A

MCVCM: A Python Program for Multi-Catalogue Visual Cross-Matching

A.1 Foreword

This software was initially designed and coded by me, *the author*, to facilitate the cross-matching of the ATLAS 1.4 GHz radio source catalogue with DES photometric source catalogue. Initially, this cross-matching was performed directly between the radio data and optical $I-$band (e.g., Figure C.1) and was later adapted to work with SWIRE 3.6 μm infrared (achieving a more reliable cross-match, as established in § 1). During the necessary modifications to facilitate this, I endeavoured to make this tool generalised and adaptable enough to be usable with any catalogue pairings. Following this, I have made the code available in a public repository (`github.com/FriskyGrub/MCVCM`). MCVCM is being used in other projects – GLASS (GAMA Legacy ATCA Southern Survey) and GLEAM (GaLactic and Extragalactic All-sky MWA Survey) – which has encouraged the on-going development of the software.

A.2 Description

MCVCM (Multi-Catalogue Visual Cross-Matching) is written in `Python 3.6` and is designed to facilitate the visual-manual cross-matching of two catalogues provided by the user.

A.3 Cutout Assembly

When MCVCM is first run, the cutout generator creates a memory mapping of the mosaicked radio continuum, radio RMS and infrared images provided by the user. As the user iterates through catalogue sources in MCVCM sources, a slice of these images centred about the target source is retrieved for the generation of each cutout. Using this memory-mapping technique, the requisite data can be retrieved fast enough that the images need not be stored entirely in computer random access memory. Using a combination of `astropy world coordinate system` utilities and the `astropy` affiliated `reproject` tool, the radio data is re-sampled (via interpolation) to match the resolution and orientation of the infrared image. These data are then overlaid and displayed to the user using the `matplotlib imshow` and `contour` plotting utilities. User interactivity is established with a combination of `matplotlib canvas` listeners and `Python`'s standard graphical user interface package `TkInter`.

A.3.1 Radio Contour Levels

MCVCM accepts a continuum map, as well as an RMS map for each of the fields being cross-matched. Using the sliced region of these data (about the source to be cross-matched) radio contours are calculated as $2^n \times \langle \mathrm{RMS} \rangle$ for this slice and overlaid onto the infrared image, this method allows for consistent contours throughout the radio field. I have not established a reliable means for drawing consistent contours when a complimentary RMS map is unavailable, but I will incorporate this in future iterations of MCVCM.

A.4 User Interaction

The script is designed to be called from within the folder containing `MCVCM.py` in the following manner:

```
>> MCVCM.py {CDFS,ELAIS,Field3} [-h] [-v] [-t] [-x] [-d] [--savefigs {png,eps,pdf}]
```

MCVCM requires the user to specify the field that they will be cross-matching as a positional argument (in this example either 'CDFS', 'ELAIS', or 'Field3'). These field options are automatically generated from the `./path_config.json` file once edited by the user.

```
positional arguments:
  {CDFS,ELAIS,Field3}   Specify if we are working on CDFS, ELAIS, or Field3

optional arguments:
  -h, --help            show this help message and exit
  -v                    toggles verbose output
  -t                    toggles output of function timings
                        (requires verbose mode)
  -x                    if specified, MCVCM processes only sources
                          previously skipped by the user
  --savefigs {png,eps,pdf}
                        if provided with an extension, saves final
                        cross-matched source cutout to that format
                        inside ./MCVCM-figs/ (e.g. --savefigs png)
```

A.4.1 Configuration

Three configurations are needed once MCVCM has been installed to begin cross-matching of data. The first step is editing the `./path_config.json` file within the main directory to specify the paths to the required data files. The specified data files can exist anywhere on the user's machine (so long as the full file-path is provided) and are opened as read-only. The following is an example of the configuration file used for the cross-matching of the ATLAS radio sources with SWIRE infrared, CDFS and ELAIS are the two ATLAS fields, and Field3 is an example. This configuration file can contain unlimited field-data structures.

```
<./path_config.json>
{
  "ELAIS": {
    "radio_continuum": "data/ELAIS/ELAISmosaic_allch_8March2015.fits",
    "radio_rms": "data/ELAIS/ELAISmosaic_allch_noise_8March2015.fits",
    "image_mosaic": "data/ELAIS/elais_mosaic.fits",
    "radio_catalog": "data/ELAIS/ATLASDR3_ELAIScmpcat_23July2015.dat",
    "infrared_catalog": "data/ELAIS/SWIRE_ELAISS1-FALL05.fits"
  },
  "CDFS": {
    "radio_continuum": "data/CDFS/CDFSmosaic_allch_8March2015.fits",
    "radio_rms": "data/CDFS/CDFSmosaic_allch_noise_8March2015.fits",
    "image_mosaic": "data/CDFS/cdfs_mosaic.fits",
    "radio_catalog": "data/CDFS/ATLASDR3_CDFScmpcat_23July2015.dat",
    "infrared_catalog": "data/CDFS/SWIRE-CDFS-FALL05.fits"
  }
  "Field3": {
    "radio_continuum": "path/to/continuum_map.fits",
    "radio_rms": "path/to/rms_map.fits",
    "image_mosaic": "path/to/image_mosaic.fits",
    "radio_catalog": "path/to/primary(radio)_source_catalogue.fits",
    "infrared_catalog": "path/to/secondary(infrared)_catalogue.fits"
  }
}
```

The second configuration provides information to MCVCM that allows it to read the correct columns from the source catalogues for source positions and identifications, the size (in image pixels) to be rendered from the image files, and two user-configuration options: the start-index (counting from 0, which row to start cross-matching from) and the figure position (this will be the position of the MCVCM cross-matching window on the users screen - (0,0) is top-left). The third configuration is adjusting the scaling of the infrared image for standardised source visibility.

```
<./parameter_config.json>
{
  "column_names": {
    "radio_ra": "RA_deg",
    "radio_dec": "Dec_deg",
    "radio_id": "ID",
    "infrared_ra": "ra",
    "infrared_dec": "dec",
    "infrared_id": "object"
  },
  "cutout_pixels": {
    "radio": 95,
    "infrared": 170
  },
  "start_index": 0,
  "figure_position": {
    "horizontal": 0,
    "vertical": 0
  }
  "image_scaling": {
    "max_saturation": 1.5,
    "power_normalise": 0.7
  }
}
```

A.4.2 Calibrating Image

Calibration of the infrared image is done via a utility packaged with MCVCM. Once `./path_config.json` and `./parameter_config.json` are established by the user, the calibration script produces an MCVCM cutout with two variable sliders that adjust the normalisation factor, and maximum saturation threshold for the image scaling. When satisfied the user can enter the displayed numbers into the `"max_saturation"`, and `"power_normalise"` fields of `./parameter_config.json`.

A.4.3 Cross-Matching

Once the MCVCM instance is created, the catalogue cross-matching is split into three phases:

1. **Infrared Host Selection**: The user is presented with a cutout image of the infrared postage stamp with radio contours overlaid (e.g., Figure A.1*a*). Infrared catalogue source positions are marked as small crosses that the user can select by clicking with the mouse. In

this phase the user may only make one selection, this will be marked with a large hollow black cross-hair that is retained through the remaining phases (Figure A.1*d*). If a mistake is made, the user can reset by pressing '`r`'.

2. **Radio Core Selection**: The image remains the same, with the cross-hair marking the location of the selected infrared host still visible, and the infrared catalogue source positions are replaced with the radio component peak positions (Figure A.1*d*). The user can now select one of these sources as the radio core if there is no viable core this phase can be skipped.

3. **Radio Component Selection**: The image will remain the same as Phase 2 with a slight change to the radio component marker style to notify the user of the new phase (Figure A.1*e*). The user can now select any number of radio components that are associated with this cross-identification (Figure A.1*f*).

At any time during the cross-matching, the user may press one of the following keys or key combinations for the associated effect.

```
- {h}  Print this help list
- {r}  Reset current progress of source cross-matching and
       re-draw starting at Phase One
- {b}  Increase the displayed area of the cutout by 33%
       (this will reset current source progress)
- {t}  toggle on and off the marked catalogue source positions
- {s}  Save an automatically named cutout of the current figure
       state to ./MCVCM-figs/
- {i}  Print information on current source from the radio
       source catalogue
- {j}  Print the last 25 rows of the cross-match catalogue being
       assembled in ./MCVCM-tables/
- {c}  Open a dialogue box for the user to enter a comment for
       the current source. This comment is saved in the final
       table under the column 'MCVCM_comment'
- {1, 2, 3, 4}  provide an integer confidence flag for the
                cross-matching of this source
- {shift + x}  Skip cross-matching of the current source, skipped
               sources can be cross-matched at any time by starting
               the script with the '-x' flag
- {shift + q}  Safely close the MCVCM instance
```

Each phase is completed when the user presses '`spacebar`', and the cross-matching of a particular source is finished when the user presses '`d`' or '`enter`', provided that at least one radio component has been selected. If no infrared core was selected in Phase one, then the cross-identification is given a '`no-ir-source`' in place of the infrared source ID. If no core is selected, then the component '`C0`' is used as the core for catalogue management. After cross-matching a unique association string is generated for each of the selected radio sources according to the following:

```
<infrared ID>*<radio core ID>*m<number of components>*C<component number>
```

For example:

```
Infrared host: SWIRE3_J003940.76-432549*EI1896*m3
Radio host:    SWIRE3_J003940.76-432549*EI1896*m3*C0
Component #1:  SWIRE3_J003940.76-432549*EI1896*m3*C1
Component #2:  SWIRE3_J003940.76-432549*EI1896*m3*C2
```

Note: Once a source is cross-matched it will be skipped by the MCVCM source incrementation, but it is not write-protected. During cross-matching, the MCVCM cutout will centre on the lowest-indexed radio component without a cross-match (skipping sources that were force-skipped by the user via '`shift + x`'). If during the cross-matching of this new source, the user deliberately or accidentally cross-matches a source other than the one that the catalogue is centred on, that cross-match is correctly stored in the cross-match catalogue (even if it was previously cross-matched). Once this cross-matching is complete MCVCM will again load the lowest indexed radio-source without a cross-match. This does allow the user to correct any radio-component mis-associations that they might have made earlier but it also allows for the possibility of erroneously altering an earlier cross-match.

A.4.4 Saving A Figure

The saving of a figure can either be done manually or automatically. If MCVCM is started with the `--savefigs {png,eps,pdf}` flag then the final cross-matched image will be saved upon completion. This can take up to one second per source and is not recommended. The final version of MCVCM will include an option to iteratively generate these image cutouts once the entire catalogue has been cross-matched. Individual figures can be saved manually by the user either via the `matplotlib` save icon in the MCVCM window (the user will need to specify file name and location) or by pressing the '`f`' key at any point during the cross-matching of the source.

A.5 Version Control

The master catalogue which MCVCM is appending to as cross-matches are made is located at `./output/tables/<field>_mcvcm_table.dat`, there will be separate files for each `<field>`

being cross-matched. Each time MCVCM is resumed from the terminal, a copy of this file is made with an incremented numerical suffix. These act as back-ups of the cross-matching at the state when the MCVCM instance was launched. At any point, the user can revert to one of these states by following the 'Rolling Back' procedure.

A.5.1 Rolling Back

Rolling-back to a previous state of the cross-matched catalogue currently requires manual file management by the user, in the future this will be handled by MCVCM. To roll-back to a previous state locate the desired version (e.g., `./output/tables/<field>_mcvcm_table-015.dat`) and replace the master file (`./output/tables/<field>_mcvcm_table.dat`) with this version. It is a good idea to make a backup of both files in case of a mistake.

A.6 Catalogue Management

Once the cross-matching has been performed, a separate script needs to be run if the user wishes to create an amalgamated catalogue. Due to the vast differences in catalogue formats and conventions (for example, the filling of missing values), MCVCM does not include a generalised means of performing this merging. Upon request, I am happy to provide assistance in constructing this tool or provide the example used for the handling of the ATLAS data for reference.

A.7 Continuing Development

Future work will see MCVCM handling as many complimentary data sets as the user desires, including the ability to over-lay multi-frequency radio contours and switch between (for example) infrared, optical and x-ray images on the fly. Supporting software will be developed to allow more robust version control and automated restoration, and final cross-matched catalogue assembly. MCVCM will continue to be developed along-side the users so that the user interface, user experience, and scientific utility are improved.

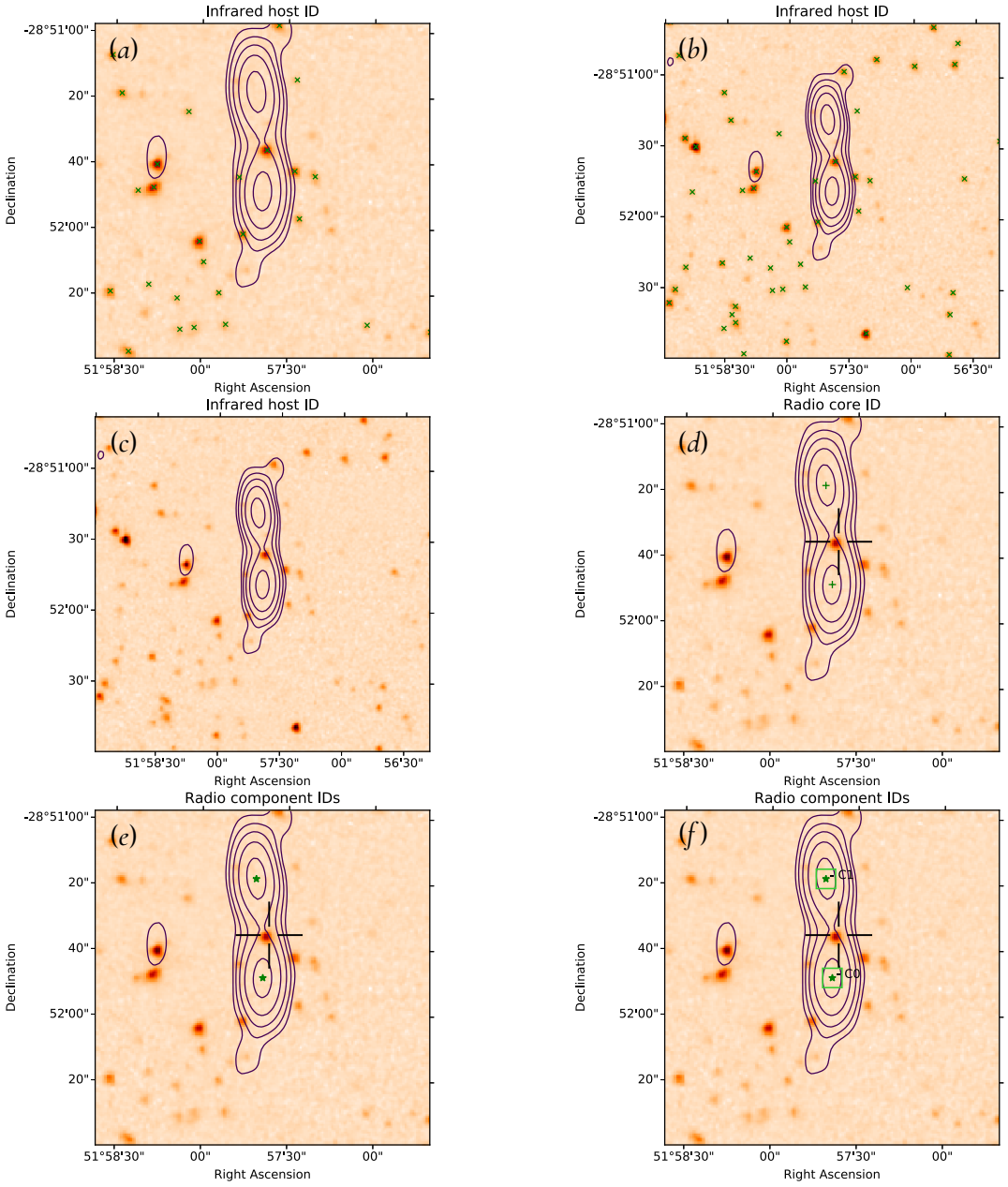

Figure A.1: Examples of various states during the three phases of source cross-identification in MCVCM. In all images, radio contours are overlaid onto an infrared image. (*a*) - Phase One: no action taken by user infrared catalogue sources are marked with green crosses. (*b*) - Phase One: user has incremented the field of view, infrared sources are marked (*c*) - Phase One: user has incremented the field of view and toggled off infrared sources (*d*) - Phase Two: user makes an infrared host selection in Phase One (black cross-hair) and has moved to Phase Two, radio source positions are now marked for core selection. (*e*) - Phase Three: identical to (*d*); no radio core was selected, and radio position markers have changed to indicate the new phase. (*f*) - Phase Three: the user has selected two radio components and is ready to finish this source cross-identification.