CSE 559A: Computer Vision

Fall 2020: T-R: 11:30-12:50pm @ Wrighton 300 / Zoom

Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Course Staff: Adith Boloor, Patrick Williams

http://www.cse.wustl.edu/~ayan/courses/cse559a/

Sep 15, 2020
HOW TO ATTEND!

- Please open the live chat window when you’re attending a lecture synchronously.
HOW TO ATTEND!

FL2020.E81.CSE.559A.01

FL2020

Home

Assignments

Grades

Zoom

Live Chat

Piazza

FL2020.E81.CSE.559A.01 - Computer Vision

This course introduces the fundamentals of designing computer vision systems—that can “look at” images and videos and reason about the physical objects and scenes they represent. We will learn about methods for image restoration and enhancement; for estimating color, shape, geometry, and motion from images; and for image segmentation, recognition, and classification. The focus of the course will be on the mathematical tools and intuition underlying these methods: models for the physics and geometry of image formation, and statistical and machine learning-based techniques for inference.

Please visit the main course website:

https://www.cse.wustl.edu/~ayan/courses/cse559a/
FL2020.E81.CSE.559A.01 - Computer Vision

This course introduces the fundamentals of designing computer vision systems—that can “look at” images and videos and reason about the physical objects and scenes they represent. We will learn about methods for image restoration and enhancement; for estimating color, shape, geometry, and motion from images; and for image segmentation, recognition, and classification. The focus of the course will be on the mathematical tools and intuition underlying these methods: models for the physics and geometry of image formation, and statistical and machine learning-based techniques for inference.

Please visit the main course website:
https://www.cse.wustl.edu/~ayan/courses/cse559a/
HOW TO ATTEND!

FL2020.E81.CSE.559A.01 - Computer Vision

This course introduces the fundamentals of designing computer vision systems—that can "look at" images and videos and reason about the physical objects and scenes they represent. We will learn about methods for image restoration and enhancement; for estimating color, shape, geometry, and motion from images; and for image segmentation, recognition, and classification. The focus of the course will be on the mathematical tools and intuition underlying these methods: models for the physics and geometry of image formation, and statistical and machine learning-based techniques for inference.
HOW TO ATTEND!

FL2020.E81.CSE.559A.01 - Computer Vision

This course introduces the fundamentals of designing computer vision systems—that can "look at" images and videos and reason about
HOW TO ATTEND!

To visit the URL, click the button below.

Open in New Tab
HOW TO ATTEND!

Enter your name

John Doe

Start Chat
This is the live chat for CSE 559A. Please post your questions and answers here, and these will be forwarded to the instructor(s). Your messages will be visible only to yourself and the instructor(s), and not other students. Note that these chats will only be monitored during live sessions.
HOW TO ATTEND!

- Please open the live chat window when you’re attending a lecture synchronously.
- Keep yourself muted and your video off. Also, do not use Zoom Chat (we won’t be monitoring it).
- Use the live chat to send one-way private messages to me. Use to ask questions, and answer questions I’m asking.
- After the class, video recordings of the lecture will be available in Canvas -> Zoom -> Cloud Recordings.
- Slides will also be posted to the main course website (through links in the Syllabus / schedule section).

All set?
INTRODUCTION

What is Computer Vision?

Endow machines with the ability to make *sense*

of the physical world by looking at

images and videos
What is Computer Vision?

Endow machines with the ability to make sense of the physical world by looking at measurements of reflected light.
INTRODUCTION

[credit: http://www.blutsbrueder-design.com]
INTRODUCTION
INTRODUCTION

Recognize Objects
INTRODUCTION
Classify Scene
INTRODUCTION

Geometry / Layout
INTRODUCTION

Identify Materials

Foliage

Ceramic
INTRODUCTION

Surface Properties

Wet?

Slippery?
WHY IS THIS HARD?
WHY IS THIS HARD?
WHY IS THIS HARD?
WHY IS THIS HARD?
WHY IS THIS HARD?

Scale Ambiguity
WHY IS THIS HARD?

Scale Ambiguity
WHY IS THIS HARD?

Scale Ambiguity
WHY IS THIS HARD?
WHY IS THIS HARD?

Illuminant / Surface Color Ambiguity
WHY IS THIS HARD?
WHY IS THIS HARD?

Indirect Reflection
WHY IS THIS HARD?
WHY IS THIS HARD?

Observe Super-position of Multiple Paths
WHY IS THIS HARD?

Variation in Appearance
WHY IS THIS HARD?

Variation in Appearance
WHY IS THIS HARD?

Variation in Appearance
WHY IS THIS HARD?

make sense of the physical world
from measurements of reflected light
WHY IS THIS HARD?

make sense of the physical world
from measurements of reflected light

Seems hopeless .... except that
humans, animals, birds, insects are able to do it
WHY IS THIS HARD?

make sense of the physical world from measurements of reflected light

Seems hopeless .... except that humans, animals, birds, insects are able to do it
SO HOW DOES IT WORK?

By making assumptions about and exploiting structure in the natural world
SO HOW DOES IT WORK?

Broad Overview of (many a) Vision Algorithm
SO HOW DOES IT WORK?

Broad Overview of (many a) Vision Algorithm

1. Understand the Image Formation Model: Scene to Image

\[ I = \mathcal{F}(S) \]
SO HOW DOES IT WORK?

**Broad Overview of (many a) Vision Algorithm**

1. Understand the Image Formation Model: Scene to Image

\[ I = F(S) \]

2. Invert the Model: Gives us Multiple Physically Feasible Solutions

\[ \{S\} = F^{-1}(I) \]
SO HOW DOES IT WORK?

Broad Overview of (many a) Vision Algorithm

1. Understand the Image Formation Model: Scene to Image

\[ I = \mathcal{F}(S) \]

2. Invert the Model: Gives us Multiple Physically Feasible Solutions

\[ \{S\} = \mathcal{F}^{-1}(I) \]

3. Learn What Natural Scenes Look Like: Use to select likely scene among those that are feasible

\[ p(S) \]
SO HOW DOES IT WORK?

Broad Overview of (many a) Vision Algorithm

1. Understand the Image Formation Model: Scene to Image

\[ I = \mathcal{F}(S) \]

2. Invert the Model: Gives us Multiple Physically Feasible Solutions

\[ \{S\} = \mathcal{F}^{-1}(I) \]

3. Learn What Natural Scenes Look Like: Use to select likely scene among those that are feasible

So, we're just guessing?
SO HOW DOES IT WORK?

Broad Overview of (many a) Vision Algorithm

1. Understand the Image Formation Model: Scene to Image

\[ I = \mathcal{F}(S) \]

2. Invert the Model: Gives us Multiple Physically Feasible Solutions

\[ \{S\} = \mathcal{F}^{-1}(I) \]

3. Learn What Natural Scenes Look Like: Use to select likely scene among those that are feasible

So, we're just guessing?

Well, sort of
ILLUSIONS

Deliberate / Artistic
ILLUSIONS

Deliberate / Artistic

Natural (rare!)
The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition!!".

Slide via Lana Lazebnik
More Recently
HISTORY OF CV

"Simple" Pattern Recognition

Slide via Lana Lazebnik
HISTORY OF CV

Face Detection in Cameras
The Smile Shutter flow

Imagine a camera smart enough to catch every smile! In Smile Shutter Mode, your Cyber-shot® camera can automatically trip the shutter at just the right instant to catch the perfect expression.

Smile Detection in Cameras

Slide via Lana Lazebnik
HISTORY OF CV

Elementary Augmented Reality
HISTORY OF CV

“Statue of Liberty”  “Half Dome, Yosemite”  “Colosseum, Rome”

Flickr photos

3D model

Noah Snavely, Sameer Agarwal, Ian Simon, Steve Seitz, Richard Szeliski, Yasutaka Furukawa, Brian Curless

Slide via Noah Snavely
Face2Face: Real-time Face Capture and Reenactment of RGB Videos

Justus Thies 1  Michael Zollhöfer 2  Marc Stamminger 1  Christian Theobalt 2  Matthias Nießner 3

1 University of Erlangen-Nuremberg  2 Max Planck Institute for Informatics  3 Stanford University

Learning Representations for Automatic Colorization
Gustav Larsson  Michael Maire  Gregory Shakhnarovich
ECCV 2016
HISTORY OF CV

Paper

Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto and Jean-François Lalonde
Deep Outdoor Illumination Estimation
[arXiv pre-print] [BibTeX]
HISTORY OF CV

Scanning and Printing a 3D Portrait of President Barack Obama

Smithsonian Digitization Program Office and USC Institute for Creative Technologies

From Paul Debevec and Colleagues
HISTORY OF CV

Image-to-Image Translation with Conditional Adversarial Nets

Phillip Isola  Jun-Yan Zhu  Tinghui Zhou  Alexei A. Efros

University of California, Berkeley
In CVPR 2017

Day to Night
input  output

Edges to Photo
input  output
For Now, Self-Driving Cars Still Need Humans

By JOHN MARKOFF JAN. 17, 2016
Sponsor roster @ CVPR 2018
Sponsor roster @ CVPR 2018
Sponsor roster @ CVPR 2018
HISTORY OF CV

Sponsor roster @ CVPR 2018
Lots of Exciting Research & Potential for Real World Impact!
THIS COURSE

All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

SYLLABUS OVERVIEW

- Image formation, representation, and processing
- Low level vision
  - Photometric
  - Geometric
  - Motion
- Segmentation & Grouping
- High Level Vision (reasons with “semantic” knowledge)
  - Using deep convolutional neural networks

Slides will be posted on course website after class
PREREQUISITES

- Programming: problem sets will be in Python (but you can pick up Python+Numpy during the course).
- The course website (Resources section) links to refresher slides on the math background.
- Read through these slides
- If the material in them seems too unfamiliar, you might want to take courses on ProbStats / Linear Algebra before taking this class.
This course

All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

- **5 Problem Sets**
  - Math (answers to be typeset in LaTeX) and Programming (in Python)
  - 15% x 5 = 75% of your grade
  - To be done individually
  - Roughly every two weeks (see website)
  - **READ** collaboration and late policy
  - Submitted using git.

- **Final Project**
  - To be done individually
  - Open ended, mini research/implementation project
  - Choose topic (suggestions on course website), submit brief proposal, and get feedback from us
  - End of Term: Report = 25% of your grade

**NO EXAMS**
This Course

All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

Problem Sets: Late Policy

- Any problem set turned in more than 3 late days will simply not be graded.
- Problem sets turned in 1-3 days late will be penalized.
- Initially, you will receive your graded problem set without any penalty.
- Then, we will count your total number of late days across problem sets.
- You get 3 total free late days.
- We will decide which problem sets to apply those late days for to give you maximum credit.
- Beyond that, you will lose 25% of your grade on a problem set for each day late (after deducting the free late days).
- 1 Late Day = 1 min to 24 hours late!

Project Proposals and Reports can not be late
Academic Honesty

- All problem sets and projects MUST BE YOUR OWN work.
- Read through collaboration policy carefully. It is OK to discuss general concepts with your friends and classmates, or look for background knowledge online (like resources for numpy, linear algebra, etc.).
- But you can not seek or offer help for specific solutions to problem set questions.
- Do not share code or look for solutions online. PERIOD.
- Acknowledge any source of discussion or information in the problem sets (included in the homework solution template).

- ANY VIOLATIONS WILL RESULT IN FAILURE OF THE COURSE.

It does not matter how many points the violation affected. This will also be reported to the university and will quite likely attract stricter sanctions beyond the course.

- For final projects, you may use external libraries or provided source code: as long as you acknowledge them explicitly in your project report. You will be evaluated on your own contribution beyond these resources.
Academic Honesty

Go read the collaboration policy. That policy will be the deciding factor in any potential cases of academic dishonesty.

If you have any doubts, ask me on Piazza (with a private post).
THIS COURSE

All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

TODO: Testing Environment & Submission Using git

- Read Problem Set section
- Install Anaconda
- Setup a LaTeX environment, or familiarize yourself with one of the online LaTeX editors.
- Set up git. Read tutorials on public key authentication if you are not familiar.
- Generate a public-private key pair if you don’t have one. Set up your environment to use the private key to authenticate with git.
- Submit your public key using the link e-mailed to you.
- Then as soon as possible, try to complete Problem Set 0. This is a trivial problem set to test your setup (LaTeX, Anaconda, Git).
  - Ungraded, but mandatory!
TODO: Testing Environment & Submission Using git

About 10-20 minutes after you upload your public key, you should be able to clone the pset0 repository as:

git clone submit@cse559.ayanc.org:wustl.key/pset0

- **IMPORTANT**: Replace wustl.key with your actual WUSTL key username.
- If it asks you for a password, your public key authentication setup isn’t working.
- If the clone is successful, you will see a newly created pset0 directory. Look at the pset.pdf file inside the directory.
- Complete the homework (write and run the code, complete the report to create solution.pdf).
- Note that our writeup MUST be called solution.pdf. Do not give it a different name!
- Then add your code and solution.pdf files, commit & push to submit, and pull to verify submission.
THIS COURSE

All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

TODO: Testing Environment & Submission Using git

About an hour after you upload your public key:

```
$ git clone submit@cse559.ayanc.org:wustl.key/pset0
$ cd pset0/
```

Work on your problem set.

```
$ git add code/*.py solution.pdf
$ git commit -m 'submission'
```

- This DOES NOT complete your submission!
All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

### TODO: Testing Environment & Submission Using git

About an hour after you upload your public key:

```
$ git clone submit@cse559.ayanc.org:wustl.key/pset0
$ cd pset0/
```

Work on your problem set.

```
$ git add code/*.py solution.pdf
$ git commit -m 'submission'
$ git push
```

- You need to `push`. To determine whether your problem set is on time or late, we will look at when it was pushed, not committed!
- Verify that it was submitted properly

```
$ git pull
$ git log
```
THIS COURSE

All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

TODO: Testing Environment & Submission Using git

- Verify that it was submitted properly

  $ git pull
  $ git log

- This will show you a message with a timestamp and all the files included in your submission.
- Hold on to this repository. Once we have graded your submission, you can do a git pull to retrieve comments and feedback.
THIS COURSE

All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

• Sign up for Piazza!
  ■ Ask ALL course-related questions on piazza
    ○ Make private posts if you wish.
  ■ Answer others’ questions (subject to collaboration policy)

• Office Hours: TBD
THIS COURSE

All information @ http://www.cse.wustl.edu/~ayan/courses/cse559a/

- Read through course website
  - Syllabus
  - Problem Set and Resource Section
  - Late Policies
  - Collaboration & Academic Honesty Policies
- Join Piazza
- Review Pre-req Slides
- Install Anaconda, LaTeX, Git
- Submit your Public Key
- Submit Problem Set 0

Questions?
THE PINHOLE CAMERA

Image = Array of Numbers

Physical Object
THE PINHOLE CAMERA

Plane with Sensor Elements

Image = Array of Numbers

Physical Object
THE PINHOLE CAMERA

Plane with Sensor Elements

Image = Array of Numbers

Form Image on Sensor Plane

Physical Object
THE PINHOLE CAMERA

Record Intensity at each sensor Location

Plane with Sensor Elements

Form Image on Sensor Plane

Physical Object

Image = Array of Numbers
THE PINHOLE CAMERA

Sensor Plane

Object
THE PINHOLE CAMERA

Sensor Plane

Object

1D for simplicity
Why can't we get an image to form by just holding the sensor in front of the object?
Because the object isn't just reflecting in straight lines!
A point on sensor plane receives light from many distinct points on the object. No image since intensities "washed out".
THE PINHOLE CAMERA

Sensor Plane

Pinhole Camera

Object
THE PINHOLE CAMERA

Sensor Plane

Pinhole Camera

Object
THE PINHOLE CAMERA

Sensor Plane

Pinhole Camera

Object

Every point on the sensor plane corresponds to a unique ray
THE PINHOLE CAMERA

Sensor Plane

Object
THE PINHOLE CAMERA