Astronomical Data Analysis Software and Systems XIV
ASP Conference Series, Vol. XXX, 2005
P. L. Shopbell, M. C. Britton, and R. Ebert, eds.

Mosaicking with MOPEX

David Makovoz, Iffat Khan

Spitzer Science Center, California Institute of Technology, Pasadena, CA 91125

Abstract. We present MOPEX - a software package for image mosaicking and point source extraction. MOPEX has been developed for the *Spitzer Space Telescope*. This paper concentrates on the mosaicking aspects of the package. MOPEX features the use of several interpolation techniques, coaddition schemes, and robust and flexible outlier detection based on spatial and temporal filtering. A number of original algorithms have been designed and implemented in MOPEX. Among them is direct plane-to-plane coordinate transformation, which allows at least an order of magnitude speed up in performing coordinate transformation by bypassing the sky coordinates. The dual outlier detection makes possible outlier detection in the areas of even minimal redundancy. Image segmentation based on adaptive thresholding is used for object detection, which is part of outlier detection. Efficient use of computer memory allows mosaicking of data sets of very deep coverage of thousands of images per pointing, as well as areas of sky covering many square degrees. Although designed for *Spitzer* data, MOPEX does not require any *Spitzer*-specific fits header keywords to run, and can be applied to other data, that have standard header information on the image geometry and pointing. The package is available for distribution at `http://ssc.spitzer.caltech.edu/postbcd/`.

1. Introduction

MOPEX is an astronomical image processing package developed at the *Spitzer* Science Center. The name MOPEX stands for MOsaicking and Point source EXtraction. The software for the package was originally developed for the automated pipeline to process the *Spitzer* Spatial Telescope data. Up to date MOPEX has been used for processing *Spitzer* data by various groups, such as SSC, IRAC Instrument Team, GLIMPSE, SWIRE, C2D Legacy teams, and others. MOPEX has several major tasks - mosaicking with outlier detection, background matching, point source extraction, pointing refinement - and some other ancillary tasks. The full description of MOPEX capability can be found in the documentation on the *Spitzer* web site `http://ssc.spitzer.caltech.edu/postbcd/`. The pointing refinement is described in Masci, Makovoz & Moshir (2004). This paper is devoted to the mosaicking capabilities of MOPEX.

MOPEX features several interpolation schemes which are described in Section 3. Image reprojection is performed using a new direct plane-to-plane coordinate transformation, the detailed description of which can be found in Makovoz (2004). As part of interpolation MOPEX performs *Spitzer* distortion correction. Robust multiframe and dual outlier detection allows removing cosmic ray hits and other artifacts in the areas of minimal coverage. Efficient memory management using tiles allows performing outlier rejection in the areas of practically unlimited depth of coverage. The users have at their disposal various coaddition options which are described in Section 5. The user specifies the mosaic size, orientation, and pixel size. The only limitations on the mosaic size is the hard drive size.

2. Input Data

The minimal input requirements to make a mosaic with MOPEX is a set of FITS images with the following set of standard WCS keywords: $BITPIX$, $NAXIS$, $NAXIS1$, $NAXIS2$, $CRVAL1$, $CRVAL2$, $CRPIX1$, $CRPIX2$, $CTYPE1$, $CTYPE2$, and the CD-matrix elements, Alternatively, instead of the CD-matrix $CDELT1$, $CDELT2$, $CROTA2$ keywords can be used.

Optionally, each input image can be accompanied with an uncertainty image, where each pixel in the uncertainty image give the measurement and processing uncertainty of the corresponding input image pixel. Also, for each input image the user can specify a number of mask images to mark pixels that are not usable for various reasons, such as dead pixels, saturated pixels, cosmic ray affected pixels, hot pixels, latent pixels, non-linearizable pixels, etc.

3. Image Interpolation

The main interpolation technique used by the Spitzer mosaicker MOPEX is the area overlap interpolation scheme. The interpolated pixel value $I(k)$ is equal to the sum of the values $J(l)$ of the input pixels l overlapping the interpolated pixel k and weighted with the area a_{kl} of the pixel overlap.

$$I(k) = \sum J(l)a_{kl}/\sum a_{kl} \tag{1}$$

This interpolation scheme accurately performs flux transfer between the input and output pixels, but is slow, since it requires computing the overlap areas.

In MOPEX we implemented a new fast interpolation technique. We call it the *grid* interpolation technique. A grid of points is placed on the input frame. Each grid point is assigned the value of the pixel it belongs to. Each grid point is projected onto the output frame and the flux associated with the grid point is added to the output image pixel into which the grid point was projected. This scheme is equivalent to approximating the value of the overlap area with the number of grid points n_{kl} in the overlap area:

$$I(k) = \sum J(l)n_{kl}/\sum n_{kl} \tag{2}$$

The number of the grid points in an input pixel is equal to G^2, where G is the grid ratio. In the limit of $G \Rightarrow \infty$ this scheme is equivalent to the area overlap interpolation. But even for small values of $G \sim 1 - 2$ the resulting mosaics are comparable in quality to those produced by the area overlap interpolation scheme, but produced in a fraction of the time necessary for the area overlap method. The grid method is very useful for quick-look mosaics and for background matching algorithm

MOPEX also provides an option of using the drizzle interpolation described, e.g. in Fruchter & Hook, (2002).

Whatever interpolation scheme is used for the input images, the uncertainty images are interpolated in the identical fashion. A coverage map is created for each interpolated image. For each output pixel the coverage map gives the fraction of the pixel covered by the input image. It reflects the presence of any input bad pixels or pixels on the edge of the input image.

4. Outlier Detection

MOPEX is designed to perform effective detection of cosmic ray hits in the input images. One single frame and two multiframe algorithms for outlier detection are implemented. The single frame outlier detection represents spatial filtering of input images and is used to detect "spiky" objects.

The second outlier detection method represents temporal filtering of the interpolated images. Interpolated images are stacked up. In each stack the mean and standard deviation σ are found. Pixels outside of $n - \sigma$ envelope are declared outliers, where n is defined by the user. This approach breaks down in cases of shallow coverage.

The third approach - dual outlier detection - consists of two-stage filtering. At the first stage, all spatial pixel outliers are detected and saved as detection maps. These detection maps include point sources and cosmic ray hits. Detection maps are interpolated to a common grid. Then for each spatial location, the pixels in the interpolated detection maps are matched. The unmatched pixels are declared outliers. This method effectively supplements the conventional multiframe outlier detection and is the only way of outlier detection in the areas of double coverage. In Figure 1 we show a portion of a mosaic of an IRAC channel 3 mosaic made without (a.) and with (b.) outlier detection. The average coverage of this observation is ~ 4 image per pointing. Both multiframe and dual outlier rejection were used here. Two examples of MOPEX outlier detection in action are given in Patten et al. (2004).

5. Image Coaddition

After the input images are interpolated to a common grid, they can be combined into a single mosaic image. Three weighting schemes available in MOPEX. The interpolated images can be combined using straight averaging, they can be weighted with the interpolated uncertainty images, with the exposure time, or with both the interpolated uncertainty images and with the exposure time. The most general expression for the value O of a coadded pixel is

$$O = (\sum_j (c_j T_j I_j)/\sigma_j^2)/(CU^{-2}) \tag{3}$$

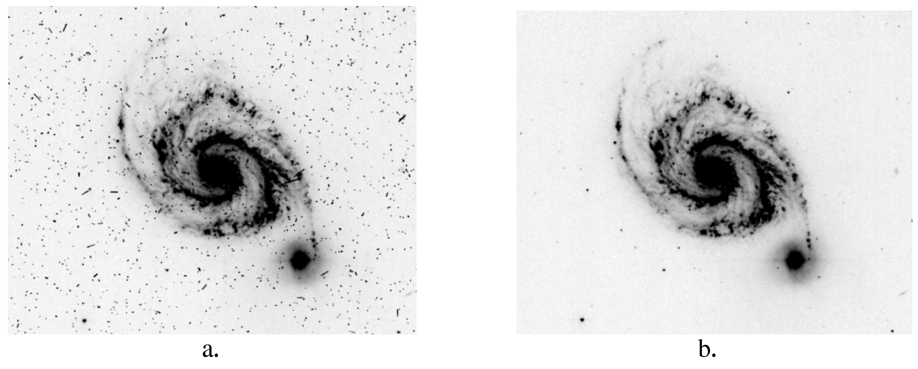

Figure 1. Mosaic without a. and with b outlier detection

where I_j, σ_j, c_j, T_j are the values of the corresponding pixel in the $j-th$ interpolated image, interpolated uncertainty image, coverage map, and the exposure time for the $j-th$ image. C is the value of the coadded coverage map pixel

$$C = \sum_j c_j T_j, \tag{4}$$

and U is the value of the coadded uncertainty image pixel

$$U^2 = \frac{\sum_j c_j T_j / \sum_j c_j}{\sum_j (c_j T_j)/\sigma_j^2}. \tag{5}$$

The mosaic uncertainty image and coverage map are saved as separate products.

MOPEX also produces the median mosaic and the corresponding uncertainty images. It is created by using the median value in the stack of interpolated pixels for the mosaic pixel value and the 68 percentile width around the median for its uncertainty. It provides a quick way of creating clean (no cosmic ray hits) mosaic images without having to run outlier rejection. However, the flux is not conserved in the median mosaics.

6. Mosaic Mask

A new kind of mosaicking implemented in MOPEX is mosaicking of mask images. The mask image interpolation have to preserve the bit information in the input mask images. Regular interpolation destroy the bit information. There is no single "correct" way to set the value of the interpolated mask pixel. We implemented the following. The value of each output interpolated pixel is set to the result of the logical OR of the values of the input mask pixels overlapping the interpolated pixel.

The coaddition have also been modified to preserve the bit information. MOPEX makes two products. First product is a mosaic fits data cube. Each plane in such a data cube corresponds to a single bit in the input masks. The pixel value in the plane equal to the number of interpolated masks that have that bit set in that pixel. The second product is single plane mosaic image. It has a bit to bit correspondence to the input masks. A bit in the pixel value of the single plane mosaic image is set if the number of the interpolated masks with that bit set in that pixel exceeds the user specified threshold.

7. Using MOPEX

MOPEX is implemented as a set of perl scripts each running a separate task. Each script runs a number of individual modules written in C/C++. The complete package consists of the perl scripts, binary executable files, sample data with configuration files, and documentation. The binaries are available for Solaris and Linux operating systems. MOPEX can be downloaded from `http://ssc.spitzer.caltech.edu/postbcd/download-mopex.html`. The perl scripts are run on the command line. Configuration files (aka namelists) are also given on the command line. They are used to turn on and of the individual modules run by the script, give the location of the input data, set up the output directories and also include separate namelist blocks for the individual modules.

Benchmarking of MOPEX is not trivial. The only step that clearly scales with the number of image and image sizes is interpolation. Interpolation of a 256x256 image without resampling on a 1 GHz Sparc 4GB RAM SunBlade takes 1.6 seconds using area overlap interpolation and 0.2 seconds using grid interpolation with the grid ratio $G = 1$. For the other steps the timing varies widely with the density of the cosmic ray hits and the coverage depth.

8. MOPEX Future

As a part of JPL R&TD High Capability Computing in Engineering and Science Strategic Initiative a project has been funded to develop a parallelized implementation of MOPEX to run on JPLs 1,024 node Xeon computer. By conforming to standard Message Passing Interface (MPI), the parallelized MOPEX software can be exported for use on other computing systems and will be of great value for other *Spitzer* surveys.

Some other current developments include mosaicking of moving objects, implementing the bicubic interpolation scheme, porting MOPEX to MAC OS 10, a graphic user interface (GUI). We also plan to extent the usage of MOPEX for other than *Spitzer* images.

Acknowledgments. This work was carried out at the *Spitzer* Science Center, with funding from NASA under contract 1407 to the California Institute of Technology and the Jet Propulsion Laboratory.

References

Masci, F. J., Makovoz, D, Moshir, M. 2004, PASP, 116, 842

Makovoz, D. 2004, PASP, 116, 971

Fruchter & Hook, 2002, PASP, 114, 144

Patten, B.M., Hora, J.L., Fazio, G.G., Barmby, P., Wang, Z., Makovoz, D., in Proc SPIE 5487, eds. Marija S. Scholl and Bjorn F. Andresen, 223