DetectorChecker Instruction Manual

For release version 1.0.0

Julia A Brettschneider, Tomas Lazauskas, Oscar Giles, Wilfrid S Kendall

1 Introduction

Quality assessment for digital X-ray detector panels is an important issue, because it impacts the quality of the images. Detector panels being costly, their replacement or refurbishing requires careful justification. Whether or not individual damaged pixels are actually problematic depends on where they are located and if they occur in close to each other. Spatial analysis of dysfunctional pixels can be the key to assessing the relevance of damage. An overview of common types of spatial patterns of dysfunctional pixels can be found in [2].

The freely accessible webtool *DetectorChecker* allows users to assess digital X-ray detector quality via spatial analysis of dysfunctional pixels. The webtool is built using the *R* webtool *shiny*, based on an open source *R*-package of the same name. This webtool will be connected to an open archive of detector data contributed by the user community: it is envisaged that users will be able to choose to create an account to upload their data to this archive, to facilitate more extensive analysis of data trends concerning detector damage.

The quality assessment provided by *DetectorChecker* can be used to support decision making about refurbishment and purchase of digital X-ray detectors. It can also be used to monitor the state of the detector over time and to link this to particular types of usage. It is hoped that *DetectorChecker* will help highlight the occurrence of particular damage patterns, thus guiding users in assessing likely reasons for such damage.

At the beginning of a session, *DetectorChecker* allows users either (a) to select one of the common detector layouts, or (b) to enter parameters defining a custom layout. After this users can upload data in the form of `xml` or `tiff` files, allowing *DetectorChecker* to identify dysfunctional pixels. *DetectorChecker* will then enable the user to create numerical and graphical summaries about the spatial distribution of the dysfunctional pixels. This can be done either on the pixel level or on the *event level*; which automatically identifies and represents clusters of damage. The event level is likely to be more closely linked to the reasons for damage, hence may provide a clearer picture of what is going on. Finally, users can request that severely damaged regions of the detector are masked so as to exclude them from the analysis. This enables the effect of one specific damaged region to be decoupled from an assessment of the remainder of the detector.

DetectorChecker has been developed by the authors as follows. JAB wrote the original

R-code and contributed to the design of the webtool, TL converted it into an *R*-package and designed and built the webtool, while WSK initiated, supervised and tested the project at all stages. We thank the Alan Turing institute for providing funding for this project. The authors are grateful to Jay Warnett (at Warwick Manufacturing Group), Andrew Ramsay (at Nikon Metrology, UK, at the time) and Nicola Tartoni and Ian Horswell (at Diamond Lightsource, UK) for guidance on X-ray machines, detector types and damages. They are also grateful to Martin Turner and Tristan Lowe (at University Manchester) for the interest in this project and their feedback. The webtool builds on previous work supported by EPSRC grant EP/K031066/1, and was funded by a Turing seed fund project awarded by the Alan Turing Institute under EPSRC grant EP/N510129/1.

2 Workflow

The webtool *DetectorChecker* can be assed via the following url:

```
https://detectorchecker.azurewebsites.net
```

It is structured around tabs that allow the user to choose actions on the left hand size and outputs numerical and graphical results on the right hand side. Tooltips are available and appear in pop-up boxes. Outputs can be saved using standard mouse click functions from the operating system. The functionalities of the webtool are also visualised in Figure 1.

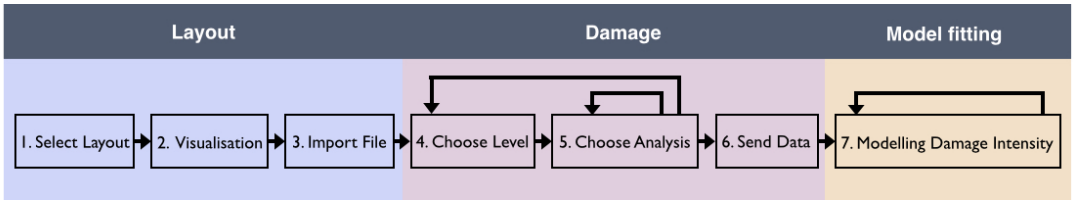

Figure 1: **Flowchart illustrating possible workflows using *DetectorChecker*.**

The basic program consists of eight steps some of which can be run multiple times with different options. Steps 1 to 2 deal with the detector panel layout. In Step 3 users upload a file containing the coordinates of dysfunctional pixels. Steps 4 to 5 are the core functions of the webtool. They conduct the analysis of spatial patterns of dysfunctional pixels. This is followed by an opportunity to create an account to share the data with the user community in Step 6. Step 7 allows to choose spatial functions related to the layout as covariates and fit a linear model. This can help identifying factors that have an influence on the decay of pixels. The steps are explained in more detail in the itemised list below.

1. **Select Layout:** Users can choose from a list of common detector layouts using a drop down menu. Alternatively, users can define their own layout by entering parameters (see Section 3).

2. **Choose Visualisation:** The default is a graphical representation of the layout. Alternatively, pixelwise functions related to the layout can be visualised. This includes measuring the distances to the centre, the corners or the edges of the panel or the edges of the subpanels. These spatial functions are being used as covariates in later model fitting (see Section 7). Both the layout itself and the spatial functions can be plotted.

3. **Import File:** Upload a file with information on dysfunctional pixel locations. This can be an `xml` file containing the coordinates of the damaged pixels or `tiff` file containing an array of pixel intensities corresponding to the mask describing the dysfunctional pixels. For example, this can be a file created by manufacturers maintenance routines (e.g. Perkin Elmer's *Bad Pixel Map*) in `xml` or `tiff` format.

   An image with solid squares indicating dysfunctional pixels is plotted automatically

4. **Choose Level:** Damage can be assessed on different levels (see Section 6).

   - *Pixels:* Each damaged pixel is seen as equally relevant occurrence.
   - *Events:* Adjacent pixels are summarised

   If *Events* is chosen, the user can also specify which types are included.

5. **Choose Analysis:** Spatial statistics tools to explore the spatial patterns of damage are available to the user. One is selected at a time.

   - Spatial density plots based on the damaged pixel occurrance
   - Counts of damaged pixels associated with subpanels of the detector
   - Arrow between nearest neighbours
   - Rose plot of the corresponding angles
   - K-, F- and G-function to explore spatial randomness on different scales
   - Dito but considering spatially inhomogeneous intensity

   Plots are displayed. The user can repeat this step running through different choices.

   Mouse clicking on individual subpanels will give individual plots showing only the subpanel in question.

6. **Send Data:** Upload data to the archive using your email address as unique identifier.

7. **Modelling Damage Intensity:** Select covariates related to detector layout to be included in a linear model explaining pixel damage and fit the model.

3 Detector Layout

There are preset layouts to choose from for common detector panels. These include Perking Elmer (full and cropped version), Pilatus and Excalibur.

Alternatively, users may define their own layout. A detector panel is typically composed of subpanels arranged in a rectangular grid. They may also be referred to as *read out groups (ROG).* They can be seamless such as in the Perkin Elmer design (e.g. [3]), but gaps between the rows and between the columns are allowed as long as they do not interfere with this being a grid. In other words, the gap between the first and second columns need to be the same in all rows. Explain custom made layout types and which parameters to enter.

4 Dysfunctional pixels

Dysfunctional pixels are referred to by many names including *bad, dead, erratic, stuck, hot, defective, broken and underperforming.* Maintenance protocols may include the creation of so-called *maps* identifying the locations of such pixels. They are identified by a combination of criteria based on signal intensities, noise levels, uniformity and lag (see e.g. [3]). Common formats to store the coordinates of these pixels are `xml` for coordinates or `tiff` for masks. *DetectorChecker* can process both of these formats.

After uploading the file, *DetectorChecker* automatically produces an image indicating the dysfunctional pixel locations by solid squares. An example for a highly damaged detector panel is shown in Figure 3. The pixels shown are not to scale but are magnified to make them large enough so even individual damaged pixels are visible.

We distinguish different types of dysfunctional pixels depending on their spatial arrangement. This includes:

- singletons
- doubles
- triplets
- larger clusters

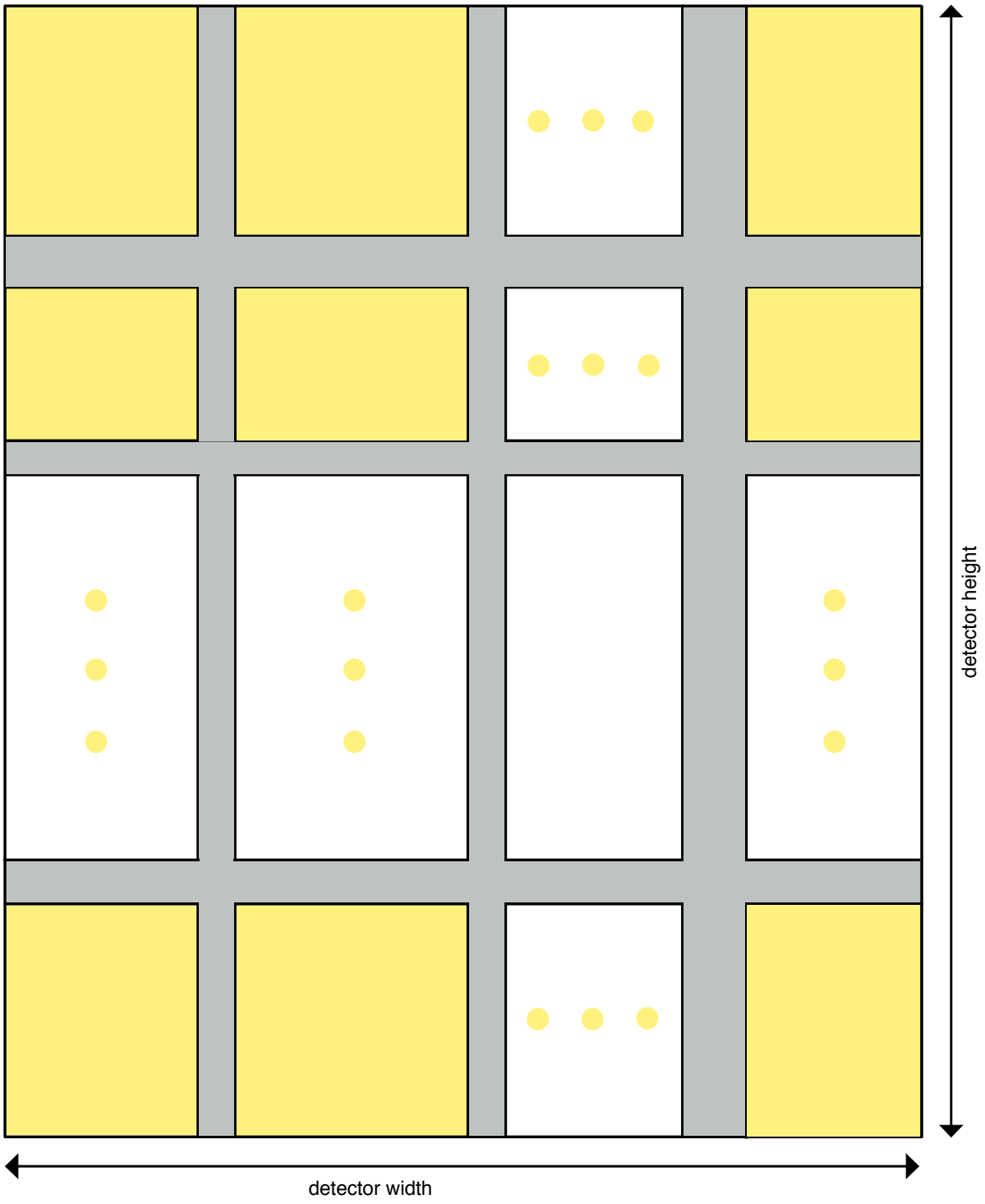

Figure 2: **General layout with modules and gaps.**

- upper and lower vertical lines
- right and left horizontal lines

singletons, doubles, small clusters, lines and high density regions ([2]). The example shown in Figure 3 has singletons all over the panel, lines intersecting with the horizontal midline indicated the two rows of subpanels, a small cluster in the top right corner and a high density region in the bottom right corner.

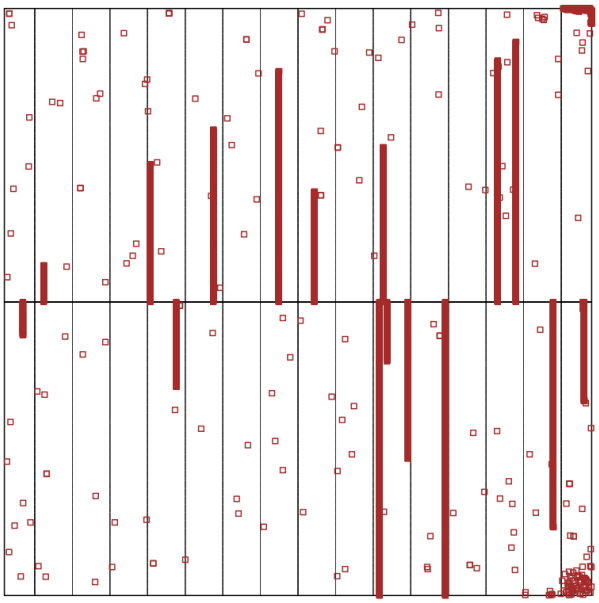

Figure 3: **Representation of detector panel with dysfunctional pixels.**

5 Spatial analysis

An important clue to the reasons for detector panel damage can come from the spatial arrangements of dysfunctional pixels. *DetectorChecker* offers a number of exploratory tools to analyse this.

To see areas of concentrated damage we recommend a density plot:
▶ In Step 5 choose *Density*.

Some of the tools offer insight into the directions between nearest neighbours. To plot an arrow for each pixel towards its nearest neighbour:
▶ In Step 5 choose *Arrows*.

To obtain the resulting distribution of all the angles corresponding to the arrows:
▶ In Step 5 choose *Angles*.

For details on the statistical methods underlying these tools see [1].

6 Levels of Assessment

While the natural entity of damage are pixels, this is not necessarily the best ways to detect the reasons for damage nor to rank their importance. For this reason, [1] introduced the notion of a higher level analysis by summarising adjacent pixels into events. Roughly speaking, lines are represented by one of their ends, small clusters by their centre etc. For more details about the pixel and event perspectives see [1].

DetectorChecker allows to use either of the two alternative levels of quality assessment, pixels or events. The default is set on pixels. To perform the alternative event based analysis proceed as follows:

▶ In Step 4 choose *Events.*

▶ In Step 6 choose Analysis as described in Section 5.

7 Modelling

In Step 2 we had the option to display some functions that assigned each pixel coordinate values to do with the distance to the edges, the corners or to the vertical and/or horizontal subpanel boundaries, if applicable. These spatial functions are used as covariates in a statistical model that aims to determine factors contributing to the decay of pixels.

References

[1] JA Brettschneider, Warnett JW, TE Nichols, and WS Kendall. Higher level spatial analysis of dead pixels on detectors based on local grid geometry. Technical report, CRiSM Working Paper Series No. 17-02, 2017. www.warwick.ac.uk/fac/sci/statistics/crism/research/17-02.

[2] JA Brettschneider, J Thornby, TE Nichols, and WS Kendall. Spatial analysis of dead pixels. Technical report, CRiSM Working Paper Series No. 14-24, 2014. www.warwick.ac.uk/fac/sci/statistics/crism/research/paper14-24.

[3] PerkinElmer. Reference Manual Digital Imaging, XRD 1621 AN/CN Digital X-Ray Detector.