Velocity in Research

CS 197 & 197C | Stanford University | Sean Liu & Lauren Gillespie
cs197.stanford.edu | cs197c.stanford.edu

Slides adapted from previous iterations of the course by Michael Bernstein
We Are CSE
Nadia Polikarpova
Velocity in Research

CS 197 & 197C | Stanford University | Sean Liu & Lauren Gillespie
cs197.stanford.edu | cs197c.stanford.edu

Slides adapted from previous iterations of the course by Michael Bernstein
Due Thurs 5/4, 10am:

**Assignment 4:** Progress Report II

Discuss milestone in advising meeting

Due Thurs 5/4, 10am:

**Assignment 3:** Introduction

Propose timeline for achieving milestone
Administrivia

197: Milestone — half-way checkpoint of your project

CAAs will scope out a milestone for your team

Expectations: 10 hrs / week / member

Goal: help pace your progress

No extensions on group assignments.
Administrivia

197: Each team member is expected to contribute equally

Team Dynamics Check-in Form (once mid-point, once at the end)

In the case of substantially imbalanced contribution, we will adjust participation grades to compensate

If your team is experiencing issues, please:

  Talk to each other first
  Inform your CA
Anonymous Feedback

Option 1: High-resolution course evaluation (polled by email)
Option 2: Go to course website; scroll to bottom

For problems with this site, contact the course staff email.
Week 4 Feedback

197: More regular check-ins?

Synchronously (1-2 per week):

Advising meeting

Schedule your team meetings during OH

Asynchronously: Email your CA (please follow communication guidelines)

HCI: 197-hci@cs.stanford.edu

AI + CompBio: 197-compbio@cs.stanford.edu

AI for sustainability: 197-sustain@cs.stanford.edu

If your team is experiencing issues, please let your CA know ASAP
Week 4 Feedback

197C:

Picking a research direction for purposes of the writing assignment

Writing is a **very** important skill you need as a researcher!

RW: help you think critically and propose a direction.

A direction is better than no direction.

Not set in stone! You’re welcome to change it later.

15 papers is time-consuming

Strategy: don’t read the full paper — practice skimming!

We can re-calibrate if necessary. Expectation: 10 hours per week on assignment or milestone
From last time…

You

Unknown Terrain
From last time…

You

Unknown Terrain
Vectoring

Unknown Terrain

You
What problem are we solving?

“Research is so much slower than industry.”

“I feel like we’re just not getting anywhere.”

“This keeps dragging on and it’s not working. I’m losing motivation.”

“I missed another submission deadline. I think my advisor is starting to lose faith.”
Bernstein theory of faculty success

To be a Stanford-tier faculty member, you need to master two skills that operate in a tight loop with one another:

**Vectoring**: identifying the biggest dimension of risk in your project right now

- not today!

**Velocity**: rapid reduction of risk in the chosen dimension

- today!
Today’s big idea: velocity

What is research velocity?
How do we achieve high velocity?
What other signals do people mistake for velocity?
What Is Velocity?
Problematic point of view

“Research is so much slower than industry.”

“I feel like we’re just not getting anywhere.”

“I missed another submission deadline.”

We’re not making enough progress.
What research is not

1. Figure out what to do.
2. Do it.
3. Publish.

What research is

Research is an iterative process of exploration, not a linear path from idea to result [Gowers 2000]
Michael’s diagnosis: The Swamp

“I have led and advised many projects at this point, and I can now say with certainty: nearly every project has a swamp.” — Michael Bernstein

The Swamp: challenges that get the project stuck for an extended length of time

Model not performing well

Design not having intended effect

Engineering challenges keep cropping up

& etc
No progress :(

You

Unknown Terrain
Swamps make progress a poor measure

Swamps can make a project appear to have no or little progress for an extended period of time.

However, swamps are when you need to be at your most creative. You need to try many different ideas, and rapidly, to orienteer your way out of a swamp.

The difference between an amazing and a merely good researcher: how effectively and rapidly you explore ways to escape the swamp.
Enter velocity

Drawn from theory and practice of rapid prototyping

Buxton, Sketching User Experiences
Schön, The Reflective Practitioner
Houde and Hill, What Do Prototypes Prototype?
CS 247 (cs247.stanford.edu)

“Enlightened trial and error succeeds over the planning of the lone genius.” - Tom Kelley
Velocity vs. progress

Progress is an absolute delta of your position from the last time we met. How far have you gotten?

Velocity is a measure of the **how much you’ve learned** in that time.

If you tried a ton of creative different ideas and they all failed…

that’s low progress but high velocity

I will be thrilled with you. You rock.
High velocity :)
Why is velocity a better measure?

Because we have likely learned a ton from the failures along the way.

Because we likely needed to experience those failures to eventually get to a success: you’re learning the landscape.

**Because the worst outcome is not failure, but tunneling unproductively.**

That’s low progress and low velocity

this is when I will be disappointed.
How do I achieve high velocity?
Restating our goal, precisely

We have a question to answer this week: Will our hunch work in a simple case? Is assumption X valid? Will this revised model overcome the problematic issue? Can we write a proof for the simple case? We’ve chosen this week’s question that we’re trying to answer carefully.

Velocity is the process of answering that question as rapidly as possible.

Choosing this question is the process of vectoring.
Approach: core vs. periphery

Achieving high velocity means sprinting to answer this week’s question, while minimizing all other desiderata for now. This means being clear with yourself on what you can ignore:

**Core**: the goal that needs to be achieved in order to answer the question

**Periphery**: the goals that can be faked, or assumed, or subsetted, or mocked in, so we can focus on the core.
Core-periphery mindset

The week’s goal is not a demo.

But this means working on everything both in the core and in the periphery.

The week’s goal is instead an answer to a question.

To answer a question, you don’t need to address all the issues in the periphery. Just focus on what’s in the core.
Core-periphery mindset

Your approach should be, necessarily, incomplete.

Be rapid. Be **ruthless**. Strip out or fake everything not required to answer the question.
Velocity

**Experimental design:** Make strong assumptions about everything that’s in the periphery

Use an easy or smaller subset of the data, make simplifying assumptions while working on your proof, ignore other nagging questions for the moment

**Research Engineering:** Do minimal engineering to answer your question (vector)

Find simple workarounds if original plan is taking long, mock in data
Social debugging: flash organizations

Problem: We had a problem of online workers not being as good as their Upwork profile suggested. We wanted workers who were experts at Angular, Django, UI, UX, marketing, etc, but often in practice they were not as good as they advertised.

We had a hunch that giving workers ~1hr starter tasks would allow us to vet them.

How do we test this hunch?
We picked a small number of domains and manually generated quick test tasks for them. We posted these as jobs, giving a time limit. We manually evaluated the results.

We didn’t care about generalizability or software integration.

Afterwards, we asked ourselves: could this scale to hundreds of people and tens of domains?
Problem: This project used multi-armed bandits to identify over several rounds of interaction whether teams should be flat or hierarchical, supportive or critical, etc. But we didn’t know: could these multi-armed bandits actually converge fast enough to be useful? We had a rough implementation of the multi-armed bandits, but it wasn’t production ready for interacting with teams.
We used a rough simulation! Assuming some roughly accurate numbers in how much each team benefited from each bandit setting, we generated teams and simulated the bandits over a few rounds.

The answer: they converged quickly enough that this might work!

(The next step: wizard of oz the interface, so we could test it “for real” without building integrating software.)
Velocity

Experimental design: Make strong assumptions about everything that’s in the periphery

Use an easy or smaller subset of the data, make simplifying assumptions while working on your proof, ignore other nagging questions for the moment

Research Engineering: Do minimal engineering to answer your question (vector)

Find simple workarounds if original plan is taking long, mock in data
Example: ML Experiment

AudioCLIP

Research question: Will this improve embeddings?

Example: ML Experiment

Example: ML Experiment

1. Download videos
2. Extract two frames
3. Integrate into image head
4. Retrain

Example: ML Experiment

1. Download videos
2. Extract two frames
3. Integrate into image head
4. Retrain

Problem:
2 weeks later, still on step 1

Example: ML Experiment

What is **core**?

What is **periphery**?

1. Download videos
2. Extract two frames
3. Integrate into image head
4. Retrain
Velocity: Research Engineering

**Core**: modify model to take in two images during training

**Periphery**: everything else

It’s easy to get distracted….

Remember: your goal is **not** to build a perfect, robust system. That’s the job of engineers, not researchers.

Your goal is to get a prototype working to answer your question!

Velocity: simplify ruthlessly
Example: ML Experiment

Priority: modify model to take in two images during training

1. Download videos
2. Extract two frames
3. Integrate into image head
4. Retrain
Example: ML Experiment

**Priority**: modify model to take in two images during training

1. Download videos

Mock two frames (e.g., all zeros)

3. Integrate into image head

4. Retrain
**Example: ML Experiment**

**Priority:** modify model to take in two images during training

Next priority: Two frames from video

1. One video
2. Two frames from a video
3. Integrate into image head
4. Retrain
Example: ML Experiment

**Priority**: modify model to take in two images during training

Next priority: Two frames from video

Next priority: Scrape videos

1. Download videos
2. Two frames from a video
3. Integrate into image head
4. Retrain
Example: ML Experiment

1. Download videos
2. Two frames from a video
3. Integrate into image head
4. Retrain

Can parallelize each stage among team members
Use mock data so no one has to wait
Core-periphery mindset

Answer questions, don’t engineer. This tends to rankle essentially every facet of your undergraduate training.

Too often, people pursue perfection in the first pass: perfect drafts, perfectly engineered software, perfect interaction design.

Remember: the goal is to answer the question, not to build that part of your system permanently (yet).
All together now

Each week, we engage in vectoring to identify the biggest unanswered question. This should be the focus of your velocity sprint for the week.

To hit high velocity, be strategic about stripping out all other dependencies, faking what you need to, etc., in order to answer the question.

Be prepared to iterate multiple times within the week!
Your turn

Pair up with someone not on your project.

5min each person: describe your project’s current state, the current question you’re trying answer. Brainstorm together how to increase velocity.

 Afterwards, we’ll share out.
A reminder: the algorithm

1. Articulate the question you’re answering.
2. Decide what’s absolutely core to answering that question.
3. Decide what’s peripheral.
4. Decide the level of fidelity that is absolutely necessary.
5. Go — but be open to reevaluating your assumptions as you go.
6. Loop with a new question.
Tips and tricks
“I’m being low velocity.”

Velocity = distance / time

So, if your velocity is low, you have two options:

1. **Cover more distance**: habits that can get you further in the same time (e.g., “try harder”, “be a better engineer”)
   
   You’re typically already maxed out on this.

2. **Decrease the time**: prototype more effectively

   WIN. Prototype more narrowly, lower your fidelity expectations (e.g., spit out any draft)
Checking email or InstaSnapFace?

This signals a lack of focus, and is a pretty certain predictor that you’re in a swamp.

It means you’re prototyping too broadly: you’re unfocused! focus your goal. Or you’re requiring too high a level of fidelity: you have unreasonable standards! lower your expectations.

Develop an internal velocity sensor, and as soon as you recognize this, apply one of the two rules.
Lowering standards: parallelism

Too often, we suffer from what’s known in the literature as **fixation**: being certain in an idea and pursuing it to the exclusion of all else. We cannot separate ego from artifact.

Instead, to answer the question, it’s often best to **explore multiple approaches in parallel**.

“While the quantity group was busily churning out piles of work—and learning from their mistakes—the quality group had sat theorizing about perfection, and in the end had little more to show for their efforts than grandiose theories and a pile of dead clay.”

— Bayles and Orland, 2001
Corollary 1: pivoting

Velocity is why cutting yourself off short and pivoting to a new project can be so dangerous in research.

Typically people pivot after a week in the swamp (the “fatal flaw fallacy”), rather than iterating with high velocity out of the swamp.

I promise that the project you pivot to will have a swamp too. Learn to increase velocity and prototype your way out of the swamp faster, instead of seeking out a swampless project.
Corollary 2: technical debt

Obviously, at some point you need to make sure you’re not too deep in technical debt, design debt, or writing debt.

But luckily, most people can only run their processors hot for a few hours a day. Everything I’ve described takes a lot out of you.

When you’re out of creative cycles, spend time maturing other parts of your project that are no longer open questions. Or, sometimes we reach a phase where we pause prototyping and focus on refinement and execution for a bit.
Why is velocity so important?
Great research requires high velocity

Don’t let 6-12 month paper deadlines obscure the velocity at which research needs to move in order to succeed.

If you want to achieve a high impact idea, you need to try a lot of approaches and refine and fail a lot. You want to do that as quickly as possible.

If you can prototype and learn and fail 5x as quickly as the next person, you will be able to achieve far more risky and impactful research.
Takeaways, in brief
l) The swamp is real, and it slows visible progress.
2) Velocity is a far better measure of yourself than progress, and it’s something you actually have control over.
3) Achieve high velocity by being clear what question you’re answering, and focusing ruthlessly on the core of that question while stripping out the periphery.
4) If you’re low velocity, velocity = distance / time. Either increase distance (rarely possible) or decrease time (often possible: you’re too broad or too perfectionist).
And finally...

Get into your project groups and discuss your strategy for velocity. What’s working? What can be improved?
Due **next** Thurs 5/11: Progress Report III

This week’s vector

What is core and periphery?

This week’s plan

This week’s result

Next week’s vector

Next week’s plan
Next Thurs 5/11: no written assignment due

Work towards your research milestone!
Reminder:
Submit your attendance on Canvas!
Velocity in Research

Slide content shareable under a Creative Commons Attribution-NonCommercial 4.0 International License.