## 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

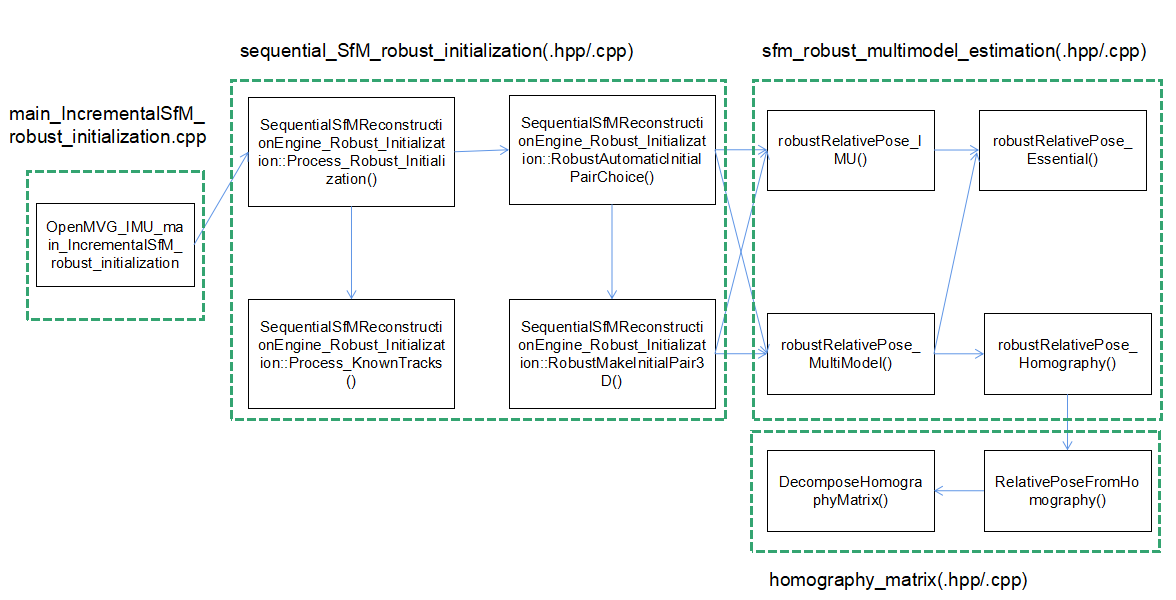
* sSfM\_Data\_Filename(“-i”): (required) input path to a SfM\_Data scene.
* sMatchesDir(“-m”): (required) path to the matches that corresponds to the provided SfM\_Data scene.
* sOutDir(“-o”): (required) path where the output data will be stored.
* sMatchFilename(“-M”): (optional) path to the match file 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/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()

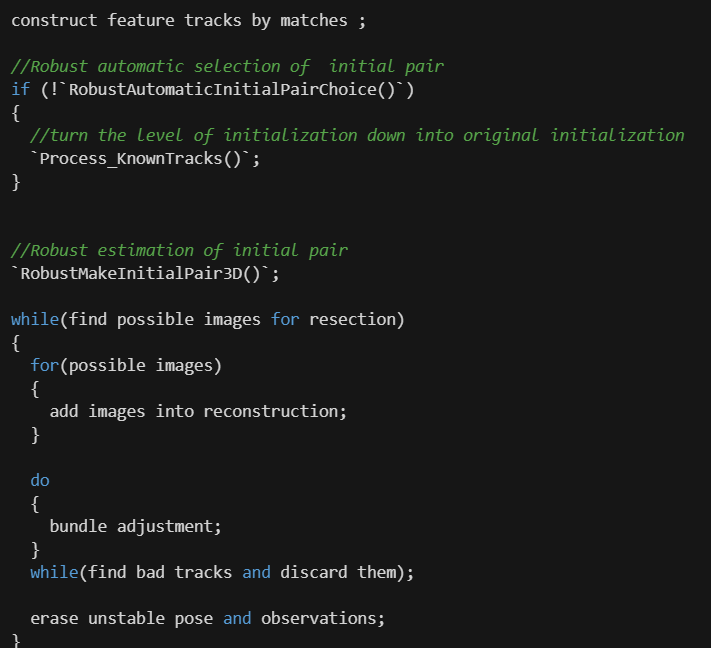
* Purpose: utilize input arguments to construct class SequentialSfMReconstructionEngine\_Robust\_Initialization
* Input:
  + sfm\_data: input sfm data file
  + soutDirectory: path where the output data will be stored.
  + initial\_max\_iteration\_count: the maximum iteration number of finding automatically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

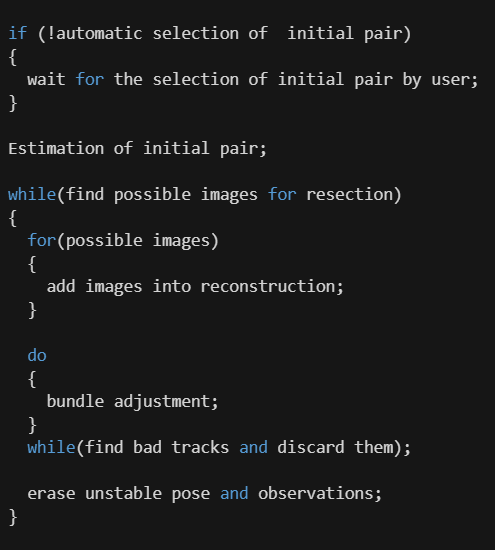
##### Member function Process\_Robust\_Initialization()

* Purpose: incremental reconstruction with robust initialization
* Pipeline:



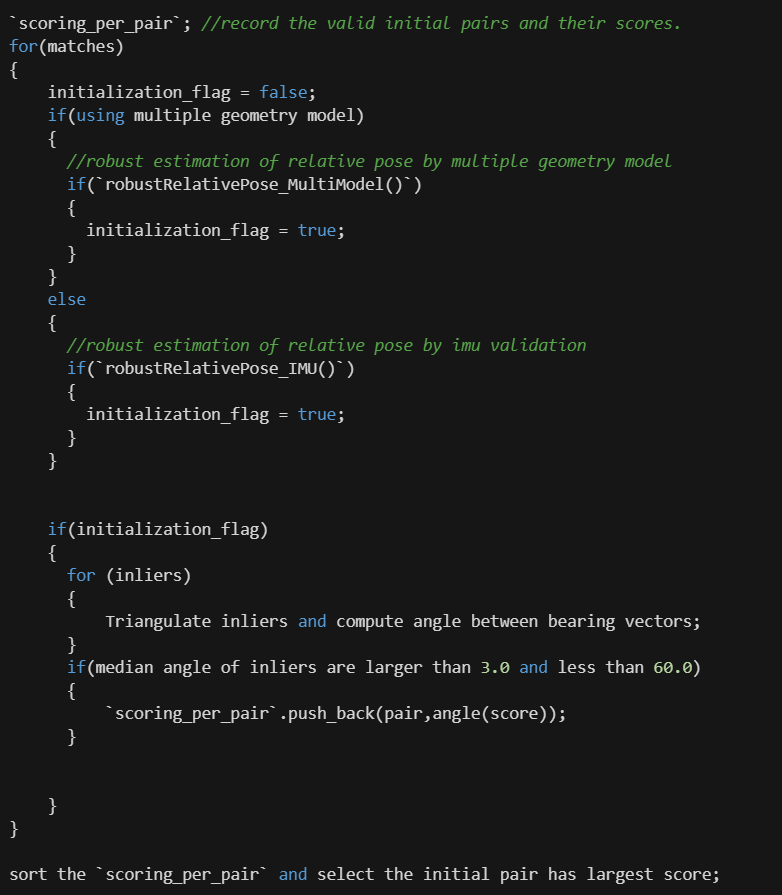
##### Member function Process\_KnownTracks()

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



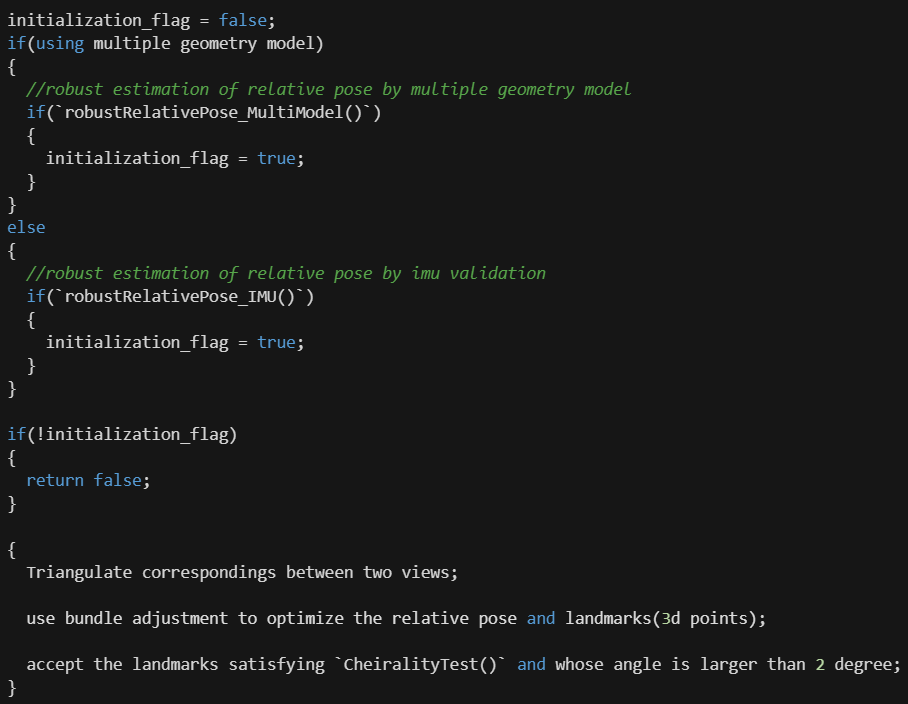
##### Member function RobustAutomaticInitialPairChoice()

* 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:

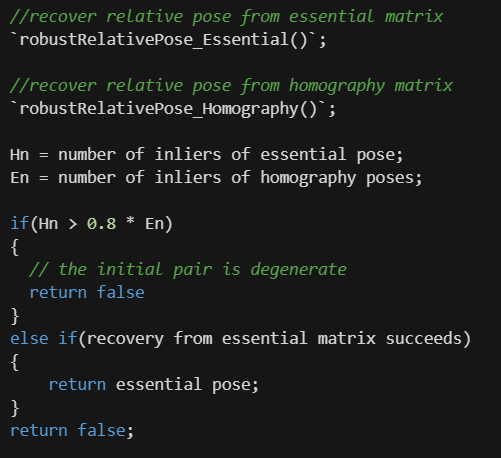


#### 2.2 pipeline/

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

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

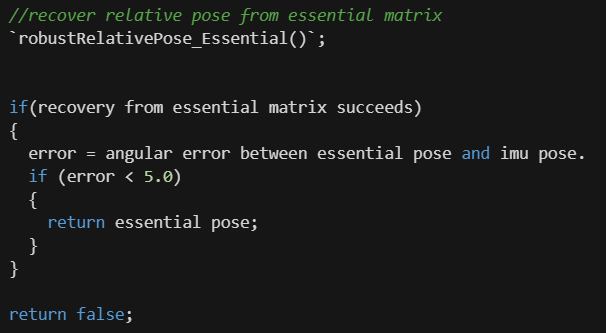
* Purpose: estimate the Relative pose between two view from point matches and K matrices by using a multiple geometry model.
* Input:
  + intrinsics1: camera 1 intrinsics
  + intrinsics2: camera 2 intrinsics
  + x1: image points in image 1
  + x2: image points in image 2
  + size\_ima1: width, height 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()

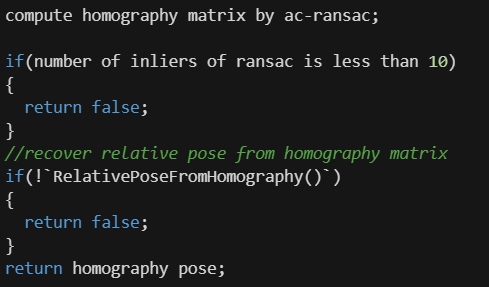
* Purpose: Estimate the Relative pose between two view from point matches and K matrices by using a robust essential matrix estimation with imu validation.
* Input:
  + intrinsics1: camera 1 intrinsic
  + intrinsics2: camera 2 intrinsic
  + x1: image points in image 1
  + x2: image points in image 2
  + 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)

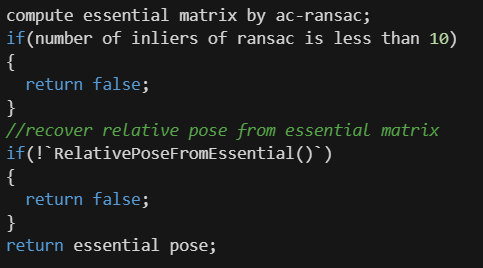
##### 2.2.1.3 Function robustRelativePose\_Homography()

* 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:

