## Software/IMU

### OpenMVG\_IMU\_main\_IncrementalSfM\_robust\_initialization

#### Purpose

The interface aims to provide an incremental SfM with robust initialization in two way(geometry multiple model / IMU validation).

#### 1.1.2 I/O

##### 1.1.2.1 Input

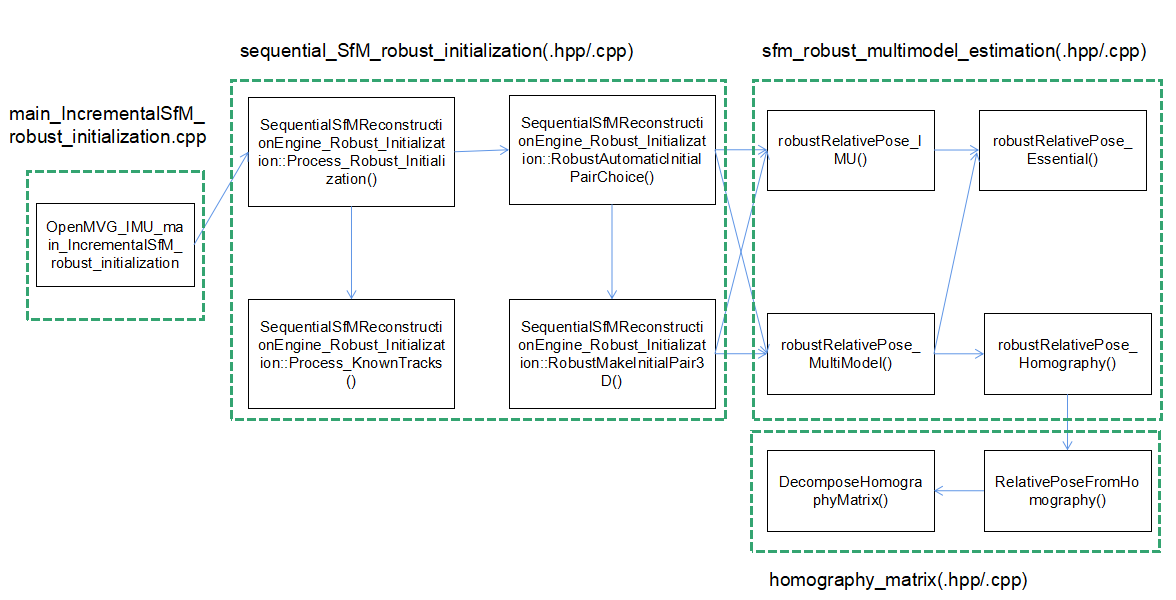
* sSfM\_Data\_Filename(“-i”): (required) input path to a SfM\_Data scene.
* sMatchesDir(“-m”): (required) path to the matches that corresponds to the provided SfM\_Data scene.
* sOutDir(“-o”): (required) path where the output data will be stored.
* sMatchFilename(“-M”): (optional) path to the match file to use (default=matches.f.txt then matches.f.bin).
* initialPairString.first(“-a”): (optional) filename of the first image (without path)
* initialPairString.second(“-b”): (optional) filename of the second image (without path)
* i\_User\_camera\_model(“-c”): (optional) Camera model type for view with unknown intrinsic:
  + 1: Pinhole
  + 2: Pinhole radial 1
  + 3: Pinhole radial 3 (default)
  + 4: Pinhole radial 3 + tangential 2
  + 5: Pinhole fisheye
* sIntrinsic\_refinement\_options(“-f”): (optional) Intrinsic parameters refinement option.
* prior\_usage(“-P”): (optional) Enable usage of motion priors (i.e GPS positions) (default: false)
* triangulation\_method(“-t”): (optional) triangulation method (default=3)
  + 0: DIRECT\_LINEAR\_TRANSFORM
  + 1: L1\_ANGULAR
  + 2: LINFINITY\_ANGULAR
  + 3: INVERSE\_DEPTH\_WEIGHTED\_MIDPOINT
* **initial\_max\_iteration\_count(“-d”)**: (optional) the maximum iteration number of finding automatically initial pair(default = 4096)
* **sIMU\_Data\_Filename(“-u”)**: (optional) the path to a IMU Pose file used for validation of essential matrix.
* **b\_robust\_initialization\_of\_imu(“-r”)**: Enable usage of imu validation in initialization.Otherwise,enable usage of multiple model in initialization.(default: false)

##### 1.1.2.2 Output

* sfm\_data.bin: the output reconstrution file, which contains information(views/intrinsic/extrinsic) of every image and observations of every 3d points.

#### 1.1.3 Pipeline

* Load all parameters required(input SfM\_Data scene, features, matches and so on).
* Construct robust incremental SfM engine class `SequentialSfMReconstructionEngine\_Robust\_Initialization`
* Start reconstrution
* Save the reconstruction file `sfm\_data.bin`



### openMVG\_IMU\_main\_ComputeMatches\_optical\_filtering

#### 功能

计算特征匹配，并用光流过滤

#### 输入

* sSfM\_Data\_Filename(“-i”): （必需），输入的SfM文件（存有图像信息和相机信息）
* sMatchesDirectory(“-o”): （必需），输出文件地址
* fDistRatio(“-r”): （可选），描述子距离阈值，为丢弃无意义的匹配
* sGeometricModel(“-g”)：（可选），几何验证的模型
* iMatchingVideoMode(“-v”)：（可选），序列化匹配的重叠图像数目
* sPredefinedPairList(“-l”)：（可选），预先定义的匹配方式文件地址
* sNearestMatchingMethod(“-n”)：（可选），特征匹配的方法
* bForce(“-f”)：（可选），如果已存在匹配，是否重新计算
* bGuided\_matching(“-m”)：（可选），是否使用已存在模型去提升特征匹配
* cache\_size(“-c”)：（可选），匹配过程使用缓存大小
* bin\_dir(“-b”)：（可选），光流跟踪文件所在文件夹
* MaxDistanceThreshold(“-k”)：（可选），光流过滤的最大距离阈值（默认10.0）
* bnotablefeaturesdetection(“-p”)：（可选），是否识别显著特征（默认为假）
* bnotablevalidation(“-q”)：（可选），是否用显著特征进行过滤（默认为假）

#### 输出

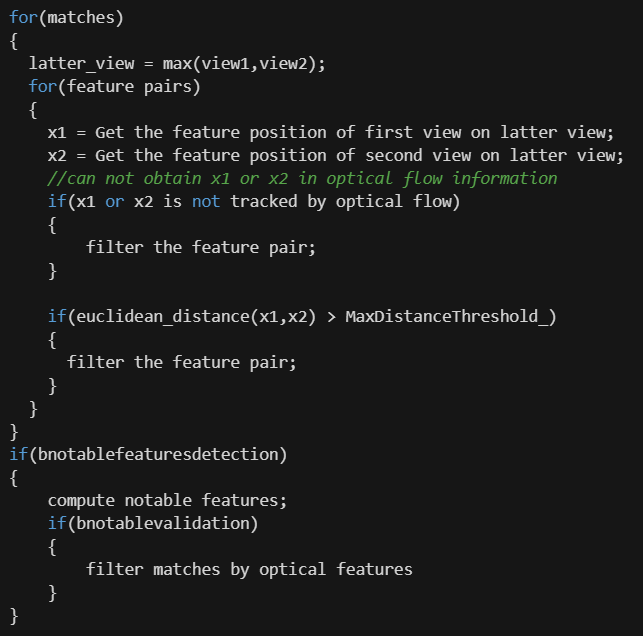
* matches.putative.bin：基于Sift特征匹配得到的matches文件
* matches.opticalfiltering.bin：光流过滤后的matches文件
* matches.f.bin：几何过滤后的matches文件

#### 流程

* 读入参数（特征等）
* 基于Sift的特征匹配
* 光流过滤匹配
* （计算显著特征）
* 几何验证过滤匹配

#### Function OpticalFiltering()

* 功能：用光流过滤特征匹配
* 输入
  + regions\_provider：特征信息
  + my\_progress\_bar：进度条ui
  + map\_PutativesMatches：匹配信息
  + bin\_dir：光流跟踪文件所在文件夹
  + MaxDistanceThreshold：光流过滤的最大距离阈值
  + SfM\_Data：提供图像信息
  + matches\_dir：输出的matches文件地址
  + bnotablefeaturesdetection：是否识别显著特征
  + bnotablevalidation：是否用显著特征进行过滤
* 输出
  + map\_PutativesMatches：光流过滤后的匹配信息
* 流程：



### openMVG\_IMU\_main\_ComputeMatches\_optical\_matching

#### 功能

层次式的匹配方法（特征双向验证，光流过滤，光流匹配）

#### 输入

* sSfM\_Data\_Filename(“-i”): （必需），输入的SfM文件（存有图像信息和相机信息）
* sMatchesDirectory(“-o”): （必需），输出文件地址
* fDistRatio(“-r”): （可选），描述子距离阈值，为丢弃无意义的匹配
* sGeometricModel(“-g”)：（可选），几何验证的模型
* iMatchingVideoMode(“-v”)：（可选），序列化匹配的重叠图像数目
* sPredefinedPairList(“-l”)：（可选），预先定义的匹配方式文件地址
* sNearestMatchingMethod(“-n”)：（可选），特征匹配的方法
  + OPTICALFLOW：进行光流匹配
  + HIERARCHICAL：进行层次式匹配
* bForce(“-f”)：（可选），如果已存在匹配，是否重新计算
* bGuided\_matching(“-m”)：（可选），是否使用已存在模型去提升特征匹配
* cache\_size(“-c”)：（可选），匹配过程使用缓存大小
* bin\_dir(“-b”)：（可选），光流跟踪文件所在文件夹
* MaxDistanceThreshold(“-k”)：（可选），光流过滤的最大距离阈值（默认10.0）
* bfeature\_validation(“-a”)：（可选），是否进行特征双向验证（默认为假）
* bopticalfiltering(“-d”)：（可选），是否进行光流过滤（默认为真）
* bdynamicdistance(“-e”)：（可选），是否使用动态的光流阈值（默认为假）
* bopticalmatching(“-h”)：（可选），是否进行光流匹配（默认为假）
* bSolveArticulationPoint(“-j”)：（可选），是否解决匹配图中的割点问题（默认为假）
* bnotablefeaturesdetection(“-p”)：（可选），是否识别显著特征（默认为假）
* bnotablevalidation(“-q”)：（可选），是否用显著特征进行过滤（默认为假）

#### 输出

* matches.putative.bin：层次匹配后得到的matches文件
* matches.f.bin：几何过滤后的matches文件

#### 流程

* 读入参数（特征等）
* 层次式匹配
  + 特征匹配
    - Sift 特征匹配
    - 双向验证 <- bfeature\_validation
  + 光流过滤 <- bopticalfiltering
    - 静态阈值
    - 动态阈值 <- bdynamicdistance
  + 计算显著特征 <- bnotablefeaturesdetection
    - 显著特征验证 <- bnotablevalidation
  + 光流匹配 <- bopticalmatching
* 几何验证过滤匹配
* 识别匹配结构中的割点 <- bSolveArticulationPoint
  + 识别割点，将新组成的图像对进行光流匹配
  + 几何验证过滤匹配

## OpenMVG\_IMU/sfm

### pipeline/sequential

#### sequential\_SfM(.hpp/.cpp)

##### Class SequentialSfMReconstructionEngine\_General

###### Purpose

A sequential SfM pipeline reconstruction engine class copy from OpenMVG/sfm/pipeline/sequential/sequential\_SfM(.hpp/.cpp) is used for inheriting , where all member variables and functions in the class are declared as public

#### sequential\_SfM\_robust\_initialization(.hpp/.cpp)

##### Class SequentialSfMReconstructionEngine\_Robust\_Initialization

###### Purpose

A sequential SfM pipeline reconstruction engine with robust initialization derived from SequentialSfMReconstructionEngine\_General.

##### Constructor SequentialSfMReconstructionEngine\_Robust\_Initialization()

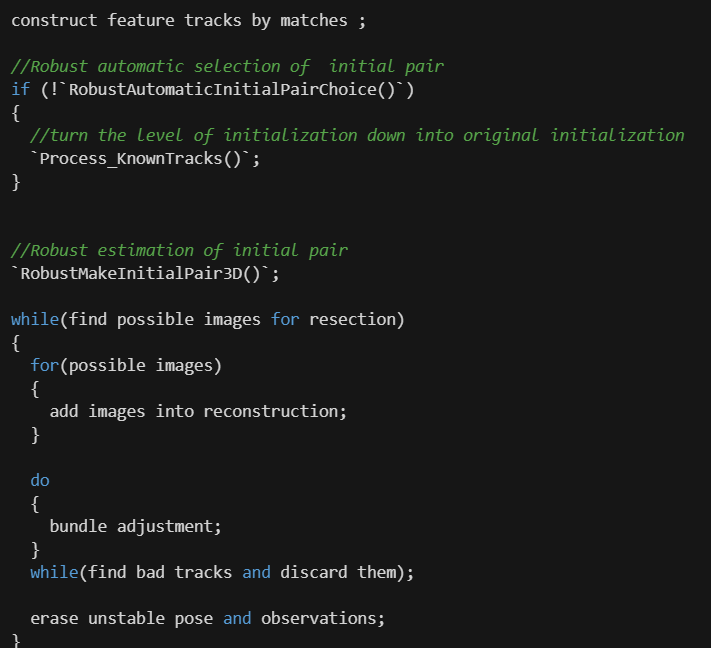
* Purpose: utilize input arguments to construct class SequentialSfMReconstructionEngine\_Robust\_Initialization
* Input:
  + sfm\_data: input sfm data file
  + soutDirectory: path where the output data will be stored.
  + initial\_max\_iteration\_count: the maximum iteration number of finding automatically initial pair
  + loggingFile: the path of SfM report file

##### Member function setIMUData()

* Purpose: Enable the usage of imu validation in initialization
* Input:
  + imu\_data: the input imu data file where store poses of every image

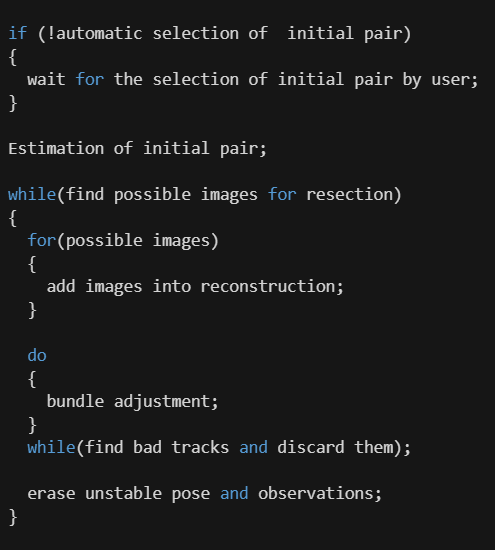
##### Member function Process\_Robust\_Initialization()

* Purpose: incremental reconstruction with robust initialization
* Pipeline:



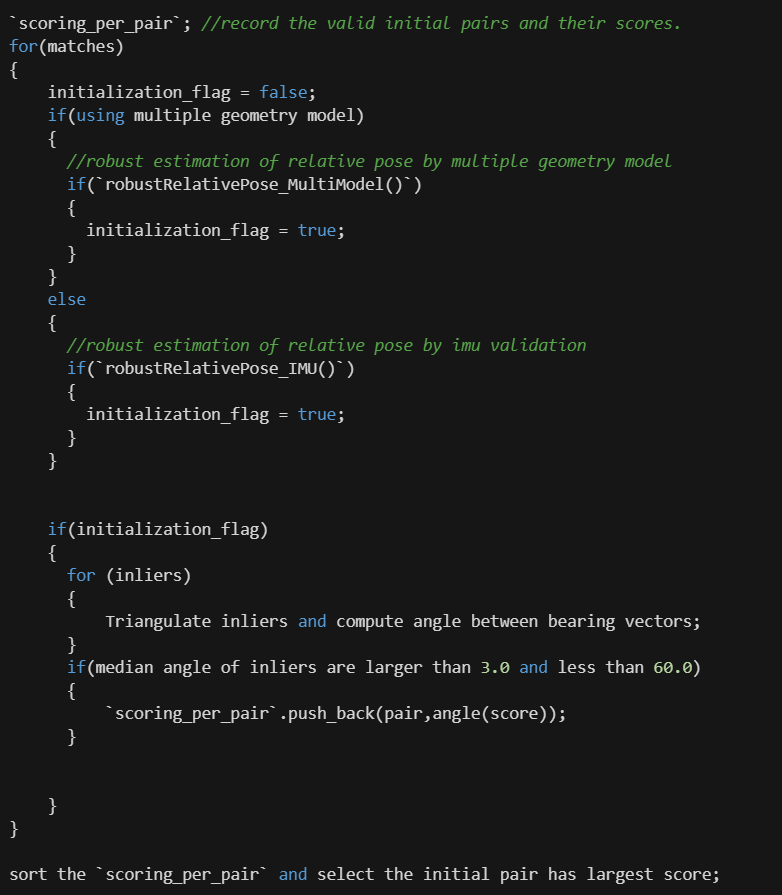
##### Member function Process\_KnownTracks()

* Purpose: incremental reconstruction with original initialization in OpenMVG if we can not find robust initial pair
* Pipeline:



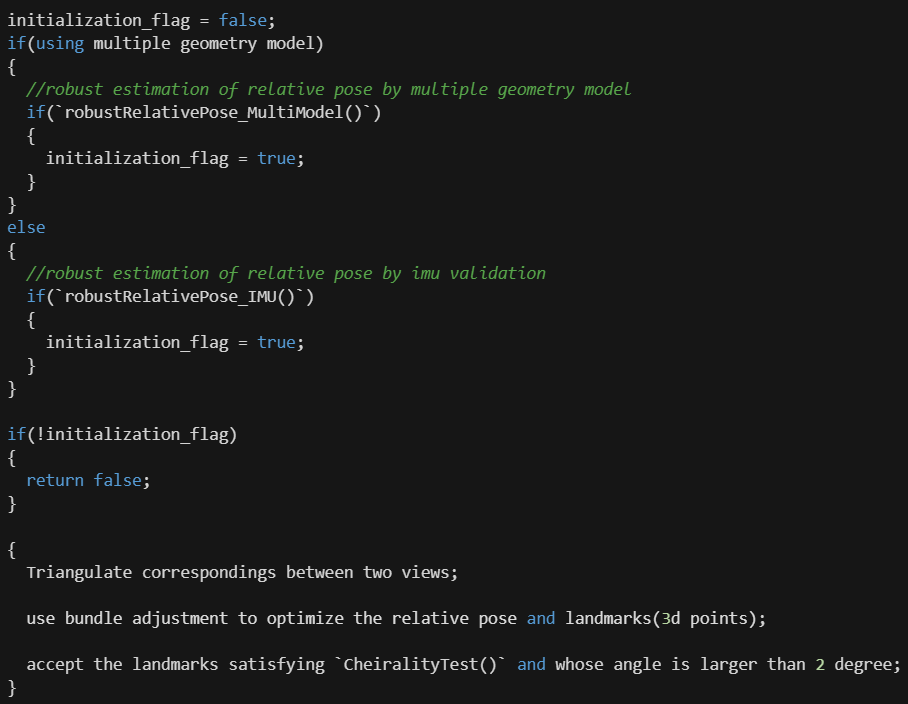
##### Member function RobustAutomaticInitialPairChoice()

* Purpose: select robust initial pair for reconstruction.
* Output:
  + initial pair: the robust initial pair have the largest baseline.
* Pipeline:



##### Member function RobustMakeInitialPair3D()

* Purpose: estimate the relative pose and triangulate correspondings between initial pair.
* Input:
  + initial pair: the robust initial pair have the largest baseline.
* Pipeline:

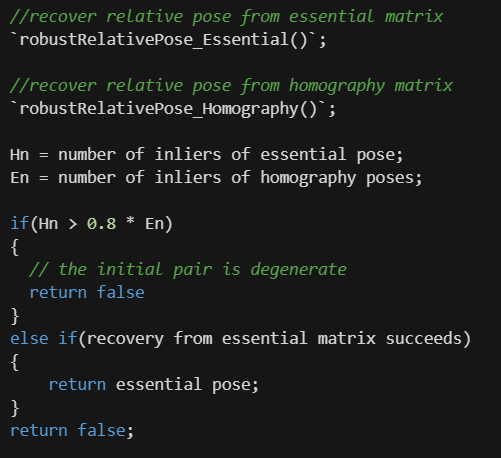


#### 2.2 pipeline/

#### 2.2.1 sfm\_robust\_multimodel\_estimation(.hpp/.cpp)

##### 2.2.1.1 Function robustRelativePose\_MultiModel()

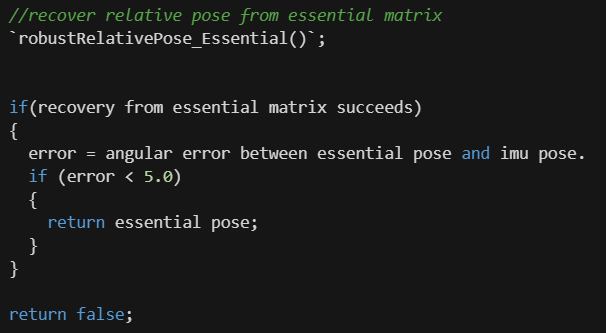
* Purpose: estimate the Relative pose between two view from point matches and K matrices by using a multiple geometry model.
* Input:
  + intrinsics1: camera 1 intrinsics
  + intrinsics2: camera 2 intrinsics
  + x1: image points in image 1
  + x2: image points in image 2
  + size\_ima1: width, height of image 1
  + size\_ima2: width, height of image 2
  + max\_iteration\_count: max iteration count(default = 4096)
* Output:
  + relativePose\_info : relative pose information computed
* Pipeline:



* Threshold: 0.8(ratio of Hn and En)

##### 2.2.1.2 Function robustRelativePose\_IMU()

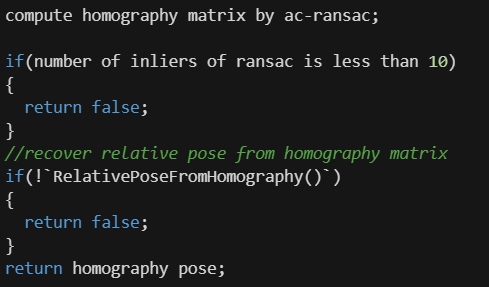
* Purpose: Estimate the Relative pose between two view from point matches and K matrices by using a robust essential matrix estimation with imu validation.
* Input:
  + intrinsics1: camera 1 intrinsic
  + intrinsics2: camera 2 intrinsic
  + x1: image points in image 1
  + x2: image points in image 2
  + imu\_relative\_pose : relative pose information from IMU
  + size\_ima1: width, height of image 1
  + size\_ima2: width, height of image 2
  + max\_iteration\_count: max iteration count(default = 4096)
* Output:
  + relativePose\_info : relative pose information computed
* Pipeline:



* Threshold: 5.0(maximum angular error between IMU pose and essential pose)

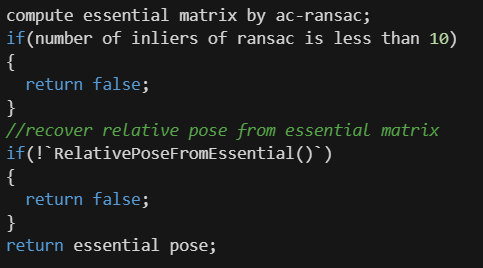
##### 2.2.1.3 Function robustRelativePose\_Homography()

* Purpose: Estimate the Relative pose between two view from point matches and K matrices by using a robust homography matrix estimation.
* Input:
  + intrinsics1: camera 1 intrinsic
  + intrinsics2: camera 2 intrinsic
  + x1: image points in image 1
  + x2: image points in image 2
  + size\_ima1: width, height of image 1
  + size\_ima2: width, height of image 2
  + max\_iteration\_count: max iteration count(default = 4096)
* Output:
  + relativePose\_info : relative pose information computed
* Pipeline:



##### 2.2.1.4 Function robustRelativePose\_Essential()

* Purpose: Estimate the Relative pose between two view from point matches and K matrices by using a robust essential matrix estimation.
* Input:
  + intrinsics1: camera 1 intrinsic
  + intrinsics2: camera 2 intrinsic
  + x1: image points in image 1
  + x2: image points in image 2
  + size\_ima1: width, height of image 1
  + size\_ima2: width, height of image 2
  + max\_iteration\_count: max iteration count(default = 4096)
* Output:
  + relativePose\_info : relative pose information computed
* Pipeline:



## openMVG\_IMU/multiview

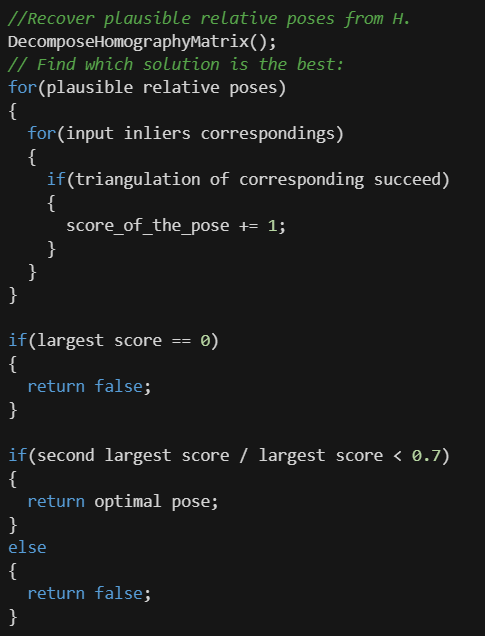
### 3.1. homography\_matrix(.hpp/.cpp)

#### 3.1.1. Function DecomposeHomographyMatrix()

* Purpose: Decompose an homography matrix into the possible rotations, translations,and plane normal vectors taken from COLMAP.
* Input:
  + H: 3x3 homography matrix
  + K: 3x3 calibration matrix
* Output:
  + R: Possible 3x3 rotation matrices
  + t: Possible translation vectors
  + n: Possible normal vectors

#### 3.1.2. Function RelativePoseFromHomography()

* Purpose: Estimate the best possible relative pose from H imitating recovery from E in OpenMVG. Four relative poses can be build from the Hmatrix decomposition. We keep the one with most of the point in front of the camera.
* Input:
  + H: 3x3 homography matrix
  + K1: 3x3 calibration matrix of first camera
  + K2: 3x3 calibration matrix of second camera.
  + x1: bearing vectors corresponding to image observation in image 1
  + x2: bearing vectors corresponding to image observation in image 2
  + bearing\_vector\_index\_to\_use: selection of x1, x2 columns that are used
  + positive\_depth\_solution\_ratio: Pourcentage ratio threshold used to discard if there is two good solution that have many points in front of the cameras(default = 0.7)
* Output:
  + relative\_pose: the estimated relative pose
  + vec\_selected\_points: return the index of bearing\_vector\_index\_to\_use that are in front of the cameras(default = nullptr)
  + vec\_points: return the 3D point corresponding to vec\_selected\_points indexes(default = nullptr)
* Pipeline:



## OpenMVG\_IMU/matching\_image\_collection

### Hierarchical\_Matcher\_Regions(.hpp/.cpp)

#### 4.1.1 Class Hierarchical\_Matcher\_Regions

##### 4.1.1.1 Constructor **Hierarchical\_Matcher\_Regions()**

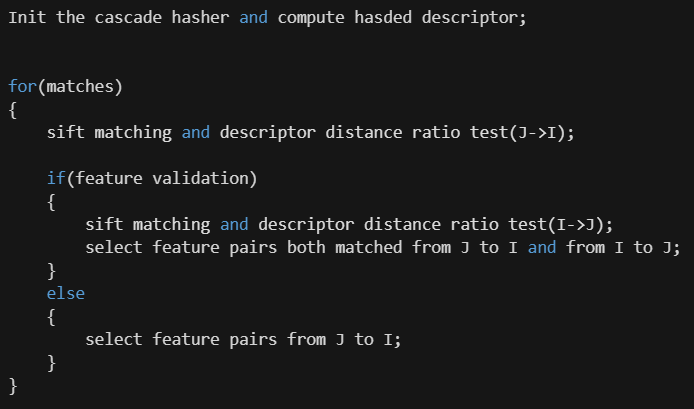
* 输入：
  + fDistRatio: 描述子距离阈值，为丢弃无意义的匹配
  + bin\_dir：光流跟踪文件所在文件夹
  + MaxDistanceThreshold：光流过滤的最大距离阈值
  + SfM\_Data：提供图像信息
  + bfeature\_validation：是否进行特征双向验证
  + bopticalfiltering：是否进行光流过滤
  + bdynamicdistance：是否使用动态的光流阈值
  + bopticalmatching：是否进行光流匹配
  + bnotablefeaturesdetection：是否识别显著特征
  + bnotablevalidation：是否用显著特征进行过滤
* 流程
  + 构造`Hierarchical\_Matcher\_Regions`对象，并依据bin\_dir读取光流文件构造`OpticalFlow\_Container`对象

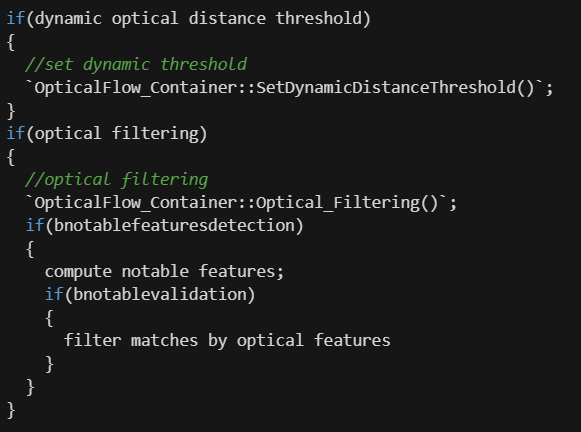
##### 4.1.1.2 Member Function **Match\_Hierarchical()**

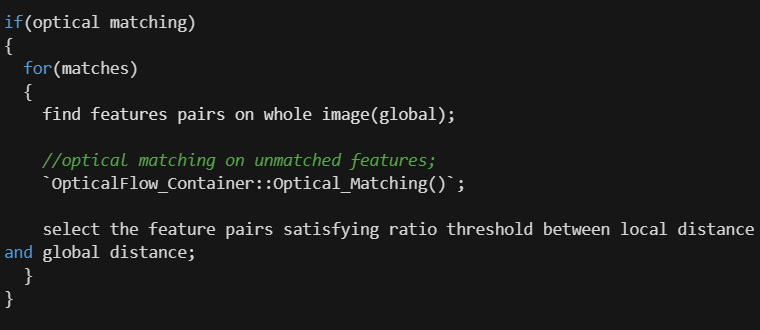
* 输入
  + regions\_provider: 特征信息
  + pairs：光流跟踪文件所在文件夹
  + my\_progress\_bar：进度条ui
* 输出
  + map\_PutativesMatches：特征匹配
* 流程
  + 调用`impl::Match\_Hierarchical()`进行层次式特征匹配

#### 4.1.2 Function impl::Match\_Hierarchical()

* 功能：进行层次式匹配
* 输入
  + regions\_provider：特征信息
  + pairs: 待匹配的图像对
  + fDistRatio：描述子距离阈值，用于过滤无意义的匹配
  + opticalflow\_container：保存和处理光流方法的控制类
  + my\_progress\_bar：进度条ui
  + bfeature\_validation：是否进行特征双向验证
  + bopticalfiltering：是否进行光流过滤
  + bdynamicdistance：是否使用动态的光流阈值
  + bopticalmatching：是否进行光流匹配
  + bnotablefeaturesdetection：是否识别显著特征
  + bnotablevalidation：是否用显著特征进行过滤
* 输出
  + map\_PutativesMatches：计算得到的匹配
  + notable\_features：计算得到的显著特征
* 流程







### Optical\_Flow\_Matcher\_Regions(.hpp/.cpp)

#### 4.2.1 Class Optical\_Flow\_Matcher\_Regions

#### 4.2.1.1 Constructor Optical\_Flow\_Matcher\_Regions()

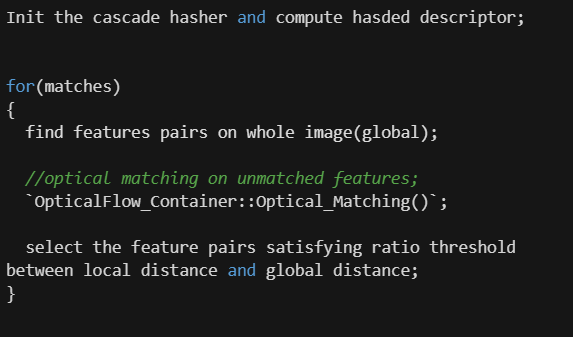
* 输入
  + fDistRatio: 描述子距离阈值，为丢弃无意义的匹配
  + bin\_dir：光流跟踪文件所在文件夹
  + MaxDistanceThreshold：光流过滤的最大距离阈值
  + SfM\_Data：提供图像信息
* 流程：构造`Hierarchical\_Matcher\_Regions`对象，并依据bin\_dir读取光流文件构造`OpticalFlow\_Container`对象

##### 4.1.1.2 Member Function **Match()**

* 输入
  + regions\_provider: 特征信息
  + pairs：光流跟踪文件所在文件夹
  + my\_progress\_bar：进度条ui
* 输出
  + map\_PutativesMatches：特征匹配
* 流程
  + 调用`impl::Match()`进行层次式特征匹配

#### 4.2.2 Funtion impl::Match()

* 功能：进行optical matching
* 输入
  + regions\_provider：特征信息
  + pairs: 待匹配的图像对
  + fDistRatio：描述子距离阈值，用于过滤无意义的匹配
  + opticalflow\_container：保存和处理光流方法的控制类
  + my\_progress\_bar：进度条ui
* 输出
  + map\_PutativesMatches：计算得到的匹配
* 流程



## OpenMVG\_IMU/matching

### cascade\_hasher.(hpp)

#### 5.1.1 Class CascadeHasher\_General

内容与OpenMVG同名类一致，区别是该类中所有成员变量和成员函数被声明为public

### cascade\_hasher2(.hpp)

#### 5.2.1 Class CascadeHasher2

继承自`CascadeHasher\_General`,用于提供一个支持输入mask的匹配函数

### optical\_flow(.hpp)

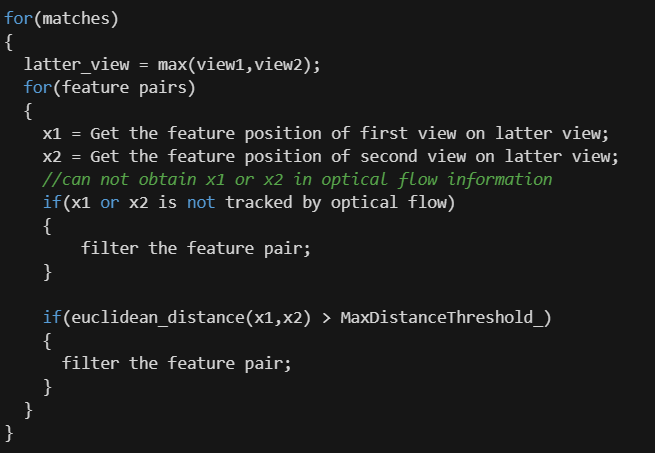
#### 5.3.1 Class OpticalFlow\_Container

##### 5.3.1.1 Constructor OpticalFlow\_Container()

* 功能：读取光流数据，并构造类对象
* 输入：
  + bin\_dir：光流跟踪文件所在文件夹
  + MaxDistanceThreshold：光流过滤的最大距离阈值
  + SfM\_Data：提供图像信息
* 流程：
  + 根据图像view\_id读取光流数据信息
  + 更新成员变量readable为true

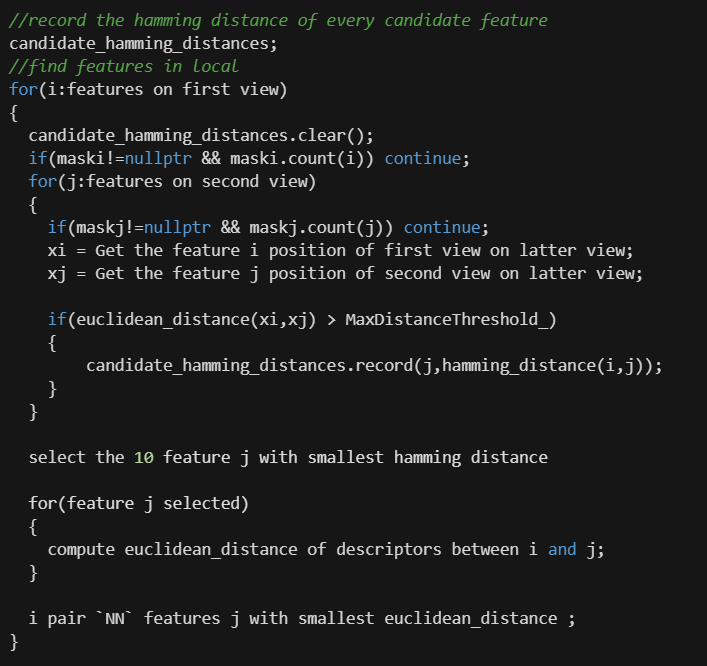
##### 5.3.1.2 Member Function Optical\_Filtering()

* 功能：用光流过滤特征匹配
* 输入
  + regions\_provider：特征信息
  + my\_progress\_bar：进度条ui
  + map\_PutativesMatches：匹配信息
* 输出
  + map\_PutativesMatches：光流过滤后的匹配信息
* 流程：



##### 5.3.1.3 Member Function Optical\_Matching()

* 功能：基于光流信息做特征匹配
* 输入
  + view\_id\_1：第一张图像的id
  + hashed\_descriptions1：第一张图像的hash描述子
  + descriptions1：第一张图像的描述子矩阵
  + view\_id\_2：第二张图像的id
  + hashed\_descriptions2：第二张图像的hash描述子
  + descriptions2：第二张图像的描述子矩阵
  + pts\_1：第一张图像特征的位置
  + pts\_2：第二张图像特征的位置
  + latter\_view\_id：两张图像中时间靠后的图像id
  + nb\_hash\_code：hash描述子的位数
  + paired\_feati：第一张图像特征的mask（默认为nullptr）
  + paired\_featj：第二张图像特征的mask（默认为nullptr）
  + NN：一个特征最少需要找到的特征数量（默认为2）
* 输出
  + pvec\_indices：计算得到的特征匹配
  + pvec\_distances：每一对特征匹配的距离
* 流程



##### 5.3.1.2 Member Function SetDynamicDistanceThreshold()

* 功能：根据输入的匹配信息，计算动态光流距离阈值
* 输入
  + regions\_provider：特征信息
  + my\_progress\_bar：进度条ui
  + map\_PutativesMatches：匹配信息
* 流程：

