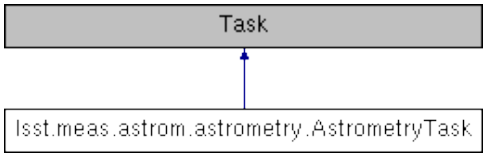
lsst.meas.astrom.astrometry.AstrometryTask Class Reference

Match an input source catalog with objects from a reference catalog and solve for the WCS. More...

Inheritance diagram for lsst.meas.astrom.astrometry.AstrometryTask:



Public Member Functions

def **__init__**
Construct an AstrometryTask. More...

def **run**
Load reference objects, match sources and optionally fit a WCS. More...

def **loadAndMatch**
Load reference objects overlapping an exposure and match to sources detected on that exposure. More...

def **solve**
Load reference objects overlapping an exposure, match to sources and fit a WCS. More...

Public Attributes

refObjLoader

Static Public Attributes

ConfigClass = **AstrometryConfig**

Private Member Functions

def **_computeMatchStatsOnSky**

def **_getExposureMetadata**
Extract metadata from an exposure. More...

def **_matchAndFitWcs**
Match sources to reference objects and fit a WCS. More...

Static Private Attributes

string **_DefaultName** = "astrometricSolver"

Detailed Description

Match an input source catalog with objects from a reference catalog and solve for the WCS.

Contents

Description

Match input sourceCat with a reference catalog and solve for the Wcs

There are three steps, each performed by different subtasks:

- Find position reference stars that overlap the exposure
- Match sourceCat to position reference stars
- Fit a WCS based on the matches

Task initialisation

Construct an AstrometryTask.

Parameters

`[in]` **refObjLoader** A reference object loader object

`[in]` **schema** ignored; available for compatibility with an older astrometry task

`[in]` **kwargs** additional keyword arguments for pipe_base Task.__init__

Invoking the Task

Load reference objects, match sources and optionally fit a WCS. This is a thin layer around solve or loadAndMatch, depending on config.forceKnownWcs

Parameters

`[in,out]` **exposure** exposure whose WCS is to be fit The following are read only:

- bbox
- calib (may be absent)
- filter (may be unset)
- detector (if wcs is pure tangent; may be absent) The following are updated:
- wcs (the initial value is used as an initial guess, and is required)

`[in]` **sourceCat** catalog of sources detected on the exposure (an lsst.afw.table.SourceCatalog)

Returns

an lsst.pipe.base.Struct with these fields:

- refCat reference object catalog of objects that overlap the exposure (with some margin) (an lsst::afw::table::SimpleCatalog)
- matches list of reference object/source matches (an lsst.afw.table.ReferenceMatchVector)
- scatterOnSky median on-sky separation between reference objects and sources in "matches" (an lsst.afw.geom.Angle), or None if config.forceKnownWcs True

- matchMeta metadata needed to unpersist matches (an lsst.daf.base.PropertyList)

Load reference objects overlapping an exposure and match to sources detected on that exposure.

Parameters

[in] **exposure** exposure that the sources overlap

[in] **sourceCat** catalog of sources detected on the exposure (an lsst.afw.table.SourceCatalog)

Returns

an lsst.pipe.base.Struct with these fields:

- refCat reference object catalog of objects that overlap the exposure (with some margin) (an lsst::afw::table::SimpleCatalog)
- matches list of reference object/source matches (an lsst.afw.table.ReferenceMatchVector)
- matchMeta metadata needed to unpersist matches (an lsst.daf.base.PropertyList)

Note

ignores config.forceKnownWcs, config.maxIter, config.matchDistanceSigma and config.minMatchDistanceArcSec

Configuration parameters

See **AstrometryConfig**

A complete example of using AstrometryTask

See meas_photocal_photocal_Example.

Debug variables

The **command line task** interface supports a flag `-d` to import **debug.py** from your `PYTHONPATH`; see **Using lsstDebug to control debugging output** for more about **debug.py** files.

The available variables in AstrometryTask are:

`display` (bool)

If True display information at three stages: after finding reference objects, after matching sources to reference objects, and after fitting the WCS; defaults to False

`frame` (int)

ds9 frame to use to display the reference objects; the next two frames are used to display the match list and the results of the final WCS; defaults to 0

To investigate the **Debug variables**, put something like

```
import lsstDebug
def DebugInfo(name):
    debug = lsstDebug.getInfo(name)        # N.b. lsstDebug.Info(name) would call us recursively
    if name == "lsst.meas.astrom.astrometry":
        debug.display = True

    return debug

lsstDebug.Info = DebugInfo
```

into your debug.py file and run this task with the `–debug` flag.

Definition at line **83** of file **astrometry.py**.

Constructor & Destructor Documentation

def lsst.meas.astrom.astrometry.AstrometryTask.__init__ (self, refObjLoader, schema = None, kwargs)

Construct an AstrometryTask.

Parameters

[in] **refObjLoader** A reference object loader object

[in] **schema** ignored; available for compatibility with an older astrometry task

[in] **kwargs** additional keyword arguments for pipe_base Task.__init__

Definition at line **157** of file **astrometry.py**.

```
157
158     def __init__(self, refObjLoader, schema=None, **kwargs):
159         """!Construct an AstrometryTask
160
161         @param[in] refObjLoader A reference object loader object
162         @param[in] schema  ignored; available for compatibility with an older astrometry task
163         @param[in] kwargs  additional keyword arguments for pipe_base Task.\_\_init\_\_
164         """
165         pipeBase.Task.__init__(self, **kwargs)
166         self.refObjLoader = refObjLoader
167         self.makeSubtask("matcher")
168         self.makeSubtask("wcsFitter")
```

Member Function Documentation

def lsst.meas.astrom.astrometry.AstrometryTask._computeMatchStatsOnSky (self, matchList) private

```
Compute on-sky radial distance statistics for a match list

@param[in] matchList  list of matches between reference object and sources;
    the distance field is the only field read and it must be set to distance in radians

@return a pipe_base Struct containing these fields:
- distMean  clipped mean of on-sky radial separation
- distStdDev  clipped standard deviation of on-sky radial separation
- maxMatchDist  distMean + self.config.matchDistanceSigma*distStdDev
```

Definition at line 359 of file astrometry.py.

```
359
360     def _computeMatchStatsOnSky(self, matchList):
361         """Compute on-sky radial distance statistics for a match list
362
363         @param[in] matchList  list of matches between reference object and sources;
364             the distance field is the only field read and it must be set to distance in radians
365
366         @return a pipe_base Struct containing these fields:
367         - distMean  clipped mean of on-sky radial separation
368         - distStdDev  clipped standard deviation of on-sky radial separation
369         - maxMatchDist  distMean + self.config.matchDistanceSigma*distStdDev
370         """
371         distStatsInRadians = makeMatchStatistics(matchList, afwMath.MEANCLIP |
    afwMath.STDEVCLIP)
372         distMean = distStatsInRadians.getValue(afwMath.MEANCLIP)*afwGeom.radians
373         distStdDev = distStatsInRadians.getValue(afwMath.STDEVCLIP)*afwGeom.radians
374         return pipeBase.Struct(
375             distMean=distMean,
376             distStdDev=distStdDev,
377             maxMatchDist=distMean + self.config.matchDistanceSigma*distStdDev,
378         )
```

def lsst.meas.astrom.astrometry.AstrometryTask._getExposureMetadata (self, exposure) private

Extract metadata from an exposure.

Returns

an lsst.pipe.base.Struct containing the following exposure metadata:

- bbox: parent bounding box
- wcs: WCS (an lsst.afw.image.Wcs)
- calib calibration (an lsst.afw.image.Calib), or None if unknown
- filterName: name of filter, or None if unknown

Definition at line 379 of file astrometry.py.

```
379
380     def _getExposureMetadata(self, exposure):
381         """!Extract metadata from an exposure
382
383         @return an lsst.pipe.base.Struct containing the following exposure metadata:
384         - bbox: parent bounding box
385         - wcs: WCS (an lsst.afw.image.Wcs)
386         - calib calibration (an lsst.afw.image.Calib), or None if unknown
387         - filterName: name of filter, or None if unknown
388         """
389         exposureInfo = exposure.getInfo()
390         filterName = exposureInfo.getFilter().getName() or None
391         if filterName == "_unknown_":
392             filterName = None
393         return pipeBase.Struct(
394             bbox=exposure.getBBox(),
395             wcs=getDistortedWcs(exposureInfo, log=self.log),
396             calib=exposureInfo.getCalib() if exposureInfo.hasCalib() else None,
397             filterName=filterName,
398         )
```

def lsst.meas.astrom.astrometry.AstrometryTask._matchAndFitWcs (self, refCat, sourceCat, refFluxField, bbox, wcs, maxMatchDist = None, exposure = None) private

Match sources to reference objects and fit a WCS.

Parameters

[in]	**refCat**	catalog of reference objects
[in]	**sourceCat**	catalog of sources detected on the exposure (an lsst.afw.table.SourceCatalog)
[in]	**refFluxField**	field of refCat to use for flux
[in]	**bbox**	bounding box of exposure (an lsst.afw.geom.Box2I)

[in] **wcs** initial guess for WCS of exposure (an lsst.afw.image.Wcs)

[in] **maxMatchDist** maximum on-sky distance between reference objects and sources (an lsst.afw.geom.Angle); if None then use the matcher's default

[in] **exposure** exposure whose WCS is to be fit, or None; used only for the debug display

Returns

an lsst.pipe.base.Struct with these fields:

- matches list of reference object/source matches (an lsst.afw.table.ReferenceMatchVector)
- wcs the fit WCS (an lsst.afw.image.Wcs)
- scatterOnSky median on-sky separation between reference objects and sources in "matches" (an lsst.afw.geom.Angle)

Definition at line **401** of file **astrometry.py**.

```
401
402                         exposure=None):
403         """!Match sources to reference objects and fit a WCS
404
405         @param[in] refCat  catalog of reference objects
406         @param[in] sourceCat  catalog of sources detected on the exposure (an
    lsst.afw.table.SourceCatalog)
407         @param[in] refFluxField  field of refCat to use for flux
408         @param[in] bbox  bounding box of exposure (an lsst.afw.geom.Box2I)
409         @param[in] wcs  initial guess for WCS of exposure (an lsst.afw.image.Wcs)
410         @param[in] maxMatchDist  maximum on-sky distance between reference objects and sources
411             (an lsst.afw.geom.Angle); if None then use the matcher's default
412         @param[in] exposure  exposure whose WCS is to be fit, or None; used only for the debug
    display
413
414         @return an lsst.pipe.base.Struct with these fields:
415         - matches  list of reference object/source matches (an
    lsst.afw.table.ReferenceMatchVector)
416         - wcs  the fit WCS (an lsst.afw.image.Wcs)
417         - scatterOnSky  median on-sky separation between reference objects and sources in
    "matches"
418             (an lsst.afw.geom.Angle)
419         """
420         import lsstDebug
421         debug = lsstDebug.Info(__name__)
422         matchRes = self.matcher.matchObjectsToSources(
423             refCat=refCat,
424             sourceCat=sourceCat,
425             wcs=wcs,
426             refFluxField=refFluxField,
427             maxMatchDist=maxMatchDist,
428         )
429         self.log.logdebug("Found %s matches" % (len(matchRes.matches),))
430         if debug.display:
431             frame = int(debug.frame)
432             displayAstrometry(
433                 refCat=refCat,
434                 sourceCat=matchRes.usableSourceCat,
435                 matches=matchRes.matches,
436                 exposure=exposure,
437                 bbox=bbox,
438                 frame=frame + 1,
439                 title="Initial WCS",
440             )
441
442         self.log.logdebug("Fitting WCS")
443         fitRes = self.wcsFitter.fitWcs(
444             matches=matchRes.matches,
445             initWcs=wcs,
446             bbox=bbox,
447             refCat=refCat,
448             sourceCat=sourceCat,
449         )
450         fitWcs = fitRes.wcs
451         scatterOnSky = fitRes.scatterOnSky
452         if debug.display:
453             frame = int(debug.frame)
```

```
454             displayAstrometry(
455                 refCat=refCat,
456                 sourceCat=matchRes.usableSourceCat,
457                 matches=matchRes.matches,
458                 exposure=exposure,
459                 bbox=bbox,
460                 frame=frame + 2,
461                 title="Fit TAN-SIP WCS",
462             )
463 
464         return pipeBase.Struct(
465             matches=matchRes.matches,
466             wcs=fitWcs,
467             scatterOnSky=scatterOnSky,
468         )
```

def lsst.meas.astrom.astrometry.AstrometryTask.loadAndMatch (self, exposure, sourceCat)

Load reference objects overlapping an exposure and match to sources detected on that exposure.

Parameters

[in] **exposure** exposure that the sources overlap

[in] **sourceCat** catalog of sources detected on the exposure (an lsst.afw.table.SourceCatalog)

Returns

an lsst.pipe.base.Struct with these fields:

- refCat reference object catalog of objects that overlap the exposure (with some margin) (an lsst::afw::table::SimpleCatalog)
- matches list of reference object/source matches (an lsst.afw.table.ReferenceMatchVector)
- matchMeta metadata needed to unpersist matches (an lsst.daf.base.PropertyList)

Note

ignores config.forceKnownWcs, config.maxIter, config.matchDistanceSigma and config.minMatchDistanceArcSec

Definition at line **200** of file **astrometry.py**.

```
200 
201     def loadAndMatch(self, exposure, sourceCat):
202         """!Load reference objects overlapping an exposure and match to sources detected on that
    exposure
203 
204         @param[in] exposure  exposure that the sources overlap
205         @param[in] sourceCat  catalog of sources detected on the exposure (an
    lsst.afw.table.SourceCatalog)
206 
207         @return an lsst.pipe.base.Struct with these fields:
208         - refCat  reference object catalog of objects that overlap the exposure (with some
    margin)
209             (an lsst::afw::table::SimpleCatalog)
210         - matches  list of reference object/source matches (an
    lsst.afw.table.ReferenceMatchVector)
211         - matchMeta  metadata needed to unpersist matches (an lsst.daf.base.PropertyList)
212 
213         @note ignores config.forceKnownWcs, config.maxIter, config.matchDistanceSigma
214             and config.minMatchDistanceArcSec
215         """
216         import lsstDebug
217         debug = lsstDebug.Info(__name__)
218 
219         matchMeta = createMatchMetadata(exposure, border=self.refObjLoader.config.pixelMargin)
220         expMd = self. getExposureMetadata(exposure)
```

```
221
222         loadRes = self.refObjLoader.loadPixelBox(
223             bbox=expMd.bbox,
224             wcs=expMd.wcs,
225             filterName=expMd.filterName,
226             calib=expMd.calib,
227         )
228
229         matchRes = self.matcher.matchObjectsToSources(
230             refCat=loadRes.refCat,
231             sourceCat=sourceCat,
232             wcs=expMd.wcs,
233             refFluxField=loadRes.fluxField,
234             maxMatchDist=None,
235         )
236
237         distStats = self._computeMatchStatsOnSky(matchRes.matches)
238         self.log.info(
239             "Found %d matches with scatter = %0.3f +- %0.3f arcsec; " %
240             (len(matchRes.matches), distStats.distMean.asArcseconds(),
 distStats.distStdDev.asArcseconds())
241         )
242
243         if debug.display:
244             frame = int(debug.frame)
245             displayAstrometry(
246                 refCat=loadRes.refCat,
247                 sourceCat=sourceCat,
248                 matches=matchRes.matches,
249                 exposure=exposure,
250                 bbox=expMd.bbox,
251                 frame=frame,
252                 title="Matches",
253             )
254
255         return pipeBase.Struct(
256             refCat=loadRes.refCat,
257             matches=matchRes.matches,
258             matchMeta=matchMeta,
259         )
```

def lsst.meas.astrom.astrometry.AstrometryTask.run (self, exposure, sourceCat)

Load reference objects, match sources and optionally fit a WCS.

This is a thin layer around solve or loadAndMatch, depending on config.forceKnownWcs

Parameters

`[in,out]` **exposure** exposure whose WCS is to be fit The following are read only:

- bbox
- calib (may be absent)
- filter (may be unset)
- detector (if wcs is pure tangent; may be absent) The following are updated:
- wcs (the initial value is used as an initial guess, and is required)

`[in]` **sourceCat** catalog of sources detected on the exposure (an lsst.afw.table.SourceCatalog)

Returns

an lsst.pipe.base.Struct with these fields:

- refCat reference object catalog of objects that overlap the exposure (with some margin) (an

lsst::afw::table::SimpleCatalog)

- matches list of reference object/source matches (an lsst.afw.table.ReferenceMatchVector)
- scatterOnSky median on-sky separation between reference objects and sources in "matches" (an lsst.afw.geom.Angle), or None if config.forceKnownWcs True
- matchMeta metadata needed to unpersist matches (an lsst.daf.base.PropertyList)

Definition at line **170** of file **astrometry.py**.

```
170
171     def run(self, exposure, sourceCat):
172         """!Load reference objects, match sources and optionally fit a WCS
173
174         This is a thin layer around solve or loadAndMatch, depending on config.forceKnownWcs
175
176         @param[in,out] exposure  exposure whose WCS is to be fit
177             The following are read only:
178             - bbox
179             - calib (may be absent)
180             - filter (may be unset)
181             - detector (if wcs is pure tangent; may be absent)
182             The following are updated:
183             - wcs (the initial value is used as an initial guess, and is required)
184         @param[in] sourceCat  catalog of sources detected on the exposure (an
    lsst.afw.table.SourceCatalog)
185         @return an lsst.pipe.base.Struct with these fields:
186         - refCat  reference object catalog of objects that overlap the exposure (with some
    margin)
187             (an lsst::afw::table::SimpleCatalog)
188         - matches  list of reference object/source matches (an
    lsst.afw.table.ReferenceMatchVector)
189         - scatterOnSky  median on-sky separation between reference objects and sources in
    "matches"
190             (an lsst.afw.geom.Angle), or None if config.forceKnownWcs True
191         - matchMeta  metadata needed to unpersist matches (an lsst.daf.base.PropertyList)
192         """
193         if self.config.forceKnownWcs:
194             res = self.loadAndMatch(exposure=exposure, sourceCat=sourceCat)
195             res.scatterOnSky = None
196         else:
197             res = self.solve(exposure=exposure, sourceCat=sourceCat)
198         return res
```

def lsst.meas.astrom.astrometry.AstrometryTask.solve (self, exposure, sourceCat)

Load reference objects overlapping an exposure, match to sources and fit a WCS.

Returns

an lsst.pipe.base.Struct with these fields:

- refCat reference object catalog of objects that overlap the exposure (with some margin) (an lsst::afw::table::SimpleCatalog)
- matches list of reference object/source matches (an lsst.afw.table.ReferenceMatchVector)
- scatterOnSky median on-sky separation between reference objects and sources in "matches" (an lsst.afw.geom.Angle)
- matchMeta metadata needed to unpersist matches (an lsst.daf.base.PropertyList)

Note

ignores config.forceKnownWcs

Definition at line 261 of file astrometry.py.

```
261
262     def solve(self, exposure, sourceCat):
263         """!Load reference objects overlapping an exposure, match to sources and fit a WCS
264
265         @return an lsst.pipe.base.Struct with these fields:
266         - refCat  reference object catalog of objects that overlap the exposure (with some
    margin)
267             (an lsst::afw::table::SimpleCatalog)
268         - matches  list of reference object/source matches (an
    lsst.afw.table.ReferenceMatchVector)
269         - scatterOnSky  median on-sky separation between reference objects and sources in
    "matches"
270             (an lsst.afw.geom.Angle)
271         - matchMeta  metadata needed to unpersist matches (an lsst.daf.base.PropertyList)
272
273         @note ignores config.forceKnownWcs
274         """
275         import lsstDebug
276         debug = lsstDebug.Info(__name__)
277
278         matchMeta = createMatchMetadata(exposure, border=self.refObjLoader.config.pixelMargin)
279         expMd = self._getExposureMetadata(exposure)
280
281         loadRes = self.refObjLoader.loadPixelBox(
282             bbox=expMd.bbox,
283             wcs=expMd.wcs,
284             filterName=expMd.filterName,
285             calib=expMd.calib,
286         )
287         if debug.display:
288             frame = int(debug.frame)
289             displayAstrometry(
290                 refCat=loadRes.refCat,
291                 sourceCat=sourceCat,
292                 exposure=exposure,
293                 bbox=expMd.bbox,
294                 frame=frame,
295                 title="Reference catalog",
296             )
297
298         res = None
299         wcs = expMd.wcs
300         maxMatchDist = None
301         for i in range(self.config.maxIter):
302             iterNum = i + 1
303             try:
304                 tryRes = self._matchAndFitWcs(  # refCat, sourceCat, refFluxField, bbox, wcs,
    exposure=None
305                     refCat=loadRes.refCat,
306                     sourceCat=sourceCat,
307                     refFluxField=loadRes.fluxField,
308                     bbox=expMd.bbox,
309                     wcs=wcs,
310                     exposure=exposure,
311                     maxMatchDist=maxMatchDist,
312                 )
313             except Exception as e:
314                 # if we have had a succeessful iteration then use that; otherwise fail
315                 if i > 0:
316                     self.log.info("Fit WCS iter %d failed; using previous iteration: %s" %
    (iterNum, e))
317                     iterNum -= 1
318                     break
319                 else:
320                     raise
321
322             tryMatchDist = self._computeMatchStatsOnSky(tryRes.matches)
323             self.log.logdebug(
324                 "Match and fit WCS iteration %d: found %d matches with scatter = %0.3f +- %0.3f
    arcsec; "
325                 "max match distance = %0.3f arcsec" %
326                 (iterNum, len(tryRes.matches), tryMatchDist.distMean.asArcseconds(),
327                  tryMatchDist.distStdDev.asArcseconds(),
    tryMatchDist.maxMatchDist.asArcseconds()))
328             if maxMatchDist is not None:
329                 if tryMatchDist.maxMatchDist >= maxMatchDist:
330                     self.log.logdebug(
331                         "Iteration %d had no better maxMatchDist; using previous iteration" %
```

```
    (iterNum,))
332                     iterNum -= 1
333                     break
334
335             maxMatchDist = tryMatchDist.maxMatchDist
336             res = tryRes
337             wcs = res.wcs
338             if tryMatchDist.maxMatchDist.asArcseconds() < self.config.minMatchDistanceArcSec:
339                 self.log.logdebug(
340                     "Max match distance = %0.3f arcsec < %0.3f = config.minMatchDistanceArcSec;
    "
341                     "that's good enough" %
342                     (tryMatchDist.maxMatchDist.asArcseconds(),
    self.config.minMatchDistanceArcSec))
343                 break
344
345         self.log.info(
346             "Matched and fit WCS in %d iterations; "
347             "found %d matches with scatter = %0.3f +- %0.3f arcsec" %
348             (iterNum, len(tryRes.matches), tryMatchDist.distMean.asArcseconds(),
349                 tryMatchDist.distStdDev.asArcseconds()))
350
351         exposure.setWcs(res.wcs)
352
353         return pipeBase.Struct(
354             refCat=loadRes.refCat,
355             matches=res.matches,
356             scatterOnSky=res.scatterOnSky,
357             matchMeta=matchMeta,
358         )
```

Member Data Documentation

string lsst.meas.astrom.astrometry.AstrometryTask._DefaultName = "astrometricSolver" static private

Definition at line 155 of file astrometry.py.

lsst.meas.astrom.astrometry.AstrometryTask.ConfigClass = AstrometryConfig static

Definition at line 154 of file astrometry.py.

lsst.meas.astrom.astrometry.AstrometryTask.refObjLoader

Definition at line 165 of file astrometry.py.

The documentation for this class was generated from the following file:

- /home/lsstsw/stack/Linux64/meas_astrom/12.0-7-ge3f9808+1/python/lsst/meas/astrom/astrometry.py