CS 6476-A Computer Vision

Fall 2026 **Scheller 100**

Instructor: **James Hays**

Course Description

This course provides an introduction to computer vision including fundamentals of image formation, camera imaging geometry, feature detection and matching, stereo, motion estimation, convolutional networks, image classification, segmentation, object detection, transformers, and 3D computer vision. We'll explore methods for depth recovery from stereo images, camera calibration, automated alignment, tracking, boundary detection, and recognition. We'll use both classical machine learning and deep learning to approach these problems. The focus of the course is to develop the intuitions and mathematics of the methods in lecture, and then to implement substantial projects that resemble contemporary approaches to computer vision.

Learning Objectives

Upon completion of this course, students should be able to:

- 1. Recognize and describe both the theoretical and practical aspects of computing with images. Connect issues from Computer Vision to Human Vision
- 2. Describe the foundation of image formation and image analysis. Understand the basics of 2D and 3D Computer Vision.
- 3. Become familiar with image processing approaches like convolution and their use in registration, alignment, and matching in images (i.e. classical computer vision).
- 4. Become familiar with machine learning (including deep learning) concepts used in object categorization, segmentation, and object detection in images. Apply similar approaches to data such as 3D point clouds from lidar sensors.
- 5. Implement computer vision techniques using Python and deep learning toolboxes such as PyTorch.

Prerequisites

No prior experience with computer vision is assumed, although previous knowledge of visual computing or signal processing will be helpful. The following skills are necessary for this class:

- Data structures: You'll be writing code that builds representations of images, features, and geometric constructions.
- Programming: Projects are to be completed and graded in Python and PyTorch. All project starter code will be in Python. TA's will support questions about Python. If you've never used Python that is OK, as long as you have programming experience, but you will be playing catch up compared to most students.
- Math: Linear algebra, vector calculus, and probability. Linear algebra is the most important and students who have not taken a linear algebra course have struggled in the past. If you want a refresher on linear algebra, check out **Prof. Gilbert Strang's lectures**.

Compute Requirements

The institute has **minimum compute requirements** for all students that you are expected to meet. The projects are somewhat compute intensive, though, so a faster

machine will let you iterate more quickly. The institute requirements say nothing about GPUs, unfortunately. The deep learning projects can benefit from (but will not require) a local GPU. Since we cannot rely on students having GPUs, projects will rely on cloud services such as Google Colab or the **PACE ICE Cluster**.

Grading

Your final grade will be made up from

- 60% 6 programming projects, each 10.00% of your final grade.
- 40% from three in class written quizzes.

Final grades will be A/B/C etc for 90.00%+, 80.00%+, 70.00%+, etc. with no rounding.

You will lose 10% each day for late projects. However, you have six "late days" for the whole course. That is to say, the first 24 hours after the due date and time counts as 1 day, up to 48 hours is two and 72 for the third late day. A late day cannot be split among projects (e.g. half a late day for project 1, and half a late day for project 2). They are used in integer amounts. There is no grace period for project handins. 5 minutes after the deadline is a full day late.

If you are taking this course, the expectation is that you have set aside the considerable amount of time needed to get your projects done in a timely manner. These late days are intended to cover unexpected clustering of due dates, travel commitments, interviews, hackathons, computer problems, extracurricular commitments, etc. Don't ask for extensions to due dates because we are already giving you a pool of late days to manage yourself. If you are seriously ill and need more time on projects beyond what late days can cover, you should submit documentation to the Dean of Students office and they will reach out to us.

Academic Integrity

Academic dishonesty will not be tolerated. This includes cheating, lying about course matters, plagiarism, code copying, or helping others commit a violation of the Honor Code. Plagiarism includes reproducing the words of others without both the use of quotation marks and citation. Students are reminded of the obligations and expectations associated with the **Georgia Tech Academic Honor Code**. We will use

tools to find code sharing in projects.

You are expected to implement the core components of each project on your own, but the extra credit opportunties often build on third party data sets or code. That's fine. Feel free to include results built on other software, as long as your handin cites the toolboxes you build on.

You should not view, edit, or copy anyone else's code. You should not publicly post code to Canvas, except for starter code or helper code that isn't related to the core project.

More detailed guidance about what is and is not allowed in terms of collaboration with other students:

Allowed

- Talking at the non-code level about project strategies, e.g. "I found it works best if you increase the channel depth of the bottleneck layers". But this does not mean you can collaborate or share answers for the non-code portions of projects and problem sets.
- Explaining concepts or project strategies at the "white board" level, for example drawing out how the SIFT ratio test should work between two sets of keypoints.
- Studying together to better understand anything in the lecture slides.
- Telling another student about a handy function you found. But it would not be allowed to tell them where to insert it in their code, what the arguments to the function should be, etc.
- Telling another student about a debugging strategy.
- New since fall 2025: Using Copilot, Cursor, or other coding assistants.

Not allowed

- Viewing, editing, or copying another student's code for any reason.
- Offering or receiving exact code or hyperparameters, e.g. "Use these parameters for layer 1, these for layer 2, with this learning rate, and this random initial seed..."

- Debugging another student's code.

Getting Help

The projects for this course are substantial and it will be common for students to need clarifications or guidance. Some of the explanation for projects will come from the lecture materials. Some guidance come from the project handout itself. If you are stumped or blocked, you can seek help from other students (within the parameters outlined above), you can ask for help on Piazza (but don't post any code unless your question is private), or you can attend TA office hours. Do not wait until the last few days of a project to post Piazza questions or attend office hours because the TAs might be swamped.

The TAs job is not to debug your code. Do not dump entire source code files onto Piazza asking TAs to fix a bug. Do not hand your laptop to a TA at office hours and ask them to fix your code. The TAs primary goal is to make sure you understand the computer vision concepts necessary to complete the projects. Sometimes there will need to be discussions or troubleshooting at the code level, but don't be surprised if TAs try to minimize such interactions. If you want help from TAs, you should be ready to present the debugging that you've already done, e.g. "On this type of input, here's a visualization of the output that I'm getting".

Over the years, we have refined the projects and added unit tests and autograders so that students can more easily debug their own projects. Unfortunately, this sometimes leads to lazy development patterns. Students will express frustration that "my code passes the unit tests but the autograder gives errors" and expect the TAs to fix this situation. Keep in mind that the unit tests are *necessary* but not *sufficient* checks for code correctness. The unit tests help catch some common errors, but it is *expected* that code might pass unit tests and fail the autograder if you did not implement something correctly. Even the autograder cannot catch all problems for all projects, because it can be difficult to check the correctness for certain computer vision algorithms. Your code might pass unit tests, get full marks from the autograder, but still have subtle errors that manifest elsewhere in the project. You will have more success in this class if you make an effort to understand the computer vision concepts being taught and not simply optimize for the autograder.

Learning Accommodations

If needed, we will make classroom accommodations for students with documented disabilities. These accommodations must be arranged in advance and in accordance with the office of disability services. (**disabilityservices.gatech.edu**).

Student and Faculty Expectations

Georgia Tech has a **broad set of expectations** that we should all strive to follow.

Important Links:

- **Canvas**, for announcements, grades, Zoom, and archived recordings of lecture. Canvas also has links to Piazza and Gradescope.
- **Piazza**, for questions and discussion
- **Gradescope**, for project handin

Contact Info and Office Hours:

If possible, please use Piazza to ask questions and seek clarifications before emailing the instructor or staff.

- James: hays[at]gatech.edu
- Yiming Chen: ychen3868[at]gatech.edu
- Kyle Kam: kkam9[at]gatech.edu

Office Hours

- James: Immediately after lecture
- TA hours: See pinned post in Piazza

Projects (available on Canvas)

Optional Project 0: Test Environment Setup.

Project 1: Convolution and Hybrid images.

Project 2: SIFT Local Feature Matching and RANSAC.

Project 3: Recognition with Deep Learning.

Project 4: Semantic Segmentation.

Project 5: Classifying Point Clouds with PointNet.

Project 6: Neural Radiance Fields (NeRF).

All starter code and projects will be in Python with the use of various third party libraries. We will support MacOS and Windows. For MacOS, we are transitioning away from supporting Intel Processors. We will not support Linux (although things may just work out of the box). The course does not teach python and assumes you have enough familiarity with procedural programming languages to complete the projects.

Textbook

Readings will be assigned in **"Computer Vision: Algorithms and Applications, 2nd edition"** by Richard Szeliski. The book is available for purchase, but is also free to download. The materials from this class rely significantly on slides prepared by other instructors, especially Derek Hoiem and Svetlana Lazebnik. Each slide set and assignment contains acknowledgements. Feel free to use these slides for academic or research purposes, but please maintain all acknowledgements.

Comments, questions to **James Hays**.