MAKING COMPUTER VISION ACCESSIBLE FOR UNDERGRADUATES [*]

Scott Spurlock and Shannon Duvall
Computer Science Department
Elon University
Elon, NC 27302
336 278-6192
sspurlock@elon.edu

ABSTRACT

This paper describes a newly developed course in computer vision targeted at mid-level undergraduate computer science students. Computer vision is an area of increasing importance in many recent and emerging applications, but has tradi-tionally been taught at the graduate or advanced undergraduate level. To make this challenging topic accessible for a wider audience, the course focuses on practical application and experimentation, with reduced emphasis on advanced program-ming and mathematics. Weekly hands-on lab assignments help students gain an in-tuitive and experimentally grounded understanding of a wide variety of topics, from color spaces to image features to object tracking. We present an overview of the course, student outcomes, and lessons learned.

INTRODUCTION

Computer vision is a subdiscipline of artificial intelligence with the goal of automatically extracting useful information from images. In recent years, computer vision has grown from isolated niche successes (e.g., in machine part inspection or optical character recognition) to successful deployment within a large number of real-world applications, such as Snapchat filters and face detection in camera appli-cations. This momentum seems likely to continue into the future with such promis-ing new developments as self-driving cars on the horizon. Within university curricula, computer vision courses are generally geared toward graduate or advanced undergraduate students.

[*] Copyright © 2017 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

This limitation is due in large part to the challenging prerequisites for an in-depth study of the topic. Typical prerequisites include advanced mathematics courses (multivariable calculus and linear algebra) as well as computer science (CS) courses such as data structures and numerical methods [1].

In this paper, we describe a course developed specifically to target mid-level undergraduate students in both our CS program and also our information science program, which does not require the more advanced theoretical courses. Instead, the only prerequisites are introductory calculus and the second course in the CS sequence. Our goal is to provide mid-level electives that are relevant and interesting to students. Previously, the only courses offered at the mid-level were core courses (algorithm analysis, programming languages). We argue that a course like computer vision is arguably a bigger draw for students. By making it available at the lower level, we hope to expose students to research areas earlier, draw more minors into our department, and potentially entice a minoring student to major.

Our course is designed to be primarily an active experience for the students, with preference given to developing an intuition for the strengths and weakness of different approaches through experimentation and application over implementing algorithms from scratch. Rather than the standard format of lectures and exams, we take a hybrid approach of some lectures with more class time spent on interactive demonstrations and hands-on laboratory exercises. This gives students experience debugging, problem solving, and critically analyzing and comparing algorithms, while at the same time exposing them to the types of higher-level math underlying the algorithms, and kinds of research questions currently being studied in the field. In this way, it bridges the lower-level, "learn-to-program" courses with the upper-level theoretical and software engineering courses.

RELATED WORK

A variety of approaches to computer vision education have been discussed in the literature. Bebis et al. provides a thorough survey [1]. More recently, several papers have discussed the development of related courses. Most commonly, the target population is graduate students. In one, a single problem (traffic sign recognition) provides the basis for the entire course within a project-based learning framework [4]. Another incorporates research benchmarks to motivate graduate student interaction with the material [5]. Other related courses have undertaken hybrid approaches that include content from mathematics [6] or embedded systems [8].

Other authors have considered an undergraduate audience for computer vi-sion. One approach focuses on leveraging computer vision coursework to attract students to become involved in research [9]. Most related to our work, Miguel de-scribes an image processing course with an applied focus and targeted at an undergraduate population [7]. This course was, like ours, conducted in a lab environment with frequent hands-on exercises; however, the focus was more narrowly on image processing rather than a broader spectrum of computer vision topics. Different from our open-source software approach, the course employed commercial software, incorporating MATLAB as the environment of choice. Sage and Unser includes a (somewhat dated) review of software options for teaching image processing and describe incorporating the Java-based ImageJ into interactive labs to supplement an existing course [11]. Other recent work describes the development of a custom, specialized Java-based library for teaching computer vision [2]. There have also been some efforts to embed computer vision into the existing CS curriculum either as a final-year, capstone-style experience [3], or with an eye toward increasing engagement and retention in introductory courses [12].

COURSE OVERVIEW

The course was offered in spring 2016. This was the first time computer vision was taught at the university. The class prerequisites were simply CS 2 and Calculus 1. The course attracted students both from the CS program, as well as our less technically rigorous information science program. Of the 28 enrolled students, each class, from freshmen through seniors, was represented. The course was taught in a computer lab, 3 days a week in 70-minute blocks.

Software

We selected the open source library OpenCV for the course. Others have in-corporated MATLAB (e.g., [3, 7]) or custom software developed specifically for the course ([2, 9]). Among the advantages afforded by OpenCV are open licensing providing free usage for both academic and industry applications, interfaces for multiple languages and platforms, and an active online community providing support, documentation, and tutorials. We elected to use the Python bindings for OpenCV, leveraging the Anaconda distribution, which includes many packages to simplify scientific computing (e.g., NumPy, Matplotlib) and a development environ-ment, Spyder, that facilitates interactive exploration.

Textbook

As others have noted, finding an appropriate textbook to accompany a com-puter vision course aimed at undergraduate students can be difficult due to the level of assumed prerequisites [7, 9]. We adopted Rosebrock's Practical Python and OpenCV [10]. Although lacking in theoretical underpinnings, the text functioned primarily as a lab manual, providing detailed instructions to develop basic applications. A key feature is that the content allowed students with no prior Python experience to develop a working competency in that language organically by following along with the examples. A more rigorous treatment of the underlying theory, mathematics, and algorithms beyond what was available in the textbook was provided through lecture and class notes.

Course Objectives

We identified a variety of objectives for the course, relating both specifically to computer vision as well as to students' overall development as computer scientists. Specifically, following the course, students should be able to

- Describe the major problems in computer vision, the fundamental challenges in solving each of these problems, and several approaches to solving these problems and their strengths and weaknesses.

- Use Python and open source libraries (OpenCV, NumPy) to apply existing computer vision techniques to solve new problems.
- Create effective visualizations to communicate algorithm functionality and to analyze parameter values.
- Use technical manuals and online resources to aid in solving computer vision problems.
- Improve ability to work as part of a team in terms of planning, scheduling, communication, and collaboration.

Course Structure

The course was taught in a computer lab, 3 days a week in 70-minute blocks. For a given week, day one would be primarily devoted to lecture and interactive demonstrations, day two would be divided between lecture and in-class work time on a lab exercise, and day three would be entirely devoted to lab work. This focus on in-class work time proved highly valuable, giving students one-on-one instruction as needed. Each assignment covered a particular application, such as finding faces in a web cam video or building a book cover matching system. In addition, there were two exams, each with a multiple-choice and programming component. In all, there were 11 lab assignments, as well as three larger homework assignments, and a large final project on a student-selected topic. Every assessment except exams required students to work in two-person teams to solve challenges and write up a report answering questions and describing their findings.

The weekly topics and related lab assignments are shown in the table below.

Week	Topics	Lab
1	Image basics, Python, NumPy	Augment a script to read and write an image, modify pixel colors, and show a simple animation.
2	Color spaces	Implement a variety of functions, including finding an image's average color within a region and color scheme transfer.
3	Thresholding, morphology	Implement color-based detection to find and label objects in a variety of different images.
4	Filtering, denoising, gradients, edges	Add and remove different types of noise in images, find image gradients, and use edge detection.
5	Template matching, precision, recall	Implement multi-scale template matching and calculate precision and recall

6	Features	Experiment with existing Harris corner and SIFT feature detectors and make observations about invariance to common image transformations.
7	Feature matching	Estimate image scale and rotation warps based on matching detected keypoints.
8	Image search, RAN-SAC, homographies	Collect a database of images (book covers, CDs, etc.) and a variety of query images. Implement algorithm to match query images against database images.
9	Face detection, working with video	Experiment with detection with previously trained models for faces, eyes, and mouths and implement real-time face swapping for video input.
10	Face recognition	Collect positive and negative cropped face images and train an eigenface-based model for recognition.
11	Tracking	Implement color-based tracking for live video input.

Homework Assignments

There were three out-of-class programming assignments for the course. Students were provided with starter code and input images in each case.

1. **Reconstructing color images:** based on an assignment developed by Alexei Efros, students are tasked with reconstructing color images from filtered gray-scale images originally captured in the early 20th century by Russian photogra-pher Sergei Mikhailovich Prokudin-Gorskii, long before the technology for printing color images existed.
2. **Hybrid images:** based on an assignment by James Hays, the goal is to develop hybrid images, or optical illusions that blend two different pictures together so that one is visible from close-up, while the other is visible from a distance.
3. **Puzzle solving:** based on an assignment developed by Richard Souvenir, stu-dents are given a set of simplified, square puzzle pieces and a reference image and asked to reconstruct the original image by solving the puzzle.

Final Project

`Students had the freedom to select their own topic for the last project, with the goal of applying the knowledge and skills gained from the course to a new computer vision problem. Students presented their projects to the class in lieu of a final exam. To help students work toward completion, three milestones were scheduled over the last six weeks of class: project proposal, midpoint status update, and pro-ject completion. The proposal milestone proved particularly important to help students select projects of the right scope and difficulty level, and establish a reasona-ble plan for moving forward. Student projects included such topics as basketball shot prediction, handwritten math solving, keypad PIN theft, and virtual hairstyle modeling. (See figure 1.)

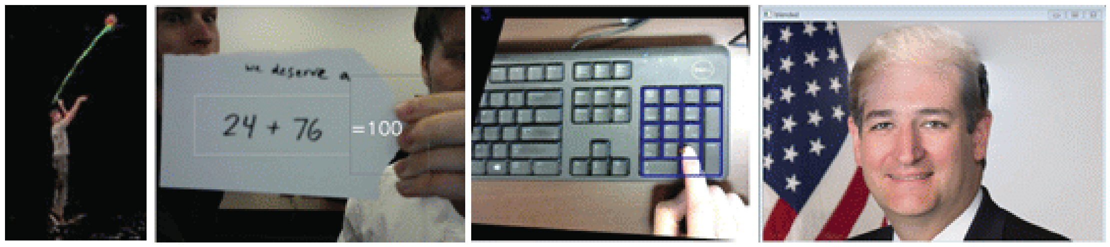

Figure 1: Example images from student final projects.

OUTCOMES

Overall, student outcomes were very successful, as evidenced by the high average quality of their work. Many of the final projects in particular were very strong, displaying creative application of computer vision techniques to novel and difficult problems. Student feedback was positive for the course. Student comments support the effectiveness of the course structure:

- "Labs were related to real world problems"
- "Very hands on … fun"
- "I enjoyed the class time to work on labs"
- "Classes were engaging and homework and labs were interesting."
- "Live demos of code, allowing students to work on labs and homeworks in class enabled students to ask questions when they encountered problems"

CONCLUSIONS

There are several areas where we encountered challenges that will influence updates for the next course offering. One issue was uneven preparation among students, since some of the students were already proficient in Python and for others, their only prior programming experience was Java. Although the approach of allowing students to discover Python features as-needed for particular assignments was generally successful, spending more time explicitly introducing Python basics would be beneficial to many. We plan to assign a homework 0 exercise to encourage students to master these basics early on. Since students had a range of skills, we included bonus challenge problems on all assignments and plan to extend these options in the future.

Another concern is that exam scores were generally lower than desired. The students showed a marked preference (and aptitude) for practical programming challenges over more theoretical questions. For the future, we plan to place slightly more emphasis on theory-based questions in lab reports to help deepen understanding of this aspect of the material.

Computer vision is topic that naturally encourages student engagement and enthusiasm, and has increasing practical importance as industry embraces related technologies in applications. The hands-on style and emphasis on practical application helped students achieve the desired learning objectives. In addition to computer vision-specific knowledge, students showed improvement in their programming, debugging, technical writing, and teamwork skills. Further, several students have shown an interest in getting involved with research following the course, a phenomenon observed by others as well [9]. We feel that the inclusion of the course in our curriculum has been well worth the effort of developing it.

REFERENCES

[1] Bebis, G., Egbert, D., Shah, M., Review of computer vision education, *IEEE Trans-actions on Education*, 2003.

[2] Cazorla, M., Viejo, D., JavaVis: An integrated computer vision library for teaching computer vision, *Computer Applications in Engineering Education*, 2015.

[3] Cielniak, G., Bellotto, N., Duckett, T., Integrating mobile robotics and vision with undergraduate computer science, *IEEE Transactions on Education*, 2013.

[4] Gerónimo, D., Serrat, J., Lopez, A., Traffic sign recognition for computer vision project-based learning, *IEEE Transactions on Education*, 2013.

[5] Hassner, T., Bayaz, I., Teaching computer vision: Bringing research benchmarks to the classroom, *ACM Transactions on Computing Education*, 2015.

[6] Ibrahim, R., Bakri, N., Salleh, T.S.A., Zin, Z., Incorporating mathematics in teaching and learning of image processing, *Intl. Congress on Engineering Education*, 2012.

[7] Miguel, A., Hands-on, Interactive Undergraduate Digital Image Processing Course, *New Trends in ECE Education*, 2006.

[8] Morison, G., Jenkins, M. D., Buggy, T., Barrie, P., An implementation focused ap-proach to teaching image processing and machine vision-from theory to beagle-board, *IEEE European Embedded Design in Ed. and Research Conf.*, 2014.

[9] Olson, C., Encouraging the development of undergraduate researchers in com-puter vision, *ACM SIGCSE Bulletin*, 2006.

[10] Rosebrock, A., *Practical Python and OpenCV*, PyImageSearch, 2016.

[11] Sage, D., Unser, M., Teaching image-processing programming in Java, *IEEE Signal* Processing Magazine, 2003.

[12] Strayer, S., Resources for integrating computer vision into the computer science curriculum, *Journal of Computing Sciences in Colleges*, 2017.