A Literature Review On Algorithm Visualizers

Sweeta Bansal[1], Karan Kohli[2], Krishna Kumar Vishwakarma[3], Kush Gupta[4]

[1] *Computer Science and Engineering, Inderprastha Engineering College, Uttar Pradesh, India*

Abstract

Algorithms and data structures as an integral part of the knowledge in computer science framework and every software developer are expected to have basic knowledge from the environment. The introduction of appropriate methods into the learning algorithms is required. Algorithm visibility shows how algorithms work in a graphic way. It aims to simplify and deepen the understanding of the operation of the algorithms. By clearly representing functional computer algorithms, algorithm visualization (AV) technology aims to help computer science students understand how algorithms work.

INTRODUCTION

One of the most common problems in computer science education is the way to explain and communicate non-material terms and conditions. As a result the complexity increases when the concept is not visible, as it is fundamental in computer science, such as algorithms. Various methods of support for teaching algorithms have been proposed, including the use of graphical and oral presentations of algorithms. The beautiful AVs make the algorithms come to life by clearly representing their various regions and highlighting the transformation between those regions. They display data structures in natural, invisible ways instead of focusing on memory addresses and activity calls.

AVs are naturally appealing to teachers, almost worldwide who look good and are consistently “liked” by students. They can be used to attract students' attention during lectures, explain ideas in concrete terms, promote a practical learning process, and facilitate better communication between students and teachers. Interactive algorithm viewing allows students to evaluate and evaluate ideas regarding their individual needs.

The purpose of this paper is to review the Visualization Algorithm textbooks, highlighting the work done so far in this area and the scope of development. We present the findings of the field of Visualization Algorithm (AV) based on our analysis. We regard the recognition of the algorithm as a priority and a point of view for further research and the application of its results in computer science education today.

LITERATURE SURVEY

A number of AV development programs are already well known in the CS education community. Collaborative visualization has been used in computer science education since the 1980s.

AV software has been used:

- help teachers demonstrate the effectiveness of algorithm in a lecture;
- assist learners as they learn and learn basic algorithms in computer science;

- to help instructors track down bugs in students linked-list programs during office hours and
- help learners learn about the basic functionality of non-computer-generated data
- science laboratory.

Modern ways of seeing software were introduced in the 1980's with the introduction of the BALSA program (Brown & Sedgewick, Brown University, USA). Since then, hundreds of algorithm images have been used and freely distributed to teachers, and points (or large) of paper have been written on them. It is widely thought that algorithm recognition can provide a more powerful alternative than standing document presentations.

Since the introduction of Java in the mid-1990s, almost all of the algorithm displays and algorithm detection tools have been used in Java. About half of the available views are provided as applets directly on web pages. However, only a handful of Java applications should be downloaded and opened locally.

These numbers are somewhat biased. There is a tendency for us to look for applets, as this turns out to be easy to do. Visuals found directly on web pages will often receive more attention from potential users, as they do not have to go through the extra step of downloading and installing the visual or display program.

Some of contemporary solutions include systems like TRAKLA23, ANIMAL4, JAWAA5 or Algorithms In Action :

- Brown University Algorithm Simulator and Animator (BALSA) BALSA [Brown et al., 1985] was one of the first algorithm simulations designed to help students understand computer algorithms. The program has served as an example of the animation of many algorithms that were later developed [Wiggins, 1998]. It was developed in the early 1980s at Brown University to achieve a number of goals. Students could use it to look at the performance of algorithms and thus gain a better understanding of their performance. Students do not have to write code but ask for someone else's code. The teachers would use it to prepare the material for the students. The algorithm designers will use the resources provided by the system to obtain a dynamic image display of their operating systems to fully understand the algorithms. Cartoons using state-of-the-art resources provided by BALSA will design and implement programs that will be featured in production. The program was written in C and the PASCAL programs were animated. BALSA has provided users with a few services. Users can control the display features of the system and thus be able to create, delete, resize, move, zoom in, and filter algorithm views. Users are given different views of the algorithm at the same time. The system allowed several algorithms to be used and displayed simultaneously. Users are also able to interact with the animation of the algorithm. For example they can start, stop, slow down, and run the algorithm backwards. After the algorithm worked once, the entire history of the algorithm was saved so that readers could refer to it and restart it. Users can save their window settings and use it to restore algorithm views later. The original version of the system introduced black and white animation.

- Tango and XTango The animation system of the Xtango algorithm was developed under the direction of Drs. John Stasko at Georgia Tech as a follower of the animation system of the Tango algorithm. XTango "supports color enhancement, real-time, 2 & 1/2 dimensional, smooth graphics of algorithms and programs" [Wiggins, 1998]. The system uses the transformation paradigm to achieve smooth animation [Stasko, 1992]. XTango is used in the X11 window system. It is distributed through sample animation programs (e.g., matrix replication, Fast Fourier Transform, Bubble Filtering, Two Stacks, AVL Trees) [Wilson et al., 1996]. The system is designed for easy use. It is intended to be useful for those who are not computer-assisted in the use of motion pictures [Wiggins, 1998]. The package can be used in two ways: Users can embed data structures and library animation packages in C-format or any other programming language that can generate a tracking file. The embedded dial system is then integrated into the Xtango and X11 window libraries. Another way to use animation package is to write a command text file read by a downloadable animated system translator along with the Xtango package. A text file can be created with a text editor or as a result of printed statements in a user simulation program.

- Generalized Algorithm Illustration through Graphical Software (GAIGS) GAIGS Algorithm Visualization was established at Lawrence University from 1988-1990. The system does not actually create an algorithm but generates summaries, while the algorithm signs, of data structures in "interesting events" [Naps et al., 1994]. Users can then view these summaries at their own pace. GAIGS introduces a different simultaneous

view of the same data structure. It also allows users to go back to the previous state and replay the sequence of the algorithm abbreviations previously introduced. However, it provides users with a standard version of the program code. Users cannot detect how a particular line of code has an effect on system performance [Wiggins, 1998]. GAIGS provides an intuitive understanding of algorithms for testing without editing them. The computer science department of more than 60 institutions has used the previous version of this program in key subjects such as Introduction to Computer Science, Software Design Principles, Systems Analysis and Design, as well as Data Structures courses and analytical algorithm.

- Dynamic Laboratory (DynaLab) DynaLab was established in the early 1990s at Montana State University. The program is designed to "open to students a broader definition of algorithms, a world of interesting problems, problems solved by algorithms, unresolved problems, unresolved problems and what practical solutions to the relevant problems from the set of solutions provided" and supports "interactive, visual and inspiring and demonstrative demonstrations. laboratory in computer science "[Boroni et al., 1996]. It had X-windows and MS-windows program animators as well as a complete animation program library. he can be move the program to the library and display it in the animator. The animation series consists of highlighting a portion of the system in use, displaying different converted values and displaying input in the system. Cartoons also store and display the total cost of making the program. To understand the confusing or complex sequence of events in the algorithm, users can undo animation with a certain number of steps and restart it in the usual way forward. DynaLab was developed for use in the field of laboratory settings where students could perform various tests with a given algorithm (e.g., to detect the time complexity of the algorithm). Such a test setting can be easily accomplished because each student can have a copy of DynaLab and a specific algorithm with which to work. No more time is needed to set up these laboratory tests and that is why we can always focus on doing them.

- SWAN [Shaffer et al., 1996] was created at Virginia Tech, and can be used to visualize data structures used in the C / C ++ system. All data structures in the system are considered graphs. The graph can be oriented or unrestricted and can be limited to special situations such as trees, columns and columns. The program has three key components: Swan Annotation Interface Library (SAIL), Swan Kernel, and Swan Viewer Interface (SVI). SAIL contains library functions that allow users to create different system views. SVI allows users to test the Swan annotation system. The Swan Kernel is the main module of the system.

- Java And Web-based Algorithm Animation (JAWAA) JAWAA [Pierson and Rodger, 1998] uses simple command language to create animations of data structures and display them through a web browser. The system is built on JAVA so it can be used on any machine. Animation to be created is written in simple script language. A text file can be easily transcribed into any text editor or can be generated as an output. Texts have one command or image function per line. The JAWAA applet returns the command file and executes it. The system interprets commands by line and performs specific functions for each command. JAWAA provides commands that allow users to create and move both archeological objects such as circles, rows, text, rectangles, etc., and data structure objects such as columns, stacks, rows, columns, trees, graphs etc. The interface of the system contains animation. canvas and panel that gives users controls such as start, stop, pause and step animation. To control the animation speed the scroll bar is also provided.

- Flexible Learning with Artificial Intelligence Repository (FLAIR) FLAIR [Ingargiola et al., 1994] is a repository of algorithms and teaching materials to be used in Artificial Intelligence courses. Students can learn from these resources, which are laboratory-based learning areas, by experimenting with them. For example, students can use the Search Module to streamline common search algorithms on a city map. Students can choose from a set of algorithms a specific algorithm for survival, various heuristics to implement the algorithm, and start and end cities. They can also adjust animation speed, create new city maps to launch the algorithm, skip through animation or pause. Along with the animation of the search algorithm students can resume detailed animation of basic data structures. To compare the performance of different search algorithms, students can have multiple windows open at the same time, each animating a different algorithm that works on the same problem at the same time. Teachers can use these resources to guide students to work on different algorithms, to analyze how different algorithms work in different contexts, and under what circumstances the given algorithm works better than other algorithms that solve

the same problem. The system operates on SUN SPAR operating stations. Recorded on Common Lisp (CL) and Common List Object System. The graphical user interface of the system was developed using Garnet System (Garnet was developed at Carnegie Mellon University using X-Windows and has a Lisp based GUI). The program can be installed on any hardware / software that supports CL running Garnet.

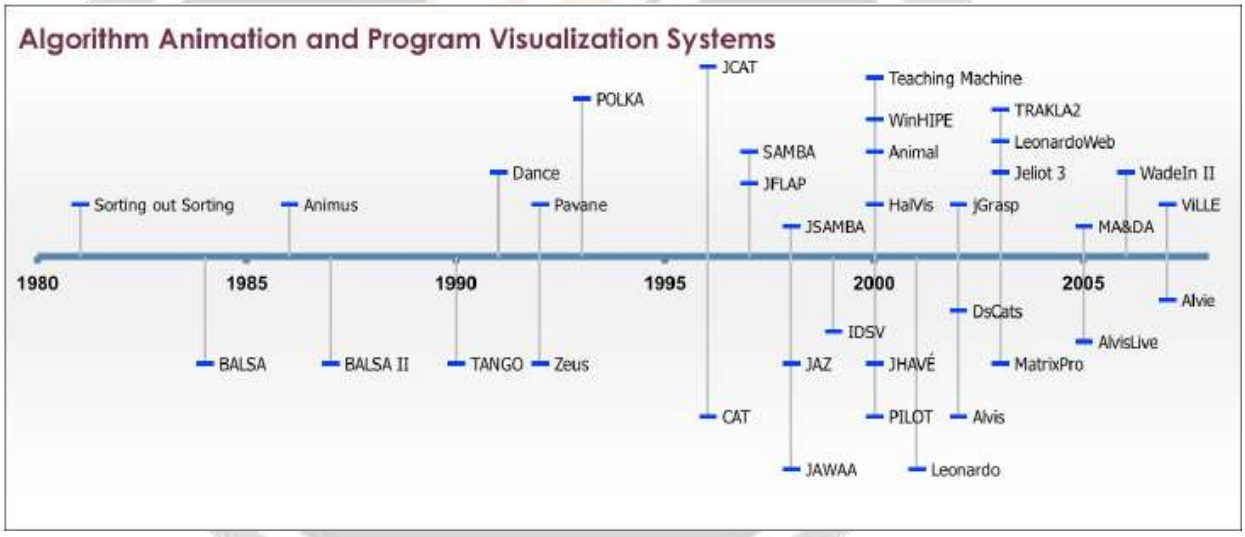
- A low-key animation focused on the corresponding system (POLKA) The animation system of the POLKA algorithm [Stasko et al., 1993] was created at Georgia Tech, under the direction of Drs. John Stasko. The program not only allows students to watch the animation of a pre-created algorithm but also allows them to create their own animation. There are two versions of POLKA: a 2D version built on the X window System and a 3D version of the Silicon Graphics GL system. Both versions of the system have two important features. They contain ancient texts that provide "true" images [Stasko et al., June 1993], a skill not found in some systems. These primitives show smooth, continuous movements and actions on an object and not just flashing objects or color changes. These high-quality photographic skills should preserve the context and promote comprehension. The first elements that show the various acts of animation make Polka particularly useful in accurately depicting the same functionality that occurs in the same system. The 3D version offers many automatic and easy parameters so that designers do not have to worry about image details. Editors do not need to know 3D image techniques such as blurring, ray tracking, etc., in order to create 3D visuals. To record animation in Polka, you need to create a C ++ program and inherit the basic classes offered by the system. The authors say the program is easy to use and report that students who are unfamiliar with C ++ have been successful in using it.

- Samba is the animation system application of the POLKA algorithm [Stasko et al., June 1993] described above. Designed for students to be able to write algorithm images as part of their classroom assignments. Samba is designed to be easy to use and read so that students can create their own pictures. Students who write animation will be tied to the algorithm and its functionality. Therefore, as they build the animation of their algorithm, students should discover the basic features and characteristics of the algorithm. Samba is a cartoon translator and generator that can work in bulk mode. It assumes as follows a sequence of ASCII commands, one command per line. There are different types of commands given to the system. One set of commands generates animation images. The second set of commands fixes things that are already in production. There are other types of commands that can be used to create complex and complex images. For example there are commands that can be used to have multiple views (windows) of the algorithm. The command group can be grouped together to run simultaneously. Samba uses the X-Window System and Motif. The program was developed for Windows and Java in 1997 [Stasko, 1997].

- Mocha [Baker et al., 1996] was developed to provide animation algorithm on the World Wide Web. It has a distributed model with a client server configuration that separates the software components of the animation system of a common algorithm. Two important features of Mocha is that users with limited computer power can access the animation of complex algorithms, and it provides code protection to what end users do not have access to algorithm scripting. Animation is considered an event-driven program of communication processes. The algorithm contains annotations of interesting events called algorithm functions. There is an animated section that provides multimedia visualization of algorithm functions. The animation component is further divided into a GUI, which handles user interaction, as well as an animator, which displays algorithm performance and user requests into flexible multimedia scenes. Users interact with the system using the HTML interface, which is transmitted to the user's machine and the interface interface. The security of the animation code is only achieved by exporting the visual interface code to the user's machine. The algorithm is applied to a server running on the provider's machine. Multithreading on GUI usage and animator is used to provide responsive feedback to users. Also, an object / component-based software structure has been used to ensure system flexibility. Mediators are used to distinguish similarities between client interactions with servers, which provides a high level of interaction.

- Hpermedia Algorithm Visualization System (HalVis) HalVis was developed at Auburn University in the late 1990s. The program is designed with the idea that, in order to make the algorithm visualization more academically effective, in addition to gaining visual attention (made by multiple algorithm visions), you also need to get mental attention and engage the student's mind while viewing algorithm visibility. . The

program displays algorithms in the multimedia area. HalVis has five modules. The 'Basic' module contains information about basic functionality (e.g. data exchange, looping performance, recursion) that is common to almost all algorithms. This module cannot be directly accessed. Requested for hyperlinks from other modules. The module introduces contextual information into a student algorithm. The 'Vision Perspective' module defines a specific algorithm for students in terms of commonly used metaphors. For example, defining a Mergesort algorithm uses a simulation of dividing playing cards and combining them to create a sequential sequence. Animation, text, and interaction are used to explain important algorithm features to students. The 'Detailed View' module defines the algorithm in a more detailed way. Two presentations are used for this. One presentation contains a textual description of the algorithm and a false code. The text description contains 'basic' module links. The second presentation consists of four windows: Animation Output window displays data updates as a result of algorithm creation using smooth animations, Activity Message window describes important events and algorithm actions using text and comment feedback, Pseudocode window displays. algorithm steps for self-expression and animation, the Performance Variables window displays a "point board similar to the panorama of the variables involved in the algorithm" [Hansen et al., 2002]. Initially, only a limited number of items can be included. This allows students to focus on low-level algorithm behavior. The 'Full View' module captures large data sets as incorporating to make high-level behavior algorithm clear to readers. The animations embedded in this concept are similar to those found in previous editions.



CONCLUSION

Based on our findings, algorithm recognition can be seen as a useful supportive tool, used in addition to conventional educational methods in the field of computer science. Algorithms are an exciting way to be used in viewing. To visualize the algorithm, we simply do not enter the data into the chart; no primary database. Instead, there are sound rules about behavior. This may be the reason why algorithm recognition is rare, as designers try novel forms to better communicate. This is a good reason to read them. But algorithms are also a reminder that visualization is more than just a tool for finding patterns in data. Visualization enhances one's visual system to enhance one's intelligence: we can use it to better understand these important invisible processes, and perhaps other things.

REFERENCES

1. 4[3] Ahmad Affandi Supli, Norshuhada Shiratuddin and Syamsul Bahrin Zaibon, "Critical Analysis on Algorithm Visualization Study", International Journal of Computer Applications (0975 – 8887), Volume 150 – No.11, Sep 2016.

2. [3] Christopher D.Hundhausen, Sarah A.Douglas And John T.Stasko, " A Meta-Study of Algorithm Visualization Effectiveness",Journal of Visual Languages & Computing,Volume 13, Issue 3, pp. 259-290, Jun 2002.

3. [1] C. Shaffer. A Practical Introduction to Data Structures and Algorithm Analysis. Prentice Hall, second edition, 2001.

4. [1] P. Saraiya. Effective features of algorithm visualizations. Master's thesis, Department of Computer Science, Virginia Tech, July 2002.

5. [2] K. Mehlhorn, P. Sanders, Algorithms and Data Structures (Springer-Verlag, Berlin Heidelberg, 2008)

6. [1] J. Genči, Possibilities to Solve Some of the Slovak Higher Education Problems Using Information Technologies, In proceedings of: 10th IEEE International Conference on Emerging eLearning Technologies and Applications, ICETA2012, Stará Lesná, The High Tatras, Slovakia, November 8-9, 2012

7. [1] S. Khuri, Designing Effective Algorithm Visualizations, In proceedings of: First International Program Visualization Workshop, ITiCSE, Porvoo, Finland, July 7 - 8, 2000, Available: http://www.cs.sjsu.edu/~khuri/invited.html

8. [3] C.D. Hundhausen, S.A. Douglas, J.T. Stasko, A Meta-Study of Algorithm Visualization Effectiveness, J. Visual Lang.Comput. 13, 259–290, 2002

9. [3] V. Lazaridis, N. Samaras, A. Sifaleras, An empirical study on factors influencing the effectiveness of algorithm visualization, Comput. Appl. Eng. Educ. 21, 410–420, 2013

10. [3] D.J. Jarc, M.B. Feldman, R.S. Heller, Assessing the benefits of interactive prediction using Web-based algorithm animation courseware, Proceedings of SIGCSE 2000 (ACM Press, New York, 2000

11. [1] S. Diehl, Software visualization: Visualizing the Structure, Behaviour, and Evolution of Software (Springer, New York, 2007) 187

12. [4] M.E. Tudoreanu, R. Wu, A. Hamilton-Taylor, E. Kraemer, Empirical Evidence that Algorithm Animation PromotesUnderstanding of Distributed Algorithms, In proceedings of: IEEE Symposium on Human Centric Computing Lan-guages and Environments, HCC02, Arlington, Virginia, September 2002

13. [3] B.A. Price, R.M. Baecker, I.S. Small, A Principled Taxonomy of Software Visualization, J. Visual Lang. Comput. 4(3),211–266, 1993.

14. [1] Ž. Šuchová, Visualization of Algorithms and Data Structures, Bachelor thesis, DCI FEEI TU of Košice, Bachelorthesis, 2010 (in Slovak)

15. [1] S. Diehl (Ed.), Software Visualization, Lecture Notes in Computer Science 2269, 2002

16. [2] M.H. Brown, R. Sedgewick, A system for algorithm animation, Proceedings of the 11th annual conference on Computer graphics and interactive techniques, SIGGRAPH'84 (ACM New York, NY, USA, 1984)

17. [2] G. Rößling, B. Freisleben, ANIMAL: A system for supporting multiple roles in algorithm animation, J. Visual Lang.Comput. 13(3), 341–354, 2002

18. [1] S. Šimoňák, Algorithm Visualization Using the VizAlgo Platform, Acta Electrotechnica et Informatica 13(2), 54–64,2013

19. [1] A. Sajko, Algorithm Visualization, Bachelor thesis, DCI FEEI TU of Košice, 2012 (in Slovak)