The Astrophysical Journal Supplement Series, 251:3 (23pp), 2020 November https://doi.org/10.3847/1538-4365/abb829

The Pan-STARRS Data-processing System

Eugene A. Magnier[1], K. C. Chambers[1], H. A. Flewelling[1], J. C. Hoblitt[2], M. E. Huber[1], P. A. Price[3], W. E. Sweeney[1], C. Z. Waters[1], L. Denneau[1], P. W. Draper[4], K. W. Hodapp[1], R. Jedicke[1], N. Kaiser[1], R.-P. Kudritzki[1], N. Metcalfe[4], C. W. Stubbs[5], and R. J. Wainscoat[1]

[1] Institute for Astronomy, University of Hawaii, 2680 Woodlawn Drive, Honolulu, HI 96822, USA
[2] LSST Project Management Office, Tucson, AZ, USA
[3] Department of Astrophysical Sciences, Princeton University, Princeton, NJ 08544, USA
[4] Department of Physics, Durham University, South Road, Durham DH1 3LE, UK
[5] Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138, USA

Received 2019 May 9; revised 2020 January 28; accepted 2020 January 29; published 2020 October 30

Abstract

The Pan-STARRS data-processing system is responsible for the steps needed to downloaded, archive, and process all images obtained by the Pan-STARRS telescopes, including real-time detection of transient sources such as supernovae and moving objects including potentially hazardous asteroids. With a nightly data volume of up to 4 TB and an archive of over 4 PB of raw imagery, Pan-STARRS is solidly in the realm of Big Data astronomy. The full data-processing system consists of several subsystems covering the wide range of necessary capabilities. This article describes the Image Processing Pipeline and its connections to both the summit data systems and the outward-facing systems downstream. The latter include the Moving Object Processing System (MOPS) and the public database: the Published Science Products Subsystem.

Unified Astronomy Thesaurus concepts: Sky surveys (1464); Astronomy data analysis (1858); Astronomy databases (83); Astronomy data reduction (1861); Photometry (1234); Astronomical techniques (1684); Astrometry (80)

1. Introduction

The 1.8 m Pan-STARRS1 telescope is located on the summit of Haleakala on the Hawaiian island of Maui. The wide-field optical design of the telescope (Hodapp et al. 2004) produces a 3°.3 field of view with low distortion and minimal vignetting even at the edges of the illuminated region. The optics and natural seeing combine to yield good image quality: 75% of the images have FWHM values less than (1″.51, 1″.39, 1″.34, 1″.27, 1″.21) for ($g_{\rm P1}$, $r_{\rm P1}$, $i_{\rm P1}$, $z_{\rm P1}$, $y_{\rm P1}$), with a floor of ∼0″.7.

The Pan-STARRS1 camera (Tonry & Onaka 2009), known as GPC1, consists of a mosaic of 60 back-illuminated CCDs manufactured by Lincoln Laboratory. The CCDs each consist of an 8 × 8 grid of 590 × 598 pixel readout regions, yielding an effective 4846 × 4868 detector. The GPC1 CCDs have the ability to move charge in both dimensions on the detector and are thus referred to as orthogonal transfer arrays (OTAs). Initial performance assessments are presented in Onaka et al. (2008). Routine observations are conducted remotely from the Advanced Technology Research Center in Kula, the main facility of the University of Hawaii's Institute for Astronomy (IfA) operations on Maui. The Pan-STARRS1 filters and photometric system have already been described in detail in Tonry et al. (2012).

For nearly 4 yr, from 2010 May through 2014 March, this telescope was used to perform a collection of astronomical surveys under the aegis of the Pan-STARRS Science Consortium. The majority of the time (56%) was spent on surveying the three-quarters of the sky north of −30° decl. with $g_{\rm P1}$, $r_{\rm P1}$, $i_{\rm P1}$, $z_{\rm P1}$, $y_{\rm P1}$ filters in the so-called 3π Survey. Another ∼25% of the time was concentrated on repeated deep observations of 10 specific fields in the Medium-Deep Survey. The rest of the time was used for several other surveys, including a search for potentially hazardous asteroids in our solar system. The details of the telescope, surveys, and resulting science publications are described by Chambers et al. (2016).

Pan-STARRS produced its first large-scale public data release, Data Release 1 (DR1) on 2016 December 16. DR1 contains the results of the third full reduction of the Pan-STARRS 3π Survey archival data, identified as PV3. Previous reductions (PV0, PV1, and PV2) were used internally for pipeline optimization and the development of the initial photometric and astrometric reference catalog (Magnier et al. 2020a). The products from these reductions were not publicly released but have been used to produce a wide range of scientific papers from the Pan-STARRS1 Science Consortium members (Chambers et al. 2016). DR1 contained only average information resulting from the many individual images obtained by the 3π Survey observations. A second data release, DR2, was made available 2019 January 28. DR2 provides measurements from all of the individual exposures and include an improved calibration of the PV3 processing of that data set.

This is the second in a series of seven papers describing the Pan-STARRS1 Surveys, the data reduction techniques, and the resulting data products. This paper (Paper II) presents a description of the Pan-STARRS data handling systems, with an emphasis on the Image Processing Pipeline (IPP). The Pan-STARRS IPP consists of a suite of software programs and data systems that are designed to reduce astronomical images, measure astronomical sources on the images, perform the calibration, and distribute the results to various users. The processing system includes extensive parallelization across a large cluster of computers in order to process the large amount of data generated by the Pan-STARRS 1 telescope.

Chambers et al. (2016, Paper I) provide an overview of the Pan-STARRS System, the design and execution of the Surveys, the resulting image and catalog data products, a discussion of the overall data quality and basic characteristics, and a brief summary of important results.

Waters et al. (2020, Paper III) describe the details of the pixel processing algorithms, including detrending, warping, and adding (to create stacked images) and subtracting (to create difference images), and the resulting image products and their properties.

Magnier et al. (2020b, Paper IV) describe the details of the source detection and photometry, including point-spread-function (PSF) and extended source fitting models, and the techniques for "forced" photometry measurements.

Magnier et al. (2020a, Paper V) describe the final calibration process and the resulting photometric and astrometric quality.

Flewelling et al. (2020, Paper VI) describe the details of the resulting catalog data and its organization in the Pan-STARRS database.

M. Huber et al. (2020, in preparation, Paper VII) describe the Medium-Deep (MD) Survey in detail, including the unique issues and data products specific to that survey. The Medium-Deep Survey is not part of DR1 or DR2 and will be made available in a future data release.

Section 2 provides an overview of the full data analysis system and breaks down the major elements of the IPP. Section 3 discusses in some detail each of the analysis steps that may be applied to the images and resulting catalogs of detected sources. Section 4 discusses the databasing system used for calibration, the calibration operations, and summarizes the construction of the public release database. Section 5 discusses the operational infrastructure of the IPP. Section 6 discusses the hardware systems used by the IPP for regular nightly operations and for processing the PV3 data release, with some details on the scale of computing needed to reduce this large number of exposures.

In this article, we use the following typefaces to distinguish different concepts:

1. SMALL CAPS for the analysis stages.
2. *Italics* for database tables and columns.
3. `Fixed-width` font for program names, variables, and miscellaneous constants.

2. Overview of Pan-STARRS Data Processing

2.1. Elements of the Pan-STARRS Data-processing System

The Pan-STARRS data analysis system consists of many elements to support a wide range of activities: archiving and management of the raw and processed image files, real-time nightly processing of images for transient and moving object science, large-scale reprocessing and calibration to produce measurements for the science collaboration and the wider public, specialized image processing to facilitate research and development of the analysis system itself, and distribution of the resulting data products to various consumers in a variety of formats and modes.

The Pan-STARRS data analysis system is divided internally into several major components:

1. Summit Processing: both the camera and observatory summit systems perform data analysis tasks needed to support the ongoing observations. In this article, we focus only on those aspects used by the off-summit analysis stages.
2. IPP: this portion of the data analysis system takes the data from raw pixels on the summit computers to calibrate measurements of astronomical objects in an internal databasing system.
3. Moving Object Processing System (MOPS): this system is responsible for linking individual detections of solar system objects together and determining the orbits (Denneau et al. 2013).
4. Published Science Products Subsystem (PSPS): this system ingests the calibrated measurements from the IPP, MOPS, and others and generates a high-availability database with web-based interactions for public consumption (Paper VI).

Management of the above set of analysis stages takes place at the IfA within the scope of responsibility of the Pan-STARRS Observatory. Across the wider Pan-STARRS collaboration(s), additional data analysis operations are performed to support science results. These collaboration-wide analysis operations range from those that are tightly coupled to the Pan-STARRS Observatory system, such as the analysis of the transient search teams and the public archive database at Mikulski Archive for Space Telescopes (MAST), to those that perform offline analysis for eventual ingest back into the Pan-STARRS databases and archive. The latter category includes the ubercal photometric analysis (Schlafly et al. 2012), the photoz analysis (Saglia et al. 2012), and the QSO/RR Lyra search efforts (Hernitschek et al. 2016). In addition, collaborations within the wider Pan-STARRS community have implemented a variety of science-level analyses of their own to support their science goals (e.g., M31 variable search; Lee et al. 2012, 2014).

Figure 1 illustrates the many elements of the Pan-STARRS data analysis system. This figure focuses on the data analysis steps that occur within the Pan-STARRS Observatory, with an emphasis on the analysis, calibration, and database ingest stages. The MOPS is described in detail by Denneau et al. (2013).

2.2. Nightly Processing Analysis Stages

Data analysis to support nightly science operations is driven by two main goals: (1) rapid detection of the moving and transient sources to enable recovery or follow-up with other telescopes, and (2) regular analysis of the images to monitor data quality and for use in longer-timescale science projects. Not all of the analysis elements listed in Figure 1 are used by the nightly analysis system. Each of the data analysis stages are discussed in detail below. In short, each image is processed independently to correct for instrumental signatures and to detect the astronomical sources (CHIP), astrometric and photometric calibrations are determined (CAMERA), and finally, images are geometrically transformed to a common pixel representation (WARP). Warped images may either be added together (STACK) or used in an image subtraction (DIFF). As part of nightly science processing, images for certain fields, such as the Medium-Deep survey fields (see M. Huber et al. 2020, in preparation), are stacked together in nightly chunks, providing deeper detection capability on 1 day timescales. Depending on the survey mode, difference images are generated for the nightly stack images (using a deep stack template) or for individual warp images. In the latter case, the warp images may be differenced against another warp from the same night or against a reference stack from the appropriate part of the sky.

2.3. Reprocessing Analysis Stages

Pan-STARRS has performed several large-scale reprocessings of both the Medium-Deep and 3π Survey data for internal

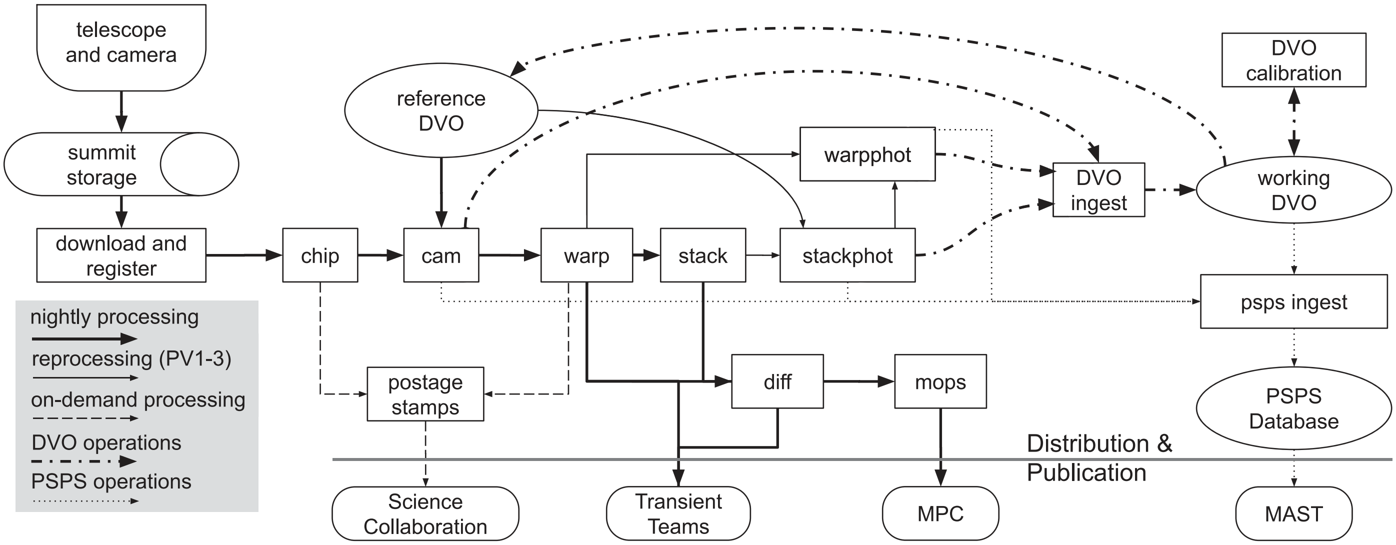

Figure 1. Elements of the Pan-STARRS1 Data Analysis System. Rectangles represent data analysis steps, ellipses represent databases, and rounded rectangles represent external groups ("customers"). The arrows show a simplified representation of the major flow of data between the analysis stages and data-processing elements.

consumption. For the 3π Survey data, we identify these large-scale reprocessings as PV1 (Processing Version 1), PV2, and PV3, with PV3 the analysis used for the first public data release, DR1. We also refer to the nightly science analysis of the data as PV0. For these reprocessing stages, the standard steps of CHIP through WARP, plus STACK and DIFF are performed, starting from raw data, usually using a single homogeneous version of the data analysis procedures. PV2 was a special case in which we started from the camera-level products of PV1 to speed up the turnaround to the community. In addition to the analysis stages listed above which are shared with the nightly processing, these large-scale reprocessing analyses include additional processing steps. A more detailed photometric analysis is performed on the stacks, including morphological analysis appropriate to galaxies (model fits, Kron and Petrosian aperture photometry, etc.). The results of the stack photometry analysis are used to drive a forced photometry analysis of the warp images. These analysis steps are discussed in detail in Paper IV. The data products from the camera, stack, and forced warp photometry analysis stages are ingested into the internal calibration database (DVO, the Desktop Virtual Observatory) and used for photometric and astrometric calibrations (see Section 4.1 and Paper V).

2.4. Data Access and Distribution

During the PS1 Science Consortium operations, data products were provided to the consortium members from many different stages of the analysis process. Data access by the PS1 Science Consortium members was managed through a variety of mechanisms depending on the data volume and type of data products desired. Figure 1 illustrates some of these connections. Access to small samples of imaging data was provided on demand via the Postage Stamp server; access to large sets of predefined raw and reduced data products was provided via the Distribution and Publication systems. The internal calibration DVO databases were provided at several stages via a separate DVO distribution mechanism. For the first two large-scale reprocessings (PV1 and PV2), the data were ingested into the PSPS database system and made available to the PS1SC community through a web portal based at the IfA as well as the MAST portal (see Paper VI for full details).

3. IPP Data-processing Stages

3.1. Processing Database

A critical element in the Pan-STARRS IPP infrastructure is the processing database. This database, using the mysql database engine, tracks information about each of the processing stages. It is used as the point of mediation of all IPP operations. Processing stages within the IPP perform queries of the database to identify the data to be processed at a given stage. As the processing for a particular stage is completed, summary information about the stage is written back to the database. In this way, the database records this history of the processing and also provides the information needed by successive processing stages to begin their own tasks.

The processing database is colloquially referred to as the "gpc1" database, because a single instance of the database is used to track the processing of images and data products related to the PS1 GPC1 camera. This same database engine also has instances (same schema, different data) for other cameras processed by the IPP, e.g., GPC2, the test cameras TC1 and TC3, and the Imaging Sky Probe. In general, processing information for different cameras is separate in different processing database; merging of output products takes place in DVO.

Within the processing database, the various processing stages are represented as a set of tables. In general, there is a top-level primary table which defines the conceptual list of processing items either to be done, in progress, or completed. An associated secondary table (or set of tables) lists the details of component elements that have been processed for each top-level item. Table 1 contains an outline of the database schema, showing the relations between tables organized by processing stage. As an example, one critical stage is the CHIP-processing stage (see Section 3.4) in which the individual chips from an exposure are detrended and sources are detected. Within the gpc1 database, the primary table is called *chipRun* in which

Table 1
GPC1 Database Schema Outline

Stage	Primary Table	Secondary Table(s)	Key
SUMMITCOPY	*pzDataStore*		
	summitExp	*summitImfile*	*summit_id*
	pzDownloadExp	*pzDownloadImfile*	
	newExp	*newImfile*	*exp_id*
REGISTRATION	*rawExp*	*rawImfile*	*exp_id*
CHIP	*chipRun*	*chipProcessedImfile*	*chip_id*
CAMERA	*camRun*	*camProcessedExp*	*cam_id*
FAKE	*fakeRun*	*fakeProcessedImfile*	*fake_id*
WARP	*warpRun*	*warpImfile*	*warp_id*
		warpSkyCellMap	
		warpSkyfile	
STACK	*stackRun*	*stackInputSkyfile*	*stack_id*
		stackSumSkyfile	
STATICSKY	*staticskyRun*	*staticskyInput*	*sky_id*
		staticskyResult	
SKYCAL	*skycalRun*	*skycalResult*	*skycal_id*
FULLFORCE	*fullForceRun*	*fullForceInput*	*ff_id*
		fullForceResult	
		fullForceSummary	
DIFF	*diffRun*	*diffSkyfile*	*diff_id*
		diffInputSkyfile	
DETREND	*detRun*	*detRunSummary*	*det_id*
		detInputExp	
		detRegisteredImfile	
		detStackedImfile	
	detProcessedExp	*detProcessedImfile*	
	detResidExp	*detResidImfile*	
	detNormalizedExp	*detNormalizedImfile*	
ADDSTAR	*addRun*	*addProcessedExp*	*add_id*
DISTRIBUTION	*distRun*	*distComponent*	*dist_id*
		distTarget	
PUBLISH	*publishRun*	*publishDone*	*pub_id*
		publishClient	
LAP	*lapSequence*	*lapRun*	*seq_id*
	lapRun	*lapExp*	*lap_id*
REMOTE	*remoteRun*	*remoteComponent*	*remote_id*

each exposure has a single entry. Associated with this table is the *chipProcessedImfile* table, which contains one row for each of the chips associated with the exposure (up to 60 for gpc1). The primary tables, such as *chipRun*, are populated once the system has decided that a specific item (e.g., an exposure) should be processed at that stage. Initially, the entry is given a state of "run," denoting that the exposure is ready to be processed. The low-level table entries, such as the *chipProcessedImfile* entries, are only populated once the element (e.g., the chip) has been processed by the analysis system. Once all elements for a given stage, e.g., chips in this case, are completed, then the status of the top-level table entry (*chipRun*) is switched from "run" to "full."

If the analysis of an element (e.g., the individual OTA chip) completed successfully, then the corresponding table row (e.g., *chipProcessedImfile*) is listed with a *fault* of 0. If the analysis failed, then a nonzero *fault* is recorded. An analysis which results in a *fault* is one in which the failure is thought to be temporary. For example, if the processing computer had a network interruption and was unable to write some of the output files, this would be an ephemeral failure which was not a failing of the data, but merely the processing system. On the other hand, if the analysis failed because of a problem with the input data, this is noted by setting a nonzero value in a different table field, *quality*. For example, if the CHIP analysis failed to discover any stars because the image was completely saturated, the analysis can complete successfully ($fault = 0$), but the *quality* field will be set to a nonzero value. The various processing stages are able to select only the good ($quality = 0$) elements of a prior stage when choosing items for processing. For example, the CAMERA calibration stage will only use data from chips with good *quality* data, dropping the failed chips from the rest of the analysis. On the other hand, a *fault* in one of the elements for a given stage will block any successive stages that depend on that result from processing that item. In this way, if such a temporary failure occurs, the system will not process an exposure through subsequent stages without the component that has failed temporarily. Because many of the *fault*s that occur are ephemeral due to current conditions of the processing cluster, the processing stages are set up to occasionally clear and retry the faulted entries. Some *fault*s represent software bugs and in the early stages of processing were accumulated until the corresponding software issue could be addressed; since the start of the PS1 Science Consortium Surveys, these types of *fault*s have largely been eliminated. Thus, automatic processing is able to keep the data flowing even in the face of occasional network glitches or hardware crashes.

3.2. Summit Copy

As exposures are taken by the PS1 telescope and GPC1 camera system, the data from the 60 OTA devices are read out by the camera software system and written to disk on a collection of computers at the summit in the PS1 facility called "pixel servers." After the images are written to disk, a summary listing of the information about the exposure and the chip images are added to the summit datastore (an internal http-based data-sharing tool; see Section 5.4).

During night-time operations, while the summit datastore is being populated, the IPP subsystem called SUMMITCOPY monitors the datastores listed in the *pzDatastore* table of the database in order to discover new exposures ready for download. Once a new exposure has been listed on the datastore, SUMMITCOPY adds an entry of the exposure to a table in the processing database (*summitExp*), indexed by an identifier that simply increments the number of exposures announced by the summit, the *summit_id*. This tells the SUMMITCOPY system to look for the list of chips, which are then added to another table (*summitImfile*). This system then attempts to download the chips (registering the results of those operations into the *pzDownloadExp* and *pzDownloadImfile* tables) from the summit pixel servers via an http request. As the image files are downloaded, their MD5 checksum values are calculated and compared with the value reported by the summit datastore. Download failures are rare and marked with a nonzero *fault*, allowing for a manual recovery, rather than automatically rejecting the failed chips. Once all the components of the exposure have been downloaded, they are further entered into the *newExp* and *newImfile* tables, which index the exposures by *exp_id*. This switch in index indicates that the exposure has successfully been copied from the summit to the IPP cluster and that further processing is no longer dependent on outside resources.

3.3. Image Registration

Once the chips for an exposure have all been downloaded, the exposure is ready to be registered. In this context, “registration” refers to the process of adding them to the database listing of known, raw exposures (not to be confused with “registration” in the sense of pixel realignment). The result of the REGISTRATION analysis is an entry for each exposure in the *rawExp* table and one for each chip in the *rawImfile* table. These tables are critical for downstream processing to identify what exposures are available for processing in any other stage. At the REGISTRATION stage, a large amount of descriptive metadata for each chip is added to the *rawImfile* table, the majority of which is extracted from the chip FITS file headers (e.g., R.A., decl., FILTER), some of which are determined by a quick analysis of the pixels (e.g., mean pixel values, standard deviation). The chip-level information is merged into a set of exposure-level metadata and added to the *rawExp* table entry. The exposure-level metadata may be the same as any one of the chip, in the case where the values are duplicated across the chip files (e.g., the name of the telescope or the date and time of the exposure), or it may be a calculation based on the values from each chip (e.g., average of the average pixel values).

Unlike much of the rest of the IPP stage, the raw exposures may only have a single entry in the REGISTRATION tables of the processing database tables (*rawExp* and *rawImfile*).

For GPC1, the REGISTRATION stage is also the stage at which the burntool analysis is run. This analysis is more completely described in Paper III. In brief, the burntool program identifies bright sources on the image and identifies persistence trails that result from the incomplete transfer of charge. As this charge can leak out in subsequent exposures, the burntool analysis is run sequentially on the exposures, based on the observation date and time listed in the headers, with the results stored on disk. As a result of the sequential nature of this analysis, the REGISTRATION of exposures is blocked until the burntool has been run on the previous exposures. Because this stage is only run once per exposure, changes to the burntool code require a semi-manual rerunning of the analysis outside of the regular processing sequence. Because this is a rare event, a standardized pipeline infrastructure was not developed for this circumstance. In a future reorganization, a standard serialized preprocessing step may be needed in the pipeline.

Once the REGISTRATION process has finished, new science exposures that have an *obs_mode* value that indicates they are part of a particular science survey are automatically launched into the science analysis by defining entries for the CHIP-processing stage, as described above. The science analysis of a given exposure can be relaunched multiple times, such as for the large-scale PV3 reprocessing. The automatically launched analysis process ensures the shortest time between observation and analysis, particularly important in the search for transient sources.

3.4. Chip Processing

The science analysis of an exposure begins with the CHIP stage, which operates on the individual OTA image files. This analysis step has two main goals: detrending the image to remove the instrumental signature from the pixel values and the detection of astronomical sources in the objects. Based on the entry, the *chipRun* primary table defining the processing details (with the *state* column indicating it needs processing) and the associated information listed in the *rawImfile*, jobs can be spawned for each component OTA.

The CHIP stage is naturally parallelized by processing data from each of the 60 OTAs independently. Several stages in the IPP analysis are parallelized in a similar fashion; although there are multiple stages that operate on an entire exposure at once, the majority of stages operate on smaller segments of a full exposure, allowing the processing tasks to be spread over the machines in the processing cluster. The pantasks environment (the system that manages the processing jobs; see Section 5.1) attempts to target the processing to a computer that is assigned to host data for the particular OTA. This capability is implemented to reduce the network I/O load by minimizing the number of operations done on nonlocal data. In practice, this targeted processing has not had as large of an impact as was originally intended: the data volume and operational details of the hardware has reduced the ability of any one node to reliably contain a particular OTA. The targeted processing has probably reduced the network load somewhat, but it has not been as critical of a requirement as originally expected.

The actual image processing is performed by the ppImage program. This program reads the raw data into memory and applies the detrend corrections (see Paper III) to each cell in the OTA (stored as different extensions in the FITS file format), and then mosaics the cells into a single contiguous CHIP-stage image. This step also creates in memory additional images to hold the mask data, which indicates which pixels may not be valid, and the variance image, constructed as the Poissonian noise on the number of electrons detected based on the original pixel value and the detector gain. A background model is then fit across the image and subtracted to remove the expected contribution from the sky (see Paper III for details).

With the image calibration procedure finished, object identification and photometry can be performed. Although this can be done using a standalone program, psphot, the underlying functions are contained in a library that allows ppImage to directly do this analysis, removing the need to write out and reread the image data. The details of the detection and characterization of the sources in the image are provided in Paper IV.

The results of the image processing are then written to disk, including the science, mask, and variance images; the binned background model subtracted, the PSF model used in the photometry process; and a FITS catalog of detected sources. Additional binned images of the full OTA are also saved, using 16×16 and 256×256 pixel binning scales for quick visualization. The processing log and a selection of summary metadata describing the processing results are also written to disk. This metadata is used to populate a row in the *chipProcessedImfile* table to indicate that the processing of this OTA is complete.

As each OTA is processed independently of the others across a number of computers, the pantasks server managing the jobs periodically runs an advance task that checks that the number of rows in *chipProcessedImfile* with *fault* equal to zero matches the associated number of rows in *rawImfile*. If this condition is met, then all processing for that exposure is finished, and the *state* field is set to “full.” If the *chipRun.end_stage* field is set to CHIP, then no further action is taken. However, this field is usually set to a subsequent stage (most often WARP), in which case an entry for this exposure is added to the *camRun* table, and processing continues.

3.5. Camera Calibration

After sources have been detected and measured for each of the chips, the next stage is to perform a basic calibration of the full exposure in the CAMERA stage. This runs as a single job for the entire exposure, passing the collection of FITS table catalogs generated from each OTA in the CHIP stage to the `psastro` program. Although the full catalog is loaded, the calibration primarily concerns the positions ($x_{\rm ccd}$, $y_{\rm ccd}$) and the instrumental PSF magnitudes. The header information in these catalogs is used to determine the coordinates of the telescope boresite (R.A., decl., position angle). These three coordinates are used, along with a predetermined model of the OTA layout within the camera, to generate an initial guess for the astrometry of each chip. Reference star coordinates and magnitudes are loaded from a reference catalog for a region corresponding to the boundaries of the exposure, padded by a large fraction (25%) of the exposure diameter to help guarantee a solution in the case of a modest pointing error. The guess astrometry is used to match the reference catalog to the observed stellar positions in the focal plane coordinate system. Early on in the PS1SC mission, the nightly processing (PV0) used a reference catalog based on a combination of external catalogs (2MASS, Tycho, USNO). Later, reference catalogs based on Pan-STARRS data was used. For the 3π PV3 analysis, the reference catalog was based on Pan-STARRS data from the PV2 analysis (see Paper V for more details).

Once an acceptable match is found, the astrometric calibration of the individual chips is performed, including a fit to a single model for the distortion introduced by the camera optics. The astrometric model includes a set of third-order polynomials for the transformations from the chip coordinate system to the camera focal plane coordinate system and a single additional third-order polynomial transformation from the camera focal plane coordinate system to the tangent plane of a tangent projection.

As discussed in detail in Paper V, we find that, for the PS1 images, small-scale structures are present in the astrometric transformation. Some of these are due to ripples in the focal surface, while others may be caused by the atmosphere. We find that including higher-order terms in both the chip to focal plane and focal plane to sky are necessary to capture significant astrometric signals. Some care must be taken in the fitting process to avoid degeneracies between terms on different scales.

For the 3π PV3 analysis, the typical astrometric residuals are in the range of 20–30 mas, sufficient to match observations of the same objects between different exposures. There are, however, inevitable outliers. Certain chips occasionally have systematically worse astrometry, with OTA XY17 notably poor in this respect.

After the astrometric analysis is completed, the photometric calibration is determined using the final match to the reference catalog. A single photometric zero point is determined for each exposure, with the airmass term fixed to the nominal linear slope for each filter. No color terms are measured between the observed photometry and the reference photometry. However, at this stage, predetermined color terms may be used to transform the reference photometry to an appropriate photometric system. For the PS1 nightly processing, the reference catalog does not include $w_{\rm P1}$ photometry, so a fixed color transformation is used to generate synthetic w-band photometry from the $r_{\rm P1}$ and $i_{\rm P1}$ photometry. For more details, see Paper V. The result of these calibrations is stored as a single multi-extension FITS table containing the results from each OTA as a separate extension.

In addition to the astrometric and photometric calibrations, the CAMERA stage also generates the dynamic masks for the images. These include masking for optical ghosts, glints, saturated stars, diffraction spikes, and electronic crosstalk. The mask information is generated based on the reference star catalog, along with models for the various effects. Note, however, that this analysis does not go back to the pixels to validate the prediction. The mask images generated by the CHIP stage are updated with these dynamic masks and a new set of files are saved for the downstream analysis stages. The CAMERA stage also merges the binned chip images (see Section 3.4) into single jpeg images of the entire focal plane. These jpeg images can then be displayed by the process monitoring system to visualize the data processing.

Again, summary metadata is saved to disk as well, and the results listed therein are used to populate a row in the *camProcessedExp* database table. As the full exposure is processed all at once, this update also updates the associated *camRun* entry, linked by *cam_id*. As with the CHIP stage, if the *camRun.end_stage* is for a subsequent stage, an appropriate entry is added to the *fakeRun* table.

3.6. Fake Analysis

The FAKE stage was originally designed to do false source injection and recovery, in order to determine the detection efficiency of sources on the exposure. However, early in the design of the IPP, this task was moved to the rest of the photometry analysis done at the CHIP stage. Removing the stage would require significant changes to the database schema. As a result, this conveniently named stage generally does no actual data processing and consists mainly of database operations to move the exposure on to the WARP stage. The operations mimic the CHIP stage, with individual jobs run for each OTA that update rows in the *fakeProcessedImfile* and an `advance` task that updates the *fakeRun* table and promotes the exposure to the next stage by adding a row to the *warpRun* table.

3.7. Image Warping

The WARP stage transforms the image pixels from the regular grid laid out on the chips in the camera to a system of pixels with consistent geometry for a location on the sky. The new image coordinate system is defined by one of a number of "tessellations" that specify how the sky is divided into individual images. A single tessellation starts with a collection of projection centers distributed across the sky. A grid of image pixels about each projection center corresponds to sky positions via a projection with a specified pixel scale and rotation. In general, the pixel grid within the projection is defined as a simplified grid with the y-axis aligned to the decl. lines and no distortion terms. The projection centers are typically separated by several degrees on the sky; for pixel scales appropriate to GPC1, the resulting collection of pixels would be unwieldy in terms of memory in the processing computers. The pixel grid is thus subdivided into smaller sub-images called "skycells."

A tessellation can be defined for a limited region, with only a small number of projection centers (e.g., for processing the M31 region) or even a single projection center (e.g., for the Medium-Deep fields). For the 3π Survey, the tessellation contains projection centers covering the entire sky. The version used for the PV3 analysis is called `RINGS.V3`. In this tessellation, projection centers are spaced every 4° in decl. and the R.A. spacing is approximately 4° as well, adjusted to ensure an integer number of equal-size regions. `RINGS.V3` uses a pixel scale of $0\overset{''}{.}25$ per pixel. The projections are subdivided into a 10×10 grid of skycells, with an overlap region of $60''$ between adjacent skycells to ensure that objects of modest size are not split on all images. The coordinate system used for these images matches the parity of the sky, with north in the positive y direction and east to the negative x direction. The tessellations used by the IPP are stored in the DVO format (see Section 4.1). A table in the processing database, *SkyTable*, lists the projection centers and image boundaries for all skycells.

The first step of the WARP stage is to determine which skycells overlap with the input exposure. These overlaps are determined by the `dvoImageOverlaps` program, which compares the astrometrically calibrated catalog from the CAMERA stage to the DVO database defining the target tessellation. The output of this command is used to populate the *warpSkyCellMap* table in the database, which contains a row for each skycell and OTA that overlap. Each skycell may contain contributions from multiple OTAs; because they are similar in size, in a typical situation, the warp is constructed from four to six neighboring OTAs.

Once this mapping has been defined, jobs to warp the pixels onto each skycell are run, passing the CAMERA stage catalog and the CHIP stage images (including the variance images and the updated masks) to the `pswarp` program. For details on the warping algorithm, see Paper III. The outputs of this program are the geometrically transformed images (signal, variance, and mask) containing all input pixels warped to the common skycell pixel grid. These can subsequently be used for stacking and difference image analyses. For the 3π Survey data, the signal, mask, and variance images generated at this stage are being made available from the image extraction tools at the MAST archive at STScI as part of the DR2 data release.

When the WARP jobs have completed, an entry for the skycell is added to the *warpSkyfile* database table, linked to the *warpRun* entry by a common *warp_id*. An `advance` task again checks that all potential skycells have been generated. At this point, the direct promotion of exposures from one stage to the next stops, as the logic for matching exposures for other combinations is more complicated than simply adding a single entry.

3.8. Stack Combination

The skycell images generated by the WARP process can be added together to make deeper, higher signal-to-noise images in the STACK stage. These stacked images also fill in coverage gaps between different exposures, resulting in an image of the sky with more uniform coverage than a single exposure.

In the IPP processing, stacks may be made with various options for the input images. During nightly science processing, the eight exposures per filter for each Medium-Deep field are automatically combined into a set of stacks for that field. These so-called "nightly stacks" are used by the transient survey projects to detect faint supernovae, among other transient events. For the PV3 3π analysis, all images in each filter from the observations for this survey were stacked together to generate a single set of images with $\sim 10\times$–$20\times$ the exposure of the individual survey exposures.

For the PV3 processing of the Medium-Deep fields, stacks have been generated for the nightly groups and for the full depth using all exposures, producing "deep stacks." In addition, a "best seeing" set of stacks have been produced using image quality cuts described in Paper VII. We have also generated out-of-season stacks for the Medium-Deep fields, in which all images *not* from a particular observing season for a field are combined into a stack. These later stacks are useful as deep templates when studying long-term transient events in the Medium-Deep fields as they are not (or less) contaminated by the flux of the transients from a given season.

When a given set of STACK stage processing is defined, exposures with existing WARP entries that match the filter, position, and other criteria such as seeing are identified (see Section 5.2 for details on how this is automated). An entry is then added for each skycell in the *stackRun* table, with the *warp_id* entries for the exposures added to the *stackInputSkyfile* table, linked to the *stackRun* entry by the *stack_id* field. This defines the mapping for which exposures contribute to the STACK. The STACK stage processing is performed at the skycell level.

The STACK jobs pass the information about the input images and catalogs to the `ppStack` program, which performs the image combinations. Input warps are combined based on a weighting defined by the median variance for each image; see Paper III for details on the stack combination algorithm. In addition to the standard image, mask, and variance produced at other stages, additional images are constructed with information about the contributions to each pixel. A number image contains the number of input exposures used for each pixel, along with an exposure time map, and a weighted exposure time map that scales the exposure time based on the relative variance of each input. These images for the 3π analysis are currently available from the MAST image extraction tools at STScI.

Upon completing the generation of these images, a row is added to the *stackSumSkyfile* table with statistics about STACK processing. As this completes all processing for the entry, no `advance` job is required.

3.9. Stack Photometry

Although images are generated in the STACK stage of the IPP, the source detection and analysis of those images is deferred to the STATICSKY stage. This separation is maintained because the photometry analysis of the STACK images, including convolved galaxy model fitting, is performed on all five filters simultaneously. By deferring this analysis, the processing system may also decouple the generation of the pixels from the source detection. This makes the sequencing of analysis somewhat easier and less subject to blocks due to a failure in the long-running stacking analysis. Similar to the STACK stage, an entry is created in the *staticskyRun* table, linked to a series of rows in the *staticskyInput* table by a common *sky_id*, each of which also contains the appropriate *stack_id* entries for the skycell under consideration.

The input images are passed to the `psphotStack` program, which does the analysis. The stack photometry algorithms are described in detail in Paper IV. In short, sources are detected in all five filter images down to the 5σ significance. The collection of detected sources is merged into a single master list. If a source is detected in at least two bands, or only in the $y_{\rm P1}$ band, then a PSF model is fitted to the pixels of the other bands in which the source was not detected. This forced photometry results in lower significance measurements of the flux at the positions of objects that are thought to be real sources, by virtue of triggering a detection in at least two bands. The relaxed limit for the $y_{\rm P1}$ band is included to allow for searches of $y_{\rm P1}$ dropout objects: it is known that faint, high-redshift quasars may be detected in the $y_{\rm P1}$ band only. Sources detected only in the $y_{\rm P1}$ band are therefore more likely to have a higher false-positive rate than the other stack sources. The parameters of the PSF model are allowed to vary with position in the skycell. The PSF model is also used to convolve the analytical galaxy models, which are the fitted to the observed flux distributions. Galaxy models include Sérsic, de Vaucouleur, and exponential profiles.

The stack photometry output files consist of a set of FITS table catalogs, with one file for each filter. Within these files, there are multiple table extensions, with different classes of measurements saved in the different extensions. The extensions include a table of the measurements of sources based on the PSF model, a table of aperture-like parameters such as the Petrosian flux and radius, a table of the convolved galaxy model fits, and a table of the radial aperture measurements. Once the photometry is complete, a row is added to the *staticskyResult* table with basic statistics from the analysis.

The stack photometry output catalogs are recalibrated for both photometry and astrometry in a process very similar to the CAMERA calibration stage. Although the individual warps that go into the stack are calibrated based on the CAMERA stage analysis, there was some concern that these calibrations might not be sufficiently well defined for some of the input warps, biasing the photometry of the stack. By recalibrating the stacks, we can be sure that the stack photometry as measured is tied to the photometric reference system.

In the case of this SKYCAL stage, each skycell is processed independently. Because of this independence, when queued for processing, the entries in the *skycalRun* table contain the *sky_id* and *stack_id* entries of the parent data directly. As in the CAMERA stage, the `psastro` program reads in the stack photometry catalog, and produces a calibrated output, with format matching the input. A different processing recipe is supplied to `psastro`, which controls for the different data. The same reference catalog is used for the CAMERA and STACK calibration stages. Upon completion, the analysis statistics are written to the *skycalResult* table.

3.10. Forced Warp Photometry

Traditionally, projects that use multiple exposures to increase the depth and sensitivity of the observations have generated something equivalent to the STACK images produced by the IPP analysis (see CFHT Legacy survey, COSMOS, etc.). In theory, the photometry of the STACK images produces the "best" photometry catalog, with best sensitivity and the best data quality at all magnitudes. In practice, these images have some significant limitations due to the difficulty of modeling the PSF variations. This difficulty is particularly severe for the Pan-STARRS 3π Survey stacks due to the combination of the substantial mask fraction of the individual input exposures, the large intrinsic image quality variations within a single exposure, and the wide range of image quality conditions under which data were obtained and used to generate the 3π PV3 stacks.

For any specific stack, the PSF at a particular location is the result of the combination of the PSFs for those individual exposures that went into the stack at that point. Because of the high mask fraction, the exposures that contributed to pixels at one location may be somewhat different just a few tens of pixels away. In the end, the STACK images have an effective PSF that is not just variable, but changing significantly on small scales in a highly textured fashion.

Any measurement that relies on a good knowledge of the PSF at the location of an object needs to determine the PSF variations present in the STACK image, or the measurement will be somewhat degraded. The highly textured PSF variations make this a very challenging problem: not only would such a PSF model need to be highly fine-grained, there would likely not be enough stars in a given STACK image to determine the model at the resolution required. The IPP photometry analysis code uses a PSF model with 2D variations using a grid of at most 6×6 samples per skycell, a number reasonably well matched to the density of stars at most moderate Galactic latitudes for the PS1 3π depths. This scale is far too large to track the fine-grained changes apparent in the stack images.

Thus, PSF photometry and convolved galaxy model analysis in the stack are degraded by the PSF variations. Aperture-like measurements are in general not as affected by the PSF variations, as long as the aperture in question is large compared to the FWHM of the PSF.

The IPP analysis solves this problem by using the sources detected in the stack images and performing forced photometry on the individual warp images used to generate the stack. This FULLFORCE analysis is performed on all warps for a single skycell and filter as a single unit within the processing database, while individual warps are processed individually in parallel as separate processing jobs. A separate PSF model is determine for each of the warp images so that the combined measurement is reliable.

When processing is queued for this stage, an entry is added to the *fullForceRun* primary database table with a reference to the corresponding stack and *skycal_id* entry that is the input source of detections to be measured. The *warp_id* values for the input WARP stage images that contributed to the STACK associated with that *skycal_id* are then added to the *fullForceInput* table, linked to the primary table by the *ff_id* identifier. The individual jobs for each warp are then run, which passes the WARP stage image products along with the SKYCAL catalog to the `psphotFullForce` program.

The convolved galaxy models are also remeasured by the FULLFORCE stage analysis using the WARP images. In this analysis, the galaxy models determined by the STATICSKY photometry analysis are used to seed the analysis in the individual WARP images. Galaxy models are not fitted independently on each warp. Rather, the analysis tests a grid of galaxy model parameters in the vicinity of the prior from the stack.

For each warp image, each set of galaxy model parameter values is used to generate a model that is then convolved with the PSF for that warp image and then compared to the observed data. A normalization and χ^2 value is determined for each set of parameter values for each warp image. For each set of parameter values, the normalizations and χ^2 values are combined across all warps to generate a single grid of parameters. The best set of galaxy model parameters, along with the corresponding normalization and χ^2 value, is then determined from the grid by interpolation.

The purpose of this galaxy model analysis is the same as the FULLFORCE PSF photometry: the PSF of the STACK image is poorly determined due to the masking and PSF variations in the inputs. Without a good PSF model, the PSF-convolved galaxy models are of limited accuracy.

Upon completion of the forced photometry, an entry is added to the *fullForceResult* table with the processing statistics for that combination of *ff_id* and *warp_id*. The individual warp measurements are combined together to produce an average warp photometry value for each object within the context of the DVO object database system, including recalibration of each warp based on the tie to the average photometry of the objects measured in the CAMERA stage.

Once all of the entries in the *fullForceInput* table have finished, a summary operation is run to combine the galaxy photometry analysis measurements into a single value. The output catalogs listed in the *fullForceResult* table are passed to the `psphotFullForceSummary` program to calculate the averages of the individual warp measurements, weighted by the signal-to-noise ratio of the flux measurements. When this analysis completes, an entry is added to the *fullForceSummary*, and the *fullForceRun* entry is marked as completed.

3.11. Difference Images

Two of the primary science drivers for the Pan-STARRS system are the search for hazardous asteroids and the search for Type Ia supernovae to measure the history of the expansion of the universe. Both of these projects require the discovery of faint, transient sources in the images. For the hazardous asteroids, and solar system studies in general, the sources are transient because they are moving between observations; supernovae are stationary but transient in brightness. In both cases, the discovery of these sources can be enhanced by subtracting a static reference image from the image taken at a certain epoch. The quality of such a difference image can be enhanced by convolving one or both of the images so that the PSFs in the two images are matched (e.g., Alard & Lupton 1998).

In the DIFF stage, the IPP generates difference images for appropriately specified pairs of images. It is possible for the difference image to be generated from a pair of WARP stage images, from a WARP and a STACK of some variety, or from a pair of STACK stage images. During the PS1 survey, pairs of exposures, called TTI pairs (see Survey Strategy in Paper I), were obtained for each pointing within a ~1 hr period in the same filter and to the extent possible with the same orientation and boresite position. The standard PS1 nightly processing generated difference images from the resulting pairs of WARP images. The nightly processing generated STACK images for the Medium-Deep fields, and these were combined with a template reference STACK image to generate "stack–stack diffs" each night they were observed. For the PV3 3π processing, the entire collection of WARP stage images for the survey was combined with images generated by the STACK processing to generate "warp–stack diffs," for eventual public release.

When a DIFF processing is defined, an entry is added to the *diffRun* table, and the appropriate input images are added to the *diffInputSkyfile* table, with one entry for each skycell that is covered by the images. For a DIFF generated from two WARP stage products, the input images have their *warp_id* values recorded in *warp1* and *warp2* for each skycell that overlaps. If two STACK stages are to be used in the difference, their *stack_id* entries are recorded in the *stack1* and *stack2* fields. As each STACK only covers a single skycell, the DIFF is usually defined indirectly, using other information from the *stackRun* table to select appropriate *stack_id* values. Similarly, DIFF processing is defined for the mixed case by creating entries that populate one of *warp1* and *stack1* and populating one of *warp2* and *stack2*. In all cases, the minuend of the subtraction to be performed is the "1" entry, and the subtrahend is the "2" entry.

Jobs are created based on the entries of *diffInputSkyfile*, with the appropriate images and catalogs passed to the `ppSub` program. This does the subtraction, as well as the photometry of any sources detected in the DIFF image. Sources may be detected as a positive source (flux in the minuend is higher than the subtrahend) or as a negative source (flux in the subtrahend is higher). The algorithm used for PSF matching is described in Paper III. Upon completion of these jobs, statistics about the processing are written to an entry in the *diffSkyfile* table. An `advance` checks for the completion of all of the components listed in *diffInputSkyfile* and marks the *diffRun* entry as such.

Table 2
Processing Failure Rates per 100,000 Items

Stage	Nightly Processing	3π PV3
Chip	48	34
Camera	262	280
Chip Astrom	N/A	307
Warp	14244	13835
Warp Pixels	N/A	3900
Stack	N/A	5

3.12. Processing Failure Rates

Table 2 lists the unrecoverable failure rates for several of the major IPP stages for both the regular nightly processing and the PV3 analysis of the 3π data set. The table gives the rate per 100,000 of the item processed. In the case of the CHIP and WARP stages, the items correspond to individual chips and skycells, respectively, while for the STACK stage, items are the stack skycells. For the CAMERA stage, the items correspond to complete exposures. The entire exposure fails for CAMERA only in extreme cases. The astrometric calibration of individual chips may fail if there are not enough stars in the image, but the rest of the exposure may then still succeed. Chips that formally succeed in the astrometry analysis but that have an astrometric calibration quality worse than our specification will also be excluded from ingest into the DVO database (see below). We list the astrometry failure rate for chips based on their absence from the DVO database.

For the warp analysis, the apparent high failure rate is something of an artifact. Target output skycells are defined based on conservatively generous boundaries for the

Table 3
DVO Database Tables

Table Name	Description
Images	The images that have objects in the DB
Average	Astronomical objects including their astrometric properties
SecFilt	Average photometry of the objects in multiple filters (one filter per row)
Measure	Detections of sources identified with an object, potentially linked to an image
StarPar	Stellar parameters determined by the Harvard group (Green et al. 2015)
Lensing	Lensing (KSB) parameters and fixed circular aperture photometry from the warps
LensObj	Average lensing and fixed circular aperture photometry
Galphot	Result of galaxy model fits (forced galaxy models)
SkyRegions	Spatial distribution of tables
Photcodes	Transformations between different photometric systems
Hosts	Computers used to store the tables

corresponding chips. This results in a number of skycells with only a small fraction of valid pixels, for which there are likely few stars to measure the PSF. In the processing, any warp skycell with less than 10% of its pixels unmasked in the output are automatically rejected. In addition, the analysis will register a poor quality if too few stars are available for the PSF modeling. To judge the rate at which the warp stage is losing pixels, either due to this effect or veritable analysis failures, we compare the total area of good (unmasked) pixels in the warp skyfiles to the total number of expected unmasked pixels from the corresponding input exposures using the masking fractions and total detector areas reported in Paper III. The result is that roughly 3.9% of the good input pixels are lost to the warp processing.

4. Database Ingest and Calibration

4.1. DVO

4.1.1. Overview

The Pan-STARRS IPP uses an internal database system, distinct from the publicly visible database system, to determine the association between multiple detections of the same astronomical object and as part of the astrometric and photometric calibration process. This database system, called “DVO,” was developed originally for the LONEOS project (Bowell et al. 1995) and used as part of the CFHT Elixir system (Magnier & Cuillandre 2004). The capabilities of this databasing system have been somewhat expanded for the Pan-STARRS context.

DVO tracks three main classes of information: (1) average properties of astronomical objects, (2) measurements of those objects (from which the average properties are derived), and (3) properties of the images that provided some or all of the measurements. In addition, certain metadata tables define general features of the database. Table 3 lists the full collection of database tables used by DVO. In the current implementation, as described in more detail below, the database tables are stored on disk using a distributed collection of FITS files, potentially distributed across a large number of computers in the cluster.

In the most basic implementation, a collection of measurements for detections from a set of images is loaded into DVO along with the metadata describing the images. The latter includes properties such as the exposure time, airmass, filter, time and date of the exposure, etc. Critically, the image metadata includes an astrometric transformation relating the detection coordinate on the image to the coordinate on the sky. As the collection of measurements is loaded into DVO, the software constructs astronomical objects based on those detections. If images overlap, multiple observations of the same astronomical object are grouped together. Thus, a single DVO database will contain a one-to-many relationship between the images and the measurements and a many-to-one relationship between the measurements and the derived astronomical objects.

4.1.2. DVO Schema

Photcodes—DVO has a special metadata table called *photcode* which identifies the photometry filter systems. Entries in this table are used to identify the source of measurements and images. Each row in the *photcode* table includes a *photcode* name, a unique numerical ID, and information about that photometry system.

There are three classes of photcodes defined within the DVO system. One class of photcodes defines the filter systems for the average photometry measurements; these are called `SEC` photcodes. A second class of photcode is associated with measurements from a specific camera for which image metadata is available, called `DEP` photcodes. There are also those measurements that come from external data sources for which DVO does not have any information to determine a calibration (e.g., instrumental magnitudes and detector coordinates). These measurements are reference values and are assigned `REF` photcodes.

The names for `SEC` photcodes are the names of filter systems, such as g, r, i or J, H, K. For `DEP` and `REF` photcodes, the names are constructed from the name of a camera or telescope (e.g., GPC1 or 2MASS), the name (or short-hand name) of a filter (e.g., $g_{\rm P1}$), and an identifier for the detector, if not unique (e.g., XY01 for a GPC1 OTA).

Additional information is associated with each photcode to define the nominal zero point and airmass slope, as well as color trends to transform a measurement in the specific photcode to a common system. For example, a `DEP` photcode GPC1.g.X01 would have the nominal zero point (24.563) and airmass term (0.147). The database elements allow for individual chips to have different color terms to bring them to a common filter system.

DVO ingest methods are defined for several large-scale surveys for which the published data represent average properties derived from multiple measurements, and for which the measurement-to-image relationship is not provided. Ingest methods have been defined, for example, for 2MASS, WISE, Gaia, and USNO-B. In each of these cases, the astrometric and photometric measurements are stored in the *Measure* table, with the data source identified by the photcode of the measurement.

Measurement Tables—In most cases, the individual measurements of the astronomical objects are carried in the table *Measure*. For measurements from PS1 in the PV3/DR1 or DR2 databases, this would consist of values determined by `psphot` for each CHIP, WARP, or STACK stage image. Measurements for other cameras processed by the IPP may also be included similarly in a DVO database. Measurements from other sources, such as SDSS, 2MASS, or WISE, can also be included in this table, distinguished by their different photcodes.

The *Measure* table includes the instrumental magnitudes for the PSF, aperture, and Kron photometry; raw position (*Xccd*, *Yccd*) and second moments (*Mxx*, *Myy*, *Mxy*); along with shape parameters of the PSF model at the position of the object (*FWx*, *FWy*, *theta*). Metadata about the exposure where the measurement was derived from is also included, such as the exposure time, the date and time of the observation, airmass, azimuth, and *photcode* information specifying the filter. The *Measure* table also carries the calibration magnitude offsets ($M_{\rm cal}$ and $M_{\rm flat}$, discussed below) and the astrometrically calibrated position. Astrometric offsets for several systematic corrections discussed below are also defined for each measurement. Photometry from CHIP, WARP, and STACK are all placed in the same table with photcodes distinguishing the source. Because stacks and forced warp fluxes may have nonsignificant values, the table is somewhat denormalized: it also carries both magnitudes as well as instrumental flux values for the PSF, aperture, and Kron photometry. In this case, we have chosen to trade storage space for computing time.

For the warp images, we also measure the weak-lensing KSB parameters related to the shear and smear tensors (Kaiser et al. 1995). These measurements are stored in the *Lensing* table, along with the radial aperture fluxes for radii numbers 5, 6, and 7 (respectively $3\farcs0$, $4\farcs63$, and $7\farcs43$). This table contains one row for every warp image on which the object was measured.

The *Galphot* table stores the results of the forced galaxy fitting analysis for each object that has been measured. This table contains one row per filter and model type (Sérsic, exponential, or de Vaucouleur) if forced galaxy models have been calculated for the object.

The *Starpar* table carries measurements provided by the Harvard team (Green, Schlafly, Finkbeiner) from the analysis of the SED of objects in the PS1 3π data, using the PV2 analysis version (Green et al. 2014, 2015). In this work, the goal was a 3D model of the dust in the Galaxy based on Pan-STARRS and 2MASS photometry. As part of this analysis, the authors fit the SEDs of all stellar sources (as determined by a cut based on the PSF–aperture magnitudes) with stellar models including free parameters of extinction, distance modulus, metallicity, and absolute *r*-band magnitude. While these photometric distance modulus measurements are not extremely precise, they provide a constraint on the distance that is used in our analysis of the astrometry (see Paper V).

Object Tables—One of the main purposes of DVO is to define the relationship between individual detections of an astronomical object and the definition of that object. New detections are generally added to the database in a group associated with, for example, an image from the GPC1 camera. As new detections are loaded, they are compared to the objects already stored in the database. If an object is already found in the database within the match radius, the new detection is assigned to that object. If more than one object exists within the database, the detection is associated with the closest object. For most data sources, a match radius of $1\farcs0$ is used, but this may be adjusted in special cases.

Two tables carry the most important information about the astronomical objects in the database: *Average* and *SecFilt*. These two tables specify the main average properties of the astronomical object. The *Average* table includes the astrometric information (α, δ, μ_α, μ_δ, π) and associated errors, data quality flags for each object, links to the other tables, and a number of IDs, with one row for each astronomical object. The *SecFilt* table[6] contains average photometric information for a collection of filters. A given DVO instance has a specified set of filters for which average photometry is stored in the *SecFilt* table. The number and choice of filters for the *SecFilt* may be modified by the database administrator fairly easily, but the process of updating the database is somewhat expensive (~24 hr for the current PV3 instance). Thus, the choice is semistatic for a given DVO instance. In the case of the PV3 DVO instance, nine average bandpasses are defined: $g_{\rm P1}$, $r_{\rm P1}$, $i_{\rm P1}$, $z_{\rm P1}$, $y_{\rm P1}$, *J, H, K*, and $w_{\rm P1}$. The *SecFilt* table contains one row for each filter for each object, thus the PV3 DVO contains nine times as many rows as the *Average* table. The order of the table is fixed in relation to the *Average* table: row *i* of *Average* defines the object with photometry contained in rows $9i \rightarrow 9i + 8$ (*i* is zero counting).

The values stored in the *Lensing* table are used to calculate average values for each of these types of measurements in each filter. The *Lensobj* table stores the averaged KSB and radial aperture photometry for each of the five filters $g_{\rm P1}$, $r_{\rm P1}$, $i_{\rm P1}$, $z_{\rm P1}$, $y_{\rm P1}$. This table contains one entry per object per filter. The table is not generally stored unsorted as it is calculated after the full database is populated. The link from *Average* to *Lensobj* is defined by the fields *Average.offsetLensobj* and *Average.Nlensobj*. Each *Lensobj* row also includes the *photcode* for which the average lensing (and radial aperture) properties have been calculated.

Image Tables—Measurements that are loaded into DVO may be associated with a specific image (such as the measurements for a single chip from the GPC1 camera) or they may not have such an association (such as measurements from 2MASS, WISE, or externally supplied reference measurements). For data that are associated with an image, a subset of the information about that image (e.g., from the header of the FITS file) is used to populate a row in the DVO *Image* table. This table contains one row for each chip image known to DVO, with information such as the filter (*photcode*), the exposure time, the airmass, the astrometric calibration terms, the photometric zero point, etc. For GPC1 and other mosaic cameras, an additional row is defined to carry the projection and camera distortion elements of the astrometry model. As images are loaded into this table, they are assigned an internal ID (a running sequence in the table), stored in the field *imageID*. Images may also be assigned an ID derived from the external source of the image (field *externID*): in the case of the GPC1 images, this ID is defined by the processing mysql database and is guaranteed to be unique within the processing system. In the case of GPC1 exposures, the external image ID is set to the database field *chipImfile.chip_imfile_id*. A second field (*sourceID*) identifies which of the possible image-like tables supplied this image, guaranteeing the uniqueness of image IDs across the different IPP stages.

Other Tables—Other tables are used to track information used by the calibration system. This includes the complete set of flat-field corrections determined by the photometry

[6] The name *SecFilt* is a bit of a historical misnomer: originally, DVO was designed for a monochromatic survey and data for a single photometric band was maintained in the *Average* table. Later, DVO was adapted to a multifilter system and additional filters were added to the *SecFilt* (Secondary Filter) table. Eventually, the schema was normalized and all photometric data placed in *SecFilt*, with the Primary filter concept being dropped, but the name has since stuck.

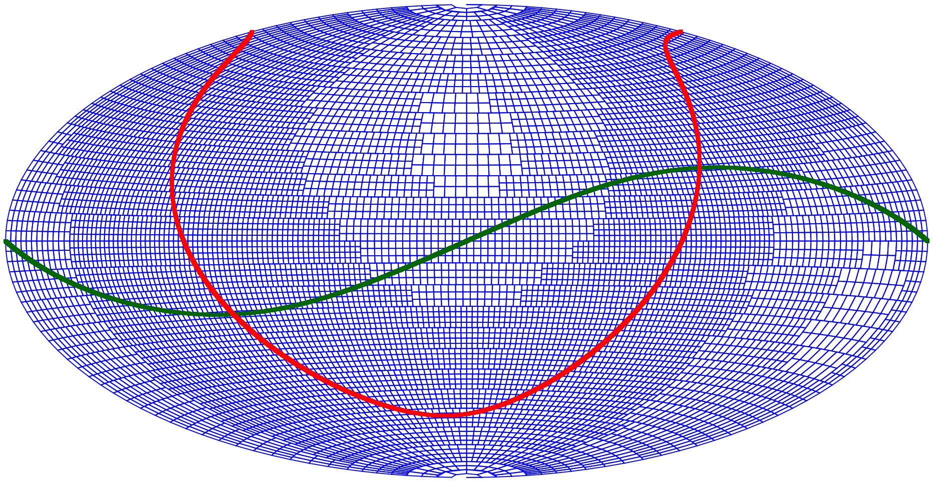

Figure 2. Level 3 sky partitioning. The blue grid shows the outlines of the different regions assigned to separate tables in the sky partitioning scheme. The Galactic plane is shown as a solid red line while the ecliptic is shown in green. This organization of the sky duplicates that used by the HST Guide Star Catalog (Jenkner et al. 1990).

calibration analysis and the astrometric flat-field corrections determined by the astrometry calibration analysis (see Paper V).

4.1.3. Sky Partition

Tables within DVO containing information about astronomical objects are partitioned on the basis of position in the sky: objects within a region bounded by lines of constant R.A., decl. are contained in a specific file. The boundaries and the associated partition names are stored in one of the supporting tables, *SkyTable*. This table contains the definitions of the boundaries for each sky region (*R_MIN*, *R_MAX*, *D_MIN*, *D_MAX*), the name of the sky region, an ID (*INDEX*, equal to the sequence number of the region in the table), and index entries to enable navigation within the table. The regions are defined in a hierarchical sense, with a series of levels each containing a finer mesh of regions covering the sky.

In the default used by the PV3 DVO, the partitioning scheme is based on the one used by the Hubble Space Telescope Guide Star Catalog files. Level 0 is a single region covering the full sky. Level 1 divides the sky in decl. into bands 7°.5 high, as defined by the HST Guide Star Catalog (GSC, Jenkner et al. 1990; Lasker et al. 1990). Level 2 subdivides these decl. bands in the R.A. direction, with spacing related to the stellar density. Level 3 divides these R.A. chunks into four to eight smaller partitions (see Figure 2). This level exactly matches the HST GSC layout and uses the same naming convention to identify the partitions: `n0000/0000`, etc. Level 4 further divides these regions by a factor of 16. In the *SkyTable*, a region at one level has a pointer to its parent region (the one which contains it) and a sequence pointing to its children (regions it contains). The *SkyTable* enables fast lookups of the on-disk partitions that map to a specific coordinate on the sky. In general, a single DVO will have the full sky represented with tables at a single level, although it is possible for mixed levels to be used. For the PV3 master database, the partitioning is at Level 4, resulting in ~150,000 regions to cover the full sky, of which ~110,000 are used for the PV3 3π data. The densest portions of the bulge contain at most ~300,000 astronomical objects in the database files, with an associated maximum of ~30 million measurements in these files. With the compression scheme described below, the largest database files are ~3 GB, which can be loaded into memory in 30 s on the processing machines that contain partition data.

The DVO software system allows the tables that are partitioned across the sky to also be distributed across multiple computers, which we call partition hosts. A single file identifies these partition hosts and the location of the database partition on the disks of that machine. The *SkyTable* contains elements to define by ID the partition host to which a set of tables has been assigned. Operations that query the database, or perform other operations on the database, are aware of the partitioning scheme and will launch their operations as remote processes on the machines that contain the data they need. For example, a query for data from a small region will launch subquery operations on the machines that contain the data overlapping the region of interest. These remote query operations will select the database information that matches the query request (i.e., applying restrictions as defined) and return the results to the master process. The results from the various partition hosts are then merged into a single result by the master process. When the parallel partitioning for a DVO instance is defined, the tables are randomly assigned to the partition hosts. As a result, queries which span more than a single partition are likely to spread the I/O load across a large number of machines. This parallelization is critical to querying and manipulating the enormous database on a reasonable timescale.

4.1.4. Object and Measurement IDs

Within the DVO system, certain integer fields are used to provide unique identifiers for measurements and objects. The original implementation of DVO was limited to 32 bit integer fields, but since the maximum number of objects and measurements was expected to be larger than 2^{32}, two 32 bit integer fields are joined together to make sufficiently large IDs.

In the table of objects (*Average*), the fields *objID* and *catID* together form a unique 64 bit integer value to identify the objects. The *catID* field is a sequence number for the sky partition table (the "catalog") in which the object is contained, while *objID* is an incrementing sequence number within that sky partition table. As long as no sky partition tables contain more that 2^{32} objects, these fields will not overflow. These two fields are included in the *Measure*, *GalPhot*, *StarPar*, *Lensing*, and *LensObj* tables to link the entries in those tables back their corresponding object. Note that *SecFilt* does not contain these ID fields; the rows in this table are maintained in the correct sequence to match the *Average* table entries.

The *Measure* table, containing the detections of objects from individual exposures or stack, or the (potentially nonsignificant) measurements from a warp, uses the 32 bit integer fields *detID* and *imageID* to uniquely identify each entry. The *imageID* is the running sequence number of the "image" (GPC1 OTA, stack, warp, or other other source of the measurement) in which the object was measured. The *imageID* is a value internal to DVO and is unique across all types of images. The *detID* field is a 32 bit integer giving the sequence number of the detection within the image. For images processed by the IPP (e.g., using `psphot`), the *detID* corresponds to the output field labeled as `IPP_IDET` in those data products. Because measurements from the same image may be spread across multiple sky partition tables, both *detID* and *imageID* must be used to uniquely identify a detection within the database.

In the *Measure* table, the field *averef* specifies the row number in the *Average* table of the associated object. The *Measure* table may be unsorted, in which case it is slow to find the measurements associated with a specific object (a full table scan is required, referencing *objID*). After the table is sorted and indexed, the *Measure* rows for a given object are grouped together. In this case, the fields *Average.measureOffset* and *Average.Nmeasure* define an index for the code to jump to the list of measurements for a single object. The field *Measure.imageID* defines the link from the measurement to the image that supplied the measurement.

DVO is also used to construct the unique object and detection IDs used by the PSPS. Within the PSPS, the field named *objID* in that database is used to allow valid joins between tables to select the different kinds of attributes of the same astronomical objects. This 64 bit integer ID is constructed based on the coordinates of the object, as described in Paper VI. In short, the digits of the R.A. and decl. coordinates are used to define a single 64 bit integer with spatial resolution of roughly 3 mas. These values used by this field are generated by the DVO system and stored in the *Average* table in the field *extID*.

Within the PSPS, the *Detection* table carries an ID that is unique for each measurement, equivalent to the DVO *detID*, *imageID* pair. In this case, the PSPS *detectID* is constructed from the MJD of the exposure, the number of the OTA (e.g., OTA64), and the detection sequence within the image to form a single unique 64 bit integer value. For detections from the stack images, the MJD is not unique, so a different rubric is used to define IDs for those detections. The field *XstackDetectID* (where "*X*" is one of *g*, *r*, *i*, *z*, *y*) is constructed from the GPC1 stack ID (*stackRun.stack_id*), the detection sequence within the stack image, and the same value used to define *sourceID* above. These two types of detection IDs are generated by the program `addstar` when the images and stacks are ingested into DVO.

4.1.5. DVO Data Storage

In the implementation of DVO used for the PV3 calibration analysis, the database tables are stored on disk using binary FITS tables. Each type of database table is stored as a separate file, or a collection of files for tables that are spatially partitioned. The binary FITS tables are compressed using the (to date) experimental FITS binary table compression strategy outlined by Pence et al. (2012). Table compression is an option in DVO; for the PV3 database, the large data volume (70 TB compressed) drove the decision to compress the tables.

The FITS binary table compression scheme uses a strategy similar to that used for FITS image compression (White & Greenfield 1999; Pence et al. 2000). The binary tabular data are compressed and stored in the "HEAP" section of the FITS table extension, with pointers to the compressed data stored in the regular data section. Each column in the FITS table is compressed as one (or more) blocks. The standard header keywords that describe the data column format (e.g., TFORM1) are replaced with keywords that describe the location and size of the compressed data in the HEAP section; the information about the uncompressed data is moved to a keyword with "Z" prepended (e.g., ZFORM1) and an additional field is added to define the compression algorithm (e.g., ZCTYP1). The column names (e.g., TTYPE1) and units (e.g., TUNIT1) are retained in their original form.

The compression algorithm can treat the entire column as a single block of data, or it may be broken into a number of chunks, each compressed in turn (this must be the same for all columns). Additional header information is added to describe the block sizes and information needed to describe the HEAP data section. The compression algorithms currently defined consist of the GZIP, RICE, PLIO, and HCOMPRESS (REFS). For GZIP, the compression algorithm may transpose the byte order before compression: for floating point data of a similar dynamic range, this choice may allow for better compression as each byte in the 4 or 8 byte floating point value is more likely to be similar to the same byte in other rows than to the other bytes of the same row value. This option is called `GZIP_2` in the FITS standard, as opposed to the standard order, `GZIP_1`. The DVO system can be set to specify the compression options for each column in the tables. In practice, we have chosen a default in which floating point numbers use `GZIP_2`, character strings use `GZIP_1`, and integers use `RICE`.

4.1.6. Addstar: DVO Ingest

Upon completion of the processing of each stage, the results of the photometry analysis are stored in a large number of individual catalog files as described in Paper IV. The data from these files are loaded into a DVO database to define the astronomical objects and to allow for calibration analysis. The program that loads the data into the DVO database is called `addstar` and is associated with the the ADDSTAR processing stage. The measurement catalogs generated by the CAMERA, SKYCAL, FULLFORCE, and DIFF stages are loaded into DVOs in this fashion, although not every measurement in each catalog is included in the master DVO that is constructed. For a particular reprocessing version, a single master DVO is constructed for the positive image stages (CAMERA, SKYCAL, FULLFORCE) and a separate one is constructed for the difference image analysis stage results.

The construction of the master DVO is performed in a hierarchical fashion. The individual catalogs are added to a mini-DVO, which is simply a DVO database defined over some subset of possible inputs. These mini-DVOs are then merged by stage into larger databases to construct a single master DVO database. In the process, an intermediate master DVO for each stage is generated. The `dvomerge` program is responsible for merging two DVO databases together. In the merge, astronomical objects are joined together using essentially the same rules as those used to associated detections into objects with one exception: the match radius may be chosen to be a different size depending on the data source. For example, when WISE data are merged with PS1 data, as discussed below, a match radius of 3″ is used due to the large beam size of the WISE telescope.

As of PV3, the process of merging mini-DVOs is not highly automated, requiring manual attention. The generation of the mini-DVOs is automated and managed by the ADDSTAR stage. Each catalog that is to be added to DVO has an entry created in the *addRun* database table. This entry notes which *stage* is the source of the catalog and links to the appropriate database table with the *stage_id* field. As some stages, such as the DIFF stage, create more than a single catalog, multiple entries with the *stage_id* are created, with the *stage_extra1* field containing an index to the individual components. The catalog specified by the entry is added to the target mini-DVO by the addstar program, updating the measurements in the appropriate DVO tables. When this completes, an entry containing the statistics of the job is added to the *addProcessedExp* table.

After the master DVO is constructed containing the PS1 data, data from other sources are also added to the database. For the PV3 DVO database, data were added from 2MASS, WISE, Gaia DR1, and Tycho. These external data sources are added by first generating a DVO database containing just the particular data source, then using the same DVO merging method to merge the external data DVO into the PS1 master.

4.2. Calibration Operations

Once the master DVO database has been constructed, high-quality astrometric and photometric calibrations can be calculated. The details of the calibration analysis are discussed in Paper V. We present a brief summary here.

Astrometric calibration consists of measuring and correcting systematic structures along with improved calibration of the transformations from chip to focal plane coordinates based on relative astrometry. These steps are performed iteratively. First, the relative astrometry analysis generates an improved solution without correction for systematic effects. Next, systematic effects are measured by querying the DVO database to determine the residual astrometric error as a function of some parameters. In the case of the PV3 astrometry analysis, systematic errors have been determined as a function of position in the camera (essentially an astrometric flat-field correction), as a function of the brightness of the star (the so-called Koppenhöfer effect; see Paper V), and as a function of airmass and color (differential chromatic refraction). Once the systematic errors have been measured, they are applied back to the measurements in the database. Within the DVO *Measure* table, the different types of systematic effects are included as separate offsets (in chip pixel coordinates) for each measurement. A single "corrected" version of the chip pixel coordinates is stored in which the systematic offsets are combined with the raw pixel coordinates for each measurement. After the systematic effects have been applied to the database, relative astrometry is again performed this time using the corrected positions.

Photometric calibration consists of determination of zero points for each exposure along with corrections for systematic effects. In this case, we rely on efforts of our external collaborators for the initial zero-point determination. The team at CfA downloaded the per-exposure catalog files ("smf files") and determined the zero points of those exposures which were believed to be obtained in photometric conditions. This process, called "ubercal," is described in detail by Schlafly et al. (2012) for the first (PV1) version and is based on the process of the same name used for SDSS calibration (Padmanabhan et al. 2008). In brief, photometric periods, with timescales of a large fraction of a night, are identified by a combination of automatic analysis and manual inspection. A single solution for all images in a given filter is determined to minimize scatter for individual stars. The free parameters in this solution consist of a single zero point and airmass slope for each photometric period along with a collection of flat-field offsets for several large time range ("flat-field seasons"). For the PV3 ubercal analysis, the flat-field offsets were determined on a 2 × 2 grid for each chip and five flat-field seasons were identified. The boundaries of the flat-field seasons were determined by independent inspection of the residuals observed in the Medium-Deep fields.

After the ubercal analysis of the photometric periods is completed, the determined zero points, airmass corrections, and flat-field terms are transmitted back to the IfA IPP team. These values are then ingested into the master DVO database. An initial relative photometry analysis is performed to tie the images without ubercal zero points to the ubercal system. Zero points from the ubercal analysis are not allowed to change, but zero points of the rest of the exposures are determined to minimize the photometric scatter for bright stars. These zero points are determined uniquely for each image. After an initial relative photometry analysis, the photometric residuals are used to determine a systematic correction as function of position in the camera. This correction is equivalent to the flat-field corrections determined as part of the ubercal analysis, but have much higher spatial resolution (40 × 40 corrections per chip) and are determined for only the full time range of PV3. This high-resolution flat-field correction addresses photometric variations due to spatial variations in the PSF due to the optics and low-level effects on the chips (see Paper V). After the systematic corrections have been determined and applied back to the database, a final relative photometry analysis pass is performed.

4.3. Construction of the PSPS Database

The publicly visible Pan-STARRS database is hosted by the Space Telescope Sciences Institute through their MAST. The underlying database at MAST is a copy of a database generated at the IfA by the PSPS. Both MAST and IfA versions of the PSPS are implemented using a collection of Microsoft SQL server instances as the underlying database engine. As in DVO, the tables holding the large volume of measurements are distributed across the different computers based on their location on the sky. Unlike DVO, the spatial distribution uses slices that span all R.A. values for a narrow range of decl. on a single computer. The PSPS design and implementation are described in some detail in Paper VI.

The construction of the PSPS version of the PS1 database starts once the PS1 photometry and astrometry measurements have been calibrated within the DVO system. The construction takes place in several stages, described in detail in Paper VI. We summarize those steps here.

The first stage of constructing the PSPS database consists of the generation of small files called "batches" which contain a complete set of measurements for a small chunk of the database tables. The program that is responsible for the construction of these batches is called ipptopsps. Several different types of batches are generated, relating to the different types of tables in PSPS. The details of the batch construction depend on the batch type.

One type of batch consists of measurements from the individual exposures. These batches are generated based on the output catalog files generated at the CAMERA stage ("smf files"). The ipptopsps program loads the complete set of measurements and metadata from the smf catalog file, then queries the DVO database for calibration parameters related to that smf file. The batch is constructed by applying the photometric calibrations to the raw flux measurements in the smf file.

A second type of batch file consists of the measurements related to the stack images. Again, ipptopsps starts with the output catalog files, selects the appropriate calibration information from the DVO, and applies the calibration data to the raw measurements in the stack catalog files.

A third type of batch file consists of average properties of the astronomical objects in the DVO database. Unlike the other two batch types, this operation is performed solely via queries to the DVO database. The complete set of average measurements for objects in a single DVO spatial partition is loaded by ipptopsps and used to generate the batch file.

As the batch files above are generated, the PSPS system can run in parallel to ingest the measurements from these batch files. PSPS downloads in sequence the batch files as they are generated and unpacks the data. The data are then loaded into a small-scale version of the PSPS database, using the full schema. After a large chunk of batches have been loaded, the resulting tables are then merged into the master PSPS database. After another large chunk of data has been merged into the master PSPS database, a large-scale copy of the database is made internally to provide a long-term backup and to aid in error recovery.

Once the full PSPS database has been loaded, or a complete set of batches for a given batch type, the entire database is copied to STScI where it can then be made visible either to the Pan-STARRS Science Consortium or to the wider public.

5. Operations and Automation

5.1. Pantasks and Parallel Processing

5.1.1. Pantasks

Sections 3 and 4 describe the analysis steps that take place in the Pan-STARRS data analysis systems. Individually, these steps appear as commands which could be run by a user within the UNIX environment of the PS1 data system. The processing database (Section 3.1) provides the logical links to relate the results of one analysis stage to another. In order to make a complete system which can run automatically, it is necessary to have a software system that can use the contents of the processing database to generate the commands corresponding to the analysis stages. This system needs to (1) regularly examine the database to find items from stages that are ready to be processed, (2) have rules that define how to construct the appropriate commands, (3) cause those commands to be executed within the processing system, (4) monitor the active processing jobs for completion, and (5) check on the results of those commands and update the processing database as needed. Within the Pan-STARRS IPP, the top-level management of these operations is performed by the program called pantasks.

The core capability of the pantasks program is to take a collection of "tasks" which describe the concept of a command which might be run and to regularly generate new commands based on that concept. The "tasks" are defined using the opihi scripting language (also shared by DVO and other user-interactive programs within the IPP).

Pantasks repeatedly checks each task in an attempt to generate a new command: we say pantasks attempts to "execute" the task. Tasks may specify the time between execution attempts, with a 1 s default.

Each task must at a minimum define a command to generate. Commands may be static or dynamic. For a task with a static command, the command is explicitly defined in the task block (see code example in Figure 3) and is identical each time the task is executed. A dynamic command is defined within a special block of the task, called task.exec. This block is a snippet of code (in the opihi language) that is run each time the task is executed. The task.exec code may refer to variables or other data structures defined by the opihi language within the pantasks environment. Within a single pantasks instance, all opihi variables and data structures have global context by default (i.e., all are visible to all tasks). Within the context of an opihi macro (equivalent of a function call), variables may be locally scoped. Other data structures (see below) are global and must be protected with name space choices.

Within the task.exec macro, the command to be run is defined by the script. Once the task.exec macro exits successfully, the defined command is then added to the list of jobs to be run within the UNIX environment. Jobs may be run in one of two ways: locally or via the parallel processing system. The task, or the task.exec macro, uses the host command to define how to run the job. If the host is set to "local," then the job is run in the background by pantasks itself (using the C execvp function). Otherwise, the job is sent to the parallel processing system to be run on another machine within the cluster. If the host is set to the special value "anyhost," then the parallel processing system is allowed to choose the processing computer arbitrarily. Any other value is taken to be the DNS name of the computer on which this job should run. The host may (optionally) be required for the command, in which case the parallel processing system must ensure that the job only runs on the specifically named computer. Otherwise, the parallel processing system may choose to redirect the command to another computer using its own rules, e.g., to balance processing load across the cluster.

When the task.exec macro is run, the code may choose (e.g., based on tests of some global variables) to exit the macro with an error condition. In this circumstance, no job is produced by the task. The task will be tried again the next time it is executed. This feature allows for the user to set processing blocks that depend on some external tests. For example, some task may check external network connectivity and set a variable based on the network status; other tasks may then choose to wait until the network is available before attempting to run.

Other task options exist to control the system behavior in detail. These options may be dynamically reset by the task.

```
task        example.static.task
  host      local
  command   ls /tmp
  periods   -exec 5.0
  npending  1
  stdout    /data/local/example.task.output
  stderr    /data/local/example.task.errors
end
```

Figure 3. Example of a simple static task in the opihi-based scripting language used by pantasks. In this example, pantasks would run a single instance of the command (`ls/tmp`) every 5 s, sending the stdout and stderr to the listed files.

`exec` macro. Some options control the number of jobs, such as limiting the number of currently outstanding jobs for a given task, or limiting the total number generated. Other options can be used to control the time when jobs of a certain task are allowed to run. It is also possible to specify the UNIX "nice" level at which the job is run when it is executed. Finally, individual tasks may be disabled while the system is still running.

5.1.2. Pcontrol

Jobs that are generated by `pantasks` may be run locally on the machine running `pantasks` or they may be distributed across many machines in the computing cluster. The parallel processing system used by `pantasks` is an independent software system called `pcontrol`.[7]

This program is based on the same `opihi` shell language used by `pantasks`. The two programs communicate via a shared set of pipes: `pantasks` sends commands to the standard input of the `pcontrol` and accepts back responses on the standard output and standard error.

`pcontrol` maintains a list of jobs (commands to be run) and a list of hosts (computers on which a job could be run). Jobs may have one of several states: pending (ready to run), running (jobs that are running), exit (job has completed), busy (job is being checked by `pcontrol`), crash (job has exited with a signal, normally `segv`).

Similarly, the hosts may also have one of several states: off, down, busy, idle, etc. A single host can accept a single job at a time. Multiple host instances corresponding to the same machine may be specified allowing a single computer to run more than one simultaneous job.

During operation, `pcontrol` accepts new jobs from `pantasks` and adds them to the list of jobs to execute. It also accepts from `pantasks` the names of computers on which it is allowed to run those jobs.

5.1.3. Pclient

When `pcontrol` is provided with the name of a computer, it will attempt to make an connection to that machine via ssh. When a connection is made, the remote shell is used to run a special interface program call `pclient`. This program accepts command lines from `pcontrol` and is responsible for executing the individual commands in the local shell environment. A single ssh connection to a remote host keeps a single `pclient` shell running for a somewhat arbitrarily long time, executing many shell commands as needed. This architecture avoids wasting overhead making the ssh connection to the remote machine each time a command is executed, allowing for rapid execution of many commands. As a result, a single job within the IPP architecture is allowed to be very light and short running if needed.

After `pcontrol` sends a job (commands) to a specific `pclient`, it checks back occasionally to see if the command has been run and executed. If it has finished, then `pcontrol` will query for the exit status, the standard output and standard error streams from the command (where do these go, back to `pantasks`?), with the results associated with the job statistics. At that point, the `pclient` on the remote machine is ready to accept a new job from `pcontrol`. If any jobs are pending in the list of jobs known to `pcontrol`, it will send those jobs to any machines which are idle.

While `pcontrol` interacts with the many remote machines, it occasionally interacts with `pantasks` to report the results from the jobs it has been monitoring. `Pantasks` occasionally requests a list of the completed jobs. It then requests the status information for each completed job, including the standard error and standard output. As `pantasks` receives this completion information, the jobs are removed from the list managed by `pcontrol`. Thus, `pcontrol` maintains at most a modest list of jobs that are "in flight," leaving all interpretation work to `pantasks`.

At the `pantasks` level, the tasks define how `pantasks` should use the exit status and output products from each job. For example, the stderr and stdout may be specified to go to a file (with static name or name dependent on the specific job). The task may define a different behavior depending on the exit code from the job.

The `pantasks` program can be run as a standalone program that presents an `opihi` shell interface to the user when it is started. This mode is useful for testing as all errors are reported back to the `opihi` shell. However, when the user exits the shell, the `pantasks` instance exits, shutting down `pcontrol` and all remote client connections. In standard operations, `pantasks` is run in a client server mode. The server runs continuously in the background, and multiple users may connect via the `pantasks_client` program. Users can the send commands to the server to load scripts, add parallel hosts, check status, and start or stop the `pantasks` operations.

5.1.4. Pantasks Scripts: ippTasks

`Pantasks` provides an environment in which commands can be generated and extensive parallel processing managed. The details of how to implement the different stages of IPP processing are captured in a collection of scripts written for `pantasks` in the `opihi` language. In general, each stage is defined by an associated script collected together under the `ippTasks` collection. While each script has its own details, there are a number of common elements.

Most stages consist of two related tasks: a `load` task, which is responsible for querying the processing database to identify entries to be processed, and a `run` task, which is responsible for managing the processing of the individual entries.

The `load` task for a particular stage generates `load` jobs, which query the processing database via a dedicated database interface program (see the discussion of `ippTool` in Section 5.5 below) for a list of processing stage entries that are waiting to be run. The `load` jobs are executed on the host running the `pantasks` server. Only one of each type of `load` job is permitted to run simultaneously, preventing race conditions.

[7] Alternatives are possible: e.g., `condor` has been experimentally integrated with `pantasks` for tests.

The results from the database query job are stored in an opihi data structure called a book within the pantasks environment. Each row in the result set is saved to a separate entry within the book. These books are a hierarchical associative array indexing the entries (pages to continue the analogy) to be accessed via a particular key. Keys for most stages are a combination of the stage ID and an identifier for the individual component for the job that will be executed. For a given row in the result set, each column in the row is stored as a separate line on the page, identified by the database column name. An additional line, the *pantasksState*, is added so pantasks can manage the processing of the job that will be generated by this page. When the page is first generated, the *pantasksState* is set to INIT, indicating that this page is a new addition to the book. Once all new pages have been added, the task then scans the book for any pages with *pantasksState* set to DONE and removes them from the book, as these represent jobs that have finished.

The associated run task generates jobs constructed from the collection of pages in the book. The task examines the book and selects the first available page with *pantasksState* of INIT. The task uses the information in the page to construct the appropriate command-line (e.g., lines in the page may include input file names and output file names for the specific item in the database). The resulting command becomes a job in the pantasks collection of jobs. Most IPP analysis stages specify that the jobs are then sent to pcontrol for a parallel process. Before task generates the job, the *pantasksState* is set to RUN so a future execution of the task will not attempt to rerun this specific job.

Upon completion of the job, it is necessary to update the processing database with the results, specifically indicating in the database that the job has completed and if it was successful. Within the IPP, this responsibility is left to the program which ran the analysis. IPP analysis steps normally consist of two main elements: a C-language program to do the data analysis work and a supporting Perl script that performs the database update upon completion. Upon completion, the pantasks RUN task is responsible for updating the status within the book, but not within the processing database. This split keeps the interactions at the pantasks level relatively light, leaving the overhead of the database interaction within the job running on one of the computing machines in the cluster.

In addition to these tasks, most stages have a revert task paired with the run task. These tasks run infrequently and generate jobs that perform an operation on the processing database to clear jobs that have failed with one of the ephemeral failure modes (see the discussion in Section 3.1). This step allows these failures to be cleared from the system, allowing those jobs to be scheduled again.

Similarly, some stages have advance tasks that update the primary table to indicate that all of its components are complete. For many of the early stages of the pipeline (the CHIP through WARP stages), this advance task also adds an entry into the database table for the next stage of processing for the exposure being considered. This step allows the data to process automatically from stage to stage without intervention.

The IPP processing database is used to manage all versions of an analysis for all analysis stages. In addition to the regular processing of the nightly data products, there may be large-scale reprocessing analysis tasks or tests of various kinds. It may be necessary for a test analysis of a particular item to use a different version of the processing software from the regular nightly analysis (for example, when testing a new algorithm for release). A mechanism is needed to manage these different processing attempts of the same items. With the IPP, this is accomplished with an extra field, label, for each processing stage. Within the load, revert, and advance tasks discussed above, the query to the processing database for new items is restricted to a set of user-defined labels. A given instance of pantasks will be supplied a set of labels that are then applied to all tasks managed by that pantasks. For example, the pantasks that manages the nightly processing of the basic science analysis stages (CHIP-WARP, STACK, DIFF) is supplied with several labels that correspond to the different kinds of observations being performed. In this way, the analysis of the nightly observations is kept separate from other processing attempts.

5.2. Stage Automation

Beyond of the basic sequence of CHIP to WARP, there is no single natural "next step." For example, a stack can be generated with any number of input warps; a difference image can be generated between a warp and any one of many other warps or stacks. Without a single sequence, more complex and sophisticated decisions much be made.

For nightly processing of data obtained at the summit, this is handled by a set of "nightly science" tasks and an associated ippScript. These scripts have a well-defined and restricted set of goals: to ensure that difference images are generated for each exposure (either by pairing together warps or pairing warps with predefined stacks), that nightly stacks are generated for MD fields, and that the nightly stacks are also differenced against an appropriate reference.

For the warp–warp difference images, pairing warps together is simplified by the observing strategy in which the same pointing is observed multiple times in a night. By limiting to warp–warp pairs from the same pointing, the problem is significantly reduced from the arbitrary case.

Queuing these warp–warp difference images is done by first examining the set of all exposures that have been taken at the summit on the current night of observing and querying information from each stage up through WARP stage. These exposures are grouped by *filter* and *object*, which is a unique identifier for each telescope pointing on the sky. Exposures in each group are then sorted by increasing observation date (*dateobs*).

The database results for each stage (CHIP-WARP) are checked to ensure that the selected exposures have been successfully processed for all stages through WARP. Exposure groups are ignored until all exposures have either been processed through warp or have failed with a bad quality, meaning the exposure (or portion) cannot be processed. Failed exposures are rejected. The remaining exposures are then paired sequentially, with the final exposure ignored in the case of an odd number of accepted exposures. Exposures paired in this way are sent to the DIFF analysis stage. Nightly processing also ensures that the difference image analysis is run using the warps in comparison to the reference stack images generated for the full 3π region.

Once observations have been completed for the night (signaled by the end-of-night dark exposures that are taken each morning after the telescope closes), and the script has generated all DIFF pairs that can be made with the above rules, a second pass is performed, this time with the exposures in each group sorted by decreasing observation date. This change in ordering allows exposures that were excluded due to an odd number of exposures to be paired with the exposure closest in time (with the exposure that was previously first ignored). Exposure pairs in which at least one exposure does not have a preexisting difference image are queued for difference image analysis.

The nightly stacks are queued based on checking that a minimum number of complete WARP entries exist for each filter and field. For the nightly MD processing, this minimum number was set to eight exposures, as this is the number of exposures taken for each field. Once this number was reached, no more exposures are expected, so STACK database entries can be queued from the WARP entries. Again, failures and weather can reduce the number of usable exposures. If no stack could be made for a given MD field with the minimum number of inputs by the time of the end-of-night darks, stacks are generated using whatever exposures are available. Nightly processing also ensures that the difference image analysis is run on these nightly stacks using a predefined reference stack.

The automatic nightly processing ensures that data are processed as soon as they are downloaded from the summit, reducing the lag between an observation and the reduced data.

The other processing task that requires automation is the reprocessing of the entire 3π Survey, as the size of the data set makes it challenging to organize the analysis manually. To manage large-scale analyses, the "large area processing" (LAP) task and script are used. The first stage of LAP generates an entry in the *lapSequence* table defining a new reprocessing. After this, individual *lapRun* entries can be queued that define a *filter* and a *projection_cell* to be considered. These projection cells correspond to the projections used by the warp tessellation to define the skycells (see Section 3.7), which have tangent plane centers matching those in the warp tessellation. For the 3π Survey analysis, a *projection_cell* is a unit of sky defined to be a square 4° on each side which has a single tangent plane projection (Paper III).

Once this entry is defined, it is populated with all exposures (stored in the *lapExp* table in the database) that are located within 5° of the center of the projection cell included. This radius ensures that any exposure that overlaps the projection cell will be included. Once the exposures have been added, the other exposures within the same sequence are checked to see if a CHIP stage entry has been generated, and if so, the *chip_id* for that entry is saved into the *lapExp* as well. This linkage ensures that each exposure is only processed once. If no entry is found, a new CHIP entry is queued for processing. The task periodically checks the status of the exposures in each *lapRun* entry, and if they have all completed the WARP stage, then a STACK is queued for each skycell contained within the *projection_cell*.

5.3. Nebulous

5.3.1. Motivation and Concept

A major concern recognized early in the Pan-STARRS project is the challenge of storing and managing the large volume of data generated by the GPC1 camera. The `Nebulous` system was designed to aid in this process. `Nebulous` is not a file system per se, but only a method of tracking the locations of files within the file system and of tracking duplicate copies of the same file. The core of `Nebulous` is a mysql database that tracks "storage objects," the equivalent concept of a file within the system. Each storage object may be associated with a number of copies of the actual files on the disks in the storage system (called "instances"), which are also recorded by the database. In the IPP cluster, the file instances are stored on a collection of computers with substantial disk partitions shared via network file system (NFS).

`Nebulous` also explicitly tracks the different computers on which the file instances are stored. This allows the system to expose files to the user only on machines that are currently active in `Nebulous`. If, for example, a storage computer crashes or needs to be taken offline, the machine can be made unavailable in `Nebulous`, in which case only instances on other machines will be supplied to users.

This localization is also useful for allowing the IPP processing to target processing to computers based on the location of the data. For example, all raw images from a specific chip in the camera could be stored on a specific computer (for at least one of the instances). All of the analysis stages that interact with that chip could then be preferentially targeted to be run on that computer. The localization in `Nebulous` and the host-targeted processing in `pantasks` can therefore work together to encourage processing to require only local disk access, reducing the I/O local on the network infrastructure. In the early stages of the Pan-STARRS project, this was important because network bandwidth was an expensive resource. In practice, the as-built IPP has had sufficient network bandwidth that this targeting was not completely required. In practice, due to the timing of hardware acquisition, occasional hardware failures, and other organizational details, targeted processing has only been used to a moderate degree within the Pan-STARRS cluster.

All of the IPP low-level C-based processing programs (e.g., `ppImage` and `ppStack` interact with `Nebulous` to find existing files and to create new files. The supporting Perl scripts also interact with Nebulous to perform file instance duplication as needed and to check for the existence of required input files and expected output files.

5.3.2. Implementation Details

The user interfaces to Nebulous consist of command-line programs as well as APIs in both C and Perl. The basic user commands to interact with Nebulous are to (1) query the database for an existing storage object and find a valid file instance associated with that object; (2) create a new storage object, which instantiates an empty file that can be opened for writing; (3) replicate an existing storage object to create more file instances; (4) cull a single file instance of storage object from the cluster; and (5) remove a storage object and ensure that all file instances are removed. The file handles returned for newly created instances can then be opened for reading and writing data to that instance.

For the Nebulous users, the identifier of a storage object is a unique string with the form of a UNIX file path: e.g., a/b/c/file. When a program creates a new file in `Nebulous`, it supplies a URI of the form `neb://HOST.VOL/PATH/FILE`. The HOST and VOL(ume) specifiers are optional, allowing a file to be created on a specific computer (HOST) and disk (VOL). The path and file name portions become the identifier and are recorded in the `storage_object` table in the `ext_id` field. A storage object entry is then created in the database for this ID, and an instance of the file created on the specified node. If the host is unspecified, or if the specified volume is full, then a host is chosen at random from available nodes.

Files are stored on specific computers in a `Nebulous` directory or directories on that computer. In the IPP system, the top-level Nebulous directories are usually placed at the root of the storage device as mounted on the machine, in a subdirectory named `nebulous`. Beneath the top-level directory are 256 subdirectories with names of the form 00—ff (i.e., two-digit hexadecimal number). Each subdirectory has 256 subdirectories with the same naming scheme.

The file name of an instance in Nebulous is deterministic and derived from the `ext_id`: the `ext_id` is hashed using the SHA-1 function, and the first four hexadecimal digits of this hash are separated into two two-digit strings and used as the top- and second-level directory location for the disk file. The instance table in the Nebulous database includes a unique auto-incrementing index to provide a unique SQL ID for each instance. Under the subdirectory identified above, the disk file name is by appending the database instance id with a string derived from the `ext_id`: forward slash characters are replaced in the name with colons so the string can represent a file in the UNIX file system. For the example URI above, this results in a file located on disk in a location like `/data/HOST.VOL/nebulous/d5/d8/42.PATH:FILE`. This file-naming structure has the benefit of providing redundancy between the file name on disk and the instance in the database.

Nebulous tracks additional information beyond just the storage objects and the associated instances. As mentioned above, the storage volumes are tracked to provide a link between a top-level nebulous directory and the computer that contains that directory. The locations and mount points for the actual NFS storage are listed in the *volume* table. This table contains columns indicating if the volume should be used for reading (*available*) and writing (*allocate*). As noted above, Nebulous will not return a file to the user if the storage volume is marked as not *available*. If a storage volume is marked as not to be *allocate*-ed, then new storage objects will not generate instances on that volume, but existing instances may be supplied to the user.

Another column, *xattr*, is used to control the behavior of this volume, with specific values used to denote desired behavior. For instance, the volume may be marked to be used only for backup, in which case it will not be used to store an instance by default, but will be used preferentially if an instance noted as a backup when it is generated. Alternatively, a volume may be marked as permanently unavailable, and should be ignored in most contexts. This latter option allows the system to retain the memory of hardware that has been retired (and potentially to retain information about instances that were previously on such machines).

In addition to the static table describing the volumes, a second dynamically generated table, *mountedvol*, lists those volumes that are currently visible and accessible from the `Nebulous` database server. This table also lists the total and currently available disk space on each volume, allowing the `Nebulous` load balancing routines to prioritize those volumes with large unused disk space before filling the volumes with only small amounts remaining. This table is updated every 10 to 20 minutes, after a scan of each of the volumes listed in the *volume* table.

The *cabinet* table organizes the individual volumes into "cabinets," a concept loosely based on the physical arrangement of the storage servers in the data center. These cabinets are used to prevent the replication of a storage object within a group of volumes where all instances of the object could be taken offline by a single failure. Because servers within a given cabinet in the data center share a common set of power delivery units (PDUs), it is important to ensure physical distance between replicated copies to guarantee that a temporary failure of one of the cabinet PDUs does not significantly impact processing.

The nebulous user APIs do not interact directly with the Nebulous mysql database. Instead, they interact with one of several computers with an Apache web server. Interactions with the Apache server are performed using the Simple Object Access Protocol (SOAP) interface, while the Apache servers interact directly with the mysql database server. This architecture avoids the overhead of setting up and tearing down the mysql connection for each Nebulous command, instead using only the low-latency SOAP communications.

The Nebulous database currently (2017 July) contains information about 5,560,533,654 file instances for 3,543,240,981 storage objects. All raw data, along with permanent products such as catalogs and the current versions of full-sky stacks, are replicated to ensure at least two copies exist in case of hardware failure. Based on the most recent database ID values (which are unique and never reused), this corresponds to roughly half of all the storage objects and file instances ever created, due to the transient nature of many pipeline products.

5.4. Datastore Repositories

Transferring data between the IPP and other parts of the Pan-STARRS system is generally accomplished via a "datastore," an HTTP service that exposes data in a common form. One of the main datastores used by the IPP is the one located at the summit. This datastore exposes a list of the exposures obtained since the start of the PS1 operations. Requests to this server may restrict to the latest by time. Each row in the listing includes basic information about the exposure: an exposure identifier (e.g., o5432g0123o; see Paper I for details), the date and time of the exposure, the telescope-commanded pointing, the filter and exposure time, and the observation comment for that exposure. The row also provides a link to a listing of the chips associated with that exposure. This listing includes a link to the individual chip FITS files as well as an md5 checksum. Systems that are allowed access may download the raw chip FITS files via HTTP requests to the provided links.

The IPP also uses datastores to provide access to its own data products. The detections identified in the DIFF stage images are organized by the PUBLISH stage, which writes output files containing those detections to a datastore that is monitored by the MOPS (Denneau et al. 2013), which analyzes the detections to identify asteroids. Separate datastores are also used by the DISTRIBUTION stage to provide access to data products to the Pan-STARRS Science Consortium members.

5.5. ippTools and ippScripts

The IPP relies on a number of common libraries and programs to handle various tasks that are shared between multiple stages of the processing. These subsystems are described in this section, to provide an introduction to these essential components that underlie the rest of the pipeline.

As shown above, the pantasks tasks rely on ippTools calls for database queries. Each stage has an appropriate ippTool, allowing the database interaction to be governed by a fixed set of inserts and queries. Isolating the database interaction in this way adds a layer of validation before queries are executed and ensures that all database modifications are handled in a uniform fashion.

In addition to simple queries and updates to entries already in the database, the various ippTools programs can also be used to define new processing runs for a stage. Again, using predefined queries wrapped in a program allows the options to be parsed to ensure that any new processing run definitions are appropriately restricted in scope, reducing the chance that mistakes will fill the database with many unwanted jobs.

Connecting the pantasks parallel processing environment to the actual IPP analysis programs are a series of ippScripts written in Perl. These scripts are what are actually executed by pantasks, with command-line options provided based on the database query performed there by the load task. These options are combined with configuration information stored in ippconfig recipe files.

The appropriate recipe is selected from the configuration information, based on the source camera of the data to be processed, and optionally modified by the *reduction* field in the database. These optional *reduction* entries provide a way to group a nonstandard set of processing options together across multiple stages by selecting a recipe that is not the default.

With the set of configuration options and database entries for the data to be processed, the ippScript checks the input files that will be used and confirms that a valid copy of each is available from the Nebulous system. For stages that have a large number of inputs (such as the STACK stage, which requires images, masks, variance maps, and detection catalogs from each of the potentially large number of WARP stage inputs), the input files are organized into temporary input list files, formatted in an appropriate way for the analysis program that will process them.

The script also sets up an output log file for this processing run, ensuring that any status information from either the script itself or the underlying analysis is stored on disk. The majority of this information is identical between calls to the script, but for rare failures of the analysis programs, retaining this information allows for such problems to be diagnosed and repaired.

The command line for the main analysis program is constructed based on the database values, the recipe options, and the input file names. The analysis program is then executed, and any failure reported back to the parent pantasks process. In the standard case of the analysis completing successfully, the script checks that the expected output products were generated, preventing hidden I/O errors from being a problem with subsequent processing of those output products.

One output product that must exist is the stats file, which is generated by the analysis program and contains statistics about the processing, including such things as the image background level, the fraction of masked pixels, and the version numbers of the analysis program. This stats file is then parsed by the ppStatsFromMetadata program, which uses the information within to generate command options for the ippTool program to ensure that these statistics are included in the database row that is created in the secondary database table for the individual component processed.

5.6. psLib and psModules

Underlying all of the analysis programs are the psLib and psModules C libraries. The more fundamental psLib library defines the internal data structures that are used (arrays of arbitrary type, vectors, images, and hash tables among others), manage data access (particularly for FITS images and tables), and organize string and error handling in a uniform fashion. This library also contains fundamental math operations, covering vector statistics, matrix operations, and function minimization. Common image operations such as binning, interpolation, and convolution are also provided, as well as the methods to to write JPEG versions of the data for visualizations. Finally, general coordinate transformations are provided between planes and projections of spheres.

The functions provided by psModules have more focused scopes that are nevertheless still shared between multiple programs. The isolation of source objects is included, providing the organization of detections that is used in the psphot photometry analysis (Paper IV). The PSF matching required for STACK and DIFF stage image combinations is as well. The unification of configuration options between config files on disk and the options specified on the command line is handled by psModules functions, as is the construction of data structures in memory to represent the astronomical camera based on the header information in the input file. The functions to generate and apply the detrend corrections to the data are also provided by this library.

6. IPP Hardware Systems

6.1. Kihei Processing Cluster

The majority of all Pan-STARRS processing has been performed on the dedicated IPP cluster, located in Kihei on Maui. This cluster was originally located at the Maui High Performance Computing Center (MHPCC), a United States Air Force research laboratory center managed by the University of Hawaii. This site was chosen based on the original development funding provided by the Air Force Research Labs (see Paper I for more details). Once the Air Force funding stopped being a significant driver for Pan-STARRS, the cluster was physically moved from the MHPCC to a similar nearby computing center located at the Maui Research and Technology Center.

The computing cluster comprises three main types of computers, with a variety of individual specifications due to the cluster being assembled from multiple generations of purchases. The data storage nodes contain several petabytes of storage space that are used to store both the raw exposure data downloaded from the telescope as well as processed data products. These nodes are also used to do processing and have jobs targeted to them in an effort to reduce the network I/O demands (see Section 3.4 for more on this process).

These storage nodes are not fully capable of completing all processing on the short timescale necessary for each night's worth of data. To increase the processing capability, we have 212 "compute" nodes that have small amounts of local storage, but are able to provide additional processing power. In addition to the direct processing of image data, these nodes are also used to manage the Nebulous file interface, as well as controlling the job scheduling for the processing.

The final type of computer in the cluster is the database servers. These computers have large memory capacity and high-speed disk access (originally fast spindle spinning disks, now migrated to SSDs) are used to store and manage both the IPP gpc1 and Nebulous databases. In addition to the main master servers, we have duplicate servers used as database replicants, which allow for quick switching from the main to backup servers in case of a hardware issue that cannot be resolved immediately. The IPP uses a set of three computers to host the Nebulous mysql database and live backups. A second set of computers is used to host the processing database and backups.

6.2. Los Alamos National Laboratory

In order to increase the processing rate for the 3π PV3 data, we partnered with Los Alamos National Laboratory to gain access to the Mustang supercomputer. The supercomputer comprises 3088 processing nodes, each with 12 cores and 64 GB of RAM. The processing nodes do not have significant local disks, but are connected to multiple petabyte scale scratch disks. Job management is controlled by the Moab HPC system,[8] which schedules resource requests among users, allocating processing nodes to satisfy jobs and terminating those jobs if they exceed their scheduled time limit.

This system is part of the lab's "Turquoise" network, allowing it to be used for research projects that do not handle sensitive data. It is, however, subject to stricter access controls than are in place at the main IPP processing cluster. Login sessions are terminated after 12 hr, requiring new sessions to be initiated regularly. Network access is also filtered, with only SSH connections allowed between the IPP cluster and Los Alamos. This restriction removes the ability for the processing to contact the IPP processing database directly.

To work around this, additional steps were needed to ensure efficient use of the computing resources. A periodic poll of outstanding tasks was done on the IPP cluster, using the information stored in the database and those tasks assigned to a processing bundle. Each component task in the bundle was then checked to identify the set of input files needed to complete the task, the commands necessary to complete the task, and the set of output files that should be generated if the task completed successfully. Once this information had been generated for all tasks, the component lists were merged, and the Moab job control file was constructed.

The control file contains the resource requests for the job, as well as the commands to be executed to complete it. The resource request was calculated based on the number of tasks included in the job bundle N_{tasks} and scaled by the expected execution time (t_{task}) and computational intensity of the component tasks (S_{task}). For a given job bundle, an initial estimate of the number of compute nodes needed is simply nodes $= S_{\mathrm{task}} * N_{\mathrm{tasks}}/12$. To ensure that jobs were not prematurely terminated, we attempted to design the requested job processing time to be 25% longer than the expected time to complete the component tasks. Based on the initial node count, we calculated the request time as $t_{\mathrm{request}} = \left\lfloor 1.25 \frac{\mathrm{nodes} * t_{\mathrm{task}}}{\mathrm{nodes}_{\mathrm{max}}} \right\rfloor + 1$, where $\mathrm{nodes}_{\mathrm{max}}$ is the maximum number of nodes that can be requested in a single job (1000 for Mustang). Table 4 contains the cost values used for the various IPP processing stages.

Table 4
Cost Values for Remote Processing

IPP Stage	t_{task} (s)	S_{task}
CHIP	150	2
CAMERA	1700	2
WARP	110	2
STACK	1500	6
STATICSKY	7200	6
FULLFORCE	300	2

Once the preparation for the job is complete, the input and output file lists, the task list, and the job control file are transferred via secure copy protocol (SCP) to the Mustang cluster. Local tasks are then initiated on the user interface nodes to SCP the input files onto the scratch storage disks if they do not already exist. Once all the input files have been copied, the job is submitted to Moab for scheduling. The Moab interface is periodically polled to determine the job status, and after it has completed, the results are retrieved in a similar way. Local tasks again SCP the output products, but copy the results back to the IPP cluster.

In addition to the standard output products, "dbinfo" files are constructed as part of the job execution. These files contain database update commands to ensure that the IPP processing database has the correct entries for the tasks that were remotely executed. These commands are executed after confirming that all retrieved output products are present.

Approximately half of the CHIP through WARP processing for the PV3 reduction was performed on Mustang, with 201,040/375,573 of the CAMERA stage products reduced there. Only processing through the STACK stage was attempted, although with a smaller fraction of the total compared to the CAMERA stage, with 290,257/998,886 being produced at Los Alamos. One reason for this decrease is that due to the memory constraints on the Mustang processing nodes, we were unable to run stacks with more than 25 inputs there. Stacks with larger numbers of inputs overflow the memory of the processing node, and as they do not have disk space available for use as virtual memory, cause the machine to hang until the job time limit is reached. These stacks were instead processed on the regular IPP cluster, where hosts with sufficient memory were available.

6.3. UH Cray Cluster

In 2014 December, the University of Hawaii installed a 178 compute node Cray supercomputer on the main Manoa campus. As part of the initial commissioning of this computer, Pan-STARRS was invited to use this resource in 2015 February, roughly corresponding with the completion of the initial processing of the CHIP through STACK processing. Although the number of nodes was much smaller than that available on Mustang, the nodes were more robust, with 20 cores and 128 GB of memory. The scratch data storage was somewhat smaller than at Los Alamos, with only a single 600 TB volume. We had the unique ability to rapidly deploy to the UH Cray, using almost all nodes for IPP processing as other users at the university were designing code. This rapid deployment was made possible by the similarity of the Slurm[9]

[8] http://www.adaptivecomputing.com/

[9] https://slurm.schedmd.com/

scheduler and tools to those used by Moab (although the UH Cray has a smaller nodes$_{\max}$ of 10).

The UH Cray was used to do processing for the STATICSKY stage, running approximately half of that photometry (101,528/200,720). We were also able to run part of the FULLFORCE photometry there as well, although more had to be run on the IPP cluster as other users started to utilize the system, with 168,685/994,890 runs processed there.

7. Conclusion

We began the development of the IPP in early 2004, soon after the initial funding for the construction of the Pan-STARRS telescopes was awarded to UH. The landscape of the software and computing world has changed in a number of ways. Some of the decisions we made at the beginning have held up well while in other cases we would probably make a different choice today.

One choice we made early on was to develop new code for the data analysis programs. This choice was driven partly by some of our experiences with the existing major systems of the time. We were advised by those with close experience with the SDSS data analysis code base against attempting to modify that system for our purposes. It was also our opinion that the IRAF suite of packages was not well suited to the large-scale automated pipeline needed for the Pan-STARRS data. The Pan-STARRS data analysis rate was going to surpass previous astronomical projects, and the cameras (with 60 detectors each of 64 cells) would have an unprecedented level of complexity. The original survey was intended to run for 10 yr, so long-term supportability was also a priority. With these design constraints in mind, we decided to develop a new code base that would be able to address the data rate and complexity.

In our design, we have tried to make the analysis programs as generic as possible, with all instrument-specific details addressed in the configuration files. Our implementation has been generally successful in this regard. The `ppImage` program contains most of the highly specific detrending details, with much more limited camera-specific features needed in the configuration files for `psastro` and `pswarp`. This generalization of the software has made it easy to run the full analysis pipeline on other cameras, both for testing and for other science analysis projects. We have used the full IPP analysis system for data from the CFHT Megacam and CFH12K cameras as well as the Subaru Hypersuprime Camera. The generalization made is relatively simple to add the second telescope and camera (PS2 + GPC2) to the regular processing when they came online for science operations in 2018.

In retrospect, the additional design and coding effort needed to keep the system general were worthwhile and have paid off. However, if we were to start from scratch today, we would probably choose to adapt the LSST pipeline for our use because it has been developed with some of the same constraints.

One early choice was to use standard C for analysis programs and to use Perl as a wrapper language. We considered other language choices, including C++ and Python. Our choice of C over C++ has not held up well: we would have done well to have the modern object-oriented features of C++, some aspect of which we have imitated in our C coding style. The choice of Perl over Python has also fared poorly. At the time, Python was fairly new and did not have the widespread acceptance it has today. The capabilities available within the Python environment would have allowed us to include interesting visualization and other high-level analysis options. It is also easier to hire astronomers with good Python coding skills than Perl coding skills.

We also find that maintaining support for our Perl code has been a challenge: changes to the Perl language syntax and changes in externally supported Perl modules have required significant effort to keep our code compatible with the changes. It is not obvious that Python would obviate that particular problem, however.

One important aspect of the design of the IPP is to use a single database to manage the processing stages, with regular queries to the database to choose the tasks that are ready to proceed. Other choices were possible. In some pipelined processing systems, completed jobs trigger the next processing step. For example, `ppImage` or its wrapper (`chip_imfile.pl`) could have been responsible for launching the `psastro` analysis, eliminating the `pantasks` manager entirely. Alternatively, a manager process could be responsible for launching the next processing step when one step has completed. For example, `pantasks` could note when the `ppImage` jobs were complete and launch the `psastro` analysis. Both of these choices can potentially result in lower latency because the next step is in principle run immediately when the previous step is completed.

Our design choice has two important advantages: first, error and failure recovery are trivial. If one of the many programs fails or is interrupted, the system can easily notice and retry the job. In a triggered system, a failure of one stage could mean the trigger never happens. Some external cleanup system would need to be implemented to check for the failures and relaunch. The second advantage of our design is that each analysis stage is highly independent and can thus be flexibly run in different ways. For example, alternative test systems can run in parallel with the nightly operations system, using the outputs of the nightly processing by simple changes to the queries used to select the elements for an analysis stage. In addition, it is easy to add new stages because they do not need to be injected into the standard processing manager (`pantasks`).

The main challenge related to this database-managed design is that the database can become a bottleneck. If the queries used to select the processing items become too large and too slow, the whole system can be slowed down. Care must be taken to avoid poorly implemented queries, and in some cases, the queries need to be restricted. For example, if too many items are queued for processing at one time under the same processing label, the associated queries can bog down. This issue is one of the reasons we manage the large-scale processing with the LAP system as it provides a method to automatically limit the scale of the queries. In addition, it is critical that the database hardware be sufficiently powerful to keep up with the demand from the processing system.

Finally, the choice to use Nebulous as a file management system is ambiguous. When we began this project, the existing cluster file systems did not seem to match the level of our project. Some were still very much in an early development state (e.g., GFS from Red Hat), while others seemed designed for much larger-scale systems, with very expensive hardware requirements (e.g., Lustre). The requirements for the file system for Pan-STARRS are somewhat different from the large-scale computing clusters such as those used by the US national labs. Because the data processing is very parallel, we do not have any strong requirements on data access concurrency. In theory, we could have simply used NFS and made backup copies of the files using some simple name-convention rules. We decided to implement the Nebulous system to allow the targeted analysis and to automate the replication of the data. In retrospect, the system has succeeded in these goals and has behaved reliably. However, the support level has been somewhat high, especially when we have needed to migrate large amounts of data within the cluster. If we were to start from scratch today, it is possible that some of the existing cluster file systems would address our needs within our budget.

Since the Pan-STARRS 1 telescope first came online in 2007, this telescope has obtained 1.43 million exposures with GPC1, amounting to a raw data volume of 4.32 PB. The Pan-STARRS IPP has archived and processed these images on the fly to produce discoveries of transient events and hazardous asteroids in real time. The IPP has been used to perform several reprocessings of large fractions of the science exposures to produce a well-calibrated data release of the 3π Survey data. To date, and including repeated analysis, the IPP has processed 2.1 million exposures, detecting 900 billion sources in those exposures (real and otherwise!). The Pan-STARRS data-processing system represents a real example of astronomy data processing on the very large scale.

The Pan-STARRS1 Surveys (PS1) have been made possible through contributions of the Institute for Astronomy, the University of Hawaii, the Pan-STARRS Project Office, the Max-Planck Society and its participating institutes, the Max Planck Institute for Astronomy, Heidelberg, and the Max Planck Institute for Extraterrestrial Physics, Garching, The Johns Hopkins University, Durham University, the University of Edinburgh, Queen's University Belfast, the Harvard-Smithsonian Center for Astrophysics, the Las Cumbres Observatory Global Telescope Network Incorporated, the National Central University of Taiwan, the Space Telescope Science Institute, the National Aeronautics and Space Administration under grant No. NNX08AR22G issued through the Planetary Science Division of the NASA Science Mission Directorate, the National Science Foundation under grant No. AST-1238877, the University of Maryland, and Eötvös Loránd University (ELTE) and the Los Alamos National Laboratory.

ORCID iDs

Eugene A. Magnier https://orcid.org/0000-0002-7965-2815
K. C. Chambers https://orcid.org/0000-0001-6965-7789
H. A. Flewelling https://orcid.org/0000-0002-1050-4056
M. E. Huber https://orcid.org/0000-0003-1059-9603
C. Z. Waters https://orcid.org/0000-0003-1989-4879
L. Denneau https://orcid.org/0000-0002-7034-148X
P. W. Draper https://orcid.org/0000-0002-7204-9802
K. W. Hodapp https://orcid.org/0000-0003-0786-2140
R. Jedicke https://orcid.org/0000-0001-7830-028X
N. Kaiser https://orcid.org/0000-0001-6511-4306
N. Metcalfe https://orcid.org/0000-0001-9034-4402
C. W. Stubbs https://orcid.org/0000-0003-0347-1724
R. J. Wainscoat https://orcid.org/0000-0002-1341-0952

References

Alard, C., & Lupton, R. H. 1998, ApJ, 503, 325
Bowell, E., Koehn, B. W., Howell, S. B., Hoffman, M., & Muinonen, K. 1995, AAS/DPS Meeting, 27, 1057
Chambers, K. C., Magnier, E. A., Metcalfe, N., et al. 2016, arXiv:1612.05560
Denneau, L., Jedicke, R., Grav, T., et al. 2013, PASP, 125, 357
Flewelling, H. A., Magnier, E. A., Chambers, K. C., et al. 2020, ApJS, 251, 7
Green, G. M., Schlafly, E. F., Finkbeiner, D. P., et al. 2014, ApJ, 783, 114
Green, G. M., Schlafly, E. F., Finkbeiner, D. P., et al. 2015, ApJ, 810, 25
Hernitschek, N., Schlafly, E. F., Sesar, B., et al. 2016, ApJ, 817, 73
Hodapp, K. W., Siegmund, W. A., Kaiser, N., et al. 2004, Proc. SPIE, 5489, 667
Jenkner, H., Lasker, B. M., Sturch, C. R., et al. 1990, AJ, 99, 2082
Kaiser, N., Squires, G., & Broadhurst, T. 1995, ApJ, 449, 460
Lasker, B. M., Sturch, C. R., McLean, B. J., et al. 1990, AJ, 99, 2019
Lee, C.-H., Koppenhoefer, J., Seitz, S., et al. 2014, ApJ, 797, 22
Lee, C.-H., Riffeser, A., Koppenhoefer, J., et al. 2012, AJ, 143, 89
Magnier, E. A., & Cuillandre, J.-C. 2004, PASP, 116, 449
Magnier, E. A., Schlafly, E. F., Finkbeiner, D. P., et al. 2020a, ApJS, 251, 6
Magnier, E. A., Sweeney, W. E., Chambers, K. C., et al. 2020b, ApJS, 251, 5
Onaka, P., Tonry, J. L., Isani, S., et al. 2008, Proc. SPIE, 7014, 70140D
Padmanabhan, N., Schlegel, D. J., Finkbeiner, D. P., et al. 2008, ApJ, 674, 1217
Pence, W., Seaman, R., & White, R. L. 2012, arXiv:1201.1340
Pence, W., White, R. L., Greenfield, P., & Tody, D. 2000, in ASP Conf. Ser. 216, Astronomical Data Analysis Software and Systems IX, ed. N. Manset, C. Veillet, & D. Crabtree (San Francisco, CA: ASP), 551
Saglia, R. P., Tonry, J. L., Bender, R., et al. 2012, ApJ, 746, 128
Schlafly, E. F., Finkbeiner, D. P., Jurić, M., et al. 2012, ApJ, 756, 158
Tonry, J., & Onaka, P. 2009, in Proc. Advanced Maui Optical and Space Surveillance Technologies Conf., ed. S. Ryan (Kihei, HI: Maui Economic Development Board), E40
Tonry, J. L., Stubbs, C. W., Lykke, K. R., et al. 2012, ApJ, 750, 99
Waters, C. Z., Magnier, E. A., Price, P. A., et al. 2020, ApJS, 251, 4
White, R. L., & Greenfield, P. 1999, in ASP Conf. Ser. 172, Astronomical Data Analysis Software and Systems VIII, ed. D. M. Mehringer, R. L. Plante, & D. A. Roberts (San Francisco, CA: ASP), 125