

QSO Guide Star Selection Tool

Design Document

Joshua Shapiro

Renaud Savalle

QSO-019/Version 1.1/2002-10-29

1 Introduction

The limited field of the Megaprime guider increases the chance that a blind pointing will not yield any guide stars. To prevent this from occurring, the QSO project intends to identify fields without guide stars for every pointing accepted via PH2 and coordinate a change in target position with the PI. In addition, the potentially large offsets in the dither patterns allow the possibility that the selected guide stars will not be available for all pointings. This software can also attempt to identify the best choice of guide stars to span an entire dither pattern. The following document intends to outline the abilities and scope of QSO software that will be developed to do this task.

2 Requirements

At a minimum, the software must identify pointings with unacceptable guide stars prior to observations and warn the user(s) of the QSO Tools. It also may be necessary to save guide star coordinates for every anticipated pointing so that the coordinates may be passed to TCS via NEO at the time of observation.

According to “Megaprime and the CFHT Observing Environment v2.0” the two guiders will be sensitive to guide stars up to magnitude 14 in clear sky conditions. Fainter stars might be guided on with no tip/tilt correction.

From conversations with Jim Thomas, the plan is to use the brighter of the stars in the two guiders to offset the telescope. Acquiring a new guide star if the old guide star is lost due to a slew or offset is expected to consume the same amount of time as setting up on a guide star from scratch. By identifying the best arrangement

of guide stars in a dithering pattern prior to observations, we can save considerable time. This means the search algorithm should attempt to locate the brightest star which overlaps the most pointings within a dither pattern.

3 Megacam Guider Geometry

There are two guiders. The field for each guide probe is defined at the telescope's focal plane as a segment of $90mm$ wide, extending from a line $162mm(37.1arcmin)$ off center to a field bounded by an arc $193mm(44.0arcmin)$ in radius.

Each guider will have a 96% probability of finding a suitably bright guide star (according to the Megacam design documents). Random queries of both USNO and GSC for stars in a $20arcmin$ box under 14mag yield from 20-30 stars. The guider field is approximately $20'$ from East to West and $7'$ from North to South at its highest point and thus approximately 35% the size of a $20'$ square box, one would expect to find $\sim 7 - 10$ stars in the guider field per pointing.

The gaps between chips on the megacam mosaic are reportedly $11 \pm 3''$ in the North-South direction and $14 \pm 3''$ in the East West direction, but the connector gaps will be $85 \pm 16''$. While the dither pattern geometry has yet to be determined, I predict the largest offsets we can expect would be about $1'$ North South and $20''$ East West. A plot of the guider field with the stars projected to the focal plane can be seen in figure 1. Each star has been projected five times assuming a different position of the telescope. The telescope positions differed by $1'$ from center in the vertical direction and $40''$ in the horizontal direction. It is immediately obvious that three of the several stars would be suitable for guiding during the entire dither pattern.

4 Computational Analysis

This section is devoted to analyzing the number of possible events that will have to take place to implement this application, and aid in making decisions on the appropriate algorithms and data structures to use. This memo will frequently refer to the following terms.

Pointing Set A set of pointings which are part of the same dither pattern.

Pointing An individual pointing within a dither pattern.

4.1 Number of targets

There are 875 Pointing Sets for 2002b, defined as $(\alpha, \delta, \Delta\alpha, \Delta\delta, DP) = \{\text{fixed target coordinates, offset, dither pattern}\}$, as calculated by the query below. The number of actual pointings including offsets within DPs is 1819. The query does not query by OB, because often times different OBs will use the same pointing parameters and only differ in the filter and exposure times. If the query is performed by selecting the pointing parameters for every OB, we compute, 1177 pointing sets and 2523 actual pointings.

```
select distinct t.reg_ra, t.reg_dec, pointing.offset_ra, pointing.offset_dec ,
pattern.id as pat_id, pattern.name as pat_name, pattern.nbexp
into #points
from target_f t, target, pointing, icseq, ic, pattern, ob, obseq, ogroup, prg, ptype
where t.id = target.id
and target.poi_id = pointing.id
and ob.tar_id = t.id
and icseq.ob_id = ob.id
and icseq.ic_id = ic.id
and ic.pat_id = pattern.id
and obseq.ob_id = ob.id
and obseq.ogr_id = ogroup.id
and ogroup.prg_id = prg.id
and prg.ptype_id = ptype.id
and ptype.value like 'REGULAR'
and prg.runid like '02B%'
and target.is_deleted = 0
and ob.is_deleted = 0
and ogroup.is_deleted = 0
and obseq.is_deleted = 0
and prg.is_deleted = 0
and ic.is_deleted = 0
select count(*) as pointing_sets, sum(nbexp) as pointings from #points
drop table #points

pointing_sets pointings
-----
875          1819
```

A star catalog query will need to be performed at each position for each guider. This means approximately 1500 – 2000 catalog queries must be executed per semester.

4.2 Determining if the guide star is visible

Another computation that will be performed is the determination of whether or not a star lies inside a guider's Field of View. There are many algorithms which can perform this function. The approximate number of stars to operate on will be the number of stars returned from the catalog query, $\sim 20 - 30$ normally. The guider FOVs have three straight sides and one side which is an arc. The algorithm chosen, should be optimized for such parameters. Most algorithms for searching geometric areas, first insert the data in a structure optimized for a particular type of search. For small data sets, this insertion can be more intensive than the actual

search procedure. The simplest search would iterate through the data set linearly and perform the following comparison on each projected coordinate.

```

North Guider
(x < guider.west &&
 x > guider.east &&
 y > guider.south &&
 y < guider.north[x])

```

```

South Guider
(x < guider.west &&
 x > guider.east &&
 y < guider.north &&
 y > guider.south[x])

```

The last comparison in each block accounts for the arc of the guider. A one time computation could be performed of the y value of the guider boundary at each x and the resulting data stored in an array.

This software component which searches the guider field is instrument specific. As such it should be modular and easily expandable to other instruments. In the flow chart, this component is depicted as `Instrument.withinGuider()`, a method which called for each guide star to determine whether or not it lies in the guider field of view.

For megacam, a simple polynomial has been provided by G. Barrick for projecting from sky coordinates (α, δ) to virtual coordinates (x, y) , and Wei Da, has implemented this in C¹. Each star returned from the star catalog query could be projected to millimeters and then determined to be in or out of the guider field using a method such as the one described in section 4.2. The disadvantage of this method is that the co-efficients to the polynomial terms are not yet well determined. A good approximation exists, but they will definitely be modified at a future date. If the co-efficients were to be hard-coded, any changes would require a recompile. The co-efficients could be stored in the database, on the status server, or in a

1

$$\begin{aligned}
 X &= -.00050567 + 259.62288741x + 1.42817388xy + 2.91590568xy^2 \\
 &\quad + 5.26773042x^3 + 2.44741455xy^3 + 1.81935434x^3y \\
 Y &= 4.46716547 + 256.56236754y + 0.50555197x^2 + 8.98197331y^2 + 3.66931979x^2y \\
 &\quad - 5.23324385y^3 + 2.99495472x^2y^2 + 0.50027975x^4 + 5.72397767y^4 \\
 x, y &\in \text{Degrees} \\
 X, Y &\in \text{millimeters}
 \end{aligned}$$

par-file but changes computed by technical staff would need to be communicated to software staff requiring some protocol for this communication. Alternatively, elixir could perform astrometry on the guider field, compute the WCS and update the co-efficients on a regular basis.

A second option assumes the guider field will always assume a well defined region in sky coordinates relative to the center of the field. Stars can be queried from the star catalogs using this region as the search area so that no projection is required. The region of sky coordinates consumed by the guider field will certainly be a non-regular area with curved boundaries, but subsequent simulations of the field of view have shown the distortion to be minimal. This approach, however, could mislead the software into thinking it needs to switch guide stars after an offset, when in fact the original guide star is still available at the edge of the field. This method eliminates the need to project each star at the expense of searching a more complicated geometric shape but, like the first method, is susceptible to changes in the instrument position on the telescope.

4.3 Choosing the best guide star

The most complicated problem is to determine if a guide star is the best choice given subsequent offsets in the dp. If the problem is considered in terms of the time it takes an OA to setup on the guide star, then the problem reduces to a shortest-path problem through a graph whose vertices are the available guide stars for each pointing and whose edges are weighted by the time it would take to acquire that guide star. If an edge connects two identical guide stars, its length would be shorter than an edge connecting two different guide stars. The shortest path through such a graph would identify the vertices, or guide stars, to use for each step in the dither pattern. A sample graph is drawn in figure 2

Graph searching algorithms have been well studied. The algorithm and data structure used depend critically on the density, or number of edges, in the graph. In general algorithms either work in V^2 time for dense graphs or $V + E$ time for sparse graphs where V is the number of vertices and E is the number of edges. The problem is further complicated by the fact that there are two guiders, and the guide star chosen in one guider can depend on the star chosen in the opposite guider. The brighter of the two stars will be used for the next offset, so the algorithm should carefully consider which guide stars to pass to TCS to ensure optimal guiding. These rules need to be well defined before they can be implemented. However, a basic implementation will choose the brightest star closest to the center of each guider.

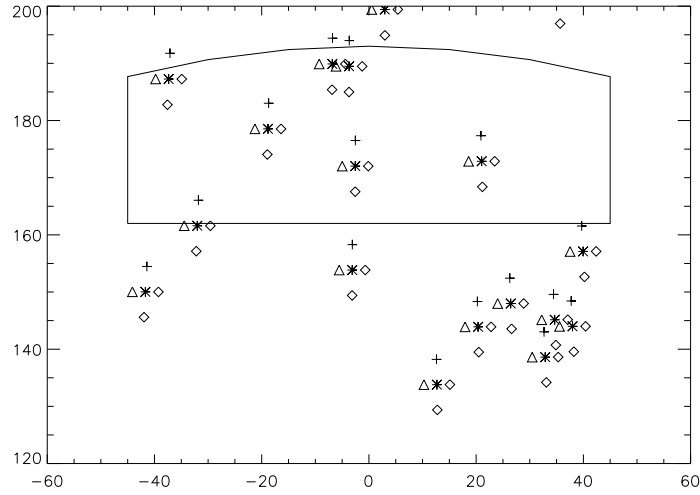


Figure 1: Anticipated position of stars on Megacam GSFU focal plane. Offsets are $\pm 40'' EW, \pm 60'' NS$

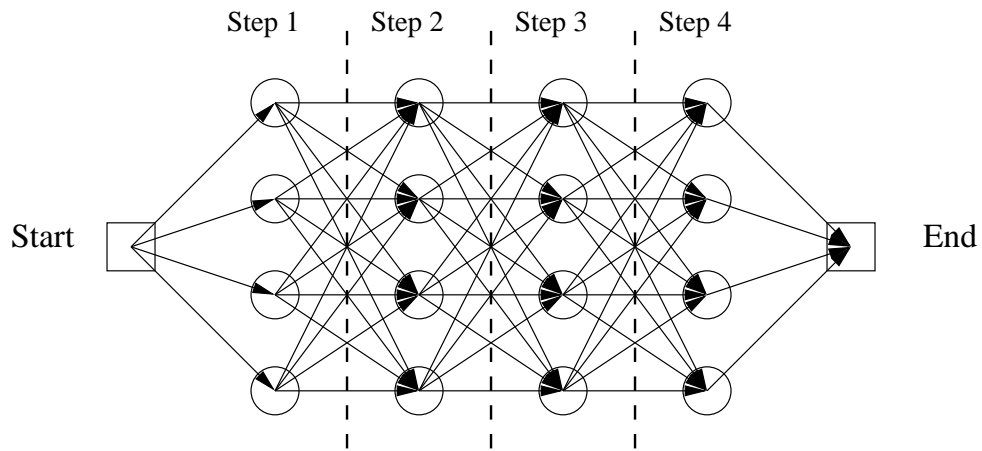


Figure 2: An example of a graph used to identify the best guide stars in a dither pattern. Each DP step is a new column, Each vertex is a guide star, and each edge represents a possible path from a guide star in one offset to a guide star in the next offset.

5 Flow Chart

A flowchart outlining two major modes of ~~CSST~~ ~~CSST~~ can be found in figure 3. The first mode depicts the guide star search loop.

1. A collection of pointing sets is built with a database query.
2. Iterate through each Pointing Set
 - (a) Query star catalog for stars in region of pointing set
 - (b) Iterate through pointings within Pointing Set
 - i. Query PH2 for guide stars at Pointing
 - ii. IF (Guide Stars Found)
THEN set pointing status to GOOD ; Next pointing
 - iii. ELSE Filter star collection to see what lies within instrument guider region at pointing
 - iv. IF the resulting collection is empty
THEN set the Pointing status to "Bad" ; Next Pointing
 - v. Pass the resulting collection, along with the current pointing to a class responsible for choosing the best guide stars out of the legitimate ones.
 - vi. Next Pointing
 - (c) Choose best guide stars for PointingSet
 - (d) Push selected guide stars into database
3. Next PointingSet

The second major mode will be where the user interacts with the GUI to fine tune the guide star selection. This will be most useful for browsing the PointingSets containing "Bad" pointings and choosing new guide stars for them.

1. Begin with a collection of Pointing Sets
2. User selects a Pointing Set
3. Query Star Catalog for stars in vicinity of Pointing Set
4. User selects individual pointing from Set
5. Query PH2 to find any guide stars already inserted for pointing
6. Render Star Catalog stars and Selected stars on screen.
7. User can select a star
8. Validate star (within guider - acceptable magnitude etc.)
9. Add to selected collection and re-render
10. Update PH2 and PointingSet/Pointing State.

11. repeat.

Renaud has proposed using Aladin for the display portion of *GAUSTAF*. Aladin offers the display features and flexibility needed by *GAUSTAF*, as outlined in section 8. Using Aladin would move the responsibility for querying the catalogs and database and all rendering operations from *GAUSTAF* to Aladin. *GAUSTAF* would then only be responsible for sending Aladin the target coordinates and appropriate IDs through the AladinStub class. Updates performed, such as selecting or de-selecting a guide star, from within Aladin, will only be reflected in the GUI if a reload button is clicked.

6 Guide Star Catalog

The user should be able to select the star catalog to query at the time of the search. An interface to multiple catalogs can be created so that the actual catalog in use is not relevant to the application.

Implementation of the star catalogs is then flexible and can be done locally at CFHT using existing star catalogs or over the internet using star catalogs at other sites.

A simple API for an interface would have a single function for querying and a single function to specify the catalog to use.

Tests performed querying the CDS vizier database look very promising. The query is very easy to implement and perform and returns in sub-second times.

7 Database Population and Retrieval

To fully implement *GAUSTAF* will require database stored procedures and table modifications or creation.

The application can query the database to build a set of tables to lookup the offset sequence for dither patterns and use these tables when calculating the actual pointings.

The following queries must be supported for the chain to work.

- insert guide star given Target, DP, and offset
- delete guide star given Target, DP, offset, and Guide Star coordinates
- select guide stars given Target & Offset

Much caution should be used when implementing these tables and procedures in the database. Any change to the target position or target offsets will destroy the integrity of any guide star coordinates in the database.

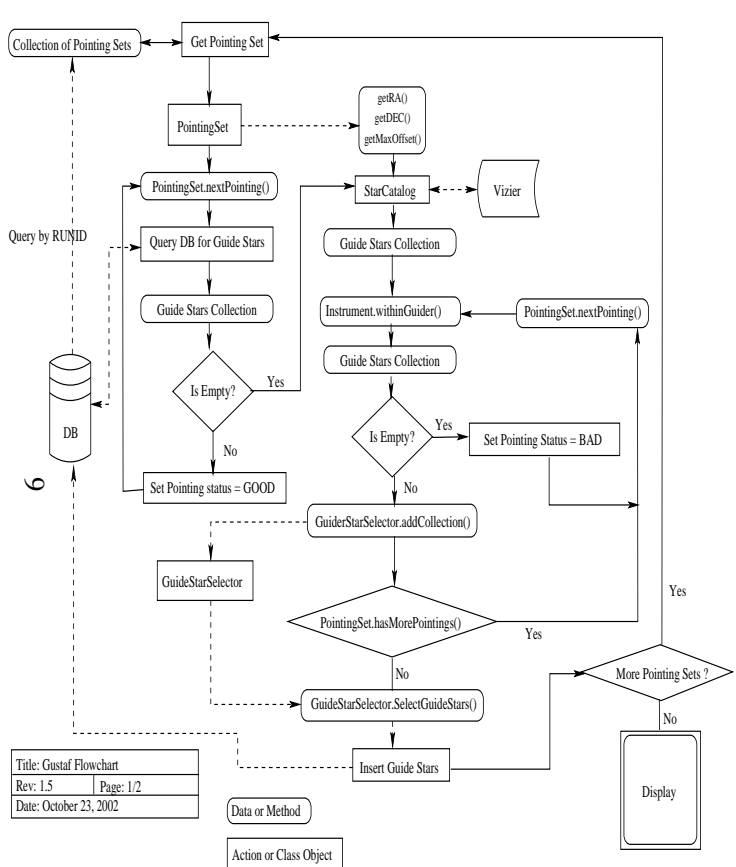
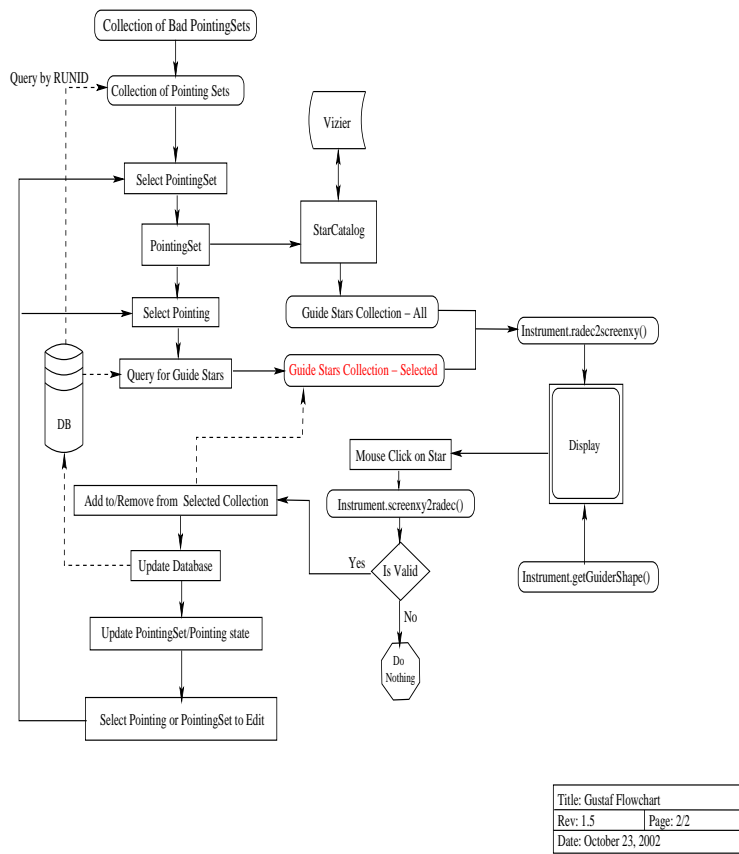


Figure 3: Flowchart for QSO-019 in two distinct modes



8 Interface

The output of the tool should clearly distinguish between targets for which guide stars have been successfully located and targets which have no viable guide stars.

To produce the summary, each Pointing Set must maintain a state to indicate whether or not guide stars have been selected for pointings within the set.

Any interface should perform the following functions.

- User selects Programs to search
- User initiates search
- User selects individual Pointing Sets/Pointings to plot
- GUI over-plots guide stars on instrument FOV
- GUI highlights selected Guide Stars
- GUI allows guide star selection/deselection
- GUI allows commitment of guide stars to database.
- GUI visually identifies Pointing Sets without saved guide stars.

The interface is envisioned as a Panel with a table of PointingSets and buttons for controlling the search, figure 4. The user will select a runid or ANY to populate the table of PointingSets from the database. The user will select the star catalog to use from a drop down list. The user will click the “Search” or “Search All” button to initiate a search for guide stars on one or all the pointing sets in another thread. A “Stop” button can halt the search at the conclusion of the current PointingSet. The user can select an individual row and choose a pointing from a drop down list in an editable table cell and click “View Selected” to display the target and search results in a graphical tool such as Aladin.


```
/* Get the offset RA */
  getOffsetRA()

/* Get the offset DEC */
  getOffsetDEC()

/* Get the Dither Pattern */
  getDitherPattern()

/* Get the status for all pointings */
  getPointingStatus()
```

9.2 class Pointing

```
/* Get the target+offset+DPoffset RA */
  getRA()

/* Get the target+offset+DPoffset DEC */
  getDEC()

/* Get the DB ID for the DP offset */
  getOffsetID()

/* Get the DB ID for the target */
  getTargetID()

/* Get the DB ID for the DP */
  getDitherPatternID()
```

9.3 class DitherPattern

```
/* Get the DP ID for this pattern */
  getID()

/* Get the DP name */
  getName()

/* Get the number of pointings */
  getNumPointings()

/* Get the RA offset for the Nth pointing in arcseconds*/
  getOffsetRA(int)

/* Get the DEC offset for the Nth pointing in arcseconds */
  getOffsetDEC(int)
```

9.4 class GuideStarSelector

```
/* Add a collection of guide stars associated with a pointing */
  addCollection(Pointing p, Collection stars)

/* Select the best guide stars from all the collections added,
   in the order they were added */
  selectGuideStars()

/* Get the pointings in the order they were added */
```

```
getPointings()
```

```
/* Get the guide stars for a pointing */  
getGuideStars(Pointing p)
```

9.5 class DBProxy

```
/* Get a collection of PointingSets given a runid pattern */  
getPointingSets(String runid_pattern)
```

```
/* Get guide stars, if any, for a Pointing from DB */  
getGuideStars(Pointing)
```

```
/* Insert guide stars for a pointing from DB*/  
insertGuideStars(Pointing p, Collection stars)
```

```
/* Delete guide stars for a pointing from DB*/  
deleteGuideStar(Pointing, Star)
```

9.6 class Star

```
/* Get the star RA */  
getRA()
```

```
/* Get the star DEC */  
getDEC()
```

```
/* Get the star Magnitude (Filter ?) */  
getMagnitude()
```

```
/* Get the star name or ID */  
getName()
```