TEACHING IMAGE AND VIDEO PROCESSING USING MIDDLE-SCHOOL MATHEMATICS AND THE RASPBERRY PI

Marios S. Pattichis[1], Sylvia Celedon-Pattichis[2], and Carlos LopezLeiva[2]

[1]Department of Electrical and Computer Engineering and
[2]Department of Language, Literacy and Sociocultural Studies
University of New Mexico, Albuquerque, NM 87131, U.S.A.
{pattichi, sceledon, callopez}@unm.edu

ABSTRACT

In this paper, we summarize some of the lessons learned from the Advancing Out-of-School Learning in Mathematics and Engineering (AOLME) project. The AOLME project uses an integrated curriculum that relies on the use of basic concepts from middle-school mathematics to teach the foundations of image and video representations. The middle-school students, mostly from underrepresented groups, learn how to program their own video representations using Python libraries running on the Raspberry Pi. Overall, we have found that the students enjoy participating in the project.

Index Terms— Engineering Education, Middle-school Mathematics, Raspberry Pi, Python.

1. INTRODUCTION

There is a strong need to develop effective curricula that can be used to broaden participation and motivate middle school students, especially from underrepresented groups, to pursue STEM careers [1]. Furthermore, it is critical for such efforts to be focused at the middle-school level, before students make up their minds about future career paths [2].

This paper describes an ongoing effort to create a middle-school afterschool program to support students from underrepresented groups in pursuing careers in STEM. The leadership team of the Advancing Out-of-school Learning in Mathematics and Engineering (AOLME) includes Prof. M.S. Pattichis from the Department of Electrical and Computer Engineering, and Professors S. Celedon-Pattichis and LopezLeiva from the UNM College of Education. The paper summarizes the curriculum, lessons learned from its initial implementations, and the actions that were taken to address the issues. Then, we present a revised version of the curriculum based on programming the Raspberry Pi using Python. We also provide a summary of the results from implementing the curriculum.

Other efforts to teach digital signal processing to wider audiences include efforts to encourage experimentation in Java [3] which was also made available on the Web [4]. For image processing, the SIVA demonstration gallery provided a comprehensive system for teaching undergraduate courses in image and video processing in Matlab [3] using an interactive, visual approach [4].

We also have a number of efforts to introduce signal processing at the high-school level [5-8]. The SAS project focused on teaching students about systems based on automatic speech recognition [5]. The INFINITY project was led by a multidisciplinary team that was focused on the development of an engineering curriculum that was built on signal processing concepts [6]. We have a focus on embedded Android applications in the RET project [7] and another effort based on a TI development starter kit as described in [8]. The introduction of the Raspberry Pi motivated the development of new curricula in control education [9] and the internet of things [10].

Our approach is significantly different from earlier efforts in that we are particularly focused on teaching image and video processing concepts to middle-school students from underrepresented groups. Furthermore, our proposed curriculum is based on middle-school mathematics.

The rest of the paper is organized into five sections. In section 2, we describe the initial version of the curriculum. Section 3 summarizes the lessons learned from the initial curriculum implementations. Section 4 provides a summary of the revised curriculum that was based on the Raspberry Pi and Python. Section 5 provides a summary of the evaluation of the curriculum. Concluding remarks and future work are given in section 6.

2. THE INITIAL VERSION OF THE CURRICULUM

The basic motivation for the curriculum came from the fact that middle-school children are already familiar with digital image and video representations. Thus, our original motivation was to integrate middle-school mathematics with the children's interests in digital images and videos to build curriculum that advances literacy in computing systems used for video processing [11].

We adopted a collaborative learning model where the students were broken into five small groups, with up to four students per group, each with an undergraduate student facilitator who coordinated the activities. The material promoted interactivity by requiring the students to work through the concepts in pencil-and-paper and short

programming exercises to ensure that they comprehended the basic mathematics and image processing concepts.

In terms of content, the initial version of level 1 of the curriculum adopted the use of number representations and coordinate systems to describe images as discrete functions over a 2D coordinate plane. Binary images were introduced first. Then, the curriculum covered grayscale images, and color images. Digital color videos were represented using a sequence of digital color images. At the end, the students were asked to work within their groups to design and implement a short digital video using their personal interests to demonstrate what they learned in the course. An example level 1 project is shown in Fig. 1(a).

A level 2 version of the curriculum was also implemented with a focus on processing actual videos. For level 2, after a review of elements of level 1, we introduced several advanced concepts. For example, the students learned how to apply adaptive thresholding, how to apply binary morphology operations to the thresholded images, and edge detection. Then, as in level 1, the students were asked to produce a video that demonstrated their understanding. An example level 2 project is shown in Fig. 1(b).

Level 1 of the curriculum was implemented in the Summer of 2012 at the University of New Mexico. Twenty students from 6th to 8th grade from many different middle-schools were allowed to participate. The groups met Tuesday to Friday from 9am until noon from June 11th to June 29th, 2012. Each student was provided with a single laptop running Matlab in a Windows environment. Then, for level 2 of the curriculum, the students were recruited from the ones who completed level 1 and more advanced students. For the second level, the students came to the University of New Mexico every Saturday from 2:30pm to 5:00pm from March 2nd to May 11th of Spring 2013.

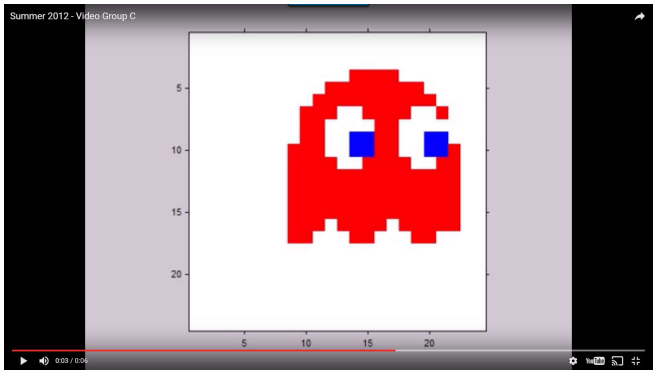
3. LESSSONS LEARNED FROM THE INITIAL CURRICULUM IMPLEMENTATIONS

To evaluate the implementation of the curriculum, we recorded all of the interactions within each student group for later review, and also transcribed specific interactions that were found to be representative of what was happening within each group. Modifications to the curriculum were thus based on a formal review process that addresses issues that were observed during the implementation. We found several issues with this first implementation.

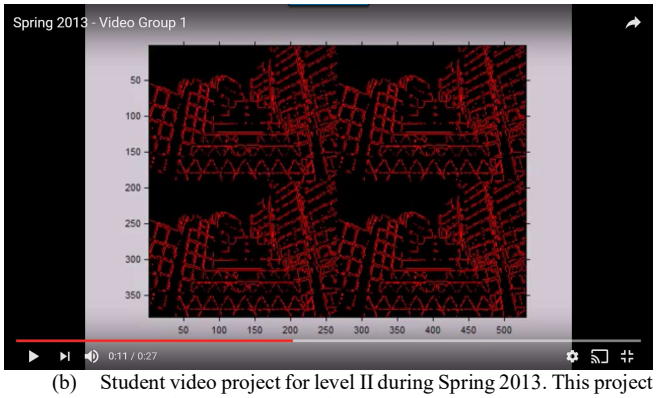
First, the use of Matlab proved prohibitively expensive. There was no way to be able to use Matlab at the middle-schools without incurring a recurring cost on the students. Thus, we decided to switch to Python that was both free and well documented.

Second, the students complained about excessive typing of complicated expressions. We addressed this problem by switching to a cut-and-paste model where the non-essential code and comments were given to the students in a template file.

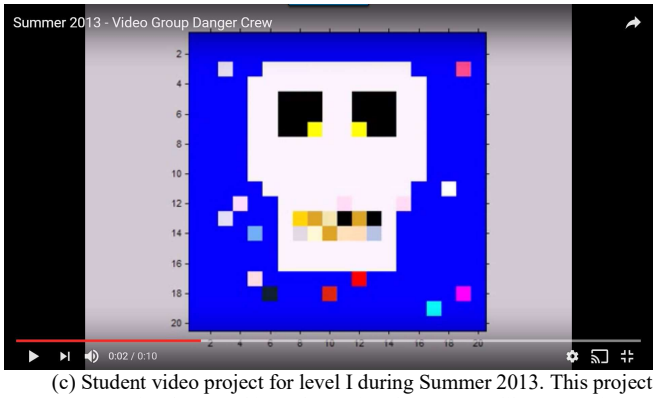
Fig. 1. Middle-school student projects.



(a) Student video project for level I during Summer 2012. This project was implemented in Matlab using a Windows laptop.



(b) Student video project for level II during Spring 2013. This project was implemented in Matlab using a Windows laptop.



(c) Student video project for level I during Summer 2013. This project was implemented in Python using the AOLME library running on the Raspberry Pi.

TABLE I. INTEGRATED CURRICULUM.

Curriculum Level / Activity	Topics	
	Mathematical Content	**Engineering Content**
Level 1: Spring Program		
Basics of Raspberry Pi & Linux		Construct a small computer system & work at the command prompt in Linux
Python Programming	Algebraic operations, variables, substitutions	Code development in an integrated development environment (IDE)
Flowcharts and functions	Mathematical functions and problem solving skills	Problem solving using flowcharts and using functions in programs
Elementary programming concepts	Boolean algebra and arithmetic progressions	Conditional statements and for loops
Coordinate systems & binary images	Coordinate systems	Binary image representations using 2d arrays and bits.
Binary numbers and hexadecimals	Binary, decimal, hexadecimal number systems and conversions between them	Number representations using 8 bits
Creating grayscale images	Positive integer number representations using 2-digit hexadecimal numbers over a 2-d coordinate system	Grayscale image representation using a 2d array of 8-bit hexadecimal numbers
Creating color images	Representing color using 3 hexadecimal numbers that represent Red, Green, and Blue content	Digital color image representation using a 2d array of hexadecimal values
Creating digital color videos	Color functions as 3d discrete functions: (2-d space and time)	Digital color video representation as three 3d arrays containing a sequence of red, green, and blue frames
Project : Small group collaboration	Apply mathematical concepts of Level 1: map continuous-space drawings to discrete representations using pixel-tiles	Understand and implement multidimensional array representations in a small group setting

Third, we had found significant issues with sharing and collaborating within the group. Often, a student with more computer or typing experience would take control of the laptop. The other students were left out although they still provided some feedback to the student typing. Even more alarmingly, some of the left-out students told us that they were afraid to type and were happy to give up their rights to working with the laptop.

To address the laptop sharing issue, we switched to a Raspberry Pi model with a wireless keyboard. This set up facilitated the rotation of the wireless keyboard after typing a maximum amount of lines of code.

The idea of rotating the keyboard worked extremely well. The students were taking turns on the wireless keyboard without being told to do so. Furthermore, while one of the students was typing, the rest of them learned that they were expected to provide feedback.

Fourth, the requirement to bring students to the University created many inconveniences among the parents. For many parents, a daily drop-off and pick-up at the University was difficult since they either needed to drive from far away and/or be available to drive the students at specific times that may not fit their schedules. To address this issue, we changed the location and time of the of the afterschool program to be onsite and compatible with the middle-school schedules. This crucial change proved to be extremely popular. During breaks, the middle-school could provide free lunches to our students. During the regular semesters, the students could simply take the last bus home. On the other hand, students who had chosen another after-school activity which conflicted with ours would not be able to attend our afterschool program.

Fifth, the requirement to build small groups made of students from diverse economic backgrounds also proved to be a challenge. Often, students from middle-schools with a stronger focus on STEM would dominate students from other schools. Furthermore, there were some issues when putting together students from different backgrounds, without any pre-existing relationships or common bonds. Luckily, most of these issues disappeared when we moved implementations to the middle-schools. At the middle-schools, student differences tended to be much less of an issue since they shared similar backgrounds.

4. MODIFIED CURRICULUM USING PYTHON ON THE RASPBERRY PI

The revised curriculum was thus implemented using Python on the Raspberry Pi. The move from the laptop to the Raspberry Pi provided a great opportunity to move beyond basic programming concepts. We provide a summary of the revised curriculum in Table 1.

First, the curriculum was revised to introduce the essential elements of a basic computer system. At the end of the first session, the students were asked to put together a computer system using a wireless keyboard, an external monitor, a mouse, the power supply, and the Raspberry Pi board. They were then asked to describe how input from the keyboard resulted in data flowing through the system from the keyboard to the Raspberry Pi to the computer screen. In the following sessions, the students would spend the first few minutes assembling their basic computer system.

Then, to allow the students to work with the Raspberry Pi, students were provided with basic Linux commands for

starting up, creating directory structures, moving and copying files, as well as shutting down the entire system when done. Initially, given the fact that most of today's students would never work at the command prompt, we were concerned that the Linux activity would turn out to be very difficult for them. Instead, after telling the students that real hackers use the command prompt, they found the use of Linux commands to be very empowering. In fact, there was competition among the students to be allowed to issue the shutdown command in Linux. They knew that if they were allowed to issue a shutdown command in a central computer system, they would effectively bring down many key services.

As described in Table 1, there was a strong integration between middle-school mathematics and engineering content. Each activity began by reviewing the relevant middle-school mathematical content. After mastering the underlying mathematical models, the students were guided through programming exercises.

The curriculum supported collaborative learning through the use of flow-charts and translation to functions. Image pixels were presented as squares that cover the 2D coordinate plane. Pixel values were introduced through the use of number representations. Starting from decimal numbers, the students were introduced to binary and hexadecimal numbers. Each color pixel was thus represented using 6 hexadecimal digits, two for each color in the RGB format. Thus, in the RGB format, #FF0000 represented red, #FFFFFF represented white, and #050505 represented a grayscale level pixel. These representations were programmed in Python as shown in Fig. 1(c).

Furthermore, to support emergent bilingual students who were learning English as a second language, the entire curriculum was translated to Spanish. Thus, in each group, students could be using Spanish only, English only, and most often, they were simply switching between the two languages.

The revised curriculum was implemented in the Summer 2013 at an urban middle-school. The students attended from 9am to noon for June 12-14, 17-21, and 24-25. After a thorough review of the activities, the curriculum was further reduced to its most essential elements. It was then re-implemented at a rural middle-school in the Spring 2014 semester. The students attended the program every Tuesday from 3:15pm until 4:30pm, immediately after they completed their regular classes.

5. EVALUATION OF THE REVISED CURRICULUM

Based on initial reviews of the recorded video interactions and student exit interviews, we found that the students enjoyed developing their color video projects using an interactive, visual environment. Furthermore, the use of digital image and video representations provided a resourceful context for applying mathematical and engineering concepts. Additionally, preliminary findings suggest favorable increase in AOLME students' mathematics and engineering attitudes through their participation in the project.

6. CONCLUSION AND FUTURE WORK

Overall, the AOLME project demonstrated that middle-school students enjoy learning basic concepts associated with digital image and video representations while working in a real programming environment. The students felt empowered by working at the Linux command prompt and did not require the use of Graphical User Interfaces for learning how to program.

In future work, level 2 of the AOLME curriculum will undergo a substantial revision to support collaborative learning using object-oriented programming. Thus, the revised version of the curriculum will maintain a strong focus on encouraging sharing and collaboration across multiple student groups.

7. ACKNOWLEDGMENTS

This material is based upon work supported in part by the National Science Foundation under NSF AWD CNS-1422031 and NSF AWD 1613637.

8. REFERENCES

[1] National Research Council. A framework for K-12 science education: Practices crosscutting concepts, and core ideas. Washington, D.C.: National Academy Press, 2011.

[2] M.A. Mooney, & T.A. Laubach, "Adventure engineering: A design centered, inquiry based approach to middle grade science and mathematics education," *Journal of Engineering Education*, vol. 91, no. 3, pp. 309-318, 2002.

[1] A. Clausen, A. Spanias, A. Xavier, and M. Tampi, "A Java signal analysis tool for signal processing experiments," *1998 IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP'98, vol. 3, pp. 1849-1852, 1998.

[2] A. Spanias et al. "Development and evaluation of a web-based signal and speech processing laboratory for distance learning," *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, ICASSP'00, vol. 6, pp. 3534-3537, 2000.

[3] U. Rajashekar, G.C. Panayi, F.P. Baumgartner, and A.C. Bovik, "The SIVA Demonstration Gallery for Signal, Image, and Video Processing Education," *IEEE Transactions on Education*, vol. 45, no. 4, pp. 323-335, Nov. 2002.

[4] A.C. Bovik, "What you see is what you learn," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 117-123, September, 2010.

[5] D. Sharma, A. Poddar, S. Manna, and P.A. Naylor, "The SAS project: Speech signal processing in high school education." *23rd European Signal Processing Conference* (EUSIPCO), pp. 1781-1785, 2015.

[6] G.C. Orsak, S.C. Douglas, R.A. Athale, D.C. Munson, J.R. Treichler, S.L. Wood, and M.A. Yoder, "The INFINITY Project: expanding signal-processing-based engineering education to the high school classroom," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, (ICASSP'01), vol. 5, pp. 2709-2712, 2001.

[7] M.K. Banavar, D. Rajan, A. Strom, A., P. Spanias, X. Zhang, H. Braun, & A. Spanias, "Embedding Android signal processing apps in a high school math class—An RET project," *IEEE Frontiers in Education Conference* (FIE), pp. 1-4, 2014.

[8] M. Jiménez, R. Palomera, M. Toledo, D. Rodriguez, A. Ramirez, and L. Bautista, "Signal Processing for High School Students: Learning from a Reach-out Experience," *9th International Conference on Engineering Education*, 2006.

[9] J. Sobota, J., Pisl, R., P. Balda, and M. Schlegel, "Raspberry Pi and Arduino boards in control education," *Proceedings of IFAC*, vol. 46, no. 17, pp. 7-12, 2013.

[10] V. Callaghan, "Buzz-boarding; practical support for teaching computing, based on the internet-of-things," *1st Annual Conference on the Aiming for Excellence in STEM Learning and Teaching, Imperial College, London & The Royal Geographical Society*, pp. 12-13, April 2012.

[11] S. Celedón-Pattichis, C.A. LópezLeiva, M.S. Pattichis, & D. Llamocca. "An interdisciplinary collaboration between computer engineering and mathematics / bilingual education to develop a curriculum for underrepresented middle school students," *Cultural Studies of Science Education*, vol. 8, no. 4, pp. 873-887, 2013.