Leap Accelerate

Release 0.13.0

Callan Gray

Feb 20, 2024

LIBRARY DOCUMENTATION

Leap Accelerate is a calibration tool implementing Low-frequency Excision of the Atmosphere in Parallel (LEAP) for low-frequency radio antenna arrays. Leap utilizes GPGPU acceleration for parallel computation across baselines, channels and polarizations and is freely available on GitLab under the GPLv2+ License.

Leap Accelerate consists of:

- *Leap Accelerate API*: a shared library for accelerated direction centering and phase calibration.
- *Leap Accelerate CLI*: a CLI interface for I/O datastream or plasma data access.

CHAPTER

ONE

LEAP ACCELERATE API

A generic library containing core functionality for leap calibration.

1.1 Getting Started

import the leap-accelerate cmake target and add the following include:

```
#include <icrar/leap-accelerate/algorithm/LeapCalibratorFactory.h>
#include <icrar/leap-accelerate/algorithm/ILeapCalibrator.h>
```

create a calibrator object using the factory method and an output callback:

```
ArgumentsValidated args;
std::vector<cpu::Calibration> calibrations;
auto outputCallback = [&](const cpu::Calibration& calibration)
{
    calibrations.push_back(calibration);
};

LeapCalibratorFactory::Create(args.GetComputeImplementation())->Calibrate(
    outputCallback,
    args.GetMeasurementSet(),
    args.GetDirections(),
    args.GetSolutionInterval(),
    args.GetMinimumBaselineThreshold(),
    args.GetReferenceAntenna(),
    args.IsFileSystemCacheEnabled());
```

1.2 Compute Implementations

Compute functionality is split into two independant namespaces/implementations:

- cpu
- cuda

1.2.1 cpu

Provides computation using Eigen3 math libraries and layout that provides trivial data layouts compatible with raw buffer copying to acceleration libraries and compatible with kernel compilers. See Eigen for more details.

1.2.2 cuda

Provides Cuda classes and functions for nvidia hardware using cuda 9.x-11.x and cublas.

1.2.3 opencl

TBA - Provides OpenCL classes and functions for OpenCL supported hardware.

1.3 Modules

- **core**
  - contains common classes and functions required by other modules
- **common**
  - contains leap specific files used by other components
- **model**
  - contains data structures for leap calibration
- **algorithm**
  - contains utility classes and functions for performing leap calibration
- **math**
  - contains generic math extensions
- **ms**
  - contains abstraction layers for measurement set objects
- **cuda**
  - contains cuda specific classes and helpers

CHAPTER

TWO

LEAP ACCELERATE CLI

Leap Accelerate CLI is a command line interface for performing leap calibration on the local system.

2.1 Leap Accelerate CLI

leap-accelerate-cli is a command line interface to performing leap calibration that requires at least a measurement set and a set of directions to produce an antenna array calibration.

2.1.1 Arguments

- `--config <path>` - config file path
- `--filepath <path>` - measurement set file path
- `--directions <array>` - directions for calibration in polar coordinates, e.g. `"[[1.2,0.8],[0.5,0.7]]"`
- `--output <path>` - Calibration output file path
- `--stations` - Overrides number of stations to use in the specified measurement set
- `--solutionInterval <[start,end,interval]>` - Sets the interval to generate solutions using numpy syntax. Additionally supports a single interval integer argument.
- `--referenceAntenna <integer>` - Selects the reference antenna index, default is the last antenna
- `--implementation <type>` - compute implementation type (cpu or cuda)
- `--useFileSystemCache <boolean>` - Whether filesystem caching is used between system calls
- `--autoCorrelations <boolean>` - Set to true if measurement set rows contain autocorrelations
- `--minimumBaselineThreshold <double>` - Minimum antenna baeline length in meters in the range 0.0 -> inf
- `--verbosity <integer>` - Logging verbosity (0=fatal, 1=error, 2=warn, 3=info, 4=debug, 5=trace), defaults to info

Examples:

```
LeapAccelerateCLI --help
LeapAccelerateCLI --config testdata/mwa_test.json
```

2.1.2 Logging

Log files are produced in the current working directory at ./log/leap_YYYY_MM_dd_{number}.log Log files rotate per day and store a maximum of 10MiB.

Logging is controlled by the verbosity setting.

2.1.3 Config File

Config files can be specified via the –config argument to specify runtime arguments as an alternative to command line arguments.

Config files currently must be written in coformant JSON format.

Schema

```
{
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "Arguments",
    "definitions": {},
    "type": "object",
    "properties": {
        "filePath": { "type": "string" },
        "outputFilePath": { "type": "string" },
        "autoCorrelations": { "type": "boolean" },
        "useFileSystemCache": { "type": "boolean" },
        "minimumBaselineTheshold": { "type": "integer" },
        "solutionInterval": {
            "type": ["integer", "array"],
            "items": { "type": ["number", "null"] },
            "minItems": 3,
            "maxItems": 3
        },
        "referenceAntenna": { "type": "integer" },
        "computeImplementation": {
            "type": "string",
            "enum": ["cpu", "cuda"]
        },
        "verbosity": { "type": "string" },
        "directions": {
            "type": "array",
            "items": {
                "type": "array",
                "items": { "type": "number" }
            }
        }
```

(continues on next page)

(continued from previous page)

```
    },
    "required": [ "filePath", "directions" ]
}
```

Note: Properties are not required when specified in as CLI arguments with a config file.

Config File Example

```
{
    "filePath": "/testdata/ska/SKA_LOW_SIM_short_EoR0_ionosphere_off_GLEAM.ms",
    "outputFilePath": "ska_low_cal",
    "computeImplementation": "cpu",
    "directions": [
        [-0.4606549305661674,-0.29719233792392513],
        [-0.4606549305661674,-0.29719233792392513]
    ],
    "solutionInterval": [0, None, 1],
    "verbosity": "info"
}
```

CHAPTER

THREE

LEAP ACCELERATE API REFERENCE

3.1 Class Hierarchy

3.2 File Hierarchy

3.3 Full API

3.3.1 Namespaces

Namespace boost

Namespace Eigen

Contents

Namespaces

Functions

Typedefs

- *Typedef Eigen::MatrixXb*
- *Typedef Eigen::VectorXb*

Namespace Eigen::internal

Contents

Functions

- *Specialized Template Function Eigen::internal::cast*

Namespace icrar

Contents

Detailed Description

ICRAR - International Centre for Radio Astronomy Research (c) UWA - The University of Western Australia Copyright by UWA(in the framework of the ICRAR) All rights reserved

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

ICRAR - International Centre for Radio Astronomy Research (c) UWA - The University of Western Australia Copyright by UWA(in the framework of the ICRAR) All rights reserved

Namespaces

Classes

- *Class Slice*

Enums

- *Enum ComputeImplementation*
- *Enum InputType*
- *Enum StreamOutType*

Functions

- *Template Function icrar::AttributeEquals*
- *Function icrar::ComputeImplementationToString*
- *Template Function icrar::ConvertMatrix(const Eigen::Matrix<T, R, C>&)*
- *Template Function icrar::ConvertMatrix(const Eigen::Matrix<T, Eigen::Dynamic, Eigen::Dynamic>&)*
- *Template Function icrar::ConvertVector*
- *Function icrar::exists*
- *Template Function icrar::fixed*
- *Function icrar::GetAllTimestepsMWACalibration*
- *Function icrar::GetAvailableCudaPhysicalMemory*
- *Function icrar::GetEachTimestepMWACalibration*
- *Function icrar::GetFirstTimestepMWACalibration*
- *Function icrar::GetTotalAvailableSystemVirtualMemory*
- *Function icrar::GetTotalCudaPhysicalMemory*
- *Function icrar::GetTotalSystemVirtualMemory*
- *Function icrar::GetTotalUsedSystemVirtualMemory*
- *Function icrar::git_has_local_changes*
- *Function icrar::git_sha1*
- *Template Function icrar::isApprox(const std::vector<T>&, const std::vector<T>&, T)*
- *Template Function icrar::isApprox(const std::complex<T>&, const std::complex<T>&, T)*
- *Template Function icrar::isApprox(const Tensor3X<T>&, const Tensor3X<T>&, double)*
- *Function icrar::IsImmediateMode*
- *Template Function icrar::matrix_hash*
- *Function icrar::memory_amount*
- *Function icrar::ParseComputeImplementation*
- *Function icrar::ParseDirections(const std::string&)*
- *Function icrar::ParseDirections(const rapidjson::Value&)*
- *Function icrar::ParseInputType*

- *Function icrar::ParseSlice(const std::string&)*
- *Function icrar::ParseSlice(const rapidjson::Value&)*
- *Function icrar::ParseStreamOutType*
- *Template Function icrar::pretty_matrix*
- *Template Function icrar::pretty_row*
- *Template Function icrar::ProcessCache(size_t, const In&, Out&, const std::string&, const std::string&, Lambda)*
- *Template Function icrar::ProcessCache(size_t, const In&, const std::string&, const std::string&, Lambda)*
- *Template Function icrar::ProcessCache(const In&, const std::string&, Lambda, Out&)*
- *Template Function icrar::ProcessCache(const In&, const std::string&, Lambda)*
- *Template Function icrar::range(IntType, IntType, IntType)*
- *Template Function icrar::range(IntType, IntType)*
- *Template Function icrar::range(IntType)*
- *Template Function icrar::read_binary(const char *, Matrix&)*
- *Template Function icrar::read_binary(std::ifstream&, Matrix&)*
- *Template Function icrar::read_hash(const char *, T&)*
- *Template Function icrar::read_hash(std::ifstream&, T&)*
- *Function icrar::RunCalibration*
- *Template Function icrar::to_underlying_type*
- *Function icrar::ToCasaDirection*
- *Function icrar::ToCasaDirectionVector*
- *Function icrar::ToCasaUVW*
- *Function icrar::ToCasaUVWVector(const std::vector<icrar::MVuvw>&)*
- *Function icrar::ToCasaUVWVector(const Eigen::MatrixX3d&)*
- *Function icrar::ToDirection*
- *Function icrar::ToDirectionVector*
- *Template Function icrar::ToFixedMatrix*
- *Function icrar::ToMatrix(const std::vector<MVuvw>&)*
- *Template Function icrar::ToMatrix(const casacore::Matrix<T>&)*
- *Function icrar::ToUVW*
- *Function icrar::ToUVWVector(const std::vector<casacore::MVuvw>&)*
- *Function icrar::ToUVWVector(const Eigen::MatrixXd&)*
- *Template Function icrar::ToVector(casacore::Vector<T>)*
- *Template Function icrar::ToVector(const std::vector<T>&)*
- *Template Function icrar::trace_matrix*
- *Function icrar::TryParseComputeImplementation*

- *Function icrar::TryParseInputType*
- *Function icrar::TryParseStreamOutType*
- *Function icrar::us_time*
- *Template Function icrar::vector_map*
- *Function icrar::version*
- *Template Function icrar::write_binary(const char *, const Matrix&)*
- *Template Function icrar::write_binary(std::ofstream&, const Matrix&)*
- *Template Function icrar::write_hash(std::ofstream&, T)*
- *Template Function icrar::write_hash(const char *, T)*

Typedefs

- *Typedef icrar::MVuvw*
- *Typedef icrar::Rangei*
- *Typedef icrar::Rangel*
- *Typedef icrar::SphericalDirection*
- *Typedef icrar::Tensor3X*

Namespace icrar::constants

Contents

- *Variables*

Variables

- *Variable icrar::constants::speed_of_light*

Namespace icrar::cpu

Contents

- *Classes*
- *Functions*

Classes

Functions

Namespace icrar::cuda

cuda

Contents

Classes

- *Class HostIntegration*
- *Class HostLeapData*

Enums

- *Enum JobType*
- *Enum MatrixOp*

Functions

- *Function icrar::cuda::AvgDataToPhaseAngles*
- *Function icrar::cuda::CalcDeltaPhase*
- *Function icrar::cuda::Empty*
- *Function icrar::cuda::mat_mul(cublasHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const float *, const float *, float *)*
- *Function icrar::cuda::mat_mul(cublasHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const int *, const int *, int *)*
- *Function icrar::cuda::mat_mul(cublasLtHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const double *, const double *, double *)*
- *Function icrar::cuda::mat_mul(cublasLtHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const float *, const float *, float *)*
- *Function icrar::cuda::mat_mul(cublasHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const double *, const double *, double *)*
- *Function icrar::cuda::mat_mul(cublasLtHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const int *, const int *, int *)*
- *Function icrar::cuda::mat_mul_add(cublasLtHandle_t, const size_t, const size_t, const size_t, const double *, const double *, const double *, double *)*
- *Function icrar::cuda::mat_mul_add(cublasLtHandle_t, const size_t, const size_t, const size_t, const float *, const float *, const float *, float *)*
- *Function icrar::cuda::mat_mul_add(cublasHandle_t, const size_t, const size_t, const size_t, const double *, const double *, double *)*
- *Function icrar::cuda::mat_mul_add(cublasLtHandle_t, const size_t, const size_t, const size_t, const int *, const int *, const int *, int *)*
- *Function icrar::cuda::mat_mul_add(cublasHandle_t, const size_t, const size_t, const size_t, const float *, const float *, float *)*
- *Function icrar::cuda::mat_mul_add(cublasHandle_t, const size_t, const size_t, const size_t, const int *, const int *, int *)*
- *Template Function icrar::cuda::multiply(cublasHandle_t, const device_matrix<T>&, const device_vector<T>&, device_vector<T>&, MatrixOp, MatrixOp)*
- *Template Function icrar::cuda::multiply(cublasHandle_t, const device_matrix<T>&, const device_matrix<T>&, device_matrix<T>&, MatrixOp, MatrixOp)*

- *Template Function icrar::cuda::multiply(cublasLtHandle_t, const device_matrix<T>&, const device_vector<T>&, device_vector<T>&, MatrixOp, MatrixOp)*
- *Template Function icrar::cuda::multiply(cublasLtHandle_t, const device_matrix<T>&, const device_matrix<T>&, device_matrix<T>&, MatrixOp, MatrixOp)*
- *Template Function icrar::cuda::multiply_add(cublasHandle_t, const device_matrix<T>&, const device_matrix<T>&, device_matrix<T>&)*
- *Template Function icrar::cuda::multiply_add(cublasHandle_t, const device_matrix<T>&, const device_vector<T>&, device_vector<T>&)*
- *Template Function icrar::cuda::multiply_add(cublasLtHandle_t, const device_matrix<T>&, const device_vector<T>&, const device_vector<T>&, device_vector<T>&)*
- *Template Function icrar::cuda::multiply_add(cublasLtHandle_t, const device_matrix<T>&, const device_matrix<T>&, const device_matrix<T>&, device_matrix<T>&)*
- *Function icrar::cuda::PhaseRotateAverageVisibilities*
- *Function icrar::cuda::pseudo_inverse(cusolverDnHandle_t, cublasHandle_t, const Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic>&, const JobType)*
- *Function icrar::cuda::pseudo_inverse(cusolverDnHandle_t, cublasHandle_t, const device_matrix<double>&, const JobType)*
- *Function icrar::cuda::SliceDeltaPhase*
- *Function icrar::cuda::svd*
- *Function icrar::cuda::ToCublasOp*

Typedefs

- *Typedef icrar::cuda::device_tensor3*
- *Typedef icrar::cuda::device_tensor4*

Namespace icrar::detail

Contents

Classes

- *Template Struct _fixed*
- *Struct _memory_amount*
- *Struct _microseconds_amount*

Functions

- *Template Function icrar::detail::operator<<(std::basic_ostream<T>&, const detail::_microseconds_amount&)*
- *Template Function icrar::detail::operator<<(std::basic_ostream<T>&, detail::_fixed<N, VT>)*
- *Template Function icrar::detail::operator<<(std::basic_ostream<T>&, const detail::_memory_amount&)*

Namespace icrar::log

Contents

Enums

- *Enum Verbosity*

Functions

- *Function icrar::log::Initialize*
- *Function icrar::log::ParseVerbosity*
- *Function icrar::log::TryParseVerbosity*

Variables

- *Variable icrar::log::DEFAULT_VERBOSITY*
- *Variable icrar::log::logging_level*

Namespace icrar::profiling

Contents

Classes

- *Struct ResourceUsage*
- *Class timer*
- *Class UsageReporter*

Functions

- *Function icrar::profiling::get_resource_usage*
- *Template Function icrar::profiling::operator<<(std::basic_ostream<CharT>&, const ResourceUsage&)*
- *Template Function icrar::profiling::operator<<(std::basic_ostream<CharT, Traits>&, const timer&)*

Typedefs

- *Typedef icrar::profiling::usec_t*

Namespace icrar::python

Contents

Classes

- *Class PyBeamCalibration*
- *Class PyCalibration*
- *Class PyLeapCalibrator*
- *Class PyMeasurementSet*

Namespace std

Namespace thrust

Contents

Detailed Description

Typedefs

- *Typedef thrust::complex*

3.3.2 Classes and Structs

Struct ArgumentsDTO

- Defined in file_icrar_leap-accelerate_common_config_Arguments.h

Struct Documentation

struct **ArgumentsDTO**

Typed arguments of `CLIArgumentsDTO`.

Public Functions

ArgumentsDTO() = default

ArgumentsDTO(*CLIArgumentsDTO* &&args)

Public Members

boost::optional<*InputType*> **inputType**

MeasurementSet source type.

boost::optional<std::string> **filePath**

MeasurementSet filepath.

boost::optional<std::string> **configFilePath**

Optional config filepath.

boost::optional<*StreamOutType*> **streamOutType**

boost::optional<std::string> **outputFilePath**

Calibration output file, print to terminal if empty.

boost::optional<int> **stations**

boost::optional<unsigned int> **referenceAntenna**

boost::optional<std::vector<*SphericalDirection*>> **directions**

boost::optional<*ComputeImplementation*> **computeImplementation**

boost::optional<*Slice*> **solutionInterval**

boost::optional<double> **minimumBaselineThreshold**

boost::optional<bool> **readAutocorrelations**

boost::optional<bool> **mwaSupport**

boost::optional<bool> **computeCal1**

boost::optional<icrar::log::*Verbosity*> **verbosity**

boost::optional<bool> **useFileSystemCache**

Whether to update a file cache for fast inverse matrix loading.

boost::optional<bool> **useIntermediateBuffer**

Whether to allocate intermediate buffers for reduced cpu->gpu copies.

boost::optional<bool> **useCusolver**

Whether to use cusolverDn for matrix inversion.

Struct CLIArgumentsDTO

- Defined in file_icrar_leap-accelerate_common_config_Arguments.h

Struct Documentation

struct **CLIArgumentsDTO**

Raw arguments received via the command line interface using boost::program_options. Only raw types std::string, bool, int, uint, float and double are allowed here.

Public Members

boost::optional<std::string> **inputType**

boost::optional<std::string> **filePath**

boost::optional<std::string> **configFilePath**

boost::optional<std::string> **streamOutType**

boost::optional<std::string> **outputFilePath**

boost::optional<int> **stations**

boost::optional<unsigned int> **referenceAntenna**

boost::optional<std::string> **directions**

boost::optional<std::string> **computeImplementation**

boost::optional<std::string> **solutionInterval**

boost::optional<double> **minimumBaselineThreshold**

boost::optional<bool> **mwaSupport**

boost::optional<bool> **computeCal1**

boost::optional<bool> **readAutocorrelations**

boost::optional<int> **verbosity**

boost::optional<bool> **useFileSystemCache**

boost::optional<bool> **useIntermediateBuffer**

boost::optional<bool> **useCusolver**

Public Static Functions

static *CLIArgumentsDTO* **GetDefaultArguments**()

Struct ComputeOptionsDTO

- Defined in file_icrar_leap-accelerate_algorithm_ComputeOptionsDTO.h

Struct Documentation

struct **ComputeOptionsDTO**

Options received from I/O that optimizes computation performance based on input data and hardware configuration. Can either be overriden by the user or intelligently determined at runtime if not set.

Public Functions

inline bool **IsInitialized**() const

Public Members

boost::optional<bool> **isFileSystemCacheEnabled**

Enables caching of expensive calculations to the filesystem.

boost::optional<bool> **useIntermediateBuffer**

boost::optional<bool> **useCusolver**

Struct Constants

- Defined in file_icrar_leap-accelerate_model_cpu_LeapData.h

Struct Documentation

struct **Constants**

Container of variables that do not change throughout calibration.

Public Functions

__host__ __device__ inline double **GetChannelWavelength**(int i) const

bool **operator==**(const *Constants* &rhs) const

Public Members

uint32_t **nbaselines**

uint32_t **referenceAntenna**

uint32_t **channels**

uint32_t **num_pols**

uint32_t **stations**

uint32_t **timesteps**

uint32_t **rows**

double **freq_start_hz**

double **freq_inc_hz**

double **phase_centre_ra_rad**

double **phase_centre_dec_rad**

double **dlm_ra**

double **dlm_dec**

Template Struct _fixed

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Struct Documentation

template<int **N**, typename **T**>

struct **_fixed**

decimal places helper

Template Parameters

- **N** – number of decimal places
- **T** – input type

Public Members

T **_val**

Struct _memory_amount

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Struct Documentation

struct **_memory_amount**

Public Members

std::size_t **_val**

Struct _microseconds_amount

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Struct Documentation

struct **_microseconds_amount**

Public Members

std::chrono::microseconds::rep **_val**

Struct ResourceUsage

- Defined in file_icrar_leap-accelerate_core_profiling_resource_usage.h

Struct Documentation

struct **ResourceUsage**

A collection of resource-related statistics.

Public Members

usec_t **utime**

Time spent in user mode, in microseconds.

usec_t **stime**

Time spent in kernel mode, in microseconds.

usec_t **wtime**

Total walltime spent since program started.

std::size_t **peak_rss**

Maximum amount of memory used, in bytes.

Struct UVWData

- Defined in file_icrar_leap-accelerate_model_PlasmaTM.h

Struct Documentation

struct **UVWData**

Public Members

std::array<double, 3> **uu**

std::array<double, 3> **vv**

std::array<double, 3> **ww**

double **interval**

double **exposure**

Struct visibility

- Defined in file_icrar_leap-accelerate_model_visibility.h

Struct Documentation

struct **visibility**

Public Functions

inline std::ostream &**operator<<**(std::ostream &os, const *visibility* &vis)

Public Members

double **frequency**

double **time**

double **u**

double **v**

double **w**

double **r**

double **i**

double **weight**

int **a1**

int **a2**

int **gcfinx**

Class async_function

- Defined in file_icrar_leap-accelerate_python_async.h

Inheritance Relationships

Base Type

- `public cpp_function`

Class Documentation

class **async_function** : public cpp_function

Public Functions

async_function() = default

inline **async_function**(std::nullptr_t)

template<typename **Return**, typename ...**Args**, typename ...**Extra**>
inline **async_function**(*Return* (*f)(*Args*...), const *Extra*&... extra)

Construct a async_function from a vanilla function pointer.

template<typename **Func**, typename ...**Extra**, typename =
detail::enable_if_t<detail::is_lambda<*Func*>::value>>
inline **async_function**(*Func* &&f, const *Extra*&... extra)

Construct a async_function from a lambda function (possibly with internal state)

template<typename **Return**, typename **Class**, typename ...**Arg**, typename ...**Extra**>
inline **async_function**(*Return* (*Class*::* f)(*Arg*...), const *Extra*&... extra)

Construct a async_function from a class method (non-const, no ref-qualifier)

template<typename **Return**, typename **Class**, typename ...**Arg**, typename ...**Extra**>
inline **async_function**(*Return* (*Class*::* f)(*Arg*...) &, const *Extra*&... extra)

Construct a async_function from a class method (non-const, lvalue ref-qualifier) A copy of the overload for non-const functions without explicit ref-qualifier but with an added &.

template<typename **Return**, typename **Class**, typename ...**Arg**, typename ...**Extra**>
inline **async_function**(*Return* (*Class*::* f)(*Arg*...) const, const *Extra*&... extra)

Construct a async_function from a class method (const, no ref-qualifier)

template<typename **Return**, typename **Class**, typename ...**Arg**, typename ...**Extra**>

inline **async_function**(*Return* (*Class*::* f)(*Arg*...) const &, const *Extra*&... extra)

Construct a async_function from a class method (const, lvalue ref-qualifier) A copy of the overload for const functions without explicit ref-qualifier but with an added &.

Protected Functions

template<typename **Func**, typename **Return**, typename ...**Args**, typename ...**Extra**>
inline void **initialize**(*Func* &&f, *Return* (*)(*Args*...), const *Extra*&... extra)

template<typename **Func**, typename ...**Args**, typename ...**Extra**>
inline void **initialize**(*Func* &&f, void (*)(*Args*...), const *Extra*&... extra)

Template Class class_async

- Defined in file_icrar_leap-accelerate_python_async.h

Inheritance Relationships

Base Type

- `public class_< type_, options... >`

Class Documentation

template<typename **type_**, typename ...**options**>

class **class_async** : public class_<*type_*, *options*...>

Public Functions

template<typename **Func**, typename ...**Extra**>
inline *class_async* &**def_async**(const char *name_, *Func* &&f, const *Extra*&... extra)

Class CppAwaitable

- Defined in file_icrar_leap-accelerate_python_async.h

Class Documentation

class **CppAwaitable**

Public Functions

inline **CppAwaitable**()

inline **CppAwaitable**(std::future<pybind11::object> &_future)

inline *CppAwaitable* ***iter**()

inline *CppAwaitable* ***await**()

inline void **next**()

Class Arguments

- Defined in file_icrar_leap-accelerate_common_config_Arguments.h

Class Documentation

class **Arguments**

Validated set of command line arguments required to perform leap calibration

Public Functions

Arguments(*ArgumentsDTO* &&cliArgs)

void **OverrideArguments**(*ArgumentsDTO* &&args)

Overrides the stored set of arguments.

Parameters
args –

void **Validate**() const

boost::optional<std::string> **GetOutputFilePath**() const

std::unique_ptr<std::ostream> **CreateOutputStream**(double startEpoch = 0.0) const

Using the outputFilePath member, creates an output stream object.

Parameters
startEpoch –

Returns
std::unique_ptr<std::ostream>

StreamOutType **GetStreamOutType**() const

Gets the configuration for output stream type.

Returns
StreamOutType

const *MeasurementSet* &**GetMeasurementSet**() const

Gets the user specifiied measurement set.

Returns
MeasurementSet&

const std::vector<*SphericalDirection*> &**GetDirections**() const

ComputeImplementation **GetComputeImplementation**() const

Slice **GetSolutionInterval**() const

boost::optional<unsigned int> **GetReferenceAntenna**() const

double **GetMinimumBaselineThreshold**() const

Gets the minimum baseline threshold in meteres. Baselines of length beneath the threshold are to be filtered/flagged.

Returns
double baseline threshold length in meters

bool **ComputeCal1**() const

Whether cal1 should be computed.

Returns
true

Returns
false

ComputeOptionsDTO **GetComputeOptions**() const

Gets configured options related to compute performance.

Returns
ComputeOptionsDTO

icrar::log::*Verbosity* **GetVerbosity**() const

Gets the configured logging verbosity.

Returns
icrar::log::Verbosity

Class BeamCalibration

- Defined in file_icrar_leap-accelerate_model_cpu_calibration_BeamCalibration.h

Class Documentation

class **BeamCalibration**

Contains the results of leap calibration for a single direction.

Public Functions

BeamCalibration(*SphericalDirection* direction, Eigen::MatrixXd calibration)

Construct a new Direction Calibration object.

Parameters

- **direction** – direciton of calibration
- **calibration** – calibration of each antenna for the given direction

BeamCalibration(const std::pair<*SphericalDirection*, Eigen::MatrixXd> &beamCalibration)

bool **IsApprox**(const *BeamCalibration* &beamCalibration, double threshold)

const *SphericalDirection* &**GetDirection**() const

Gets the calibration direction.

Returns
const SphericalDirection

const Eigen::VectorXd &**GetAntennaPhases**() const

Get the phase calibration vector for the antenna array in the specified direction.

Returns
const Eigen::VectorXd

void **Serialize**(std::ostream &os, bool pretty = false) const

Serializes to JSON format.

Parameters
os – JSON output stream

template<typename **Writer**>
inline void **Write**(*Writer* &writer) const

Public Static Functions

static *BeamCalibration* **Parse**(const rapidjson::Value &doc)

Class Calibration

- Defined in file_icrar_leap-accelerate_model_cpu_calibration_Calibration.h

Class Documentation

class **Calibration**

Contains a single calibration solution.

Public Functions

Calibration(double startEpoch, double endEpoch)

Creates an empty calibration.

Parameters

- **startEpoch** –
- **endEpoch** –

Calibration(double startEpoch, double endEpoch, std::vector<cpu::*BeamCalibration*> &&beamCalibrations)

double **GetStartEpoch**() const

double **GetEndEpoch**() const

bool **IsApprox**(const *Calibration* &calibration, double tolerence)

const std::vector<*BeamCalibration*> &**GetBeamCalibrations**() const

std::vector<*BeamCalibration*> &**GetBeamCalibrations**()

void **Serialize**(std::ostream &os, bool pretty = false) const

template<typename **Writer**>
inline void **Write**(*Writer* &writer) const

Public Static Functions

static *Calibration* **Parse**(std::istream &is)

static *Calibration* **Parse**(const std::string &json)

static *Calibration* **Parse**(const rapidjson::Value &doc)

Class CalibrationCollection

- Defined in file_icrar_leap-accelerate_model_cpu_calibration_CalibrationCollection.h

Class Documentation

class **CalibrationCollection**

Contains a collection of calibrations.

Public Functions

inline **CalibrationCollection**(std::vector<cpu::*Calibration*> &&calibrations)

inline const std::vector<cpu::*Calibration*> &**GetCalibrations**() const

inline void **Serialize**(std::ostream &os, bool pretty = false) const

template<typename **Writer**>
inline void **Write**(*Writer* &writer) const

Class CpuLeapCalibrator

- Defined in file_icrar_leap-accelerate_algorithm_cpu_CpuLeapCalibrator.h

Inheritance Relationships

Base Type

- `public icrar::ILeapCalibrator` (*Class ILeapCalibrator*)

Class Documentation

class **CpuLeapCalibrator** : public icrar::*ILeapCalibrator*

Leap Calibration implementation using.

Public Functions

void **Calibrate**(std::function<void(const cpu::*Calibration*&)> outputCallback, const icrar::*MeasurementSet* &ms, const std::vector<*SphericalDirection*> &directions, const *Slice* &solutionInterval, double minimumBaselineThreshold, bool computeCal1, boost::optional<unsigned int> referenceAntenna, const *ComputeOptionsDTO* &computeOptions) override

Interface for Leap calibration implementations.

Calibrates by performing phase rotation for each direction in `directions` by splitting uvws into integration batches per timestep.

Public Static Functions

static void **BeamCalibrate**(*LeapData* &leapData, const *SphericalDirection* &direction, bool computeCal1, std::vector<*Integration*> &integrations, std::vector<*BeamCalibration*> &output_calibrations)

Performs rotation, summing and calibration for `direction`.

Parameters

- **leapData** – leapData object containing data required for calibration
- **direction** – the direction to calibrate for
- **input** – batches of uvws and visibilities to process
- **output_calibrations** – output calibration from summing a function of uvws and visibilities

static void **PhaseRotateAverageVisibilities**(*Integration* &integration, *LeapData* &leapData)

Performs phase rotation and averaging over each baseline, channel and polarization.

Parameters

- **integration** – The input integration batch of uvws and visibilities
- **leapData** – The leapData object where AverageData is written to

static void **ApplyInversion**(cpu::*LeapData* &leapData, const *SphericalDirection* &direction, bool computeCal1, std::vector<cpu::*BeamCalibration*> &output_calibrations)

Applies inversion to input averaged visibilities to generate beam calibrations.

Parameters

- **leapData** –
- **output_calibrations** –

Class Integration

- Defined in file_icrar_leap-accelerate_model_cpu_Integration.h

Inheritance Relationships

Derived Type

- public icrar::cuda::HostIntegration (*Class HostIntegration*)

Class Documentation

class **Integration**

A container for storing a visibilities tensor for accumulation during phase rotating.

Subclassed by *icrar::cuda::HostIntegration*

Public Functions

Integration(int integrationNumber, Eigen::Tensor<double, 3> &&uvws,
Eigen::Tensor<std::complex<double>, 4> &&visibilities)

bool **operator==**(const *Integration* &rhs) const

inline int **GetIntegrationNumber**() const

inline size_t **GetNumPolarizations**() const

inline size_t **GetNumChannels**() const

inline size_t **GetNumBaselines**() const

inline size_t **GetNumTimesteps**() const

inline const Eigen::Tensor<double, 3> &**GetUVW**() const

Gets the UVW object of shape (3, baselines, timesteps)

Returns
const std::vector<icrar::MVuvw>& uvws

inline const Eigen::Tensor<std::complex<double>, 4> &**GetVis**() const

Get the Visibilities object of shape (polarizations, channels, baselines, timesteps)

Returns
Eigen::Tensor<std::complex<double>, 4>& visibilities

inline Eigen::Tensor<std::complex<double>, 4> &**GetVis**()

Get the Visibilities object of shape (polarizations, channels, baselines, timesteps)

Returns
Eigen::Tensor<std::complex<double>, 4>& visibilities

Public Static Functions

static *Integration* **CreateFromMS**(const icrar::*MeasurementSet* &ms, int integrationNumber, const *Slice* &timestepSlice, const *Slice* &polarizationSlice = *Slice*(0, boost::none, 1))

Protected Attributes

int **m_integrationNumber**

Eigen::Tensor<double, 3> **m_UVW**

Eigen::Tensor<std::complex<double>, 4> **m_visibilities**

Friends

friend class icrar::cuda::DeviceIntegration

Class LeapData

- Defined in file_icrar_leap-accelerate_model_cpu_LeapData.h

Inheritance Relationships

Derived Type

- `public icrar::cuda::HostLeapData` (*Class HostLeapData*)

Class Documentation

class **LeapData**

container of phaserotation constants and variables for calibrating a single beam. Can be mutated to calibrate for multiple directions.

Subclassed by *icrar::cuda::HostLeapData*

Public Functions

LeapData(const icrar::*MeasurementSet* &ms, boost::optional<unsigned int> refAnt = boost::none, double minimumBaselineThreshold = 0.0, bool computeInverse = true, bool useCache = true)

Construct a new LeapData object. SetDirection() must be called after construction.

Parameters

- **ms** – measurement set to read observations from
- **refAnt** – the reference antenna index, default is the last index

- **minimumBaselineThreshold** – baseline lengths less that the minimum in meters are flagged
- **computeInverse** – whether to compute inverse using cpu inversion
- **useCache** – whether to load Ad matrix from cache

LeapData(const icrar::*MeasurementSet* &ms, const *SphericalDirection* &direction, boost::optional<unsigned int> refAnt = boost::none, double minimumBaselineThreshold = 0.0, bool computeInverse = true, bool useCache = true)

Construct a new LeapData object.

Parameters

- **ms** – measurement set to read observations from
- **direction** – the direction of the beam to calibrate for
- **refAnt** – the reference antenna index, default is the last index
- **minimumBaselineThreshold** – baseline lengths less that the minimum in meters are flagged
- **useCache** – whether to load Ad matrix from cache

const *Constants* &**GetConstants**() const

const Eigen::MatrixXd &**GetA**() const

Matrix of baseline pairs of shape [baselines, stations].

const Eigen::VectorXi &**GetI**() const

Vector of indexes of the stations that are not flagged in A of shape [baselines].

const Eigen::MatrixXd &**GetAd**() const

The pseudoinverse of A with shape [stations, baselines].

inline virtual void **SetAd**(Eigen::MatrixXd &&Ad)

const Eigen::MatrixXd &**GetA1**() const

Matrix of baselines using the reference antenna of shape [stations+1, stations] where the last row represents the reference antenna.

const Eigen::VectorXi &**GetI1**() const

Vector of indexes of the stations that are not flagged in A1 of shape [stations].

Returns

const Eigen::VectorXi&

const Eigen::MatrixXd &**GetAd1**() const

inline virtual void **SetAd1**(Eigen::MatrixXd &&ad1)

inline const *SphericalDirection* &**GetDirection**() const

inline const Eigen::Matrix3d &**GetDD**() const

void **SetDirection**(const *SphericalDirection* &direction)

void **ComputeInverse**()

Computes the A and A1 inverse matrices.

void **ValidateInverse**() const

Output logs on the validity of inverse matrices.

Eigen::Matrix3d **GenerateDDMatrix**(const *SphericalDirection* &direction) const

Utility method to generate a direction matrix using the configured zenith direction.

Parameters
direction –

Returns
Eigen::Matrix3d

inline const Eigen::VectorXcd &**GetAvgData**() const

inline Eigen::VectorXcd &**GetAvgData**()

bool **operator==**(const *LeapData* &rhs) const

inline bool **operator!=**(const *LeapData* &rhs) const

Protected Attributes

Constants **m_constants**

double **m_minimumBaselineThreshold**

bool **m_useCache**

Eigen::MatrixXd **m_A**

Eigen::VectorXi **m_I**

Eigen::MatrixXd **m_A1**

Eigen::VectorXi **m_I1**

Eigen::MatrixXd **m_Ad**

The pseudo-inverse of m_A, late intitialized.

Eigen::MatrixXd **m_Ad1**

std::vector<icrar::*MVuvw*> **m_UVW**

SphericalDirection **m_direction**

Eigen::Matrix3d **m_dd**

Eigen::VectorXcd **m_avgData**

Friends

friend class icrar::cuda::DeviceLeapData

friend class icrar::cuda::ConstantBuffer

Class CpuComputeOptions

- Defined in file_icrar_leap-accelerate_algorithm_cpu_CpuComputeOptions.h

Class Documentation

class **CpuComputeOptions**

Public Functions

inline **CpuComputeOptions**(const *ComputeOptionsDTO* &dto, const icrar::*MeasurementSet* &ms)

Determines ideal calibration compute options for a given MeasurementSet.

Parameters

- **computeOptions** –
- **ms** –

inline bool **IsFileSystemCacheEnabled**() const

Class ComputeDevice

- Defined in file_icrar_leap-accelerate_cuda_compute_device.h

Class Documentation

class **ComputeDevice**

Class ConstantBuffer

- Defined in file_icrar_leap-accelerate_model_cuda_DeviceLeapData.h

Class Documentation

class **ConstantBuffer**

Container class of uniform gpu buffers available to all cuda threads that are const/immutable per calibration.

Public Functions

ConstantBuffer(const icrar::cpu::*Constants* &constants, *device_matrix*<double> &&A, *device_vector*<int> &&I, *device_matrix*<double> &&Ad, *device_matrix*<double> &&A1, *device_vector*<int> &&I1, *device_matrix*<double> &&Ad1)

Construct a new Constant Buffer object.

Parameters

- **constants** –
- **A** –
- **I** –
- **Ad** –
- **A1** –
- **I1** –
- **Ad1** –

inline const icrar::cpu::*Constants* &**GetConstants**() const

inline const *device_matrix*<double> &**GetA**() const

inline const *device_vector*<int> &**GetI**() const

inline const *device_matrix*<double> &**GetAd**() const

inline const *device_matrix*<double> &**GetA1**() const

inline const *device_vector*<int> &**GetI1**() const

inline const *device_matrix*<double> &**GetAd1**() const

void **ToHost**(icrar::cpu::*LeapData* &host) const

void **ToHostAsync**(icrar::cpu::*LeapData* &host) const

Class CudaLeapCalibrator

- Defined in file_icrar_leap-accelerate_algorithm_cuda_CudaLeapCalibrator.h

Inheritance Relationships

Base Type

- `public icrar::ILeapCalibrator` (*Class ILeapCalibrator*)

Class Documentation

class **CudaLeapCalibrator** : public icrar::*ILeapCalibrator*

LEAP calibration object implemented using CUDA.

Public Functions

CudaLeapCalibrator()

~CudaLeapCalibrator() override

virtual void **Calibrate**(std::function<void(const cpu::*Calibration*&)> outputCallback, const icrar::*MeasurementSet* &ms, const std::vector<*SphericalDirection*> &directions, const *Slice* &solutionInterval, double minimumBaselineThreshold, bool computeCal1, boost::optional<unsigned int> referenceAntenna, const *ComputeOptionsDTO* &computeOptions) override

Interface for Leap calibration implementations.

Calibrates by performing phase rotation for each direction in `directions` by splitting uvws and visibilities into integration batches per timestep.

void **CalculateAd**(*HostLeapData* &leapData, *device_matrix*<double> &deviceA, *device_matrix*<double> &deviceAd, bool isFileSystemCacheEnabled, bool useCuda)

Calculates Ad into deviceAd, writes to cache if `isFileSystemCacheEnabled` is true.

Parameters

- **hostA** – matrix to invert
- **deviceA** – output device memory of A
- **hostAd** – output host memory of Ad (optionally written to)
- **deviceAd** – output device memory of Ad
- **isFileSystemCacheEnabled** – whether to use file caching
- **useCuda** – whether to use cuda solvers

void **CalculateAd1**(*HostLeapData* &leapData, *device_matrix*<double> &deviceA1, *device_matrix*<double> &deviceAd1)

Calculates Ad1 into deviceAd1.

Parameters

- **hostA1** – matrix to invert
- **deviceA1** – output device memory of A1
- **hostAd1** – output host memory of Ad1 (optionally written to)
- **deviceAd1** – output device memory of Ad1

void **BeamCalibrate**(const *HostLeapData* &hostMetadata, *DeviceLeapData* &deviceLeapData, const *SphericalDirection* &direction, cuda::*DeviceIntegration* &input, bool computeCal1, std::vector<cpu::*BeamCalibration*> &output_calibrations)

Performs only visibilities rotation on the GPU

Template Class device_matrix

- Defined in file_icrar_leap-accelerate_cuda_device_matrix.h

Inheritance Relationships

Base Type

- `private boost::noncopyable`

Class Documentation

template<typename **T**>

class **device_matrix** : private boost::noncopyable

A cuda device buffer object that represents a global memory buffer on a cuda device. Matrix size is fixed at construction and can only be resized using move semantics.

Note: See https://www.quantstart.com/articles/Matrix-Matrix-Multiplication-on-the-GPU-with-Nvidia-CUDA/

Note: See https://forums.developer.nvidia.com/t/guide-cudamalloc3d-and-cudaarrays/23421

Template Parameters
T – numeric type

Public Functions

inline **device_matrix**()

Default constructor.

inline **device_matrix**(*device_matrix* &&other) noexcept

Move Constructor.

Parameters
other –

inline *device_matrix* &**operator=**(*device_matrix* &&other) noexcept

Move Assignment Operator.

Parameters
other –

Returns
device_matrix&

inline **device_matrix**(size_t rows, size_t cols, const *T* *data = nullptr)

Construct a new device matrix object of fixed size and initialized asyncronously if data is provided.

Parameters

- **rows** – number of rows
- **cols** – number of columns
- **data** – constigous column major data of size rows*cols*sizeof(T) to copy to device

inline explicit **device_matrix**(const Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic> &data)

template<int **Rows**, int **Cols**>
inline explicit **device_matrix**(const Eigen::Matrix<*T*, *Rows*, *Cols*> &data)

inline **~device_matrix**()

__host__ inline *T* ***Get**()

__host__ inline const *T* ***Get**() const

__host__ inline size_t **GetRows**() const

__host__ inline size_t **GetCols**() const

__host__ inline size_t **GetCount**() const

__host__ inline size_t **GetSize**() const

__host__ inline void **SetZeroAsync**()

__host__ inline void **SetDataSync**(const *T* *data)

Performs a synchronous copy of data into the device buffer.

Parameters
data –

Returns
host

__host__ inline void **SetDataAsync**(const *T* *data)

Copies data from device to host memory.

Parameters
data –

Returns
host

__host__ inline void **ToHost**(*T* *out) const

__host__ inline void **ToHost**(std::vector<*T*> &out) const

__host__ inline void **ToHost**(Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic> &out) const

template<int **Rows**, int **Cols**>
__host__ inline void **ToHost**(Eigen::Matrix<*T*, *Rows*, *Cols*> &out) const

__host__ inline Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic> **ToHost**() const

__host__ inline void **ToHostAsync**(*T* *out) const

__host__ inline void **ToHostAsync**(Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic> &out) const

__host__ inline void **ToHostVectorAsync**(Eigen::Matrix<*T*, Eigen::Dynamic, 1> &out) const

__host__ inline Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic> **ToHostAsync**() const

Template Class device_tensor

- Defined in file_icrar_leap-accelerate_cuda_device_tensor.h

Class Documentation

template<typename **T**, uint32_t **NumDims**>

class **device_tensor**

A cuda device tensor buffer object that references a tensor in cuda device memory buffer for manipulation by the host.

Note: See https://www.quantstart.com/articles/Matrix-Matrix-Multiplication-on-the-GPU-with-Nvidia-CUDA/

Note: See https://forums.developer.nvidia.com/t/guide-cudamalloc3d-and-cudaarrays/23421

Template Parameters

- **T** – the tensor data type
- **NumDims** – number of tensor dimensions

Public Functions

inline **device_tensor**(Eigen::DSizes<Eigen::DenseIndex, *NumDims*> shape, const *T* *data = nullptr)

inline **device_tensor**(size_t sizeDim0, size_t sizeDim1, size_t sizeDim2, const *T* *data = nullptr)

inline **device_tensor**(size_t sizeDim0, size_t sizeDim1, size_t sizeDim2, size_t sizeDim3, const *T* *data = nullptr)

inline **device_tensor**(const Eigen::Tensor<*T*, *NumDims*> &tensor)

inline **device_tensor**(*device_tensor* &&other)

Copy Constructor.

Parameters
other –

inline *device_tensor* &**operator=**(*device_tensor* &&other) noexcept

inline **~device_tensor**()

__host__ inline *T* ***Get**()

Gets the raw pointer to device buffer memory.

Returns
T*

__host__ inline const *T* ***Get**() const

Gets the raw pointer to device buffer memory.

Returns
T const*

__host__ inline Eigen::DenseIndex **GetDimensionSize**(int dim) const

__host__ inline Eigen::DSizes<Eigen::DenseIndex, *NumDims*> **GetDimensions**() const

__host__ inline size_t **GetCount**() const

Gets the total number of elements in the tensor.

Returns
host

__host__ inline size_t **GetSize**() const

Gets the total number of elements in the tensor.

Returns
host

__host__ inline size_t **GetByteSize**() const

Gets the total number of bytes in the memory buffer.

Returns
host

__host__ inline void **SetDataSync**(const *T* *data)

Performs a synchronous copy of data into the device buffer.

Parameters
data –

Returns
host

__host__ inline void **SetDataAsync**(const *T* *data)

Set the Data asyncronously from host memory.

Parameters
data –

Returns
host

__host__ inline void **SetDataAsync**(const *device_tensor*<*T*, *NumDims*> &data)

__host__ inline void **ToHost**(*T* *out) const

__host__ inline void **ToHost**(std::vector<*T*> &out) const

__host__ inline void **ToHost**(Eigen::Tensor<*T*, *NumDims*> &out) const

__host__ inline void **ToHostAsync**(*T* *out) const

Template Class device_vector

- Defined in file_icrar_leap-accelerate_cuda_device_vector.h

Inheritance Relationships

Base Type

- `private boost::noncopyable`

Class Documentation

template<typename **T**>

class **device_vector** : private boost::noncopyable

A cuda device buffer object that own a vector memory buffer on a cuda device. Vector size is fixed at construction.

Note: See https://www.quantstart.com/articles/Matrix-Matrix-Multiplication-on-the-GPU-with-Nvidia-CUDA/

Note: See https://forums.developer.nvidia.com/t/guide-cudamalloc3d-and-cudaarrays/23421

Template Parameters
T – numeric type

Public Functions

inline **device_vector**()

Default constructor.

__host__ inline **device_vector**(*device_vector* &&other) noexcept

__host__ inline *device_vector* &**operator=**(*device_vector* &&other) noexcept

__host__ inline explicit **device_vector**(size_t count, const *T* *data = nullptr)

Construct a new device buffer object.

Parameters

- **size** –
- **data** –

__host__ inline explicit **device_vector**(const std::vector<*T*> &data)

__host__ inline explicit **device_vector**(const Eigen::Matrix<*T*, Eigen::Dynamic, 1> &data)

__host__ inline explicit **device_vector**(const Eigen::Matrix<*T*, 1, Eigen::Dynamic> &data)

__host__ inline **~device_vector**()

__host__ __device__ inline *T* ***Get**()

__host__ __device__ inline const *T* ***Get**() const

__host__ __device__ inline size_t **GetCount**() const

Gets the number of elements in the buffer.

__host__ __device__ inline size_t **GetRows**() const

Gets the number of rows in the column vector.

__host__ __device__ inline constexpr size_t **GetCols**() const

__host__ __device__ inline size_t **GetSize**() const

Gets the buffer size in bytes.

__host__ inline void **SetZeroAsync**()

__host__ inline void **SetDataSync**(const *T* *data)

Performs a synchronous copy of data into the device buffer.

Parameters

data – data buffer for host to device copying

Pre

data points to a buffer of byte size >= GetSize()

__host__ inline void **SetDataAsync**(const *T* *data)

Sets buffer data from pinned host memory.

Parameters

data – data buffer for host to device copying

Pre

Heap memory must be pinned using cudaHostRegister(..., cudaHostRegisterPortable)

Pre

data points to a buffer of byte size >= GetSize()

__host__ inline void **ToHost**(*T* *out) const

__host__ inline void **ToHost**(std::vector<*T*> &out) const

__host__ inline void **ToHost**(Eigen::Matrix<*T*, Eigen::Dynamic, 1> &out) const

__host__ inline void **ToHostAsync**(*T* *out) const

Sets buffer data from pinned host memory.

Parameters

data – data buffer for device to host copying

Pre

Heap memory must be pinned using cudaHostRegister(..., cudaHostRegisterPortable)

Pre

data points to a buffer of byte size >= GetSize()

__host__ inline void **ToHostAsync**(std::vector<*T*> &out) const

__host__ inline void **ToHostAsync**(Eigen::Matrix<*T*, Eigen::Dynamic, 1> &out) const

__host__ inline Eigen::Matrix<*T*, Eigen::Dynamic, 1> **ToHostAsync**() const

Class DeviceIntegration

- Defined in file_icrar_leap-accelerate_model_cuda_DeviceIntegration.h

Class Documentation

class **DeviceIntegration**

A Cuda memory buffer instance of visibility data for integration.

Public Functions

DeviceIntegration(int integrationNumber, Eigen::DSizes<Eigen::DenseIndex, 3> uvwShape, Eigen::DSizes<Eigen::DenseIndex, 4> visShape)

Construct a new Device Integration object where visibilities is a zero tensor of @shape.

Parameters
shape –

DeviceIntegration(const icrar::cpu::*Integration* &integration)

Construct a new Device Integration object with a data syncronous copy.

Parameters
integration –

__host__ void **Set**(const icrar::cpu::*Integration* &integration)

Set the Data object.

Parameters
integration –

__host__ void **Set**(const icrar::cuda::*DeviceIntegration* &integration)

Set the Data object.

Parameters
integration –

inline int **GetIntegrationNumber**() const

inline size_t **GetNumPolarizations**() const

inline size_t **GetNumChannels**() const

inline size_t **GetNumBaselines**() const

inline size_t **GetNumTimesteps**() const

inline const *device_tensor3*<double> &**GetUVW**() const

inline const *device_tensor4*<std::complex<double>> &**GetVis**() const

inline *device_tensor4*<std::complex<double>> &**GetVis**()

__host__ void **ToHost**(cpu::*Integration* &host) const

Copies device data to a host object.

Parameters
host – object with data on cpu memory

Class DeviceLeapData

- Defined in file_icrar_leap-accelerate_model_cuda_DeviceLeapData.h

Class Documentation

class **DeviceLeapData**

Represents the complete collection of LeapData that resides on the GPU for leap-calibration

Public Functions

DeviceLeapData(*DeviceLeapData* &&other) noexcept = default

DeviceLeapData &**operator=**(*DeviceLeapData* &&other) noexcept = default

explicit **DeviceLeapData**(const icrar::cpu::*LeapData* &leapData)

Construct a new Device LeapData object from the equivalent object on CPU memory. This copies to all device buffers.

Parameters
leapData –

DeviceLeapData(std::shared_ptr<*ConstantBuffer*> constantBuffer, std::shared_ptr<*DirectionBuffer*> directionBuffer)

Construct a new Device LeapData object from the equivalent object on CPU memory. This copies to all device buffers.

Parameters

- **constantBuffer** –
- **directionBuffer** –

const icrar::cpu::*Constants* &**GetConstants**() const

inline const *SphericalDirection* &**GetDirection**() const

inline const Eigen::Matrix3d &**GetDD**() const

inline const *ConstantBuffer* &**GetConstantBuffer**() const

inline const *device_matrix*<std::complex<double>> &**GetAvgData**() const

inline *device_matrix*<std::complex<double>> &**GetAvgData**()

void **SetAvgData**(int v)

void **ToHost**(icrar::cpu::*LeapData* &host) const

icrar::cpu::*LeapData* **ToHost**() const

void **ToHostAsync**(icrar::cpu::*LeapData* &host) const

Class DirectionBuffer

- Defined in file_icrar_leap-accelerate_model_cuda_DeviceLeapData.h

Class Documentation

class **DirectionBuffer**

LeapData Variables allocated per direction.

Public Functions

DirectionBuffer(*SphericalDirection* direction, Eigen::Matrix3d dd, const Eigen::MatrixXcd &avgData)

Constructs a new Direction Buffer object initializing all memory.

Parameters

- **direction** –
- **dd** –
- **avgData** –

DirectionBuffer(int avgDataRows, int avgDataCols)

Constructs a new Direction Buffer object for late initialization.

Parameters

- **avgDataRows** –
- **avgDataCols** –

inline const *SphericalDirection* &**GetDirection**() const

inline const Eigen::Matrix3d &**GetDD**() const

inline *device_matrix*<std::complex<double>> &**GetAvgData**()

void **SetDirection**(const *SphericalDirection* &direction)

void **SetDD**(const Eigen::Matrix3d &dd)

Class HostIntegration

- Defined in file_icrar_leap-accelerate_model_cuda_HostIntegration.h

Inheritance Relationships

Base Type

- `public icrar::cpu::Integration` (*Class Integration*)

Class Documentation

class **HostIntegration** : public icrar::cpu::*Integration*

A cuda decorator for cpu::Integration. This class stores data on the host using pinned memory to allow for asyncronous read and write with cuda.

Public Functions

inline **HostIntegration**(int integrationNumber, Eigen::Tensor<double, 3> &&uvws, Eigen::Tensor<std::complex<double>, 4> &&visibilities)

inline **~HostIntegration**()

Public Static Functions

static inline *HostIntegration* **CreateFromMS**(const icrar::*MeasurementSet* &ms, int integrationNumber, const *Slice* &timestepSlice, const *Slice* &polarizationSlice = *Slice*(0, boost::none, 1))

Class HostLeapData

- Defined in file_icrar_leap-accelerate_model_cuda_HostLeapData.h

Inheritance Relationships

Base Type

- `public icrar::cpu::LeapData` (*Class LeapData*)

Class Documentation

class **HostLeapData** : public icrar::cpu::*LeapData*

A cuda decorator for cpu::Integration. This class stores data on the host withs pinned memory calls to allow for asyncronous read and write with cuda.

Public Functions

inline **HostLeapData**(const icrar::*MeasurementSet* &ms, boost::optional<unsigned int> refAnt, double minimumBaselineThreshold, bool computeInverse, bool useCache)

inline **~HostLeapData**()

inline virtual void **SetAd**(Eigen::MatrixXd &&Ad) override

inline virtual void **SetAd1**(Eigen::MatrixXd &&Ad1) override

Class CudaComputeOptions

- Defined in file_icrar_leap-accelerate_algorithm_cuda_CudaComputeOptions.h

Class Documentation

class **CudaComputeOptions**

Validates and determines the best compute features for calibration depending on measurement set data and hardware configuration.

Public Functions

CudaComputeOptions(const *ComputeOptionsDTO* &computeOptions, const icrar::*MeasurementSet* &ms, const *Rangei* &solutionRange)

Determines ideal calibration compute options for a given measurementSet.

Parameters

- **computeOptions** –
- **ms** –

Public Members

bool **isFileSystemCacheEnabled**

Enables caching of expensive calculations to the filesystem.

bool **useIntermediateBuffer**

enables an intermediate buffer containing unrotated visibilities to improve per direction performance

bool **useCusolver**

Uses cusolver for Ad calculation.

Class exception

- Defined in file_icrar_leap-accelerate_exception_exception.h

Inheritance Relationships

Base Type

- `public std::exception`

Derived Types

- public icrar::file_exception (*Class file_exception*)
- public icrar::invalid_argument_exception (*Class invalid_argument_exception*)
- public icrar::json_exception (*Class json_exception*)
- public icrar::not_implemented_exception (*Class not_implemented_exception*)

Class Documentation

class **exception** : public std::exception

Generic exception with source tracing.

Subclassed by *icrar::file_exception*, *icrar::invalid_argument_exception*, *icrar::json_exception*, *icrar::not_implemented_exception*

Public Functions

exception(const std::string &msg, const std::string &file, int line)

Constructs a new exception object.

Parameters

- **msg** – exception reason
- **file** – exception file location
- **line** – exception line location

const char ***what**() const noexcept override

Class file_exception

- Defined in file_icrar_leap-accelerate_exception_exception.h

Inheritance Relationships

Base Type

- public icrar::exception (*Class exception*)

Class Documentation

class **file_exception** : public icrar::*exception*

Exception raised when a file system operation fails.

Public Functions

inline **file_exception**(const std::string &msg, const std::string &filename, const std::string &file, int line)

Class ILeapCalibrator

- Defined in file_icrar_leap-accelerate_algorithm_ILeapCalibrator.h

Inheritance Relationships

Base Type

- `private boost::noncopyable`

Derived Types

- `public icrar::cpu::CpuLeapCalibrator` (*Class CpuLeapCalibrator*)
- `public icrar::cuda::CudaLeapCalibrator` (*Class CudaLeapCalibrator*)

Class Documentation

class **ILeapCalibrator** : private boost::noncopyable

Interface for Leap calibration implementations.

Subclassed by *icrar::cpu::CpuLeapCalibrator*, *icrar::cuda::CudaLeapCalibrator*

Public Functions

virtual **~ILeapCalibrator**() = default

virtual void **Calibrate**(std::function<void(const cpu::*Calibration*&)> outputCallback, const icrar::*MeasurementSet* &ms, const std::vector<*SphericalDirection*> &directions, const *Slice* &solutionInterval, double minimumBaselineThreshold, bool computeCal1, boost::optional<unsigned int> referenceAntenna, const *ComputeOptionsDTO* &computeOptions) = 0

Performs Leap calibration for single or multiple solutions.

Parameters

- **outputCallback** – callback for each solution interval calibration result
- **ms** – the mesurement set containing all input measurements

- **directions** – the directions to calibrate for
- **minimumBaselineThreshold** – the minimum baseline length to use in calibrations
- **solutionInterval** – the arbitrary interval to calculate solutions for
- **referenceAntenna** – the reference antenna of metrix A1
- **computeOptions** – collection of compute implementation options

Returns
CalibrationCollection the calibrationn result

Class invalid_argument_exception

- Defined in file_icrar_leap-accelerate_exception_exception.h

Inheritance Relationships

Base Type

- `public icrar::exception` (*Class exception*)

Class Documentation

class **invalid_argument_exception** : public icrar::*exception*

Exception raised when an invalid argument is passed into a function.

Public Functions

inline **invalid_argument_exception**(const std::string &msg, const std::string &arg, const std::string &file, int line)

Class json_exception

- Defined in file_icrar_leap-accelerate_exception_exception.h

Inheritance Relationships

Base Type

- `public icrar::exception` (*Class exception*)

Class Documentation

class **json_exception** : public icrar::*exception*

Exception raised when parsing invalid json.

Public Functions

inline **json_exception**(const std::string &msg, const std::string &file, int line)

Class LeapCalibratorFactory

- Defined in file_icrar_leap-accelerate_algorithm_LeapCalibratorFactory.h

Class Documentation

class **LeapCalibratorFactory**

Factory class for creating a specialized LEAP calibrator.

Public Static Functions

static std::unique_ptr<*ILeapCalibrator*> **Create**(*ComputeImplementation* impl)

Creates a calibrator object that performs leap calibration with specified implementation.

Parameters
impl –

Returns
std::unique_ptr<ILeapCalibrator>

Class MeasurementSet

- Defined in file_icrar_leap-accelerate_ms_MeasurementSet.h

Class Documentation

class **MeasurementSet**

Provides an abstraction layer around a casacore MeasurementSet that provides all data required for leap calibration. This class additionally stores runtime user specificed variables and cached variabled calculated from the underlying measurement set.

Public Functions

MeasurementSet(const std::string &filepath)

inline boost::optional<std::string> **GetFilepath**() const

inline bool **IsAutoCorrelationsEnabled**() const

inline const casacore::MeasurementSet ***GetMS**() const

Gets a non-null pointer to a casacore::MeasurementSet.

Returns
const casacore::MeasurementSet*

inline const casacore::MSMainColumns ***GetMSMainColumns**() const

Gets a non-null pointer to a casacore::MSMainColumns.

Returns
const casacore::MSMainColumns*

inline const casacore::MSColumns ***GetMSColumns**() const

Gets a non-null pointer to a casacore::MSColumns.

Returns
const casacore::MSColumns*

uint32_t **GetTotalAntennas**() const

Gets the total number of antennas including flagged antennas.

uint32_t **GetNumStations**() const

Gets the number of stations excluding flagged stations.

Returns
uint32_t

uint32_t **GetNumBaselines**() const

Get the number of baselines in the measurement set using the current autocorrelations setting and including stations not recording rows.

Note: TODO: baselines should always be n*(n-1) / 2 and without autocorrelations

---

Returns
uint32_t

uint32_t **GetNumPols**() const

Get the number of polarizations in the measurement set.

Returns
uint32_t

uint32_t **GetNumChannels**() const

Gets the number of channels in the measurement set.

Returns
uint32_t

uint32_t **GetNumRows**() const

Gets the number of rows in the measurement set (non-flagged baselines * timesteps).

Returns
uint32_t

Eigen::VectorXi **GetAntenna1**() const

Gets the indexes of the first antenna in baselines.

Returns
Eigen::VectorXi

Eigen::VectorXi **GetAntenna2**() const

Gets the indexes of the second antenna in baselines.

Returns
Eigen::VectorXi

uint32_t **GetNumTimesteps**() const

Gets the total number of timesteps in the measurement set.

Returns
uint32_t

double **GetTimeInterval**() const

Gets the time interval of visibilities (assuming the same for all rows)

std::vector<double> **GetEpochs**() const

Get the Epochs object.

Returns
std::vector<double>

Eigen::*VectorXb* **GetFlaggedBaselines**() const

Gets a vector of size nBaselines with a true value at the index of unflagged baselines. Flag is is logical and of channels and polarizations.

Returns
Eigen::VectorXb

uint32_t **GetNumFlaggedBaselines**() const

Get the number of baselines that are flagged by the measurement set.

Returns
uint32_t

Eigen::*VectorXb* **GetShortBaselines**(double minimumBaselineThreshold = 0.0) const

Gets a flag vector of short baselines.

Parameters
minimumBaselineThreshold – baseline threshold

Returns
Eigen::VectorXb

uint32_t **GetNumShortBaselines**(double minimumBaselineThreshold = 0.0) const

Get the number of baselines that below the `minimumBaselineThreshold`.

Parameters
minimumBaselineThreshold – baseline threshold

Returns
uint32_t

Eigen::*VectorXb* **GetFilteredBaselines**(double minimumBaselineThreshold = 0.0) const

Gets flag vector of filtered baselines that are either flagged or short.

Parameters
minimumBaselineThreshold – baseline threshold

Returns
Eigen::VectorXb

uint32_t **GetNumFilteredBaselines**(double minimumBaselineThreshold = 0.0) const

Gets the number of baselines that are flagged baselines or short baselines.

Parameters
minimumBaselineThreshold – baseline threshold

Returns
uint32_t

Eigen::Tensor<double, 3> **ReadCoords**() const

Reads UVW coordinates from the measurement set.

Returns
Eigen::Tensor<double, 3>

Eigen::Tensor<double, 3> **ReadCoords**(const *Slice* &timestepSlice) const

Gets the Coords/UVWs of a specified time interval.

Parameters

- **startTimestep** –
- **intervalTimesteps** –

Returns
Eigen::Tensor<double, 3> of dimensions (3, baselines, timesteps)

Eigen::Tensor<double, 3> **ReadCoords**(uint32_t startTimestep, uint32_t intervalTimesteps) const

Gets the Coords/UVWs of a specified time interval.

Parameters

- **startTimestep** –
- **intervalTimesteps** –

Returns
Eigen::Tensor<double, 3> of dimensions (3, baselines, timesteps)

Eigen::Tensor<std::complex<double>, 4> **ReadVis**() const

Gets the visibilities from all baselines, channels and polarizations for the first timestep.

Returns
Eigen::Tensor<std::complex<double>, 4> of dimensions (polarizations, channels, baselines, timesteps)

Eigen::Tensor<std::complex<double>, 4> **ReadVis**(const *Slice* &timestepSlice, const *Slice* &polarizationSlice = *Slice*(0, boost::none, 1)) const

Gets visibilities from the specificed dimension slices of a specified timestep slice.

Parameters

- **startTimestep** – start timestep index
- **intervalTimesteps** – number of timesteps

Returns
Eigen::Tensor<std::complex<double>, 4> of dimensions (polarizations, channels, baselines, timesteps)

Eigen::Tensor<std::complex<double>, 4> **ReadVis**(std::uint32_t startTimestep, std::uint32_t intervalTimesteps, const *Slice* &polarizationSlice = *Slice*(0, boost::none, 1)) const

Gets visibilities from the specificed dimension slices of a specified timestep slice.

Parameters

- **startTimestep** – start timestep index
- **intervalTimesteps** – number of timesteps

Returns
Eigen::Tensor<std::complex<double>, 4> of dimensions (polarizations, channels, baselines, timesteps)

Eigen::Tensor<std::complex<double>, 4> **ReadVis**(uint32_t startTimestep, uint32_t intervalTimesteps, *Range*<int32_t> polarizationRange, const char *column) const

Reads from file visibilities using specified dimension slices.

Parameters

- **startTimestep** –
- **intervalTimesteps** –
- **polarizationRange** –
- **column** –

Returns
Eigen::Tensor<std::complex<double>, 4> of dimensions (polarizations, channels, baselines, timesteps)

std::set<int32_t> **GetMissingAntennas**() const

Gets the antennas that are not present in any baselines.

Returns
std::set<int32_t>

std::set<int32_t> **GetFlaggedAntennas**() const

Gets the antenna indexes that are either not present in any baselines or are flagged in all of it's baselines.

Indexes are out of the total antennas

Returns
std::set<int32_t>

Class not_implemented_exception

- Defined in file_icrar_leap-accelerate_exception_exception.h

Inheritance Relationships

Base Type

- `public icrar::exception` (*Class exception*)

Class Documentation

class **not_implemented_exception** : public icrar::*exception*

Public Functions

not_implemented_exception(const std::string &file, int line)

Class PlasmaTM

- Defined in file_icrar_leap-accelerate_model_PlasmaTM.h

Class Documentation

class **PlasmaTM**

Public Functions

inline **PlasmaTM**(boost::python::object &obj)

int **GetNumChannels**() const

double **GetFreqStartHz**() const

double **GetFreqIncHz**() const

int **GetNumStations**() const

int **GetNumBaselines**() const

bool **IsAutocorrelated**() const

int **GetNumPols**() const

std::pair<int, int> **GetPhaseCentreRadecRad**() const

void ***GetNearestData**(double time) const

std::pair<int, *UVWData*> **GetMatchingData**(double current_mjd_utc) const

Class timer

- Defined in file_icrar_leap-accelerate_core_profiling_timer.h

Class Documentation

class **timer**

Public Types

using **clock** = std::chrono::high_resolution_clock

using **duration** = typename *clock*::duration

Public Functions

inline *duration* **get**() const

Class UsageReporter

- Defined in file_icrar_leap-accelerate_core_profiling_UsageReporter.h

Class Documentation

class **UsageReporter**

Reports high-level, process-wide resource usage values on destruction.

Public Functions

UsageReporter() = default

UsageReporter(*UsageReporter*&&) = default

UsageReporter(const *UsageReporter*&) = default

UsageReporter &**operator=**(const *UsageReporter*&) = default

UsageReporter &**operator=**(*UsageReporter*&&) = default

~UsageReporter() noexcept

Class PyBeamCalibration

- Defined in file_icrar_leap-accelerate_python_PyLeapCalibration.h

Class Documentation

class **PyBeamCalibration**

Public Functions

PyBeamCalibration()

Eigen::Vector2d **GetDirection**()

Eigen::VectorXi **GetAntennaPhases**()

Class PyCalibration

- Defined in file_icrar_leap-accelerate_python_PyLeapCalibration.h

Class Documentation

class **PyCalibration**

An boost::python adapter/wrapper for Calibrations. Method overloading and types without python bindings are not allowed in signatures.

Public Functions

PyCalibration()

double **GetStartEpoch**()

double **GetEndEpoch**()

std::vector<*PyBeamCalibration*> **GetBeamCalibrations**()

Class PyLeapCalibrator

- Defined in file_icrar_leap-accelerate_python_PyLeapCalibrator.h

Class Documentation

class **PyLeapCalibrator**

An adapter to a boost::python compatible class.

Note: the Calibrate signature may change as needed

Public Functions

PyLeapCalibrator(*ComputeImplementation* impl, icrar::log::*Verbosity* verbosity = icrar::log::*Verbosity*::*warn*)

PyLeapCalibrator(std::string impl, int verbosity = 40)

PyLeapCalibrator(const *PyLeapCalibrator* &other)

void **Calibrate**(const *MeasurementSet* &msPath, const Eigen::Ref<const Eigen::Matrix<double, Eigen::Dynamic, 2, Eigen::RowMajor>> &directions, const *Slice* &solutionInterval, const double minimumBaselineThreshold, const std::function<void(const cpu::*Calibration*&)> &callback)

void **CalibrateToFile**(const *MeasurementSet* &msPath, const Eigen::Ref<const Eigen::Matrix<double, Eigen::Dynamic, 2, Eigen::RowMajor>> &directions, const *Slice* &solutionInterval, const double minimumBaselineThreshold, boost::optional<std::string> outputPath)

void **PythonCalibrate**(const std::string &msPath, const Eigen::Ref<const Eigen::Matrix<double, Eigen::Dynamic, 2, Eigen::RowMajor>> &directions, const pybind11::slice &solutionInterval, const double minimumBaselineThreshold, const std::function<void(const cpu::*Calibration*&)> callback)

A python interop compatible signature for leap calibration.

void **PythonCalibrateToFile**(const std::string &msPath, const Eigen::Ref<const Eigen::Matrix<double, Eigen::Dynamic, 2, Eigen::RowMajor>> &directions, const pybind11::slice &solutionInterval, const double minimumBaselineThreshold, const pybind11::object &outputPath)

A python interop compatible signature for leap calibration.

void **PythonPlasmaCalibrate**(const pybind11::object &plasmaTM, const Eigen::Ref<const Eigen::Matrix<double, Eigen::Dynamic, 2, Eigen::RowMajor>> &directions, const pybind11::object &outputPath)

A python interop campatible signature for calibrate.

Class PyMeasurementSet

- Defined in file_icrar_leap-accelerate_python_PyMeasurementSet.h

Class Documentation

class **PyMeasurementSet**

An boost::python adapter/wrapper for MeasurementSets. Method overloading and types without python bindings are not allowed in signatures.

Public Functions

PyMeasurementSet(std::string msPath)

inline const *MeasurementSet* &**Get**() const

Eigen::Tensor<double, 3> **ReadCoords**(std::uint32_t startTimestep, std::uint32_t intervalTimesteps)

Eigen::Tensor<std::complex<double>, 4> **ReadVis**(std::uint32_t startTimestep, std::uint32_t intervalTimesteps)

Template Class Range

- Defined in file_icrar_leap-accelerate_common_Range.h

Class Documentation

template<typename **T**>

class **Range**

Represents a forwards linear sequence of indexes for some finite collection. (Indexes are always positive and can be converted to Eigen ArithmeticSequence)

Public Functions

inline **Range**(*T* start, *T* end, *T* interval)

inline *T* **GetStart**() const

inline *T* **GetEnd**() const

inline *T* **GetInterval**() const

inline *T* **GetSize**() const

Gets the number of elements in the range.

Returns
int

inline Eigen::ArithmeticSequence<Eigen::Index, Eigen::Index, Eigen::Index> **ToSeq**()

Class Slice

- Defined in file_icrar_leap-accelerate_common_Slice.h

Class Documentation

class **Slice**

Represents a forwards linear sequence of indexes for some arbitrary collection. Python equivalent is the slice operator [start:end:interval]. Eigen equivalent is Eigen::seq(start, end, interval). Matlab equivalent is slice operator (start:interval:end)

Note: No support for reverse order, e.g. (end:0:-1)

Public Functions

Slice() = default

Slice(boost::optional<int64_t> interval)

Slice(boost::optional<int64_t> start, boost::optional<int64_t> end)

Slice(boost::optional<int64_t> start, boost::optional<int64_t> end, boost::optional<int64_t> interval)

inline boost::optional<int64_t> **GetStart**() const

Gets the starting index of an arbitrary collection slice. -1 represents the end of the collection. none represents the element after the end of the collection.

inline boost::optional<int64_t> **GetEnd**() const

Gets the end exclusive index of an arbitrary collection slice. -1 represents the end of the collection. none represents the element after the end of the collection.

inline boost::optional<int64_t> **GetInterval**() const

Gets the interval betweeen indices of an arbitrary collection slice. none represents the length of the collection.

template<typename **T**>
inline *Range*<*T*> **Evaluate**(*T* collectionSize) const

Evaluates a slice from an arbirary index range to a positive integer index range for use with C++ collections.

Template Parameters
T – integer type

Parameters
collectionSize – size of the collection that to evaluate against

Returns
Range<T>

Public Static Functions

static inline *Slice* **First**()

static inline *Slice* **Last**()

static inline *Slice* **Each**()

static inline *Slice* **All**()

3.3.3 Enums

Enum ComputeImplementation

- Defined in file_icrar_leap-accelerate_core_compute_implementation.h

Enum Documentation

enum class `icrar::`**ComputeImplementation**

Specifier for the compute implementation of a LeapCalibrator.

Values:

enumerator **cpu**

enumerator **cuda**

Enum JobType

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_invert.h

Enum Documentation

enum class `icrar::cuda::`**JobType** : signed char

Corresponds to job types of CusolverDn API (e.g. cusolverDnDgesvd)

Values:

enumerator **A**

All - Entire dense matrix is used.

enumerator **S**

Slim/Thin - Minimal matrix dimensions.

Enum MatrixOp

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_op.h

Enum Documentation

enum class `icrar::cuda::MatrixOp`

Values:

enumerator **normal**

enumerator **transpose**

enumerator **hermitian**

enumerator **conjugate**

Enum InputType

- Defined in file_icrar_leap-accelerate_core_InputType.h

Enum Documentation

enum class `icrar::InputType`

Values:

enumerator **file**

Read from a casacore table file.

enumerator **stream**

Read from a spead2 stream (unsupported)

Enum Verbosity

- Defined in file_icrar_leap-accelerate_core_log_Verbosity.h

Enum Documentation

enum class `icrar::log::`**`Verbosity`**

Selects the reporting level filter for log messages.

Values:

enumerator **`fatal`**

Unexpected execution path, report issue to code maintainers.

enumerator **`error`**

Known execution error, address exception message before reporting.

enumerator **`warn`**

Automatically resolved user exception.

enumerator **`info`**

Regular execution reporting.

enumerator **`debug`**

Debug mode reporting.

enumerator **`trace`**

Developer targeted reporting.

Enum StreamOutType

- Defined in file_icrar_leap-accelerate_core_stream_out_type.h

Enum Documentation

enum class `icrar::`**`StreamOutType`**

A configurable enumaration type that can be used for specifying how calibrations are streamed to the output during computation.

Values:

enumerator **`collection`**

Calibrations are written to a collection in a single file.

enumerator **`singleFile`**

Calibrations are continously rewritten to a single file as computed.

enumerator **`multipleFiles`**

Calibrations are continously written to multiple files as computed.

3.3.4 Functions

Function assert_near_cd

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_cd**(const std::complex<double> &expected, const std::complex<double> &actual, double tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

ICRAR - International Centre for Radio Astronomy Research (c) UWA - The University of Western Australia Copyright by UWA(in the framework of the ICRAR) All rights reserved

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Function assert_near_matrix3_d

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_matrix3_d**(const Eigen::Matrix3d &expected, const Eigen::Matrix3d &actual, double tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

Function assert_near_matrix_cd

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_matrix_cd**(const Eigen::MatrixXcd &expected, const Eigen::MatrixXcd &actual, double tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

Function assert_near_matrix_d

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_matrix_d**(const Eigen::MatrixXd &expected, const Eigen::MatrixXd &actual, double tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

Function assert_near_matrix_i

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_matrix_i**(const Eigen::MatrixXi &expected, const Eigen::MatrixXi &actual, int tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

Function assert_near_metadata

- Defined in file_icrar_leap-accelerate_tests_test_helper.h

Function Documentation

void **assert_near_metadata**(const icrar::cpu::*LeapData* &expected, const icrar::cpu::*LeapData* &actual, const std::string &ln, const std::string &rn, const std::string &file, int line)

Template Function assert_near_tensor

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

template<typename **T**>
void **assert_near_tensor**(const Eigen::Tensor<*T*, 3> &expected, const Eigen::Tensor<*T*, 3> &actual, *T* tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

Function assert_near_vector_b

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_vector_b**(const Eigen::*VectorXb* &expected, const Eigen::*VectorXb* &actual, const std::string &ln, const std::string &rn, const std::string &file, int line)

Function assert_near_vector_d(const Eigen::VectorXd&, const Eigen::VectorXd&, double, const std::string&, const std::string&, const std::string&, int)

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_vector_d**(const Eigen::VectorXd &expected, const Eigen::VectorXd &actual, double tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

Function assert_near_vector_d(const std::vector<double>&, const std::vector<double>&, double, const std::string&, const std::string&, const std::string&, int)

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_vector_d**(const std::vector<double> &expected, const std::vector<double> &actual, double tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

Function assert_near_vector_i

- Defined in file_icrar_leap-accelerate_tests_math_eigen_helper.h

Function Documentation

void **assert_near_vector_i**(const Eigen::VectorXi &expected, const Eigen::VectorXi &actual, int tolerance, const std::string &ln, const std::string &rn, const std::string &file, int line)

Template Function DimensionsVector

- Defined in file_icrar_leap-accelerate_python_pybind_eigen.h

Function Documentation

template<typename **Scalar**, int **Dims**>
std::vector<Eigen::Index> **DimensionsVector**(const typename Eigen::DSizes<Eigen::Index, *Dims*> &dimensions)

Converts eigen dimensions vector into a std::vector.

ICRAR - International Centre for Radio Astronomy Research (c) UWA - The University of Western Australia Copyright by UWA(in the framework of the ICRAR) All rights reserved

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111 - 1307 USA

Template Parameters

- **Scalar** –
- **Dims** –

Parameters
dimensions –

Returns
std::vector<Eigen::Index>

Specialized Template Function Eigen::internal::cast

- Defined in file_icrar_leap-accelerate_math_cpu_eigen_extensions.h

Function Documentation

template<>
EIGEN_DEVICE_FUNC inline std::complex<double> `Eigen::internal::cast`(const std::complex<float> &x)

Template Function Eigen::ToMatrix

- Defined in file_icrar_leap-accelerate_math_cpu_eigen_extensions.h

Function Documentation

template<typename **Scalar**>
auto `Eigen::ToMatrix`(const Eigen::Tensor<*Scalar*, 2> &tensor)

Template Function Eigen::ToVector

- Defined in file_icrar_leap-accelerate_math_cpu_eigen_extensions.h

Function Documentation

template<typename **Scalar**>
auto `Eigen::ToVector`(const Eigen::Tensor<*Scalar*, 1> &tensor)

Function enable_async

- Defined in file_icrar_leap-accelerate_python_async.h

Function Documentation

pybind11::class_<*CppAwaitable*> **enable_async**(pybind11::module m)

Function GetCudaDeviceCount

- Defined in file_icrar_leap-accelerate_cuda_cuda_info.h

Function Documentation

int **GetCudaDeviceCount**()

Gets the number of available Cuda Devices.

Returns
int

Template Function icrar::AttributeEquals

- Defined in file_icrar_leap-accelerate_tests_gtest_helper.h

Function Documentation

template<typename **T**>
::testing::AssertionResult `icrar::`**AttributeEquals**(MyObject const &obj, *T* value)

Function icrar::ComputeImplementationToString

- Defined in file_icrar_leap-accelerate_core_compute_implementation.h

Function Documentation

std::string `icrar::`**ComputeImplementationToString**(*ComputeImplementation* value)

Converts an enum `value` to a string.

Parameters
value – compute implementation value

Returns
std::string serialized compute imeplementation

Template Function icrar::ConvertMatrix(const Eigen::Matrix<T, R, C>&)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

template<typename **T**, int **R**, int **C**>
casacore::Matrix<*T*> **icrar::ConvertMatrix**(const Eigen::Matrix<*T*, *R*, *C*> &value)

Converts an Eigen3 matrix to the equivalent casacore matrix.

Template Function icrar::ConvertMatrix(const Eigen::Matrix<T, Eigen::Dynamic, Eigen::Dynamic>&)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

template<typename **T**>
casacore::Matrix<*T*> **icrar::ConvertMatrix**(const Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic>
&value)

Converts an Eigen3 matrix to the equivalent casacore matrix.

Template Function icrar::ConvertVector

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

template<typename **T**>
casacore::Vector<*T*> **icrar::ConvertVector**(const Eigen::Matrix<*T*, Eigen::Dynamic, 1> &value)

Converts an Eigen3 column-vector into a casacore Array.

Template Parameters
T – scalar type

Parameters
value – eigen3 vector

Returns
casacore::Array<T>

Template Function icrar::cpu::ceil_div

- Defined in file_icrar_leap-accelerate_math_cpu_math.h

Function Documentation

template<typename **T**>
constexpr *T* `icrar::cpu::`**`ceil_div`**(*T* x, *T* y)

Performs integer division but rounds up if there is a remainder.

Template Parameters
T – integer type

Parameters

- **x** – numerator
- **y** – denominator

Function icrar::cpu::PhaseMatrixFunction

- Defined in file_icrar_leap-accelerate_algorithm_cpu_PhaseMatrixFunction.h

Function Documentation

std::pair<Eigen::MatrixXd, Eigen::VectorXi> `icrar::cpu::`**`PhaseMatrixFunction`**(const Eigen::VectorXi &a1, const Eigen::VectorXi &a2, const Eigen::*VectorXb* &fg, uint32_t refAnt, bool allBaselines)

Generate Phase Matrix Given the antenna lists from an MS and (optionally) reference antenna and antenna flags: If non-negative RefAnt is provided it only forms the matrix for baselines with that antenna. If True Map is provided it returns the index map for the matrix (only useful if RefAnt set).

This function generates and returns the linear matrix for the phase calibration (only)

Parameters

- **a1** – indexes vector of 1st antenna of each baselines
- **a2** – indexes vector of 2nd antenna of each baselines
- **refAnt** – the reference antenna index
- **fg** – a flag vector of flagged baselines to ignore when true
- **allBaselines** – whether to generate phase matrix for all baselines or just ones with reference antenna

Returns
std::pair<Matrixd, Matrixi> for refAnt = none: first matrix is of size [baselines,stations] and seconds of size[baselines,1] for 0 <= refAnt < stations: first matrix is of size [stations,stations] and seconds of size[stations,1]

Template Function icrar::cpu::pseudo_inverse

- Defined in file_icrar_leap-accelerate_math_cpu_matrix_invert.h

Function Documentation

template<typename **T**>
Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic> `icrar::cpu::`**`pseudo_inverse`**(const Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic> &a)

Invert as a function If non-negative RefAnt is provided it only forms the matrix for baselines with that antenna.

This function generates and returns the inverse of the linear matrix to solve for the phase calibration (only) given a MS. The MS is used to fill the columns of the matrix, based on the baselines in the MS (and RefAnt if given)

The output will be the inverse matrix to cross with the observation vector.

Parameters
A –

Template Function icrar::cpu::SVDPseudoInverse

- Defined in file_icrar_leap-accelerate_math_cpu_matrix_invert.h

Function Documentation

template<typename **Matrix_T**>
Matrix_T `icrar::cpu::`**`SVDPseudoInverse`**(const *Matrix_T* &a, double epsilon = std::numeric_limits<typename *Matrix_T*::Scalar>::epsilon())

Calculates the pseudo_inverse matrix of size N * M for a given M * N matrix. Satisfies the equation A = A * Ah * A.

Parameters

- **A** –
- **epsilon** –

Returns
Matrix_T

Function icrar::cuda::AvgDataToPhaseAngles

- Defined in file_icrar_leap-accelerate_algorithm_cuda_kernel_PolarizationsToPhaseAnglesKernel.h

Function Documentation

__host__ void icrar::cuda::**AvgDataToPhaseAngles**(const *device_vector*<int> &I1, const *device_matrix*<std::complex<double>> &avgData, *device_vector*<double> &phaseAnglesI1)

Copies the argument of the 1st column/polarization in avgData to phaseAnglesI1.

Parameters

- **I1** – the index vector for unflagged antennas
- **avgData** – the averaged data matrix
- **phaseAnglesI1** – the output phaseAngles vector

Function icrar::cuda::CalcDeltaPhase

- Defined in file_icrar_leap-accelerate_algorithm_cuda_kernel_ComputePhaseDeltaKernel.h

Function Documentation

__host__ void icrar::cuda::**CalcDeltaPhase**(const *device_matrix*<double> &A, const *device_vector*<double> &cal1, const *device_matrix*<std::complex<double>> &avgData, *device_matrix*<double> &deltaPhase)

Computes the phase delta vector for the first polarization of avgData.

Parameters

- **A** – Antenna matrix
- **cal1** – cal1 matrix
- **avgData** – averaged visibilities
- **deltaPhase** – output deltaPhase vector

Function icrar::cuda::Empty

- Defined in file_icrar_leap-accelerate_algorithm_cuda_kernel_EmptyKernel.h

Function Documentation

__host__ void icrar::cuda::**Empty**()

An empty kernel for testing cuda configurations.

Function icrar::cuda::mat_mul(cublasHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const double *, const double *, double *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void `icrar::cuda::`**`mat_mul`**(cublasHandle_t handle, *MatrixOp* transa, *MatrixOp* transb, const size_t m, const size_t n, const size_t k, const double *a, const double *b, double *out)

Function icrar::cuda::mat_mul(cublasHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const float *, const float *, float *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void `icrar::cuda::`**`mat_mul`**(cublasHandle_t handle, *MatrixOp* transa, *MatrixOp* transb, const size_t m, const size_t n, const size_t k, const float *a, const float *b, float *out)

Function icrar::cuda::mat_mul(cublasHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const int *, const int *, int *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void `icrar::cuda::`**`mat_mul`**(cublasHandle_t handle, *MatrixOp* transa, *MatrixOp* transb, const size_t m, const size_t n, const size_t k, const int *a, const int *b, int *out)

Function icrar::cuda::mat_mul(cublasLtHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const double *, const double *, double *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void `icrar::cuda::`**`mat_mul`**(*cublasLtHandle_t* handle, *MatrixOp* transa, *MatrixOp* transb, const size_t m, const size_t n, const size_t k, const double *a, const double *b, double *out)

Function icrar::cuda::mat_mul(cublasLtHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const float *, const float *, float *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void `icrar::cuda::`**`mat_mul`**(*cublasLtHandle_t* handle, *MatrixOp* transa, *MatrixOp* transb, const size_t m, const size_t n, const size_t k, const float *a, const float *b, float *out)

Function icrar::cuda::mat_mul(cublasLtHandle_t, MatrixOp, MatrixOp, const size_t, const size_t, const size_t, const int *, const int *, int *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void `icrar::cuda::`**`mat_mul`**(*cublasLtHandle_t* handle, *MatrixOp* transa, *MatrixOp* transb, const size_t m, const size_t n, const size_t k, const int *a, const int *b, int *out)

Function icrar::cuda::mat_mul_add(cublasHandle_t, const size_t, const size_t, const size_t, const double *, const double *, double *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void `icrar::cuda::`**`mat_mul_add`**(cublasHandle_t handle, const size_t m, const size_t n, const size_t k, const double *a, const double *b, double *c)

Function icrar::cuda::mat_mul_add(cublasHandle_t, const size_t, const size_t, const size_t, const float *, const float *, float *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void `icrar::cuda::`**`mat_mul_add`**(cublasHandle_t handle, const size_t m, const size_t n, const size_t k, const float *a, const float *b, float *c)

Function icrar::cuda::mat_mul_add(cublasHandle_t, const size_t, const size_t, const size_t, const int *, const int *, int *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void icrar::cuda::**mat_mul_add**(cublasHandle_t handle, const size_t m, const size_t n, const size_t k, const int *a, const int *b, int *c)

Function icrar::cuda::mat_mul_add(cublasLtHandle_t, const size_t, const size_t, const size_t, const double *, const double *, const double *, double *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void icrar::cuda::**mat_mul_add**(*cublasLtHandle_t* handle, const size_t m, const size_t n, const size_t k, const double *a, const double *b, const double *c, double *d)

Function icrar::cuda::mat_mul_add(cublasLtHandle_t, const size_t, const size_t, const size_t, const float *, const float *, const float *, float *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void icrar::cuda::**mat_mul_add**(*cublasLtHandle_t* handle, const size_t m, const size_t n, const size_t k, const float *a, const float *b, const float *c, float *d)

Function icrar::cuda::mat_mul_add(cublasLtHandle_t, const size_t, const size_t, const size_t, const int *, const int *, const int *, int *)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

__host__ void icrar::cuda::**mat_mul_add**(*cublasLtHandle_t* handle, const size_t m, const size_t n, const size_t k, const int *a, const int *b, const int *c, int *d)

Template Function icrar::cuda::multiply(cublasHandle_t, const device_matrix<T>&, const device_vector<T>&, device_vector<T>&, MatrixOp, MatrixOp)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

template<typename **T**>
__host__ void `icrar::cuda::multiply`(cublasHandle_t handle, const *device_matrix*<*T*> &a, const *device_vector*<*T*> &b, *device_vector*<*T*> &c, *MatrixOp* transa = *MatrixOp*::*normal*, *MatrixOp* transb = *MatrixOp*::*normal*)

Performs matrix-vector multiplication where C = opA(A) * opB(B). The transpose and hermetian of A and B can be used instead.

Template Parameters
T – scalar type

Parameters

- **handle** – cublas handle
- **a** – A matrix in C = opA(A) * opB(B)
- **b** – B matrix in C = opA(A) * opB(B)
- **c** – C matrix in C = opA(A) * opB(B)
- **opA** – A matrix operation in C = opA(A) * opB(B)
- **opB** – B matrix operation in C = opA(A) * opB(B)

Template Function icrar::cuda::multiply(cublasHandle_t, const device_matrix<T>&, const device_matrix<T>&, device_matrix<T>&, MatrixOp, MatrixOp)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

template<typename **T**>
__host__ void `icrar::cuda::multiply`(cublasHandle_t handle, const *device_matrix*<*T*> &a, const *device_matrix*<*T*> &b, *device_matrix*<*T*> &c, *MatrixOp* transa = *MatrixOp*::*normal*, *MatrixOp* transb = *MatrixOp*::*normal*)

Template Function icrar::cuda::multiply(cublasLtHandle_t, const device_matrix<T>&, const device_vector<T>&, device_vector<T>&, MatrixOp, MatrixOp)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

template<typename **T**>
__host__ void **icrar::cuda::multiply**(*cublasLtHandle_t* handle, const *device_matrix*<*T*> &a, const *device_vector*<*T*> &b, *device_vector*<*T*> &c, *MatrixOp* transa = *MatrixOp*::*normal*, *MatrixOp* transb = *MatrixOp*::*normal*)

Template Function icrar::cuda::multiply(cublasLtHandle_t, const device_matrix<T>&, const device_matrix<T>&, device_matrix<T>&, MatrixOp, MatrixOp)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

template<typename **T**>
__host__ void **icrar::cuda::multiply**(*cublasLtHandle_t* handle, const *device_matrix*<*T*> &a, const *device_matrix*<*T*> &b, *device_matrix*<*T*> &c, *MatrixOp* transa = *MatrixOp*::*normal*, *MatrixOp* transb = *MatrixOp*::*normal*)

Template Function icrar::cuda::multiply_add(cublasLtHandle_t, const device_matrix<T>&, const device_vector<T>&, const device_vector<T>&, device_vector<T>&)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

template<typename **T**>
__host__ void **icrar::cuda::multiply_add**(*cublasLtHandle_t* handle, const *device_matrix*<*T*> &a, const *device_vector*<*T*> &b, const *device_vector*<*T*> &c, *device_vector*<*T*> &d)

Template Function icrar::cuda::multiply_add(cublasHandle_t, const device_matrix<T>&, const device_matrix<T>&, device_matrix<T>&)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

template<typename **T**>
__host__ void **icrar::cuda::multiply_add**(cublasHandle_t handle, const *device_matrix*<*T*> &a, const *device_matrix*<*T*> &b, *device_matrix*<*T*> &c)

Template Function icrar::cuda::multiply_add(cublasHandle_t, const device_matrix<T>&, const device_vector<T>&, device_vector<T>&)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

template<typename **T**>
__host__ void `icrar::cuda::`**`multiply_add`**(cublasHandle_t handle, const *device_matrix*<*T*> &a, const *device_vector*<*T*> &b, *device_vector*<*T*> &c)

Template Function icrar::cuda::multiply_add(cublasLtHandle_t, const device_matrix<T>&, const device_matrix<T>&, const device_matrix<T>&, device_matrix<T>&)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Function Documentation

template<typename **T**>
__host__ void `icrar::cuda::`**`multiply_add`**(*cublasLtHandle_t* handle, const *device_matrix*<*T*> &a, const *device_matrix*<*T*> &b, const *device_matrix*<*T*> &c, *device_matrix*<*T*> &d)

Function icrar::cuda::PhaseRotateAverageVisibilities

- Defined in file_icrar_leap-accelerate_algorithm_cuda_kernel_PhaseRotateAverageVisibilitiesKernel.h

Function Documentation

__host__ void `icrar::cuda::`**`PhaseRotateAverageVisibilities`**(*DeviceIntegration* &integration, *DeviceLeapData* &leapData)

Calculates avgData in leapData.

Parameters

- **`integration`** – the input visibilities to integrate
- **`leapData`** – the leapData container

Function icrar::cuda::pseudo_inverse(cusolverDnHandle_t, cublasHandle_t, const Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic>&, const JobType)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_invert.h

Function Documentation

Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> `icrar::cuda::pseudo_inverse`(cusolverDnHandle_t cusolverHandle, cublasHandle_t cublasHandle, const Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> &a, const *JobType* jobtype = *JobType*::*S*)

Computes the moore penrose pseudo inverse where A'A = I (left inverse)

Parameters

- **cusolverHandle** – cusolver handle
- **cublasHandle** – cublas handle
- **a** – cpu memory matrix to invert
- **jobtype** – SVD matrix dimension type

Returns
Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic>

Function icrar::cuda::pseudo_inverse(cusolverDnHandle_t, cublasHandle_t, const device_matrix<double>&, const JobType)

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_invert.h

Function Documentation

device_matrix<double> `icrar::cuda::pseudo_inverse`(cusolverDnHandle_t cusolverHandle, cublasHandle_t cublasHandle, const *device_matrix*<double> &matrix, const *JobType* jobType = *JobType*::*S*)

Performs matrix inversion using cusolver and cublas.

Parameters

- **cusolverHandle** – cusolver handle
- **cublasHandle** – cublas handle
- **matrix** – device memory matrix to invert
- **jobType** – type of device buffer matrices used by kernel

Returns
device_matrix<double>

Function icrar::cuda::SliceDeltaPhase

- Defined in file_icrar_leap-accelerate_algorithm_cuda_kernel_SliceDeltaPhaseKernel.h

Function Documentation

__host__ void **icrar::cuda::SliceDeltaPhase**(const *device_matrix*<double> &deltaPhase, *device_vector*<double> &deltaPhaseColumn)

Copies the first column of deltaPhase into deltaPhaseColumn.

Parameters

- **deltaPhase** – The delta phase matrix
- **deltaPhaseColumn** – The output delta phase vector/column

Function icrar::cuda::svd

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_invert.h

Function Documentation

std::tuple<*device_matrix*<double>, *device_vector*<double>, *device_matrix*<double>> **icrar::cuda::svd**(cusolverDnHandle_t cusolverHandle, const *device_matrix*<double> &deviceA, const *JobType* jobType)

Computes the U, S and Vt values of matrix singular value decomposition.

Parameters

- **cusolverHandle** – cusolver handle
- **deviceA** – device memory matrix to invert
- **jobType** – type of device buffer matrices used by kernel

Returns

std::tuple<device_matrix<double>, device_vector<double>, device_matrix<double>> tuple of U, S and V matrices

Function icrar::cuda::ToCublasOp

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_op.h

Function Documentation

cublasOperation_t `icrar::cuda::`**`ToCublasOp`**(*MatrixOp* op)

Converts a matrix operation to a cublas operation.

Parameters
op –

Returns
cublasOperation_t

Template Function icrar::detail::operator<<(std::basic_ostream<T>&, detail::_fixed<N, VT>)

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Function Documentation

template<typename **T**, int **N**, typename **VT**>
inline std::basic_ostream<*T*> &`icrar::detail::`**`operator<<`**(std::basic_ostream<*T*> &os, detail::*_fixed*<*N*, *VT*> v)

Template Function icrar::detail::operator<<(std::basic_ostream<T>&, const detail::_memory_amount&)

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Function Documentation

template<typename **T**>
inline std::basic_ostream<*T*> &`icrar::detail::`**`operator<<`**(std::basic_ostream<*T*> &os, const detail::*_memory_amount* &m)

Template Function icrar::detail::operator<<(std::basic_ostream<T>&, const detail::_microseconds_amount&)

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Function Documentation

template<typename **T**>
inline std::basic_ostream<*T*> &**icrar::detail::operator<<**(std::basic_ostream<*T*> &os, const detail::*_microseconds_amount* &t)

Function icrar::exists

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

inline bool **icrar::exists**(const std::string &name)

Template Function icrar::fixed

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Function Documentation

template<int **N**, typename **T**>
inline detail::*_fixed*<*N*, *T*> **icrar::fixed**(*T* v)

When streamed, this manipulator will print the given value with a precision of N decimal places.

Template Parameters

- **N** – number of decimal places
- **T** – input type

Parameters

v – The value to send to the stream

Returns

detail::_fixed<N, T>

Function icrar::GetAllTimestepsMWACalibration

- Defined in file_icrar_leap-accelerate_tests_algorithm_PhaseRotateTestCaseData.h

Function Documentation

cpu::*CalibrationCollection* **icrar::GetAllTimestepsMWACalibration**()

Gets the expected calibration output averaging over all timesteps. From LEAP-Cal:ported.

Returns

a vector of direction and antenna calibration pairs

Function icrar::GetAvailableCudaPhysicalMemory

- Defined in file_icrar_leap-accelerate_core_memory_system_memory.h

Function Documentation

size_t `icrar::GetAvailableCudaPhysicalMemory()`

Gets the currently available/free physical cuda memory of the current cuda device. This excludes the memory used by the current process.

Function icrar::GetEachTimestepMWACalibration

- Defined in file_icrar_leap-accelerate_tests_algorithm_PhaseRotateTestCaseData.h

Function Documentation

cpu::*CalibrationCollection* `icrar::GetEachTimestepMWACalibration()`

Gets the expected calibration output of each timestep with no averaging. From LEAP-Cal:ported.

Returns

a vector of direction and antenna calibration pairs

Function icrar::GetFirstTimestepMWACalibration

- Defined in file_icrar_leap-accelerate_tests_algorithm_PhaseRotateTestCaseData.h

Function Documentation

cpu::*CalibrationCollection* `icrar::GetFirstTimestepMWACalibration()`

Gets the expected calibration output averaging over the first timestep.

Returns

std::vector<std::pair<SphericalDirection, std::vector<double>>>

Function icrar::GetTotalAvailableSystemVirtualMemory

- Defined in file_icrar_leap-accelerate_core_memory_system_memory.h

Function Documentation

size_t `icrar::GetTotalAvailableSystemVirtualMemory()`

Gets the currently available/free virtual system memory.

Function icrar::GetTotalCudaPhysicalMemory

- Defined in file_icrar_leap-accelerate_core_memory_system_memory.h

Function Documentation

size_t `icrar::GetTotalCudaPhysicalMemory()`

Gets the total physical cuda memory on the current cuda device.

Function icrar::GetTotalSystemVirtualMemory

- Defined in file_icrar_leap-accelerate_core_memory_system_memory.h

Function Documentation

size_t `icrar::GetTotalSystemVirtualMemory()`

Gets the total amount system virtual memory. This includes the system's dynamic RAM plus swap space.

Function icrar::GetTotalUsedSystemVirtualMemory

- Defined in file_icrar_leap-accelerate_core_memory_system_memory.h

Function Documentation

size_t `icrar::GetTotalUsedSystemVirtualMemory()`

Gets the total amount of used system virtual memory.

Function icrar::git_has_local_changes

- Defined in file_icrar_leap-accelerate_core_git_revision.h

Function Documentation

bool `icrar::git_has_local_changes()`

Returns

Whether the local clone of the repository has uncommitted changes.

Function icrar::git_sha1

- Defined in file_icrar_leap-accelerate_core_git_revision.h

Function Documentation

std::string **icrar::git_sha1**()

Returns
The git SHA1 value for the current source code commit.

Template Function icrar::isApprox(const Tensor3X<T>&, const Tensor3X<T>&, double)

- Defined in file_icrar_leap-accelerate_common_Tensor3X.h

Function Documentation

template<typename **T**>
bool **icrar::isApprox**(const *Tensor3X*<*T*> &lhs, const *Tensor3X*<*T*> &rhs, double tolerance)

Template Function icrar::isApprox(const std::complex<T>&, const std::complex<T>&, T)

- Defined in file_icrar_leap-accelerate_math_complex_extensions.h

Function Documentation

template<typename **T**>
bool **icrar::isApprox**(const std::complex<*T*> &lhs, const std::complex<*T*> &rhs, *T* threshold)

returns true if the magnitude of the difference between two values are approximately equal (within the specified threshold)

Template Parameters
T – input type

Parameters

- **lhs** – left value
- **rhs** – right value
- **threshold** – threshold where an equal absolute differance is considered as not approximately equal

Returns
true if left value approximately equals right value

Returns
false if left value does not approximately equals right value

Template Function icrar::isApprox(const std::vector<T>&, const std::vector<T>&, T)

- Defined in file_icrar_leap-accelerate_math_vector_extensions.h

Function Documentation

template<typename **T**>
bool `icrar::`**`isApprox`**(const std::vector<*T*> &lhs, const std::vector<*T*> &rhs, *T* tolerance)

Returns true if all vector elements of `lhs` are within the tolerance threshold to `rhs`.

Template Parameters
T – numeric type

Parameters

- **lhs** – left hand side
- **rhs** – right hand side
- **tolerance** – tolerance threshold

Function icrar::IsImmediateMode

- Defined in file_icrar_leap-accelerate_core_stream_out_type.h

Function Documentation

bool `icrar::`**`IsImmediateMode`**(*StreamOutType* streamOutType)

True if solutions should be written to IO as soon as they are computed.

Function icrar::log::Initialize

- Defined in file_icrar_leap-accelerate_core_log_logging.h

Function Documentation

void `icrar::log::`**`Initialize`**(*Verbosity* verbosity = *DEFAULT_VERBOSITY*)

Initializes logging singletons.

Parameters
verbosity – The verbosity to initialize the library with, higher values yield more verbose output.

Function icrar::log::ParseVerbosity

- Defined in file_icrar_leap-accelerate_core_log_Verbosity.h

Function Documentation

Verbosity `icrar::log::ParseVerbosity`(const std::string &value)

Parses string argument into an enum, throws an exception otherwise.

Parameters
`value` – serialized verbosity string

Returns
Verbosity

Function icrar::log::TryParseVerbosity

- Defined in file_icrar_leap-accelerate_core_log_Verbosity.h

Function Documentation

bool `icrar::log::TryParseVerbosity`(const std::string &value, *Verbosity* &out)

Returns
true if value was converted succesfully, false otherwise.

Template Function icrar::matrix_hash

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename `T`>
std::size_t `icrar::matrix_hash`(const *T* &matrix)

Hash function for Eigen matrix and vector. The code is from `hash_combine` function of the Boost library. See http://www.boost.org/doc/libs/1_55_0/doc/html/hash/reference.html#boost.hash_combine .

Template Parameters
`T` – Eigen Dense Matrix type

Function icrar::memory_amount

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Function Documentation

inline detail::*_memory_amount* **icrar::memory_amount**(std::size_t amount)

Sent to a stream object, this manipulator will print the given amount of memory using the correct suffix and 3 decimal places.

Parameters
amount – The value to send to the stream

Returns
detail::_memory_amount

Function icrar::ParseComputeImplementation

- Defined in file_icrar_leap-accelerate_core_compute_implementation.h

Function Documentation

ComputeImplementation **icrar::ParseComputeImplementation**(const std::string &value)

Parses string argument into an enum, throws an exception otherwise.

Parameters
value – serialized compute imeplementation

Returns
ComputeImplementation compute implementation value

Function icrar::ParseDirections(const std::string&)

- Defined in file_icrar_leap-accelerate_common_SphericalDirection.h

Function Documentation

std::vector<*SphericalDirection*> **icrar::ParseDirections**(const std::string &json)

Parses a json string to a collection of MVDirections.

Parameters
json –

Returns
std::vector<SphericalDirection>

Function icrar::ParseDirections(const rapidjson::Value&)

- Defined in file_icrar_leap-accelerate_common_SphericalDirection.h

Function Documentation

std::vector<*SphericalDirection*> icrar::**ParseDirections**(const rapidjson::Value &doc)

Parses a json object to a collection of MVDirections.

Function icrar::ParseInputType

- Defined in file_icrar_leap-accelerate_core_InputType.h

Function Documentation

InputType icrar::**ParseInputType**(const std::string &value)

Parses string argument into an enum, throws an exception otherwise.

Parameters
value –

Returns
StreamOutType

Function icrar::ParseSlice(const std::string&)

- Defined in file_icrar_leap-accelerate_common_Slice.h

Function Documentation

Slice icrar::**ParseSlice**(const std::string &json)

Function icrar::ParseSlice(const rapidjson::Value&)

- Defined in file_icrar_leap-accelerate_common_Slice.h

Function Documentation

Slice icrar::**ParseSlice**(const rapidjson::Value &doc)

Function icrar::ParseStreamOutType

- Defined in file_icrar_leap-accelerate_core_stream_out_type.h

Function Documentation

StreamOutType icrar::**ParseStreamOutType**(const std::string &value)

Parses string argument into an enum, throws an exception otherwise.

Parameters
value –

Returns
StreamOutType

Template Function icrar::pretty_matrix

- Defined in file_icrar_leap-accelerate_common_eigen_stringutils.h

Function Documentation

template<typename **Matrix**>
std::string icrar::**pretty_matrix**(const Eigen::MatrixBase<*Matrix*> &value)

Prints a formatted matrix to a string with a maximum of 6 rows and columns displayed.

Template Parameters
Matrix – Eigen Matrix type

Parameters
value – the matrix to print

Returns
std::string the formatted string result

Template Function icrar::pretty_row

- Defined in file_icrar_leap-accelerate_common_eigen_stringutils.h

Function Documentation

template<typename **RowVector**>
void icrar::**pretty_row**(const *RowVector* &row, std::stringstream &ss)

Prints a formatted displaying up to 6 elements.

Template Parameters
RowVector – Eigen RowVector type

Parameters

- **row** – the row to print
- **ss** – the stream to print to

Template Function icrar::ProcessCache(size_t, const In&, Out&, const std::string&, const std::string&, Lambda)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename **In**, typename **Out**, typename **Lambda**>
void `icrar::ProcessCache`(size_t hash, const *In* &in, *Out* &out, const std::string &hashFile, const std::string &cacheFile, *Lambda* transform)

Reads the hash file and writes to cache if the hash file is different, else reads the cache file if hash file is the same.

Template Parameters

- **In** – Matrix type
- **Out** – Matrix type
- **Lambda** – lambda type of signature Out(const In&) called if hashes do not match

Parameters

- **in** – The input matrix to hash and transform
- **out** – The transformed output
- **transform** – the transform lambda
- **cacheFile** – the transformed out cache file
- **hashFile** – the in hash file

Template Function icrar::ProcessCache(size_t, const In&, const std::string&, const std::string&, Lambda)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename **In**, typename **Out**, typename **Lambda**>
Out `icrar::ProcessCache`(size_t hash, const *In* &in, const std::string &hashFile, const std::string &cacheFile, *Lambda* transform)

Template Function icrar::ProcessCache(const In&, const std::string&, Lambda, Out&)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename **In**, typename **Out**, typename **Lambda**>
void `icrar::`**ProcessCache**(const *In* &in, const std::string &cacheFile, *Lambda* transform, *Out* &out)

Template Function icrar::ProcessCache(const In&, const std::string&, Lambda)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename **In**, typename **Out**, typename **Lambda**>
Out `icrar::`**ProcessCache**(const *In* &in, const std::string &cacheFile, *Lambda* transform)

Function icrar::profiling::get_resource_usage

- Defined in file_icrar_leap-accelerate_core_profiling_resource_usage.h

Function Documentation

ResourceUsage `icrar::profiling::`**get_resource_usage**()

Returns the maximum Resident Storage Size of this process.

Template Function icrar::profiling::operator<<(std::basic_ostream<CharT>&, const ResourceUsage&)

- Defined in file_icrar_leap-accelerate_core_profiling_resource_usage.h

Function Documentation

template<typename **CharT**>
std::basic_ostream<*CharT*> &`icrar::profiling::`**operator<<**(std::basic_ostream<*CharT*> &os, const *ResourceUsage* &ru)

Stream output operator for instances of ResourceUsage.

Template Function icrar::profiling::operator<<(std::basic_ostream<CharT, Traits>&, const timer&)

- Defined in file_icrar_leap-accelerate_core_profiling_timer.h

Function Documentation

template<typename **CharT**, typename **Traits**>
std::basic_ostream<*CharT*, *Traits*> &**icrar::profiling::operator<<**(std::basic_ostream<*CharT*, *Traits*> &os,
const *timer* &timer)

Template Function icrar::range(IntType, IntType, IntType)

- Defined in file_icrar_leap-accelerate_math_vector_extensions.h

Function Documentation

template<typename **IntType**>
std::vector<*IntType*> **icrar::range**(*IntType* start, *IntType* stop, *IntType* step)

returns a linear sequence of values from start at step sized intervals to the stop value inclusive

Template Parameters
IntType – integer type

Parameters

- **start** – start index
- **stop** – exclusive end inex
- **step** – increment between generated elements

Returns
std::vector<IntType>

Template Function icrar::range(IntType, IntType)

- Defined in file_icrar_leap-accelerate_math_vector_extensions.h

Function Documentation

template<typename **IntType**>
std::vector<*IntType*> **icrar::range**(*IntType* start, *IntType* stop)

returns a linear sequence of values from start to stop

Template Parameters
IntType – integer type

Parameters

- **start** – start index
- **stop** – exclusive end index

Returns
std::vector<IntType>

Template Function icrar::range(IntType)

- Defined in file_icrar_leap-accelerate_math_vector_extensions.h

Function Documentation

template<typename **IntType**>
std::vector<*IntType*> `icrar::`**`range`**`(`*IntType* stop`)`

returns a linear sequence of values from 0 to stop

Template Parameters
`IntType` – integer type

Parameters
`stop` – exclusive end index

Returns
std::vector<IntType>

Template Function icrar::read_binary(const char *, Matrix&)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<class **Matrix**>
void `icrar::`**`read_binary`**`(`const char *filepath, *Matrix* &matrix`)`

Reads `matrix` from a file by resizing and overwriting the existing matrix (throws if fails)

Template Parameters
`Matrix` – Eigen Matrix type

Parameters

- **`filepath`** – filepath to read from
- **`matrix`** – matrix to read

Template Function icrar::read_binary(std::ifstream&, Matrix&)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<class **Matrix**>
void `icrar::`**`read_binary`**`(`std::ifstream &in, *Matrix* &matrix`)`

Template Function icrar::read_hash(const char *, T&)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename **T**>
void `icrar::`**`read_hash`**(const char *filename, *T* &hash)

Reads a file containing a binary hash at `filename` and outputs to `hash`.

Template Parameters
T – the hash type

Parameters

- **`filename`** – the hash file to read
- **`hash`** – output parameter

Template Function icrar::read_hash(std::ifstream&, T&)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename **T**>
void `icrar::`**`read_hash`**(std::ifstream &stream, *T* &hash)

Function icrar::RunCalibration

- Defined in file_icrar_leap-accelerate_algorithm_Calibrate.h

Function Documentation

void `icrar::`**`RunCalibration`**(const *Arguments* &args)

Runs leap calibration using a set of arguments.

Parameters
args –

Template Function icrar::to_underlying_type

- Defined in file_icrar_leap-accelerate_common_enumutils.h

Function Documentation

template<typename **T**>
std::underlying_type_t<*T*> **icrar::to_underlying_type**(*T* e)

Safely casts an enum to its underlying type.

Template Parameters
T – enum with underlying type

Parameters
e – enum value

Returns
std::underlying_type_t<T> the cast value

Function icrar::ToCasaDirection

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

casacore::MVDirection **icrar::ToCasaDirection**(const *SphericalDirection* &value)

Converts an icrar spherical direction to a casacore direction.

Parameters
value – value to convert

Returns
casacore::MVDirection

Function icrar::ToCasaDirectionVector

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

std::vector<casacore::MVDirection> **icrar::ToCasaDirectionVector**(const std::vector<*SphericalDirection*> &value)

Convers an icrar spherical direction vector to a casacore direction vector.

Parameters
value – value to convert

Returns
std::vector<casacore::MVDirection>

Function icrar::ToCasaUVW

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

casacore::MVuvw **icrar::ToCasaUVW**(const icrar::*MVuvw* &value)

Converts an icrar UVW value to a casacore UVW value.

Function icrar::ToCasaUVWVector(const std::vector<icrar::MVuvw>&)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

std::vector<casacore::MVuvw> **icrar::ToCasaUVWVector**(const std::vector<icrar::*MVuvw*> &value)

Converts an icrar UVW vector to a casacore UVW vector.

Function icrar::ToCasaUVWVector(const Eigen::MatrixX3d&)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

std::vector<casacore::MVuvw> **icrar::ToCasaUVWVector**(const Eigen::MatrixX3d &value)

Converts an icrar UVW vector to a casacore UVW vector.

Parameters

value – value to convert

Returns

std::vector<casacore::MVuvw>

Function icrar::ToDirection

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

SphericalDirection **icrar::ToDirection**(const casacore::MVDirection &value)

Converts a casacore direction to an icrar spherical direction.

Parameters

value – value to convert

Returns

SphericalDirection

Function icrar::ToDirectionVector

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

std::vector<*SphericalDirection*> **icrar::ToDirectionVector**(const std::vector<casacore::MVDirection> &value)

Converts a casacore Direction vector to an icrar Spherical Direction.

Parameters
value – value to convert

Returns
std::vector<SphericalDirection>

Template Function icrar::ToFixedMatrix

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

template<typename **T**, int **R**, int **C**>
Eigen::Matrix<*T*, *R*, *C*> **icrar::ToFixedMatrix**(const casacore::Matrix<*T*> &value)

Converts a casacore matrix to a fixed size eigen3 matrix.

Template Parameters

- **T** – scalar type
- **R** – rows
- **C** – columns

Parameters
value – casacore matrix to convert

Returns
Eigen::Matrix<T, R, C>

Template Function icrar::ToMatrix(const casacore::Matrix<T>&)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

template<typename **T**>
Eigen::Matrix<*T*, Eigen::Dynamic, Eigen::Dynamic> `icrar::ToMatrix`(const casacore::Matrix<*T*> &value)

Converts a casacore matrix to the equivalent eigen3 matrix

Function icrar::ToMatrix(const std::vector<MVuvw>&)

- Defined in file_icrar_leap-accelerate_model_cpu_MVuvw.h

Function Documentation

Eigen::Matrix<double, Eigen::Dynamic, 3> `icrar::ToMatrix`(const std::vector<*MVuvw*> &uvws)

Function icrar::ToUVW

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

icrar::*MVuvw* `icrar::ToUVW`(const casacore::MVuvw &value)

Converts a casacore UVW value to an icrar UVW value.

Parameters
value – casacore uvw

Returns
icrar::MVuvw

Function icrar::ToUVWVector(const std::vector<casacore::MVuvw>&)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

std::vector<icrar::*MVuvw*> `icrar::ToUVWVector`(const std::vector<casacore::MVuvw> &value)

Converts a casacore UVW vector to an icrar UVW vector.

Parameters
value – value to convert

Returns
std::vector<icrar::MVuvw>

Function icrar::ToUVWVector(const Eigen::MatrixXd&)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

std::vector<icrar::*MVuvw*> **icrar::ToUVWVector**(const Eigen::MatrixXd &value)

Converts a column-major matrix of size Nx3 into a vector of UVWs.

Parameters
value – value to convert

Returns
std::vector<icrar::MVuvw>

Template Function icrar::ToVector(casacore::Vector<T>)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

template<typename **T**>
Eigen::Matrix<*T*, Eigen::Dynamic, 1> **icrar::ToVector**(casacore::Vector<*T*> value)

Converts a casacore vector to the equivalent Eigen3 vector.

Template Function icrar::ToVector(const std::vector<T>&)

- Defined in file_icrar_leap-accelerate_math_math_conversion.h

Function Documentation

template<typename **T**>
Eigen::Matrix<*T*, Eigen::Dynamic, 1> **icrar::ToVector**(const std::vector<*T*> &value)

Converts a std vector to the equivalent Eigen3 vector.

Template Function icrar::trace_matrix

- Defined in file_icrar_leap-accelerate_common_eigen_stringutils.h

Function Documentation

template<typename **Matrix**>
void **icrar::trace_matrix**(const *Matrix* &value, const std::string &name)

Dumps a matrix to file **name** .txt.

Template Parameters
Matrix – Eigen Matrix type

Parameters

- **value** – matrix to dump to file
- **name** – name of the matrix to dump

Function icrar::TryParseComputeImplementation

- Defined in file_icrar_leap-accelerate_core_compute_implementation.h

Function Documentation

bool icrar::**TryParseComputeImplementation**(const std::string &value, *ComputeImplementation* &out)

Safely parses a string to a compute implementation by returning true if the conversion was successful.

Parameters

- **value** – serialized compute implementation string
- **out** – out compute implemation value that is mutated on success, unmodified otherwise

Returns

true if value was converted succesfully, false otherwise

Returns

false if value was converted unsucessfully

Function icrar::TryParseInputType

- Defined in file_icrar_leap-accelerate_core_InputType.h

Function Documentation

bool icrar::**TryParseInputType**(const std::string &value, *InputType* &out)

Returns

true if value was converted succesfully, false otherwise

Function icrar::TryParseStreamOutType

- Defined in file_icrar_leap-accelerate_core_stream_out_type.h

Function Documentation

bool icrar::**TryParseStreamOutType**(const std::string &value, *StreamOutType* &out)

Returns

true if value was converted succesfully, false otherwise

Function icrar::us_time

- Defined in file_icrar_leap-accelerate_core_memory_ioutils.h

Function Documentation

inline detail::*_microseconds_amount* **icrar::us_time**(std::chrono::microseconds::rep amount)

Sent to a stream object, this manipulator will print the given amount of nanoseconds using the correct suffix and 3 decimal places.

Parameters

amount – The value to send to the stream

Returns

detail::_microseconds_amount

Template Function icrar::vector_map

- Defined in file_icrar_leap-accelerate_math_vector_extensions.h

Function Documentation

template<typename **Func**, typename **Seq**>
auto **icrar::vector_map**(*Func* func, const *Seq* &seq)

Performs a std::transform on a collection into a newly allocated std::vector.

Template Parameters

- **Func** – function of signature return_type(const value_type&)
- **Seq** – iterable collection of type value_type

Parameters

- **func** – transformation function
- **seq** – iterable collection to transform

Returns

std::vector<value_type>

Function icrar::version

- Defined in file_icrar_leap-accelerate_core_version.h

Function Documentation

std::string icrar::**version**()

Returns the version of this library as a single string

Returns
The version of this library

Template Function icrar::write_binary(const char *, const Matrix&)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<class **Matrix**>
void icrar::**write_binary**(const char *filepath, const *Matrix* &matrix)

Writes `matrix` to a file overwriting existing content (throws if fails)

Template Parameters
Matrix – Eigen Matrix type

Parameters

- **filepath** – filepath to write to
- **matrix** – matrix to write

Template Function icrar::write_binary(std::ofstream&, const Matrix&)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<class **Matrix**>
void icrar::**write_binary**(std::ofstream &stream, const *Matrix* &matrix)

Template Function icrar::write_hash(const char *, T)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename **T**>
void icrar::**write_hash**(const char *filename, *T* hash)

Writes a hash value to a specified file.

Template Parameters
T – the hash value

Parameters

- **filename** – the hash file to write to

- **hash** – the hash value

Template Function icrar::write_hash(std::ofstream&, T)

- Defined in file_icrar_leap-accelerate_common_eigen_cache.h

Function Documentation

template<typename **T**>
void `icrar::`**write_hash**(std::ofstream &stream, *T* hash)

Template Function operator<<(std::ostream&, const std::set<T>&)

- Defined in file_icrar_leap-accelerate_common_stream_extensions.h

Function Documentation

template<typename **T**>
std::ostream &**operator<<**(std::ostream &os, const std::set<*T*> &v)

Prints a set of streamable values.

Template Parameters
T – streamable type

Parameters

- **os** – output stream
- **v** – set

Returns
std::ostream&

Template Function operator<<(std::ostream&, const std::map<T, S>&)

- Defined in file_icrar_leap-accelerate_common_stream_extensions.h

Function Documentation

template<typename **T**, typename **S**>
std::ostream &**operator<<**(std::ostream &os, const std::map<*T*, *S*> &m)

Prints a mapping of streamable key-value pairs.

Template Parameters

- **T** – streamable key type
- **S** – streamable value type

Parameters

- **os** – output stream
- **m** – map

Returns
std::ostream&

Template Function operator<<(std::ostream&, const std::vector<T>&)

- Defined in file_icrar_leap-accelerate_math_vector_extensions.h

Function Documentation

template<typename **T**>
std::ostream &**operator<<**(std::ostream &os, const std::vector<*T*> &v)

Provides stream operator for std::vector as a json-like literal.

ICRAR - International Centre for Radio Astronomy Research (c) UWA - The University of Western Australia
Copyright by UWA(in the framework of the ICRAR) All rights reserved

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Template Parameters
T – streamable type

Parameters

- **os** – output stream
- **v** – vector

Returns
std::ostream&

Function printCudaVersion

- Defined in file_icrar_leap-accelerate_cuda_cuda_info.h

Function Documentation

void **printCudaVersion**()

Prints running cuda device info to the output log.

Template Function PybindEigenTensor

- Defined in file_icrar_leap-accelerate_python_pybind_eigen.h

Function Documentation

template<typename **Scalar**, int **Dims**, typename ...**InitArgs**>
void **PybindEigenTensor**(pybind11::module &m, const char *name)

Creates a class binding for an Eigen Tensor template. Supports both buffer protocol and eigen array wrappers to python types.

Template Parameters

- **Scalar** – scalar datatype
- **Dims** – number of dimensions
- **InitArgs** – constructor argument types

Parameters

- **m** – module
- **name** – class name

3.3.5 Variables

Variable icrar::constants::speed_of_light

- Defined in file_icrar_leap-accelerate_common_constants.h

Variable Documentation

constexpr double `icrar::constants::`**`speed_of_light`** = 299792458.0

Speed of light in meters per second.

Variable icrar::log::DEFAULT_VERBOSITY

- Defined in file_icrar_leap-accelerate_core_log_logging.h

Variable Documentation

constexpr *Verbosity* `icrar::log::`**`DEFAULT_VERBOSITY`** = *Verbosity*::*info*

The default verbosity level with which the logging system is initialized.

Variable icrar::log::logging_level

- Defined in file_icrar_leap-accelerate_core_log_logging.h

Variable Documentation

extern ::boost::log::trivial::severity_level `icrar::log::`**`logging_level`**

The logging level set on the application.

Variable pretty_width

- Defined in file_icrar_leap-accelerate_common_eigen_stringutils.h

Variable Documentation

constexpr int **`pretty_width`** = 12

3.3.6 Typedefs

Typedef cublasLtHandle_t

- Defined in file_icrar_leap-accelerate_math_cuda_matrix_multiply.h

Typedef Documentation

using **cublasLtHandle_t** = int

ICRAR - International Centre for Radio Astronomy Research (c) UWA - The University of Western Australia Copyright by UWA(in the framework of the ICRAR) All rights reserved

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Typedef Eigen::MatrixXb

- Defined in file_icrar_leap-accelerate_math_cpu_eigen_extensions.h

Typedef Documentation

using `Eigen::`**MatrixXb** = Eigen::Matrix<bool, Eigen::Dynamic, Eigen::Dynamic>

Typedef Eigen::VectorXb

- Defined in file_icrar_leap-accelerate_math_cpu_eigen_extensions.h

Typedef Documentation

using `Eigen::`**VectorXb** = Eigen::Vector<bool, Eigen::Dynamic>

Typedef icrar::cuda::device_tensor3

- Defined in file_icrar_leap-accelerate_cuda_device_tensor.h

Typedef Documentation

template<typename **T**>

using `icrar::cuda::`**`device_tensor3`** = *device_tensor*<*T*, 3>

Typedef icrar::cuda::device_tensor4

- Defined in file_icrar_leap-accelerate_cuda_device_tensor.h

Typedef Documentation

template<typename **T**>

using `icrar::cuda::`**`device_tensor4`** = *device_tensor*<*T*, 4>

Typedef icrar::MVuvw

- Defined in file_icrar_leap-accelerate_model_cpu_MVuvw.h

Typedef Documentation

using `icrar::`**`MVuvw`** = Eigen::Vector3d

Typedef icrar::profiling::usec_t

- Defined in file_icrar_leap-accelerate_core_profiling_resource_usage.h

Typedef Documentation

using `icrar::profiling::`**`usec_t`** = decltype(timeval::tv_sec)

Typedef icrar::Rangei

- Defined in file_icrar_leap-accelerate_common_Range.h

Typedef Documentation

using `icrar::Rangei` = *Range*<int32_t>

Typedef icrar::Rangel

- Defined in file_icrar_leap-accelerate_common_Range.h

Typedef Documentation

using `icrar::Rangel` = *Range*<int64_t>

Typedef icrar::SphericalDirection

- Defined in file_icrar_leap-accelerate_common_SphericalDirection.h

Typedef Documentation

using `icrar::SphericalDirection` = Eigen::Vector2d

Typedef icrar::Tensor3X

- Defined in file_icrar_leap-accelerate_common_Tensor3X.h

Typedef Documentation

template<typename **T**>

using `icrar::Tensor3X` = Eigen::Tensor<*T*, 3>

Typedef thrust::complex

- Defined in file_icrar_leap-accelerate_math_cpu_eigen_extensions.h

Typedef Documentation

template<typename **Scalar**>

using thrust::**complex** = std::complex<*Scalar*>

CHAPTER

FOUR

COMPILING FROM SOURCE

leap-accelerate compilation is compatible with g++ and clang++ on debian or ubuntu. Support for macOS is currently experimental.

4.1 Binaries

4.1.1 Dependencies

Recommended Versions Compatibility

- g++ 9.3.0
- cuda 10.1
- boost 1.71.0
- casacore 3.1.2

Minimum Versions Compatibility

- g++ 6.3.0
- cuda 9.0
- boost 1.63.0
- cmake 3.15.1
- casacore 3.1.2

Ubuntu/Debian Dependencies

20.04 LTS

- sudo apt-get install gcc g++ gdb doxygen cmake casacore-dev clang-tidy-10 libboost1.71-all-dev libgsl-dev
- https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=200

18.04 LTS

- sudo apt-get install gcc g++ gdb doxygen cmake casacore-dev clang-tidy-10 libboost1.65-all-dev libgsl-dev
- https://developer.nvidia.com/cuda-10.1-download-archive-update2?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu

16.04 LTS

- https://askubuntu.com/questions/355565/how-do-i-install-the-latest-version-of-cmake-from-the-command-line
- sudo apt-get install gcc-6 g++-6 gdb doxygen casacore-dev libboost1.58-all-dev libgsl-dev
- https://developer.nvidia.com/cuda-92-download-archive?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_ve

CMake Options

Use `cmake .. -D<OPTION>=<VALUE> ...` or `ccmake ..` to set cmake options.

Setting an environment variable of the same name will also override these cmake options:

`CUDA_ENABLED` - Enables building with cuda support

`CMAKE_CUDA_ARCHITECTURES` - Selects the target cuda streaming multiprocessor and compute levels (default is all)

`WERROR` - Enables warnings as Errors

`WCONVERSION` - Enables warnings on implicit numeric conversions

`TRACE` - Traces data to the local directory

`CMAKE_RUN_CLANG_TIDY` - Enables running clang-tidy with the compiler

`USE_PCH` - Use pre-compile headers internally, if possible (defaults to `ON`)

Compile Commands

From the repository root folder run:

```
git submodule update --init --recursive
```

NOTE: pulling exernal submodules is now automated by CMake. When downloading the source files via tools other than git the folder `external/` will need to be copied manually.

GCC

Build Debug

```
cmake -B build/Debug -DCMAKE_BUILD_TYPE=Debug -DCMAKE_CXX_FLAGS_DEBUG="-g -O1"
-DCUDA_ENABLED=TRUE
```

With tracing to file:

```
cmake -B build/Debug -DCMAKE_BUILD_TYPE=Debug -DCMAKE_CXX_FLAGS_DEBUG="-g -O1"
-DCUDA_ENABLED=TRUE -DTRACE=ON
```

With gcovr analysis:

```
cmake -B build/Debug -DCMAKE_BUILD_TYPE=Debug -DCMAKE_CXX_FLAGS_DEBUG="-g -O1"
-DCMAKE_CXX_FLAGS="-coverage" -DCMAKE_EXE_LINKER_FLAGS="-coverage"
```

Build Release

```
cmake -B build/Release -DCMAKE_BUILD_TYPE=Release -DCUDA_ENABLED=TRUE
```

CUDA Hints

If cmake fails to detect CUDA try adding the following hint variables:

```
export CUDA_HOME=/usr/local/cuda
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${CUDA_HOME}/lib64:${CUDA_HOME}/extras/CUPTI/
↪lib64
export PATH=$PATH:$CUDA_HOME/bin
```

4.2 Testing

Testing provided via googletest. To test using the google test runner, test binaries can be executed directly using the following commands from the output folder:

`./bin/tests/LeapAccelerate.Tests ./bin/tests/LeapAccelerateCLI.Tests`

To test using CTest use the following command in build/linux:

`make test` or `ctest`

for verbose output use:

`ctest --verbose` or `ctest --output-on-failure`

4.3 Documentation

Generated documentation is available locally at the following file location:

`docs/sphinx/index.html`

Once deployed to a branch the docs will be available here:

https://developer.skao.int/projects/icrar-leap-accelerate/en/latest/

4.4 Test Coverage (Debug Only)

```
cmake ../.. -DCMAKE_BUILD_TYPE=Debug -DCMAKE_CXX_FLAGS_DEBUG="-g -O1"
-DCMAKE_CXX_FLAGS="-coverage" -DCMAKE_EXE_LINKER_FLAGS="-coverage"
```

```
make coverage
```

4.4.1 Building from Source

Doxygen documentation is generated for all C++ and cuda files with the following target:

```
make doxygen
```

Sphinx/Breath/Exhale docs is a dependent target generated with the following command:

```
make sphinx
```

4.4.2 gitlab repo

The CI/CD on gitlab used a pre-built base build image along with a cpp_build_base image to speed this process.

CHAPTER

FIVE

DOCKER IMAGE BUILD AND USAGE

The following procedure will generate a small docker image containing just the bare minimum binary and libraries to run LeapAccelerateCLI. Start with cloning out the repository:

```
git clone https://gitlab.com/ska-telescope/icrar-leap-accelerate.git
```

and then change into the directory:

```
cd icrar-leap-accelerate
```

All the following commands assume that you are in the root directory of the repository.

5.1 Docker image build

The Dockerfile builds the image from scratch, but that takes pretty long. Depending on the network connection this build can take a long time. It is downloading the CUDA tool chain which is about 2.7 GB. After the download the unpacking and installation takes significant time in addition.

```
docker build . --tag icrar/leap_cli:big
```

Typically, after the first build, subsequent builds are much faster.

5.1.1 Stripping the image

Due to the size of the CUDA tool chain the initial image created by the initial build is very large (~ 6GB). In order to strip this down to a reasonable size another step is recommended, which reduces the docker image size by more than a factor of 100.

In order to clean this up, it is highly recommended to run the tool from https://github.com/mvanholsteijn/strip-docker-image.git

```
cd .. ; git clone https://github.com/mvanholsteijn/strip-docker-image.git; cd icrar-leap-
↪accelerate
```

and then

```
../strip-docker-image/bin/strip-docker-image -i icrar/leap_cli:big -t icrar/leap_
↪cli:`cat version.txt` -f /usr/local/bin/LeapAccelerateCLI -f /bin/bash -f /usr/bin/cat␣
↪-f /usr/bin/ls -f /etc/passwd -f /home/ray
```

The resulting image is less than 50MB and contains just the required binary.

5.1.2 Testing the image

From the main directory of the leap_accelarate checkout run install.sh in the testdata directory:

```
cd testdata; ./install.sh; cd ..
```

and then in the main directory of leap_accelarate:

```
docker run -w /testdata --user ray -v "$(pwd)"/testdata:/testdata icrar/leap_cli:`cat␣
↪version.txt` LeapAccelerateCLI -f /testdata/mwa/1197638568-split.ms -i cpu -d "[[-0.
↪4606549305661674,-0.2971923379239251 3]]"
```

The output should be a JSON data structure.

You can also use a configuration file to run the same test run:

```
docker run -w /testdata --user ray -v "$(pwd)"/testdata:/testdata icrar/leap_cli:`cat␣
↪version.txt` LeapAccelerateCLI --config /testdata/mwa_test.json
```

In this case the output will be generated in a file called testdata/mwa_cal.out. NOTE: The tests above are using the CPU implementation of the algorithm.

5.2 Pulling and testing the docker image

NOTE: This part of the guide still assumes that you have cloned the repository and are located in the root directory of the repo, but that is only required if you want to run the test, else the docker pull is sufficient.

The leap-accelerate docker image is also available on dockerhub. Using that is very straight forward:

```
docker pull icrar/leap_cli:`cat version.txt`
```

The testing procedure is still the same. Download and unpack the testdata

```
cd testdata; ./install.sh; cd ..
```

and then execute the actual test:

```
docker run -w /testdata --user ray -v "$(pwd)"/testdata:/testdata icrar/leap_cli:`cat␣
↪version.txt` LeapAccelerateCLI -f /testdata/mwa/1197638568-split.ms -i cpu -d "[[-0.
↪4606549305661674,-0.2971923379239251 3]]"
```

CHAPTER

SIX

SUBMODULES

6.1 Pull Submodules

Needs to be performed once for any checked out branch:

```
git submodule update --init --recursive
```

6.2 Add Submodules

Example commands for how to add a new submodule:

```
git submodule add --name cmake-modules https://gitlab.com/ska-telescope/cmake-modules.
↪git external/cmake-modules
```

```
git submodule add --name gtest-1.11.0 https://github.com/google/googletest.git external/
↪gtest-1.11.0
```

```
git submodule add --name eigen-3.4.90 https://gitlab.com/libeigen/eigen.git external/
↪eigen-3.4.90
```

```
git submodule add --name rapidjson-1.1.0 https://github.com/Tencent/rapidjson.git␣
↪external/rapidjson-1.1.0
```

CHAPTER

SEVEN

CMAKE STYLE GUIDE

- See https://cliutils.gitlab.io/modern-cmake/

CHAPTER

EIGHT

C++ STYLE GUIDE

Refer to:

- https://github.com/ska-telescope/developer.skatelescope.org/blob/master/src/development/cplusplus-codeguide.rst

8.1 Extra

Refer to:

- https://github.com/isocpp/CppCoreGuidelines
- https://google.github.io/styleguide/cppguide.html

CHAPTER

NINE

CUDA/C++ STYLE GUIDE

9.1 File Structure

- All C++ headers (.h, .hpp) must be includable in sources built without cuda support
- Use C++ source files (.cc, .cpp) where possible for improved compilation speed
- Use Cuda source files (.cu) only for code blocks containing device code (at least 1 *_device_* or *_global_* definition)
- Use empty *_host_* and *_device_* definition guards in function headers to make them portable for builds without cuda support
- Do not declare *_global_* functions in C++ header or source file (.h, .hpp, .cc, .cpp)
- Declare kernel calling functions in C++ headers (.h, .hpp) and encapsulate pointers to device and host memory locations

9.2 Naming Conventions

- Use `g_` prefix for *_global_* functions to signify a cuda kernel entry point

CHAPTER

TEN

INDICES AND TABLES

- genindex
- search

INDEX

O

P

T