Programming Visuals, Visualising Programs

Phillip Brooker
University of Liverpool, UK / p.d.brooker@liverpool.ac.uk
Wes Sharrock
University of Manchester, UK
Christian Greiffenhagen
The Chinese University of Hong Kong, Hong Kong

Abstract

This article examines the role of visualisations in astrophysics programming work, showing that visualisations are not only outputs for those producing them, but can help those developing them understand how to do their work. Studies of visualization in programming have mainly been of social and cultural factors influencing scientific research. We concentrate on the material aspects of scientific work, as of interest in their own right and on methodological grounds (since capturing the material practices of computer screen-work is an underexplored area). Using a 'video-aided ethnographic' method we analyse an episode of computational astrophysics involving the use of the Python programming language. We identify a selection of activities comprising the screen work of an astrophysics researcher to unpack how those activities contribute to the production of scientific knowledge.

Keywords: Astrophysics, programming, visualisations, video-aided ethnography

Programming visuals, visualising programs

The spread of computing throughout social life has impacted the natural sciences such that the use of computers to simulate phenomena or automate the gathering and analysis of data has become an alternative to physical data collection and experiments (for studies of computational programming work see Button and Sharrock, 1994, 1995, 1996; Knuuttila, 2006; Knuuttila et al., 2006; Knuuttila and Boon, 2011; Martin and Rooksby, 2006; Merz, 2006; Rooksby et al., 2006). Using video-recordings of a researcher testing out a program to convert electronic input relayed from an orbital telescope into a set of images enabling the identification of gravitational lenses, we explore an assortment of problems that the researcher meets in trying to ensure that his program is dependably categorising these galactic images.

Our attention to the visual features of computation reflects a growing interest in how scientists engage with visualisations (Amann and Knorr Cetina, 1990; Burri and Dumit, 2008; Carusi et al., 2010; Lynch, 2011; Messeri, 2017). Our work is aligned with studies exploring the material work

of dealing with digital images and visual data in scientific research (Alač, 2011; Carusi, 2008, 2011; Coopmans, 2006, 2011; Daipha, 2010; Hoeppe, 2012, 2014; Sormani, 2014; Spencer, 2012; Vertesi, 2012).[1] We focus on the practices involved in visualisation-based and visually-oriented research work, and how those practices intertwine with the wider scientific knowledge and context of that work. Alač notes of this:

> The materiality of the scientific data – their digital character – allows the practitioners to understand what they are working with as something that is mathematical, while it, at the same time, moves and needs to be rotated, squished and squashed. (Alač, 2011: 145)

Images and visualisations are *used* by the practitioners that generate them as part of their routine work, in such a way that "scientific visuals do not *represent* knowledge and problem solving, but are a part of such processes" (Alač, 2011: 162). Our approach to visual-work is grounded in Coulter and Parsons' (1990: 255) claim that "'seeing' is akin to an *achievement* and is not any sort of activity, process, or undertaking". Therefore we attend to the various activities that generate and construe an adequate 'seeing' of an astronomical phenomenon – the 'searching for', the 'inspecting', the 'observing', etc – on the part of one astrophysics researcher, to show more clearly what constitutes an achievement of this kind.

We begin by exploring two relevant literary bodies around the roles of computing in scientific research work and the underdevelopment of social research attending to its material practices, outlining a series of methodological concerns around capturing and analysing 'independently-executed' computer screen work. After depicting the context of the activities that form our topic, we analyse our data along six themes capturing a variety of material practices involved in the visual-work of scientific research. These themes are: making code visual; highlighting for visibility; finding through searching; finding visual utility in images; arranging for comparison, and, finally, visual diagnostics.

Background: Science and programming

As computer tools have become increasingly prominent in routine scientific work, so they have become increasingly pertinent to social studies of science, which focus on the constructing and constraining functions of interaction in an era of computational and digital science. There are two related issues in this body of work: firstly, distinctions between 'science' and 'computing' work, and secondly, the neglect of the material work of using computers to do science (relative to a focus on communal and collaborative elements).

Several studies (e.g. Agar, 2006; Bijker et al, 2016; Bruun and Sierla, 2008; Götschel, 2011; Hine, 2006; Larivière et al, 2016; Louvel, 2012; Pettersson, 2011; Mulinari et al., 2015; Rall, 2006; Sundberg, 2010; Voskuhl, 2004) present computer-aided scientific projects as comprising distinct expertises: the practical hands-on skills of programmers and the conceptual/theoretical knowledge of the scientist, combined through collaboration. Taking a selection of such studies as exemplars, this theme is apparent in Agar's claim that historically, "one difference that [the introduction of] computers made to science was deepening the division of labour – and expanding one side of the division, professional computing services" (Agar, 2006: 900). Similarly, Hine argues that:

> This division of labour [between science/knowledge and computing/programming] is conventional in [the] development of information systems. The database developer is responsible for identifying 'user requirements', and is expected to get to know users and find out what their needs are. (Hine, 2006: 281).

On the 'shop floor', scientific projects and the problem-solving work they involve are depicted as presenting the cultural challenge of combining skills and expertise by managing group work to integrate members' different capabilities. This is exemplified by the following quotations:

> This particular problem had nothing to do with acoustics or digital-signal processing. Rather, it was a problem that required those mystical skills which enable 'computer wizards' to rescue and manipulate their machines from the most hopeless situations...My informants would refer to those who were capable of successfully manipulating computers as being 'wizards' who always knew a 'trick', an obscure command, or another solution to a problem. (Voskuhl, 2004: 405)

> Feynman[2] is everywhere in this story...Against the odds, as the problems increased in size and complexity, his team continued to improve [in their ability to provide the calculative power necessary for the project]. (Rall, 2006: 955)

What these two accounts (and those of Agar, 2006; Bruun and Sierla, 2008; and Hine, 2006) convey is a sense of scientific knowledge as achieved through integrating disparate skills and understandings into a socially-constructed unified (though distributed) solution. However, where Voskuhl (2004) refers to the mystical skills of 'computer wizards' as tricks of programming, our interest falls upon what such 'tricks' practically consist of, and how they might constitute the practical work of *doing* acoustics and/or digital-signal processing with computers. Similarly, if Feynman is everywhere in Rall's (2006) story it is because Rall is narrating Feynman's endeavours as a team manager, whereas we would be interested in the story Rall *doesn't* tell of Feynman's role as a physicist.[3]

Some researchers seeking to investigate the organisation of scientific knowledge as a topic completely separable from the content of scientific knowledge; e.g. in Sundberg's (2010: 39) analysis of 'simulation code collectives' – groups whose collective and cultural properties implicate "the definition and control of simulation code use and development", whilst others extend STS' remit to include a singular concern with the cultural aspects of research. Pettersson (2011: 47) for instance aims explicitly "to analyse experimental practices among plasma physicists as gender creating processes with perspectives from masculinity studies", Götschel (2011) studies how physics has been used to reinforce misogyny, Louvel (2012) investigates the industrialisation of doctoral scientific work as representative of a grand shift in what constitutes scientific work, and Mulinari et al. (2015: 55) critique the "uneven, partial and sometimes even contradictory" neoliberal social and political factors surrounding stem cell research.

We do not dispute the findings of these studies – rather, we suggest that their accounts of 'knowledge production' in scientific research are partial, inasmuch as they do not capture the practical activities through which scientists produce knowledge in their labs (or at their computers). Thus the aforementioned researchers preclude a demonstration of the ways in which the social and cultural factors that form their topic enter into the day-to-day doings of scientific research as knowledge production. Their focus comes at the expense of acknowledging the material practices of *doing* scientific work, and how those practices execute scientific tasks – for example, the hands-on nature of experimentation in neurobiology (Lynch, 1985), or the aspects of embodiment involved in understanding how a Mars Rover moves and sees (Vertesi, 2012, 2015), or in the case of the present paper, leveraging computer programming skills to explore gravitational lensing as an astrophysical phenomenon. We aim to reinforce a focus on the 'content' of scientific knowledge (and the scientific business of making analysable records of it), by shifting focus from surrounding social and cultural factors and towards the practical activities comprising the execution of the work. Though we acknowledge the wider social context in which one astrophysics researcher's work is embedded (and account for this in detail below), the purpose of this paper is to concentrate more intently on the 'independently executed' aspects of scientific work as the underexplored counterpart to the great wealth of studies which focus more on the directly collaborative and/or interactional activities of scientists.

Methods

Our choice to focus on the material aspects of scientific programming is partly methodological – as a hitherto underdeveloped site of research, it is worth exploring what sorts of activity scientific programming might comprise even if only to elucidate on how such things might be captured

for future social research. The neglect of the material practices of computational scientific work has been attributed by Bruun and Sierla (2008) to the difficulties in locating and capturing such activities. As they note:

> Recordings of real-time actions and interactions of the project members would have contributed to an in-depth understanding of the circumstances through which knowledge networking solutions were produced. This could have been accomplished through video-recording, but many of the interactions, decisions and deliberations in research projects were difficult to capture in real time, even with a video camera, because they were not fixed in time and space…What is more, in software development much of the crucial interaction occurs when engineers browse, study, modify and integrate artefacts that have been developed by colleagues. These activities dominate the experience of most software engineers and constrain many of their decisions, but there is little overt, bodily behaviour to be observed: only mouse and keyboard use. (Bruun and Sierla, 2008: 140)

Bruun and Sierla (2008) have two complaints: firstly, that people won't stand still long enough for their interactions to be videoed, and secondly, that what does take place in a static setting – mouse and keyboard use – is not of any interest. However, they thereby overlook the sense in which the operational work of mouse and keyboard usage is embedded within scientific knowledge. It is precisely this arena involving little overt bodily behaviour in which much of the work of programming-for-a-scientific-project takes place, and it is the goings on within this arena that forms the focus of the research presented here.

It is not our claim that screen-work – work performed and achieved using the visual resources available within a computer screen – is an asocial endeavor. Indeed, screen-work is sometimes a thoroughly collaborative affair, as in the case of traders in the foreign exchange market dependent on information appearing on-screen in Knorr Cetina's (2003) examination of the role of 'scopic media' in their work, or in Vertesi's (2012, 2015) work on the role of images and image construction across the different disciplinary teams collaborating on the Mars Exploration Rover project. However, in the cases analysed here, screen-work is done without much (if any) face-to-face or even remote (i.e. online) collaboration. That much scientific activity is collaborative does not exclude the fact that it can also be performed via solitary effort. We agree with Carusi's (2011: 332) claim that there is more to visualisation work than face-to-face interaction, and that "the sociality of visual practices – the fact of their being shared by communities – is not sufficient to account for what is seen through those practices".[4] This is evidenced in Vertesi's (2015) work which attends to the ways in which images pertinent to the Mars Exploration Rover project move between two types of setting: the collaborative team-based planning meetings and conference calls, and the desks and screens of individuals scientists. Vertesi's ethnography demonstrates that though the work of image construction is inevitably achieved through individual effort – mouse and keyboard usage (cf. Bruun and Sierla, 2008) – their efforts are designed and conducted *precisely so that* they feed into, and even *display*, the broader social and cultural context work of the Mars Exploration Rover project. Failing to acknowledge the movement of images between the two settings would entirely misrepresent what it is those individuals are doing, and their reasons for doing those things in the way they do. Just so with the astrophysics researcher whose work forms our subject – we explore the specific ways in which this occurs for our case-at-hand below.

For present purposes however, it is worth noting in a general sense that the social elements of the tasks of screen-work, at least for the astrophysics researcher whose work forms our subject, are visible in the work only in an asynchronous fashion. This is something captured by Button and Sharrock (1996) who characterise the annotating work of programmers, as holding a utility not for their current task but for future users and developers of their program. In an even more fundamental sense, the work of programming rests on the performance of other forms of interactivity which consist of irrevocably social elements – no more can there be a private programming language than there can be a private linguistic one (cf. Wittgenstein, 1974)! Yet there remains a

degree to which certain episodes of project work are isolated from the communal scientific action that typically forms the subjects of sociological interest, as Vertesi (2012) notices of the embodied scientific work of remotely controlling the Mars Rover:

> During my fieldwork, I certainly witnessed situations in which such semiotic acts [embodied movements representing the physical hardware of the Mars Rover] were communicative in nature, in which a wheelie chair maneuver or a skilled twist of the elbow was a central articulation in the work of coordinating action at a distance. However, the vast majority of times I witnessed these gestures, *there was no one else in the room*. Most frequently, scientists, engineers, and technicians alike gestured in what were clearly formal, codified, standardized ways of enacting the Rover, but they did so entirely alone, speaking to mutually invisible interlocutors on a telecon line. (Vertesi, 2012: 402)

Similarly, the astrophysics researcher's work depicted in this paper may be understood as *independently executed* – work achieved in large part without guidance or consultation, though nonetheless embedded in a collaborative structure reliant on remote and asynchronous connection through infrastructure rather than face-to-face interactions.

In saying that the work is 'independently executed' we have the following in mind: (1) in relation to the gradual acquisition of professional competence, postgraduate researchers (such as HR, whose work we report) can be making the transition toward being able to work independently of close supervision and evaluation in carrying out a research task on their own behalf, (2) in relation to the task, whose execution does not depend on interaction with and contributions from others but can be carried out in (relative) solitude and (3) in relation to the division of labour within the project where the task at hand is self-contained and does not require connections to the several other comparable graduate projects that are contributing to the wider goals of the research group.

Video-aided ethnography

Methodologically, this presents a problem for a social study of science – what is to be found in a setting where nothing explicitly social seems to have happened? And what might constitute an appropriate method of capturing whatever work might be involved? We have used an analytic orientation that captures key features of the settings as they appear to those involved (i.e. astrophysics programmers). Our understandings of the setting rely on knowledge gained through fieldwork[5] as well as repeated viewings of the video. Drawing on ethnomethodology (Garfinkel, 1967) and associated video analysis techniques (Goodwin, 1994, 2001), our approach attempts to understand how the organization of the work at hand is displayed – made accountable – within the resources available on the computer screen where the work is sited. This is patent to the practitioner doing the work, it being his routine activity, but needs to be accommodated in sociological descriptions of that work. The adoption of a video-aided ethnographic method is designed to elicit access to the resources with which scientific researchers using computer technologies can achieve their work independently, and to examine what sociologists can draw from this seemingly arid environment.

This paper examines work from a larger project investigating the use of computerised technologies (typically, programming languages) in different sciences which combine research with training. The focus is on early-stage researchers doing project work toward the attainment of a postgraduate qualification. The focus on this stage in a research career facilitates the observation and understanding of the settings in question as exploratory endeavours in both scientific knowledge and method, both of which are actively topicalised by participants as part of their work. Furthermore, through that topicalising, both of those things are made available to social research, i.e. made 'accountable' for both participants and observers (Garfinkel, 1967).

Our approach to video collection and analysis draws from existing ethnomethodologically-informed studies (e.g. Alač, 2011;Sormani et al., 2017; Bezemer et al., 2011; Goodwin, 1994, 2001; Lindwall, 2008), and equips our video analytic work with a strongly contextualised under-

standing of the setting. Preparation was undertaken by the principal author to furnish the video analysis with the level of scientific competence necessary to understand the finer details of the activities at hand (see footnote five). It is difficult to quantify the time spent on preparation – preparatory work has continued throughout the analysis and presentation of the research, each iteration prompting more 'preparation' to understand previously unnoticed features of the video. Recording the video took much less time – approximately twenty hours over several days.

Results

Context of the Study

We examine one astrophysics researcher's activities over one working day, as he attends to a problem in developing a program for automatically identifying instances of the astronomical phenomenon of gravitational lensing. The researcher in question (HR) was a postgraduate student, with an undergraduate degree in physics incorporating the learning of programming languages in addition to classes in more conceptual topics (i.e. fluid dynamics, quantum mechanics, stellar evolution, etc). HR was working on his dissertation within a research group consisting of 15 other postgraduates, all under supervision by a professor of astronomy. The projects undertaken by each 'team' member were topically diverse and coordinated by their shared supervisor (who had developed each of their projects to feed into his ongoing research interests and projects). The projects underway at this time were typically designed to address technical and/or procedural research questions – the relative 'mundanities' of astronomical research which may not promise discoveries in the sense of finding and explaining new phenomena or objects, but which address the requirements for *doing* discovery work.

Returning to the ways in which HR's 'independently executed' work is conducted within a broader scientific context, we note that this is evidenced most clearly in two ways. First, that HR's position as a postgraduate researcher, working under a supervisor who manages a thematically-organised team of postgraduate researchers, places him as a cog in a grander machine. In this sense, HR's work is inherently integrated with other researchers working under his supervisor, as well as with the supervisor and their colleagues (who are vested in the success of postgraduate projects to feed into their own research). Second, and related, the code and images HR works with are *designed* to be used and viewed by others. His work (described below) is to produce a *technique* which can be replicated and applied in other scientific contexts and by other scientists. Hence, the value of HR's code and visualisations is, and can only possibly be, evaluated on the basis of their contribution to other scientific efforts. Taken in this way, the problems that HR encounters in his work (some of which we outline below) not only obstruct the successful completion of a postgraduate dissertation, but present difficulties in terms of the capacity for the work to be used by others in the scientific community. However, it is worth reiterating that for both these reasons, constant face-to-face coordination is not essential to the undertaking of HR's project, even despite its inherent connectedness with other scientists' work. HR's scientific activities are social, without co-present collaborators at the time of their undertaking.

Turning now to the specifics of HR's project, we note that his project was to investigate the potential for an automated computational method of gravitational lens detection to displace the non-automated/time-consuming practice of identifying lenses solely 'by eye'. HR's work was designed to be achievable through his independent research activities – having been given the project brief and some initial suggestions as 'jumping-off points', HR was expected (by his supervisor and by the design of his project as a postgraduate dissertation) to develop and deploy the necessary skills to complete the work individually and without need for supervisory guidance. It was HR's sole responsibility to learn how to see and read features of his visualisations, grounded in his existing programming and astrophysics learning. Despite HR's project involving writing a bespoke program, this work was conducted using widely available and 'off-the-shelf' tools which are simultaneously task-specific and all-purpose, consisting of a freely-available dataset (see below), a standard laptop computer, a program-

ming language (Python), several freely-available Python libraries providing functionality relevant to handling numbers and images within Python, and a text editor interface within which the programming language could be developed.

HR's method to identify gravitational lenses was to find, by looking at the images on the screen, two peaks of radiation emission relating to each stellar object in each of the images of his 2148-strong dataset – his data consisted of 537 possible lensing events, each of which had 4 images describing a different EM (electromagnetic) radiation profile. This information was used to ascertain if there was a visible (to HR) distortion of the radiation emitted by each object and from that decide if the image represents a gravitational lens.[6] HR's data was drawn from the Sloan Digital Sky Survey (2013), a large database of images available to scientists and the general public. The SDSS is a long-running data collection enterprise using a dedicated telescope at Apache Point Observatory (New Mexico, USA) to collect astronomical images for a variety of purposes. HR had acquired, via his supervisor, a curated subset of an SDSS dataset, containing candidate images of gravitational lenses, and it is these which HR is attempting to classify so as to develop an automated classification procedure.

The video shows HR working on a basic program he had already written which, so far, classified with a maximum 80% accuracy (this being determined by the computer's inability to produce any kind of result for around 20% of the images). To improve (and more systematically measure) the program's accuracy, HR worked on developing a manual input system which would provide his algorithmic technique with information about the coordinates of the two radiation peaks on an image, to develop his program's capacity to locate radiation peaks on the images it processed. The reasoning it thus: while some images may contain anomalies which confuse the computer's ability to decide, if the program is told which of the two peaks are relevant (and ignore all others) then it should yield better decisions about whether an image represents a lenses.

Having decided how to improve his image-classification program, HR wrote a script to allow a viewing of each of the 2148 images in turn and entry of the coordinates locating the peaks in the image. The usage of this script – effectively a front-end for contributing new metadata to each image by cycling through the corpus and appending the location by two mouse clicks – is captured on video. The process can be boiled down to the following (ideal) steps: he inspects the image to see if the position of the peaks is obvious (as is the case in Figure 1, in which there are two clear peaks with a clear lensing interaction between them). For more ambiguous cases, HR can use other images of the same galactic system in other wavelengths to cross-check against the image being worked on (see Figure 2 – also note the sub-display which magnifies the section

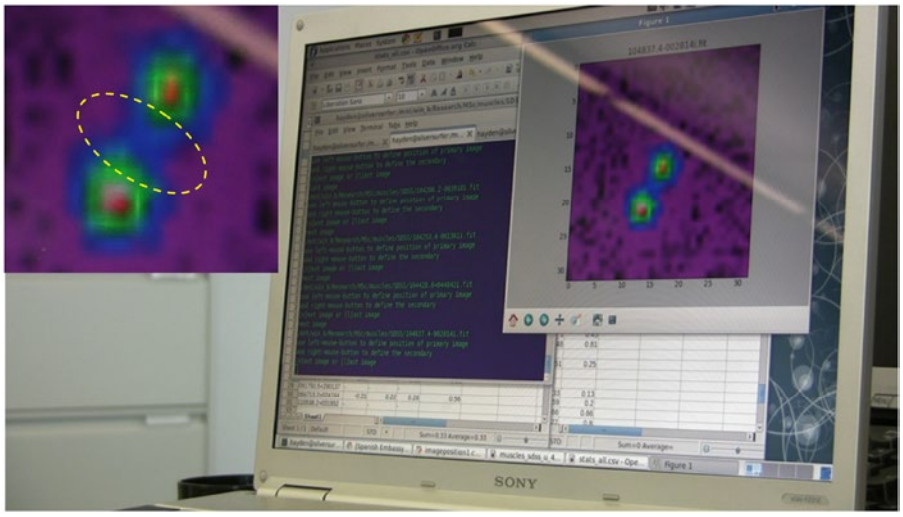


Figure 1. A 'good' lens with a clear lensing interaction (highlighted).

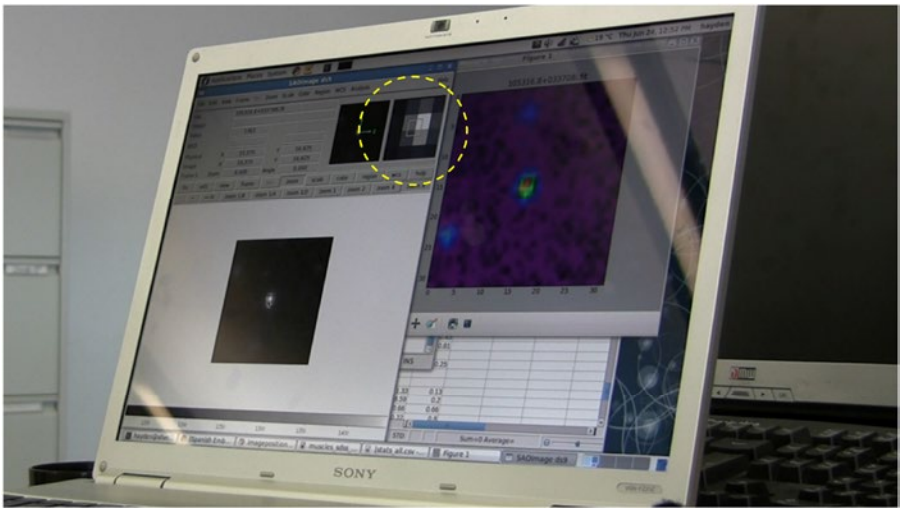


Figure 2. Cross-checking in another wavelength.

around the cursor). Having identified the peaks 'by eye', HR can record the location of the first peak by clicking on it with the left mouse button, the second peak with the right mouse button, then keystroke [n] to move on to the next image and repeat the process. Various elements of 'looking for' and 'finding' activities bear on HR's work.

Making code visual

Code is scripted text providing a list of operations (and the instruction to run them) collated under the larger structure of a program, and is written in a dedicated programming language (i.e. a software package for mathematical and computational processing) which a computer can implement. However, it is vital that not only computers but *programmers* can read and understand code, and as Davis and Hersh (1981) note of the work of mathematics (which has a direct relationship to the work of programming in a multitude of ways):

> The layman might get the idea that a skilful mathematician can sight-read a page of mathematics in the way that Liszt sight-read a page of difficult piano music. This is rarely the case. The absorption of a page of mathematics on the part of the professional is often a slow, tedious, and painstaking process. (Davis and Hersh, 1981: 281)

Familiarity and skill with a programming language is often essential to absorbing the vast amounts of code making up complex programs, but as Button and Sharrock (1995) note, the visual organisation of the code is crucial to making explicit the specific reasons as to why it might be structured in one way and not another. One method by which programmers enable understandings of their code is through comments. Comments never form a functioning part of the program; their presence does not affect the program. However, their use is common, and not only for documentation to guide future users.

According to Button and Sharrock (1996), some programmers see documentation as an annoyance that is irrelevant to the 'real' task of getting a program to work. In contrast, HR's comments are for his own use in navigating his program, highlighting their dual functions of organizing and pathfinding. Though the program is inherently structured for the purposes of machine readability – code always executes from line 1 down the page (though this may also incorporate functions and loops instructing the program to return to a previous line) – the programmer has to organise and notate this structure for human readability. As Spencer notes:

> Scientific software is an intricate labyrinth, one whose construction and navigation are accomplished by one and the same movement. (Spencer, 2012: 99)

To elucidate this aspect of programming, we examine HR's division of his code into separate

sections (to mark points where one coding task becomes another) by making a border of blue[7] commented hash marks at the start and end of each section. Practically, this means that HR can easily search for specific sections of the program, relying on visual cues. HR also uses comments to label distinct coding tasks – visual tags that make the subsequent code more understandable. For instance, HR has a line appearing as follows:

```
#BEGINNING OF PARSELTONGUE[8] SCRIPT
```

This comment marks out the following code as other than typical Python language – since ParselTongue is different to Python it is useful for HR to remind himself to read the following code as pertaining to ParselTongue specifically (as opposed to Python generally); this provides clarity when it comes to reading, debugging and other tasks. However, comments are not just labels for code. Comments can also situate code as part of a process. For instance, HR has the following comment in his code:

```
#now mask out a few pixels around this
peak position, to detect the second
peak
```

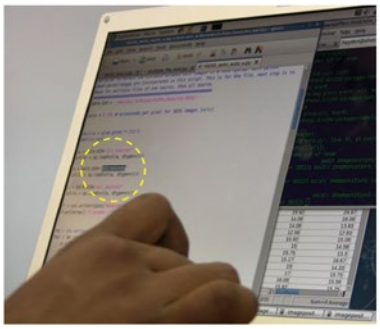
As Button and Sharrock (1995) note, the visual organisation of a program accounts for (i.e. makes visibly apparent) its own computational organisation, and comments such as this help HR to navigate through the master code screen by giving some indication of where in the code HR is if this is the section he's looking at. The comment above, by implication, relates to a section of code that must be after the section that deals with finding the *first* peak on an image. As such, if HR was to search for the specific code dealing with finding the first peak, the comment is a resource for ascertaining whether to look before or after (and also, how far before or after) the section of code currently on screen. HR enforces what Brown and Laurier (2005: 252) refer to in mobile-based cartography as a 'structure of places', which simultaneously locates the boundaries of entities within the structure (be they geographic areas or coding tasks) and renders the structure navigable. HR's practical work with comments also displays the utility of comments as navigational devices; signposts that point programmers in the right direction, helping them find the code they're searching for against otherwise visually undifferentiated lines of code.

Highlighting for visibility

HR's work also involves the integration of information from different sources (i.e. his database of manually inputted peak coordinates, image files, the master code screen, etc). HR practically transitions between windows by creating a temporary visibility arrangement through highlighting his current location in one window. In editing, HR added to a variable in the master code to integrate his new peak coordinates database into it. Effectively, he tells the computer not to look at the raw images, but to use his new coordinates databases to direct where it focuses with regard to the two peaks. This editing involves making two copies of the (linked) variables below:

```
a = DATA_DIR+'all_sources'
afile = np.loadtxt(a, dtype=str)
```

This copying of variables reflects a known feature of programming – there is 'a propensity towards re-use and economy in finding solutions rather than working out a solution from scratch' (Martin and Rooksby, 2006: 8). HR edits the copied versions of this variable by changing variable names and associated data (from 'a' to 'b' and 'a' to 'c', from 'afile' to 'bfile' and 'afile' to 'cfile' etc). Most crucially, the 'all_sources' script needs changing to reflect the filenames that HR wants the new variables to pull his manual input data from. To do this, HR must check the filenames of these databases, navigating temporarily away from the master code window to the database itself (which features the filename in its title bar). Prior to moving windows, HR highlights the 'all_sources' script in the new variable 'b', to make it stand out against the background of other code on-screen. HR then goes to the database to retrieve the filename and upon his return to the master code window, is able to use the highlight to reorient himself quickly and easily to the section of code that this filename should replace – the 'all_sources' script in variable 'b' is changed to 'imageposition1', and variable 'c' is changed to 'imageposition2' accord-



```
a = DATA_DIR+'all_sources'
afile = np.loadtxt(a, dtype=str)

b = DATA_DIR+'imageposition1'
bfile = np.loadtxt(b, dtype=str)

c = DATA_DIR+'imageposition2'
cfile = np.loadtxt(c, dtype=str)
```

Figure 3. Highlighting 'all sources', plus the finished edit of the section of code under development.

ingly (see Figure 3 below for HR doing the highlighting work, and a representation of the section of code after editing).

Here, highlighting is a quick, easy and temporary marker, which can serve as a placeholder as the code is developed (Button and Sharrock, 1995). HR's highlighting work is thus an example of a 'micro-practice' of screen- or scopic-work (cf. Alač, 2011; Knorr Cetina, 2003; Lynch and Edgerton, 1988) which is non-intrusive to the development of the program (in that it does not change the machine instructions) but can provide a visual emphasis on the script-to-be-changed to make it more 'findable' and thereby easily editable.

How to find through searching

Clearly, recoverability is a key issue for HR – he has to be able to find specific images, various databases (and particular information within them), filenames, sections of code, etc. Often, the location of the thing HR is searching for is not defined exactly and the best possible direction can only be phrased as 'somewhere within this database' or 'somewhere in this set of images'. Various practices of 'looking for' items such as these come up in HR's work, and these practices use resources available through HR's design of his working practices. As Martin and Rooksby (2006: 8) note of coding, "knowledge of the code base is knowledge of your way round it, how things might be connected and what the implications of changing a piece of code may be". This applies to HR's visualisation work in a variety of ways. For some sought after items, finding them can be simply entering a filename into a form, e.g., HR is searching for an image file in his database of peak coordinates, and, being able to refer to original image filename as it is on screen, he can copy this information into the 'find' form, keystroke [Enter], and the computer skips through the database directly to the desired filename (see Figure 4 below).

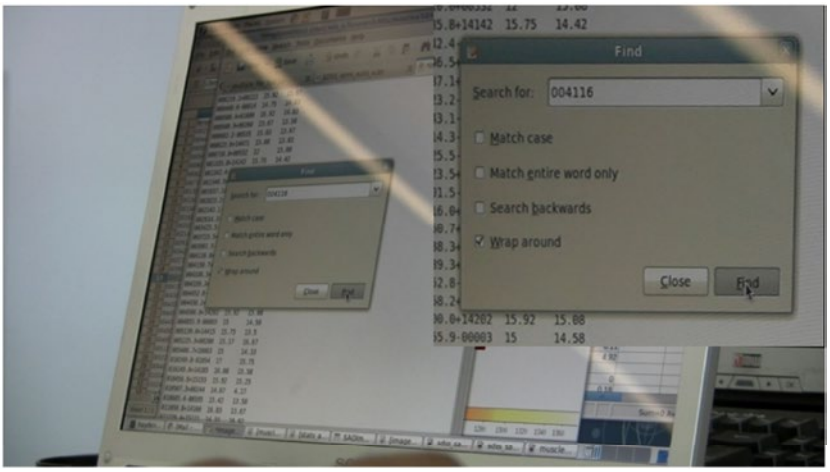
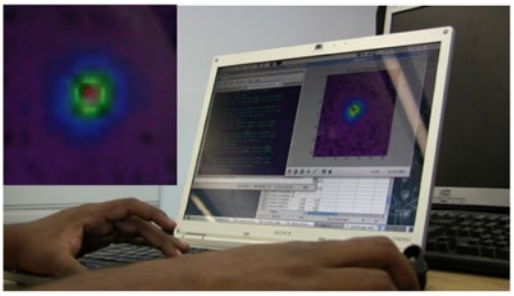


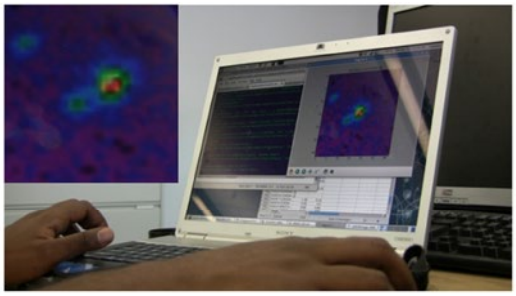
Figure 4. A 'find' menu.

In other cases a simple 'call-and-response' solution is unavailable – as Suchman (1994: 185) notes, "The problem is not simply that communicative troubles arise that do not arise in human communication, but rather that when the inevitable troubles do arise, there are not the same resources available for their detection and repair". In these cases, HR relies on other (visual) resources, e.g. HR makes a mistake in clicking on an image (image 1) and only realises this after moving to the next image (image 2) (see Figure 5 below). HR then needs to go back, re-examine image 1, delete the information mistakenly entered, then re-process the image). He does this by temporarily stepping out of the confines of the manual input/image processing work to recall it.

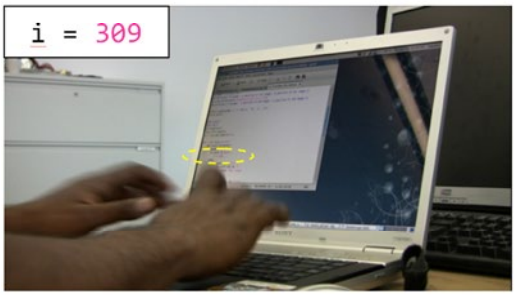
Working outside the program, HR has to call up images using the master code window. He has to start the manual input program again, but can choose at which point in the sequence of images to start: if the value of the variable 'i' is changed to 309 (as in the video), then the program calls the three hundred and ninth image in that set. HR chooses a value of 'i' that he thinks relates image 1 (i = 309), only to find that the image this value brings up is not the one he wants. He has to use other resources to ascertain the value of 'i' for the image he does want; having seen the unwanted image now on screen he can use its visual features to work out its likely position relative to image 1 (i = 309). The image on-screen at this point was the one after the image he needs to redo – he can



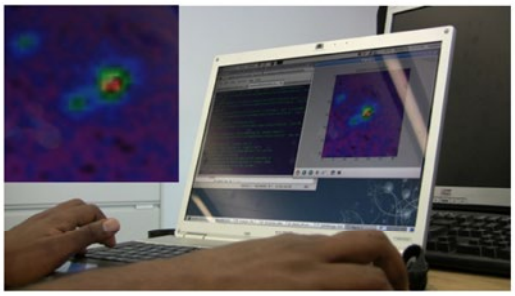
(a) HR (mistakenly) processes image 1.



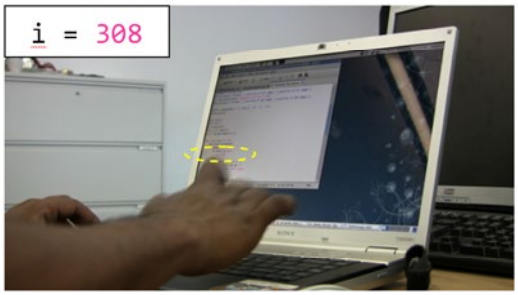
(b) Before he realises the mistake, he has moved on to image 2.



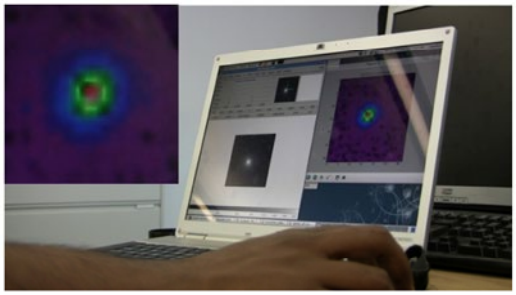
(c) He changes the value of 'i'.



(d) 'i' = 309 cues up image 2; the image *after* the one he wants.



(e) HR edits 'i' to cue up the image *before* i = 309...



(e) ...and this brings him back to image 1, which HR can then analyse and process properly.

Figure 5. Storyboard of events.

see image 2, but he wants to be able to see image 1 – and as such, HR can infer that the value of 'i' he actually requires to continue with his work is *one fewer* than 309 (i = 308). Here, HR has to draw on visual properties of the images on-screen (i.e. does it look like the one he wants? If not, can he recognise it? If so, can he pinpoint where in the sequence this unwanted image is and infer the relative position of the wanted image?) to tie specific images to their specific points in the process. As Goodwin (2001: 179) notes, "visual phenomena become meaningful through the way in which they help elaborate, and are elaborated by, a range of other semiotic fields" such as sequential organization, and by relying on various identifiable visual properties of the things he is searching for, HR is able to draw on a set of resources that makes his working with visualisations achievable.

Finding visual utility in images

HR's program is meant to distinguish between gravitational lensing systems and other non-lens objects, given an input of images of those objects in one or more wavelengths. At this point in HR's work the program is in the process of being developed; its capacity to do this consistently is therefore in question. As Lynch notes of his own work on biology lab science, 'artifacts' – "moments in the work, where the ordinary transitivity of practices was a confounding issue" (Lynch, 1985: 84) – "were not collected and analyzed in lab research, but 'fell out' as occasioned troubles in 'visibility' or 'interpretation'" (Lynch, 1985: 89). However, for HR, the possibility of artifacts is more expected given the uncertainty around the program's ability to perform classifications. HR is mindful of such artifacts appearing in his results as questions-that-have-yet-to-be-addressed – are the images the program identifies as lenses *actually* lenses? Are the other objects it identifies as non-lenses *actually* non-lenses? Are the images for which the computer produces a 'je' error[9] *actually* ambiguous? All of these questions are answerable only upon the production of results, and to determine whether or not the results the program produces are (likely to be) accurate, HR has first to classify the images himself.

The work HR puts in to classifying the set of images manually allows him to match results to images and make an informed decision about how well the program is performing, which is something the program cannot yet do. In one instance, HR comes across a 'nice' image (see Figure 6) during manual input which he picks out because of an interesting feature that is clearly visible on it – a galactic arm.[10] This feature is interesting to HR for a number of reasons, chief amongst which are that it is rare to see something so well defined among these images, which makes it of general interest astronomically. Hence, HR sets this image aside – he selects (Lynch, 1988) and values (Vertesi, 2012) it at least in part for its aesthetic qualities as a clear representation of a galactic object. However, the presence of this feature is also relevant to the current programming, in that it stands as a strong indicator that the image is a gravitational lens (because at least one of the primary objects is very likely to be a galaxy, which is the case for a good deal of positively identified lensing systems), and would therefore be useful as a test case for checking against the result the program produces. It is the finding of a distinguishing feature in a specific image that provides its utility. As HR explains:

> This looks kinda cool, I think this is a gravitational lens and is a-, this one looks very close to the...to the...so you- you tend only to have one bright lens: another one and this [the secondary object] one looks close to the galaxy cos you can see some sort of galactic arm. So, that might be nice to see what's gonna happen.

For HR, images like this, where there are criteria for judging this a 'strong' lens or non-lens, are useful in getting the program to work. Goodwin (2001: 163) notes that it is particularly important to attend to "the contextually based practices of the participants who are assembling and using […] images to accomplish the work that defines their profession". With this in mind, being able to spot these 'strong' images as they come up becomes a key element of HR's programming work. He can capitalise on his ability to make scientifically-informed visual classifications of single images, which when combined with the program's capacity to process lots of images quickly

(and with quantified statistical information that indicates how accurate it judges its results to be) provide adequate resources for refining the program. Lynch and Edgerton (1988) mark a quantitative/qualitative distinction in the scientific use of images in astronomy, citing examples of astronomers noting that images do not enable quantitative tasks, but allow for broader and more intuitive viewings of the data by eye. Lynch and Edgerton's (1988) approach, with which we would agree, is not to argue that these qualitative viewings are 'unscientific' in any way, but to recast the work of producing quantitative (scientific) results as something that can legitimately be achieved by a work process featuring qualitative (subjective, creative) elements.[11] As HR looks at the image of a galaxy with a visible galactic arm, he is able to spot at a glance what his program has (as yet) no ability to 'see'. This asymmetry between HR's and the programs' capabilities provides a tool for progressing towards a positive outcome.

Arranging for comparison

For HR, this day's work is to improve the program's ability to discriminate lenses from non-lenses (i.e. to reduce the number of 'je' errors in the results, currently in around 20% of cases). HR therefore needs to ask if this day's work is contributing to this objective, and finding a way of checking this becomes an issue. In one instance, HR compares the results produced by two different versions of the program: version 1 (the original program, which takes basic data from all images) and version 2 (the 'new' program, which integrates information about the peak coordinates defined by HR through manual input). This is intended to reveal what is happening in the new version, and both versions of results are fundamentally comparable – there are entries for each individual image in both versions. This is similarity to the reading work mammographers apply to their images, as characterised by Slack et al.:

> Mammograms are arranged to be viewed in a manner that renders the biography of a particular breast visible. Mammograms from previous screenings are juxtaposed with those from the current round. Practically, this enables the radiologist to assess if any changes have taken place and to examine features in a retrospective-prospective manner (Slack et al., 2007: 178)

Furthermore, Amann and Knorr Cetina note that, "Analyzability is not just imposed upon the visual record by labelling and other techniques. Rather, it is *built into* the record from the beginning through the way the experiment is designed"' (Amann and Knorr Cetina, 1990: 107), and in ways that rely on the visual arrangement of on-screen information in the name of facilitating the work to be done with them (Knorr Cetina, 2003). Comparably, HR has pre-designed the day's task such that he can produce, arrange and correlate two tables of results (from version 1 and version 2) for single images and use any differences in results to judge whether the new program is better or worse in terms of its ability to discriminate lenses from non-lenses.

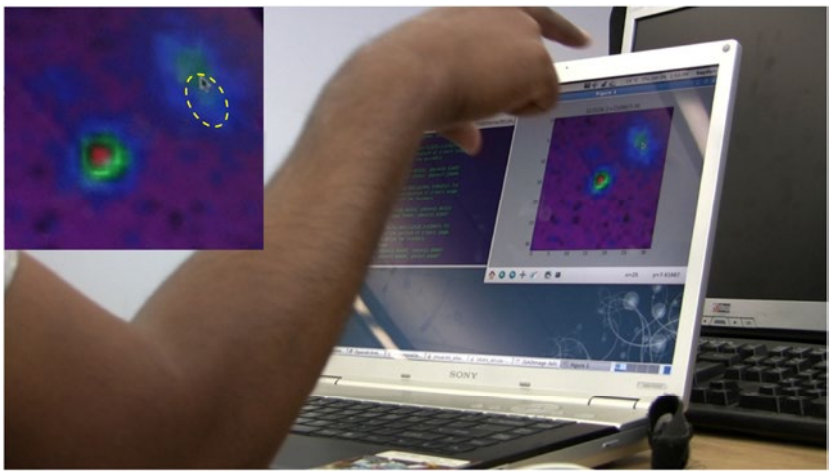


Figure 6. A 'nice' image featuring a galaxy with visible arm (highlighted).

To amplify this comparability and make it more visually manifest, HR arranges the two results screens side-by-side, such that the results for individual images are broadly on a level (see Figure 7). With this configuration of the two versions' results, HR makes an at-a-glance comparison of the first few cases – so far, the results seem improved in that there appear to be fewer 'je' errors in version 2 than in version 1. However, looking more closely, HR begins to compare individual cases from both versions' results, accenting these cases by clicking on cells within the row (thereby drawing attention to individual lines on each display to enable an easy shifting of gaze between them). Thus, HR highlights the cells in case three in version 1, then the cells in case three in version 2, allowing him to see that for this case, version 2 produces a 'je' error whereas version 1 produces a valid result. It is this fact that prompts HR to pick out case three specifically – for case three, the supposedly 'improved' program (version 2) can no longer classify an image that was classifiable in version 1. The program's capability to make a decision should have been improved across the board; that it has worsened in some cases is a possible cause for concern. HR goes through more case-by-case comparisons for cases in version 2 resulting in a 'je' error, and finds that this is not a one-off, but recurs. HR eventually attends to case nineteen (see the magnified section of Figure 7) and explains:

> [The program] gives me one [a 'je' error] here- oof! Thissa bad one. This is bad... I'll just have to go through the data to...it seems that it's not as ideal as I thought.

Because case nineteen has a particularly strong numerical result in version 1, the presence of a 'je' error in version 2 has a stronger resonance for HR's work, instigating a diagnostic approach to ascertain why this is so (see *Visual Diagnostics* below). As Lynch notes of his biology lab researchers, when their experiments failed to work, a question remained: "'Did we do it correctly? Is there anything we could have done that would have made it work?' Such questions arise in the absence of a possible authoritative resolution by means of comparisons to a standard" (Lynch, 1985: 114). HR however *does* have a standard (of sorts) since he understands how the two versions differ and so is able to use an earlier version of results as a 'substandard' (the comparative criterion being that the old results should be worse than the new). From looking at how this comparison is made, it is clear that there is a marked difference between what HR can see at first glance (i.e. that version 2 is in fact an improvement) and what can be seen on closer inspection (i.e. that that improvement has some concerning caveats which must be further investigated). Through visually arranging the two sets of results for comparison HR allows himself

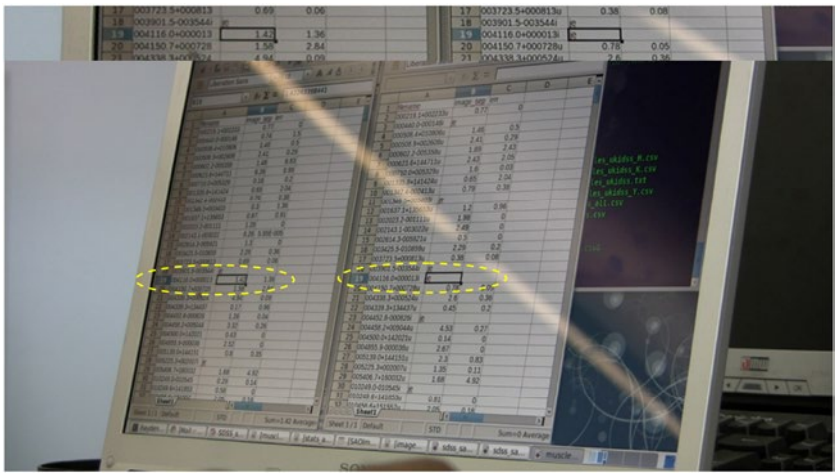


Figure 7. Comparing results side-by-side, with case nineteen highlighted in each set.

both a broad at-a-glance comparison between them, and sets the stage for a more detailed case-by-case comparison which counts towards a positive development of the project.[12]

Visual diagnostics

As with any other endeavour, working with visualisations often heralds problems, and diagnostic work must be performed to search for, locate and solve them. Complex problems might even 'hide' errors from view, and programmers might have to rely on a variety of diagnostic techniques to come to a solution. These are, in the ethnomethodological parlance, the 'normal troubles' of programming work. Given his reliance on visualisations, HR makes use of visual resources for diagnoses. To ascertain why his new version of the program is producing 'je' errors where there were no errors in the original untreated results. HR checks results case-by-case, and notices that case nineteen is giving a 'je' error in version 2 of the program but a valid result in version 1. However, the question why this should be remains – which version of the program has made the correct call – perhaps the program is *right* to call image nineteen a 'je' error if the object is genuinely ambiguous (i.e. that it is difficult to tell whether it is or is not a gravitational lens)? Or perhaps, as the weight of evidence of unexpected 'je' errors in version 2's results suggests, the program is somehow not using HR's manual input as he would like it to? To resolve his problem HR calls up the original image for case nineteen (see Figure 8 below) to classify it with his own visual judgment. As Knuuttila notes of particular types of programs used in syntactic analysis called 'parsers':

> above all, the parser must function well, which means that a parser must be able to carry out some of the tasks (i.e. syntactic analysis) that humans can. To do this, parsers do not necessarily have to be 'psychologically realistic,' and it is highly probably that they will not be so. (Knuuttila, 2006: 47)

Here, HR is attempting to ensure that his own program functions well by pitting his own abilities against the 'psychologically unrealistic' program's. From a quick visual analysis of the image, HR can see that the image for case nineteen looks to be a clear example of a gravitational lens. HR concludes that version 2 must be mistaken in its classifying of image nineteen as 'unclassifiable', and therefore it is something in the program that is at fault and not the image or the lens itself. As HR notes at this point:

> This is weird; this is a really good lens! It gave me an error on something that supposed to be, well, perfectly fine. Oh boy. This is not going to be good.

This is a significant problem for HR's project, requiring work to understand why the program is not able to classify certain lenses that he can easily classify himself. As Lynch notes of his biology lab researchers, for them, "the most interesting (and problematic) artifacts were not definite 'things,' but were 'possibilities' […] As possibilities they were not, as yet, specific features of any microscopic scene, but were tied to readings of the scene" (Lynch, 1985: 86). This is exactly how HR uses visual clues to diagnose problems – he infers, from various visual properties of what can be seen on screen, the possibilities of what might be happening. As it stands, the next obvious possibility as to what might be happening is that maybe HR's manual input – his clicking on the two peaks in each image – was to blame.

HR opens the two databases of his peak coordinates (*x* and *y* coordinates of where he clicked on the primary peak, and *x* and *y* coordinates of where he clicked on the secondary peak) to ascertain exactly where on the image he clicked. This can then be compared against the image itself – this particular screen features a cursor magnification function allowing HR to zoom in on the area around his cursor and thus locate both peaks more precisely (see Figure 2 above and Figure 8 below). Comparing his previous clicks on the image against where he would now click, having taken more care in identifying the peaks, HR finds his original clicking was not accurate enough: the coordinates in the database are some distance from the coordinates of the peaks as they appear under the magnified cursor. Therefore, HR concludes that his original manual input will need to be re-done if it is to be of any use in terms of improving the program. HR's inaccuracy is comparable with Suchman's (1994) concept of a 'garden path result', whereby during the course of his manual input work, HR:

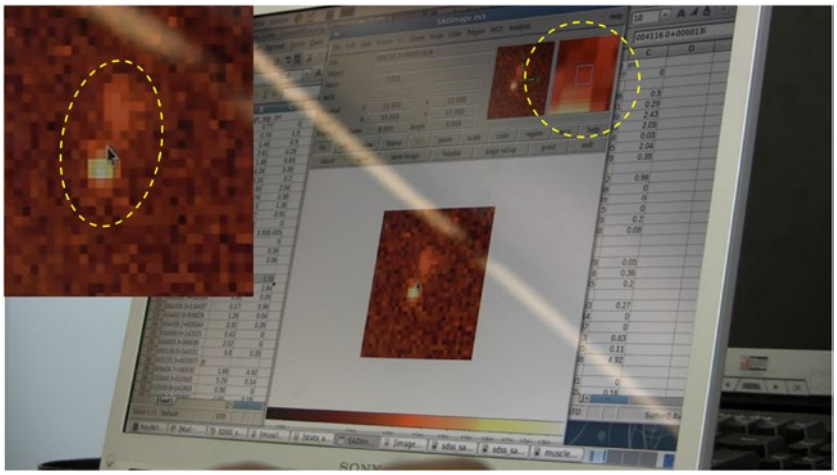


Figure 8. The image for case nineteen – the clear distortion of the radiation emitted by the two objects indicates a good lens. Also note HR's use of the magnification display to closely analyse this distortion.

> takes an action that is in some way faulted, which nonetheless satisfies the requirements of the design under a different but compatible interpretation [i.e. that two clicks have been made, regardless of their accuracy]. As a result, the faulty action goes by unnoticed at the point where it occurs. At the point where the trouble is discovered by the user [or programmer], its source is difficult or impossible to reconstruct. (Suchman, 1994: 170)

Here, however, HR *is* ultimately able to diagnose and reconstruct the trouble's source and find the problem and its solution, through looking more closely at that which (as he understands it now) he had rushed through. As Spencer (2012: 92) notes, "visualisation can also draw the scientist beyond the fact of error, towards its underlying cause and towards the future of its eventual resolution", and it is this feature of visualisations that HR draws upon in returning to the pictorial view of the data. HR is checking if the program can produce something he can identify visually, and finds the issue is his own precision placement in a visual field; his accuracy with the manual input, which limits the program's ability to consistently distinguish gravitational lens.

Discussion

The argument presented here is deeply-rooted in major themes within the field of STS dealing with the interactivity and collaboration involved in producing scientific knowledge, particularly pertaining to the usage of digital data and programming languages. This work has been characterised by some as purely a matter of the social and cultural organisation of scientific research, where success in science is achieved through the effective bringing together different knowledges and skills through collaborative interaction (Agar, 2006; Bruun and Sierla, 2008; Götschel, 2011; Hine, 2006; Louvel, 2012; Mulinari et al., 2015; Pettersson, 2011; Rall, 2006; Sundberg, 2010; Voskuhl, 2004). The present paper extends its scope to settings where there is "little overt, bodily behavior" (Bruun and Sierla, 2008: 140) other than independently-conducted mouse and keyboard use. Though we do not deny the sociability inherent to all scientific work, we focus our attention on precisely such 'independently executed' activities, in order to round out the discussion beyond the more overt social and cultural focus that has historically been given primacy in the field of science and technology studies.

With this in mind, our attention falls upon the ways in which work is achieved through the material and practical usage of screen-based resources – the visuals and visualisations that are generated and used in routine tasks that inform reasoning and inference based on what can be seen on-screen. The material aspects of scientific research work raise a perennial question for STS around a (supposed) contradiction: the experimenter's regress. Ruivenkamp and Rip (2010) describe Collins' (1992) original conception of the problem: "The unknown is to be captured in an experiment, using instruments adequate to the task. However, we do not know whether the instrument is adequate until we are sure it gives us

correct readings. But since the phenomenon itself is unknown yet, there is no way to decide what correct readings are" (Ruivenkamp and Rip, 2010: 4). This paper has aimed indirectly to puncture this standard conception by demonstrating, in the fine detail of their scopic work, that scientists can find ways to measure without opening up a regress. Hence, the routine character of these practices is (or, rather, should be) a critical topic for STS researchers. As Garfinkel et al. (1981:139) note, "Situated inquiries are practical actions and so they must get done as vulgarly competent practices". It is practices such as these that we have gone some way towards unpacking here.

Invariably, for researchers working with visualisations these practices are bound up in the visual resources available, not just within code but throughout the visualisations themselves. As Burri and Dumit (2008: 302) note, "Visual expertise also creates its own form of literacy and specialization". Such literacy involves the skill to use visualisations as resources and as *sources* of resources. Throughout the day's work HR could draw on the clues left as part of comments in his code, temporary visibility arrangements, the 'sequentiality' of images and visible features of the images themselves, the ability to distinguish by eye between 'good' lenses and non-lenses, arrangements to facilitate both general (i.e. between tables) and direct (i.e. between individual cases) comparisons of results, and comparisons of different versions of the program. This particular constellation of visual resources is useful to HR because achieving a working program is the object of his work. HR's visualisations are not simply outputs; they are new resources for doing new things. Visuality is both the topic and the means to address that topic, meaning HR does not have to rely entirely on the results produced by the program to inform his work – the results themselves can be legitimately questioned. This makes the program an interplay between the original observed data (the images) and the results, facilitating an iterative process that requires a 'building up' of understanding of what effects manual input might have on results, and accordingly, what information can be drawn from the results and associated diagnostic work about the quality of the manual input. There is no decisive criterion of which iteration might be the last, yet this nevertheless allows for the development of a program that will eventually be able to discriminate lenses from non-lenses with so few 'je' errors as to make the whole collection of results statistically useful.

We have tried to show how the practical tasks involved in visualisation-based research and programming iteratively inform each other, and more widely, how work of this kind is conducted in such a way as to contribute to the successful progress of a scientific project which is reflective of scientific research in a computational age. It is worthy of note that HR is not any kind of special combination of programmer and scientist, in that many recent science graduates now have some introduction to and hands-on experience with one or more programming languages as an essential part of their training. For HR, learning how to construe visualisations is a joint product of his disciplinary knowledge of astrophysics and his programming skill. The instances considered simultaneously reflect programming activities for scientific purposes, the two inextricably bound together in the work. What our analysis of the collected video data has shown is that despite the work at hand being visible through a computer screen and associated keyboard and mouse usage, it is possible also to attend to the sense in which it makes available a set of material practices for achieving scientific knowledge.

We have developed six 'themes' in HR's work activities, revealing a selection of work activities that are mundane and routine in astrophysics programming, but which have been, at times, overlooked from sociology's accounts owing to their material character. Without denying that scientific work is extensively collaborative and interactive and affected by social and cultural factors, we do take issue with how such a focus might be singularly applied to the effect of neglecting other aspects of what is going on. In this regard, we have explicated HR's work as a 'twinned'[13] problem-space of scientific phenomena and software. The software constructs and constrains HR's perception of the data – literally, his ability to perceive gravitational lenses in the images– whilst the phenomenon constructs and constrains the use of the software (in that his programming work relies upon an accurate scientific understanding of gravitational lenses).

References

Agar J (2006) What Difference Did Computers Make? *Social Studies of Science* 36(6): 869-907.

Alač M (2011) *Handling Digital Brains: A Laboratory Study of Multimodal Semiotic Interaction in the Age of Computers*. Cambridge, MA: The MIT Press.

Amann K and Knorr Cetina K (1990) The Fixation of (Visual) Evidence. In: Lynch M and Woolgar S (eds) *Representation in Scientific Practice*. Cambridge, MA: The MIT Press, pp. 85-121.

Bezemer J, Cope A, Kress G and Kneebone R (2011) "Do You Have Another Johan?" Negotiating Meaning in the Operating Theatre. *Applied Linguistics Review* 2: 313-334.

Bijker EM, Sauerwein RW and Bijker WE (2016) Controlled Human Malaria Infection Trials: How Tandems of Trust and Control Construct Scientific Knowledge. *Social Studies of Science* 46(1): 56-86.

Brown B and Laurier E (2005) Designing Electronic Maps: An Ethnographic Approach. In: Meng L, Zipf A and Reichenbacher T (eds) *Map-Based Mobile Services*. Berlin: Springer-Verlag, pp. 241-257.

Bruun H and Sierla S (2008) Distributed Problem Solving in Software Development: The Case of an Automation Project. *Social Studies of Science* 38(1): 133-158.

Burri RV and Dumit J (2008) Social Studies of Scientific Imaging and Visualization. In: Hackett EJ, Amsterdamska O, Lynch M and Wajcman J (eds) *The Handbook of Science and Technology Studies, Third Edition*. Cambridge, MA: The MIT Press, pp. 297-318.

Button G and Sharrock W (1994) Occasioned Practices in the Work of Software Engineers. In: Jirotka M and Goguen J (eds) *Requirements Engineering: Social and Technical Issues*. London: Academic Press/Harcourt Brace and Company, pp. 217-240.

Button G and Sharrock W (1995) The Mundane Work of Writing and Reading Computer Programs. In: ten Have P and Psathas G (eds) *Situated Order: Studies in the Social Organisation of Talk and Embodied Activities (Studies in Ethnomethodology and Conversation Analysis No. 3)*. Washington DC: University Press of America, pp. 231-258.

Button G and Sharrock W (1996) Project Work: The Organisation of Collaborative Design and Development in Software Engineering. *Computer Supported Cooperative Work: The Journal of Collaborative Computing* 5: 369-386.

Carusi A (2008) Scientific Visualisations and Aesthetic Grounds for Trust. *Ethics and Information Technology* 10(4): 243-254.

Carusi A (2011) Computational Biology and the Limits of Shared Vision. *Perspectives on Science* 19(3): 300-336.

Carusi A, Novakovic G and Webmoor T (2010) Are Digital Picturings Representations? In: *Electronic Visualisation and the Arts*, London, UK, 5-7 July 2010: 174-184. Available at: http://www.bcs.org/upload/pdf/ewic_ev10_s7paper2.pdf (accessed 3.3.2017).

Collins H (1992) *Changing Order: Replication and Induction in Scientific Practice*. London: SAGE.

Coopmans C (2006) Making Mammograms Mobile: Suggestions for a Sociology of Data Mobility. *Information, Communication & Society* 9(1): 1-19.

Coopmans C (2011) 'Face Value': New Medical Imaging Software in Commercial View. *Social Studies of Science* 41: 155-176.

Coulter J and Parsons ED (1990) The Praxiology of Perception: Visual Orientations and Practical Action. *Inquiry* 33: 251-272.

Daipha P (2010) Visual Perception at Work: Lessons from the World of Meteorology. *Poetics* 38: 150-164.

Davis PJ and Hersh R (1981) *The Mathematical Experience*. Boston: Birkhauser.

Garfinkel H (1967) *Studies in Ethnomethodology*. New Jersey: Prentice Hall, Inc.

Garfinkel H, Lynch M and Livingston E (1981) The Work of a Discovering Science Construed with Materials from the Optically Discovered Pulsar. *Philosophy of the Social Sciences* 11(2): 131-158.

Goodwin C (1994) Professional Vision. *American Anthropologist* 96(3): 606-633.

Goodwin C (2001) Practices of Seeing Visual Analysis: An Ethnomethodological Approach. In: Van Leeuwen T and Jewitt C (eds) *Handbook of Visual Analysis.* London: Sage, pp. 157-182.

Götschel H (2011) The Entanglement of Gender and Physics: Human Actors, Work Place Cultures, and Knowledge Production. *Science Studies* 24(1): 66-80.

Hine C (2006) Databases as Scientific Instruments and their Role in the Ordering of Scientific Work. *Social Studies of Science* 36: 269-298.

Hoeppe G (2012) Astronomers at the Observatory: Place, Visual Practice, Traces. *Anthropological Quarterly* 85(4): 1141-1160.

Hoeppe G (2014) Working Data Together: The Accountability and Reflexivity of Digital Astronomical Practice. *Social Studies of Science* 44(2): 243-270.

Kettenis M, Van Langevelde HJ, Reynolds C and Cotton B (2005) ParselTongue: AIPS Talking Python. In: *Astronomical Data Analysis* Software *and Systems XV*, San Lorenzo de El Escorial, Spain, 2-5 October 2005: 497-500. Available at: http://articles.adsabs.harvard.edu/cgi-bin/nph-iarticle_query?2006ASPC..351..497K&data_type=PDF_HIGH&whole_paper=YES&type=PRINTER&filetype=.pdf (accessed 3.3.2017).

Knorr Cetina K (2003) From Pipes to Scopes: The Flow Architecture of Financial Markets. *Distinktion* 7: 7-23.

Knuuttila T (2006) From Representation to Production: Parsers and Parsing in Language Technology. In: Lenhard J, Küppers G and Shinn T (eds) *Simulation: Pragmatic Construction of Reality*. Dordrecht: Springer, pp. 41-55.

Knuuttila T and Boon M (2011) How Do Models Give Us Knowledge? The Case of Carnot's Ideal Heat Engine. *European Journal for Philosophy of Science* 1(3): 309-334.

Knuuttila T, Merz M and Mattila E (2006) Editorial: Computer Models and Simulations in Scientific Practice. *Science Studies* 19(1): 3-11.

Larivière V, Desrochers N, Macaluso B, Mongeon P, Paul-Hus A and Sugimoto CR (2016) Contributorship and Division of Labor in Knowledge Production. *Social Studies of Science* 46(3): 417-435.

Lindwall O (2008) *Lab Work in Science Education: Instruction, Inscription, and the Practical Achievement of Understanding*. Linköping: Linköping University Faculty of Arts and Sciences.

Louvel S (2012) The 'Industrialization' of Doctoral Training? A Study of the Experiences of Doctoral Students and Supervisors in the French Life Sciences. *Science & Technology Studies* 25(2): 23-45.

Lynch M (1985) *Art and Artifact in Laboratory Science: A Study of Shop Work and Shop Talk in a Research Laboratory*. London: Routledge.

Lynch M (1988) The Externalized Retina: Selection and Mathematization in the Visual Documentation of Objects in the Life Sciences. *Human Studies* 11(3): 201-234.

Lynch M (2011) Image and Imagination: An Exploration of Online Nano-Galleries. In: *Visualisation in the Age of Computerisation*, Oxford, UK, 25-26 March.

Lynch M and Edgerton SY (1988) Aesthetics and Digital Image Processing: Representational Craft in Contemporary Astronomy. In: Fyfe G and Law J (eds) *Picturing Power: Visual Depiction and Social Relations*. London: Routledge and Kegan Paul, pp. 184-220.

Martin D and Rooksby J (2006) Knowledge and Reasoning About Code in a Large Code Base. *TeamEthno-online*, 2: 3-12, https://archive.cs.st-andrews.ac.uk/STSE-Handbook/Other/Team%20Ethno/Issue2/Martin.pdf (accessed 3.3.2017).

Messeri L (2017) Extra-Terra Incognita: Martian Maps in the Digital Age. *Social Studies of Science* 47(1): 75-94.

Merz M (2006) Locating the Dry Lab on the Lab Map. In: Lenhard J, Küppers G and Shinn T (eds) *Simulation: Pragmatic Construction of Reality*. Dordrecht: Springer, pp. 155-172.

Mulinari S, Holmberg T and Ideland M (2015) Money, Money, Money? Politico-Moral Discourses of Stem Cell Research in a Grant Allocation Process. *Science & Technology Studies* 28(2): 53-72.

Pettersson H (2011) Making Masculinity in Plasma Physics: Machines, Labour and Experiments. *Science Studies* 24(1): 47-65.

Pickering A (ed) (1992) *Science as Practice and Culture*. London: The University of Chicago Press.

Rall D (2006) The 'House That Dick Built': Constructing the Team that Built the Bomb. *Social Studies of Science* 36: 943-957.

Rooksby J, Martin D and Rouncefield M (2006) Reading as Part of Computer Programming. An Ethnomethodological Enquiry. In: Romero P, Good J, Chaparro EA and Bryant S (eds) *Proceedings of the 18th Workshop of the Psychology of Programming Interest Group*, Sussex, UK, 7-8 September 2006: 198-212. Salford: Psychology of Programming Interest Group. Available at: http://www.ppig.org/papers/18th-rooksby.pdf (accessed 3.3.2017).

Ruivenkamp M and Rip A (2010) Visualizing the Invisible Nanoscale Study: Visualization Practices in Nanotechnology Community of Practice. *Science Studies* 23(1): 3-36.

Slack R, Hartswood M, Procter R and Rouncefield M (2007) Cultures of Reading: On Professional Vision and the Lived Work of Mammography. In: Hester S and Francis D (eds) *Orders of Ordinary Action: Respecifying Sociological Knowledge*. Aldershot: Ashgate Publishing Limited, pp. 175-193.

Sloan Digital Sky Survey (2013) Mapping the Universe. Available at: www.sdss.org (accessed 3.3.2017).

Sormani P (2014) *Respecifying Lab Ethnography: An Ethnomethodological Study of Experimental Physics*. Surrey, UK: Ashgate.

Sormani P, Alač M, Bovet A and Greiffenhagen C (2017) Ethnomethodology, Video Analysis and STS. In: Felt U, Fouché R, Miller CA, Smith-Doerr L (eds) *The Handbook of Science and Technology Studies*. London: The MIT Press, pp. 113-138.

Spencer M (2012) Image and Practice: Visualization in Computational Fluid Dynamics Research. *Interdisciplinary Science Reviews* 37(1): 86-100.

Suchman LA (1994) *Plans and Situated Actions: The Problem of Human Machine Interaction*. Cambridge: Cambridge University Press.

Sundberg M (2010) Organizing Simulation Code Collectives. *Science Studies* 23(1): 37-57.

Vertesi J (2012) Seeing Like a Rover: Visualization, Embodiment, and Interaction on the Mars Exploration Rover Mission. *Social Studies of Science* 42: 393-414.

Vertesi J (2015) *Seeing Like a Rover: How Robots, Team, and Images Craft Knowledge of Mars*. London: The University of Chicago Press.

Voskuhl A (2004) Humans, Machines and Conversations: An Ethnographic Study of the Making of Automatic Speech Recognition Technologies. *Social Studies of Science* 34: 393-421.

Wittgenstein L (1974) *Philosophical Investigations*. Oxford: Basil Blackwell & Mott, Ltd.

Notes

1. Certainly the astrophysics research presented here is computational through and through, yet there are elements to other types of astrophysics work which are decidedly 'manual' and which may only use software rather than develop it – see for instance Hoeppe (2012) on the work of collaboratively operating a satellite telescope to collect. In this sense we say only that the specific type of astrophysics work depicted here is inherently computational, and explore how this specific type of work is achieved.
2. The Feynman under discussion is noted physicist Richard Feynman. Rall (2006) investigates his work as the manager of a computing team building the atomic bomb, which first consisted of a) untrained scientists' wives, then b) computer-trained WACs (Women's Army Corps) and finally c) soldiers with computer training and full knowledge of the project objectives.
3. The two versions of Feynman discussed here – Rall's (2006) Feynman-as-manager and our Feynman-as-scientist – are not the same in that they do not do the same things, they do not use the same technical languages, they do not talk to the same people, they do not draw on the same fields of knowledge to achieve their work, and so on.
4. It may be important to note that although our goal is the same – to see what else there might be to visualisation- and visual-work beyond interactive and collaborative face-to-face sociality – our project differs from Carusi's (2011). Where Carusi (2011) aims to explore the problem philosophically, our work treats the issue as empirical (cf. a similar debate between Bloor and Lynch in Pickering, 1992).
5. This preparatory work has involved (on the part of the principal author): talking to participants and their peers and supervisors about their project work and their role in wider research projects and groups; learning elements of undergraduate-level textbook science and mathematical techniques; acquiring a working knowledge of the Python programming language, and; attending undergraduate lectures across all four years of the University of Manchester's MPhys degree (including lectures on theoretical physics, mathematical requirements for physicists, and various aspects of astrophysics including stellar evolution, galaxies and early universe cosmology).
6. A gravitational lens is a phenomenon whereby electromagnetic radiation (ultraviolet rays, radio waves, visible light in the optical range, x-rays, etc) is 'bent' by the gravity of another high-mass object nearer to us in our line of sight. Therefore, a lensing system can be identified by the presence of an interconnected distortion between the radiation that each object emits, and a non-lens can be identified by the absence of this feature.
7. Python comments in the editor HR is using are (primarily) signified by the use of a hash symbol and appear in blue, further visually distinguishing them against other code.
8. ParselTongue is an interface to simplify complicated data reduction in Python (i.e. turning long strings of numerical information into images) with techniques from an add-on Python module (Astronomical Image Processing System, or AIPS) (Kettenis et al., 2005).
9. A 'je' error in HR's program was a result that signified that the program was unable to classify the image in question as a lens or otherwise – most likely the program has identified significant evidence for both instances (i.e. the image is a lens, the image is a non-lens) and can't thereby reject either.
10. The 'objects' in lensing systems are often galaxies. Though there are different types of galaxy, spiral galaxies (such as our own Milky Way) are comprised of a central concentrated 'bulge' of stars and a flat rotating disc of stars, dust and gas. This disc features long thin 'arms' of stars, which appear like a spiral due to their rotation.
11. This may in fact be a key reason for the continuing human involvement in science despite the sweeping advances offered by computing power – where computers are far more capable as number crunchers, they are somewhat lacking in the qualitative and creative department, which seems to be just as much a requirement for the production of scientific knowledge (Lynch and Edgerton, 1988).

12. Although these results look bad after close comparison, this is not an unrecoverable disaster for HR – it certainly is an upset that means his programmed technique for finding lenses and non-lenses is not working *yet*. However, it also points to a need (and direction) for further development and improvement, without which the project would be incomplete.
13. This is not to limit the problem-space to two factors only. This statement should be considered as part of the argument against limiting sociology's remit to only the interactional features of scientific work.