

```

classDiagram
    class SLAM {
        +m_ID: size_t
        +m_KeyFrames: vector<shared_ptr<TrackingFrame>>
        +m_PointCloud: PointCloud<PointXYZRGB>Ptr
        +MapBlockKeyFrames: vector<shared_ptr<TrackingFrame>>, points: const PointCloud<PointXYZRGB>+k)
        +MapBlock(otherBlock: const MapBlock+)
        +operator=(other: const MapBlock+) bool
        +GetID() size_t
        +GetID() size_t
        +GetKeyFrames() vector<shared_ptr<TrackingFrame>>
        +operator=(other: const MapBlock+) bool
        +GetParent() PointCloud<PointXYZRGB>+ConstPtr
    }
    class MapDatabase {
        +m_Blocks: unordered_map<size_t, shared_ptr<MapBlock>>
        +m_PointCloudIndices: map<size_t, tuple<size_t, size_t>>
        +m_CurrentPointCloud: pcl::PointCloud<PointXYZRGB>Ptr
        +m_UpdateMutex: Mutex
        +m_NextID: size_t
    }
    class MappingSystem {
        +m_CameraGraphIDs: unordered_map<size_t, int>
        +m_UnprocessedBlocks: vector<shared_ptr<MapBlock>>
        +m_OptimizationGraph: unique_ptr<OptimizationGraph>
        +m_OpticalFlowEstimator: unique_ptr<OpticalFlowEstimator>
        +m_Reconstructor: shared_ptr<ReconstructIO>
        +m_MapDatabase: shared_ptr<MapDatabase>
        +MappingSystem(reconstructor: shared_ptr<ReconstructIO>)
        +MappingSystem(mappingSystem: shared_ptr<MappingSystem>)
        +StartOptimizationThread() void
        +AddUnprocessedKeyframes(points: const PointCloud<PointXYZRGB>+k, const keyframes: const vector<shared_ptr<TrackingFrame>>+k) void
        +GetMappingCloud: PointCloud<PointXYZRGB>Ptr) void
        +GetMappingDatabase() shared_ptr<MapDatabase>
        +LocalCameraIndex() void
        +FullBA() void
    }
    class Tracker {
        +m_FrameFeatureExtractor: shared_ptr<FrameFeatureExtractor>
        +m_Reconstructor: shared_ptr<ReconstructIO>
        +m_MappingSystem: shared_ptr<MappingSystem>
        +m_KeyframeDatabase: shared_ptr<KeyframeDatabase>
        +m_KeyframePoseVertices: unordered_map<size_t, int>
        +m_Keyframe3DPosVertices: unordered_map<size_t, tuple<int, int>>
        +m_CurrentPose: Mat_4d
        +m_OptimizationGraph: unique_ptr<OptimizationGraph>
        +m_OpticalFlowEstimator: unique_ptr<OpticalFlowEstimator>
        +Tracker(featureExtractor: shared_ptr<FrameFeatureExtractor>, reconstructor: shared_ptr<ReconstructIO>, mappingSystem: shared_ptr<MappingSystem>, keyframeDB: shared_ptr<KeyframeDatabase>)
        +Track()
        +TrackFrame(frame: shared_ptr<TrackingFrame>) void
        +GetFrame() Mat_4d
        +TrackFrame(currentFrame: shared_ptr<TrackingFrame>, recentKeyframe: shared_ptr<TrackingFrame>) void
    }
    class TrackingFrame {
        +m_Reconstructor: shared_ptr<ReconstructIO>
        +m_ID: size_t
        +m_CameraImage: Mat
        +m_Disparity: Mat
        +m_Mask: Mat
        +m_Pose: Isometry3d
        +TrackingFrame(cameraImage: const Mat&, disparity: const Mat&, reconstructor: shared_ptr<ReconstructIO>)
        +TrackingFrame()
        +GetID() size_t
        +GetDescensor(PointCloud) PointCloud<PointXYZRGB>Ptr
        +GetCameraImageAsIs() Mat
        +GetCameraImage() Mat
        +GetDisparity() Mat
        +SetTrackedPose(pose: Isometry3d&) void
        +SetID() size_t
        +operator=(other: const TrackingFrame) bool
        +operator=(other: const TrackingFrame) bool
        +PrintDebugImages(disparity: Mat, mask: Mat) void
        +SetFrame() void
    }
    class KeyframeDatabase {
        +m_NextAvailableID: size_t
        +m_LastInsertedID: size_t
        +m_Keyframes: unordered_map<map<size_t, shared_ptr<TrackingFrame>>
        +m_DatabaseMutex: mutex
        +KeyframeDatabase()
        +InsertKeyframe(shared_ptr<TrackingFrame>) size_t
        +SelectKeyframe(size_t) shared_ptr<TrackingFrame>
        +SelectKeyframe(Keyframe) shared_ptr<TrackingFrame>
        +UpdateKeyframePose(id: size_t, pose: Isometry3d) void
        +IsEmpty() bool
        +GetCount() size_t
    }
    class CameraCompute {
        +m_IsStereoRectified: bool
        +m_StereoSettings: StereoCalib
        +CameraCompute(settings: StereoCalib)
        +FundamentalMatrix(leftImage: const Mat&, rightImage: const Mat&, R: const Mat_3d, RR: const Mat_3d) Mat_3d
        +EssentialMatrix(leftImage: const Mat&, rightImage: const Mat&, R: const Mat_3d, RR: const Mat_3d) Mat_3d
        +GetRectifiedStereoSettings() StereoCalib
        +ComputingFundamentalMatrix(const Mat&, rightImage: const Mat&, pointLeft: vector<Point2d>, pointRight: vector<Point2d>) void
        +RectifyImage(Size: Size) void
    }
    class CameraCalibParser {
        +ParseStereoCalibJSONFile(filePath: const string&, stereoCalib: StereoCalib) void
        +ParseK(json: json) Mat_3d
        +ParseR(json: json) Vector3
        +ParseT(json: json) Vector3
        +ParseMn(json: json, m: int, n: int) Matrix3f
    }
    class StereoCalib {
        +K: Mat_3d
        +D: Vector3f
        +imageUndistort(Pixels: Vector3)
    }
    class StereoRectification {
        +Rt: Mat
        +RR: Mat
        +PR: Mat
        +Q: Mat
        +ValidRectLeft: Rect
        +ValidRectRight: Rect
    }
    class OpticalFlowEstimator {
        +m_FarnebackOF: FarnebackOpticalFlowPtr
        +m_OpticalFlowEstimator()
        +OpticalFlowEstimator()
        +EstimateCorrespondingPixels(image1: const Mat&, image2: const Mat&, points1: vector<Keypoint>, mask: InputArray) void
        +EstimateCorrespondingPixels(image1: const Mat&, trackedPoints: vector<vector<Keypoint>>+k, mask: InputArray) void
    }
    class ReconstructIO {
        +m_StereoCameraSetup: StereoCalib
        +m_StereoMatcher: StereoBlockMatcherType
        +m_StereoBlockMatcherType: StereoBlockMatcherType
        +ReconstructIO(stereoSetup: const StereoCalib&, config: Config)
        +GenerateSparseMayLeftImage(const Mat&, rightImage: const Mat&, Mat
        +TriangulateIDisparity(const Mat&, cameraImage: const Mat&, mask: InputArray) PointCloud<PointXYZRGB>
        +TriangulateUVI(v: float, v: float, disparity: float, color: const Vec3b&) PointXYZRGB
        +BackProjectPhrnc(float, y: float) PointXYZ
        +TriangulatePoints(disparity: const Mat&, cameraImage: const Mat&, points: vector<Keypoint>+k, triangulatedPoints: vector<PointXYZRGB>+k) void
        +GetCameraParameters(f: float, fx: float, cy: float, camNumber: int) void
        +RectifyImages(leftImage: const Mat&, rightImage: const Mat&, rectLeftImage: const Mat&, rectRightImage: const Mat&) void
        +SetStereoWindowSizes(size: int) void
        +SetStereoBMWinDisparitySum(int) void
        +SetBlockMatcherType(type: StereoBlockMatcherType) void
        +ConfigureStereoBlockMatcherConfig(Config) void
    }
    class ReconstructionSystem {
        +m_Config: Config
        +m_ReusedShadows: atomic<bool>
        +m_Tracker: unique_ptr<Tracker>
        +m_FeatureExtractor: shared_ptr<FrameFeatureExtractor>
        +m_Reconductor: shared_ptr<ReconstructIO>
        +m_MappingSystem: shared_ptr<MappingSystem>
        +m_KeyframeDatabase: shared_ptr<KeyframeDatabase>
        +ReconstructionSystem(config: const Config&, stereoCalib: const StereoCalib&)
        +ReconstructionSystem()
        +RequestShutdown() void
        +ProcessStereoFrame(stereoFrame: const StereoFrame&) void
        +GetCurrentMappingCloud: PointCloud<PointXYZRGB>Ptr) void
        +GetMappingDatabase() shared_ptr<MapDatabase>
    }
    class ReconstructionServer {
        +m_UserRequestedQuit: atomic<bool>
        +m_ProcessingStarted: atomic<bool>
        +m_RunNameProcessed: atomic<long>
        +m_ProcessingThread: thread
        +m_Config: Config
        +m_Visualizer: unique_ptr<Visualizer>
        +m_ReconstructionSystem: unique_ptr<ReconstructionSystem>
        +m_Calib: unique_ptr<StereoCalib>
        +ReconstructionServer()
        +ReconstructionServer(unique_ptr<ReconstructionSystem>)
        +ReconstructionServer()
        +Run() ReconstrServerStatusCode
        +RunProcessingThread()
    }
    class MessageConverter {
        +ConvertCalibMessageToCalibMessage(const CalibMessage&) StereoCalib
        +ConvertStereoMessageToStereoMessage(const StereoMessage&) StereoFrame
    }
    class StereoFrame {
        +ID: long
        +LeftImage: Mat
        +RightImage: Mat
        +Translation: Vector3f
        +Rotation: Mat_3d
    }
    class Config {
        +NumDisparities: int
        +BlockSize: int
        +UniquenessRatio: int
        +SpeckleRange: int
        +SpeckleWindowSize: int
        +PreFilterCap: int
        +MinDisparity: int
    }
    class SBM {
        +NumDisparities: int
        +WindowSize: int
    }
    class SGBM {
        +NumDisparities: int
        +BlockSize: int
        +UniquenessRatio: int
        +SpeckleRange: int
        +SpeckleWindowSize: int
        +PreFilterCap: int
        +MinDisparity: int
    }
    class StereoBlockMatcherType {
        +StereoBlockMatcherType
        +STEREO_BLOCK_MATCHER
        +STEREO_GLOBAL_BLOCK_MATCHER
    }
    class SLAM --> MapDatabase
    class SLAM --> MappingSystem
    class SLAM --> Tracker
    class SLAM --> KeyframeDatabase
    class SLAM --> OpticalFlowEstimator
    class SLAM --> ReconstructIO
    class SLAM --> ReconstructionSystem
    class SLAM --> ReconstructionServer
    class SLAM --> MessageConverter
    class SLAM --> StereoFrame
    class SLAM --> Config
    class SLAM --> SBM
    class SLAM --> SGBM
    class SLAM --> StereoBlockMatcherType
    class SLAM --> Tracker --> TrackingFrame
    class SLAM --> Tracker --> KeyframeDatabase
    class SLAM --> Tracker --> OpticalFlowEstimator
    class SLAM --> Tracker --> ReconstructIO
    class SLAM --> Tracker --> ReconstructionSystem
    class SLAM --> Tracker --> ReconstructionServer
    class SLAM --> Tracker --> MessageConverter
    class SLAM --> Tracker --> StereoFrame
    class SLAM --> Tracker --> Config
    class SLAM --> Tracker --> SBM
    class SLAM --> Tracker --> SGBM
    class SLAM --> Tracker --> StereoBlockMatcherType
    class SLAM --> Tracker --> CameraCompute
    class SLAM --> Tracker --> CameraCalibParser
    class SLAM --> Tracker --> StereoCalib
    class SLAM --> Tracker --> StereoRectification
    class SLAM --> Tracker --> OpticalFlowEstimator
    class SLAM --> Tracker --> ReconstructIO
    class SLAM --> Tracker --> ReconstructionSystem
    class SLAM --> Tracker --> ReconstructionServer
    class SLAM --> Tracker --> MessageConverter
    class SLAM --> Tracker --> StereoFrame
    class SLAM --> Tracker --> Config
    class SLAM --> Tracker --> SBM
    class SLAM --> Tracker --> SGBM
    class SLAM --> Tracker --> StereoBlockMatcherType
    class SLAM --> Tracker --> CameraCompute
    class SLAM --> Tracker --> CameraCalibParser
    class SLAM --> Tracker --> StereoCalib
    class SLAM --> Tracker --> StereoRectification
    class SLAM --> Tracker --> OpticalFlowEstimator
    class SLAM --> Tracker --> ReconstructIO
    class SLAM --> Tracker --> ReconstructionSystem
    class SLAM --> Tracker --> ReconstructionServer
    class SLAM --> Tracker --> MessageConverter
    class SLAM --> Tracker --> StereoFrame
    class SLAM --> Tracker --> Config
    class SLAM --> Tracker --> SBM
    class SLAM --> Tracker --> SGBM
    class SLAM --> Tracker --> StereoBlockMatcherType
    class SLAM --> Tracker --> CameraCompute
    class SLAM --> Tracker --> CameraCalibParser
    class SLAM --> Tracker --> StereoCalib
    class SLAM --> Tracker --> StereoRectification
    class SLAM --> Tracker --> OpticalFlowEstimator
    class SLAM --> Tracker --> ReconstructIO
    class SLAM --> Tracker --> ReconstructionSystem
    class SLAM --> Tracker --> ReconstructionServer
    class SLAM --> Tracker --> MessageConverter
    class SLAM --> Tracker --> StereoFrame
    class SLAM --> Tracker --> Config
    class SLAM --> Tracker --> SBM
    class SLAM --> Tracker --> SGBM
    class SLAM --> Tracker --> StereoBlockMatcherType
    class SLAM --> Tracker --> CameraCompute
    class SLAM --> Tracker --> CameraCalibParser
    class SLAM --> Tracker --> StereoCalib
    class SLAM --> Tracker --> StereoRectification
    class SLAM --> Tracker --> OpticalFlowEstimator
    class SLAM --> Tracker --> ReconstructIO
    class SLAM --> Tracker --> ReconstructionSystem
    class SLAM --> Tracker --> ReconstructionServer
    class SLAM --> Tracker --> MessageConverter
    class SLAM --> Tracker --> StereoFrame
    class SLAM --> Tracker --> Config
    class SLAM --> Tracker --> SBM
    class SLAM --> Tracker --> SGBM
    class SLAM --> Tracker --> StereoBlockMatcherType
    class SLAM --> Tracker --> CameraCompute
    class SLAM --> Tracker --> CameraCalibParser
    class SLAM --> Tracker --> StereoCalib
    class SLAM --> Tracker --> StereoRectification
    class SLAM --> Tracker --> OpticalFlowEstimator
    class SLAM --> Tracker --> Reconstruct
```





