Basic Structure

Matches

Store the match information of two image.

```
struct Matches
{
    int count;//the number of matched keypoints
    vector<Point2f> trainPts;//The coordinate of keypoints in train image.
    vector<Point2f> queryPts;//The coordinate of keypoints in query image.
    vector<int> trainIdxs;// The indices of matched training keypoints in the original keypoints array.
    int refIdx;//This is the index of reference frame which matches best for the current frame.
    float *data;  //data包含count组数据，每组4个float，(x_1,y_1,x_2,y_2)，如果第一帧那么x_1=x_2,y_1=y_2
};
```

MatchList

```
typedef std::vector<std::pair<int, int> > MatchList; // keypoint indices between two images.
```

ANNparams

```
struct ANNparams
{
    ANNpointArray pa;//point array pointer
    ANNkd_tree *kd;//kd-tree pointer
    int npoints;//the number of points
    int d;//point dimension
};
```

Feature_Match

Class Feature_Match

Get several best matched images of the input image.

Feature_Match::Feature_Match(string featureType,int clusterCount)

The Feature_Match constructor.

Parameters

featureType – the type of feature used to match(SIFT or SURF).

clusterCount – the number of cluster used in BoWMather class.

void Feature_Match::setRefs(vector<Mat> &images)

Set the reference images.

Parameters

images – a vector of mat storing all the reference images.

void Feature_Match::addRef(Mat image)

Add one reference image to the training set.

Parameters

image – The reference image to add.

void train()

After adding all the reference images, call the train method to train internal BoWMatcher.

(1) void matchNimages(Mat image, int n,vector<Matches> &matches,int &keyPointSize, vector<int> &matchPointSize, double &detectfps, double &matchfps)

(2) void matchNimages(Mat image, int n, vector<Matches> &matches)

After training, use this method to get the n best matched images of the input image.

Parameters

image – the imput image.

n – the number of images to return

matches – the matched keypoints pair of each matched image.

Class BoWMatcher

Using bag of words model to match the images.

BoWMatcher::BoWMatcher(int _clusterCount,string _feature);

The BoWMatcher constructor.

Parameters:

_clusterCount: Number of clusters to split the set by.

_feature : The feature type of the descriptor to cluster.

void BoWMatcher::addDescriptor(Mat &descriptor)

Add one descriptor to the training set.

Parameters:

Descriptor – the descriptor to add.

void BoWMatcher::train()

After adding all the descriptor, use this method to training the matcher.

int BoWMatcher::match(Mat Descriptor,int n,vector<int> &matches)

Get n best matched descriptors.

Parameters:

Descriptor: the query descriptor.

n; the number of matched descriptors to return

matches: return the indices of matched descriptors in the refDescritor set.

Return:

Return the best matched descriptor index.

Class ANN_Matcher

Use the approximate nearest neighborhood (ANN) method to match keypoints.

ANN_Matcher::ANN_Matcher(string featureType)

The ANN_Matcher constructor.

Parameters:

featureType – the feature type to match(SIFT or SURF).

void ANN_Matcher::setRefs(Mat &descriptor)

Set the reference keypoints set.

Parameters:

descriptor – Set reference feature keypoints, each row is a keypoint.

MatchList ANN_Matcher::match(Mat descriptor)

Calculate the approximate nearest neighborhoods of every keypoints.

Parameters:

descriptor – the query keypoints, each row is a keypoint.

Return:

Return the indices pair of training keypoint and query keypoint

Class CasHash

Use the cascade hash method to match keypoints.

CasHash::CasHash(string featureType)

The CasHash constructor.

Parameters:

featureType – the feature type of keypoint(SIFT or SURF)

void CasHash::setRefs(vector<Mat> refDescriptors)

Set the reference image descriptors.

Parameters:

refDescriptors – the set of reference image descriptors, each row of descriptor is a keypoint.

void CasHash::addRef(Mat refDescriptor)

Add the reference image descriptor.

Parameters:

refDescriptor – the reference image descriptor to add.

void CasHash::train()

After setting the reference image, call this method to training the reference image descriptors.

Feature_Track

Class Feature_Track

The class Feature_Match provide the method to get the N candidate images to track, and the Class Feature_Track is used to determine the best image to track using binary features such as BRISK or ORB.

Feature_Track::Feature_Track(string feature)

The Feature_Track constructor.

Parameters:

feature - the type of feature to use(BRIEF,ORB or BRISK)

void Feature_Track::match(Mat train, Mat query, Matches &matches)

Input the train image and the query image, return the matched keypoints pair.

Parameters:

train - the train image.

query - the query image.

matches - return the matched keypoints pair.