Scientific Programming 15 (2007) 75–81
IOS Press

Parallel MOPEX: Computing mosaics of large-area Spitzer surveys on a cluster computer

Joseph C. Jacob[a,*], Peter Eisenhardt[a] and David Makovoz[b]
[a]*Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA*
[b]*CACI International Inc., Lanham, MD 20706, USA*

Abstract. The Spitzer Science Center's MOPEX software is a part of the Spitzer Space Telescope's operational pipeline that enables detection of cosmic ray collisions with the detector array, masking of the corrupted pixels due to these collisions, subsequent mosaicking of image fields, and extraction of point sources to create catalogs of celestial objects. This paper reports on our experiences in parallelizing the parts of MOPEX related to cosmic ray rejection and mosaicking on a 1,024-processor cluster computer at NASA's Jet Propulsion Laboratory. The architecture and performance of the new Parallel MOPEX software are described. This work was done in order to rapidly mosaic the IRAC shallow survey data, covering a region of the sky observed with one of Spitzer's infrared instruments for the study of galaxy clusters, large-scale structure, and brown dwarfs.

Keywords: Astronomy, image mosaicking, cluster computing, parallel processing, Spitzer space telescope

1. Introduction

NASA's Spitzer Space Telescope [1], launched as SIRTF (Space Infrared Telescope Facility) in 2003, contains science instruments capable of obtaining spectra and images in the infrared wavelengths, which are particularly sensitive to cool, dusty, or extremely distant objects. Infrared instruments reveal regions of the Universe that are hidden from optical telescopes; when observing from space they are free of background contamination from atmospheric effects. One such instrument is Spitzer's Infrared Array Camera (IRAC), which provides 5.12 $\times$ 5.12 arc minute (256 $\times$ 256 pixel) images at 4 wavelengths, 3.6, 4.5, 5.8, and 8 microns [2].

IRAC is a general purpose camera used for a wide variety of science programs, but the specific program supported by the research reported in this paper is the IRAC Shallow Survey [3], an 8.5 square degree survey of the Boötes field of the National Optical Astronomy Observatory (NOAO) Deep Wide-Field Survey (NDWFS) with Spitzer's IRAC. A mosaic of the survey region is shown in Fig. 1. Science objectives for the IRAC shallow survey are, among others, to detect galaxy clusters at $z > 1$ (z is red-shift, an indicator of distance, and thus age), to better understand the evolution of large-scale structure out to $z \sim 2$, and to detect brown dwarfs. Although few brown dwarfs are known, these "failed stars" are thought to outnumber normal stars, but lack sufficient mass to sustain nuclear fusion, and hence are too cool to emit optical radiation.

Construction of astronomical image mosaics is widely recognized as an essential processing step in accomplishing the objectives of astronomical research programs like the IRAC Shallow Survey. Finding rare objects like brown dwarfs and galaxy clusters at $z > 1$ requires both observing a large volume of space (best accomplished by mapping large areas) and highly reliable detections (best accomplished by repeated observations of the same region). In addition, mosaics

*Corresponding author: Joseph C. Jacob, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Mail Stop 126-347, Pasadena, CA 91109-8099, USA. Tel.: +1 818 354 0673; Fax: +1 818 393 6141; E-mail: Joseph.C.Jacob@jpl.nasa.gov.

ISSN 1058-9244/07/$17.00

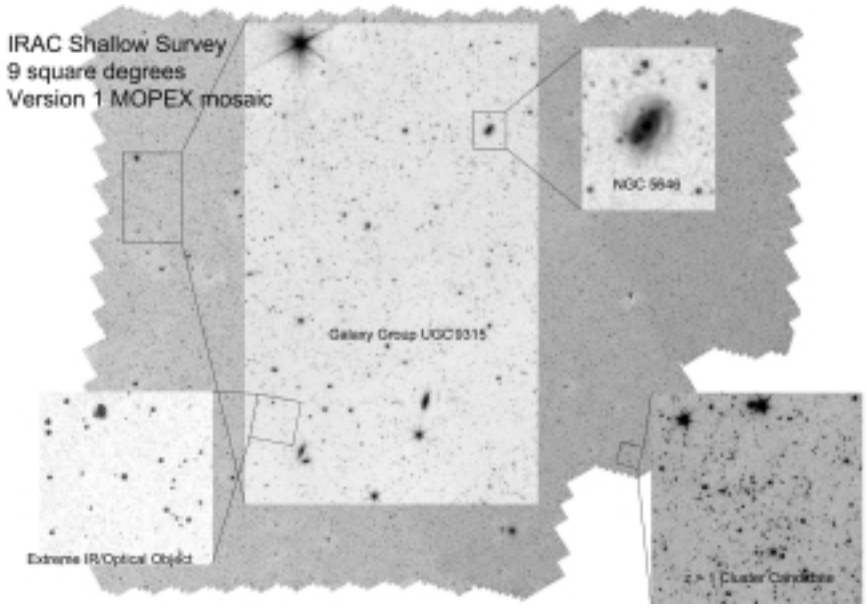

Fig. 1. IRAC Shallow Survey mosaic.

are needed for objects that fall at the boundary of multiple image frames, or are too large to fit in a single frame. A number of astronomical image mosaicking software packages exist, including Montage [4], SWarp [5], yourSky [6], and MOPEX [7]. MOPEX is specifically designed for the Spitzer Space Telescope data processing and is the software package that was parallelized in the research reported here.

MOPEX (MOsaicking and Point source EXtraction) is an astronomical image-processing package developed at the Spitzer Science Center (SSC) at the California Institute of Technology. MOPEX can be used to construct mosaics with outlier detection, extract point sources from images, match background levels across image collections, and perform pointing refinement (i.e., resolve discrepancies between positions of point sources matched in overlapping pairs of images captured with slightly different telescope pointing). The MOPEX processing steps of mosaicking and outlier detection are of particular interest in this paper; mosaicking for the reasons described above, and outlier detection because, for the space-based astrophysical imaging done by Spitzer, it is a critical step in filtering out the contamination from cosmic ray collisions with the detector array.

MOPEX has been used to generate an initial reduction consisting of a single mosaicked image in each wavelength band for the IRAC Shallow Survey, from which catalogs have been extracted and distant candidate galaxy clusters identified (see Fig. 1). The MOPEX runs which produced these version 1 survey mosaics required approximately 70 hours on a dual processor 3GHz Pentium computer with 2GB of RAM.

Speeding up the mosaicking and outlier detection processing steps directly supports large area Spitzer Space Telescope imaging surveys, in particular the IRAC Shallow Survey, by facilitating the identification of field brown dwarfs in the 17,076 input frames of this survey. Identification of the handful of brown dwarfs expected (a major scientific objective of the IRAC shallow survey) remains problematic at present because the images contain artifacts from a significant number of cosmic ray collisions with the detector array. Optimization of MOPEX parameters to improve cosmic ray detection is challenging because of the long processing times. Improved performance is necessary in order to optimize cosmic ray detection and produce a version 2 set of IRAC shallow survey mosaics, to facilitate searches for brown dwarfs and other rare objects.

In this paper we report on our experiences in parallelizing the image mosaicking and outlier detection portions of MOPEX version date “101504”. The resulting software, Parallel MOPEX, was run on a 512-node (1,024-processor) Xeon cluster computer at NASA’s Jet Propulsion Laboratory (JPL). In Section 2 we review the serial MOPEX software before describing the new Parallel MOPEX in Section 3. Section 4 shows performance results for the Parallel MOPEX software running on a cluster computer. Finally, a summary is provided in Section 5.

Table 1
The main MOPEX modules for cosmic ray rejection and mosaicking

Processing Stage	MOPEX Module	Description	Data Products
Input		User provided inputs.	Input images, mask images
	snestimator	Compute uncertainty (sigma) images if they aren t provided as input.	Uncertainty images
	fiducial_image_frame	Compute output mosaic parameters (sky position, orientation, size, resolution, etc).	Fiducial Image Frame (FIF) file
	medfilter	Subtract background from the input images	Median filtered images
Interpolation	mosaic_int	Interpolate input images to the FIF	Interpolated input images, uncertainty images, and coverage maps
Outlier Detection (Single Frame)	detect_radhit	Detect single frame spatial outliers	Single frame outlier maps
Outlier Detection (Multi-frame, Temporal)	mosaic_outlier	Detect multi-frame temporal outliers	Multi-frame outlier maps
Outlier Detection (Multi-frame, Spatial-Temporal)	detect	Compute detection maps of point sources and cosmic ray hits	Outlier detection maps
	mosaic proj	Interpolate detection maps	Interpolated detection maps
	mosaic_covg	Compute coverage map	Coverage map
	mosaic_dual_outlier	Detect multi-frame spatial-temporal outliers	Dual outlier detection maps
	level	Correct dual outlier detections	Corrected dual outlier detection maps
	mosaic_rmask	Create outlier mask (combining the 3 outlier detection methods)	Outlier mask
Reinterpolation	mosaic_reinterp	Reinterpolate those pixels containing outliers	Reinterpolated input images, uncertainty images, and coverage maps
Coaddition	mosaic_coadd	Sum interpolated images into tiles	Coadded image tiles uncertainty tiles, Tile coverage maps
Combining	mosaic_combine	Stitch together images into a single mosaic image	Mosaics of input images, uncertainty images, and coverage maps

2. Review of MOPEX architecture

The MOPEX software includes a mosaicking toolkit with the processing stages, modules, and data products listed in Table 1. The Interpolation stage maps the coordinate systems of the input images to the output mosaic coordinate system, called the Fiducial Image Frame (FIF). Cosmic ray hits are detected in the Outlier Detection stage. MOPEX includes one single frame and two multi-frame algorithms for outlier detection. The outlier detection algorithms can be fine-tuned for a particular data set by setting a number of parameters. Reinterpolation is a repeat of the interpolation stage for pixels containing outliers detected in the Outlier Detection stage. The Coaddition sums interpolated images into tiles, and these are then "stitched together" into a single mosaic image in the Combining stage.

Each MOPEX module is a compiled C or C++ executable. These modules are called by a top level Perl script, mosaic.pl, which manages the entire workflow. A configuration file called the "name list file" contains keyword-value pairs that assign values to parameters or specify which MOPEX modules are to be run.

3. Parallelization approach

Since the mosaics created by MOPEX are science data products, any modifications to the MOPEX software need to be carefully validated for science quality. In order to ensure that the parallelization of the MOPEX software doesn't degrade its usability or the science quality of the mosaics it produces, the paral-

Table 2
The Parallel MOPEX modules and sequence of operations. A barrier synchronization is done after each parallel step. Parallelism is achieved by either assigning images or output mosaic tiles to processors, as indicated

Parallel Step	Parallel MOPEX Module	Description	Data Parallelism Approach
1	fiducial_image_frame	Compute output mosaic parameters (sky position, orientation, size, resolution, etc).	None (run serially)
	snestimator	Compute uncertainty (sigma) images if they aren t provided as input.	Round robin assignment of images to processors
2	medfilter	Subtract background from the input images	Round robin assignment of images to processors
	detect_radhit	Single frame spatial outlier detection	
	mosaic_int	Interpolate input images to the FIF	
	detect	Compute detection maps of point sources and cosmic ray hits	
	mosaic proj	Interpolate detection maps	
3	tiler_imglist	Create tiled image lists and FIFs for those modules parallelized by mapping output mosaic tiles to processors	None (run serially)
	tiler_fif		
4	mosaic_covg	Compute coverage maps (one map for each overlapping image-tile pair)	Round robin assignment of mosaic tiles to processors
	mosaic_dual_outlier	Detect multi-frame spatial-temporal outliers (one map for each overlapping image-tile pair)	
	mosaic_outlier	Detect multi-frame temporal outliers (one map for each overlapping image-tile pair)	
5	img_combiner	Merge tiled coverage maps, outlier maps, and dual outlier maps from phase 4 into complete maps for each image	Round robin assignment of images to processors
6	level	Correct dual outlier detections	Round robin assignment of images to processors
7	mosaic_rmask	Create outlier mask (combining the 3 outlier detection methods)	Round robin assignment of images to processors
	mosaic_reinterp	Recompute the interpolation for pixels containing outliers	
8	mosaic_coadd	Sum interpolated images into tiles	Round robin assignment of mosaic tiles to processors
9	mosaic_combine	Stitch together image tiles into a single mosaic image	None (run serially)

lelization was designed with the following guidelines in mind:

- The parallelization should be as transparent to the users as possible (i.e., there should be minimal changes to calling syntax, parameters, and other usage).
- The software modifications should be as minimal and as isolated as possible.

3.1. Software modifications

The following changes were made to parallelize the original serial MOPEX software:

1) Added a C module (tiler_imglist) to partition an image list into a set of "tiled" image lists, where each tiled list contains those images that overlap a tiled subset of the output mosaic.

2) Added a C module (tiler_fif) to partition the FIF or Fiducial Image Frame (the main MOPEX data structure for specifying the region of coverage and coordinate system for a mosaic) into a set of tiled FIFs.

3) Added a C module (img_combiner) to perform a pixel-by-pixel sum of two FITS images, ignoring NANs.

4) Added a top level C wrapper program (mosaicMPI), which starts up the MPI processes, synchro-

nizes the processors between critical steps in the mosaic construction, and serves as a top-level user interface for the parallel MOPEX. Care was taken to make absolutely no changes to the user interface except modifying the serial MOPEX calling sequence in order to run the parallel software with MPI (see below).

5) Modified the MOPEX module mosaic_dual_outlier to only compute results for the pixels that overlap the specified FIF.

6) Converted the original Perl wrapper script, mosaic.pl, to a new script, mosaicMPI.pl, which has the same calling sequence except for the addition of two arguments that give the number of processors (-p NumProcs) and MPI rank of the calling processor (-r rank). The new mosaicMPI.pl wrapper script runs all of the MOPEX modules in parallel.

3.2. User interface modifications

Care was taken to make absolutely no changes to the user interface except changing from the serial MOPEX calling sequence of:

mosaic.pl <standard MOPEX arguments>

to:

mpirun.lsf mosaicMPI <standard MOPEX arguments>.

A name list file that works with serial MOPEX should work with parallel MOPEX.

However, an optional name list parameter was added to allow specification of a directory to be used for some of the MOPEX intermediate output. If the parallel nodes have their own local disks that are not shared across all the nodes, this name list parameter can be set to point to the local disk in order to offload some of the I/O load from the global shared disk to the local disks owned by the individual nodes. The performance results in Section 4 show that using the local disks in this way can significantly improve performance for this data-intensive application.

3.3. Parallel design

The inter-processor communication and synchronization in Parallel MOPEX were implemented using the Message Passing Interface (MPI) [8]. In this section we describe the elements of our design related to multi-processor data partitioning and synchronization. Table 2 provides a summary of the sequence of processing steps in Parallel MOPEX, including the data partitioning mechanism used in each step.

3.3.1. Data partitioning

In general, the parallelization approach was to run the same MOPEX modules in the same order as in the serial version, but with each processor working on a subset of the input data. The partitioning of the input data across processors was done in different ways for the different modules, depending on the nature of the underlying algorithm. For instance, for some of the modules, an equal number of input images were assigned to each processor. Other modules were assigned all of the input images that contribute to the computation of some subset (or "tile") of the output mosaic. Parallel MOPEX includes additional modules called tiler_imglist and tiler_fif, that are used to create tiled image lists and FIF files, respectively. These tiled files are computed as the overlap of the original image list or FIF and an output mosaic tile.

The following modules are inherently serial and are run on one processor: fiducial_image_frame and mosaic_combine. The following modules are parallelized by round robin assignment of images to processors: snestimator, mosaic_int, medfilter, detect_radhit, detect, mosaic_proj, level, mosaic_rmask, and mosaic_reint. The following modules are parallelized by spatially tiling the output mosaic space and assigning tiles to processors in a round robin assignment: mosaic_coadd, mosaic_covg, mosaic_dual_outlier, and mosaic_outlier.

3.3.2. Synchronization

The parallel software functions by running the new parallel mosaicMPI.pl script several times, with a "barrier" synchronization among the processors done after each *step* (or pass through the script). Each step uses its own name list file that is automatically generated, and consists of independent computations that are well suited for parallel processing on a cluster computer. The synchronization between steps is necessary because data dependencies inherent in the MOPEX mosaicking workflow makes it necessary for the processors to share information before starting certain module's computations.

4. Performance

The mosaics of the IRAC Shallow Survey field used in the benchmarks reported in this paper required about 1.25 GB of input data and about 87 GB of intermediate and final output. The first mosaic of the IRAC Shallow Survey field was computed with the serial MOPEX software on a 3 GHz Pentium computer with 2 GB

of RAM, and took about 70 hours of wall clock time for mosaics in all four IRAC bands. Porting the serial software to a single Xeon processor of the cluster computer, with no parallelization, enabled the IRAC Shallow Survey mosaic to be computed in 16.7 hours per wavelength. In the rest of this section, performance results for Parallel MOPEX (based on MOPEX version dated "101504") are indicated using this new 16.7-hour (1003.6 minutes) single-processor time as the baseline.

The main limitation to the performance of Parallel MOPEX is the bandwidth to disk. The current configuration of the JPL cluster has local disks on each node, which are visible to the node only, and NFS-mounted scratch disks that are visible to all nodes and shared by all users. Parallel MOPEX uses both the local and shared disks for a mosaic run. Name list file parameters are used to specify the directory on each of these disks to use for output. In general, performance is improved as disk I/O is moved off of the shared disks to the local disks.

Figure 2 shows a plot of the wall clock time required to compute a mosaic with Parallel MOPEX for channel 1 of the 16,014 $\times$ 16,579 pixel IRAC shallow survey mosaic with single precision floating-point pixels. This mosaic was constructed from 4,251 input images. The plot shows wall clock time in minutes as a function of number of nodes on the cluster. Note that even though each node on the cluster has 2 processors, these results are for using only one processor per node. The fastest run was accomplished on 32 processors in 121.6 minutes for a speedup of 8.3 over the single-processor result.

To illustrate the difference in performance of the globally shared, NFS-mounted disks and the private, local disks, a run was also done on 32 nodes using only the shared disks. This degraded the performance to 194.5 minutes, or a speedup of 5.2 over the single-processor version. Worse yet, a couple attempts at a 64 node run were done using only the shared disks, but the I/O node on the machine suffered an unrecoverable failure both times and had to be rebooted.

Clearly, the NFS-mounted shared disks do not effectively scale up to such a large number of simultaneous writes from many processors. JPL has a plan to upgrade the cluster with a true parallel file system. We expect this will result in significant improvements in performance for this data-intensive application. In addition, further speedup would be expected by further reducing the amount of data being written to the shared disks.

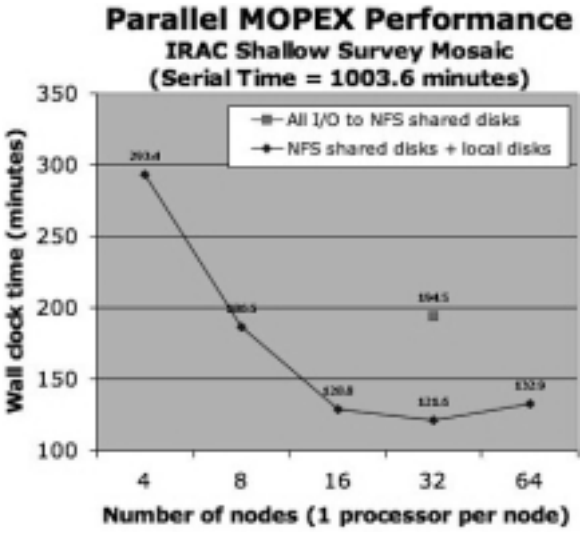

Fig. 2. Wall clock time for running Parallel MOPEX on a cluster computer with the indicated number of Xeon processors.

5. Summary

MOPEX is the Spitzer Science Center's software for cosmic ray rejection, image mosaicking, and point source extraction. This paper described Parallel MOPEX, a new version of MOPEX that can run the cosmic ray rejection and mosaicking parts of MOPEX in parallel on a cluster computer. Parallel MOPEX was designed to make minimal changes to the original serial code base and user interface, while achieving the speedup of parallel computation on the target platform, a 1,024 Xeon processor cluster computer at NASA's Jet Propulsion Laboratory. The MOPEX modules were run in 9 steps separated by barrier synchronization, and the modules were parallelized by distributing either input images or output mosaic tiles onto processors. The new Parallel MOPEX software was used to mosaic the IRAC Shallow Survey dataset with a speedup of 8.3 over the single-processor version. The bottleneck for this data-intensive application is the disk subsystem bandwidth, which had to be shared among all of the processors. JPL is installing a parallel file system on the cluster computer, which is expected to significantly improve performance for this application. Plans for the future include migrating the Parallel MOPEX to the new parallel file system, and updating the underlying code base to the latest version of MOPEX.

Acknowledgements

The support of the JPL Research and Technology Development program and of the Spitzer Science Center

is gratefully acknowledged. Ted Hesselroth and Iffat Khan of the Infrared Processing and Analysis Center (IPAC) at the California Institute of Technology provided expert assistance with the detailed workings of the MOPEX software. This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] M. Werner, T. Roellig, F. Low, G. Rieke, M. Rieke, W. Hoffmann, E. Young, J. Houck, B. Brandl, G. Fazio, J. Hora, R. Gehrz, G. Helou, B. Soifer, J. Stauffer, J. Keene, P. Eisenhardt, D. Gallagher, T. Gautier, W. Irace, C. Lawrence, L. Simmons, J. Van Cleve, M. Jura and E. Wright, *The Spitzer Space Telescope Mission*, 2004, *The Astrophysical Journal Supplement Series* **154** (September 2004), 1–9.

[2] G.G. Fazio , J.L. Hora, L.E. Allen, M.L.N. Ashby, P. Barmby, L.K. Deutsch, J.-S. Huang, S. Kleiner, M. Marengo, S.T. Megeath, G.J. Melnick, M.A. Pahre, B.M. Patten, J. Polizotti, H.A. Smith, R.S. Taylor, Z. Wang, S.P. Willner, W.F. Hoffmann, J.L. Pipher, W.J. Forrest, C.W. McMurty, C.R. McCreight, M.E. McKelvey, R.E. McMurray, D.G. Koch, S.H. Moseley, R.G. Arendt, J.E. Mentzell, C.T. Marx, P. Losch, P. Mayman, W. Eichhorn, D. Krebs, M. Jhabvala, D.Y. Gezari, D.J. Fixsen, J. Flores, K. Shakoorzadeh, R. Jungo, C. Hakun, L. Workman, G. Karpati, R. Kichak, R. Whitley, S. Mann, E.V. Tollestrup, P. Eisenhardt, D. Stern, V. Gorjian, B. Bhattacharya, S. Carey, B.O. Nelson, W.J. Glaccum, M. Lacy, P.J. Lowrance, S. Laine, W.T. Reach, J.A. Stauffer, J.A. Surace, G. Wilson, E.L. Wright, A. Hoffman, G. Domingo and M. Cohen, *The Infrared Array Camera* (*IRAC*) *For The Spitzer Space Telescope*, *The Astrophysical Journal Supplement Series* **154** (September 2004), 10–17.

[3] P.R. Eisenhardt, D. Stern, M. Brodwin, G.G. Fazio, G.H. Rieke, M.J. Rieke, M.W. Werner, E.L. Wright, L.E. Allen, R.G. Arendt, M.L.N. Ashby, P. Barmby, W.J. Forrest, J.L. Hora, J.-S. Huang, J. Huchra, M.A. Pahre, J.L. Pipher, W.T. Reach, H.A. Smith, J.R. Stauffer, Z. Wang, S.P. Willner, M.J.I. Brown, A. Dey, B.T. Jannuzi and G.P. Tiede, *The Infrared Array Camera* (*IRAC*) *Shallow Survey*, *The Astrophysical Journal Supplement Series* **154** (September 2004), 48–53.

[4] G.B. Berriman, D. Curkendall, J. Good, J. Jacob, D.S. Katz, T. Prince and R. Williams, *Montage: An On-Demand Image Mosaic Service for the NVO*, Proceedings of Astronomical Data Analysis Software and Systems XII, ASP Conference Series, Vol. 295, 2003, H.E. Payne, R.I. Jedrzejewski and R. N. Hook, eds.

[5] E. Bertin, *SWarp User's Guide*.

[6] J.C. Jacob, R.J. Brunner, D. Curkendall, S.G. Djorgovski, J.C. Good, L. Husman, G. Kremenek and A. Mahabal, *yourSky: Rapid Desktop Access to Custom Astronomical Image Mosaics*, Proceedings of SPIE Astronomical Telescopes and Instrumentation: Virtual Observatories Conference, August 2002.

[7] D. Makovoz and I. Khan, *Mosaicking with MOPEX*, Proceedings of Astronomical Data Analysis Software and Systems XIV, ASP Conference Series, (Vol. 347), 2005, P.L. Shopbell, M.C. Britton and R. Ebert, eds.

[8] Message Passing Interface Forum, *MPI: A Message Passing Interface Standard*, The International Journal of Supercomputing Applications and High Performance Computing, Vol. 8, pp. 159–416, May 1994.