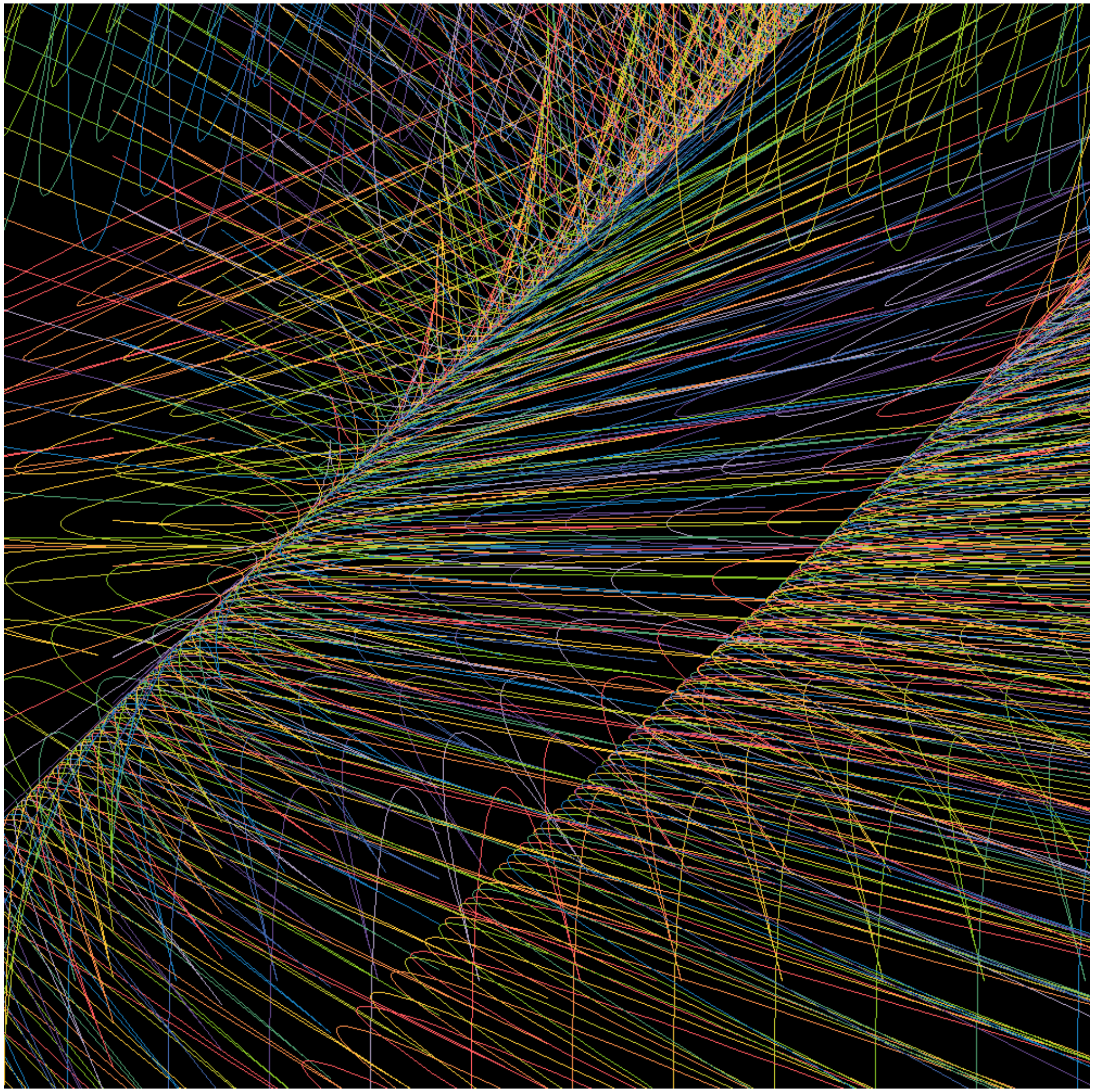
Conclusion

Announcements



Call Expressions

Problem Definition

From Discussion 0:

You can call:

- f(x): Subtracts one from x

- g(x): Doubles x

- h(x, y): Concatenates the digits of x and y.

What's the shortest expression using only f, g, h, and 5
that evaluates to 2024?

g(h(g(5),g(g(f(f(5)))))) has 7 calls

5➡️10 5➡️4➡️3➡️6➡️12

➡️1012➡️2024

A Computational Approach

```
def f(x):
    return x - 1
def g(x):
    return 2 * x
def h(x, y):
    return int(str(x) + str(y))

class Number:
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return str(self.value)

    def calls(self):
        return 0

class Call:
    """A call expression."""
    def __init__(self, f, operands):
        self.f = f
        self.operands = operands
        self.value = f(*[e.value for e in operands])

    def __str__(self):
        return f'{self.f.__name__}({",".join(map(str, self.operands))})'

    def calls(self):
        return 1 + sum(o.calls() for o in self.operands)
```

```
>>> n = Number(5)
>>> print(n)
5
>>> n.value
5
>>> Call(f, [n]).value
4

>>> h(g(f(5)), 5)
85
>>> c = Call(h, [Call(g, [Call(f, [n])]), n])
>>> print(c)
h(g(f(5)),5)
>>> c.value
85
>>> c.calls()
3
```

A Computational Approach

```
def f(x):
    return x - 1
def g(x):
    return 2 * x
def h(x, y):
    return int(str(x) + str(y))

class Number:
    def __init__(self, value):
        self.value = value

    def __str__(self):
        return str(self.value)

    def calls(self):
        return 0

class Call:
    """A call expression."""
    def __init__(self, f, operands):
        self.f = f
        self.operands = operands
        self.value = f(*[e.value for e in operands])

    def __str__(self):
        return f'{self.f.__name__}({",".join(map(str, self.operands))})'

    def calls(self):
        return 1 + sum(o.calls() for o in self.operands)
```

```
def smalls(n):
    "Yield all Calls and Numbers with n calls."
    if n == 0:
        yield Number(5)
    else:
        for operand in smalls(n-1):
            yield Call(f, [operand])
            yield Call(g, [operand])
        for k in range(n):
            for first in smalls(k):
                for second in smalls(n-k-1):
                    if first.value > 0 and second.value > 0:
                        yield Call(h, [first, second])

result = []
for i in range(8):
    result.extend([e for e in smalls(i) if e.value == 2024])
print(result[0])  # prints g(h(g(5),g(g(f(f(5))))))
```

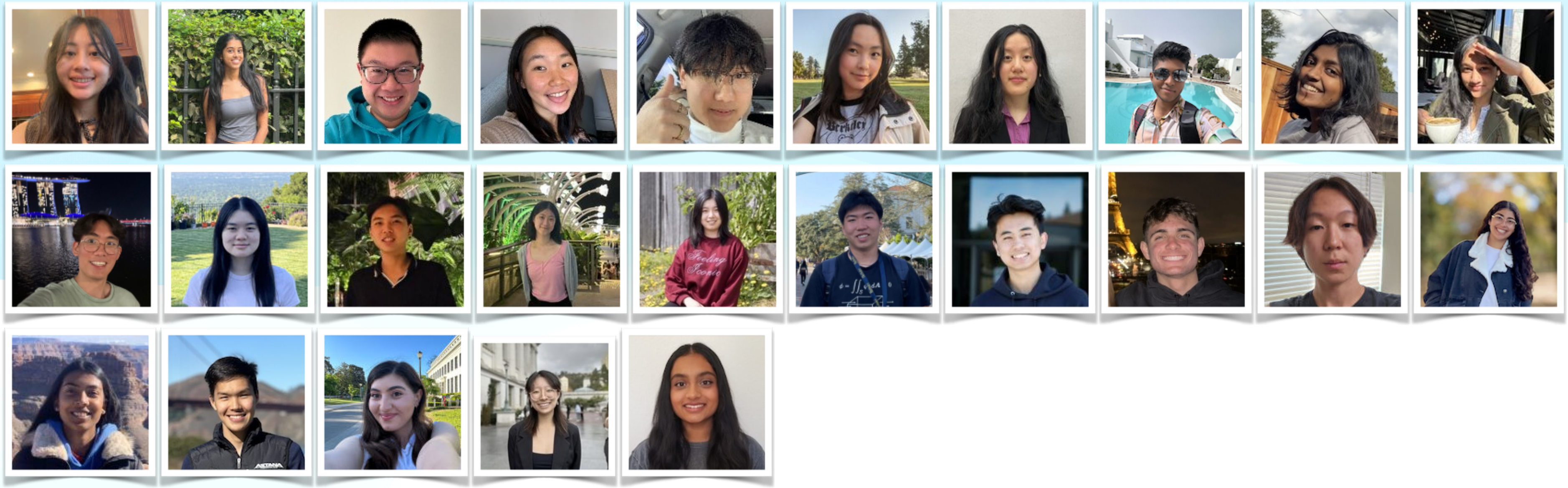
Course Staff

Thank you TAs!!!

We couldn’t have done it without you <3

Thank you tutors!!!

We couldn’t have done it without you <3



Thank you academic interns!!!

We couldn’t have done it without you <3

Akhil Korupolu, Alex Huang, Anuska Parajuli, Arjun Gonuguntla, Audrey Zhu, Baha Alkhatib, Dakota Daveed Feldkamp, Dylan Dang, Evan Woo, Haoyang Zhong, Isabella Hu, Jamie Jang, Jennifer Finkelstein, Karina Anders, Kye Lin, Laura Sophie Grimberg, Maile Caroline Frankwick, Maria Ma, Nathan Mun, Neil Chen, Olivia Guo, Owen Lam, Rushil Saraf, Sohum Phadke

So…why should I get involved?

- Teaching is, for lack of a better term - magical! Why? Here are a few reasons..
  - Supporting those that come after you. We're all in this together!
  - Meeting some of the coolest people that Berkeley has to offer :D
  - Refining your own technical understanding of course concepts
  - Autonomy to explore. The classroom is a mini-laboratory of sorts. Especially at Berkeley, course staff do a LOT (student support, writing infrastructure, iterating on the projects)

Ok, you’ve sold me. How do I start?

- Apply to be an Academic Intern and help in a lab section, mentored by a TA.
    - CS365 is a fairly new innovation that aims to standardize this experience.
- Apply directly to course staff! UCS1 (tutor) positions are a nice, gentle on-ramp to refine your pedagogy (the way you teach) mainly through office hours and small group tutoring. You *are* qualified.
    - If you want a quicker boost - teaching over the summer is a great way to jump directly into a UCS2 (teaching assistant) position. You get to teach your own section!
        - Applications for summer usually open mid-March.
        - More information here. You can join the EECS 101 EdStem for an announcement.
- Alternatively, Computer Science Mentors is a club on campus that also does small group tutoring! I got my start teaching in CSM!

How Did We Get Here?

Snapshot of Jedi’s Undergraduate Life

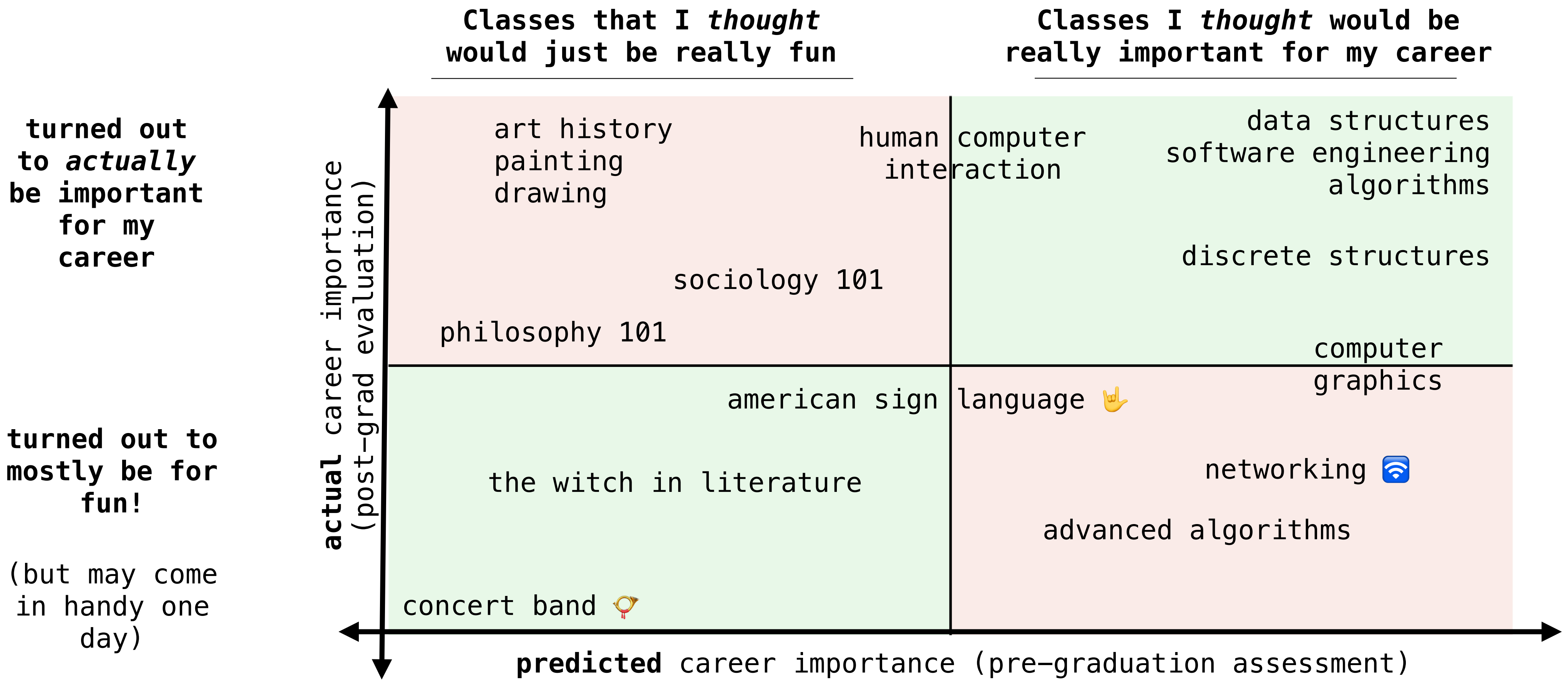
Junior Spring
COMPSCI 61C: Great Ideas of Computer Architecture
PBHLTH C160: Environmental Health and Development
POLSCI 103: Congress
POLSCI 171: California Politics
(and a unit for CSM and URAP)

(super senior) Fall
COMPSCI 170: Efficient Algorithms and Intractable Problems
MATH 54: Linear Algebra and Differential Equations
PE 1: Hip Hop Dance
PE 3: Intermediate Volleyball

Most of my time:
(First three years of undergrad): ASUC, City of Berkeley Commissioner,
student worker at Berkeley Law
(Last two years of undergrad): Teaching! (CSM, CS61B)

shm’s slide is a lot cooler. so i’m putting mine first >:D

Shmundergrad: the space of undergrad courses shm found valuable



Classes that I *thought* would just be really fun

Classes I *thought* would be really important for my career

turned out to *actually* be important for my career

turned out to mostly be for fun!

(but may come in handy one day)

actual career importance (post-grad evaluation)

predicted career importance (pre-graduation assessment)

art history
painting
drawing

human computer interaction

data structures
software engineering
algorithms

discrete structures

sociology 101

philosophy 101

computer graphics

american sign language 🤟

the witch in literature

networking 📶

advanced algorithms

concert band 📯

also super valuable & fun: research, tutoring, game design club, poetry club

Undergrad John

Intro courses galore: Philosophy, Linguistics, Economics, Computer Science, Math, etc.

The coolest thing (in my opinion): How people use and understand language to communicate

- **Linguistics** is the study the language and its use.
- **Philosophy** tries to answer questions about the world that the scientific method doesn't.
- **Computers** can carry out simulations of using and understanding language.

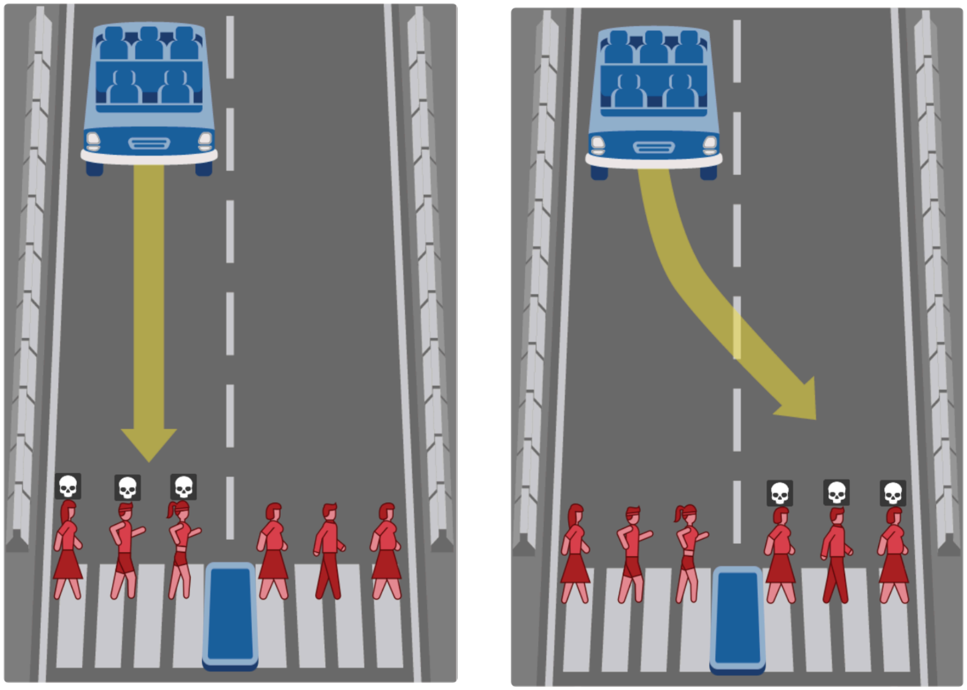
Another cool thing: How decisions are made

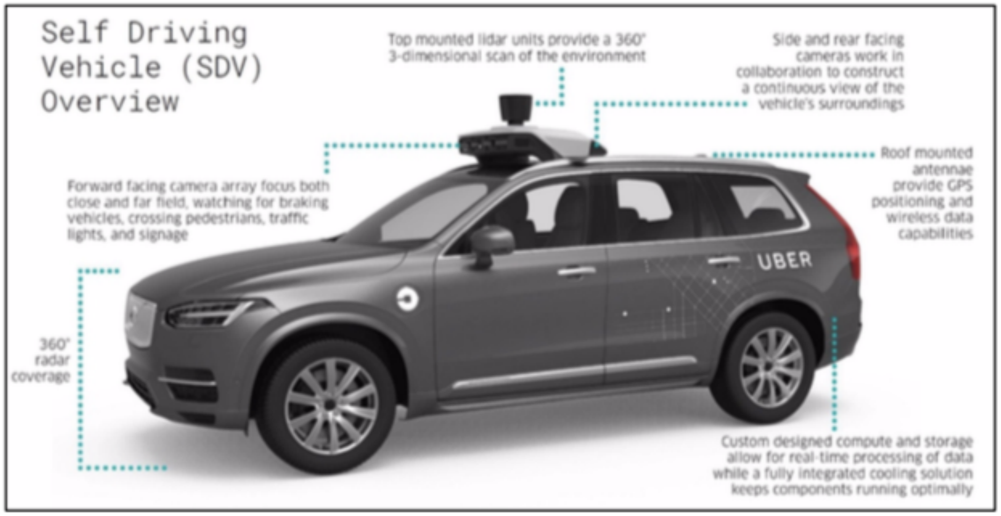
- **Economics** describes the individual & collective decisions of human beings.
- **Artificial Intelligence** implements automated decision making policies.
- **Probability** provides a language for making precise statements about uncertainty.
- **Literature** describes how complex people and their decisions really are.

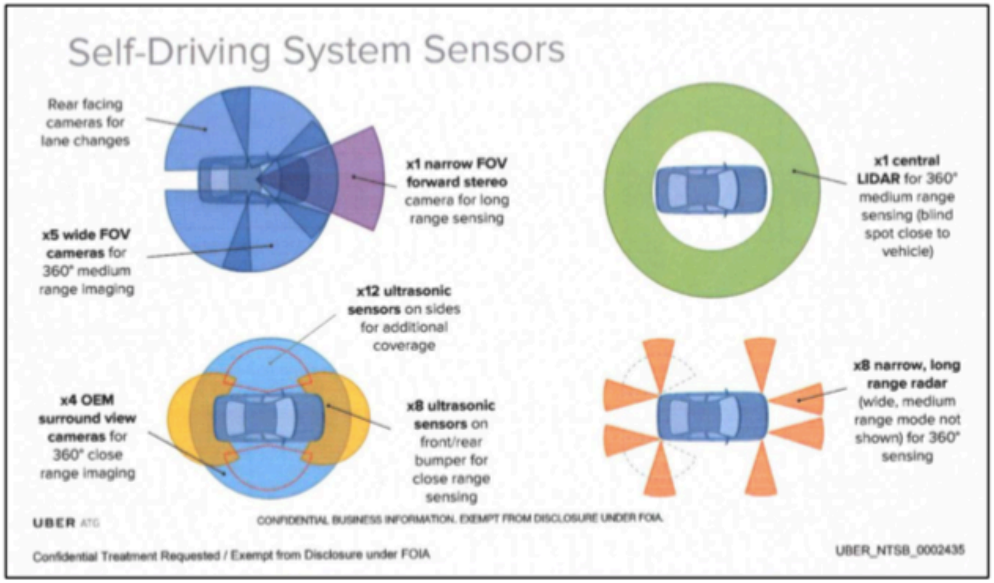
Society

Automated Decision Making

What should the self-driving car do?




Self Driving Vehicle (SDV) Overview
Top mounted lidar units provide a 360° 3-dimensional scan of the environment
Side and rear facing cameras work in collaboration to construct a continuous view of the vehicle's surroundings
Forward facing camera array focus both close and far field, watching for braking vehicles, crossing pedestrians, traffic lights, and signage
Roof mounted antennae provide GPS positioning and wireless data capabilities
360° radar coverage
UBER
Custom designed compute and storage allow for real-time processing of data while a fully integrated cooling solution keeps components running optimally


Self-Driving System Sensors
Rear facing cameras for lane changes
x1 narrow FOV forward stereo camera for long range sensing
x5 wide FOV cameras for 360° medium range imaging
x1 central LIDAR for 360° medium range sensing (blind spot close to vehicle)
x12 ultrasonic sensors on sides for additional coverage
x4 OEM surround view cameras for 360° close range imaging
x8 ultrasonic sensors on front/rear bumper for close range sensing
x8 narrow, long range radar (wide, medium range mode not shown) for 360° sensing
UBER ATG
CONFIDENTIAL BUSINESS INFORMATION. EXEMPT FROM DISCLOSURE UNDER FOIA.
Confidential Treatment Requested / Exempt from Disclosure under FOIA
UBER_NTSB_0002435

Life

That's all. Thanks!