STUDENT STARTUP CENTER (SSC) × IEEE STUDENT BRANCH

SKILLBUILDER SERIES · SESSION 2 OF 2

Vibecoding 2

Computer Vision Applications

Vibe-code real-time detection apps with Antigravity + YOLO

Dr. Alfredo Costilla Reyes · UC Davis · April 29, 2026

Builds on Vibecoding 1 | Python | OpenCV | YOLO v8

Recap — What We Learned in Vibecoding 1

In Session 1 you built:

- **Antigravity IDE** — AI-powered coding with Gemini
- **Vibecoding workflow** — describe → AI generates → review → run → git commit
- **React + Tailwind app** — Tinder-style matching app built with prompts

KEY PRINCIPLE

You **describe outcomes**, AI writes code, you **steer with prompts**. Local git keeps every version safe.

Today we go deeper:

VIBECODING 2 STACK

Camera / Video Input

↓

YOLO v8 — Object Detection

↓

Your Custom Logic (vibe-coded)

↓

Save with Git

Today's Roadmap

- **0:00** **Recap + Setup** — environment, Python
- **0:10** **Part 1 — CV Concepts** — YOLO explained
- **0:20** **Part 2 — First Detection** — YOLO on webcam
- **0:35** **Part 3 — Build Your Project** — vibe-code it
- **0:55** **Demo + Share** — commit with git

WHAT YOU NEED

Laptop from Session 1

Antigravity IDE (desktop app)

Python 3.10+ installed

Or Google Colab (browser)

Webcam (built-in is fine)

No webcam? We have extras available! You can also use a video file or images.

Step 0 — Environment Setup

Run these in your terminal / Antigravity terminal:

```
# 1. Verify Python version
$ python3 --version
Python 3.10.x or higher ✓
# 2. Create project folder + clone your repo
$ mkdir -p ~/vibecoding-cv && cd ~/vibecoding-cv
# Or open your existing project folder
# 3. Install ultralytics (YOLO + OpenCV bundled)
$ pip install ultralytics
# 4. Verify install
$ yolo --version
Ultralytics 8.x.x ✓
```

CHECKPOINT

- ✓ `python3 --version` shows 3.10+
- ✓ `yolo --version` returns 8.x
- ✓ Your project folder is open in Antigravity

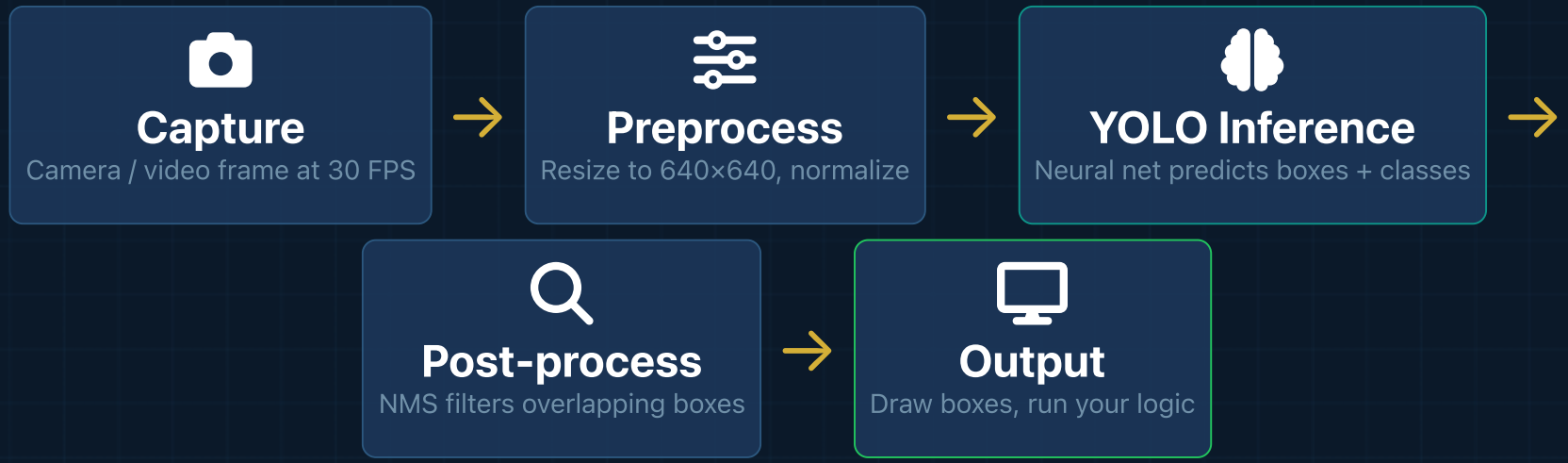
⚠ **pip not found?** Try `pip3` or `python3 -m pip`

⚠ **No Python?** Use Google Colab — open colab.research.google.com and run the same commands with `!` prefix

COLAB ALTERNATIVE

Everything works in the browser — no install needed. We will demo both paths.

Part 1 — Computer Vision Pipeline



WHAT YOLO DETECTS

80 classes: person, car, phone, laptop, bottle, chair, dog, cat...
Can **fine-tune** for custom objects

SPEED ON A LAPTOP

yolov8n — ~30–60 FPS
yolov8s — ~20–40 FPS
yolov8m — ~10–20 FPS

OUTPUT PER DETECTION

Box x,y,w,h · **Class** "person"
Confidence 0–1.0
Track ID (with tracker)

Part 2 — YOLO on Webcam in 5 Minutes

Run this in your terminal:

```
# Quickest way — one CLI command
$ yolo detect predict model=yolov8n.pt source=0
show=True
# source=0 → first webcam, source=1 → second
# Press Q to quit the preview window
```

Or use Python (easier to customize):

```
from ultralytics import YOLO
import cv2
model = YOLO("yolov8n.pt") # downloads auto
cap = cv2.VideoCapture(0) # webcam
while cap.isOpened():
  ret, frame = cap.read()
  if not ret: break
  results = model(frame, verbose=False)
  annotated = results[0].plot()
  cv2.imshow("YOLO", annotated)
```

WHAT TO EXPECT

- ✓ Window opens with webcam feed
- ✓ Boxes drawn around detected objects
- ✓ Labels like `person 0.97` appear
- ✓ Real-time at 20–60 FPS

💡 Point camera at yourself — you'll see **person** detected. Show your phone — **cell phone**. Show a bottle — **bottle**.

⚠ **Mac + camera permission:** System Settings → Privacy → Camera → allow your terminal app

Vibecoding a CV App — Antigravity + YOLO

HOW TO PROMPT ANTIGRAVITY FOR CV CODE

Paste your base detection script into Antigravity, then ask:

- *"Add a person counter — display total count on screen"*
- *"Alert me when more than 3 people are detected"*
- *"Only draw boxes around people, ignore others"*
- *"Log detections with timestamps to a CSV file"*

</> READING YOLO RESULTS IN PYTHON

```
results = model(frame)
for r in results:
  for box in r.boxes:
    cls = int(box.cls[0])
    conf = float(box.conf[0])
    name = model.names[cls]
    x1,y1,x2,y2 = box.xyxy[0]
    # your logic here
```

💡 Paste this into Antigravity and say: *"Explain each line, then modify it to [your goal]"*

Part 3 — Pick Your CV Project

Push-Up Counter
Track body position across frames — count reps automatically using pose estimation

yolov8n-pose

People Counter
Count people in frame, log entries/exits via virtual line crossing

yolov8n

Pose Detector
Detect body keypoints — classify standing, sitting, hands-up poses

yolov8n-pose

Safety Compliance
Detect if a person is wearing a hard hat or vest — alert if missing

custom model

Object Classifier
Identify objects on a surface — count, log to CSV, trigger alerts

yolov8n

Your Idea
Anything that needs a camera + AI. Describe it to Antigravity and build

open

💡 Not sure which to pick? Start with **People Counter** — it is the easiest to verify works (just walk in front of your camera).

Hands-on Build — Step by Step

1. **Open your repo in Antigravity**
   Create a new file: `cv_app.py`
2. **Paste the base script** (from Slide 6)
   Get it running with just bounding boxes first
3. **Open Antigravity Chat** (`Ctrl+Shift+I`)
   Describe your project goal in detail
4. **Apply the AI suggestion** — run and test
   Does it do what you want? Iterate with follow-up prompts
5. **Commit when it works**

```
$ git add . && git commit -m "add people counter"
$ git push
```

cv2 not found? Run `pip install opencv-python-headless`

20-MIN SPRINT

- 5m Get base detection running
- 10m Add custom logic via prompts
- 5m README + git commit

DONE WHEN:

- ✓ App runs on your webcam
- ✓ Custom logic works
- ✓ Committed to git with README

Going Further — Advanced Prompts

Level up your app with these prompts:

LOGGING

"Save detections with timestamp and confidence to detections.csv"

ALERTS

"Alert when 5+ people are detected simultaneously"

VISUALIZATION

"Add a bar chart showing real-time counts of top 5 classes"

TRACKING

"Use ByteTrack to give each person a persistent ID across frames"

YOLO MODEL CHEAT SHEET

`yolov8n.pt`	Nano — fastest
`yolov8s.pt`	Small — balanced
`yolov8n-pose.pt`	Pose estimation
`yolov8n-seg.pt`	Segmentation masks
`yolov8n-obb.pt`	Oriented boxes

Full docs: docs.ultralytics.com

80 COCO classes list:
github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/coco.yaml

Push to GitHub + Write a Great README

Final steps:

1. Add a `requirements.txt`

```
$ pip freeze > requirements.txt
```

2. Update your README — use Antigravity:

"Write a README.md for my people counter app — include what it does, how to run, and requirements"

3. Commit and push everything:

```
$ git add .
$ git commit -m "final cv app with readme"
$ git push
```

4. Add a demo GIF to your README

Use `ffmpeg` or online converter — shows your app live

README TEMPLATE (AI-PROMPT THIS)

```
# CV App Name
## What it does
One sentence description
## Demo
![demo](demo.gif)
## Installation
pip install ultralytics
## Usage
python cv_app.py
## Built with
Python + OpenCV · YOLO v8 · Antigravity IDE
```

Demo Time — Show Your App!

Share your screen — 60 seconds each

1. Run your app — show the detection live
2. Show your GitHub repo — commits, README, code
3. Tell us: one Antigravity prompt that saved you the most time

WHAT MAKES A GREAT DEMO

- Clear video window showing detection boxes
- Your custom logic visible (counter, alert, log)
- GitHub repo is clean and has a README

You built a real AI app.

With a real-time computer vision pipeline. That runs on your machine. And lives on GitHub. In under an hour.

Computer Vision | YOLO v8 | GitHub | Antigravity

What's Next

KEEP BUILDING WITH THIS STACK

- **Train a custom YOLO model** — label your own images on Roboflow, train in Colab
- **Deploy to Raspberry Pi 5** — edge inference (see Vibecoding Session 3)
- **Add a web dashboard** — Flask + WebSockets for real-time detection feed in browser
- **ENG 108 at UC Davis** — full course on embedded AI and computer vision

RESOURCES

- docs.ultralytics.com
- roboflow.com — datasets + labeling
- colab.research.google.com — free GPU
- antigravity.google

🎓 ENG 108: EMBEDDED AI

Full course — take the skills from this workshop to production:

- Raspberry Pi + camera hardware
- Custom YOLO model training
- Edge deployment + optimization
- Real team projects

ENG 8 prerequisite waiver available — ask Dr. Alfredo after this session!

SkillBuilder Series

✓ Complete	✓ Today	Coming
Session 1	**Session 2**	**Session 3**
Antigravity IDE	Computer Vision	Edge + Pi Deploy

🏁 Wrap-up

Across both sessions you learned:

- ✅ Vibecoding — AI-first development mindset
- ✅ GitHub — version control + portfolio
- ✅ Antigravity IDE — prompt-driven code generation
- ✅ YOLO v8 — real-time object detection
- ✅ Built and shipped a full computer vision app

KEEP IN TOUCH

- Follow your classmates on GitHub
- acostillareyes@ucdavis.edu — ENG 108 info
- @ieee.ucdavis — tag us when you ship!

You can see with AI now.

Every camera is a sensor. Every frame is data. You now know how to make machines understand what they see.

Questions? Find us after the session

Dr. Alfredo Costilla Reyes · SSC × IEEE Student Branch

SkillBuilder Series · SSC × IEEE Student Branch · April 29, 2026