Embedding Topical Elements of Parallel Programming, Computer Graphics, and Artificial Intelligence across the Undergraduate CS Required Courses

http://dx.doi.org/10.3991/ijep.v5i1.4090

J. Wolfer
Indiana University South Bend, South Bend, USA

Abstract—**Traditionally, topics such as parallel computing, computer graphics, and artificial intelligence have been taught as stand-alone courses in the computing curriculum. Often these are elective courses, limiting the material to the subset of students choosing to take the course. Recently there has been movement to distribute topics across the curriculum in order to ensure that all graduates have been exposed to concepts such as parallel computing. Previous work described an attempt to systematically weave a tapestry of topics into the undergraduate computing curriculum. This paper reviews that work and expands it with representative examples of assignments, demonstrations, and results as well as describing how the tools and examples deployed for these classes have a residual effect on classes such as Comptuer Literacy.**

Index Terms— **Pedagogy, Artificial Intelligence, Computer Graphics, Parallel Computing, Instructional approaches.**

I. INTRODUCTION

Distributing a variety of traditionally elective topics across the required CS curriculum is both pedagogically sound and fits the spirit of the new ACM/IEEE Computer Science Curricula 2013 guidelines [1]. Earlier work [2] profiled efforts to weave concepts from three of these elective courses, Computer Graphics, Parallel Processing, and Artificial Intelligence through a series of required undergraduate courses, giving students early and sustained exposure to vocabulary and concepts. This work expands that initial attempt to systematically expose students to these concepts by providing specific assignments, examples, and demonstrations provided to the students. In addition, observations on how the infrastructure developed to support this approach can provide residual benefit to other classes, such as Computer Literacy are also provided.

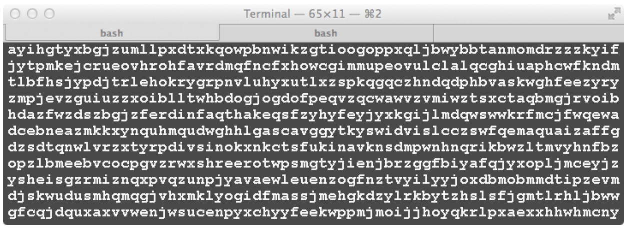
In addition to the ACM/IEEE Guidelines, other researchers have looked at topical distribution in the curriculum. For example, Sheldon and Turbak [3] describe distributing aspects of computing topics ranging from computational theory to artificial intelligence. They also discuss challenges and tradeoffs involved such as identifying prerequisites and the necessity for faculty collaboration.

Given the NSF/IEEE-TCPP Parallel and Distributed Computing [4] guidelines, the move to distribute parallel computing concepts may be the most mature at this time. For example, Minaie and Sanati-Mehrizy [5] review a variety of programs, both domestic and international, in terms of their parallel computing curriculum. They report that in China there are two primary approaches, either a dedicated parallel processing course, or integrating parallel concepts into the Computer Organization, Architecture, Operating Systems, and/or Embedded Systems courses among others. Their review of seventeen universities in the United States identified four integration models: an independent undergraduate course, an independent graduate course, concept integration into existing courses, or a combination of the three previous approaches. In all cases the trend is to move topics out of the elective-only curriculum.

Finally, Danner and Newhall [6] describe a systematic approach to ensuring that all their CS students are exposed to “parallel thinking” during the course of their studies. The approach includes a required introductory course that includes parallel concepts in anticipation of future coursework. The balance of the parallel topics in their program are distributed among various more advanced courses such as Compilers, Operating Systems, and Computer Graphics, among others.

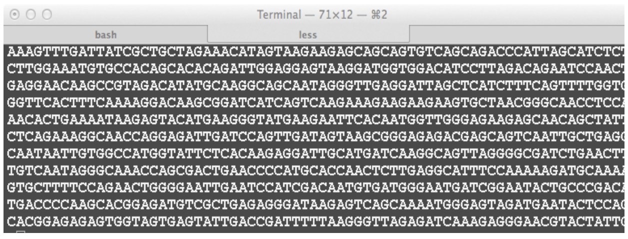
While all of these approaches seem to be effective in the distribution of parallel concepts across the curriculum, all except [3] ignore the potential of distributing topics from other specialties. This work describes one effort to identify three specific courses and to systematically integrate concepts from each of them into a subset of classes that all CS students must take. This serves two purposes. First it exposes the students to the topics, even if only in passing for some cases. Second, it serves as a “sampler” to give the students a taste of each topic to help inform their future course selection. The courses and topics included in this effort are Computer Graphics, Parallel Computing, and Artificial Intelligence.

In the Computer Science program at our institution the Computer Graphics, Parallel Computing, and Artificial Intelligence classes are entirely elective, upper-level computing classes. While this means that students in the classes are there by choice, it also implies that many students will not have encountered some of these topics prior to graduation. Furthermore, those who do take these classes often arrive with no significant concept of the constituents of the respective class. This approach to the problem is to, as seamlessly as possible, introduce concepts from Computer Graphics, Parallel Computing, and Artificial Intelligence early in a student’s program, then allowing the concepts to re-emerge in subsequent courses.



```
ayihgtyxbgjzumllpxdtxkqowpbnwikzgtioogoppxqljbwybbtanmomdrzzzkyif
jytpmkejcrueovhrohfavrdmqfncfxhowcgimmupeovulclalqcghiuaphcwfkndm
tlbfhsjypdjtrlehokrygrpnvluhyxutlxzspkqgqczhndqdphbvaskwghfeezyry
zmpjevzguiuzzxoiblltwhbdogjogdofpeqvzqcwawvzvmiwztsxctaqbmgjrvoib
hdazfwzdszbgjzferdinfaqthakeqsfzyhyfeyjyxkgijlmdqwswwkrfmcjfwqewa
dcebneazmkkxynquhmqudwghhlgascavggytkyswidvislcczswfqemaquaizaffg
dzsdtqnwlvrzxtyrpdivsinokxnkctsfukinavknsdmpwnhnqrikbwzltmvyhnfbz
opzlbmeebvcocpgvzrwxshreerotwpsmgtyjienjbrzggfbiyafqjyxopljmceyjz
ysheisgzrmiznqxpvqzunpjyavaewleuenzogfnztvyilyyjoxdbmobmmdtipzevm
djskwudusmhqmqgjvhxmklyogidfmassjmehgkdzylrkbytzhslsfjgmtlrhljbww
gfcqjdquxaxvvwenjwsucenpyxchyyfeekwppmjmoijjhoyqkrlpxaexxhhwhmcny
```

Figure 1. CS1: Random Text Generation



```
AAAGTTTGATTATCGCTGCTAGAAACATAGTAAGAAGAGCAGCAGTGTCAGCAGACCCATTAGCATCTC
CTTGGAAATGTGCCACAGCACACAGATTGGAGGAGTAAGGATGGTGGACATCCTTAGACAGAATCCAAC
GAGGAACAAGCCGTAGACATATGCAAGGCAGCAATAGGGTTGAGGATTAGCTCATCTTTCAGTTTTGGT
GGTTCACTTTCAAAAGGACAAGCGGATCATCAGTCAAGAAAGAAGAAGAAGTGCTAACGGGCAACCTCC
AACACTGAAAATAAGAGTACATGAAGGGTATGAAGAATTCACAATGGTTGGGAGAAGAGCAACAGCTAT
CTCAGAAAGGCAACCAGGAGATTGATCCAGTTGATAGTAAGCGGGAGAGACGAGCAGTCAATTGCTGAG
CAATAATTGTGGCCATGGTATTCTCACAAGAGGATTGCATGATCAAGGCAGTTAGGGGCGATCTGAACT
TGTCAATAGGGCAAACCAGCGACTGAACCCCATGCACCAACTCTTGAGGCATTTCCAAAAAGATGCAAA
GTGCTTTTCCAGAACTGGGGAATTGAATCCATCGACAATGTGATGGGAATGATCGGAATACTGCCCGAC
TGACCCCAAGCACGGAGATGTCGCTGAGAGGGATAAGAGTCAGCAAAATGGGAGTAGATGAATACTCCA
CACGGAGAGAGTGGTAGTGAGTATTGACCGATTTTTAAGGGTTAGAGATCAAAGAGGGAACGTACTATT
```

Figure 2. H1N1 DNA Sequence Samples

II. FOUNDATION COURSES

From a pedagogical perspective being exposed to concepts multiple times, and in multiple contexts, is an important part of information transfer [7,8]. Additional benefits include potential interest in the elective classes as well potential recruits for undergraduate research programs.

That having been said, when classes are structured topically, there are constraints to weaving external topics into the class. Specifically, the following principles are adopted:

- Any embedding must not be gratuitous. There must be a natural fit to the material being covered in the host class.
- Any topical embedding must not take excessive time to explain and illustrate.
- Any topical embedding should provide topical insight for future elective classes.

With these principles in mind, the balance of this work provides examples of this embedding in both early and advanced CS courses.

III. CS1

The first opportunity for introducing concepts from the three classes featured here is CS1, the introductory programming class. This class is taught using C++ for the programming language. While prior exposure to programming is encouraged, it is not required for admission to the course. Therefore this class must start with an overview of basic computer concepts and proceed to develop algorithms and programs at a beginning level. Classroom examples and corresponding assignments serve to subtly, and seamlessly, weave a pre-AI thread into the course.

One example is a simple simulation of Eddington's Monkeys [9], based on the proposition that monkeys typing randomly will eventually create all the works of Shakespeare. For the CS1 student this introduces the concept of characters, their underlying numeric encoding, the modulus operator (to constrain them to the alphabet), and random number generators. It also introduces the concept of simulation using the computer. Using these tools the students create a randomly typing "monkey" that generates text. Sample output is shown in Figure 1.

From a "weaving AI" perspective it allows mention of Natural Language Processing and Computational Linguistics.

Later the skills for dealing with text are extended to elementary analysis in the context of learning array handling. Skills such as indexing are acquired within a familiar context, having already worked with character simulation. For example, in one assignment students evaluate the number of occurrences of each character in Tolstoy's War and Peace [10], then calculate the space to total-characters ratio. This, in turn, invites a short discussion of other applications of more sophisticated Natural Language Processing (NLP) and textual analysis such as the IBM Watson project [11], thus putting their work in a broader AI context.

Other assignments expand on this early text processing introduction. Students write code to count the number of each nucleotide in the DNA sequence for, recently, the H1N1 virus, using an integer array to contain the respective counts. Figure 2 shows a sample of the DNA sequence for a New York strain of the H1N1 virus. A representative assignment associated with this dataset might be:

For this assignment you are to create an array large enough to hold an entire DNA sequence for H1N1. To be safe, let's make it much larger than we need, say 20,000 characters in length.

Using the fstream library, read a DNA sequence into the array, making the sequence into a proper cstring.

Scan the string to find:

1. *The number of occurrences of each letter in the DNA string by using the letter as an index into an integer array to accumulate totals. Print the number of instances of each letter.*
2. *The longest substring of repeated characters. Print which character it is, the length of the repeated substring, and where in the string the repeated sequence begins.*

Approaching the assignment in this manner allows for a short description of Bioinformatics, Artificial Intelligence, and Pattern Recognition without serious impact on time or the core objectives for the class.

CS1 also introduces computer graphics, lightly, in the context of learning input-output manipulation and formatting. Figure 3 shows the output of one cycle of a sine wave plotted using the '*' character on the screen. This serves to introduce the graphics concepts of translation and scale (to fit one cycle on the screen) as well as dynamic output manipulation.

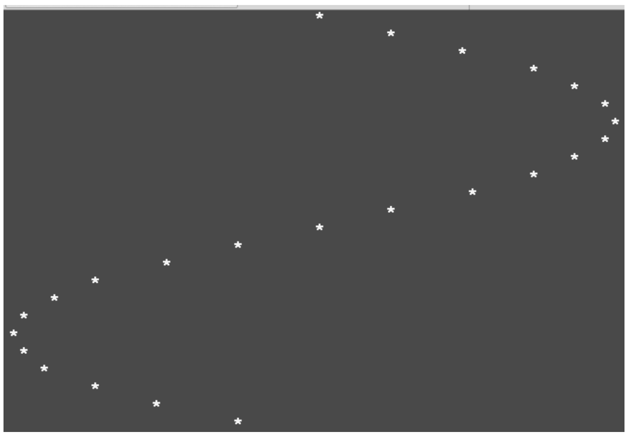


Figure 3. CS1: First "graphics" encounter, character-plot sine wave

IV. CS2

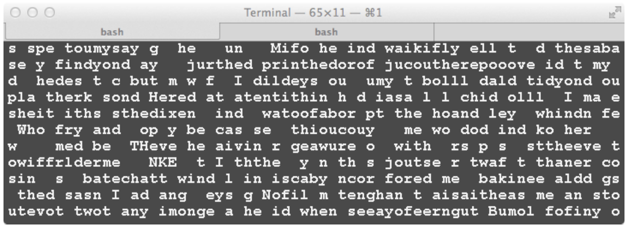
The CS2 course extends both the student's knowledge and experience with programming and offers incrementally more latitude for interjecting relevant samples from AI, Graphics, and Parallel Computing. Currently CS2 both expands syntactic knowledge and introduces elementary data structures. For example, multi-dimensional arrays and linked lists are introduced. Other topics include the use of pointers, C++ structs and classes, function and operator overloading, templates, as well as best practice topics such as conditional compilation, error trapping, and separate compilation of program components.

Since students are gaining maturity, the topics from CS1 are expanded in CS2. For example, the "Eddington's Monkey" program is expanded to create bigram and trigram correlation matrices from large bodies of English (in our case), which are then used to inform the "monkey" simulator – increasing the probability that they will produce letter sequences consistent with the language. Sample output is shown in Figure 4. Note that some English words, such as "with", "who", and "me" appear. Spaces delimit character strings at typical word-like intervals, and many character sequences are pronounceable, if not sensible. This assignment makes an excellent segue for discussing additional aspects of NLP, and to introduce tools such as the Google N-gram Viewer [12] as a way of tracking concept and sentiment in the literature over the span of decades as shown in Figure 5.

Computer Graphics illustrations are also a natural fit in a discussion of multidimensional arrays as well as those for dynamic memory allocation. During the introduction of multi-dimensional arrays the concept of a gray-scale pixel is introduced as a character element in a 2-D array. Various C++ functions are developed to populate the array of pixels forming various simple geometric forms. To avoid the complexity of user-interface and graphics library development at this early stage in their experience these arrays are wrapped in appropriate header information and exported in PGM format to be displayed by an independent display program. More complex images, such as the Mona Lisa (Figure 6), illustrate the association of pixel value and image intensity as displayed on the screen. Practice problems using simple image processing, such as local averaging, is sometimes introduced as a means of teaching elementary array manipulation. Finally, images are used as an example data element to be dynamically allocated when developing the concepts and implementation of pointers, dynamic memory allocation, and linked lists.

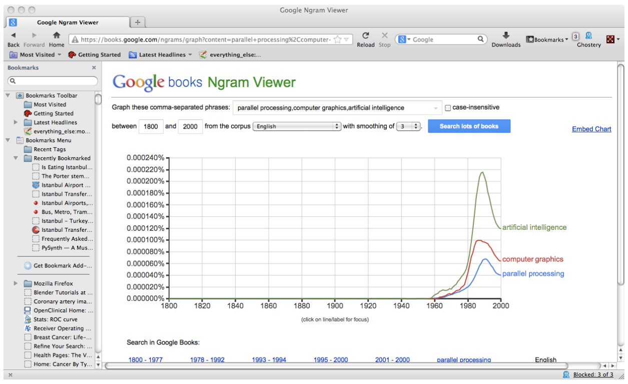
V. COMPUTER ORGANIZATION

While CS1 and CS2 provide opportunities for touching on AI and computer graphics, parallel processing is barely mentioned. It is in the context of Computer Organization that Parallel Processing gets it's undergraduate debut. The Computer Organization course introduces basic computer structures at the logic level, such as full- and half-adders, multiplexors, flip-flops, and decoders. Students are then exposed to a significant assembly language project culminating in the development of a simulated CPU with embedded robot control instructions. The students then write programs using their own CPU to actually control robots as shown in Figure 7. Since the robots are programmed to respond to sensors and react to their environment, it is an



```
s spe toumysay g  he   un   Mifo he ind waikifly ell t  d thesaba
se y findyond ay   jurthed printhedorof jucoutherepooove id t my
d  hedes t c but m w f  I dildeys ou  umy t bolll dald tidyond ou
pla therk sond Hered at atentithin h d iasa l l chid olll  I ma e
sheit iths sthedixen  ind  watoofabor pt the hoand ley  whindn fe
 Who fry and  op y be cas se  thioucouy   me wo dod ind ko her
w    med be  THeve he aivin r geawure o  with  rs p s  sttheeve t
owiffrlderme   NKE  t I ththe  y n th s joutse r twaf t thaner co
sin  s  batechatt wind l in iscaby ncor fored me  bakinee aldd gs
 thed sasn I ad ang  eys g Nofil m tenghan t aisaitheas me an sto
utevot twot any imonge a he id when seeayofeerngut Bumol fofiny o
```

Figure 4. CS1: First "graphics" encounter, character-plot sine wave


Google Ngram Viewer
https://books.google.com/ngrams/graph?content=parallel+processing%2Ccomputer-
Google books Ngram Viewer
Search lots of books
Embed Chart
artificial intelligence
computer graphics
parallel processing
(click on line/label for focus)
Search in Google Books:

Graph these comma-separated phrases: parallel processing,computer graphics,artificial intelligence

between 1800 and 2000 from the corpus English with smoothing of 3

Figure 5. Google N-Gram Viewer

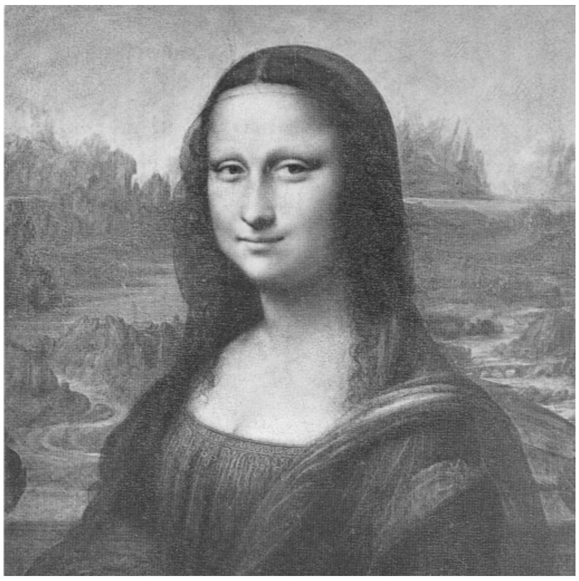


Figure 6. Mona Lisa

excellent introduction to some of the pragmatic aspects of Artificial Intelligence such as navigation, sensor interpretation, and autonomous navigation.

Computer Graphics is introduced to the Computer Organization class indirectly, in the context of haptic interaction. To give students a feel for the limited sensor capability of the robots – that is, to give students a "robot-eye" view of the world – we developed a haptic maze environment for them to explore with their eyes closed. A haptic mouse, the Logitech iFeel mouse [13] (Figure 8), is programmed to give vibration sensations when encountering a boundary in the maze shown in Figure 9. The goal for the student is to traverse the maze blind, using only the contact information – thus simulating the limited perspective of the robots and enhancing their understanding of the constraints under which they must develop their assembly language software.

Computer Organization also offers the first serious introduction to parallel processing. Both in terms of the basic architectural aspects as illustrated with Flynn's Taxonomy, but also an opportunity to introduce contemporary parallel software development tools.

Specifically, to tie the discussion to equipment that which the students may already possess, we include an introduction to OpenMP [14] for shared-memory machines represented by our current multi-core CPUs, and a short introduction to Graphics Processing Units (GPU) illustrated by a survey of both Nvidia's CUDA [15] language and the vendor-neutral OpenCL [16] programming language.

VI. OPERATING SYSTEMS

Concepts from all three areas are touched upon in the Operating Systems course. This class is positioned late in the overall program to act as a captstone course encapsulating much of the knowledge and experience gained during the previous three or so years. The class covers classical operating system concepts such as process management, scheduling, interprocess communication, memory and file management, security, and device handling.

Parallel processing is a natural fit for this class. Topics include multi-core threading, OpenMP, and the use of the GPU at the operating system level. Artificial Intelligence is considered in two specific contexts, scheduling where genetic optimization is discussed and security where invasion detection techniques such as Artificial Immune Systems are introduced. Finally, low-level support for Computer Graphics is introduced in the context of device drivers for interactive devices. Specific attention is made at the device-driver and user library level to haptic devices supporting Computer Graphics such as the Novint Falcon (Figure 11) and the aforementioned Logitech haptic mouse (Figure 8).

VII. AI, GRAPHICS, AND PARALLEL PROGRAMMING COURSES

The courses described in Sections II through VI are particularly important since they represent a sequence that all students in the program must take. This ensures that all of the students get some exposure to the relevant material even if they do not elect to enroll in the AI, Graphics, or Parallel Computing courses. That having been said, to weave a tapestry the individual threads must be entangled to create the whole. This section describes interconnecting the Parallel, Graphics, and AI courses.

The Computer Graphics course is a fairly typical graphics course embedded in a CS program. Beginning with the concept of raster and pixels the class moves quickly to 3D, including volumetric imagery, 3D modeling, transformation, and rendering. The class is sometimes taught with a medical motif, giving the opportunity to develop a model from a series of CT images to a volumetric representation and, finally, a physical model as shown in Figure 10. These models, in turn, are extracted for haptic interaction as shown in Figure 11, where a Novint Falcon [17] haptic robot is poised to explore a model of a carotid artery. The development of these models enables AI related discussions of segmentation and computer vision.

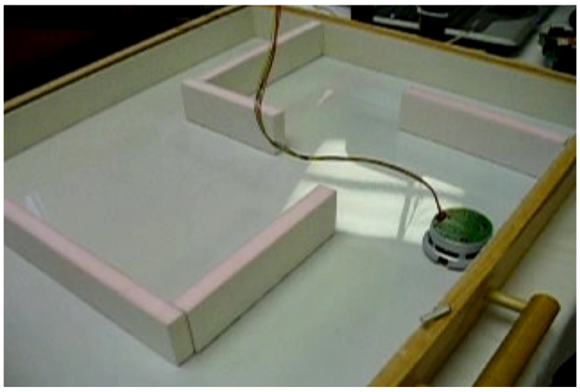


Figure 7. Robot Maze

Figure 8. Logitech IFeel Mouse

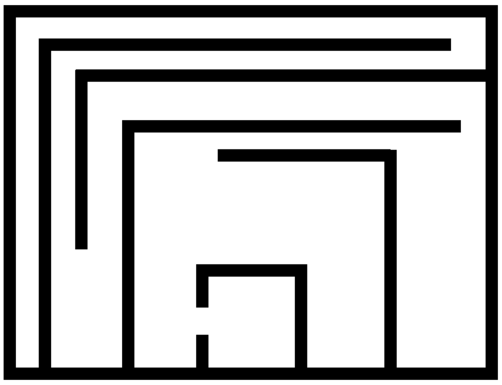


Figure 9. Simulated Robot Maze

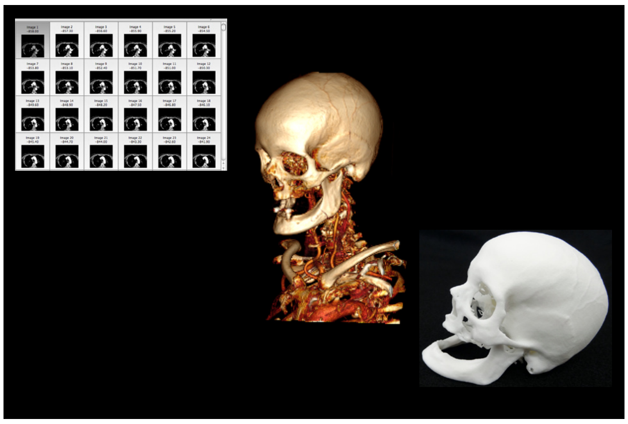


Figure 10. CT sequence, Volume Rendering, Physical Model

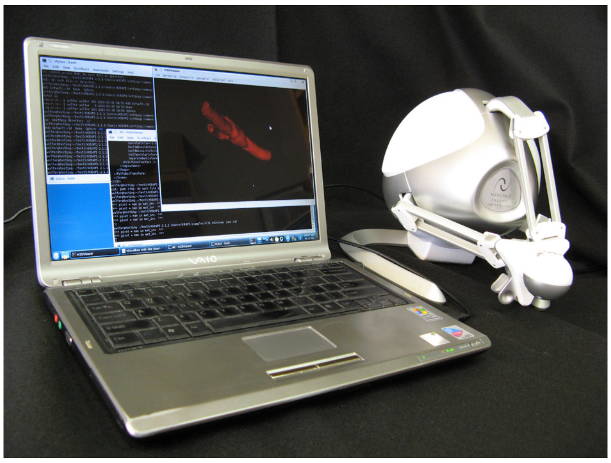


Figure 11. Haptic Aorta Exploration

The course also includes a significant project component, including formal proposals and final presentations. Since the GPU is central to modern transformation and shading, an introduction to parallel concepts is intrinsic to the graphics course. A short diversion into general-purpose GPU programming is included since computer graphics and image processing have a largely intersecting knowledge base.

Likewise, graphics is easy to interject into the parallel computing class. The Parallel Processing course covers a wide range of architectures and algorithms, concentrating on tools, algorithms, and project implementation. Software tools include Message Passing Interface (MPI) [18] for distributed clusters, OpenMP and threading for multi-core CPU's, and Nvidia CUDA and OpenCL for programming the GPU. Graphics illustrations, such as dissecting large images, such as mammograms, and distributing their processing provide a very visual indication of success or failure. A quick glance will indicate whether the resulting image is reconstructed, or scrambled!

AI is introduced into parallel processing in the context of parallelizing AI paradigms such as Pulse-Coupled Neural Networks (PCNN) [19], a biologically inspired model for computer vision and image preprocessing. Figure 12, for example, shows eight frames from an echocardiographic cineloop, or "movie", which have been border-enhanced using the PCNN. Figure 13 shows the speedup plot as additional processors are added to a PCNN implementation processing these images.

Finally, the Artificial Intelligence class offers many opportunities to integrate concepts from both graphics and parallel processing. The AI class is a hybrid of a discipline survey and senior/graduate project-based seminar. Topics include search, machine learning, computer vision, decision support, knowledge representation, neural networks, genetic algorithms, information theory, and natural language processing. In addition to lectures and assignments, students are required to propose and implement a formal project and present the results of their investigation as an integral part of the class.

Computer graphics is a natural fit for the computer vision component of the AI class. For example, Figure 14 shows the result of one Genetic Programming experiment for artistic expression. Here computer programs are evolved under selective pressure. The programs are then executed, producing an image, in this case an abstraction

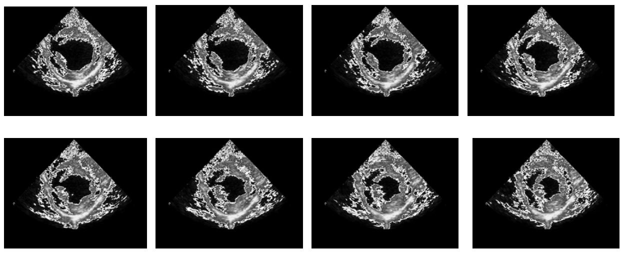


Figure 12. Pulse-Coupled Neural Network Echocardiogram Cineloop

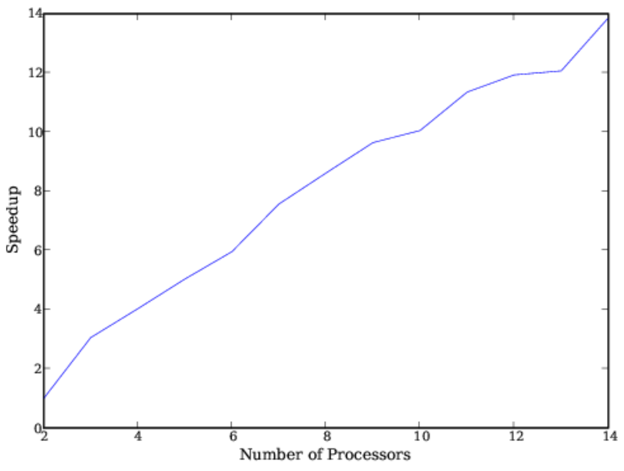


Figure 13. Speedup for PCNN Echocardiogram Processing

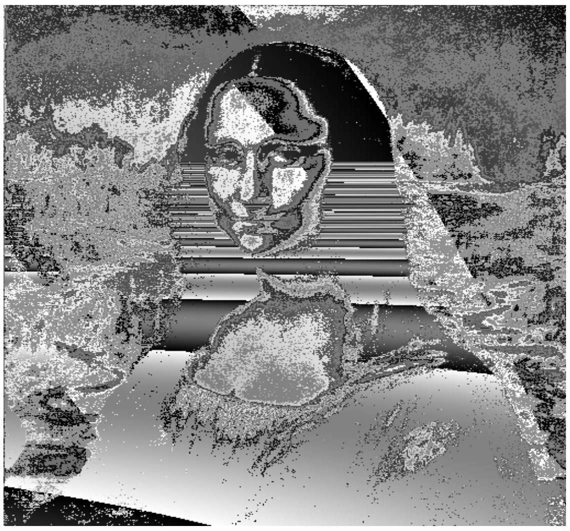


Figure 14. Genetic Programming for Visual Art

of the Mona Lisa. Additional computer graphics informed AI discussions include natural language processing and image retrieval, and a serious discussion of intelligent image segmentation algorithms.

While there would, on the surface, seem to be less overt opportunity for a discussion of parallel processing in AI, in fact parallel concepts form an important topic for discussion. Since much of what we currently believe about intelligence stems from massively parallel biological systems, such as the human brain, there is opportunity to explore parallel topics at a fundamental level. This includes alternate information encodings, such as neural spike intervals, as well as parallel control ranging from that in animal ethology to robotic applications.

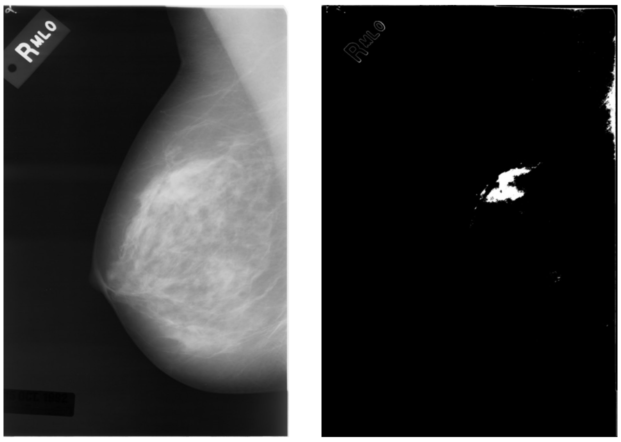


Figure 15. Mammogram with PCNN Identified Lesion

Pragmatically, it is often advantageous to use parallel processing for AI projects. For example Figure 15 shows a mammogram from the Digital Database for Screening Mammography [20] and the corresponding PCNN-isolated malignant lesion. The mammograms represented in these images are on the order of 2800x4700 12-bit pixels in a 16-bit container. Parallelizing this process using a gaming GPU reduced the processing from 13.4 to 1.5 minutes/mammogram with a naïve implementation.

VIII. RESIDUAL BENEFITS

One, often overlooked, benefit of this approach to topical distribution in the curriculum is the accumulation of concepts, examples, and demonstrations suitable for the general education Computer Literacy course [21]. Computer literacy classes for non-computing majors are sometimes received with less than enthusiastic regard by students. They are often viewed as simply another requirement toward graduation, and do not command the attention we would like from students forming the next generation of professionals and policy makers. By featuring a series of demonstrations and experiences, largely drawn from the topical embedding described in this work, we engage students with real-life examples demonstrating both computing principles and possibilities.

Suitable topics include the robotics, haptic, medical imaging, evolutionary computing, art, and natural language processing featured in this paper. Additional topics such as 3D printing, affective computing, encryption, data sonification, and brain-computer interfacing are straightforward extensions of the infrastructure. Taken together, we believe that these experiences provide broad insight into computing that can serve the non-computing students in their future work with only an incremental addition to the effort required for the topical embedding for majors.

IX. SUMMARY AND CONCLUSION

While a formal assessment of the impact of this approach has not been attempted, and it is unclear how to measure such a distributed approach, several anecdotal observations can be made. First, the topics included have been designed, and have in practice, had a minimal time impact on the respective classes. Secondly, there is a record of several students becoming interested and ultimately pursuing both undergraduate and graduate research in areas including AI for medical imaging, bio-inspired computing for mammogram analysis, and GPU programming to accelerate such processing, some leading to publication.

In conclusion, we believe that we have formed an approach to distributing topics that preserves the depth afforded by individual classes while distributing foundational concepts across a subset of the curriculum, thus weaving a fabric for future studies.

REFERENCES

[1] ACM/IEEE-CS Joint Taskforce, "Computer Science Curricula 2013 Final Report 0.9, Pre-release version", *http://cs2013.org*, Oct, 2013

[2] Wolfer,J., "Topical Tapestry: Weaving Threads of Parallel Programming, Computer Graphics, and Artificial Intelligence into Undergraduate CS Courses", IEEE Educon, April, 2014.

[3] Sheldon, M. and Turbak,F., "An Aspect-Oriented Approach to the Undergraduate Programming Language Curriculum", *SIGPLAN Programming Language Curriculum Workshop*, May, 2008.

[4] Prasad S. et al., "NSF/IEEE-TCPP Curriculum Initative on Parallel and Distributed Computing – Core Topics for Undergraduates", *http://www.cs.gsu.edu/~tcpp/curriculum/*, 2012.

[5] Minaie, A. and Sanati-Mehrizy,R., "Incorporating Parallel Computing in the Undergraduate Computer Science Curriculum", *Proceedings 2009 ASEE Annual Conference and Exposition*, 2009.

[6] Danner, A. and Newhall, T., "Integrating Parallel and Distributed Computing Topics into an Undergraduate CS Curriculum", *Proceedings Workshop on Parallel and Distributed Computing Education (EduPar-13)*, 2013.

[7] Mastascusa, E.j., Snyder, W. J., and Hoyt, B. S.,*Effective Instruction for STEM Disciplines: From Learning Theory to College Teaching*, Jossey-Bass, 2011.

[8] Ambrose, S. A. et al.,*How Learning Works: Seven Research-Based Principles for Smart Teaching*, Jossey-Bass, 2010.

[9] Bennett, W. R.,*Scientific and Engineering Problem-solving with the Computer*, Prentice Hall, 1976.

[10] Project Guttenberg Ebook, http://www.gutenberg.org

[11] IBM, "Watson Project", http://www.ibm.com/watson

[12] Google, "Ngram Viewer", http://books.google.com/ngrams

[13] Logitech Products, "iFeel Mouse", http://www.logitech.com

[14] OpenMp, http://openmp.org

[15] Nvidia,"CUDA", http://www.nvidia.com/object/cuda_home_new.html

[16] Khronos, "OpenCL", http://www.khronos.org/opencl

[17] Novint, http://www.novint.com/index.php/novintfalcon

[18] OpenMPI, http://www.open-mpi.org

[19] T. Lindblad and J. M. Kinser, Image Processing using Pulse-Coupled Neural Networks, 2nd. Ed. Springer Verlag, 2005.

[20] Heath, M., Bowyer, K., Kopans, D., Moore, R., and Kegelmeyer, W.P., Digital Database for Screening Mammography, *Proceedings of the Fifth International Workshop on Digital Mammography*, M.J. Yaffe, ed., 212-218, Medical Physics Publishing, 2001.

[21] Wolfer,J., Refreshing the Computer Literacy Course: Computing for the General Education Student, International Congress on Engineering and Technology Education, April, 2014.

AUTHOR

J. Wolfer is with the Computer Science department of Indiana University South Bend, South Bend, IN 46634 USA.

Submitted 01 August 2014. Published as resubmitted by the authors on 14 February 2015.