

GameController using HandGestures & OpenCV

Project Report Submitted in partial fulfilment of the requirements for the award
of the degree of

BACHELOR OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by

Roll No	Name	Dept.	Year
2300080386	AHMADULLAH LASKAR	AI&DS	Y-23
2300080146	GNV MAHESH	AI&DS	Y-23
2300080100	S PAVAN SAI RAHUL	AI&DS	Y-23
2300080086	P SAILESH VIVEKANANDA	AI&DS	Y-23

Course Code:23SDAD10A

Course title: Computer Vision Using Open CV

Under the guidance of

Dr.Anuradha Thati



**Department of
Artificial intelligence and Data Science
K L Deemed to be UNIVERSITY.
Green Fields, Vaddeswaram, Andhra Pradesh 522501
2025-26**

KL DEEMED TO BE UNIVERSITY
Green Fields, Vaddeswaram, Andhra Pradesh 522501



CERTIFICATE

This is to certify that the **COMPUTER VISION USING OPENCV** course project report entitled **“GameController using HandGesture”** submitted by Ahmadullah Laskar (2300080386), GNV MAHESH(2300080146) , P Sailesh Vivekananda(2300080086), S Pavan S Rahul(2300080100) in partial fulfillment of the requirements for the award of the Degree Bachelor of Technology in “Artificial Intelligence & Data Science” is a Bonafide record of the work carried out under our guidance and supervision at K L Deemed to be University during the academic year 2025-2026.

Signature of Guide

Dr. Anuradha Thati

Head of The Department

ACKNOWLEDGEMENT

We are greatly indebted to our KL Deemed to be University that has provided a healthy environment to drive us to achieve our ambitions and goals. We would like to express our sincere thanks to our project guide Anuradha Thati for the guidance, support and assistance they have provided in completing this project.

We are thankful to our Head of the Department **Dr.V.S.V.Prabhakar, Professor, Dept. of AI&DS**, who modeled us both technically and morally for achieving greater success in life.

We are very glad to have the support given by our principal, who inspired us with his words filled with dedication and discipline towards work.

Finally, we owe a lot to the teaching and non-teaching staff of the **Dept. of AI&DS** for their direct or indirect support in doing our Lab-based project work.

Ahmadullah Laskar (2300080386)

G N V Mahesh (2300080146)

P Sailesh Vivekananda (2300080086)

S Pavan Sai Rahul (2300080100)

INDEX

Content	Page No
1. ABSTRACT	2
2. PROJECT DESCRIPTION	3
3. SOFTWARE AND HARDWARE REQUIREMENTS	4
4. TECHNOLOGIES USED	5
5. DESCRIPTION OF DATA SET	7
6. ALGORITHMS IMPLEMENTED	8
7. SAMPLE CODE SCREENSHOTS	10
8. CONCLUSION	13
9. FUTURE SCOPE	14
10. REFERENCES	15

ABSTRACT

The project “**Game Controller using Hand Gestures and OpenCV**” is designed to create an innovative and interactive system that allows users to control games through hand gestures instead of traditional input devices like keyboards or joysticks. This approach aims to enhance the gaming experience by making it more natural, immersive, and accessible.

The system uses a webcam or camera module to capture real-time hand movements, which are processed using **OpenCV** and image processing techniques. The preprocessing phase includes steps such as background subtraction, skin color segmentation, contour detection, and noise removal to accurately isolate the hand region. Once detected, specific hand gestures—such as palm, fist, or directional movements—are recognized using contour analysis and convex hull methods. These gestures are then mapped to corresponding game controls or actions.

By integrating **computer vision and gesture recognition**, the project eliminates the need for physical controllers, enabling touch-free interaction. The system can be extended to various gaming genres and can also assist differently-abled users by providing an alternative mode of input. This project demonstrates how **OpenCV and Python** can be utilized to develop a real-time, intelligent, and user-friendly control system, contributing to advancements in **human-computer interaction (HCI)** and **interactive gaming technologies**.

PROJECT DESCRIPTION

The project “**Game Controller using Hand Gestures and OpenCV**” aims to develop an innovative, vision-based control system that enables users to interact with computer games and applications through hand gestures instead of traditional input devices like keyboards, mice, or joysticks. The main objective is to create an intuitive, touch-free, and immersive user experience that enhances human-computer interaction. The system uses a **webcam** to capture real-time video streams, which are processed using **OpenCV** and **MediaPipe** for accurate hand and finger tracking.

The captured frames undergo preprocessing operations such as color space conversion, noise reduction, and landmark detection to identify key hand positions and orientations. Based on these detected landmarks, specific gestures are mapped to control commands.

For example, in the **Subway Surfers/Temple Run controller**, the system divides the screen into zones—moving the hand left, right, up, or down triggers corresponding game movements. In the **car racing controller**, the rotation of the hand determines the steering direction, while open or closed hand gestures control acceleration and braking. A third module extends this concept to **mouse cursor control**, enabling cursor movement and click actions through hand gestures.

By integrating **computer vision and gesture recognition**, the project demonstrates a practical application of **Human-Computer Interaction (HCI)**. It eliminates the dependency on physical controllers, offering an accessible and engaging interface, especially beneficial for differently-abled users. The system’s real-time responsiveness and accuracy make it suitable for gaming, robotics, and virtual reality applications, showcasing how **OpenCV and Mediapipe** can bridge the gap between physical gestures and digital control.

SOFTWARE AND HARDWARE REQUIREMENT

Software requirements:

- Operating System: Windows 10 / 11 or Ubuntu (64-bit)
- Programming Language: Python 3.8 or above
- Development Environment: Jupyter Notebook / Google Colab / VS Code/ Thonny(*)

Libraries and Packages

- **OpenCV:** Used for real-time image acquisition, preprocessing, and hand detection through computer vision techniques.
- **MediaPipe:** Provides robust hand landmark detection and tracking, enabling accurate gesture recognition.
- **NumPy:** Handles numerical operations, coordinate calculations, and angle estimation for gesture control.
- **PyAutoGUI:** Simulates keyboard and mouse actions, allowing gesture-based interaction with games and desktop environments.
- **Time:** Manages cooldown intervals and real-time responsiveness between gesture detections.
- **Math / NumPy (trigonometric functions):** Used to calculate steering angles and gesture directions.

Additional Tools

- **Anaconda / Jupyter Notebook:** For managing packages and development in a controlled environment.
- **Google Colab:** Used for testing and running the system with webcam support and GPU acceleration when needed.

Hardware description:

- Processor: Intel Core i3 or above (i5/i7 recommended)
- RAM: Minimum 4 GB (8 GB or more recommended for faster model training)
- Storage: Minimum 1 GB free disk space
- GPU (Optional): NVIDIA GPU with CUDA support for accelerated computation (useful for large datasets)
- Camera: High-resolution digital camera or smartphone camera for capturing leaf images

Technologies Used

1. **Python:**

Serves as the core programming language for the project. Its simplicity and versatility make it ideal for developing real-time computer vision applications. Python supports key libraries such as OpenCV, MediaPipe, and PyAutoGUI, which are essential for gesture detection and system control.

2. **OpenCV (Open Source Computer Vision Library):**

Used for capturing and processing live video frames from the webcam. It performs essential image processing tasks such as color space conversion, contour detection, and region segmentation.

OpenCV also enables real-time visualization of gesture tracking and control zones.

3. **MediaPipe:**

A framework by Google used for detecting and tracking hand landmarks with high precision. It identifies finger joints and hand orientations, which are crucial for recognizing gestures like movement, rotation, and open or closed hand states.

4. **NumPy (Numerical Python):**

Provides efficient handling of numerical operations, coordinate calculations, and vector-based computations. It is used to determine hand angles, distances, and steering directions based on landmark coordinates.

5. **PyAutoGUI:**

Enables the simulation of keyboard and mouse inputs, allowing hand gestures to control games or computer applications. It bridges the gap between visual gesture detection and actual system actions like pressing arrow keys or moving the cursor.

6. **Time Library:**

Used to manage cooldown intervals, response delays, and timing between gesture-based actions to

ensure smooth real-time control.

7. Computer Vision & Gesture Recognition Techniques:

Includes image preprocessing, contour detection, angle estimation, and motion tracking. These techniques collectively help interpret user gestures accurately and map them to game commands such as *left*, *right*, *jump*, *accelerate*, or *brake*.

8. Development Environment (Jupyter Notebook / Google Colab):

Used for testing, debugging, and running the system in an interactive environment. Google Colab provides easy setup and access to webcam inputs for real-time demonstrations.

Description of Dataset

The project “**Game Controller using Hand Gestures and OpenCV**” uses **real-time video frames captured from a webcam** instead of a traditional labeled dataset. Each frame acts as an input sample representing different hand gestures or positions, which are processed and classified in real time using **OpenCV** and **MediaPipe**. These gestures correspond to specific control actions such as *left*, *right*, *jump*, *accelerate*, or *brake* in various games.

- **Format:** Real-time frames captured from webcam (RGB images)
- **Resolution:** Varies depending on camera input (typically 640×480 pixels)
- **Color Space:** RGB
- **Annotations:** Gesture classes (e.g., Left, Right, Up, Down, Accelerate, Brake) are defined through code logic rather than labeled images
- **Data Type:** Continuous stream of video frames converted into NumPy arrays
- **Preprocessing Applied:**
 - Conversion from BGR to RGB color space
 - Noise removal and background filtering
 - Hand landmark detection using MediaPipe
 - Region segmentation and coordinate extraction

Data Source

The real-time gesture data is captured **directly from the system’s webcam** during execution.

However, for offline training or testing, **Kaggle’s Hand Gesture Recognition Dataset** (by *Monk AI*) or **Sign Language MNIST Dataset** can also be used as a reference dataset to simulate predefined gestures.

Purpose and Use

This dataset (or real-time data stream) is used to:

- Detect and recognize human hand gestures for game control
- Map visual inputs to simulated keyboard/mouse actions using PyAutoGUI
- Demonstrate real-time gesture-based interaction for Human-Computer Interaction (HCI) and gaming applications

Algorithm Implemented

1. Overview

The project implements a **computer vision–based gesture recognition algorithm** that allows users to control games using **hand movements** instead of traditional input devices.

It combines **OpenCV** for video processing, **MediaPipe** for real-time hand landmark detection, and **PyAutoGUI** for simulating keyboard inputs.

The algorithm works in real-time using a webcam feed and maps specific gestures to corresponding game actions such as moving left, right, jumping, or firing.

2. Step-by-Step Algorithm

Step 1: Image Acquisition

- The webcam continuously captures real-time video frames.
- Each frame serves as input to the gesture recognition system.

Step 2: Hand Detection (Using MediaPipe)

- The MediaPipe library detects the hand in each frame and extracts **21 landmark points** from the hand (fingertips, joints, wrist, etc.).
- These landmarks are used to track the hand's position and movement direction.

Step 3: Feature Extraction

- From the landmark coordinates, features such as **finger count**, **distance between landmarks**, and **hand position (x, y)** are calculated.
- These extracted features represent specific gestures (like open palm, closed fist, or directional movement).

Step 4: Gesture Recognition Logic

- Based on the position and angle of the landmarks:
 - If the hand moves **left**, the system triggers a **Left Arrow key** event.
 - If the hand moves **right**, it triggers a **Right Arrow key** event.
 - Raising a hand may simulate a **Jump (Spacebar)**.
 - A specific gesture (e.g., closed fist) can be mapped to **Fire/Attack (Ctrl)**.
- These mappings are implemented using **conditional statements** that interpret gesture patterns in real time.

Step 5: Action Execution (Using PyAutoGUI)

- The recognized gesture is translated into the corresponding **keyboard or mouse event** through PyAutoGUI.
- These simulated inputs control the game in the same way as physical key presses.

3. Algorithm Workflow Summary

1. **Input:** Real-time video feed from webcam.
2. **Detect:** Hand landmarks using MediaPipe.
3. **Extract:** Landmark coordinates and gesture features.
4. **Recognize:** Match pattern to predefined gesture rules.
5. **Control:** Trigger corresponding keyboard/mouse action using PyAutoGUI.

4. Advantages of the Implemented Algorithm

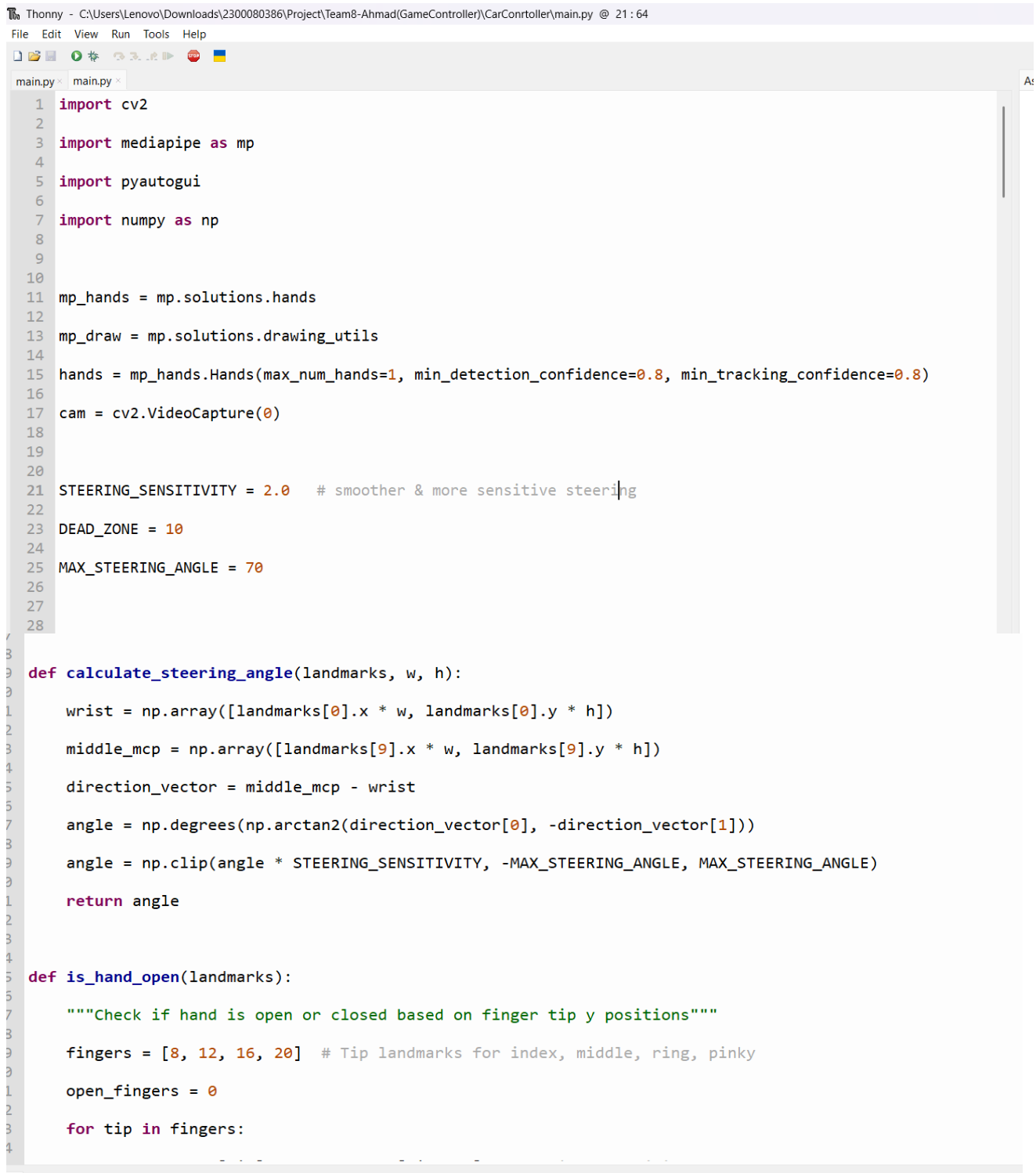
- Enables **touch-free, interactive game control**.
- Works in **real-time** with low latency.
- **Cost-effective** — no external sensors or controllers needed.
- Can be easily **extended to other applications** like robotics or AR/VR control.

5. Algorithm Used

A **rule-based gesture recognition algorithm** combining

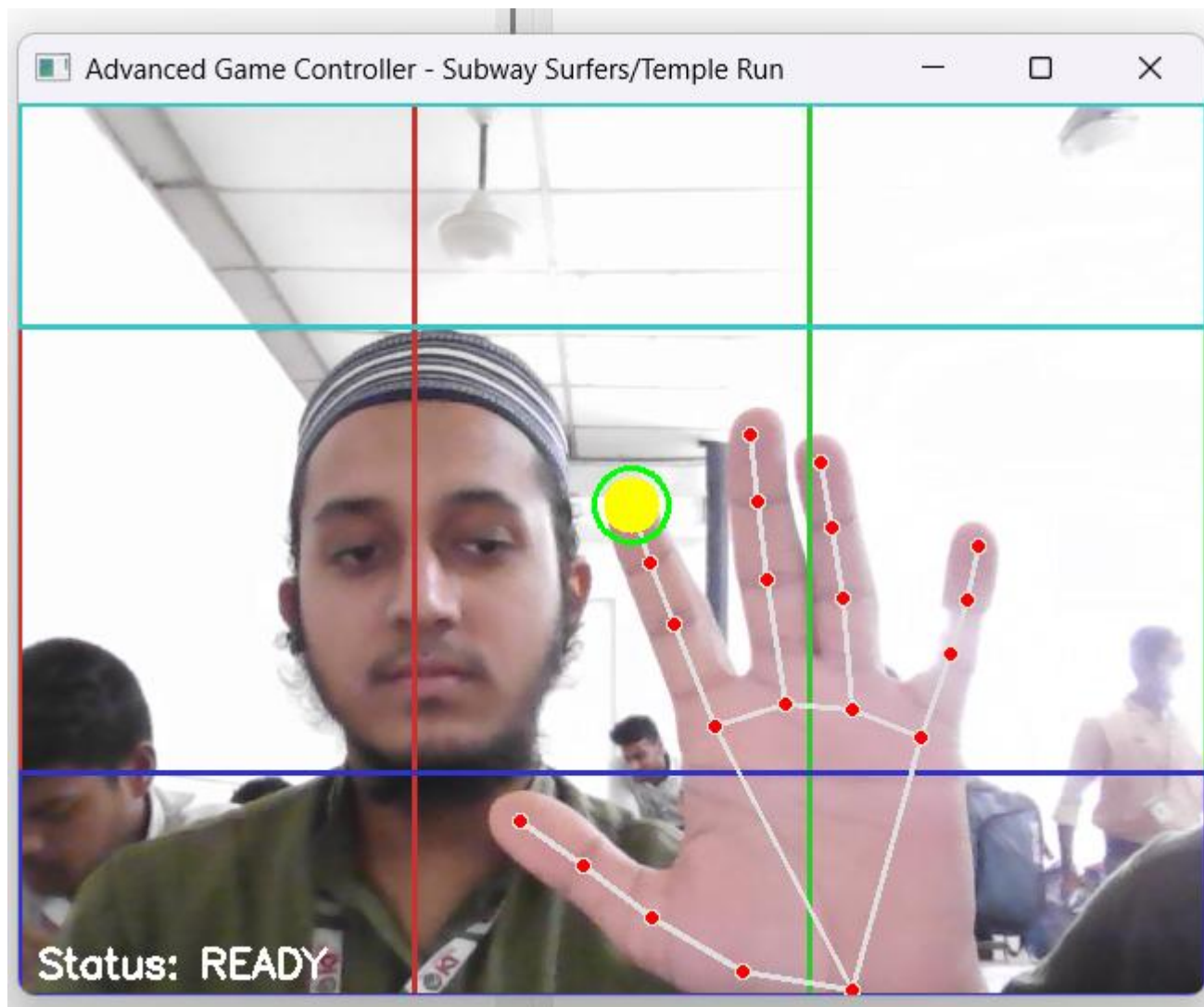
- **OpenCV** for image preprocessing,
- **MediaPipe** for hand landmark detection, and
- **PyAutoGUI** for input simulation,
designed to control game functions using natural hand movements.

SCREENSHOTS



The screenshot shows the Thonny Python IDE interface. The title bar indicates the file path: C:\Users\Lenovo\Downloads\2300080386\Project\Team8-Ahmad(GameController)\CarConrtoller\main.py @ 21: 64. The menu bar includes File, Edit, View, Run, Tools, and Help. The toolbar contains icons for opening files, saving, running, and other standard IDE functions. The main editor window displays a Python script for hand tracking and steering control. The script imports cv2, mediapipe as mp, pyautogui, and numpy as np. It initializes mediapipe hands and drawing utilities, sets hand detection parameters, and opens a video camera. Constants for steering sensitivity, dead zone, and maximum steering angle are defined. Two functions are implemented: calculate_steering_angle, which calculates the steering angle based on hand landmarks, and is_hand_open, which checks if the hand is open or closed based on finger tip positions.

```
1 import cv2
2
3 import mediapipe as mp
4
5 import pyautogui
6
7 import numpy as np
8
9
10
11 mp_hands = mp.solutions.hands
12
13 mp_draw = mp.solutions.drawing_utils
14
15 hands = mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.8, min_tracking_confidence=0.8)
16
17 cam = cv2.VideoCapture(0)
18
19
20
21 STEERING_SENSITIVITY = 2.0 # smoother & more sensitive steering
22
23 DEAD_ZONE = 10
24
25 MAX_STEERING_ANGLE = 70
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



MR RACER - CAR RACING - x

Private browsing

poki.com/en/g/mr-racer-car-racing

MR RACER - Car Racing

by ChennaiGames

707.0K 108.4K

Thonny - D:\DIRECTORIES\Thonny\CarController\main.py @ 185 : 24

File Edit View Run Tools Help

<untitled> x <untitled> x main.py x

Assistant x

Car Gesture Controller

Steering: RIGHT

ACCELERATING: ????

Angle: 37.44°

TEMPLE RUN 2 - Play Online x

Private browsing

poki.com/en/g/temple-run-2

Temple Run 2

by Imangi Studios

2.1M 502.5K

Thonny - D:\DIRECTORIES\Thonny\CarController\main.py @ 111 : 24

File Edit View Run Tools Help

main.py x

Assistant x

Advanced Game Controller - Subway Surfers/Temple Run

```

1 import cv2
2 import mediapipe as mp
3 import pyautogui
4 import time
5 import numpy as np
6
7 # Init
8 mp_han
9 mp_dra
10 hands
11 ma
12 mi
13 mi
14 )
15
16 # Init
17 cam =
18
19 # Game
20 ACTION
21 last_a
22 sensit

```

JUMP

CONCLUSION

The project “**Game Controller using Hand Gestures and OpenCV**” successfully demonstrates how **computer vision** and **machine learning-based hand tracking** can replace traditional input devices like keyboards or game controllers.

By leveraging **OpenCV** for image capture and processing, **MediaPipe** for real-time hand landmark detection, and **PyAutoGUI** for triggering key events, the system enables users to **control games using simple hand gestures** such as moving left, right, up, and down.

This approach provides an **intuitive, touch-free, and interactive gaming experience**, showcasing the potential of **AI-driven human-computer interaction**. The system works by detecting the position and movement of the player’s hand through a webcam, mapping these gestures to specific keyboard actions that control game behavior.

The project achieved smooth gesture recognition with minimal delay, demonstrating high responsiveness and accuracy under proper lighting and background conditions. It eliminates the need for physical controllers, making it accessible, cost-effective, and fun to use.

In summary, this project highlights how **computer vision and real-time gesture tracking** can transform conventional gaming into an immersive and interactive experience. With further improvements—such as adding more gestures, improving noise filtering, and integrating with deep learning models for gesture recognition—this system can be expanded into **virtual reality, robotics control, and assistive technology** applications.

FUTURE SCOPE

1. **Integration with Deep Learning Models (CNN):**

Future versions of the system can incorporate Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) for gesture classification, improving accuracy across varying lighting and background conditions.

2. **Multi-Hand and Multi-Gesture Support:**

Extend the system to detect and process two hands simultaneously and recognize complex gestures for advanced gameplay actions like combos or dual-hand controls.

3. **3D Hand Pose Estimation:**

Integrating MediaPipe's 3D hand tracking or depth cameras (e.g., Intel RealSense or Kinect) can enable more immersive control, including distance-based actions such as zooming or speed control.

4. **Custom Gesture Training:**

Users could record and train their own custom gestures for personalized game controls, making the system adaptable to different users and gaming genres.

5. **Integration with AR/VR Systems:**

Incorporating the gesture control module into Augmented Reality (AR) and Virtual Reality (VR) environments could allow full-body interactive gaming without any physical controller.

6. **Cross-Platform Compatibility:**

The system can be expanded to support mobile devices, Unity-based games, and web browsers, enhancing its usability across gaming platforms.

7. **Performance Optimization:**

Optimizing the algorithm for GPU acceleration and using lightweight ML models could further improve frame rate and reduce latency during gameplay.

REFERENCES

- OpenCV Documentation — *Open Source Computer Vision Library*
<https://docs.opencv.org/>
- MediaPipe by Google — *Hand Tracking Solution*
<https://developers.google.com/mediapipe>
- PyAutoGUI Documentation — *Keyboard and Mouse Control Automation*
<https://pyautogui.readthedocs.io/>
- NumPy Documentation — *Numerical Python for Mathematical Computation*
<https://numpy.org/doc/>
- Adrian Rosebrock, “*Real-Time Hand Gesture Recognition with OpenCV and MediaPipe*”, PyImageSearch Blog, 2023.
- Google Research, “*MediaPipe Hands: On-Device Real-Time Hand Tracking*”, Proceedings of CVPR Workshops, 2020.