An Introduction to Computer Vision for First-Year Electrical and Computer Engineering Students

Mr. Daniel Tai Klawson, University of Maryland, College Park

Daniel Klawson is a senior studying electrical engineering at the University of Maryland, College Park. He has been a teaching assistant for ENEE101 for the past four semesters.

Mr. Nathaniel Alexander Ferlic, University of Maryland

Current graduate student at the University of Maryland who's current teaching assistant position is for the course ENEE101.

Mr. Cheng Peng, University of Maryland, College Park

Advised by Prof. Rama Chellappa

Work in Progress: An Introduction to Computer Vision for First-Year Electrical and Computer Engineering Students

Daniel T. Klawson, Nathaniel A. Ferlic, and Cheng Peng
Department of Electrical and Computer Engineering, University of Maryland, College Park

***Abstract*-- This work-in-progress paper will detail one of ENEE101's newest modules, computer vision. ENEE101 is the introductory course to electrical and computer engineering (ECE) at the University of Maryland (UMD) [1] [2]. This course provides first-year students with a glimpse into the broad field of ECE through high-level hands-on labs, with the goal of increasing student retention rates and boosting performance in sophomore-year courses; preliminary results have shown an upward trend in major retention and a downward trend in failures. Faculty-proposed modules cover a wide range of sub-disciplines in ECE, including optical communications, internet of things, and computer vision. Computer vision has become a popular topic in academia and industry due to its applications in machine learning, artificial intelligence, image recognition, self-driving cars, and more. Through our computer vision module for ENEE101, we seek to answer the following question: how can freshman students, with possibly no prior knowledge of basic programming, actively learn and engage with computer vision? Our solution is to present students with three hands-on labs using the well-known Microsoft Kinect hardware along with open source computer vision software libraries. The labs we introduce cover depth sensing, hand tracking, facial recognition, and body detection. Each topic covers a single day of lab where the students are taught the basics of each concept and complete a C++ template with simple but elegant solutions, built and executed with Microsoft Visual Studio. The goal is to help students realize the impactful applications of computer vision by exposing them to complex computer vision topics through easily understandable, real-life scenarios. By achieving this goal, we better prepare students for careers as scientists and engineers.**

***Index Terms* — Body Tracking, Computer Vision, Depth Sensing, Facial Recognition, Gesture Recognition, Module Based Learning**

I. Introduction

Introduction to Electrical and Computer Engineering (ENEE101) at the University of Maryland, College Park (UMD) is a mandatory, first-year course for all electrical and computer engineering (ECE) majors at UMD. The course introduces students to ECE topics through engaging, hands-on labs covering most of the topics in ECE. At the end of each week, a seminar is given by a prominent faculty member who specializes in each respective topic.

Computer vision is fundamental to many significant research fields, and it is a complex, technically intensive topic which is difficult even for undergraduates to grasp. Through this module for ENEE101, students are gently but purposefully introduced to computer vision topics to captivate and inspire. Students that experience hands-on computer vision labs will hopefully spur interest in the emerging fields of machine learning, artificial intelligence, image processing, and self-driving cars.

II. Equipment Setup

The equipment includes the Microsoft Kinect for Xbox 360, a Windows 10 PC, and open source libraries included in OpenNi, NiTE2, and OpenCV. The Kinect was chosen for its unique sensing capability that allow simultaneous image acquisition and pixel-level range mapping. It is also very affordable (under $100) and thus accessible to most educational institutions. Additionally, it is also quite familiar to many students since it was the cornerstone of Microsoft's popular gaming console from the past decade. The functions from the software libraries interface with the Kinect, called by C++ codes and compiled in Microsoft Visual Studio. OpenNi and NiTE2 allow control of the Kinect's sensors. OpenNi interfaces with the depth sensor and extract data such as pixel depth and XYZ position [3], while NiTE2 is a powerful engine that deploys 'skeleton' body tracking and gesture recognition [4]. Using OpenNi and NiTE2 for feature recognition has been deployed by others [5], and some of their algorithms were employed in the development. The OpenCV library interfaces with OpenNi and NiTE2 to allow for our real-time vision applications [6].

III. Lab Procedures

A. Depth Sensing

The students were first given a pre-lab reading assignment to familiarize themselves with depth sensing by the Kinect (see Appendix A), as well as simple C++ code. Then, students are given a C++ skeleton code that provides the lateral coordinates (X position, Y position) with respect to the screen, and depth of each image pixel. However, the depth rating they are given is in units unique to Kinect, yielding no easily discernable meaning. They are instructed to modify the software to answer the following, using this procedure:

1. *Map a function converting pixel depth from the Kinect's units to depth in meters*
2. *Calculate 2-D distance between two objects, in a normal plane with the Kinect*
3. *Calculate 3-D distance between two objects, considering differences in depth*
4. *Evaluate error between theoretical and actual distances, and explain why error occurs*

After the exercise, students would have successfully created their own distance measurement program. Along with the coding practice, students are reminded of the multiple applications of distance sensing such as self-driving cars, object recognition, smart surveillance, position reckoning in robots, and so forth. After the lab, students would have reduced the abstract concept of distance sensing into practice.

B. *Facial Recognition*

The next 110-minute lab is devoted to facial recognition - it is pedagogically structured similarly to the first. Students start with an informative pre-lab assignment, introducing Haar cascades at a high level, i.e. simple black and white patterns used to detect common facets of an object, such as shadows under one's eyes [7]. The key idea is that the Haar cascade algorithm uses simple geometric shapes to recognize complex facial features (see Appendix B). Having some understanding of how to implement the computer vision algorithm to determine a face, students then demonstrate these concepts in real-time. Using the skeleton code provided, they are tasked with replacing all faces in a video frame with a picture of their choice. The lab heuristic is as follows:

1. *Load in image of choice and resize to the size of each face in frame*
2. *Place image to location of each face, changing with each frame*
3. *Save over the frame and output to video stream*

An example of the expected output is shown in Figure 1. Like the first lab, this rudimentary exploration of facial recognition helps students appreciate the use of a specific image recognition algorithm. More advanced students are encouraged to explore the algorithm deeper - more sophisticated codes were demonstrated by researchers from the University of Erlangen-Nuremberg, where they transferred facial expressions in real time from a source actor to a target actor using advanced feature recognition [8]. At the end of this lab, students understand how image processing algorithms are implemented and applied in the real world.

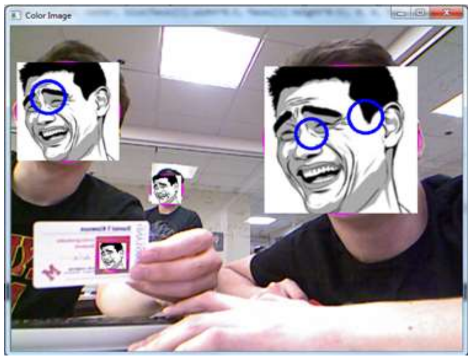

Figure 1: Each face in the frame is replaced with an appropriately sized picture, including the face on the subject's ID card

C. *Facial Recognition*

The final lab is focused on yoga pose identification. The students start with a pre-lab assignment familiarizing themselves with the NiTE2 libraries. The goal of this lab is for students to use existing NiTE2 body tracking architecture to recognize distinct yoga poses - the lab software detecting a Warrior Two yoga pose can be seen in Figure 2. The hands are held even, and the subject's right leg is set at a 90-degree angle and the left leg is set at 45 degrees. Each pose has distinct actions that makes it unique - the students are challenged with creating a program to recognize a unique yoga pose. The skeleton code provides the green nodes and blue branches making up the subject skeleton seen in Figure 2. These are called points and joints respectively. From this code template, the students follow the given heuristic:

1. *Select a well-known and executable yoga pose*
2. *Using point locations, joint locations, and joint angles, logically program a yoga pose detector*
3. *Test the code for false positives or false negatives and refine*
4. *Determine the limitations of the body tracking software*

At the end of the lab, students would have gained exposure to logical programming skills applied to feature recognition, using a breadth of conditional statements to check for each part of the yoga pose.

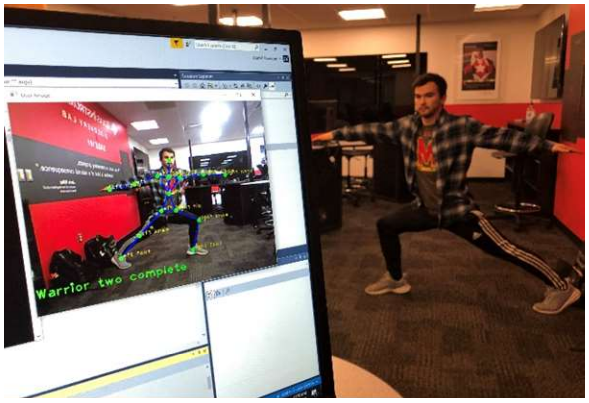

Figure 2: A demonstration of successful pose identification – a subject completes the Warrior Two pose, and the lab application prints a success

IV. Results

At the end of the semester, students were asked to complete an anonymous survey about their experience in ENEE101. The students ranked the modules from most to least favorite (1-12), and explained the reasoning behind their ranking. The survey results are used to determine where improvements can be made to the next semester's curriculum. Changes could be as minor as editing lab manuals, upgrading hardware, or debugging software, or as major as removing and replacing entire modules.

Figure 3 shows a plot of the average ranking per module from the Spring 2019 semester. Compiled from 90 student responses, a lower ranking signifies the module was well received, whereas a higher ranking means the module was not. On the chart, the x axis (1-12) corresponds to a chronological ordering of the modules as follows: Optical Communications (1), Energy Harvesting (2), Brainwaves (3), MATLAB Programming (4), Image Processing (5), Engineering Ethics (6), Computer Vision (7), Cybersecurity (8), Digital Circuits (9), Microprocessors (10), Android App Inventor (11), and Internet of Things (12).

The computer vision result is highlighted in red.

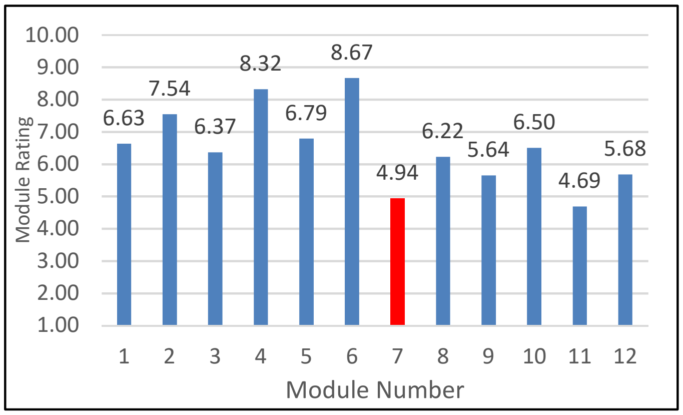

Figure 3: Average rankings of each module from the Spring 2019 semester

Compared to the other eleven modules, the computer vision module was the second most popular. This was likely due to the exciting, hands-on nature of each lab – the following quotations are sample student responses as to why they ranked the computer vision module as their favorite:

- *"My favorite was the computer vision module. It felt like the most cohesive module, and it was nice to see results."*
- *"Computer vision was really fun because it let you actually see what the computer could see and train it."*
- *"I enjoyed the computer vision module because it is a topic I am very interested in. Artificial intelligence and tracking images is an emerging field with great implications."*
- *"Computer vision, interesting topic and fun to implement and experiment"*
- *"The Computer Vision due to how advanced it can be on a high level, but also with how simple it was to implement through the provided libraries."*

V. Outlook

It is clear that ENEE101's computer vision module was well received by students, according to the Spring 2019 survey. The student body response points toward successfully reaching the first goal the module strove to meet – to captivate and inspire first year ECE students. The survey responses show that the module made the complex topic of computer vision fun and accessible, and consequently well liked. What needs to be explored more is if this module, rather than just solidifying students' interest in ECE, truly ignites students' passion for computer vision.

VI. Conclusion

Computer vision is an abstract and mathematically intense field that requires knowledge of image analysis, filtering and linear algebra and algorithms. In order to be accessible to first year students, these concepts are taught by using a ubiquitous gaming hardware and skeleton C++ code. Through the three labs in the module, students build their own computer vision system to sense depth, detect faces, and recognize yoga poses. This proves to students that computer vision is not an incomprehensible, abstract topic but an attainable one. Survey results show that the students enjoyed the module because of its understandable, interactive nature. Hopefully, as more students are introduced to this topic through these simple labs, the next generation of computer vision researchers will be born.

VII. Appendices

Appendix A:

Lab documents, including prelab assignments, lab manuals, and the computer vision application can be pulled from GitHub using the following link: https://github.com/NateFe/ComputerVision

Appendix B:

Documentation for how OpenCV detects faces with Haar cascades can be found using the following link: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html

VIII. Acknowledgments

The authors would like to thank Dr. Romel Gomez for his enthusiastic encouragement, experienced advising, and thorough editing of this paper. Furthermore, the authors thank Dr. Joseph JaJa for his financial support of our efforts, along with Dr. Ramalingam Chellappa and Dr. Julius Goldhar for their support. Finally, the authors would like to thank Mr. Kevin Ho and Mr. Cheng Qu for their work on the computer vision module.

IX. References

[1] R. Gomez, B. Babadi, S. Bhattacharyya, J. Goldhar, A. Khaligh, N. Mogul, W. S. Levine, M. Wu and R. Chellappa, "ENEE 101: On a gadgets driven freshman course for improving first year retention rates,"

[2] R. Gomez, B. Babadi, S. Bhattacharyya, J. Goldhar, A. Khaligh, N. Mogul, W. Levine, M. Wu and R. Chellappa, "ENEE 101 "What's Cool in ECE" – A teaching innovation for first year retention"

[3] "Kinect and OpenNI — OpenCV 2.4.13.7 documentation", *OpenCV Documentation*, 2014. [Online]. Available: https://docs.opencv.org/2.4/doc/user_guide/ug_kinect.html. [Accessed: 07- Mar- 2019].

[4] PrimeSense Inc., "Prime Sensor NITE 1.3 Algorithms Notes", 2010.

[5] E. H. Miller S. Gasparrini, E. Cippitelli, S. Spinsante and E. Gambi, "A Depth-Based Fall Detection System Using a Kinect® Sensor", *Sensors*, vol. 14, no. 2, pp. 2756-2775, 2014. Available: 10.3390/s140202756 [Accessed 7 March 2019].

[6] *OpenCV*. Intel, 2013.

[7] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," *in Computer Vision and Pattern Recognition*, Kauai, HI, USA, 2001.

[8] J. Thies, M. Zollhöfer, M. Nießner, L. Valgaerts, M. Stamminger and C. Theobalt, "Real-time expression transfer for facial reenactment", *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 1-14, 2015. Available: 10.1145/2816795.2818056 [Accessed 7 March 2019]. Biographies