The Dark Energy Survey Mapping and Coaddition Pipeline

James Annis
Fermilab

1.1 Introduction

The Dark Energy Survey Coadd is one of the core piplelines of the survey. In this document we will describe what is needed for the coadd. Before we begin, there are a set of core guidelines that we follow.

First, the core pipline should be able to be run on a desktop, easily and smoothly. The processing infrastructure should transparently generalize this to less hospitable environments

Second, we believe in a tool kit approach rather than a monolithic approach. We aim at single purpose tools, probably in c, that can be strung together in a pipeline using a scripting language.

Third, and directly relevent to the coadd, we are going to resample the pixels. Once this is accepted, one has great freedom to define output image: we will define the output images to be those of a distortion free instrument at a finer pixel scale than the input images. We will map the input images to the output image, and then coadd them.

1.2 Inputs

1.2.1 Output image definition

The image size, orientation, and pixel scale are defined ahead of time. The actual input is an ra,dec, image size, and pixel scale.

We follow the NVO hyperatlas standard in defining the output images. We choose ISEA-7-TAN-23: the Inverse-Snyder-Equal-Area grid at resolution 7 (21,872 hexes over the entire sky and each hex having 1.9 sq-degrees) using a TAN projection, better known as a gnomic projection, at pixel scale 23 (0.125”). The cell is a hex but we will adopt square images that bound the hex with a buffer zone; this implies that we need a mask that denotes the primary area inside the hex and secondary outside the hex.

At 0.125”/pixels the square is about 40,000 pixels square. This is too big to work with. We will break this up into 64 sub-images, an 8x8 square of images, each of 5000x5000 pixels. This corresponds to 625” on a side, and 0.17° on a side. Most of the sub-images will be entirely inside the primary mask, simplifying boundry effects. The galaxy measurement code an either work with the 5000^2 sub-images or simultaneously on all 64 sub-images.

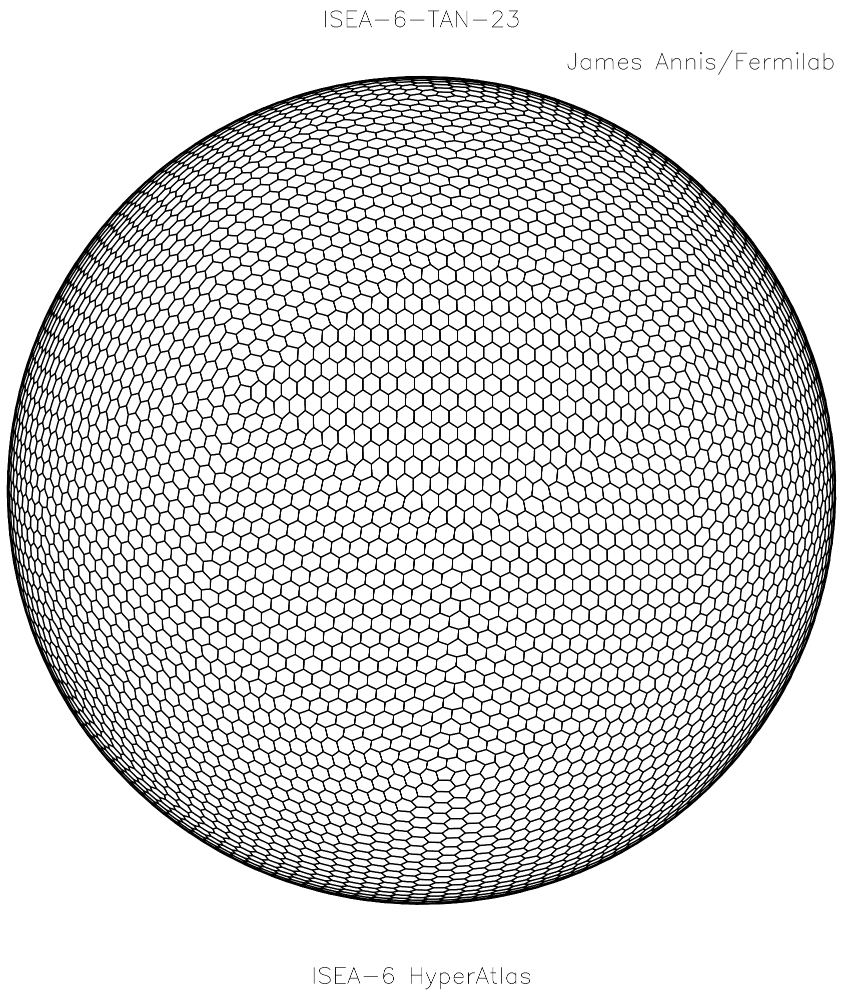

Fig. 1.1. The hexes on the sky of the ISEA-6 hyperatlas. We use ISEA-6 instead of ISEA-7 to save space in the figure. The ISEA-6 has 7292 cells rather than ISEA-7's 21,872.

1.2.2 Input file list

The primary input to the pipeline is the list of images that overlap the output image. The file list will include the file name, global root/file archive, path information along with ra, dec, and seeing.

1.2.3 Input mask list

In serious work, there will be masks on the input images describing bad columns, satellite and meteor trails (to be ignored in the coadd), and saturated pixels (to be treated as unclean; all pixels that partake of these are unclean). There might be cosmic ray masks (ignore these pixels in the coadd). In the SDSS many of these effects were interpolated across, on the principle that a local estimate of the missing pixel was better than no information at all. If we adopt this principle, we will need an interpolated mask, which will be set to a small but non-zero number for the coadd. As for the images, the file will include filename, global root/file archive, and path information.

1.2.4 Input PSF list

The actual form of this input depends on what form the PSF representation takes in the survey. If it takes the form of the SDSS, image by image KL coeffiecients or if it is a similar approach using shapelets, than we need to input a file with entries for each input image containing the coefficents and the basis set. If it is a global static PSF polynomial solution similar to the Jarvis and Jain approach, we need the coefficents of the polynomial fit. The PSF is needed here for one of two reasons: either it will be used in the coadd proper for image circularization, or we may be keeping track of the PSF by coadding the input image PSFs.

1.2.5 Flux scale list

For each image we need the multiplictive factor that brings the images onto a common flux scaling. This is used in the coadd both to scale the image and to scale the variance.

1.2.6 Astrometric transformation

We need astrometric transformations accurate to better than 0.1”. The proposal is that the astrometric solution uses the IPAC SIRTF convention of affine terms plus a polynomial fit for each axis, and inverse transformation. This convention conforms to the draft standards in the WCS IV paper. The form is the standard WCS affine terms, plus polynomials of the form

$$U = x + c_1x^2 + c_2y^2 + c_3xy$$

$$V = y + c_4x^2 + c_5y^2 + c_6xy$$

The degree of the polynomial and the inverse transformation is part of the convention. The coadd will need both the forward and inverse transformations.

1.3 Pre-burner

This stage is empty for desktop application.

On the other hand, there are significant data handling that has to occur for less hospitable environments. The pre-burner has to:

1.3.1 Create the image and mask list

The creation of the file and mask list is a query against a metadata catalog*. The query will include: ra,dec,filter,seeing, and run/reduction level. The response will include a “filename”, which includes the filename and path relative to a global root. Candidates for the metadata catalog include the NOAO Image Archive and the USC/ISI MetaData Catalog; we plan on exploring first the NOAO Image Archive.

* A metadata catalog is a database containing information about images, one of the key pieces of which is location

1.3.2 File staging

The images and masks need to be brought to “.”. This step might be a “cp” (easy), perhaps a “ln -s” to a local copy (minimizes disk space), or a “gsiftp” (from a remote site). At the moment, we plan on using “srm-get”, based on gsiftp but including reliability and disk reservation features inside a tool-kit. This tool comes from either the Fermilab/JeffersonLab “SRM” or from the LBL “DRM”.

1.3.3 Get the flux list

The relative flux scalings of the images come from the relative photometry pipeline. It is unclear whether they will be in the headers; probably not. It would ideal if they were in the meta data catalog; if so the meta-data catalog will have to allow easy insertion and deletion of metadata. We advocate adopting the SWARP “.head” convention, where items that by rights should be in the image header but aren’t are instead put into a “.head” file that consists of a FITS header without an image. This header overrides the header associated with the image itself. It may be tracked, of course, and it ends up in the output images. This takes care of the case where archive images, beyond our ability to modify, have headers that need updating or changing.

1.3.4 Get the PSF representation

Retrieving the PSF representation has simiar problems as the flux scalings.

1.3.5 Get the astrometric transformation

The astrometric transformation is easy to find: we get it from the header, at least in the current plan of performing the astrometry in the single image pipeline. We very much like the approach of keeping things in the header.

1.4 Core Code

The core codes consist of two engines: a mapping engine and a coadd engine.

1.4.1 The mapping engine

The mapping engine is used to map the input images onto the output image definition. This is the most computationally expensive part of the coadd.

The masks are mapped so that during the coadd we can ignore or de-weight bad columns, bleed trails, satellite trails, meteor trails, and perhaps cosmic rays.

The output image footprint is an input. For each pixel of the output image footprint, we ask whether it is contained in the input image. If so, the footprint of the pixel on the input image is determined and an estimate of the flux that pixel would have recieved is made by interpolating from the surrounding pixels.

The science here is in two places: the choice of the interpolation kernel and in the flux conservation. The interpolation kernel is used to map the input image pixels onto an output image pixel. In our current implentation we chose a windowed sinc function called Lanczos-2: $\mathrm{sinc}(\pi x)\dot{\mathrm{sinc}}(\frac{\pi}{2}x)$. Flux conservation demands first that the output image is multiplied by the Jacobean of the coordinate transformation, and second that each input pixel is actually sampled by the output image. The latter is easily met and is one of the reasons the output image is sampled at a finer resolution the the input images. The first is more subtle, and in our current implementation the Jacobean is assumed to be constant across the image. In general this is not true, and we will have to determine whether this approximation is good enough.

The current mapping engine is a modified version of `Swarp` . We are aware of two other candidates, `PyMultidrizzle` and `MSCRED` . The first is functionally equivalent to `Swarp` (we judge it so after a -very- cursory examination) though it has extra features we are not likely to make use of.[*]

1.4.2 The coadd engine

The coadd itself is not overly complicated, except that it is impossible to load all of the images into memory. Both `Swarp` and `PyMultidrizzle` have (modular) code which allow fast working on subsections of images from disk.

There are several coadds techniques that we have the luxury of exploring: median, clipped average, inverse variance weighted, χ^2 against sky color, subspace filtering for red z=1 galaxies. We will also pursue two different types of coadd, depth optimized and weak lensing optimized. The latter concentrates more on the good seeing data and on uniformity of output, on the theory that edges are bad.

We thus break out the coadd into a completely separate step from mapping. This has advantages side effects. We can store the mapped input images so as to make coadds much faster. We can pursue the various prep-steps on the mapped images multiple times without the expense of mapping. The prep steps includes

- sky subtraction: the SDSS `photo` sky subtraction is superior to the `Swarp` sky subtraction, for instance.
- flux scalings: these will change as we get more data and the relative photometry improves.
- variance: the variance of the input images to the coadd depend both on the sky σ and on the flux scaling. If we use variance weighting, this is of course central: we want the weight maps to be variance maps.

Our current implentation uses a version of `Swarp` with most features turned off, using just as the coadd engine, and uses `daedaleos` and `photo` to perform all of the scalings and variance calculations.

1.4.3 The tools

We aim to make the mapping engine a standalone tool, and the coadd engine a standalone tool. This means that when we map the masks we do this as a separate call to the mapping engine rather than just one giant call with images, masks, and PSFs included. We believe that these standalone tools, that one can just download, compile, and run, have a greater use to the community than monolithic pipelines.

1.5 Outputs

- mapped input images
- mapped input masks
- mapped input psf
- coadded image

[*] Multidrizzle finds bad columns, though only negative bad columns; the SDSS has better code than this. Multidrizzle locates cosmic rays by an iterative coadd-subtraction technique; this is perhaps of greater interest, though the SDSS has single image cosmic ray detection codes that we can use to mask CR without the very expensive iterative coadd step.

- coadded image variance/weight map
- coadded image n-map
- coadded image primary mask

1.6 Post-burner

The output of both the mapping engine is single FITS images; the output of the mapping stage is a multi-extension FITS image containing all of the mapped images for a given output image. Likewise the output of the coadd engine is a single FITS file; the output of the coadd stage is a multi-extension FITS file containing the image, the masks, and perhaps the PSF.

The output is

- mapped input images, masks, and psfs. These will probably be in a single multi-extension FITS file.
- coadded image.
- coadded image variance/weight map. This contains the information about the inverse variance per pixel that the galaxy measurement code will use.
- coadded image n-map. This contains a map of how many images contributed to each pixel. Such as map is remarkably useful in quality assurance and trouble shooting.
- coadded image primary mask. This contains the mask telling the galaxy measurement code which part of the image is to be considered “primary” area and which is duplicate sky, to be considered “secondary”.

If the entire coadd module is running on the desktop, we are done. If it is being run as part of the infrastructure the post-burner will not be empty. The post-burner has to:

1.6.1 Put the files into the archive

The files to be saved have to go into the archive. This means that a) they have to be moved to permanent storage, and b) relevent metadata stored in the metadata catalog.

For our next round of implementation, we plan on using SRM to move the files to dcache/enstore, and to store the metadata in the NOAO Image Archive.

1.7 Data Volume

The current SDSS Coadd consists of 3 Terabytes, roughly 10 images per field. There are 8000 fields. Since there are 5 colors, this translates to 40,000 jobs. Note that if each job is considered separately, then the “input” data grows considerably since a given image can contribute to more than one output image; we estimate the treating the jobs as independent expands the input data to 8 terabytes. While the code runs, the images are expanded by a factor of 4, so the output of the mapping stage is 12 Terabytes. The coadded data is of course much smaller than the input images, and masses only 0.5 Terabytes. By the way, clearly it is non-optimal to treat each jobs as independent.

For the DES coadd we will consider the ISEA-7-TAN-23 2 sq-degree patches. Each of the images is $40{,}000^2$ pixels, which at 2 bytes comes to 3.5 Gig. A single tiling of 5000 sq-degrees consists of about 3000 of these images, so will come to about 10 Terabytes. This can be compared with the input data, which will mass about 2 TB.

The first tile contributes 100% overlap. The second tile overlaps 33%, so three tiles will contribute. The third tile does the same. The fourth tile (and fifth and sixth) overlap two tiles at 44% and two more at 8%. For our estimation purposes we can ignore the 8% contributions though of course we will in practice use them.

The input data is 2 TB per tiling. The output data is 2 TB.

We can now calculate the data volume of year 1 for one bandpass: For input there are 2 tilings, so the input is 4 TB. There are 3000 tiles per tiling, and 3.5 Gig per tile for the expanded images so 10 TB per tiling. The mapping stage output (10 TB + 3*10 TB) = 40 TB. The mapping stage output is 1 tiling at $\times 4$ expansion, and 1 tiling at $\times 4$ expansion, each tile of which contributes to 3 ouput tiles.

The data volume of year 2 consists of 4 tilings, so 8 TB of input and 2 TB of output. The mapping stage output (10 TB + 3*10 TB) + (3*10 TB + 2*10 TB) = 90 TB. The mapping stage output includes one tiling which contributes to three output tiles and one tiling that contributes to two output tiles.

Each of these numbers will be multiplied by 4 bandpasses. In year 1 the total is 16 TB in, 8 TB out, and 160 TB intermediate. In year 2 the total is 32 TB in, 8 TB out, and 360 TB intermediate. Clearly we will be putting the mapping stage output onto mass storage.

This calculation suggests that for year 1, and for the simulation of year 1, will be dealing with about 16 TB of input and about 160 TB of output. As of Jan 2005, finding 1 TB of disk space on a compute cluster is annoying but not impossible. The disk space doubling time is about 1.5 years, so three years hence disk space will have gone up by a factor of 4. In Jan 2008 we can expect that finding 4 TB on a cluster is troublesome but possibe. (Jan 2008 is about the time we are in full planning the reduction of 1 years worth of simulated data). We will probably want to break this up into ≈ 5 jobs (which we'll now call campaigns), each covering 1000 sq-degrees. In the planning, we'll want to plan to reserve 4 TB of static disk per campaign, plan on consuming cumulatively an additional 20 TB (though at any instant we'll never need this amount), and plan on reserving 40 TB of mass storage space.

Thus for our SDSS testing “thunder runs”, we'll see about tests of 1 TB static, 5 TB cumulative, and 6 TB of mass storage space. This is what we would get if we divide the SDSS coadd into three thunder runs per coadd. We will explore this regime in the Spring of 2005.