Turning Black into White through Visual Programming: Peeking into the Black Box of Computational Manuscript Analysis

Vinodh Rajan[*], H. Siegfried Stiehl[*†]
[*]*Research Group Image Processing, iXMan_Lab, Department of Informatics*
[†]*SFB 950 Manuscript Cultures in Asia, Africa and Europe*[1]
Universität Hamburg, Germany
{sampath, stiehl}@informatik.uni-hamburg.de

Abstract
In the recent past, an ever-increasing plethora of computational methods and tools for Computational Manuscript Research, and by extension Digital Paleography, have been developed and provided by the scientific community of digital image processing and analysis (or, in a wider sense, computational vision). However, invariably many of them suffer from low usability from the point-of-view of scholars from the Humanities, the actual end users. The black box problem is the most commonly cited reason for this sheer fact. While it may not be possible to completely eliminate the problem, we can alleviate it by dismantling the computational black box into smaller ones and eventually turn them white. We discuss how Visual Programming as a paradigm will make peeking into the computational black box possible. In this context, we introduce the iXMan_Lab as well as AMAP, a collaborative web-based platform that facilitates the development and experimental validation of tools and workflows utilizing an innovative Visual Programming paradigm. We will also briefly relate our approach to Design Thinking and Open Science.

1. Introduction

1.1 Computing in the Context of Interdisciplinary Manuscript Research

With the recent advances in theories, methods, and applications of various computational methods (pertaining to digital image processing, analysis and archiving - among others) in the Humanities and the consequent emergence of Digital Humanities as a scientific discipline, a burst of tools enabling thorough multi-facetted manuscript understanding and Digital Paleography has been witnessed. Even though a wide variety of these method and tools are aimed at supporting scholarly work in Computational Manuscript Research (CMR), only a few of the tools (particularly in the case of Digital Paleography) have found wide-spread and consistent acceptance and use. The reasons can be outlined as follows.

Most of the tools, assumed or even claimed to be readily applicable to real-world problems in manuscript research, are developed solely from the point of view of Informatics and often do not take into consideration the specific requirements of cognizant end-users in a specific domain of Humanities-grounded manuscript research. The main consequence of the minimal involvement of the Humanities-side is the provision of computer-based solutions (viz. tools) to end-user’s problems that are barely understandable, comprehensible and controllable by the users in the respective target domain. As a consequence, tools appear as black boxes (Hassner et al., 2014) and, thus, their usability and usefulness is acutely affected. Frequently, users do not understand how or why the tools behave and perform in a certain way or even question the results produced by them. In addition, they cannot influence or change the way the tools behave due to not being able to understand the way single tools (or their chained aggregation to a fully-fledged system) work. With only a parochial - or even no - way to understand, test or influence the underlying theories and computational methods, many of the tools end up being unusable in a rigorous academic/scientific setting (Stokes, 2012). As an important consequence, an inter-/transdisciplinary approach has to be taken, which at first demands a genuine methodology for building bridges between i) humanistic manuscript research primarily grounded in Hermeneutics and ii) computational manuscript research grounded in Algorithmics.

1.2 Document Image Analysis

According to Kasturi et al. (2002), Document Image Analysis (DIA) refers to ‘algorithms and techniques that are applied to images of documents to obtain a computer-readable description from pixel data methods’. In our

[1] https://www.manuscript-cultures.uni-hamburg.de

context, the general term documents refers to either contemporary or historical prints and manuscripts, while images is synonymic to digitized manuscripts. DIA methods such as image preprocessing, keypoint detection, visual feature extraction, text line finding, page layout analysis, writing style analysis and hand identification are employed within Computational Manuscript Research for a wide variety of purposes - ranging from simple image quality enhancement to increased legibility up to advanced applications such as writer identification for paleographic dating. Given the stage of research in computational vision, however, it is safe to state that tools for particular tasks and domains cannot be directly derived from published theories and methods for at least two reasons: Almost all methods entail parameters to be carefully adapted to the particular task and domain at hand and, worth noting, their performance in terms of, e.g. accuracy, reliability and robustness, severely depends on the quality of the digitized image (e.g. resolution and contrast as well as degradations such as noise). In a strict sense, also methods or services from open source repositories (such as Github) are hardly applicable to a specific task in a, say, blind fashion – let alone the myth of error-free software. Consequently, for computational vision, the real issue is multi-faceted and multi-staged: From theory to algorithms to methods to experiments to tools to services and systems – in total with the aim of supporting the workflow of scholars from Humanities involved in their hermeneutic cycle. Since computer-assisted manuscript research on the basis of tools also requires thorough validation, evaluation and benchmarking of methods and tools from above, scientifically grounded – but, alas time-robbing - experimentation with synthetic, real and ground truth data is indispensable even only for, e.g., constraining the space of parameters and understanding the propagation of errors through a chain of methods. Last but not least, two more critical aspects have to be mentioned: First, the lack of a clear-cut methodology and thus also engineering approach in order to consistently link the realm of theory to the reality of workflow support for scholars and, second, the involvement or even embedding of affected scholars in the whole, notably cylcic, process of requirement analysis, system design, realization, evaluation and re-design as well as provision and maintenance of performant, web-based, interoperable and platform-independent tools. Clearly within contemporary Informatics such an integral approach has Design Thinking including advanced, e.g. agile, software design and development as one of its pillars and lays foundation for a rather much needed than only desirable methodology.

Many of the tools can be, simplistically speaking, considered to be a composite of various DIA methods. However, such a composition is not as simplistic as it may appear to a layman or even nitwit, and the resulting tools require proper consideration in terms of building blocks, or sub-modules, and their connectedness, parameter regimes, applicability, appropriate presentation of results, among others (see also above). Many of these building blocks are not exposed to end-users, and even if exposed, do not provide a meaningful way of interaction in order to truly keep the user in the loop and in control. This invariantly results in a composite tool becoming a monolithic black box that end-users have trouble interacting with and trusting in.

1.3 User-in-the-Loop Paradigm

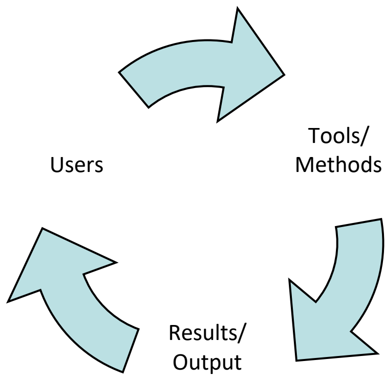


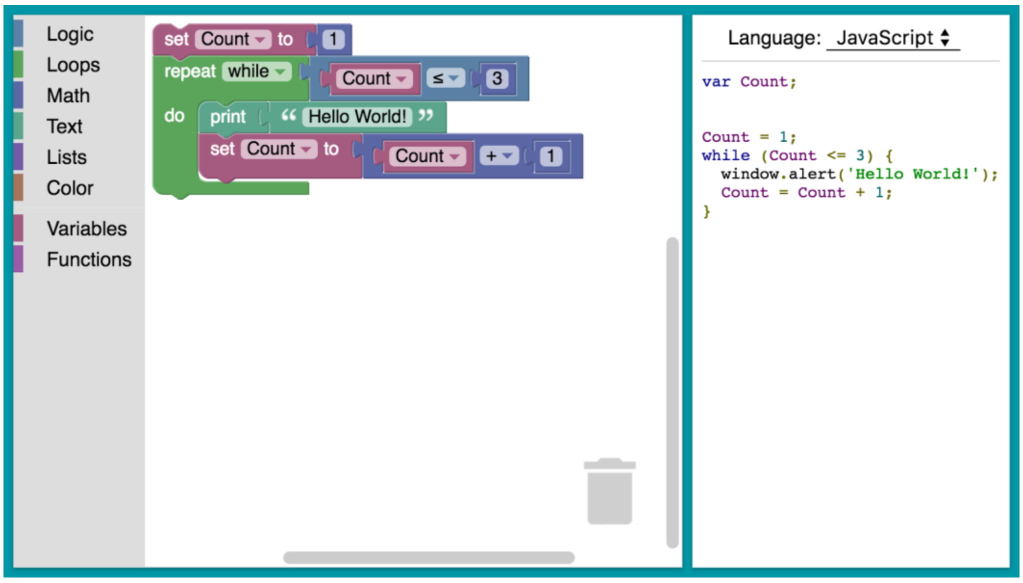
Fig. 1: A simplified User-in-the-loop

User-in-the-loop as a paradigm (Hassner, 2013) is often suggested as a work-around to resolve many of the critical issues mentioned above. In general, it attempts to give more control to users by i) opening up the tools to further useful and purposeful interaction, ii) having more control over the range, scale and configuration of method and iii) even providing feedback for improvement of e.g. overall performance. The main idea here is to i) involve, not to say embed or even immerse, users – being just experts from their respective fields – in the cyclic process sketched above in the best possible way and ii) metaphorically speaking, make them part of the algorithm and the methodology. Thus, what is in order is transforming the tools from being computational soothsayers that take in data and render the divination of results towards actually performant, reliable and trustworthy systems that work in tandem with users taking into serious consideration their needs, alterations and feedback. From the point of view of *condicio humana*, a tool (or computer in general) is to serve a human and not vice versa. Hence, a scholarly expert in manuscript research spiraling the hermeneutic cycle should by no means be demeaned or vilified by a technologically allotted role as key stroker or mouse clicker (see below).

While we adopt the user-in-the-loop paradigm as the right and principal approach to open up the seeming black box, the paradigm appears quite late in the software engineering process. We strongly propose that gaining an understanding of the black box must ideally occur quite early into the whole process outlined so far, and in fact, this would mean that users must be ideally involved in the initial stages of the process of designing and building tools as well and not just in the final stages of using them, be it for experimental or practical purposes. This allows users to i) get at least an idea of what exactly goes into the tools and ii) garner a better understanding of the final tools or even system including internal components, performance characteristics, degrees-of-freedom, parameter sensitivity, etc. As will be argued below, the adoption of Visual Programming (VP) as a tool-building paradigm will enable us to accomplish the goals laid out above in an efficient, effective and user-friendly way.

1.4 Visual Programming

Visual Programming (VP) is a rather novel but pioneering programming paradigm that allows users to develop computer programs by spatially arranging software modules, or computational methods in our case, as graphical symbols in typically two dimensions, e.g. on a monitor or screen (Myers, 1990). Hence, users are enabled to build programs by putting together computational “Lego”-like blocks on a screen, or even a multi-touch table as in our case, in an interactive fashion. The graphical symbols could be either low-level containing only control structures and variables or, optionally, high-level supporting various abstract modules relevant to the application domain. Several types of VP paradigms have been made available as yet, the most popular ones being block-based, and graph-based or flow-based VP languages. Blockly (Fraser, 2015) and AppInventor (Wolber, 2011) are block-based visual languages that allow users to arrange and fit various blocks into pre-defined slots and holes to create programs. On the contrary, Microsoft Visual Programming Language is a flow-based visual language that allows users to define programs as a graph using nodes and edges, with the nodes usually denoting some sort of processing and the edges the in- and out-flow of data. Although indeed several other VP paradigms do exist, they are neither very popular nor relevant to our current research span.



Language: JavaScript

```
var Count;


Count = 1;
while (Count <= 3) {
  window.alert('Hello World!');
  Count = Count + 1;
}
```

Fig. 2: Google Blockly

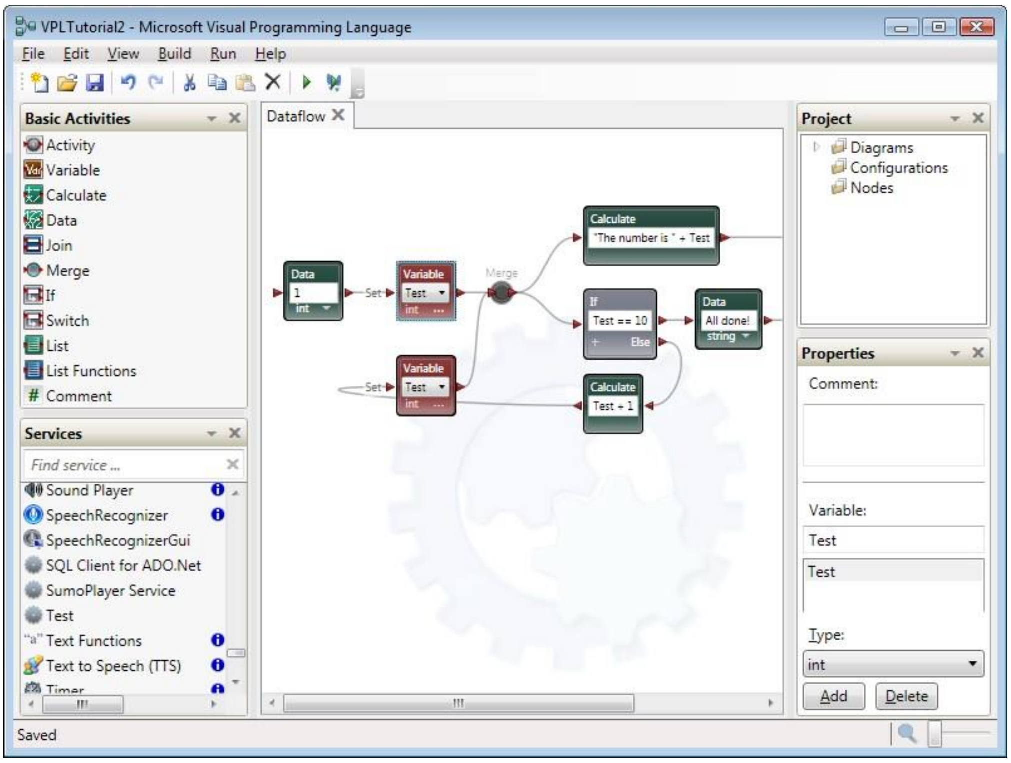


Fig. 3: Microsoft Visual Programming Language

VP is becoming a popular programming paradigm in various domains, particularly in education to teach programming concepts and in Do-It-Yourself (DIY) environments for domain specialists to create both programs and workflows. VP allows specialists to create (or at the minimum, to co-create) solutions to well-defined domain problems in an easy and interactive way without the overhead of learning a mainstream textual programming language. A prominent example for the latter would be AppInventor, which provides a VP language and the necessary environment to develop web apps without the need for any programming experience.

2 Peeking into the Black Box

To properly eliminate, or at least alleviate, the black box problem, one has to completely understand the mathematical and functional limitations of all the computational methods that make up a tool or system. Often, this is not achievable without a strong mathematical background of tool developers, particularly when it comes to advanced DIA methods (see above) and even more cutting-edge methods e.g. stemming from the research field of machine learning (ML) and, particularly, deep learning (DL) in layered neural networks, which are frequently claimed to be black boxes per se due to their lack of accountability and transparency (or, in a strictly algorithmic sense, lack of proof of correctness and uniqueness). While it may not be possible to completely eliminate the black box property itself, it is feasible to deconstruct them into multiple smaller entities, which themselves could be black boxes nevertheless, such as to reduce complexity down to a level of better transparency and, thus, understanding. This allows both peeking into the seeming black box and being able to conceive the multitude of interconnected parts contained within it. On a practical level, sometimes it is just enough if one understands the different components that make up a hierarchical, or trivially sequential system and how they interact with each other (e.g. via linking data or cross-effects of parameters as in the simple case of a typical image processing pipeline for easy tasks). Evidently, in such cases, a great deal of understanding can be easily provided. This is required in order to understand the overall workings of the system and, accordingly, tailor or streamline it towards compliance with user requirements. The opaque nature of the decomposed subcomponents can further be reduced by exposing their working through appropriate visualization techniques.

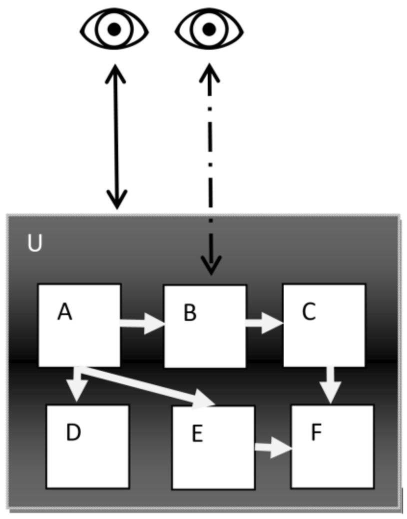


Fig. 4: Users usually only see the overall system *U.* But it is necessary they also know about the subsystems

Again, due to the preponderance of non-trivial and non-mainstream scientific problems in interdisciplinary computational manuscript research, e.g. generic layout analysis or hand identification given abundant degradations in digitized manuscripts, sparseness of training data, lack of ground truth and benchmarks far-off scholar's workflows, the demanded tools and systems do have a complexity in theoretical, methodical, technical and human-factor terms, which is way off toy problems such as recognizing cats and dogs or preventing a moving robot from ramming a table leg (though engineering feats).

As already mentioned above, with the rise of training data-intensive deep learning (DL) methods, it simply is more difficult or even virtually impossible to open up such systems, as a sufficiently deep neural network itself is a black box from both a theoretical and algorithmic perspective w.r.t. only transparency. However, even DL tools are not always monolithic, since some kind of pre-processing and post-processing steps almost always go alongside the core DL method. Hence, ongoing research in the respective communities is about to open those as much as possible again for the sake of a better understanding on the sides of both the developers and the users – particularly in the context of applications with the need for trustworthy computational results as in signature verification. Although scientific problems with respect to CMR may well be characterized as non-critical ones (e.g. in comparison to fraud detection), accountability, transparency, trustworthiness etc. are still for good reason key features of tools and systems – otherwise the willingness of scholars for their utilization in their daily routine will fade or not even be kindled. On top, by briefly touching upon issues of accuracy, reliability, replicability, etc., an added-value of any CMR tool/system devoted to support solving scholarly research questions must be clearly demonstrated – otherwise it's l'art pour l'art or a nogoodnik.

As already mentioned, a VP environment can be helpful in deconstructing computational black boxes by taking two different approaches. In the first approach, a VP-based DIY environment can be provided to manuscript scholars in the Humanities. It is seldom possible to provide specific computational solutions that are generalizable and applicable to scenarios with problem settings that scholars face on a day-to-day basis. Therefore, we better focus on providing them with genuinely required toolsets that would let them i) explore various methods on their own, ii) deal with digitized manuscripts and iii) create custom-built solutions by themselves in an interactive way using the VP paradigm. They can choose various techniques, explore them and assemble them together to produce solutions in accordance with their needs. Needless to say that a certain training of CMR novices and/or IT affinity is presupposed even in this ideal. Since, nothing comes indeed from nothing (and academic ignorance up to hauteur is hard to cure). In the second approach, a VP-like environment is used as part of the software engineering methodology to interact and communicate with scholars from the Humanities. By using such an environment, experts from Informatics are empowered to discuss/develop various solutions effectively by i) providing the scholars with a hands-on interactive experience in solution building and ii) admitting them a co-creating role. Rajan & Stiehl (2018b) coined this approach "interactive Exploration (iX)" and discussed it in quite a detail as part of a Software/System Development Methodology (SDM) for tools in CRM. In both approaches, users (in our case manuscript scholars in the Humanities) dismantle the monolithic black-box-like solutions either by themselves (in the first ludically-like approach) or by jointly working in tandem or team with CMR experts being sufficiently knowledgeable in computational vision (as in the second more principled approach by explicitly constructing solutions as an assembly of more basic subcomponents in order to gain an understanding of their interaction and, hence, the overall functionality of the system).

Even in the worst case of users not being able to participate in the tool creation process, VP-like features can still be used to i) create an effective visualization of the final tool structure and functionality and ii) communicate the overall working mode and scope of the tool to the end-users. Not only will this still help in making the black box at least translucent but also utilized as a teaching and training environment for students and up-and-coming academics with strong interest in CMR.

3. Using Visual Programming to Turn the Black Box White

Below, we elaborate on how various features and properties of VP enable the dismantling process and also facilitate further opening up the black box in various other ways.

3.1 General Advantages of Visual Programming

Visual Programming offers several advantages compared to textual programming. Blackwell (1996) notes that ‘typical statements are that VP is more user friendly, helpful, satisfying, intuitive, readable, familiar, appealing, accessible, reliable, pleasant, straightforward, alluring, immediate and obvious than other programming techniques’. As such, VP encourages non-programmers to play around with visual elements, letting them explore and freely experiment with digital objects to attain their desired programming objectives. A functional computer program could thus be created in a short-period of time by merely putting some graphical objects in an orderly manner. Also, a solution produced in a VP language can be (under certain circumstances) more understandable and communicable compared to a solution produced in a textual language.

One of the main advantages relevant to our problem setting in CMR is that VPs are better at expressing the problem structure. Their diagrammatic nature coupled with the semantic spatial arrangement, allows users to better grasp the structures of a reasonably complex solution. The other major advantage is its resemblance to the real world of a particular application stemming from hermeneutic manuscript research. By mimicking the real world by its visual representation, a VP language can map the manipulation of real-world objects to those of digital objects by choosing an appropriate interaction metaphor. Such factors can be seemingly taken advantage of in order to minimize the black-box-problem by enabling users to understand the intricacies of the tool or system better.

3.1 Specific Features of Visual Programming

Visual Programming as a paradigm attempts to implement four core features: Concreteness, directness, explicitness and liveness (Burnett, 2002). In the following, we explore how those specific features can help to open up the computational black box with specific references to creating tools in CMR.

3.1.1 Concreteness

Concreteness denotes expressing programmatic aspects using particular instances, e.g. mapping some aspect of semantics to desired behavior using a specific object or property. A black and white brush realized as a tool could denote a binarization (i.e. turning color images into black and white) process and the size of the brush could directly be proportional to the threshold of binarization (i.e. pixels below a certain threshold become white). Thus, we are mapping abstract methods such as binarization to concrete graphical entities in a VP environment. By converting the abstract into concrete, users get a better grasp (also in the physical sense of the word) of the inner workings of a tool.

3.1.2 Directness

Directness can be described as ‘the feeling that one is directly manipulating the object’, which implies a minimal distance between an objective and the actions required to achieve it. This is usually implemented by choosing an appropriate interaction paradigm that maps the digital objects to appropriate real-world metaphors. To continue the previous example, binarizing a digital image could be implemented as moving a brush over the image. This allows users to intuitively interact with the system directly and make changes.

3.1.3 Explicitness

The internal aspects of a system are made visually explicit allowing requiring users to infer those aspects intuitively. Particularly, in our context this refers to making dataflow in e.g. a chain of computational methods explicit by visualizing the intermediates and also making parameters associated with various methods explicitly visual for direct control. By exposing the various methods, parameters, dependencies and the interconnectedness of the components by use of graphical objects and visual metaphors, users get a better understanding of the overall tool or system for CMR.

3.1.4 Liveness

The immediacy of feedback that is automatically provided by a program, tool or system is termed as liveness. Tanomoto (1990) enumerates four levels of liveness. The first level corresponds to the static visual representation of the system and is by no means interactive. It is meant to be only a diagrammatic representation to help understand the structure and flow of a program. CMR tools must strive to provide at least this level of liveliness, even if they do not use the VPL paradigm. In the second level of liveness, the system is interactive, and users are able to build the system with graphical elements and run the system to view the results. But users must explicitly execute the setup to view the results. In the third level, the users need not explicitly run the system whenever something is changed, since the system automatically runs in response to changes initiated by users. This encourages scholars to explore and try out different combinations e.g. sub-modules and parameters and get immediate feedback. In the last level of liveness, the system is always on and provides temporal feedback based on the current state of the system. This can be very relevant for DL systems that typically take a long time to train or any system that handles high-volume data streams. A continuous visual feedback will keep the users engaged and interested by providing a glimpse of the current set and state of running processes such as computational methods along with their intermediate results in dependence of the parameter settings. Also, e.g. in the case of experimentation with CMR tools in the iXMan_Lab, it is useful to keep full track of the progress by compiling a comprehensive lab logbook in order to stick to standards in scientifically grounded experimentation (as known from paragons in experimental physics, psychology or social sciences).

4. iXMan_Lab

In these CMR, DIA and VP contexts, we now introduce the *iXMan_Lab* (interactive eXploration of Manuscripts Laboratory) whose realization is one of goals of the Scientific Service Project Z03 of the SFB 950. The underlying motto for the laboratory is to develop concepts, paradigms and prototypes that contribute to the realization of usable and useful CMR tools for manuscript scholars, which they can use in their day-to-day activities as discussed earlier. More specifically, not only methods and tools being developed within Z03, e.g. for word spotting and writing style analysis (see e.g. HAT 2.0), but also open source methods and tools (e.g. from OpenCV; see also OpenX Workshop of June 2018[2]) as well as web-accessible services from various sources (see e.g. DIVAServices of Université de Fribourg) will be integrated in order to enhance the current scope of functionality. Web access to the lab is assured due to interoperability and platform independence – also meaning ubiquity of CMR functionality for scholars out-of the Department of Informatics (as current lab host) only equipped with standard IT equipment such as desktop computers, laptops or tablets (whereby touch-based devices are preferred).

The lab is driven by an interdisciplinary team while utilizing a multi-touch table environment (powered by high-performance computing equipment) as a collaboration, cooperation and communication (C3) medium for a two-fold aim: First, experimentally designing a manageable, feasible and reliable processing chain based on computational vision methods for processing/analyzing digitized manuscripts and, second, freezing-in a jointly

[2] https://www.manuscript-cultures.uni-hamburg.de/cal-details/180612_OpenX_Programme.pdf

validated (or even evaluated or benchmarked) processing chain by interdisciplinarily reached consensus in order to deliver a useful tool for a broad range of users. In terms of hardware infrastructure, the laboratory is currently equipped with a custom-built 65-inch Multi-Touch Table (MTT) supported by a high-performance multi-core gaming engine. Furthermore, the MTT is additionally capable of being adjustable to a wide range of height and angle settings in order to allow for various forms of team-wise collaboration. The laboratory is completely equipped in terms of running both GPU-accelerated image processing/analysis algorithms, and if necessary, deep learning methods as well.

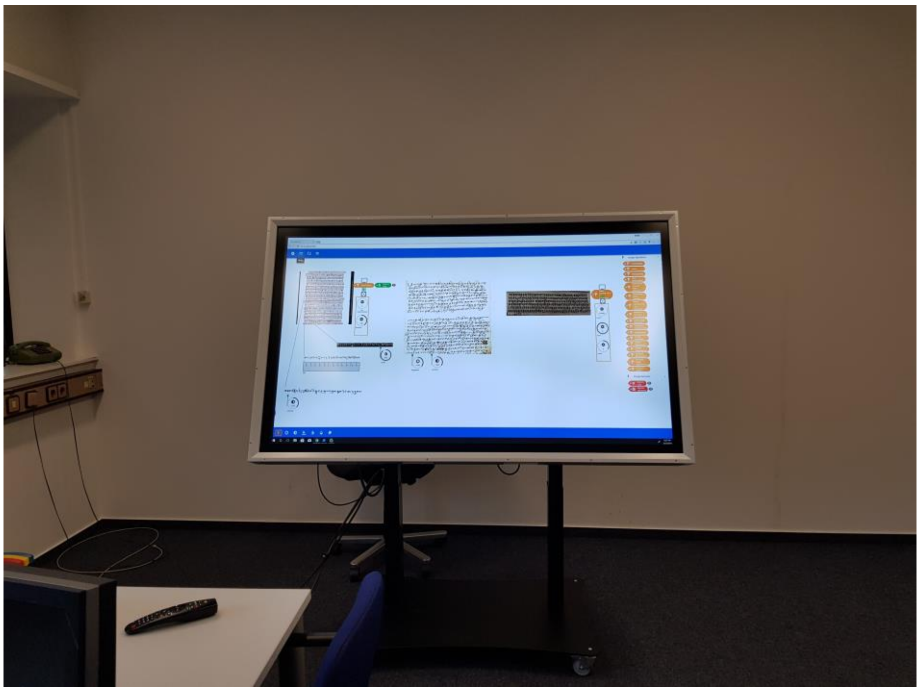


Fig. 5: The MTT setup in iXMan_Lab

Even though primarily situated within the Department of Informatics as mentioned above, the lab is uniquely placed within the Centre for the Study of Manuscript Cultures, the host of SFB 950, as well through its ability to web-interact with various scholars from sub-projects of *SFB 950*. In summary, the laboratory, once fully-fledged, will allow to i) perform meticulous requirement engineering, ii) design and realize experiments, and iii) provide workflow-supporting tools due to close interaction between scholars from Manuscript Studies and Informatics.

5 Advanced Manuscript Analysis Portal (AMAP)

Currently, the main focus of the iXMan_Lab is concerning the further development of the *Advanced Manuscript Analysis Portal (AMAP)*, the conceptual development of which started 2015 in the Scientific Service Project Z03 with the begin of the second funding phase of the SFB. In Rajan & Stiehl (2018a), a first outline of the design, architecture and functionality of AMAP is given, being equipped with an intuitive interaction paradigm in the context of a multi-touch table. In brief, it will allow users to i) intuitively deal with various advanced image processing/analysis methods and other manuscript-related methods and ii) create their problem-specific and thus customized chains of computational methods. The general goal is to design and develop AMAP in such a way that even advanced methods could be applied in an easy and intuitive manner by scholars without any programming and only rudimentary technical background. Furthermore, we are designing AMAP to be able to encourage and foster the exploration of various methods, tools, services and workflows and, at the same time, to enable ease-of-use without any steep learning curve. AMAP ultimately aims to provide a VP-based environment and to support both approaches of deconstructing the black box (as discussed in section 2). It can serve as both a DIY-like environment for scholars working in the Humanities and at the same time as an experimental platform for facilitating interaction and communication of interdisciplinarily constituted teams from Humanities and Informatics during the Software Development Process.

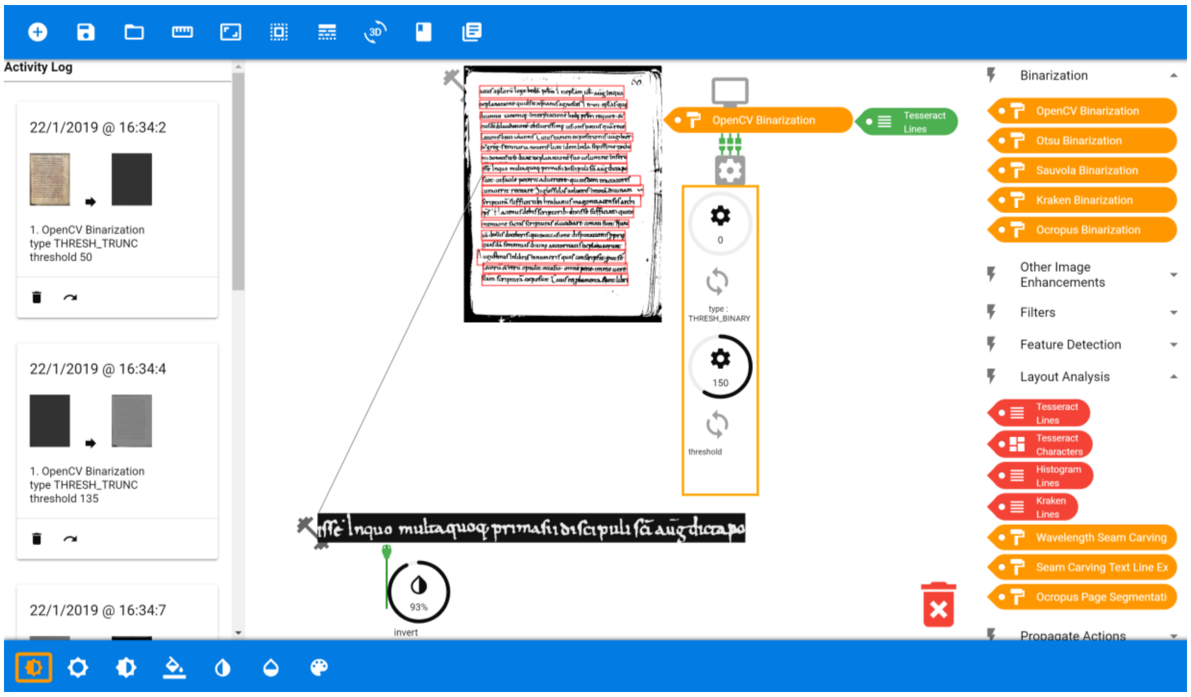
We particularly chose to realize AMAP via an MTT, as touch-based technology is gaining huge traction and has the potential of becoming the primary mode of interaction in the near future. Even today, touch-based interfaces are becoming increasingly popular compared to the traditional Windows-Icons-Mouse-Pointers (WIMP)-based interfaces. Also, having a large-scale interaction/interface area available is necessary to interact with multiple high-resolution images, which is usually the case with analyzing digitized manuscripts. The MTT can be further augmented to allow for multiple input modalities that could be harnessed to make the system even more natural by its ability to model and mirror physical real-world interaction, e.g. by speech and deixis, with manuscripts as much as possible. A MTT is also an ideal medium to encourage real-time collaboration of a team of scholars from Humanities and Informatics through the provision of a sharable large-scale monitor-based interaction device.

AMAP currently offers a rich selection of various functionalities such as image filtering, binarization, visual feature detection, word spotting, page layout analysis, and writing style analysis (Mohammed et al., 2017). In fact, multiple methods providing the same functionality are also included to enable users to experiment and choose a method that is best suitable for their task at hand. Our system also offers the ability to integrate other backend systems that provide DIA techniques as web-based services. This has been realized by integrating methods available at DIVAServices (Würsch et al., 2016) as part of AMAP. Such integrations demonstrate the flexibility of our approach as well as the ability to assimilate wide-ranging manuscript-related computational methods from the OpenX community (where X stands for data, methods, tools, services, etc. in accordance with the Open Science paradigm) into our platform.

5.1 VP & AMAP Design

We are currently implementing an innovative hybrid VP language that integrates both a flow-based approach and a block-based approach. The paradigm works on the principle of visualizing the digitized documents and computational methods as virtual objects that can be manipulated spatially in relation to each other in order to aggregate various processing chains e.g. in the context of experimentation and/or to create task- and problem-specific workflows for scholars from Humanities. The UI is particularly designed to reflect real-world metaphors as much as possible in terms of interaction. Users can i) directly attach/detach various methods to/from the digitized manuscripts before and during chaining to processing pipelines and ii) manipulate parameters and subsets of manuscripts to process them even further. Additionally, it also supports natural-like interaction includes actions such as piling the pages and turning them to take notes.

Fig. 6 shows a sample screenshot of a rather simple processing chain that has been realized with AMAP for demonstration purposes. It includes a textline detection method which requires a binarization step beforehand as preprocessing. Instead of being a single block, this decomposition explicitly shows that the quality of the textline detection is at least partially dependent on the pre-processing step and by controlling the preprocessing step the quality of the results can be adjusted to the user’s need. Also, by adding several other preprocessing steps, the results can be improved even further through goal-directed experimentation. The user is able to have some control over the system and understand how the process works, as opposed to a single processing block that only provides the output to a given input. Furthermore, a detected single textline from the output can also been seen to be extracted as a subset and a filter being applied on it specifically to increase the readability. One can also see a digital logbook recoding all previous operations performed with AMAP along with their timestamp and parameters of the operations.



Activity Log

22/1/2019 @ 16:34:2
1. OpenCV Binarization
type THRESH_TRUNC
threshold 50

22/1/2019 @ 16:34:4
1. OpenCV Binarization
type THRESH_TRUNC
threshold 135

22/1/2019 @ 16:34:7

Fig. 6: AMAP Prototype

6. Conclusion

We reported on the current state of computational manuscript research (CMR) and the inherent black box problem that particularly results in low acceptance of computational tools in scientific settings within scholarly manuscript research in the Humanities. We proposed to alleviate the black box problem by dismantling the computational black box(es) into smaller, thus transparent and tractable, elements through Visual Programming in order to keep the-user-in-the-loop-and-control –
either during experiment-based configuration of processing chains for specific tasks or collaboration-driven design of workflows for solving scholarly problems of manuscript research in the Humanities. The advantages of VP and its various features, that allow for the effective dismantling of the black box problem, was elaborated in detail. Moreover, we embedded our design and realization approach in the broader context of Software Development Methodology inspired also by Design Thinking and Human Computer Interaction/Communication. We finally introduced our iXMan_Lab concept and AMAP as its tool instance, being a web-based prototype tool that attempts to deconstruct the black box by offering potential users from SFB sub-projects various DIA methods in a supportive VP environment. Our adoption of at least some of the principles of the VP paradigm to deconstruct the computational black box is the first step in the long journey ahead to completely eliminate it for the sake of truly inter-/transdisciplinary, effective, and efficient computational manuscript research.

Acknowledgments

This work was made possible due to the support from the Deutsche Forschungsgemeinschaft (DFG). 3M sponsored the necessary hardware for the Multi-Touch Table (MTT). We thank the Z03 team for providing the necessary support to our research. Andreas Fischer and Marcel Würsch from the University from Fribourg helped integrating DIVA services into AMAP. The various collaborating SFB sub-projects also are to be mentioned here for providing the necessary data and sharing their domain knowledge. Last but not the least that need to be mentioned are our various bachelors and masters students that have contributed to the iXMan_Lab.

References

1. Blackwell, A. F. (1996). Metacognitive Theories of Visual Programming: What Do We Think We are Doing?. In Visual Languages, 1996. Proceedings., IEEE Symposium on (pp. 240-246). IEEE.

2. Burnett, M. (2002). Software Engineering for Visual Programming Languages. In Handbook of Software Engineering and Knowledge Engineering: Volume II: Emerging Technologies (pp. 77-92).
3. Fraser, N. (2015). Ten Things We've Learned from Blockly. In Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE (pp. 49-50). IEEE.
4. Hassner, T., Rehbein, M., Stokes, P. A., & Wolf, L. (2015). Computation and Palaeography: Potentials and limits. Kodikologie und Paläographie im Digitalen Zeitalter 3: Codicology and Palaeography in the Digital Age 3, 1.
5. Hassner, T., Sablatnig, R., Stutzmann, D., & Tarte, S. (2014). Digital Palaeography: New Machines and Old Texts (Dagstuhl Seminar 14302). In Dagstuhl Reports (Vol. 4, No. 7). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
6. Kasturi, Rangachar, Lawrence O'gorman, and Venu Govindaraju. Document image analysis: A primer. Sadhana 27.1 (2002): 3-22.
7. Mohammed, H., Märgner, V., Konidaris, T., & Stiehl, H. S. (2017). Normalised Local Naïve Bayes Nearest-Neighbour Classifier for Offline Writer Identification. In Document Analysis and Recognition (ICDAR), 14th IAPR International Conference on (Vol. 1, pp. 1013-1018). IEEE.
8. Myers, B. A. (1990). Taxonomies of Visual Programming and Program Visualization. Journal of Visual Languages & Computing, 1(1), 97-123.
9. Rajan, V. & Stiehl, H. S. (2018a). Bringing Paleography to the Table: Developing an Interactive Manuscript Exploration System for Large Multi-Touch Devices. In Document Analysis Systems (DAS), 13th IAPR International Workshop.
10. Rajan, V., & Stiehl, H. S. (2018b). From Eye-to-Eye to Hand-in-Hand: Collaborative Solution Building in Interdisciplinary Manuscript Research. INF-DH-2018.
11. Smith, R. (2007, September). An Overview of the Tesseract OCR Engine. In Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on (Vol. 2, pp. 629-633). IEEE.
12. Stokes, P. (2012). Modeling Medieval Handwriting: A New Approach to Digital Palaeography. DH2012 Book of Abstracts, ed. by Jan Christoph Meister et al, 382-385.
13. Tanimoto, S. L. (1990). VIVA: A Visual Language for Image Processing. Journal of Visual Languages & Computing, 1(2), 127-139.
14. Wolber, D. (2011). App Inventor and Real-World Motivation. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 601-606). ACM.
15. Würsch, M., Ingold, R., & Liwicki, M. (2016). DIVAServices—A RESTful Web Service for Document Image Analysis Methods. Digital Scholarship in the Humanities, 32(1), i150-i156.