OpenCV
  - ❑ How to access each SURF feature
- SURF Feature Matching
  - ❑ Searching the most similar SURF feature to the reference one
  - ❑ Computing the similarity between two SURF features
- Extracting SURF features from all images

Code for SURF Feature Extraction

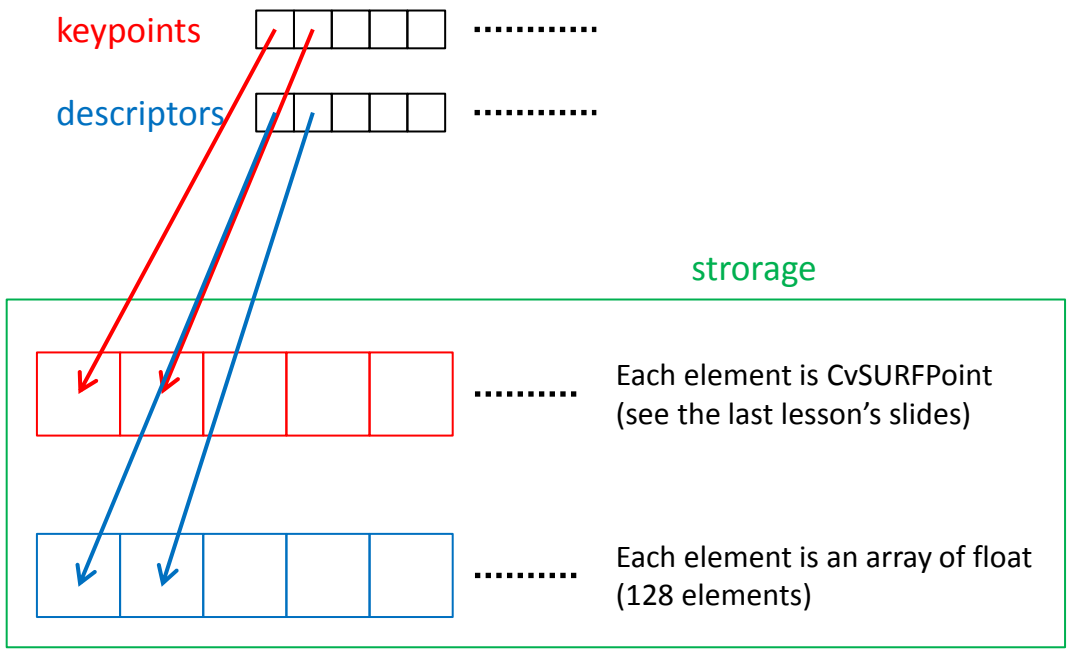
```
#include "opencv2\opencv.hpp"
#include "opencv2\nonfree\nonfree.hpp"

#pragma comment(lib,"C:\\opencv\\build\\x86\\vc10\\lib\\opencv_nonfree246d.lib")

#pragma comment(lib,"C:\\opencv\\build\\x86\\vc10\\lib\\opencv_nonfree246.lib")

int main(int argc, char* argv[]){
    cv::initModule_nonfree(); // Very Important: Initialization of the nonfree library

    // Load an image in gray-scale mode (Used to SURF feature extraction)
    // Load the same image in color mode (Used to display extracted SURF features)

    CvMemStorage* storage = cvCreateMemStorage(0);
    CvSeq* keypoints = 0;
    CvSeq* descriptors = 0;
    CvSURFParams params = cvSURFParams(500, 1);

    // Extract SURF features from img
    cvExtractSURF(img, 0, &keypoints, &descriptors, storage, params);
    cout << ">> # of extracted SURF features = " << descriptors->total << endl;

    // Draw extracted keypoints (local regions from each a SURF descriptor is extracted)
    for(int i = 0; i < keypoints->total; i++){
        CvSURFPoint* point = (CvSURFPoint*)cvGetSeqElem(keypoints, i);
        CvPoint center; // Center of a keypoint
        int radius; // Radius of a keypoint (local region size)
        center.x = cvRound(point->pt.x);
        center.y = cvRound(point->pt.y);
        radius = cvRound(point->size * 1.2 / 9.0 * 2.0);
        cvCircle(img2, center, radius, cvScalar(0,255,255), 1, 8, 0);
    }

    // Show the drawn image using cvNamedWindow

    cvClearSeq(descriptors);
    cvClearSeq(keypoints);
    cvReleaseMemStorage(&storage);
    // Other variables should be released here
}
```



Structure of Extracted SURF Features




```
CvMemStorage* storage = cvCreateMemStorage(0);
CvSeq* keypoints = 0; // local regions
CvSeq* descriptors = 0;
CvSURFParams params = cvSURFParams(500, 1);

// Extract SURF features from img
cvExtractSURF(img, 0, &keypoints, &descriptors, storage, params);
```



Accessing Elements in CvSeq

CvSeq: Variable-length array in OpenCV

```
#define CV_SEQUENCE_FIELDS() \
    …
    int total; /* Number of elements in CvSeq */ \
    …

typedef struct CvSeq{
    CV_SEQUENCE_FIELDS()
} CvSeq;
```

To get the i-th element in CvSeq, you need to specify the variable type of this element

- CvSURFPoint* pt = (CvSURFPoint*)cvGetSeqElem(keypoints, i);
- float* desc = (float*)cvGetSeqElem(descriptors, i);

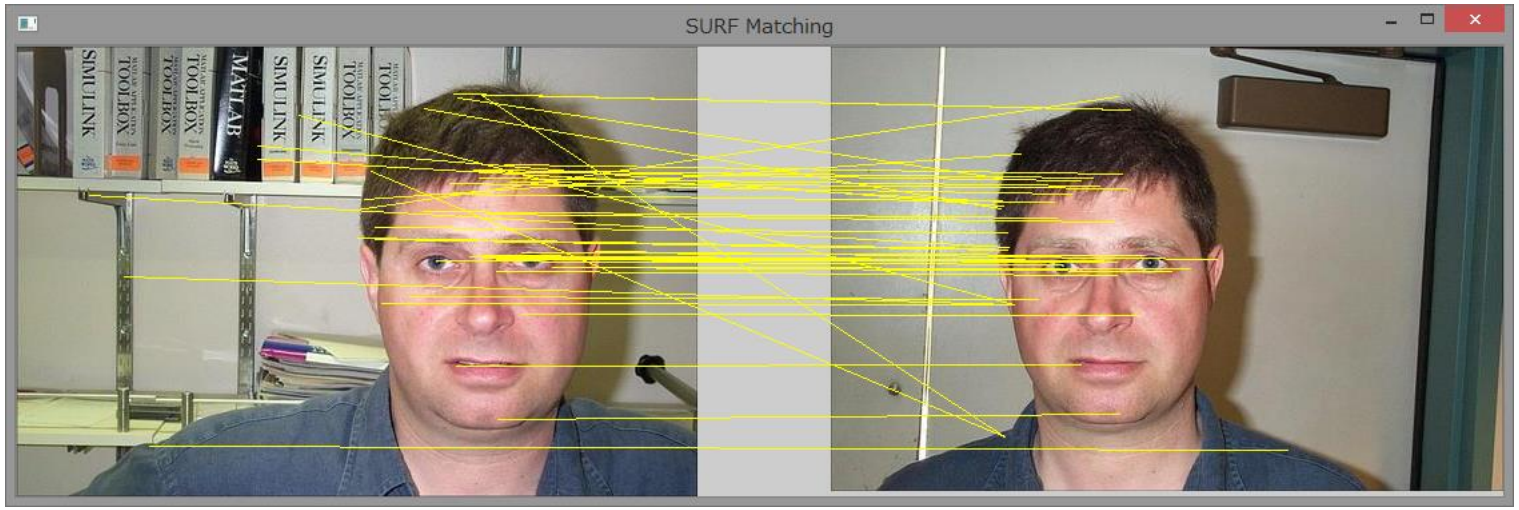
Overview of SURF Feature Matching (1/2)

```
const int margin = 100;

int main(int argc, char* argv[]){

    cv::initModule_nonfree();
    const char* img_filename1 = "(some filename)";
    const char* img_filename2 = "(some filename)";

    // Load images in grays-scale, from which SURF features are extracted
    // Assume two IplImage* variables, gray_img1 and gray_img2

    // Load images in color-mode, which are used to create the image of a matching result
    // Asuume two IplImage* variables, color_img1 and color_img2

    // Create a big IplImage, which shows the matching result
    // Compare the height of color_img1 to that of color_img2, and use the larger one as the height of match_img
    CvSize sz = cvSize( color_img1->width + color_img2->width + margin,
                        ((color_img1->height >= color_img2->height) ? color_img1->height : color_img2->height) );
    IplImage* match_img = cvCreateImage(sz, IPL_DEPTH_8U, 3);
    // Please check the following function by referring to the OpenCV reference
    cvSetImageROI(match_img, cvRect(0, 0, color_img1->width, color_img1->height));
    cvCopy(color_img1, match_img);
    cvSetImageROI(match_img, cvRect(color_img1->width+margin, 0, color_img2->width, color_img2->height));
    cvCopy(color_img2, match_img);
    cvResetImageROI(match_img);

    // Prepare variables needed for SURF feature extraction
    CvSeq *keypoints1 = 0, *descriptors1 = 0;
    CvSeq *keypoints2 = 0, *descriptors2 = 0;
    CvMemStorage* storage = cvCreateMemStorage(0);
    CvSURFParams params = cvSURFParams(500, 1);

    cvExtractSURF(gray_img1, 0, &keypoints1, &descriptors1, storage, params);
    cvExtractSURF(gray_img2, 0, &keypoints2, &descriptors2, storage, params);
```

Continue to the next slide…

Overview of SURF Feature Matching (2/2)

```
    cout << ">> # of SURF features (1) = " << descriptors1->total << endl;
    cout << ">> # of SURF features (2) = " << descriptors2->total << endl;

    // SURF feature matching
    vector<int> ptpairs; // Two consecutive integers represents the IDs of matched SURF features
    findPairs(keypoints1, descriptors1, keypoints2, descriptors2, ptpairs);

    for(int i = 0; i < (int)ptpairs.size(); i += 2){
        CvSURFPoint* pt1 = (CvSURFPoint*)cvGetSeqElem(keypoints1, ptpairs[i]);
        CvSURFPoint* pt2 = (CvSURFPoint*)cvGetSeqElem(keypoints2, ptpairs[i + 1]);
        CvPoint from = cvPointFrom32f(pt1->pt);
        CvPoint to = cvPoint( cvRound(color_img1->width + pt2->pt.x + margin), cvRound(pt2->pt.y) );
        cvLine(match_img, from, to, cvScalar(0, 255, 255));
    }

    cvNamedWindow("SURF Matching");
    cvShowImage("SURF Matching", match_img);
    cvWaitKey(0);

    // Release allocated memory regions

}
```



SURF Feature Matching

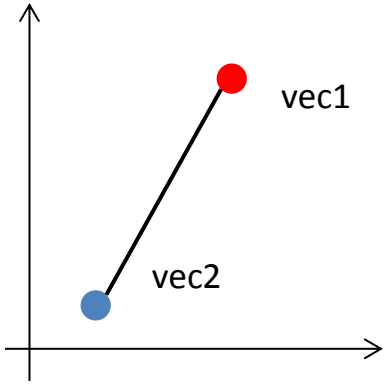
```
void findPairs(CvSeq* keypoints1, CvSeq* descriptors1, // Keypoints and descriptors for SURF features extracted from image1
               CvSeq* keypoints2, CvSeq* descriptors2, // Keypoints and descriptors for SURF features extracted from image2
               vector<int>& ptpairs){ // vector where two consecutive integers represent the IDs of matched SURF features

    // For each SURF feature in image1, find the most similar SURF feature in image2
    for(int i = 0; i < descriptors1->total; i++){

        // Get the i-th SURF feature in image1
        CvSURFPoint* pt1 = (CvSURFPoint*)cvGetSeqElem(keypoints1, i);
        float* desc1 = (float*)cvGetSeqElem(descriptors1, i);

        // Among SURF features of images, find the one which is the most similar to the i-th SURF feature of image1
        // nn represents the ID of the found SURF feature
        int nn = nearestNeighbor(desc1, pt1->laplacian, keypoints2, descriptors2);

        // "nn < 0" means that all of SURF features of image2 are not so similar to the i-th SURF feature of image1
        // If the SURF feature of image2 which is sufficiently similar to the i-th SURF feature of image1, register their IDs
        if(nn >= 0){
            ptpairs.push_back(i);
            ptpairs.push_back(nn);
        }

    }

}
```

Searching the Most Similar SURF Feature

```
const int DIM_VECTOR = 128;    // Number of dimensions of a SURF feature
const double THRESHOLD = 0.32;  // Threshold for matching SURF features

int nearestNeighbor(float* vec, int laplacian,
                    CvSeq* keypoints, CvSeq* descriptors){

    int neighbor = -1; double minDist = 1e6;
    // Search all SURF features
    for(int i = 0; i < descriptors->total; i++){

        CvSURFPoint* pt = (CvSURFPoint*)cvGetSeqElem(keypoints, i);
        // If the laplacian of the i-th SURF feature is different  from that of the reference, ignore it (see the last lesson's slides)
        if (laplacian != pt->laplacian) continue;
        float* v = (float*)cvGetSeqElem(descriptors, i);
        double d = euclidDistance(vec, v, DIM_VECTOR);

        // If the i-th SURF feature is more similar to the reference than the previous ones, update minDist and neighbor
        if(d < minDist){ minDist = d; neighbor = i; }

    }

    // If the found SURF feature is sufficiently similar to the reference, return its ID, otherwise return -1
    if (minDist < THRESHOLD)
        return neighbor;
    else
        return -1;

}
```

Computing the Similarity between Two SURF Features

Euclidian distance: The distance between two points in an n-dimensional space



double euclidDistance(float* vec1, float* vec2, int length){

Compute $\sqrt{\sum_{i=0}^{length-1}(vec1_i - vec2_i)^2}$

}

Extract SURF Features from All Images

By referring to the histogram extraction program, please try to extract SURF features from all images, and save them into text files

NOTE:

1. From one image, more than one hundred SURF features are extracted, so the file size is very big. For each category, extract SURF features only from the first 10 images (image_0001-0010.jpg). Even so, the file size (surf_features.txt) becomes more than 300MB.

2. I recommend you to create the following two files:

(ids.txt: Each line shows "<(global) image ID> <filename>"

0 C:\Users\公章\Documents\MM_course\101_ObjectCategories\accordion\image_0001.jpg
1 C:\Users\公章\Documents\MM_course\101_ObjectCategories\accordion\image_0002.jpg
2 C:\Users\公章\Documents\MM_course\101_ObjectCategories\accordion\image_0003.jpg
…

(surf_features.txt: Each line shows one SURF feature extracted from an image)
The format of each line is "<ImageID> <Laplacian> <SURF features (128 real numbers)>".

These lines show SURF features extracted from the 0-th image

0 -1 0.000508623 0.000582028 0.00310504 0.00313327 …
0 -1 -0.000545836 0.00198606 -0.00487012 0.00736166 …
0 -1 0.00668763 0.00689368 -0.000423902 0.00164965 …
…
1 -1 -0.000486499 0.000777787 2.71906e-005 2.71906e-005 …
1 -1 0.00660171 0.00672847 -0.000191882 0.00413228 …
1 -1 -0.000414784 0.000560557 -0.00837432 0.00903313 …
…