ADAM DOCUMENTATION V0.60

MATTI VIIKINKOSKI

1. Introduction

This is a collection of routines for $3D$ asteroid shape modelling from observations. **This package is not meant to used as a black box, usage requires some understanding how and why these methods work and their limitations. The software is provided 'as is' without warranty of any kind.** The program is still very much a work in progress and more will features will be added in the future. The main features are:

- Gradient based optimization
- Parametric shape supports for general, non-starliked and nonconvex surfaces
- Boundary contour extraction is not required for disk-resolved images
- PSF may be easily incorporated
- Robust convergence
- Parallel computing

The novel feature is data fitting on the Fourier plane. This allows circumventing tedious subpixel fitting on the spatial domain and offers natural interpretation for the pixel size as maximum frequency content present in the data. Observational methods currently supported are

- Disk-integrated photometry (lightcurves)
- HST/FGS data
- Disk resolved optical images, e.g. adaptive optics
- Disk resolved interferometry
- Disk-resolved thermal observations
- Range-Doppler radar images

The theory behind these reconstruction methods is explained in Viikinkoski et al. (2015); Viikinkoski & Kaasalainen (2014); Kaasalainen & Viikinkoski (2012).

2. Shape supports

Shapes are represented as a collection of triangular facets formed from vertices, and the vertex coordinates depend on parameters. Currently supported parametric shape representations are octantoids (Kaasalainen & Viikinkoski 2012) and subdivision surfaces.

An octantoid is a surface given by $p \in \mathbf{R}^3$ that can be parametrized in the form

$$p(\theta,\varphi) = \begin{cases} x(\theta,\varphi) = & e^{a(\theta,\varphi)} \sin\theta\cos\varphi, \\ y(\theta,\varphi) = & e^{a(\theta,\varphi)+b(\theta,\varphi)} \sin\theta\sin\varphi, \\ z(\theta,\varphi) = & e^{a(\theta,\varphi)+c(\theta,\varphi)} \cos\theta, \end{cases}$$

where a, b and c are conveniently expressed as linear combinations of the (real) spherical harmonic functions $Y_l^m(\theta,\varphi)$, with coefficients a_{lm}, b_{lm} and c_{lm}, respectively. Note that (θ,φ), $0 \le \theta \le \pi$, $0 \le \varphi < 2\pi$, are coordinates on the unit sphere S^2 parametrizing the surface.

A subdivision surface is constructed by first choosing an initial control set (vertices), which are the parameters determining shape, and the final surface is formed by applying a subdivision method to the initial shape.

3. Regularization methods

As the shape reconstruction from observations is an inverse problem, regularization methods are needed to keep the shape coherent and to compensate for conflicting data. Since the shape is represented by a triangular mesh, regularization should keep the mesh both coherent and smooth. Following regularization methods are implemented:

- `Octantoid_Reg` is for octantoids only. This regularization penalizes divergence from a starlike shape.
- `Convex_Reg` tries to keep shapes locally convex.
- `area_reg` measures facet areas and divergence from the mean facet area is penalized.
- `dihedral_angle_reg` prefers planar regions by discouraging large angles between the normal vectors of adjacent facets.

Usually, octantoid and convex regularization are used for octantoids, and convex, area and dihedral regularization for the subdivision surfaces.

4. Coordinate frames and rotations

ADAM uses three different coordinates systems: the asteroid-centric coordinate frame (object frame) with coordinate axes fixed to the asteroid, the asteroid-centric inertial frame (world frame), and the camera frame, which is determined by the instrument orientation geometry. The plane-of-sky view of an asteroid is obtained by projecting the asteroid in the camera frame to the xy-plane.

Orientation of the asteroid in the world frame is determined by three angles (and time t): ecliptic latitude $\beta \in [0, \pi]$ (**NB**: DAMIT uses $90^\circ - \beta$)), ecliptic longitude $\lambda \in [0, 2\pi]$ and the rotation rate ω (rad/day). Transfer matrix is

$$R_z(\lambda)R_y(\beta)R_z(\omega \cdot t),$$

where R_i the standard three dimensional rotation matrix with respect to the axis i. Matlab routine for calculating the (transposed) transfer matrix is `Rot_Matrix`. It also determines partial derivatives wrt. angles. Conversion from the world frame to the camera frame is determined by two vectors: camera direction $\mathbf{E}$ as seen from the world frame (camera look direction is $-\mathbf{E}$), and camera up direction $\mathbf{v}$ which determines how camera is rotated wrt. world. This is useful is in the case observations are in the equatorial coordinates. Camera frame conversion matrix R_C maps the observable quantities to the first two coordinates, third coordinate axis is chosen to complete the coordinate system. Thus the total conversion matrix from the object to two dimensional plane is

$$PR_CR_z(\lambda)R_y(\beta)R_z(\omega \cdot t),$$

where P is the projection matrix discarding the third coordinate.

The camera matrix for adaptive optics and other similar observations is determined by the function `Cam_Matrix`. The case of Range-Doppler radar is slightly more complicated, since the camera mapping depends on rotation rate and radar frequency, among other things. The mapping transfers the triangular mesh to a new orthogonal coordinate system, where first two coordinates are respectively line-of-sight velocity and relative range.

5. Support for parallel computing

Due to shift from sparse (boundary contour) to dense (Fourier plane) data, more computational resources are required. Most of the computation time is spent on calculating partial derivatives with respect to vertex coordinates. Fortunately this task is easily parallelizable. Matlab's parallel computing toolbox facilitates effortless transfer from serial to parallel computing. The lightcurve routine uses Matlab parallel toolbox. However, parallel computing toolbox does not support shared memory, making it slow at large datasets. An alternative for parallel computing toolbox is OpenMP API, which is used in adaptive optics and range-Doppler radar routines.

6. Getting started

Here we will briefly outline the structure of the inversion routines. The main point to remember is that the data fitting is done on the Fourier plane, with an exception of the lightcurves. For more detailed information, one should peruse comments in the examples.

(1) **Preparation of data**. Each data image is transformed to the Fourier plane by taking the two-dimensional Fourier transform (usually FFT is used). However, one should keep in mind that the FFT only approximates the Fourier transform, especially in the case where the pixel size non-negligible. Then the frequencies used for data fitting are selected. The trivial choice is take all the available points (with zero frequency and negative frequencies discarded), but often smaller number of frequency points is sufficient. This is especially true with the range-Doppler data, where the instrument resolution is not necessarily the same as the usable resolution. Also the ephemeris information at the observation time is recorded.
(2) **Initialization of shape**. The spin vector and size of the shape will be set to some initial values. Note that the especially rotation period should be chosen carefully, since the optimization routine is not meant for the period determination. This is usually not a problem, since the rotation period can be determined from the lightcurves beforehand.
(3) **Initialization of weight coefficients**. This is singularly the most important step. The regularization weights should be chosen to keep the shape consistent during the optimization. On the other hand, too large weights could result non-optimal solution. The weighting between different data modes is also important. Theoretical guideline called *maximum compatibility estimate* (Kaasalainen 2011) can be used to determine the optimal weighting.
(4) **Shape optimization** For each observation, the model fit to data is calculated and Jacobian matrix is formed. All the examples here use the Levenberg-Marquardt optimization, which is computationally demanding when large datasets are used.

6.1. Available routines.

The main purpose of this software package is the calculation of Fourier-transformed generalized projection of the shape model at selected frequencies and the formation of Jacobian matrix, which can be used in optimization routines. Most of the routines are implemented in C (with an exception of thermal modelling), and Matlab is used only for combining results and for the optimization routine. Each routine calculates partial derivatives with respect to the vertex coordinates and rotation angles, making the Jacobian matrix independent of parametrization.

- `Generate_LC_Matrix` calculates model fit to the lightcurves. The actual computation is done in C, since the raytracing required for nonconvex shapes in computationally demanding. Lightcurve data can be read from a file (in DAMIT

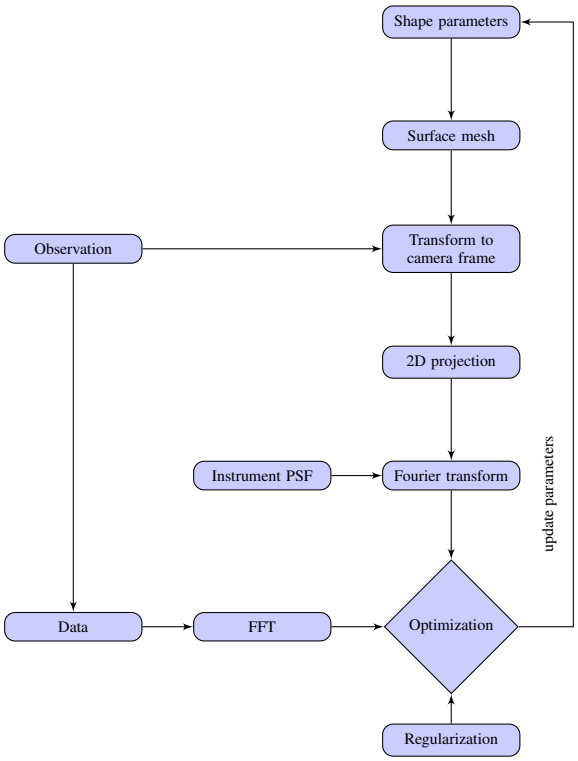

FIGURE 1. ADAM optimization algorithm as a schematic for one image type.

format) using routine `read_lightcurve_struct`. Only relative lightcurves are used, absolute lightcurves may be converted to the relative lightcurves. It is important to remember that the absolute size of the asteroid cannot be determined if only relative lightcurves is used. See examples `Metis_lc_Example_oct` and `Metis_lc_Example_subdiv`.

- `Generate_AOFT_Matrix_mex` calculates model fit the adaptive optics images (can also be used for other kinds of images and HST/FGS data). It is also possible to include a PSF. See comments in the example files `AO_Invert_Octantoid_Example` and `AO_Invert_Subdiv_Example`. More complicated example is included in `Extras` folder. That example is used to invert the asteroid Metis from lightcurves and AO data using a nontrivial PSF (rough estimate, since true PSF is not known). Unfortunately the processed AO data cannot be included, as it is not publicly available. Of course, data from the Keck archive may be used.
- `Calc_Temp` calculates temperature distribution of triangular mesh using the FFT approximation. Also partial derivatives with respect to vertices and rotation angles are returned. This routine is written in Matlab, but will be converted to C.
- `Generate_HF_Matrix` determines heat flux (normalized wrt total flux) from an object and calculates partial derivatives. `Thermal_interf_Example` shows how this routine can be used to invert shape from thermal infrared interferometry data (simulated). Data for the example was generated by first solving heat equation using the finite difference method, and then the CASA software was used for adding

atmospheric noise and for nonuniform sampling corresponding to the antenna array.
- `Generate_RD_Matrix` calculates model fit to range-Doppler data, and partial derivatives with respect to vertex coordinates, rotation angles, offsets and scaling terms.
- `Octantoid_to_Trimesh` converts a shape represented by octantoid parametrization to a standard triangular mesh. Also partial derivatives of vertex coordinates wrt. octantoid parameters are returned.
- `Sqrt3_Subdiv` subdivides given triangular mesh. Subdivision level can be chosen. Increasing subdivision level adds stability to inversion, but the number of facets increases exponentially wrt subdivision level. Partial derivatives wrt. vertex coordinates are returned.

6.2. **Data structures.** Data is passed to subroutines by using a struct with cell array fields. Following fields are required (let's call the struct variable `FT`):

- `FT.E`: Unit vector, camera direction in the global frame as seen from the asteroid.
- `FT.E0`: Unit vector, Sun direction as seen from the asteroid (not used for range-Doppler).
- `FT.TIME`: Observation time.
- `FT.freq`: $n \times 2$ matrix, containing the sample frequency points on the Fourier plane. Note that this vector should not contain $(0, 0)$.
- `FT.data`: $n \times 1$ complex vector, Fourier transform of data corresponding to frequencies listed in `FT.freq`.
- `FT.distance`: Distance (in AU) between the asteroid and the camera (not needed for range-Doppler)
- `FT.Hdistance`: Distance between the asteroid and the Sun (used only for thermal modelling).
- `FT.WL`: Wavelength (only for thermal modelling)
- `FT.pdf`: Fourier transform of the point-spread function at the points determined by `FT.freq` (currently only for AO and HST/FGS)
- `FT.up`: Unit vector, determines camera orientation (not needed for range-Doppler)
- `FT.radarfreq`: Radar frequency (only for range-Doppler);

Lightcurves use a different structure. See `read_lcurve_struct`.

6.3. **Offset and scale.** For each data image, it is assumed that the position with respect to model is unknown. Offset in the spatial plane corresponds to phase change on the Fourier plane. Derivatives wrt. offset term are calculated and optimal offset are determined. It is usually a good idea to approximately center images beforehand, to facilitate better converge. This is especially true for range-Doppler images.

There are two approaches to image scaling. If images are good with low noise levels, we divide images with total brightness. In cases where noise or artifacts dominate (e.g. range-Doppler or thermal images) it is better to leave images unnormalized and fit a scaling term for each image.

Acknowledgements

Thermal modelling routine uses KissFFT[1] for the FFT calculations.

[1] `http://sourceforge.net/projects/kissfft`

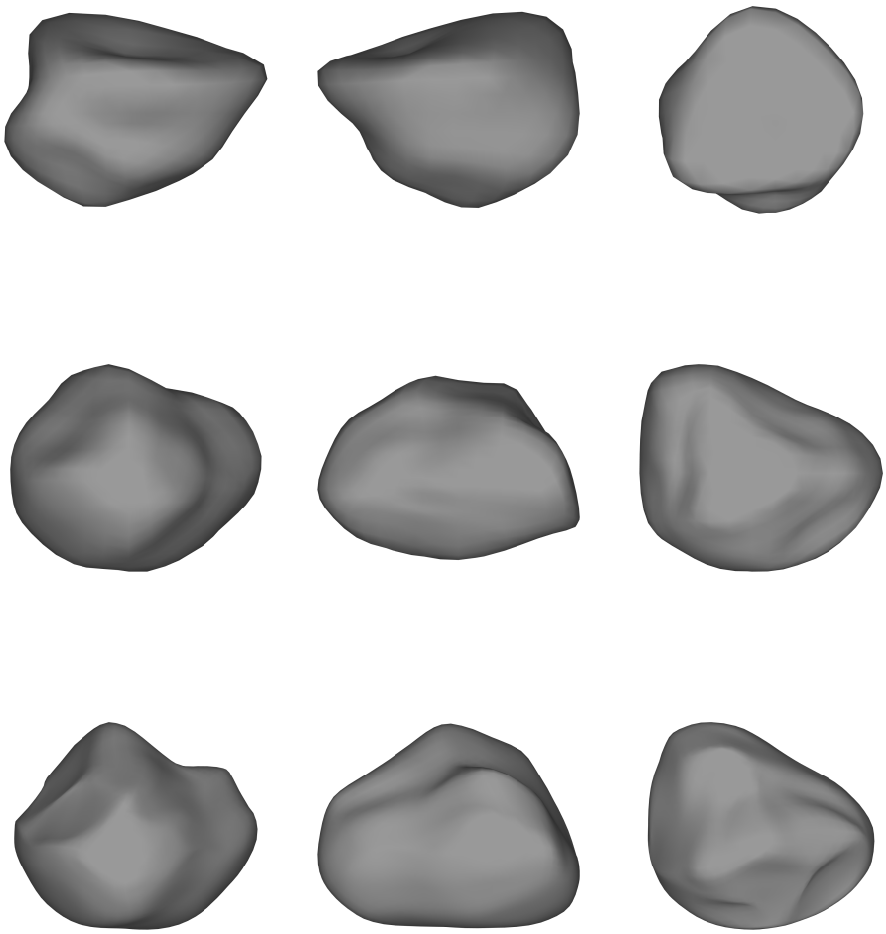

FIGURE 2. Example reconstructions from lightcurve and adaptive optics data.

REFERENCES

Kaasalainen, M. 2011, Inverse Problems and Imaging, 5, 37
Kaasalainen, M. & Viikinkoski, M. 2012, Astronomy & Astrophysics, 543
Viikinkoski, M. & Kaasalainen, M. 2014, Inverse Problems and Imaging, 8, 885
Viikinkoski, M., Kaasalainen, M., & Ďurech, J. 2015, Astronomy & Astrophysics

E-mail address: matti.viikinkoski@gmail.com