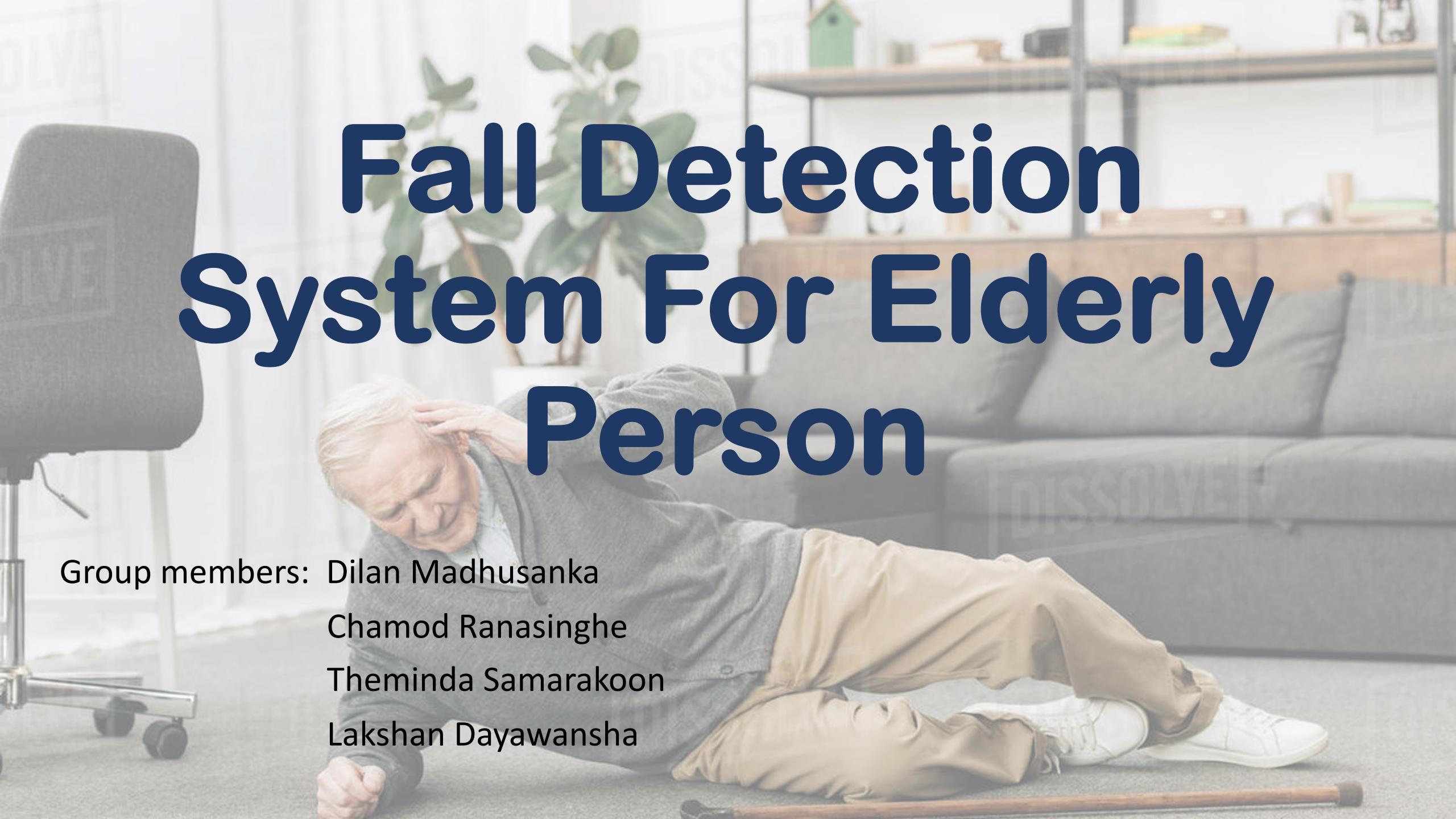


Fall Detection System For Elderly Person

A photograph of an elderly man with white hair, wearing a grey shirt and brown pants, lying on his back on a light-colored carpet. He is holding his head with one hand. In the background, there is a grey sofa, a wooden coffee table with some items on it, and a potted plant. A wooden cane lies on the floor next to his right foot. The overall atmosphere is somber, illustrating the subject of fall detection for the elderly.

Group members: Dilan Madhusanka
Chamod Ranasinghe
Theminda Samarakoon
Lakshan Dayawansha

Background

- Falls are a major cause of injury and death among elderly people
- Detecting falls in a timely manner is crucial for ensuring prompt medical attention and reducing the risk of serious injury or death.



Fall Detection System

- Automated fall detection systems are needed to quickly and accurately identify fall events and alert caregivers or emergency services.
- Wearable sensors such as the MPU6050 have made it possible to detect falls using machine learning algorithms.
- The proposed project aims to develop a real-time fall detection system using Arduino and Python with KNN machine learning algorithm to monitor elderly people and alert caregivers or emergency services in case of a fall event.

Scope of the project

- Detect accidental falls by physically weak or partially disabled elderly people.
- As the target demographic is not physically proactive, the scope can be narrowed down to distinguishing between falling and other slow movements like walking and sitting/standing up.

Data
collection
and
processing



Tools

Mpu 6050

Arduino uno

Python

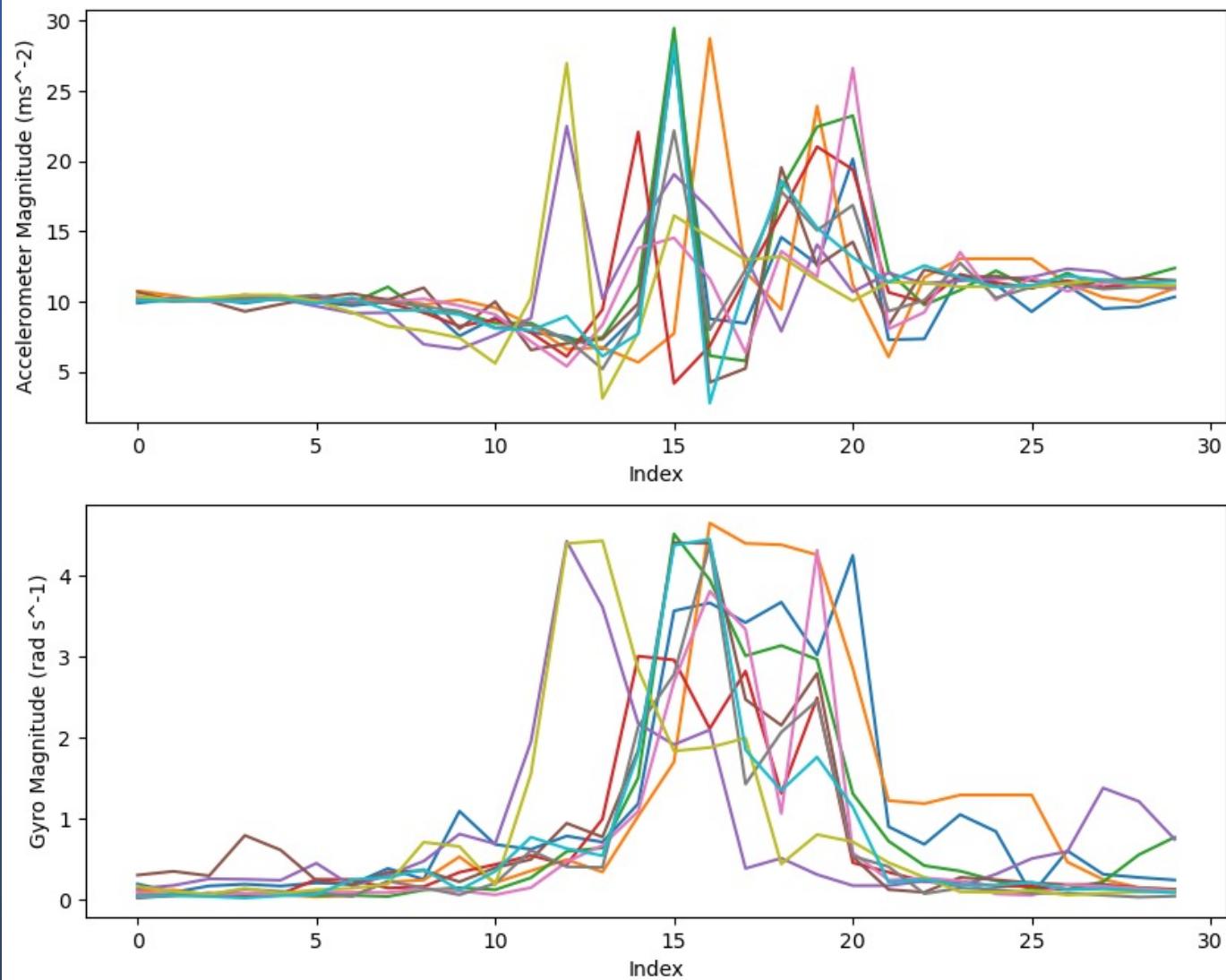


Collecting data for events

1. Falling
2. Walking
3. Sitting



Accelerometer and Gyro Magnitudes for Multiple falling events



Processing data for training the algorithm

- Average Magnitudes of accelerometer data
- Weighted average magnitudes of gyroscope data
- Standard deviation

Code Explanation And Real Time Implementation

```
mirror_mod = modifier_obj.modifiers.new("MIRROR", type="MIRROR")
# set mirror object to mirror
mirror_mod.mirror_object = selected_obj
if operation == "MIRROR_X":
    mirror_mod.use_x = True
    mirror_mod.use_y = False
    mirror_mod.use_z = False
elif operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
else:
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

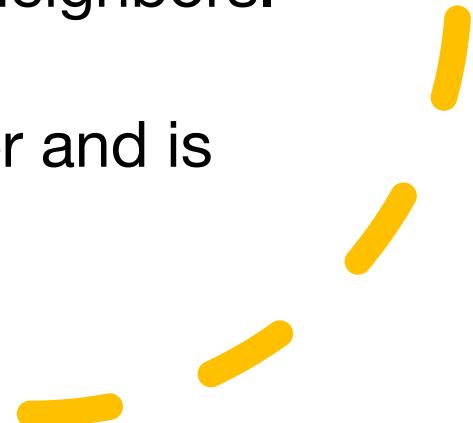
# selection at the end - add
one.select = 1
selected_obj.select=1
context.scene.objects.active = one
print("Selected" + str(modifier_obj))
selected_obj.select = 0
bpy.context.selected_objects.clear()
data.objects[one.name].select = 1
print("please select exactly one object")

-- OPERATOR CLASSES ---

@types.Operator:
def execute(self, context):
    # X mirror to the selected
    # object.mirror_mirrror_x"
    # mirror X"
    if context:
        if context.active_object is not None:
```

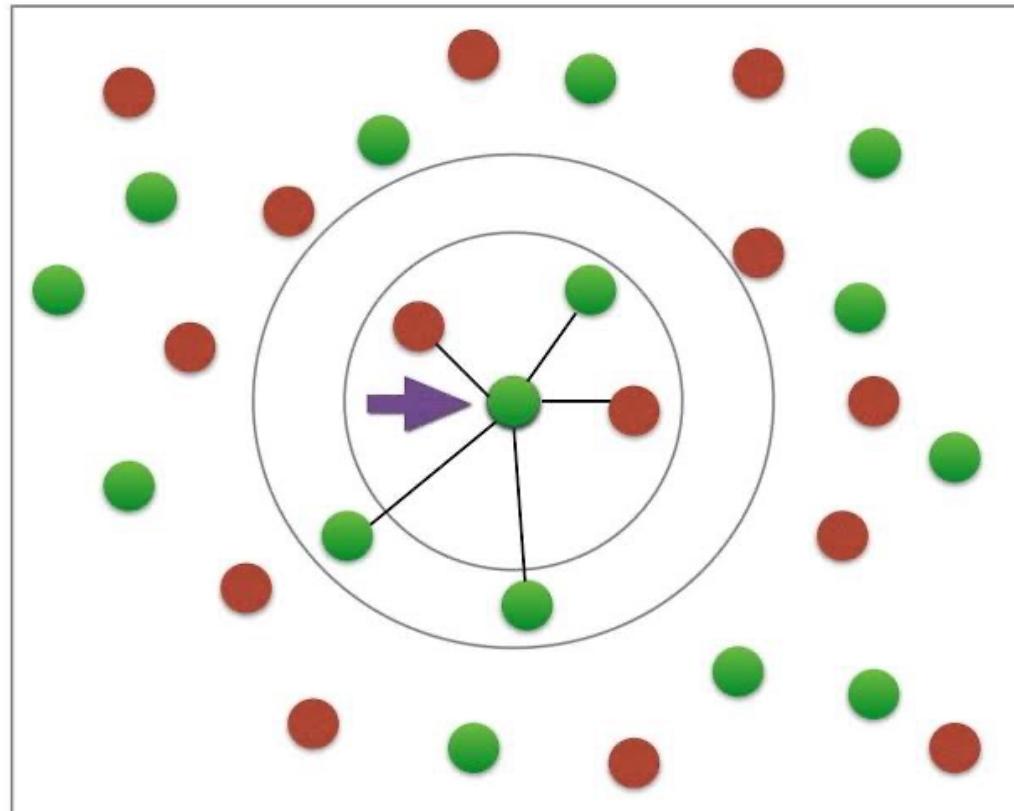
How KNN algorithm works

- KNN algorithm is a type of machine learning algorithm used for classification (and regression).
- It works by finding the k nearest neighbors of a given data point and assigns a label to it based on the most common class among its k nearest neighbors.
- The algorithm calculates the distance between the given data point and all other data points in the training set to find the nearest neighbors.
- The value of k is chosen by the user and is typically a small odd number.



How KNN algorithm works

- The value of k is chosen by the user and is typically a small odd number.



Why KNN?

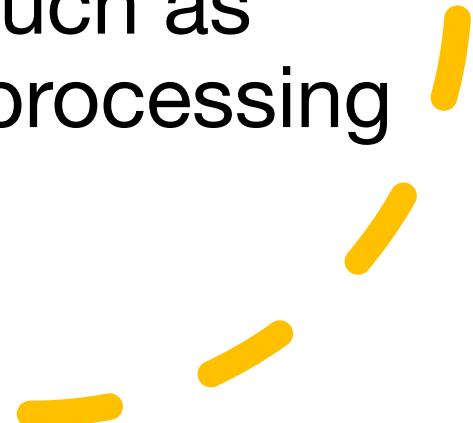
- Simplicity
- Easy to implement
- Fairly accurate for classification purposes

(A potentially better alternative: Random Forest algorithm)



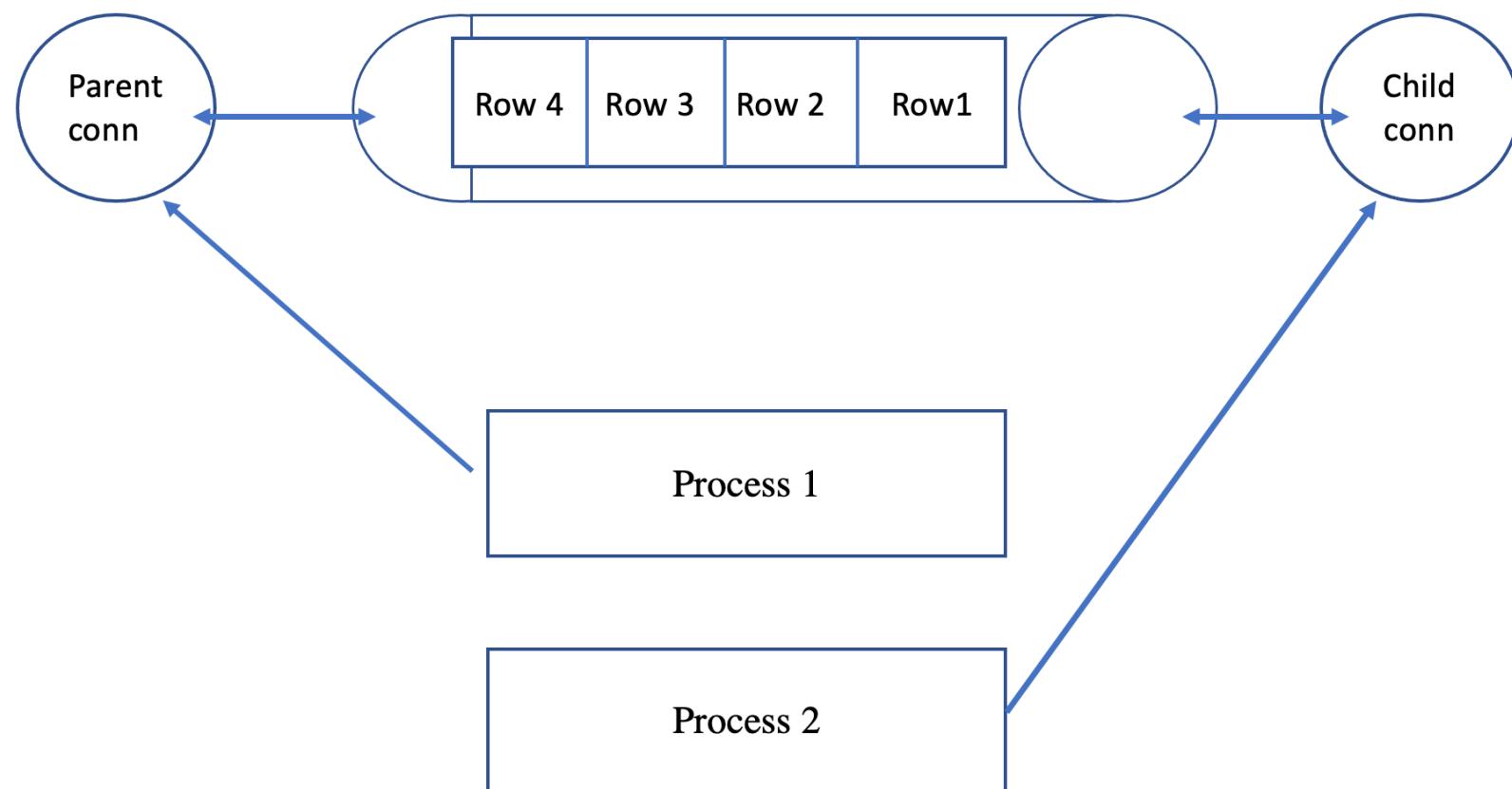
Real-time data collection (sliding window technique)

- Real-time data collection involves continuously collecting data as it is generated.
- The technique is useful for detecting falls quickly and providing timely alerts or notifications.
- Real-time implementation requires consideration of factors such as data sampling rates and processing speed.



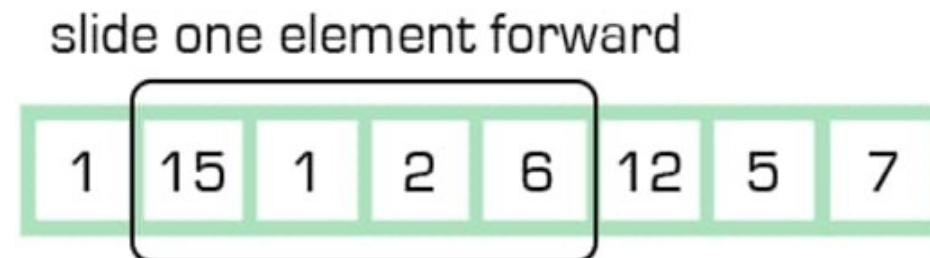
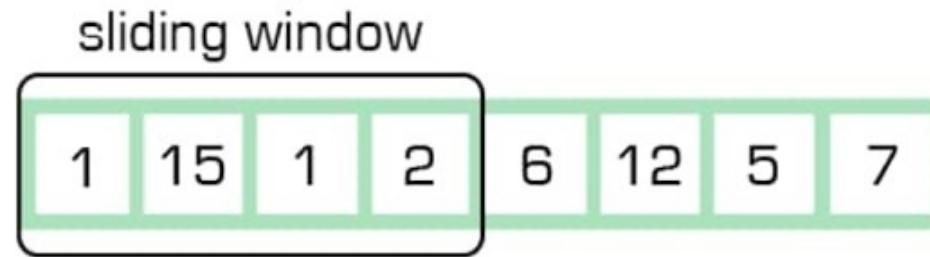
Python

Multiprocessing.Process() method and pipe()



Sliding Window Technique

- Sliding window technique involves dividing the data into overlapping windows of a fixed length.



Results

- The system can distinguish between falling, walking, and sitting.
- The system was tested using two individuals. During testing, accuracy was over 90%.

Future work

- Integration with Smart Home Systems
- Wearable Fall Detection Devices
- Cloud-based Analytics
- Mobile Applications
- Integration with Emergency Services

