Syllabus

Table of Contents

Course Description (top)

This course is a broad introduction to computer vision. Topics include camera models, multi-view geometry, reconstruction, some low-level image processing, and high-level vision tasks like image classification and object detection.

Here is a rough outline of topics and the number of lectures spent on each:

- Image formation / projective geometry / lighting *(3 lectures)*
- Practical linear algebra *(1 lecture)*
- Image processing / descriptors *(3 lectures)*
- Image warping *(2 lectures)*
- Linear models + optimization *(2 lectures)*
- Neural networks *(3 lectures)*
- Applications of neural networks *(3 lectures)*
- Motion and flow *(1 lectures)*
- Single-view geometry *(2 lectures)*
- Multi-view geometry *(3 lectures)*
- Applications *(2 lectures)*

Lectures (top)

Lectures will be delivered via Zoom, and recordings will be posted after each lecture. Lecture attendance is encouraged but not required.

You are free to come to either live lecture. While the registrar has set the course up as two sections, you are free to come to whichever lecture and work with whoever for homeworks and projects.

Only enrolled students may join the live lectures. The Zoom meeting ID and passcode are posted on Canvas; please don't share them with other students. If you are on the waitlist or otherwise interested to follow along with the course, you can watch the recorded videos that we post after each lecture.

Zoom Etiquette:

- Leave your camera and mic off unless you are speaking
- You will be muted by default
- To ask a question, you can either:
  - Type your question into the Zoom chat; we will actively monitor the chat and respond to questions
  - Use the "raise hand" functionality in Zoom; then we will unmute you to ask your questino

Office Hours (top)

Office hours are your time to ask questions one-on-one with course staff, and get clarification on concepts from the course. We encourage going to GSI office hours for implementation questions about the homework and faculty office hours for conceptual questions.

All office hours will be held virtually (tbd).

Prerequisites (top)

Some students may have had some prior exposure to computer vision, machine learning, or image processing, but none of these are required. We will assume you have a basic level of expertise in programming, computer science, mathematics (and specifically linear algebra).

Concretely, we will assume that you are familar with the following topics; they will not be reviewed in class:

- **Programming** including algorithms and data structures at the level of EECS 281.
- **Python**: All course assignments will involve programming in Python. You should either have prior experience with Python, or be willing to quickly learn a new language. One homework will help teach and reinforce some of the concepts of numerical programming in python.

It will also be extremely helpful for you to have background in the following topics. We will provide some refreshers of the necessary concepts as they arise, but we may not go through a comprehensive treatment of these topics:

- **Array Manipulation**: Homework assignments will involve manipulating multidimensional arrays using numpy and PyTorch. Some prior exposure to either of these frameworks will be useful, but if you've never used them before then the first homework assignment will help get you up to speed.
- **Linear Algebra**: In addition to basic matrix and vector operations, it will be good to know least squares, Eigen- and singular-value decompositions.
- **Calculus**: You should be comfortable with the chain rule, and taking gradients and partial derivatives of vector-valued functions.

Much of modern computer vision involves applying ideas from linear algebra to real-world data, and doing so often requires doing calculus with vector-valued functions. If you are unfamiliar with linear algebra or calculus, you should consider taking both: without these tools, you are likely to struggle with the course. Don't worry if you have seen these topics before but are rusty; we will provide some math refreshers of the necessary topics. These refreshers will be insufficient as a first introduction to these topics, but should serve to remind you of concepts that you've forgotten or that weren't covered in your previous courses.

Textbooks (top)

There are no required textbooks. However the following optional books may be useful, and we will provide suggested reading from these books to accompany some lectures:

- *Computer Vision: Algorithms and Applications* by Richard Szeliski. Available for free online.
- *Computer Vision: A Modern Approach (Second Edition)* by David Forsyth and Jean Ponce.
- *Elements of Statistical Learning* by Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Available for free online (Warning: Direct PDF link).
- *Multiple View Geometry in Computer Vision (Second Edition)* by Richard Hartley and Andrew Zisserman. Available for free online through the UM Library (Login required).

Grading Policy (top)

Your grade will be based on:

- **Homework Assignments (76%)**: There will be seven programming assignments over the semester. Homeworks 1-5 are each worth 12%; the first assignment (HW0) is shorter than the rest and is worth 6%; the last one is easier than the rest because you will be working on the project, and is worth 10%.
- **Course Project (24%)**: You will work together in groups of 3-5 students to produce a substantial project over the second half of the semester.

We will use the following base scheme to convert raw scores in the course into letter grades:

- A+: >= 98%
- A: >= 94%
- A-: >= 90%
- B+: >= 88%
- B: >= 84%
- B-: >= 80%
- C+: >= 78%
- C: >= 74%
- C-: >= 70%
- D+: >= 68%
- D: >= 64%
- D-: >= 60%
- F: < 60%

Depending on the distribution of raw scores at the end of the semester, we may apply a curve; however **the final curve will be no stricter than the base scheme**. For example, if you receive a raw score of 85% in the course, you will be guaranteed a letter grade of at least B; however you may receive a higher letter grade depending on the overall distribution of raw scores.

Course Communication (top)

The primary way to communicate with the course staff is through Piazza. If you have questions about course concepts, need help with homework, or have questions about course logistics, you should **post on Piazza instead of emailing course staff directly**. Since Piazza is a shared discussion forum, asking and answering questions there can benefit other students as well.

However **do not post solutions to homework assignments on Piazza**. If you have questions about a particular piece of code you have written for an assignment, then you should make a private post on Piazza.

We will also use Piazza to make announcments about the course, such as the release of new homework assignments, details about the midterm exam, or other logistical issues.

If you need to discuss a sensitive matter that you would prefer not be shared with the entire course staff then you can email the instructor directly. However the vast majority of questions about the course should be asked on Piazza.

Homework Policies (top)

Formatting and submission

We will not accept handwritten homework in any shape or form. You do not need to typeset your assignments in LaTeX or otherwise make them pretty. You can make a reasonable pdf in Google Docs, Microsoft Word, Open Office, or LaTex. If you'd like to try LaTex, try Overleaf.

Type it up in any format and submit a pdf to gradescope, marking the document up for where the questions are on gradescope. Stuff that doesn't follow this simple formatting requirement will get a *0 that you can fix by resubmitting it in the right format.*

Collaboration / Honor Code

This is a class with a lot more freedom and self-teaching compared to intro-sequence courses. You are **strongly encouraged to work on the homework in teams** (and you have to for the project anyway). The role of the teaching staff is to provide you guidance and background knowledge, but the learning comes from you engaging with the material. It's often useful to have peers along this journey.

- **With Students**: You should never know the specific implementation details of anyone else's homework or see their code. Working in teams and dispensing general advice about output or strategies is great and highly encouraged (e.g., "oh if the image is overly dark where it overlaps, you're doing averaging wrong"). But pair-programming is forbidden, as is sitting next to someone and debugging their code. We believe that the overwhelming majority of UM students are ethical and honorable. However, to change incentives for those who are less honorable, we will run data mining software periodically on the submitted homework.
- **Consulting Outside Material**: In this class, you're going to get some fairly vague specifications, potentially at the level of turn that 40-word description on a slide into a functioning program. You can, and should, turn to other documentation (the textbooks I suggest, other professors' lecture notes, other professors' slides, documentation from various libraries) What you may not do is read a bunch of actual code (pseudocode is fine however). If you come across code, close the window, but don't worry about it.

- **Things you should never worry about**: (some adapted from here): reading documentation of publicly available libraries; clarifying ambiguities and mistakes in assignments, slides, handouts, or textbooks, or documentation; discussing the general material; helping with specific class-independent stuff like cryptic numpy errors; discussing the assignments to better understand what's expected and general solution strategies; discussing the starter code; discussing general strategies for writing and debugging code.
- **If you're panicking and think copying will solve your issues**: Don't panic. We've tried to structure the incentives so that doing the right thing is rewarded. Submit it late (that's why we have late days and a sliding late-penalty). Submit it half-done (that's why we have partial credit). In the absolute worst case, don't submit it, and take a 0. In a year or two, you will not be worried about doing poorly on any one particular homework.

Late Policy

Stuff happens, so there's a generous late policy to account for the unexpected (e.g., food poisoning, a surprisingly difficult homework in this or another class, an interview, an alien abduction, etc, etc.).

- **Penalties**: Late homework will be penalized at a rate of 1% per hour, rounding to the nearest hour, ties away from zero. For example, if the homework is due Monday at 11:59:59pm, homework submitted until Tuesday 12:30am will be accepted with no penalty. After 12:30, there is a 1% penalty. If you submit it Wednesday 6:30am, there is a 30% penalty. Penalties are applied additively: for instance, if you got a 90 and you were 20 hours late, it is now a 70.
- **Late Hours**: Because unexpected things happen, you will have 72 late hours over the semester that you can use on homework. This will be done automatically using times recorded in gradescope. I will not keep track of lateness for you.
- **Course Project**: The project proposal is subject to the same late policy as the homework assignments. The 72 free late hours are shared between the homework assignments and the proposal. **We will not accept late work for the final project report and showcase video**.

Given this lenient policy, please **do not ask course staff to waive late deductions or give extra late days**, except for truly exceptional circumstances that merit it.

Regrade Policy

In a large class, it is likely that we are going to make a few mistakes. We are committed to making mistakes right. You are free to submit regrade requests, especially if you believe there is a serious error (although not all regrade requests will be approved).

- **Regrade Deadline**: Submit your regrade request within one week of the grades being released. This regrade deadline is firm. We will announce when grades are released. It is your responsibility to check it.
- **Small Judgment Calls**: Please do not quibble about small ambiguous decisions. Regrade requests for things that are off by ≤ 1 points on any particular problem, or which in total change the homework by ≤ 3 points will not be considered. If you are worried this will change your grade due to rounding, send an email to course staff titled `EECS 442W20 Minor Regrade Request` detailing your request and, at the end of the semester, if the points could change your grade, then we'll handle it then.
- **Administrative Fs**: If you submit handwritten homework or don't mark the pages in gradescope, you will get a 0. These are subject to the same deadlines as other regrades. Typeset your homework or mark the pages in gradescope and we'll happily grade them.

Course Project (top)

This is an opportunity to explore a topic in depth and should involve substantial work. This can be in implementation (e.g., implementing an existing algorithm), applications (e.g., applying computer vision to an existing problem), or research (e.g., trying something new in computer vision). Your project should amount to about two homeworks' worth of work per person.

If you cannot find a partner, there will be a piazza discussion for finding project partners. All written work should be in CVPR format. Overall, please remember that we do not see your hard work, we only see the products you deliver.

Concretely, the course project has the following deliverables:

Proposal (2 pages – 2%): The proposal should aim to explain what the problem is, why it's feasible to solve in the given timeline, and how you plan to achieve it. **While this counts for few points, this is important to get right.** In particular, your project proposal should explain:

1. What specific problem are you trying to solve? Are you appling computer vision, trying to extend computer vision, or implementing something? All are reasonable options. But please make it clear which you are trying to accomplish.
2. How are you going to solve the problem? You don't need to have everything already planned out, but you should have a clear idea. What parts are likely to work, and how can you still get work done if your plan fails.
3. What do you need that you don't currently have access to? This could be data, knowledge of certain machine learning techniques, or particular code.
4. What do you expect as an outcome and how will you identify and quantify success?

Final Project Report (4 pages, 14%): Please write up your project in the style of a CVPR paper (see samples here). Please remember to claim credit for everything you did and be clear what part of the project is yours and what part comes from others (e.g., did the data come from somewhere else? the initial implementation that you improved? a 3rd party library?). Your final report should explain:

1. The problem that you are trying to solve. Describe it concretely and explain an intuition about why it should be solvable.
2. Your approach. Please include diagrams – math is useful once the high-level idea is established, but diagrams will help us more quickly get up to speed.
3. Your results. Please explain not only the results but also why you ran the experiments you did. What do the experiments actually test? Please include both quantitative results (i.e., measuring numerically what's happening) as well as qualitative results (i.e., describing in words and images what's happening).
4. Details about your implementation. If you are building on the code of others, please indicate what is yours and what was others' work.

Virtual Showcase (8%) Instead of a poster presentation, you will participate in a virtual poster session. Most of this will be you a ~4 minute video of your group members presenting your project. This doesn't need to be fancy or overly produced; talking over a set of PowerPoint slides is perfectly acceptable.

The exact format and structure of the presentation can vary depending on the nature of your project. But you should clearly cover the following:

- What is the problem you were solving?
- Why is this problem important?

- How have people tried to solve this problem before? Give a brief summary of 1-3 key pieces of related work.
- How do you approach the problem? What is your key insight?
- What are your main experimental results? You don't have to go into all details, but you should mention your main experiments.
- What challenges remain? If you were to keep working on this project, what would you do in the future?

General Remarks (top)

- **Doing well in class**: You are highly encouraged to read the course document on doing well: computer vision is a relatively difficult subject and requires simultaneous mastery of linear algebra, programming, and converting relatively vague specs into fairly specific code. If you approach the assignments from the wrong angle, they will be incredibly burdensome.
- **Classroom etiquette**: Above all, please avoid being disruptive in class, for both the instructor and for your classmates.
- **Accommodations**: If you think that you need an accommodation for a learning or other disability, please let me know. We will work with the Office of Service for Students with Disabilities to make proper accommodations.
- **Counseling Center**: Staff at the Counseling Center are trained to help you deal with a wide range of issues, such as how to deal with exam-related stress and other academic and nonacademic issues. Services are free and confidential and do not impact student records.
- **Academic Integrity**: All students in the class are: (a) presumed to be decent and honorable; (b) bound by the College of Engineering Honor Code; and (c) expected to read, understand, and follow the honor code. Information about this can be found here. You are highly encouraged to discuss the course materials, assignments, and projects. You should read the discussion in the homework policy of what constitutes acceptable collaboration.

David Fouhey, Justin Johnson
fouhey@umich.edu,
justincj@umich.edu