

# **Powering-Up My Pole Vault Technique**

Alex Epstein - 5/21/25 - Senior Project Independent Study



.....

# Why This Project?

## Build Coding Skills

- My college major
- Love of coding

## Combining Passions

- Pole vault
- Coding
- Math/physics/science

## Future Impact

- Can build on the code
- Others can build on and improve it
- Can help new & advanced pole vaulters





# Objective 1:

>>>>

Researching the  
Biomechanics of Pole Vault

.....





# What did I do?

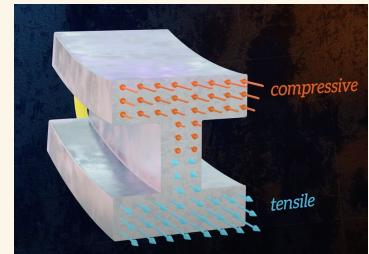
## Research Papers

- The Biomechanics of Pole Vault
- A New Way of Looking at the Biomechanics of the Pole Vault

>>>>

## Videos

- AI Golf Swing Analyzer - What Can We Learn
- An Introduction to Stress and Strain
- The meaning of the dot product | Linear algebra makes sense



## Articles

- Estimating 3D Poses of Athletes at Live Sporting Events, NY Times



.....



# What did I learn?

- Necessary math concepts
- How past AI sports devices work
- Physics equations for modeling a jump
- The different phases of a pole vault



>>>>

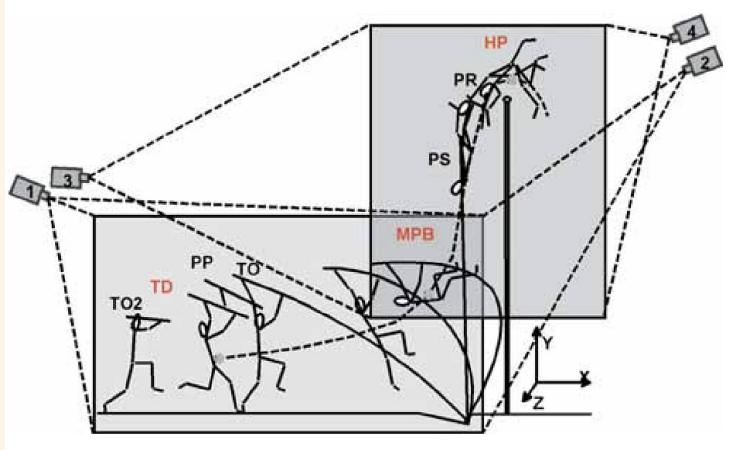
Partial derivatives

$$f(x, y) = x^2 y + \sin(y)$$
$$\frac{\partial f}{\partial x}(1, 2) = \frac{\partial}{\partial x} (x^2 y + \sin(y)) = 2x y + 0$$
$$\frac{\partial f}{\partial y}(1, 2) = \frac{\partial}{\partial y} (x^2 y + \sin(y)) = x^2 + \cos(y)$$



.....

# What did I learn?

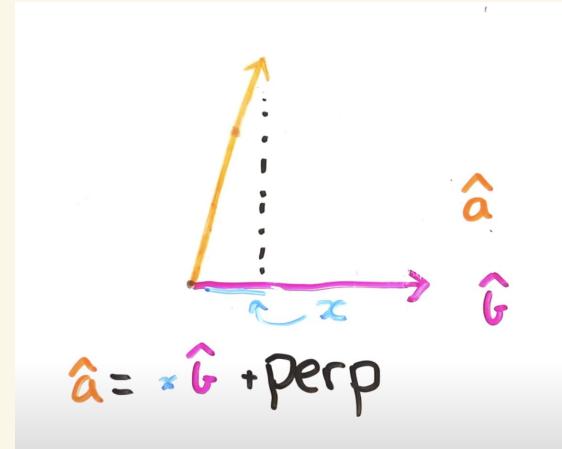


Many phases make up  
a pole vault jump



$$E_{tot} = \sum_{i=1}^{12} m_i g h_i + \sum_{i=1}^{12} \frac{m_i \mathbf{v}_i}{?}$$

COE equation



Dot  
product

.....

>>>



# Objective 2:

>>>>

To learn AI and machine  
learning video recognition

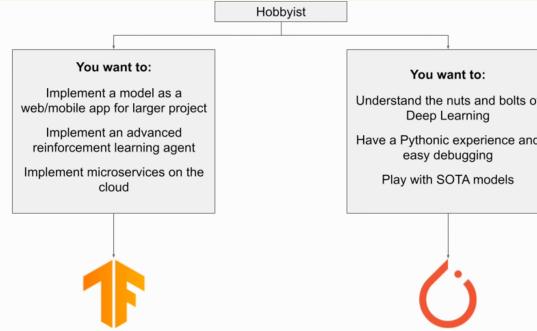
.....





# What did I do?

- Researched pose estimation libraries
- Started a mini-project analyzing tennis swings
- Debugged...intensely...
- Touched up on related coding concepts



```
from robotflow import Robotflow
r = Robotflow(api_key='your-api-key-here')
project = r.get_project('your-project-name').project('tennis-ball-detection')
version = project.version_id
dataset = version.dataset('your-dataset')
```

```
File "/usr/lib/python3.11/site-packages/robotflow/api/client.py", line 289, in __init__
    self._session = session or Session()
  File "/usr/lib/python3.11/site-packages/requests/sessions.py", line 518, in __init__
    transport = transport or HTTPAdapter()
  File "/usr/lib/python3.11/site-packages/requests/adapters.py", line 100, in __init__
    self._socket_pool = ConnectionPool(
  File "/usr/lib/python3.11/site-packages/urllib3/poolmanager.py", line 190, in __init__
    self._connection_class = connection_class or HTTPConnection
  File "/usr/lib/python3.11/site-packages/urllib3/connectionpool.py", line 65, in __init__
    self._ssl_context = ssl.create_default_context()
  File "/usr/lib/python3.11/ssl.py", line 1000, in create_default_context
    raise ValueError("SSLContext must be initialized with a backend")
ValueError: SSLContext must be initialized with a backend
```

Python



# What did I do?



>>>>



.....

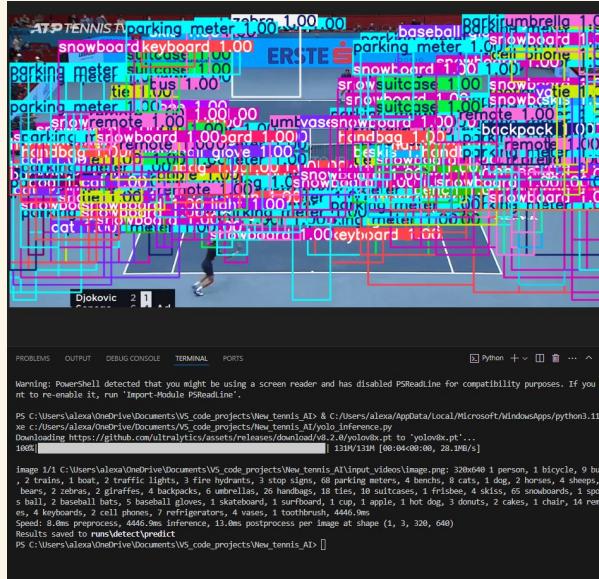


# What did I learn?

- How to apply machine learning to sports
- Working with YOLO v8
- Isolating certain elements
- Debug and manage coding frustration



>>>>



.....

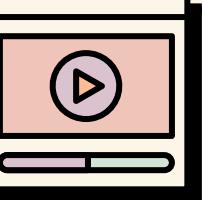


# Objective 4:

>>>>> To develop and train an AI-powered  
pole vault technique analyzer  
application

.....

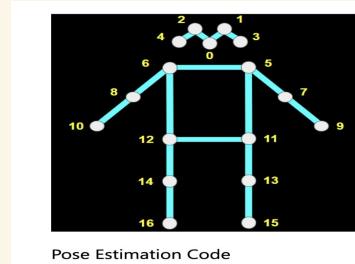
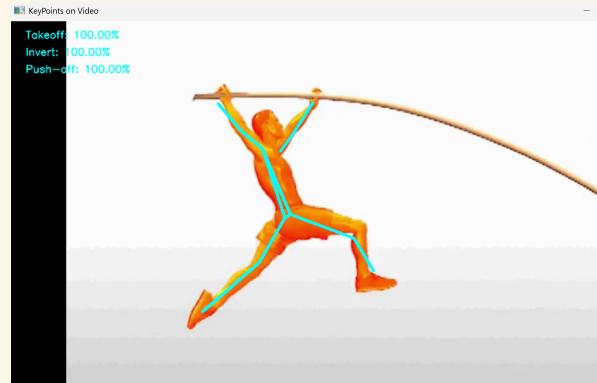




# What did I do?

>>>>

- Developed an algorithm to track a pole vault jump
- Integrated the pose estimator with OpenCV for visual feedback
- Tracked key joint angles during the vault
- Addressed issues with fluctuating keypoints and missing detections
- Scored each phase of the vault based on confidence levels



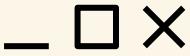


# What did I do?



The screenshot shows a Python code editor with a file named `pose-estimation.py`. The code implements a pose estimation pipeline, including loading a video, estimating poses, and visualizing angles over time. A separate window titled "KeyPoints on Video" displays a basketball player in a yellow jersey performing a free throw. A cyan line highlights the player's right leg and foot, likely indicating a specific joint or keypoint being tracked.

```
File Edit Selection View Go Run Terminal Help
pose-estimation.py > 41 Senior Project
pose-estimation.py > 42 PoseEstimation > 43 analyze_pose
43     def analyze_pose(self, show_angles=False):
44         cap = cv2.VideoCapture('lebron.mkv')
45         pe = PoseEstimation()
46         pe.analyze_pose(show_angles=show_angles)
47
48         if __name__ == '__main__':
49             run_analyze_pose(show_angles=False)
50             # run_analyze_pose(show_angles=True)
51
52
53     def run_analyze_pose(show_angles):
54         pe = PoseEstimation('lebron.mkv')
55         pe.analyze_pose(show_angles=show_angles)
56
57         if __name__ == '__main__':
58             run_analyze_pose(show_angles=False)
59             # run_analyze_pose(show_angles=True)
60
61
62     def _compute_angle(keypoints):
63         angle = np.degrees(np.arctan2(
64             keypoints[1][1] - keypoints[0][1],
65             keypoints[1][0] - keypoints[0][0]))
66
67         if angle < 0:
68             angle += 360
69
70         return angle
71
72     def _append_angle(keypoints):
73         angle = self._compute_angle(keypoints)
74         self.angles.append(angle)
75
76     def _append_time():
77         time.append(frame_count)
78
79     def _plot_angles():
80         ax = plt.gca()
81         ax.plot(time, angles, marker='o', color='orange')
82         plt.xlabel('Time')
83         plt.ylabel('Angle (degrees)')
84         plt.title('Angle vs. Time')
85         plt.draw()
86         plt.pause(.05)
87
88     def _plot_text(frame, angle):
89         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
90
91     def _draw_keypoints(frame, keypoints):
92         for i in range(1, len(keypoints)):
93             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
94             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
95
96     def _show_angles(frame, keypoints):
97         for i in range(1, len(keypoints)):
98             angle = self._compute_angle(keypoints[i])
99             self._append_angle(angle)
100
101             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
102
103             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
104             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
105
106             if show_angles:
107                 angle = self._compute_angle(keypoints[i])
108                 self._append_angle(angle)
109
110                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
111
112                 cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
113                 cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
114
115             cv2.imshow("KeyPoints on Video", frame)
116
117             if cv2.waitKey(1) & 0xFF == ord('q'):
118                 break
119
120         cap.release()
121         cv2.destroyAllWindows()
122
123
124     def _analyze_pose(self, show_angles):
125         pe = PoseEstimation('lebron.mkv')
126         pe.analyze_pose(show_angles=show_angles)
127
128
129     def run_analyze_pose(show_angles):
130         pe = PoseEstimation('lebron.mkv')
131         pe.analyze_pose(show_angles=show_angles)
132
133
134     def _compute_angle(keypoints):
135         angle = np.degrees(np.arctan2(
136             keypoints[1][1] - keypoints[0][1],
137             keypoints[1][0] - keypoints[0][0]))
138
139         if angle < 0:
140             angle += 360
141
142         return angle
143
144     def _append_angle(keypoints):
145         angle = self._compute_angle(keypoints)
146         self.angles.append(angle)
147
148     def _append_time():
149         time.append(frame_count)
150
151     def _plot_angles():
152         ax = plt.gca()
153         ax.plot(time, angles, marker='o', color='orange')
154         plt.xlabel('Time')
155         plt.ylabel('Angle (degrees)')
156         plt.title('Angle vs. Time')
157         plt.draw()
158         plt.pause(.05)
159
160     def _plot_text(frame, angle):
161         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
162
163     def _draw_keypoints(frame, keypoints):
164         for i in range(1, len(keypoints)):
165             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
166             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
167
168     def _show_angles(frame, keypoints):
169         for i in range(1, len(keypoints)):
170             angle = self._compute_angle(keypoints[i])
171             self._append_angle(angle)
172
173             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
174
175             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
176             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
177
178             if show_angles:
179                 angle = self._compute_angle(keypoints[i])
180                 self._append_angle(angle)
181
182                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
183
184             cv2.imshow("KeyPoints on Video", frame)
185
186             if cv2.waitKey(1) & 0xFF == ord('q'):
187                 break
188
189         cap.release()
190         cv2.destroyAllWindows()
191
192
193     def _analyze_pose(self, show_angles):
194         pe = PoseEstimation('lebron.mkv')
195         pe.analyze_pose(show_angles=show_angles)
196
197
198     def run_analyze_pose(show_angles):
199         pe = PoseEstimation('lebron.mkv')
200         pe.analyze_pose(show_angles=show_angles)
201
202
203     def _compute_angle(keypoints):
204         angle = np.degrees(np.arctan2(
205             keypoints[1][1] - keypoints[0][1],
206             keypoints[1][0] - keypoints[0][0]))
207
208         if angle < 0:
209             angle += 360
210
211         return angle
212
213     def _append_angle(keypoints):
214         angle = self._compute_angle(keypoints)
215         self.angles.append(angle)
216
217     def _append_time():
218         time.append(frame_count)
219
220     def _plot_angles():
221         ax = plt.gca()
222         ax.plot(time, angles, marker='o', color='orange')
223         plt.xlabel('Time')
224         plt.ylabel('Angle (degrees)')
225         plt.title('Angle vs. Time')
226         plt.draw()
227         plt.pause(.05)
228
229     def _plot_text(frame, angle):
230         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
231
232     def _draw_keypoints(frame, keypoints):
233         for i in range(1, len(keypoints)):
234             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
235             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
236
237     def _show_angles(frame, keypoints):
238         for i in range(1, len(keypoints)):
239             angle = self._compute_angle(keypoints[i])
240             self._append_angle(angle)
241
242             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
243
244             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
245             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
246
247             if show_angles:
248                 angle = self._compute_angle(keypoints[i])
249                 self._append_angle(angle)
250
251                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
252
253             cv2.imshow("KeyPoints on Video", frame)
254
255             if cv2.waitKey(1) & 0xFF == ord('q'):
256                 break
257
258         cap.release()
259         cv2.destroyAllWindows()
260
261
262     def _analyze_pose(self, show_angles):
263         pe = PoseEstimation('lebron.mkv')
264         pe.analyze_pose(show_angles=show_angles)
265
266
267     def run_analyze_pose(show_angles):
268         pe = PoseEstimation('lebron.mkv')
269         pe.analyze_pose(show_angles=show_angles)
270
271
272     def _compute_angle(keypoints):
273         angle = np.degrees(np.arctan2(
274             keypoints[1][1] - keypoints[0][1],
275             keypoints[1][0] - keypoints[0][0]))
276
277         if angle < 0:
278             angle += 360
279
280         return angle
281
282     def _append_angle(keypoints):
283         angle = self._compute_angle(keypoints)
284         self.angles.append(angle)
285
286     def _append_time():
287         time.append(frame_count)
288
289     def _plot_angles():
290         ax = plt.gca()
291         ax.plot(time, angles, marker='o', color='orange')
292         plt.xlabel('Time')
293         plt.ylabel('Angle (degrees)')
294         plt.title('Angle vs. Time')
295         plt.draw()
296         plt.pause(.05)
297
298     def _plot_text(frame, angle):
299         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
300
301     def _draw_keypoints(frame, keypoints):
302         for i in range(1, len(keypoints)):
303             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
304             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
305
306     def _show_angles(frame, keypoints):
307         for i in range(1, len(keypoints)):
308             angle = self._compute_angle(keypoints[i])
309             self._append_angle(angle)
310
311             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
312
313             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
314             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
315
316             if show_angles:
317                 angle = self._compute_angle(keypoints[i])
318                 self._append_angle(angle)
319
320                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
321
322             cv2.imshow("KeyPoints on Video", frame)
323
324             if cv2.waitKey(1) & 0xFF == ord('q'):
325                 break
326
327         cap.release()
328         cv2.destroyAllWindows()
329
330
331     def _analyze_pose(self, show_angles):
332         pe = PoseEstimation('lebron.mkv')
333         pe.analyze_pose(show_angles=show_angles)
334
335
336     def run_analyze_pose(show_angles):
337         pe = PoseEstimation('lebron.mkv')
338         pe.analyze_pose(show_angles=show_angles)
339
340
341     def _compute_angle(keypoints):
342         angle = np.degrees(np.arctan2(
343             keypoints[1][1] - keypoints[0][1],
344             keypoints[1][0] - keypoints[0][0]))
345
346         if angle < 0:
347             angle += 360
348
349         return angle
350
351     def _append_angle(keypoints):
352         angle = self._compute_angle(keypoints)
353         self.angles.append(angle)
354
355     def _append_time():
356         time.append(frame_count)
357
358     def _plot_angles():
359         ax = plt.gca()
360         ax.plot(time, angles, marker='o', color='orange')
361         plt.xlabel('Time')
362         plt.ylabel('Angle (degrees)')
363         plt.title('Angle vs. Time')
364         plt.draw()
365         plt.pause(.05)
366
367     def _plot_text(frame, angle):
368         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
369
370     def _draw_keypoints(frame, keypoints):
371         for i in range(1, len(keypoints)):
372             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
373             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
374
375     def _show_angles(frame, keypoints):
376         for i in range(1, len(keypoints)):
377             angle = self._compute_angle(keypoints[i])
378             self._append_angle(angle)
379
380             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
381
382             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
383             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
384
385             if show_angles:
386                 angle = self._compute_angle(keypoints[i])
387                 self._append_angle(angle)
388
389                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
390
391             cv2.imshow("KeyPoints on Video", frame)
392
393             if cv2.waitKey(1) & 0xFF == ord('q'):
394                 break
395
396         cap.release()
397         cv2.destroyAllWindows()
398
399
400     def _analyze_pose(self, show_angles):
401         pe = PoseEstimation('lebron.mkv')
402         pe.analyze_pose(show_angles=show_angles)
403
404
405     def run_analyze_pose(show_angles):
406         pe = PoseEstimation('lebron.mkv')
407         pe.analyze_pose(show_angles=show_angles)
408
409
410     def _compute_angle(keypoints):
411         angle = np.degrees(np.arctan2(
412             keypoints[1][1] - keypoints[0][1],
413             keypoints[1][0] - keypoints[0][0]))
414
415         if angle < 0:
416             angle += 360
417
418         return angle
419
420     def _append_angle(keypoints):
421         angle = self._compute_angle(keypoints)
422         self.angles.append(angle)
423
424     def _append_time():
425         time.append(frame_count)
426
427     def _plot_angles():
428         ax = plt.gca()
429         ax.plot(time, angles, marker='o', color='orange')
430         plt.xlabel('Time')
431         plt.ylabel('Angle (degrees)')
432         plt.title('Angle vs. Time')
433         plt.draw()
434         plt.pause(.05)
435
436     def _plot_text(frame, angle):
437         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
438
439     def _draw_keypoints(frame, keypoints):
440         for i in range(1, len(keypoints)):
441             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
442             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
443
444     def _show_angles(frame, keypoints):
445         for i in range(1, len(keypoints)):
446             angle = self._compute_angle(keypoints[i])
447             self._append_angle(angle)
448
449             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
450
451             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
452             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
453
454             if show_angles:
455                 angle = self._compute_angle(keypoints[i])
456                 self._append_angle(angle)
457
458                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
459
460             cv2.imshow("KeyPoints on Video", frame)
461
462             if cv2.waitKey(1) & 0xFF == ord('q'):
463                 break
464
465         cap.release()
466         cv2.destroyAllWindows()
467
468
469     def _analyze_pose(self, show_angles):
470         pe = PoseEstimation('lebron.mkv')
471         pe.analyze_pose(show_angles=show_angles)
472
473
474     def run_analyze_pose(show_angles):
475         pe = PoseEstimation('lebron.mkv')
476         pe.analyze_pose(show_angles=show_angles)
477
478
479     def _compute_angle(keypoints):
480         angle = np.degrees(np.arctan2(
481             keypoints[1][1] - keypoints[0][1],
482             keypoints[1][0] - keypoints[0][0]))
483
484         if angle < 0:
485             angle += 360
486
487         return angle
488
489     def _append_angle(keypoints):
490         angle = self._compute_angle(keypoints)
491         self.angles.append(angle)
492
493     def _append_time():
494         time.append(frame_count)
495
496     def _plot_angles():
497         ax = plt.gca()
498         ax.plot(time, angles, marker='o', color='orange')
499         plt.xlabel('Time')
500         plt.ylabel('Angle (degrees)')
501         plt.title('Angle vs. Time')
502         plt.draw()
503         plt.pause(.05)
504
505     def _plot_text(frame, angle):
506         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
507
508     def _draw_keypoints(frame, keypoints):
509         for i in range(1, len(keypoints)):
510             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
511             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
512
513     def _show_angles(frame, keypoints):
514         for i in range(1, len(keypoints)):
515             angle = self._compute_angle(keypoints[i])
516             self._append_angle(angle)
517
518             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
519
520             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
521             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
522
523             if show_angles:
524                 angle = self._compute_angle(keypoints[i])
525                 self._append_angle(angle)
526
527                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
528
529             cv2.imshow("KeyPoints on Video", frame)
530
531             if cv2.waitKey(1) & 0xFF == ord('q'):
532                 break
533
534         cap.release()
535         cv2.destroyAllWindows()
536
537
538     def _analyze_pose(self, show_angles):
539         pe = PoseEstimation('lebron.mkv')
540         pe.analyze_pose(show_angles=show_angles)
541
542
543     def run_analyze_pose(show_angles):
544         pe = PoseEstimation('lebron.mkv')
545         pe.analyze_pose(show_angles=show_angles)
546
547
548     def _compute_angle(keypoints):
549         angle = np.degrees(np.arctan2(
550             keypoints[1][1] - keypoints[0][1],
551             keypoints[1][0] - keypoints[0][0]))
552
553         if angle < 0:
554             angle += 360
555
556         return angle
557
558     def _append_angle(keypoints):
559         angle = self._compute_angle(keypoints)
560         self.angles.append(angle)
561
562     def _append_time():
563         time.append(frame_count)
564
565     def _plot_angles():
566         ax = plt.gca()
567         ax.plot(time, angles, marker='o', color='orange')
568         plt.xlabel('Time')
569         plt.ylabel('Angle (degrees)')
570         plt.title('Angle vs. Time')
571         plt.draw()
572         plt.pause(.05)
573
574     def _plot_text(frame, angle):
575         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
576
577     def _draw_keypoints(frame, keypoints):
578         for i in range(1, len(keypoints)):
579             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
580             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
581
582     def _show_angles(frame, keypoints):
583         for i in range(1, len(keypoints)):
584             angle = self._compute_angle(keypoints[i])
585             self._append_angle(angle)
586
587             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
588
589             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
590             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
591
592             if show_angles:
593                 angle = self._compute_angle(keypoints[i])
594                 self._append_angle(angle)
595
596                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
597
598             cv2.imshow("KeyPoints on Video", frame)
599
600             if cv2.waitKey(1) & 0xFF == ord('q'):
601                 break
602
603         cap.release()
604         cv2.destroyAllWindows()
605
606
607     def _analyze_pose(self, show_angles):
608         pe = PoseEstimation('lebron.mkv')
609         pe.analyze_pose(show_angles=show_angles)
610
611
612     def run_analyze_pose(show_angles):
613         pe = PoseEstimation('lebron.mkv')
614         pe.analyze_pose(show_angles=show_angles)
615
616
617     def _compute_angle(keypoints):
618         angle = np.degrees(np.arctan2(
619             keypoints[1][1] - keypoints[0][1],
620             keypoints[1][0] - keypoints[0][0]))
621
622         if angle < 0:
623             angle += 360
624
625         return angle
626
627     def _append_angle(keypoints):
628         angle = self._compute_angle(keypoints)
629         self.angles.append(angle)
630
631     def _append_time():
632         time.append(frame_count)
633
634     def _plot_angles():
635         ax = plt.gca()
636         ax.plot(time, angles, marker='o', color='orange')
637         plt.xlabel('Time')
638         plt.ylabel('Angle (degrees)')
639         plt.title('Angle vs. Time')
640         plt.draw()
641         plt.pause(.05)
642
643     def _plot_text(frame, angle):
644         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
645
646     def _draw_keypoints(frame, keypoints):
647         for i in range(1, len(keypoints)):
648             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
649             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
650
651     def _show_angles(frame, keypoints):
652         for i in range(1, len(keypoints)):
653             angle = self._compute_angle(keypoints[i])
654             self._append_angle(angle)
655
656             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
657
658             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
659             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
660
661             if show_angles:
662                 angle = self._compute_angle(keypoints[i])
663                 self._append_angle(angle)
664
665                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
666
667             cv2.imshow("KeyPoints on Video", frame)
668
669             if cv2.waitKey(1) & 0xFF == ord('q'):
670                 break
671
672         cap.release()
673         cv2.destroyAllWindows()
674
675
676     def _analyze_pose(self, show_angles):
677         pe = PoseEstimation('lebron.mkv')
678         pe.analyze_pose(show_angles=show_angles)
679
680
681     def run_analyze_pose(show_angles):
682         pe = PoseEstimation('lebron.mkv')
683         pe.analyze_pose(show_angles=show_angles)
684
685
686     def _compute_angle(keypoints):
687         angle = np.degrees(np.arctan2(
688             keypoints[1][1] - keypoints[0][1],
689             keypoints[1][0] - keypoints[0][0]))
690
691         if angle < 0:
692             angle += 360
693
694         return angle
695
696     def _append_angle(keypoints):
697         angle = self._compute_angle(keypoints)
698         self.angles.append(angle)
699
700     def _append_time():
701         time.append(frame_count)
702
703     def _plot_angles():
704         ax = plt.gca()
705         ax.plot(time, angles, marker='o', color='orange')
706         plt.xlabel('Time')
707         plt.ylabel('Angle (degrees)')
708         plt.title('Angle vs. Time')
709         plt.draw()
710         plt.pause(.05)
711
712     def _plot_text(frame, angle):
713         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
714
715     def _draw_keypoints(frame, keypoints):
716         for i in range(1, len(keypoints)):
717             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
718             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
719
720     def _show_angles(frame, keypoints):
721         for i in range(1, len(keypoints)):
722             angle = self._compute_angle(keypoints[i])
723             self._append_angle(angle)
724
725             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
726
727             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
728             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
729
730             if show_angles:
731                 angle = self._compute_angle(keypoints[i])
732                 self._append_angle(angle)
733
734                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
735
736             cv2.imshow("KeyPoints on Video", frame)
737
738             if cv2.waitKey(1) & 0xFF == ord('q'):
739                 break
740
741         cap.release()
742         cv2.destroyAllWindows()
743
744
745     def _analyze_pose(self, show_angles):
746         pe = PoseEstimation('lebron.mkv')
747         pe.analyze_pose(show_angles=show_angles)
748
749
750     def run_analyze_pose(show_angles):
751         pe = PoseEstimation('lebron.mkv')
752         pe.analyze_pose(show_angles=show_angles)
753
754
755     def _compute_angle(keypoints):
756         angle = np.degrees(np.arctan2(
757             keypoints[1][1] - keypoints[0][1],
758             keypoints[1][0] - keypoints[0][0]))
759
760         if angle < 0:
761             angle += 360
762
763         return angle
764
765     def _append_angle(keypoints):
766         angle = self._compute_angle(keypoints)
767         self.angles.append(angle)
768
769     def _append_time():
770         time.append(frame_count)
771
772     def _plot_angles():
773         ax = plt.gca()
774         ax.plot(time, angles, marker='o', color='orange')
775         plt.xlabel('Time')
776         plt.ylabel('Angle (degrees)')
777         plt.title('Angle vs. Time')
778         plt.draw()
779         plt.pause(.05)
780
781     def _plot_text(frame, angle):
782         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
783
784     def _draw_keypoints(frame, keypoints):
785         for i in range(1, len(keypoints)):
786             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
787             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
788
789     def _show_angles(frame, keypoints):
790         for i in range(1, len(keypoints)):
791             angle = self._compute_angle(keypoints[i])
792             self._append_angle(angle)
793
794             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
795
796             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
797             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
798
799             if show_angles:
800                 angle = self._compute_angle(keypoints[i])
801                 self._append_angle(angle)
802
803                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
804
805             cv2.imshow("KeyPoints on Video", frame)
806
807             if cv2.waitKey(1) & 0xFF == ord('q'):
808                 break
809
810         cap.release()
811         cv2.destroyAllWindows()
812
813
814     def _analyze_pose(self, show_angles):
815         pe = PoseEstimation('lebron.mkv')
816         pe.analyze_pose(show_angles=show_angles)
817
818
819     def run_analyze_pose(show_angles):
820         pe = PoseEstimation('lebron.mkv')
821         pe.analyze_pose(show_angles=show_angles)
822
823
824     def _compute_angle(keypoints):
825         angle = np.degrees(np.arctan2(
826             keypoints[1][1] - keypoints[0][1],
827             keypoints[1][0] - keypoints[0][0]))
828
829         if angle < 0:
830             angle += 360
831
832         return angle
833
834     def _append_angle(keypoints):
835         angle = self._compute_angle(keypoints)
836         self.angles.append(angle)
837
838     def _append_time():
839         time.append(frame_count)
840
841     def _plot_angles():
842         ax = plt.gca()
843         ax.plot(time, angles, marker='o', color='orange')
844         plt.xlabel('Time')
845         plt.ylabel('Angle (degrees)')
846         plt.title('Angle vs. Time')
847         plt.draw()
848         plt.pause(.05)
849
850     def _plot_text(frame, angle):
851         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
852
853     def _draw_keypoints(frame, keypoints):
854         for i in range(1, len(keypoints)):
855             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
856             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
857
858     def _show_angles(frame, keypoints):
859         for i in range(1, len(keypoints)):
860             angle = self._compute_angle(keypoints[i])
861             self._append_angle(angle)
862
863             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
864
865             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
866             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
867
868             if show_angles:
869                 angle = self._compute_angle(keypoints[i])
870                 self._append_angle(angle)
871
872                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
873
874             cv2.imshow("KeyPoints on Video", frame)
875
876             if cv2.waitKey(1) & 0xFF == ord('q'):
877                 break
878
879         cap.release()
880         cv2.destroyAllWindows()
881
882
883     def _analyze_pose(self, show_angles):
884         pe = PoseEstimation('lebron.mkv')
885         pe.analyze_pose(show_angles=show_angles)
886
887
888     def run_analyze_pose(show_angles):
889         pe = PoseEstimation('lebron.mkv')
890         pe.analyze_pose(show_angles=show_angles)
891
892
893     def _compute_angle(keypoints):
894         angle = np.degrees(np.arctan2(
895             keypoints[1][1] - keypoints[0][1],
896             keypoints[1][0] - keypoints[0][0]))
897
898         if angle < 0:
899             angle += 360
900
901         return angle
902
903     def _append_angle(keypoints):
904         angle = self._compute_angle(keypoints)
905         self.angles.append(angle)
906
907     def _append_time():
908         time.append(frame_count)
909
910     def _plot_angles():
911         ax = plt.gca()
912         ax.plot(time, angles, marker='o', color='orange')
913         plt.xlabel('Time')
914         plt.ylabel('Angle (degrees)')
915         plt.title('Angle vs. Time')
916         plt.draw()
917         plt.pause(.05)
918
919     def _plot_text(frame, angle):
920         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
921
922     def _draw_keypoints(frame, keypoints):
923         for i in range(1, len(keypoints)):
924             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
925             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
926
927     def _show_angles(frame, keypoints):
928         for i in range(1, len(keypoints)):
929             angle = self._compute_angle(keypoints[i])
930             self._append_angle(angle)
931
932             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
933
934             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
935             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
936
937             if show_angles:
938                 angle = self._compute_angle(keypoints[i])
939                 self._append_angle(angle)
940
941                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
942
943             cv2.imshow("KeyPoints on Video", frame)
944
945             if cv2.waitKey(1) & 0xFF == ord('q'):
946                 break
947
948         cap.release()
949         cv2.destroyAllWindows()
950
951
952     def _analyze_pose(self, show_angles):
953         pe = PoseEstimation('lebron.mkv')
954         pe.analyze_pose(show_angles=show_angles)
955
956
957     def run_analyze_pose(show_angles):
958         pe = PoseEstimation('lebron.mkv')
959         pe.analyze_pose(show_angles=show_angles)
960
961
962     def _compute_angle(keypoints):
963         angle = np.degrees(np.arctan2(
964             keypoints[1][1] - keypoints[0][1],
965             keypoints[1][0] - keypoints[0][0]))
966
967         if angle < 0:
968             angle += 360
969
970         return angle
971
972     def _append_angle(keypoints):
973         angle = self._compute_angle(keypoints)
974         self.angles.append(angle)
975
976     def _append_time():
977         time.append(frame_count)
978
979     def _plot_angles():
980         ax = plt.gca()
981         ax.plot(time, angles, marker='o', color='orange')
982         plt.xlabel('Time')
983         plt.ylabel('Angle (degrees)')
984         plt.title('Angle vs. Time')
985         plt.draw()
986         plt.pause(.05)
987
988     def _plot_text(frame, angle):
989         cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
990
991     def _draw_keypoints(frame, keypoints):
992         for i in range(1, len(keypoints)):
993             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
994             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
995
996     def _show_angles(frame, keypoints):
997         for i in range(1, len(keypoints)):
998             angle = self._compute_angle(keypoints[i])
999             self._append_angle(angle)
1000
1001             cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
1002
1003             cv2.line(frame, keypoints[i-1], keypoints[i], color=(0, 255, 0))
1004             cv2.circle(frame, keypoints[i], 5, color=(0, 255, 0))
1005
1006             if show_angles:
1007                 angle = self._compute_angle(keypoints[i])
1008                 self._append_angle(angle)
1009
1010                 cv2.putText(frame, f'(round(angle))', (270, 1500), cv2.FONT_HERSHEY_SIMPLEX, 5, (0, 255, 0), 2)
1011
1012             cv2.imshow("KeyPoints on Video", frame)
1013
1014             if cv2.waitKey(1) & 0xFF == ord('q'):
1015                 break
1016
1017         cap.release()
1018         cv2.destroyAllWindows()
1019
1020
1021     def _analyze_pose(self, show_angles):
1022         pe = PoseEstimation('lebron.mkv')
1023         pe.analyze_pose(show_angles=show_angles)
1024
1025
1026     def run_analyze_pose(show_angles):
1027         pe = PoseEstimation('lebron.mkv')
1028         pe.analyze_pose(show_angles=show_angles)
1029
1030
1031     def _compute_angle(keypoints):
1032         angle = np.degrees(np.arctan2(
1033             keypoints[1][1] - keypoints[0][1],
1034             keypoints[1][0] - keypoints[0][0]))
1035
1036         if angle < 0:
1037
```



# What did I do?

>>>>

```
if largest_bbox is not None:
    x1, y1, x2, y2 = largest_bbox
    # Calculate the center of the bounding box
    x_center, y_center = (x1 + x2) / 2, (y1 + y2) / 2

    # Motion tracking: Compare the current and previous bounding box centers
    if prev_center is not None:
        prev_x, prev_y = prev_center
        movement_threshold = 50 # Adjust this threshold to account for the speed of movement
        if abs(x_center - prev_x) > movement_threshold or abs(y_center - prev_y) > movement_threshold:
            # This indicates that the person has moved significantly, likely the main subject
            self.center_x, self.center_y = x_center, y_center
        prev_center = (x_center, y_center)

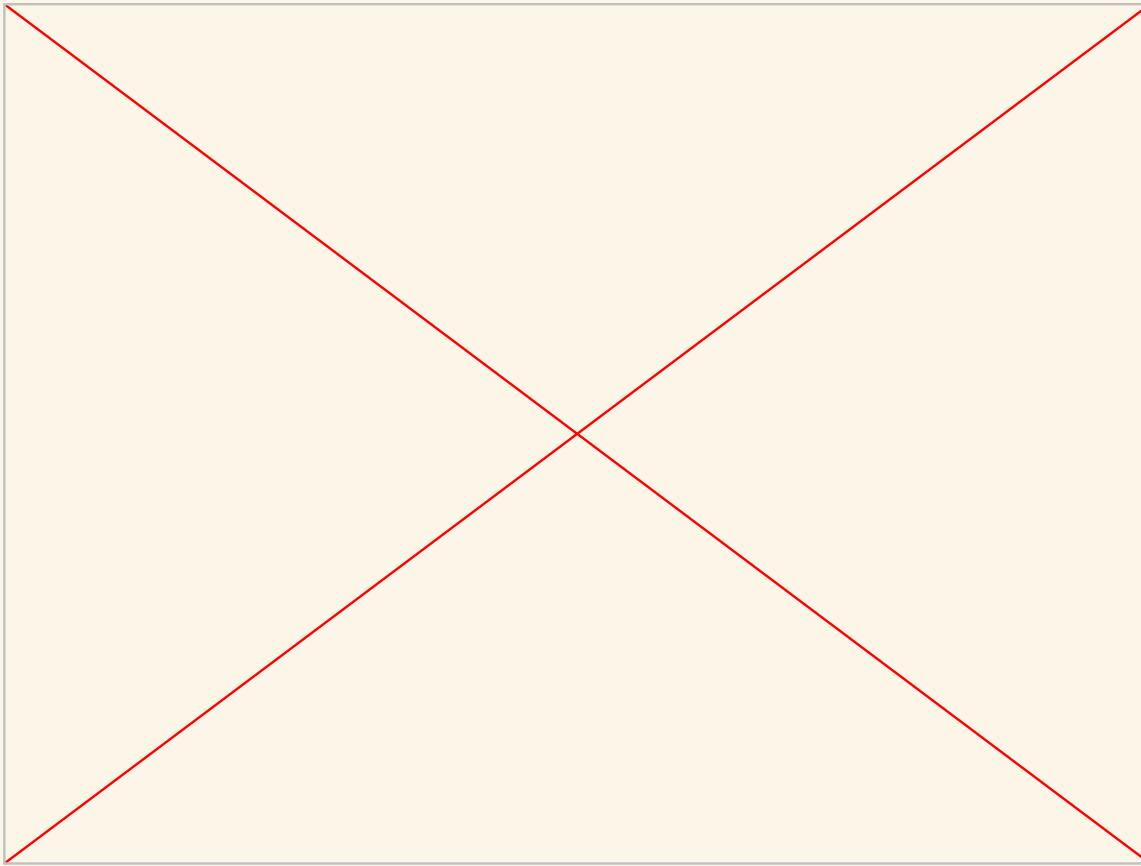
    # Ensure detection confidence is high enough before processing
    confidence = result.boxes.conf.cpu().numpy()[0]
    if confidence > confidence_threshold:
        keypoints = result.keypoints.xy.cpu().numpy()[0]
```



.....



>>>>





# What did I learn?

- Practical experience in coding and debugging
- Understood how AI can be applied to sports performance analysis
- How to use YOLO for pose estimation
- ChatGPT can be very helpful with debugging



>>>>

## Fix:

We'll use `result.boxes.xywh` or `result.boxes.xyxy` (whichever is available in the version you're using) to get the bounding box details and ensure that we only proceed if boxes are detected.

## Updated Code:

```
python

from ultralytics import YOLO
import cv2
import numpy as np
import matplotlib.pyplot as plt

def compute_angle(start, middle, end):
    vector1 = middle - start
    vector2 = end - middle
    dot_product = np.dot(vector1, vector2)
    magnitude_v1 = np.linalg.norm(vector1)
    magnitude_v2 = np.linalg.norm(vector2)
    cos_angle = dot_product / (magnitude_v1 * magnitude_v2)
    angle_rad = np.arccos(np.clip(cos_angle, -1.0, 1.0))
    angle_deg = np.degrees(angle_rad)
    return angle_deg

def rate_takeoff(keypoints):
    # Measure knee angle to determine takeoff power
    knee_angle = compute_angle(keypoints[11], keypoints[13], keypoints[15])
    if knee_angle < 45: # More bent knees mean better takeoff
        return 1.0
    elif knee_angle < 90:
```

Copy

Edit

.....



# Objective 3:

>>>>

To create a help website for  
new pole vaulters

.....





# What did I do?

- Created a simple, clean website using HTML for novice vaulters
- Added educational content - video tutorials, training tips
- Embedded my code for others to use / build from

>>>>



Pole Vaulting Help & Resources

Essential Training Videos

These videos cover everything from the basics of pole vaulting to advanced techniques and drills.

Pole Vaulting Drills for Beginners   How to Pole Vault - Full Guide   Advanced Pole Vaulting Training Tips   Pole Vaulting Technique for Strength

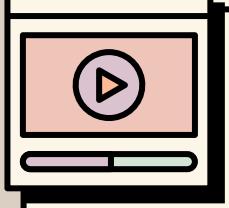
History of Pole Vaulting

Pole vaulting has its roots in ancient times, originating as a way for people to cross rivers and obstacles. It evolved into a competitive sport in the mid-19th century in Europe. The modern pole vault has been a part of the Olympic Games since 1988 for men and 2000 for women. The technique has undergone many changes, from using simple wooden poles to the carbon fiber poles used today. Over the years, pole vaulting has become one of the most challenging and athletic events in track and field, requiring speed, strength, and perfect technique.

Training Tips for Pole Vaulting

Here are some essential training tips for aspiring pole vaulters:

- Focus on building upper body and core strength to help with the pole's manipulation.
- Work on your sprinting technique and foot speed to gain maximum momentum during the approach.



>>>>



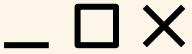
```
Senior Project Website.txt SP.html
File Edit View
</header>

<div class="content">
    <section class="section">
        <h2>Essential Training Videos</h2>
        <p>These videos cover everything from the basics of pole vaulting to advanced techniques and drills.</p>
        <div class="video-buttons">
            <a href="https://www.youtube.com/watch?v=KZ-MCujnjcA" target="_blank" class="video-button">Pole Vaulting Drills for Beginners</a>
            <a href="https://www.youtube.com/watch?v=2ByLsiNcsv0" target="_blank" class="video-button">How to Pole Vault - Full Guide</a>
            <a href="https://www.youtube.com/watch?v=z-Occerj2yA" target="_blank" class="video-button">Advanced Pole Vaulting Training Tips</a>
            <a href="https://www.youtube.com/watch?v=pQFctRtmZM0" target="_blank" class="video-button">Pole Vaulting Technique for Strength</a>
        </div>
    </section>

    <section class="section">
        <h3>About Pole Vaulting</h3>
        <p>Pole vaulting is a track and field event where athletes use a pole to jump over a high bar. It combines strength, technique, and agility, requiring athletes to master the pole's use while also timing their takeoff and release. Pole vaulting is one of the most challenging events in track and field, as it demands both explosive power and refined technique.</p>
    </section>

    <section class="section">
        <h3>Training Tips</h3>
        <p>Here are some essential training tips for aspiring pole vaulters:</p>
        <ul>
            <li>Focus on building upper body and core strength to help with the pole's manipulation.</li>
            <li>Work on your sprinting technique and foot speed to gain maximum momentum during the approach.</li>
            <li>Incorporate flexibility exercises to improve your range of motion for better technique.</li>
            <li>Practice with a coach to refine your form and work on specific skills like takeoff, pole planting, and clearance.</li>
        </ul>
    </section>

    <section class="section">
        <h3>Code Sharing and Community</h3>
        <p>I've uploaded my pole vaulting algorithm's code to the website for others to use and improve upon. This open-source approach allows
```





## Pole Vaulting Help & Resources



### Essential Training Videos

These videos cover everything from the basics of pole vaulting to advanced techniques and drills.

Pole Vaulting Drills for  
Beginners

How to Pole Vault - Full  
Guide

Advanced Pole Vaulting  
Training Tips

Pole Vaulting Technique for  
Strength

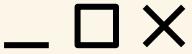
### History of Pole Vaulting

Pole vaulting has its roots in ancient times, originating as a way for people to cross rivers and obstacles. It evolved into a competitive sport in the mid-19th century in Europe. The modern pole vault has been a part of the Olympic Games since 1896 for men and 2000 for women. The technique has undergone many changes, from using simple wooden poles to the carbon fiber poles used today. Over the years, pole vaulting has become one of the most challenging and athletic events in track and field, requiring speed, strength, and perfect technique.

### Training Tips for Pole Vaulting

Here are some essential training tips for aspiring pole vaulters:

- Focus on building upper body and core strength to help with the pole's manipulation.
- Work on your sprinting technique and foot speed to gain maximum momentum during the approach.
- Incorporate flexibility exercises to improve your range of motion for better technique.
- Practice with a coach to refine your form and work on specific skills like takeoff, pole planting, and clearance.



# What did I learn?

- How to write in HTML (again)
  - Links
  - Buttons
  - colors
- Basics of website design and functionality for educational purposes
- How to share code to the public (GitHub)



>>>>



.....



# Objective 5:

>>>>

To research and apply to a  
4-year university for  
engineering

.....



# What did I do?

- Researched colleges with good engineering programs
- Made college list
- Complete a common app essay
- Finished supplemental essays for all colleges
- Applied to colleges and paid all college application fees

>>>>



The screenshot shows a search interface for "Engineering Schools". The results page displays two main entries:

- University of California, Berkeley**
  - Rank: #3 in Best Engineering Schools
  - Tuition and Fees (Master's): \$12,762 per year (in-state, full-time) / \$27,864 per year (out-of-state, full-time)
  - Enrollment (Full-Time): 2,690
  - Average Quantitative GRE: 550
  - Unlock with Compass
- Georgia Institute of Technology**
  - Rank: #4 in Best Engineering Schools
  - Tuition and Fees (Master's): \$14,416 per year (in-state, full-time) / \$30,598 per year (out-of-state, full-time)
  - Enrollment (Full-Time): 5,315
  - Average Quantitative GRE: 550
  - Unlock with Compass

College list - ALL

File Edit View Insert Format Data Tools Extensions Help

Menus 100% \$ .00 123 Arial - + B I Z A

A1 College

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
1	College	Decision	RESCINDED?	Plans...	Chance	Desire	City, State / type / weather	Acceptance %, eng %, mech rank	Total \$	Enrollment	Architecture/campus	NAIGC	Cycle	Apps deadline	Tracks dem interest?	Pros	Cons	
2	Duke	waitlisted->	<input checked="" type="checkbox"/>		super reach	amazing	Durham, NC - suburban - mild weather	8%, #19	\$87k	mid, 7k	Gothic, modern	no	RD	Jan 2	YES			
3	Columbia	waitlisted->	<input checked="" type="checkbox"/>		super reach	a lot	New York, NY - urban - cold winters hot summers	6%, #15	\$87k	mid, 7k	Neoclassical	no	RD	Jan 1				
4	Carnegie Mellon	waitlisted->	<input checked="" type="checkbox"/>		super reach	a lot	Pittsburgh, PA - urban - four seasons	14%, #8	\$85k	mid, 7k	Gothic, modern	yes	RD	Jan 3				
5	Johns Hopkins	waitlisted->	<input checked="" type="checkbox"/>		super reach	sure												
6	Northeastern	waitlisted->	<input checked="" type="checkbox"/>		reach	not really												
7	Georgia Tech	deferred->ACCEPTED->COMMITTED		YAY WOW IM COMMITTED ))))<3	reach	a lot	Atlanta, GA - urban - warm weather	20%, #4	\$81k OOS	big, 16k	Modern, tech-focused	yes	EA 1	Nov 1	top co-ops/internships, city vibes	in the south, kinda ho		
8	UC Berkeley	ACCEPTED	<input checked="" type="checkbox"/>	WOAAAH	super reach	so much	Berkeley, CA - urban - mild weather all year	11%, #3	\$78k OOS	big, 32k	Gothic, modern	yes	RD	-		perf cool weather all yr round		
9	UCLA	accepted!!	<input checked="" type="checkbox"/>	\$76k+ (85k+ w all expenses) -> prob way too expensive and far to justify it -- + no scholarships (max i could get would be \$40k anyway)	super reach	very nice	Los Angeles, CA - urban - warm weather	14%, #14	\$76k OOS	big, 32k	Modern, Spanish Revival	yes	RD	-				
10	UVA	accepted - \$0	<input checked="" type="checkbox"/>	prob commit here )) - doing scholarships... (HA U THOUGHT UVA)	hard target	yup	Charlottesville, VA - suburban	19%	\$38k	big, 22k	Jeffersonian		EA	Nov 1	AMAZING in a lot if I wanted to switch out of engineering or focus something else, great business which I'll lowkey have to do in the future	white/preppy, a little bit so (bad or no??)		
11	VT	accepted	<input checked="" type="checkbox"/>		easy target	kinda	Blacksburg, VA - rural - four seasons	55%, #13	\$44k in-state	big, 30k	Collegiate Gothic	yes	EA	Nov 15	No	people I LOVE it, top tier food, nature, great alumni network	in the south, far away from probably not too many gay (REALLY HUGE FACTOR happiness)	
12	Michigan	deferred->accepted	<input checked="" type="checkbox"/>		reach	somewhat - too big & cold?	Ann Arbor, MI - suburban - cold winters	20%, 10%, #7	\$70k OOS	31,329	Gothic, modern	yes	EA	Nov 1	YES	big school = more gay people & more friend prospects, #4 MSEIIH, amazing job prospects etc	eng building far	
13	UCSD	accepted	<input checked="" type="checkbox"/>		hard target	yes	San Diego, CA - suburban - mild weather	31%, #19	\$71k OOS	big, 31k	Modern	yes	RD	UC Dec 2		serious engineering and stem focus ), amazing research facilities, AMAZING PEREGRICO WEATHER all year, great research focus, I like a lot of the buildings & architecture, great hospital, #5 acting program	apparently not amazing soc lot of walking on campus expensive, 31k people's classes, so expensive out for not a top school (I'd prol if I get great aid), sucky p apparently not a super t campus	
14	U Washington	accepted - \$4,200/yr	<input checked="" type="checkbox"/>		easy target	kinda	Seattle, WA - urban - rainy	39%, #25	\$56k OOS	big, 32k	Gothic, modern	yes	EA	Nov 15		rainyyy, BEAUTIFUL CAMPUS AND LIBRARY		
15	Pitt	accepted	<input checked="" type="checkbox"/>		easy target	prob not	Pittsburgh, PA - urban - four seasons	59%, #45	\$52k OOS	big, 20k	Gothic, modern	yes	Roll	before Mar 1st?	YES	some lovely, grand buildings	i've heard it's quite bad we not a city vibe-more small hard and annoying to get campus	
16	Wisconsin	accepted	<input checked="" type="checkbox"/>		hard target	maybe	Madison, WI - urban - cold winters	>20%, #13!!	\$56k OOS	big, 34k	Gothic, modern	yes	EA	Nov 1		saili says amazing campus, #13 eng		



## FINAL Alex Common App Essay

File Edit View Insert Format Tools Extensions Help

Share



Menus



100%

Normal text



Arial



11



Document tabs



Tab 1

Headings you add to the document  
will appear here.

**Prompt 2:** *The lessons we take from obstacles we encounter can be fundamental to later success. Recount a time when you faced a challenge, setback, or failure. How did it affect you, and what did you learn from the experience?*

"Dear Alex—This is Sascha from Etsy's Marketplace Integrity Team. While we thank you for taking an interest in our community, this notice is to inform you that Etsy has elected to revoke your account privileges permanently. It has come to our attention that you are only 11 years old."

From a very young age, I was obsessed with *making* things: glittery bath bombs that fizzed and made my bathtub water pink; rainbow loom bracelets that lined my wrists; candles that Dad never let me light. Then, I discovered slime. I made it from scratch and gave it out at school. Everyone loved it. I started charging my fellow 6th graders \$2 for custom-made slime. Soon, I made plans to expand. With my mom as the frontman, I opened my Etsy store: EverySlimeShop.

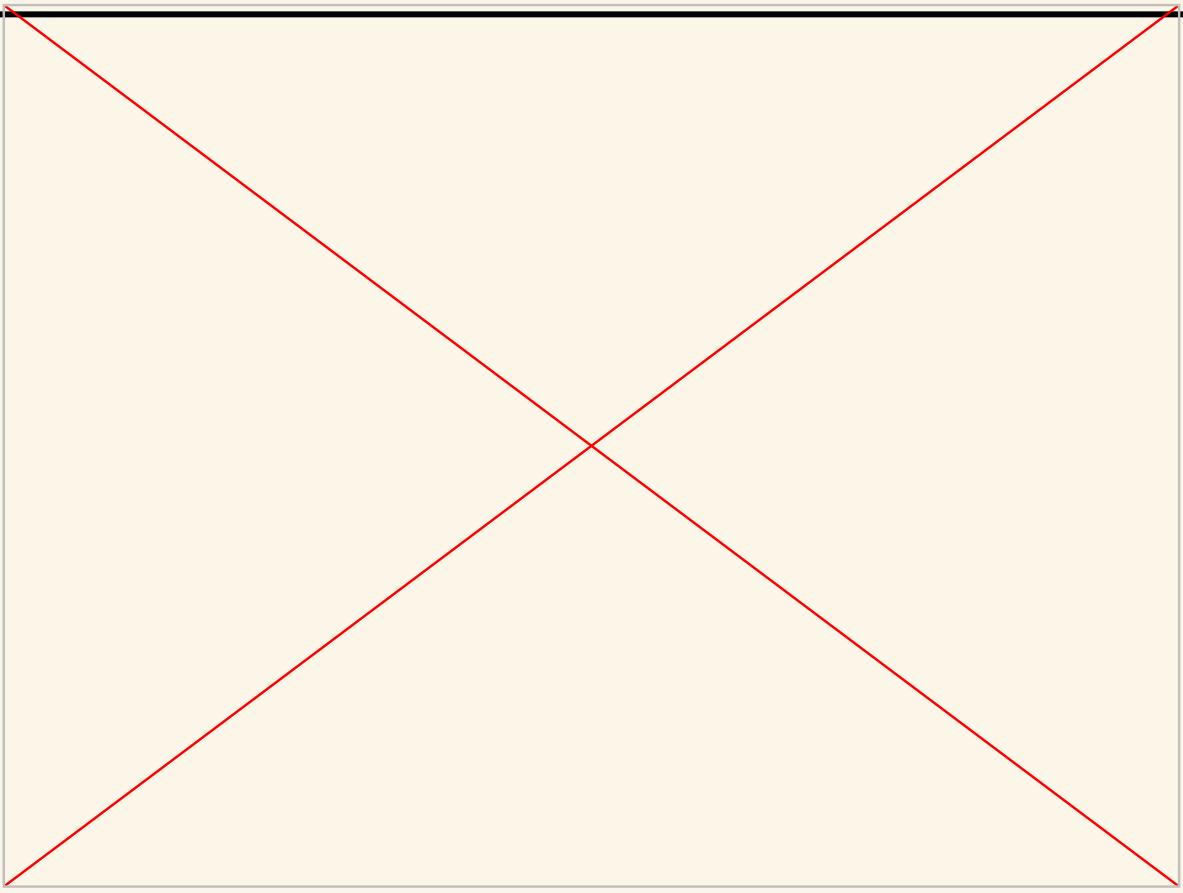
On Sept 29, 2018, I made my first sale: 10 oz of Orangesicle Cloud Slime to Candi in Reagan, Texas for \$6. I couldn't believe it. Gallons of glue and empty containers soon arrived at our doorstep. Dad would peek his head in and wonder why my bedroom was set up like a factory assembly line. Many nights I'd stay up mixing Elmer's glue with my water-borax solution until small tendrils started to latch onto the spoon.

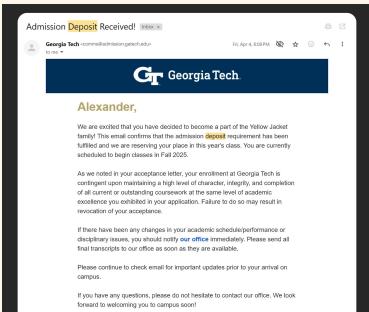
"Another order!" Susan from Blountville, Tennessee. Two days later, Janet from Gardena, California. I was ecstatic. I invested the \$27 I'd earned into new listings: Cotton Candy Cloud Slime and Piña Colada Glossy Slime. I even began to offer a complimentary Mystery Slime Box with every purchase.

I became well-known at the Etsy customer support line. In the next 3 months, in my best businessman voice and despite Mom's warnings to lay low, I barraged Laisha, Jackie, Autumn, Destini, Jakob, and Laura with my many questions.

I learned the hard way about profit margins (at the time what I called "why am I not making more money!?"), and hidden costs ("what do you mean Etsy gets a cut!?"). I peppered Dad with questions. We had long conversations about how to improve my business. Soon, people started to notice me. YouTuber "DoveSlimes" reached out asking to sample my products.

\_ □ X





- My future home ❤️❤️❤️
- Georgia Tech 2029
- Computer Engineering



# What did I learn?

~~~~~

- 30+ colleges I never knew about, and tons of majors
- Myself - through writing essays
- How to write about myself
- Insane discipline and time management
- To not overextend myself for no reason
- Apply to less colleges



>>>>



.....



>>>>

# Obstacles

.....



&gt;&gt;&gt;&gt;

# Problems & Solutions



## Time management / scheduling

- Started in summer
- Juggling with college apps

## Debugging / code not working

- Versions of YOLO
- Macbook restrictions
- Dependency issues
- Key points detecting wrong

## Wanting to give up

- Code not working → even ChatGPT can't help → felt not smart enough

## • Solutions

- Venv for dependencies
- Got python from the web instead
- YOLO→Focused on the largest model (confidence score)
- YouTube tutorials (YOLO)
- Bought code (\$5) to get started

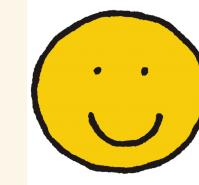


.....

&gt;&gt;&gt;&gt;

# Positive Takeaways

- Learned importance of staying ahead
  - Got ahead, then fell behind, stress me the whole year
- Relearned Python, working in Terminal, VS Code, Git/Github
  - For my major!
- Explored a whole new area of computer science
  - Computer vision! (future: self-driving cars?)
- Became a better writer / self-reflection
  - Supplements
  - Common App essay
  - SP essay



```
index.html
1<?xml version="1.0" encoding="UTF-8"?>
2<!DOCTYPE html>
3<html>
4   <head>
5     <meta charset="UTF-8" />
6     <title>File Drop Zone</title>
7     <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
8     <link href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css" rel="stylesheet">
9     <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/js/bootstrap.bundle.min.js"></script>
10    <style>
11      .fileDropZone {
12        border: 2px solid #ccc;
13        padding: 10px;
14        width: 100px;
15        height: 100px;
16        margin: auto;
17        border-radius: 50px;
18        background-color: #f0f0f0;
19      }
20      .solid-border {
21        border: 2px solid #ccc;
22      }
23      .outline-border {
24        border: 2px dashed #ccc;
25      }
26    </style>
27  </head>
28  <body>
29    <div class="fileDropZone" ondrop="handleDrop(event)" ondragover="allowDrop(event)">
30      <img alt="Smiley face icon" style="width: 100%; height: 100%; object-fit: cover;">
31    </div>
32    <div id="output">
33      <p>Drop files here or click to upload!</p>
34    </div>
35    <script>
36      const events = [
37        'dragenter',
38        'dragleave',
39        'dragover',
40        'drop'
41      ];
42      events.forEach(e => {
43        fileDropZone.addEventListener(e, (e) => {
44          e.preventDefault();
45          if (e.type === 'dragenter') {
46            fileDropZone.classList.add('solid-border');
47          }
48          if (e.type === 'dragleave') {
49            fileDropZone.classList.remove('solid-border');
50          }
51          if (e.type === 'drop') {
52            fileDropZone.classList.remove('solid-border');
53            handleFiles(e.dataTransfer.files);
54            thenValues => values.map(tag => {
55              tag.setAttribute('class', 'border rounded img-responsive');
56            });
57            fileDropZone.appendChild(tag);
58          }
59        });
60      });
61    </script>
62  </body>
63</html>
```



# Questions?



Me pole vaulting :)



Mondo Duplantis!  
(World Record Holder)

>>>>

.....