LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)
Data Management

Image Display WG

Yusra AlSayyad (chair), Scott Daniel, Gregory Dubois-Felsmann, Simon Krughoff, Lauren A. MacArthur, John Swinbank

DMTN-126

Latest Revision: 2019-10-21

Abstract

This document describes the findings of the LSST DM Image Display Working Group.

Change Record

Version	Date	Description	Owner name
1	YYYY-MM-DD	Unreleased.	Yusra AlSayyad

Document source location: `https://github.com/lsst-dm/dmtn-126`

Contents

Image Display WG

1 Introduction

The QA Working Group (QAWG) was formed in mid-calendar-2017, with a wide-ranging remit to suggest improvements to processes and tools used for quality assurance across the Data Management subsystem, following the charge defined in **?**. The QAWG produced an extensive report (**?**), which touched on issues in the area of image display and visualization. However, due to uncertainty around project scope and the future of development of image display tools within DM, the QAWG did not issue effective recommendations in this area.

The Image Display Working Group (IDWG) was constituted in summer 2019 with the primary goal of rectifying this shortcoming, as well as issuing recommendations on image display for the purposes of commissioning (which fell outside the QAWG scope) and considering whether useful recommendations can be made about tool development in support of diagnostic work within the Camera Subsystem. The detailed terms of the Working Group are provided in **?**.

2 Approach

The IDWG identified three ways to approach its charge.

1. We considered the various contexts or environments in which a scientist or developer might wish to view images, and established how well these are served by current tools. For example, these environments include a developer working on a standalone laptop, or a scientist using the LSST Science Platform. This analysis is presented in §**??**.
2. We surveyed groups of key stakeholders across the LSST Project, including DM pipeline developers, members of the Commissioning Team, and members of the Camera Team, to understand what they regard as the most important use cases for visualization, and how well those use cases are served by existing tools. This analysis is presented in §**??**.
3. We identified existing tools — including those developed within DM, those produced by other subsystems within LSST, and those available in the wider community — to assess their capabilities and relevance to the key use cases which have been identified. This analysis is presented in §**??**.

Finally, in §**??** we present conclusions and recommendations drawn from the above.

3 Environments

[YA: TO DO: Simon] There are eight environments in which a DM member would view images.

Simon's table.

Simon's chart in words.

We focused on the lsst-dev and Nublado environments, which are partially implemented.

DM requested these essential features. Features = some action users want to be able to complete or some characteristic of the image viewer. Use cases are the commissioning, science validation, pipelines writing tasks that prompt these actions.

Some of these features are currently possible and easy on lsst-dev and Nublado; some are not.

We distinguish between "possible" and "easy."

4 Features and User Stories

[YA: As a <type of user>, I want <some feature> so that <some reason>. Is feature request obvious/self-evident? (Yes/No) If no, seek use cases. Can you do it already on lsst-dev? If yes, how? If not, how much work would it be to make it possible? Can you do it already on LSP? If yes, how? If not, how much work would it be to make it possible?]

4.1 All-Sky Visualization

[YA: TO DO: Lauren] High on the priority list for surveys (such as LSST) with a large footprint on the sky is a tool enabling a visualization of the (to-be) observed area on the full sky covered over some time period (e.g. per night, survey-to-date). For example, one might like to overlay

the instrument footprint of all the single-epoch images accumulated to-date or in a given time period and be able to visualize their on-sky RA/Dec positions with respect to common celestial objects/reference points (e.g. the Milky Way galaxy, the Solar System and its members, constellations, Messier objects, other surveys, etc.)

Specifically, this could include functionality to both:

- display a nominal focal plane footprint at any desired location on the sky, e.g., for use in understanding planned observations; and
- display over a coadd the outlines of the single-epoch images used to generate the coadd, or of all the single-epoch images that contain a given source.

It would also be desirable to be able to display information associated with the footprints of interest (e.g. single-epoch statistics, MAF metrics). This would include quantities that are evaluated across the entire (LSST-observed) sky and it would be desirable to have the ability to explore them at a range of scales; from all-sky down to the finest-grained level at which they are defined. It was noted that this becomes increasingly important in the context of LSSTCam and particularly the Science Validation surveys when larger contiguous regions of sky coverage start to emerge.

At the time of writing, such a capability does not exist specifically for LSST, although some of the desired functionality does overlap with those provided by Firefly (see, e.g. (**?**)). The hscMap viewing tool developed for the HSC-SSP (see §**??**, `https://hscmap.mtk.nao.ac.jp/hscMap4`) provides some of this functionality for all-sky survey footprint visualization.

4.2 Full Focal Plane Visualization

[YA: Thank you Scott for the draft. TO DO: John Edit] Commissioning, Camera, and DRP expressed interest in being able to load a full focal plane and zoom continuously from a large scale view down to pixel-level images. In order to make pixel-level information useful, any viewer will also need to support overplotting of mask planes.

Tony Johnson's image viewer, developed for the camera team, currently supports raft-level images and could be extended to support full focal planes. Luis da Costa has also developed

a large scale viewer to handle preview data as it comes off the camera. This viewer can handle full focal planes. There is talk of combining the development effort on these two tools.

There are third party packages that can do large scale image visualization, but these involve resampling images onto some dynamic pixellization of the sky (i.e.HEALPIX) and thus, on some level, erase the information contained in individual LSST pixels. The principal barrier to just adopting Tony Johnson's image viewer is that it is enabled my a massive hardware investment, which may make it prohibitive for hundreds of users to be viewing hundreds of focal planes simultaneously.

4.3 Mark and save positions on images

[YA: TO DO: Yusra]

Define regions of interest.

Use Case: Crowd Sourced Image Inspection Inspect and markup images during commissioning in order to find surfaces that need black paint (analogous to the des-exp-checker)

Showing a particular instance used by DESC for DC2 r2.1i. Can label image data with ?problems? (e.g. tree rings) that apply at particular positions. These are then stored to file (either just as text, or maybe a SQLite database), although it's not clear in exactly what representation. The ?masks? shown are the actually LSST mask, rendered as opaque colour on top of the image. It's not clear exactly which mask planes are used. Nice UI. Clearly a special-purpose tool. Simple enough that it is appealing to use for appropriate tasks. ?Could use this as a simple GalaxyZoo?. How would you stand this up for use by the DM team, e.g. for looking at diffim outputs? Discussed how one might deploy it on e.g. LDF Kubernetes Commons at the T/CAM standup. And authentication options. Not appropriate for use for a single user, but could be used for crowd-sourcing visual inspection of data. This addresses one particular set of use cases, but that set of use cases is highlighted as important in the inputs we've collected.

4.4 Individual Image Inspection

[YA: The title of this feature is vague. Is this the use case supporting maintaining the ability to open images in ds9? Also note that users are most proficient with DS9.] Independent of all

other recommendations, there is still strong interest in the ability to, outside of any notebook or JupyterLab-like environment, examine individual raw images in detail using DS9 or some equivalent tool. Users are agnostic as to how this tool is provided (i.e. whether it is served through a web browser or through some kind of terminal forwarding in the vein of xpra). The principal concerns are that it be able to rapidly load and manipulate pixel level data and that it be able to operate in a server-client mode so that users do not have to download terabytes of image data onto their local machines.

4.5 Compare Images

[YA: TO DO: Yusra]

- blinking (animation/movie of images
- side-by-side
- Locking two frames by WCS or pixels coordinates
- Locking two frames by scale and limits
- Pan/Zoom

All groups polled requested the ability to directly compare two images, by locking WCS or pixel coordinates, scale and limits, and both blinking and side by side. After locking users want to be able to pan and zoom while viewing side-by-side or blinking. Two groups also requested a "crossfade" option as "nice to have" rather than essential. Crossfade means that two images are overlaid at the same time, and the user moves a slider to shift the relative weighting of each image. Ideas's put forth include also include using a movable "curtain" UI element to wipe one image across the other. *[YA: Whenever photo editing apps that let you compare the before and after of a photographs with a movable curtain, and I always wish they just let me blink.]*

Using ds9 on lsst-dev this is possible and easy (for frequent users of ds9). Using Firefly or matplotlib on Nublado, it may be possible, but it is not easy.

User stories

- As a pipelines developer, I want to be able to blink two locked images so that I can assess the effect of an algorithm modification.
- As a camera team developer, I want to be able to change the scale limits of two side-by-side images by hand, so that I can see the overscan.

4.6 Overlay Maps/Masks

[YA: Thank you Scott for the draft. TO DO: Gregory edits]

Commissioning, AP, and SST specifically requested the ability to layer images on top of each other with varying degrees of transparency and multiple colors. This will be used to overlay the masks (detected, saturated, interpolated, etc.) from the LSST Science Pipelines as well as maps of known defects on top of images during visual inspection. The functionality should be provided through a realtime UI for use when inspecting an image "by hand" and through a programmatic API for use when inspecting an image in a JupyterLab/notebook environment. afwDisplay/ds9 and Firefly both currently support this functionality.

Overlaying maps onto images (currently use afwDisplay Toggle individual mask planes and choose colors

SST: Individual mask planes (including: across a full focal plane; for coadds) "Individual" apparently meaning that one or more mask planes to be displayed could be selected programmatically or by UI, with flexible assignment of colors and transparency. Princeton says: need full focal plane defect overlays at the summit mask integration (Firefly does this well) Obvious nans (e.g. in another color)

Colorized overlays on greyscale images (with tunable alpha) This refers to the rendering of single-channel image data with a hue or a partially transparent color overlay based on an additional channel (e.g., colorizing a single-channel flux image by the per-pixel variance). Note that mask-overlay display could be treated as an instance of such a capability, but for tracking purposes we'll treat it separately as mask overlays (in paragraph above) which benefit from specialized UIs/APIs for extracting and naming bits from packed bit masks.

4.7 Callbacks (Run arbitrary code on a pixel or region)

[YA: 7/12. TO DO: I'm not sure who is the best person to write this: Gregory John Simon] Robert says that callbacks are important because they give the use the ability to add any feature to the image viewer. This makes the extant features available in any particular viewer less important. Yes, If you write it in a way that you can implement arbitrary code and analysis. Powerful. But I think in JS9 you'd have to write in javascript

Per-source drill down Select a source/object (in a table, over plotted on an image, in an x-y plot) and be able to link to additional visualizations or data displays related to that source/object/ KB: It would be very nice to be able to click on a source (or collection of sources) in a pixel-level image and have callback to the notebook aspect to run further analysis on those sources. Basically, brushing and linking integration of pixel-level image visualization with the notebook aspect.

4.7.1 Extensible displays

[YA: TO DO: Simon? John?] Everything must be scriptable. We only care that there are APIs for all these things

Because the set of interesting specific visualizations far exceeds what could be provided centrally, make it possible for users to define visualizations, constructed from the lower-level visualization capabilities in the system (or an external library), in a way that makes them straightforward to apply on demand.

4.8 Snappy User experience

[YA: TO DO: Simon] User experience: There is a big difference between a viz platform being able to do something and having it be easy or responsive enough to not cause an extra barrier to usage. A few specific examples of functionality that needs to be gotten right or the tool will suffer from users finding ways to use other tools:

- Trivial adjustment of range and zero point of image scaling. It should be a one click (or hotkey) operation to get a good guess at the scaling range (e.g. z-scale). Dynamic scaling needs to be achieved through intuitive gestures Example: When looking at raw

frames, the bias has not yet been taken out. This means each amp needs a different scaling. When switching between amps, it must be trivial to adjust the scale parameters via the mouse so that a reasonable set of scale parameters can easily be found (ds9 is the standard for this, as far as I know).

- Pan and zoom must be desktop-like Even a small delay between a drag and the frame panning causes momentary disorientation. Similarly, if the zoom is not close to continuous and instantaneous, the user must reorient after every change in zoom. Example: It is common to be at a fairly high zoom level and want to pan around to find a specific type of object or feature. If the pan takes any fraction of a second, the user needs to figure out where they are so they know where to pan again. Similarly, one may want to zoom out to find other examples. If the zoom isn?t smooth, requires cognitive effort to figure out a) when the right zoom has been reached and b) where to pan after the zoom.

- Mouse over values need to update with ultra-low latency The value of the pixel under the mouse needs to be displayed prominently. The value must update with ultra-low latency as the mouse is moved over the image. Example 1: It is common to want to explore pixel values to get an average by eye of the background. Just by running the mouse over pixels between objects, it is simple to get a sense of the background value if the mouse over value updates essentially continuously. Example 2: It is common to want to try to find the max pixel value in an object. Rather than drawing a trace, it is often sufficient to run the mouse over the pixels of the object to look for a gradient and follow it up. This is much harder to do if it takes significant time to update the mouse over value.

- It needs to be simple to match multiple images on either image coordinates or WCS coordinates. Given a reference image, snap all other images in other frames to the same image or sky (configurable) coordinates and zoom level with less than three mouse clicks Once co-registered, pan and zoom should be linked in all panes This includes the ability to cycle through the frames in the same pane, i.e. ?blinking? through the frames sequentially via a blink selection in a menu option or hotkey. I.e. not sequentially clicking through a series of tabs or list entries. Example: There are lots of moving objects in images of any depth. To find them load three epochs of the same area separated by a few days. Register all images to the first image to the same sky coordinates. Stack the images and in one pane and cycle through them automatically with a delay of 0.5 seconds. Look for points sources that follow a linear trajectory relative to the surrounding stationary point sources.