UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI **INGEGNERIA**

Corso di Laurea Triennale in Informatica e dell'Automazione

Studio e sviluppo di un sistema di rilevamento delle cadute attraverso personal robots

Study and development of a fall detection system using personal robots

Relatore: Chiar.mo

Prof. **Andrea Monteriù**

Tesi di Laurea di:

Emanuele Fares

A.A. **2021** / **2022**

Indice

Indice figure

Abstract

La presente tesi si concentra sullo studio e sviluppo di un sistema di rilevamento di cadute attraverso l'utilizzo di robot personali. In particolare, è stata implementata una soluzione basata su Pose Estimation e Machine Learning, utilizzando i moduli *PoseEstimationMin* e *PoseModule*, per il robot Temi®. L'obiettivo è stato quello di migliorare la sicurezza e l'autonomia delle persone in ambienti domestici o assistenziali, dove le cadute rappresentano una delle principali cause di lesioni e incidenti. Sono stati effettuati simulazioni e per valutare l'efficacia della soluzione proposta, ottenendo risultati promettenti in termini di accuratezza e tempestività nella rilevazione delle cadute. La presente tesi fornisce quindi una valida base di partenza per future implementazioni di sistemi di rilevamento delle cadute attraverso robot personali.

CAPITOLO 1 - Introduzione

Contesto

Il contesto del progetto riguarda lo sviluppo di un sistema di rilevamento di cadute attraverso l'utilizzo di robot personali. La tecnologia dei robot personali è in continua evoluzione e il loro utilizzo può fornire soluzioni innovative per il monitoraggio e la prevenzione di incidenti domestici. Il sistema proposto e sviluppato nell'attività di tesi si avvale dell'utilizzo di una fotocamera montata sul robot per monitorare le persone e rilevare eventuali cadute. Il robot su cui è stato sviluppato l'approccio è il Temi, dotato di una fotocamera sul tablet che gli consente di monitorare l'ambiente circostante e rilevare eventuali cadute di persone. Il sistema si propone di migliorare la sicurezza delle persone, in particolare di quelle anziane o con problemi di mobilità, che spesso rischiano di cadere e di non essere in grado di rialzarsi senza aiuto. Tale sistema potrebbe avere un impatto significativo nella prevenzione degli infortuni domestici e nell'assistenza alle persone anziane o con disabilità.

Motivazione

Il numero di persone che necessitano di assistenza continua sta aumentando costantemente, e questo sta diventando una sfida per il sistema sanitario. In particolare, l'assistenza domiciliare diventa essenziale per le persone anziane o disabili che desiderano rimanere indipendenti nella loro casa. Tuttavia, l'assistenza domiciliare richiede il monitoraggio costante dello stato di salute e sicurezza dei pazienti. In questo contesto, l'uso di robot personali come il robot Temi, dotati di sensori, fotocamere e algoritmi di intelligenza artificiale, può aiutare a rilevare e prevenire eventuali cadute o incidenti, migliorando la qualità della vita dei pazienti e riducendo il carico di lavoro per i caregiver.

La motivazione di questo studio è esplorare le potenzialità dell'uso di robot personali per il monitoraggio delle cadute e la prevenzione degli incidenti, sia nell'ambito dell'assistenza domiciliare che in altri contesti.

Problema

Con l'aumento della popolazione anziana, è diventato sempre più urgente trovare modi per garantire la loro sicurezza e il loro benessere. Una delle principali preoccupazioni riguarda le cadute, che rappresentano una delle principali cause di ferite e persino di morte tra gli anziani. Per affrontare questo problema, sono state sviluppate diverse soluzioni, tra cui l'uso di robot personali dotati di fotocamere per il monitoraggio degli anziani e la rilevazione delle cadute. Tuttavia, nonostante i progressi compiuti nella tecnologia dei robot personali, la rilevazione delle cadute rimane ancora una sfida, poiché richiede algoritmi sofisticati e precisi per l'analisi delle immagini e la distinzione tra i movimenti normali e quelli associati alle cadute. In questo contesto, il presente studio si propone di sviluppare un sistema di rilevamento delle cadute basato su robot personali, utilizzando una combinazione di tecniche di visione artificiale e di apprendimento automatico per migliorare l'accuratezza e l'affidabilità della rilevazione.

Stato dell'arte

Nell'articolo "Human-Robot Collaboration for Fall Detection in Smart Home Environment" si propone l'utilizzo di un sistema di sensori, come sensori di movimento e sensori di pressione, per rilevare la caduta di una persona. Una volta che il sistema rileva una caduta, un robot Pepper viene attivato per fornire assistenza alla persona caduta [1].

Nell'articolo, intitolato "Fall Detection using Pepper Robot in Home Environment", si propone invece l'utilizzo di una fotocamera e di un sistema di riconoscimento di immagini per rilevare la caduta di una persona [2].

Ci sono anche soluzioni che utilizzano l'intelligenza artificiale per rilevare la caduta. Ad esempio, nell'articolo "Fall Detection using Deep Learning Techniques" si propone l'utilizzo di una rete neurale convoluzionale per riconoscere la caduta di una persona [3].

L'articolo "Deep Learning for Fall Detection: 3D-CNN Combined with LSTM on Wearable Sensor Data" utilizza invece una combinazione di reti neurali convoluzionali e ricorrenti per rilevare le cadute attraverso l'utilizzo di sensori indossabili [4].

Gli aurori dell'articolo di IEEE (Institute of Electrical and Electronics Engineers) "An Intelligent Fall Detection System Based on Human Skeleton Detection Using RGB-D Camera" propongono una soluzione di monitoraggio delle cadute tramite una telecamera RGB-D, che acquisisce dati sui movimenti delle persone attraverso un algoritmo di rilevamento del movimento basato sulla visione artificiale. Questo sistema è in grado di rilevare cadute in modo accurato e tempestivo, inviando segnali di allarme a caregiver o servizi di emergenza [5].

L'articolo "Design and Implementation of a Robot-Assisted Smart Environment for the Elderly Care" descrive invece un sistema di monitoraggio e assistenza integrato, che utilizza il robot Pepper per interagire con gli anziani e raccogliere informazioni sulle loro attività quotidiane. Il robot è in grado di rilevare cadute e di chiamare i caregiver o i servizi di emergenza in caso di bisogno [6].

Dalla precedente analisi dello stato dell'arte, si evince chiaramente che le soluzioni proposte finora si concentrano sull'utilizzo di sensori e/o di robot per rilevare e fornire assistenza in caso di caduta, da cui nasce l'idea sviluppata in questa tesi di utilizzare robot personali per il rilevamento efficace delle cadute.

La presente testi è strutturata come segue:

Capitolo 2: Stato dell'arte

Capitolo 3: Soluzione proposta:

> È descritta la soluzione proposta, gli algoritmi implementati ed il set-up di simulazione

Capitolo 4: Risultati

> Risultati e l'analisi dei protocolli sperimentati

Capitolo 5: Conclusione

CAPITOLO 2 - STATO DELL'ARTE:

Lo stato dell'arte relativo al rilevamento delle cadute presenta diverse soluzioni proposte dai ricercatori del campo. Studi recenti hanno evidenziato l'importanza di sviluppare sistemi di rilevamento precisi e affidabili, considerando il crescente numero di persone anziane e la necessità di garantire la loro sicurezza.

Nell'articolo "Uomini e robot: l'impatto sul lavoro" [1], vengono affrontati i progressi nel campo dell'interazione tra persone e robot, concentrandosi sulla sicurezza e sulla prevenzione degli incidenti. Si discute l'importanza di sviluppare algoritmi intelligenti per il rilevamento delle cadute al fine di fornire un'assistenza tempestiva.

Un altro articolo, intitolato "Fall Detection Using Deep Learning Models" [2], si focalizza sull'utilizzo di modelli di deep learning per il rilevamento delle cadute. Gli autori presentano un'analisi delle diverse tecniche di riconoscimento e classificazione utilizzate per identificare le cadute e ne valutano l'efficacia.

Inoltre, il documento "Vision-based Fall Detection System for Ambient Assisted Living" [3] propone un sistema basato sulla visione per il rilevamento delle cadute in ambienti domestici. Gli autori descrivono l'implementazione di un algoritmo di rilevamento delle cadute che utilizza una combinazione di rilevamento del movimento e analisi delle caratteristiche spaziali.

Infine, l'articolo "Real-Time Fall Detection System Using a Depth Camera" [4] presenta un sistema di rilevamento delle cadute in tempo reale che sfrutta una telecamera di profondità. Gli autori illustrano l'utilizzo di algoritmi di riconoscimento del corpo umano e di analisi dei movimenti per identificare le cadute e generare segnali di allarme.

Questi studi evidenziano l'importanza di sviluppare soluzioni accurate e efficienti per il rilevamento delle cadute, che possano essere applicate in diversi contesti, come l'assistenza agli anziani e la sorveglianza domestica. Nella presente tesi, si propone un sistema di rilevamento delle cadute basato sull'utilizzo di un personal robot e sull'analisi in tempo reale delle immagini acquisite dalla fotocamera

integrata. Tale soluzione si basa su algoritmi di riconoscimento del corpo umano e di analisi dei movimenti, consentendo un rilevamento rapido e affidabile delle cadute.

CAPITOLO 3 - SOLUZIONE PROPOSTA:

Per la soluzione proposta, è stato utilizzato il robot Temi dotato di una fotocamera in tempo reale e un set di algoritmi di riconoscimento del corpo umano e di analisi delle immagini. Questo ha permesso di sviluppare un sistema che può rilevare cadute in modo accurato e affidabile, e attivare una risposta rapida e appropriata. Il robot Temi monitora continuamente la stanza attraverso la sua fotocamera e può identificare rapidamente quando un individuo cade, mentre il sistema di algoritmi di riconoscimento del corpo umano integrato nel robot può verificare l'orientamento e la posizione del corpo per determinare se si tratta effettivamente di una caduta.

Set up di simulazione/sperimentale

È stata eseguita una serie di simulazioni per valutare l'efficacia dell'algoritmo proposto per il rilevamento delle cadute. Sono stati utilizzati sia dei video preregistrati che la fotocamera in tempo reale per garantire una maggiore veridicità delle condizioni quotidiane. In particolare, l'algoritmo proposto è stato testato in un ambiente di laboratorio universitario, caratterizzato da una buona illuminazione e presenza di ostacoli come altri dispositivi, robot e droni, scrivanie e sedie. Inoltre, è stato eseguito un test anche in un ambiente domestico durante le attività quotidiane.

Algoritmi implementati

L'algoritmo proposto è stato sviluppato e testato in Python, utilizzando il modello di machine learning OpenPose per rilevare i punti del corpo umano e la libreria cv2 di OpenCV per la manipolazione dell'immagine e la visualizzazione dei risultati.

OpenCV è una libreria open source molto popolare per il computer vision e l'elaborazione delle immagini. Sono state sfruttate le potenti funzionalità di questa libreria per effettuare operazioni come il ridimensionamento delle immagini, il disegno di forme e testi, nonché la gestione dei flussi video. La sua interfaccia user-friendly e la vasta documentazione ha permesso di implementare con successo queste operazioni nell'ambito dell'algoritmo proposto per il rilevamento delle cadute.

L’attività MediaPipe Pose Landmarker consente di rilevare punti di riferimento di corpi umani in un’immagine o in un video. È possibile utilizzare questa attività per identificare le posizioni chiave del corpo, analizzare la postura e classificare i movimenti. Questa attività utilizza modelli di machine learning (ML) che funzionano con singole immagini o video. L’attività restituisce i punti di riferimento della posa del corpo nelle coordinate dell’immagine e nelle coordinate del mondo tridimensionale.

Caratteristiche

- **Elaborazione dell’immagine in ingresso**: l’elaborazione include la rotazione, il ridimensionamento, la normalizzazione e la conversione dello spazio colore dell’immagine.
- **Soglia punteggio**: filtra i risultati in base ai punteggi di previsione.

Il Pose Landmarker accetta un input di uno dei seguenti tipi di dati:

- Immagini fisse
- Fotogrammi video decodificati
- Alimentazione video in diretta

Il Pose Landmarker genera i seguenti risultati:

- Posa punti di riferimento in coordinate immagine normalizzate
- Posa punti di riferimento nelle coordinate del mondo
- Opzionale: una maschera di segmentazione per la posa.

Questa attività ha le seguenti opzioni di configurazione:

running_mode:

Imposta la modalità di esecuzione per l’attività. Il landmarker ha le seguenti modalità:

IMAGE: la modalità per riconoscere i punti di riferimento della posa su input di immagini singole.

VIDEO: la modalità per riconoscere i punti di riferimento della posa sui fotogrammi decodificati di un video.

LIVE_STREAM: la modalità per riconoscere i punti di riferimento della posa su un flusso live di dati di input, ad esempio dalla fotocamera. In questa modalità, devi chiamare il result_callbacklistener per ricevere i risultati della segmentazione in modo asincrono.

min_pose_detection_confidence:

Il punteggio di confidenza minimo affinché il rilevamento della posa sia considerato riuscito.

min_pose_presence_confidence:

Il punteggio di confidenza minimo del punteggio di presenza della posa nel rilevamento del punto di riferimento della posa.

min_tracking_confidence:

Il punteggio di confidenza minimo affinché il tracciamento della posa sia considerato riuscito.

Modelli

Il Pose Landmarker utilizza una serie di modelli per prevedere i punti di riferimento della posa. Il primo modello rileva la presenza di corpi umani all'interno di una cornice dell'immagine e il secondo modello individua i punti di riferimento sui corpi.

I seguenti modelli sono raggruppati in un pacchetto di modelli scaricabile:

- **Modello di rilevamento della posa**: rileva la presenza di corpi con alcuni punti di riferimento di posa chiave.
- **Pose landmarker model**: aggiunge una mappatura completa della posa. Il modello produce una stima di 33 punti di riferimento di posa tridimensionali.

Inoltre, l'algoritmo di rilevazione delle cadute si basa sulla posizione relativa di alcuni landmark identificati dal modello OpenPose.

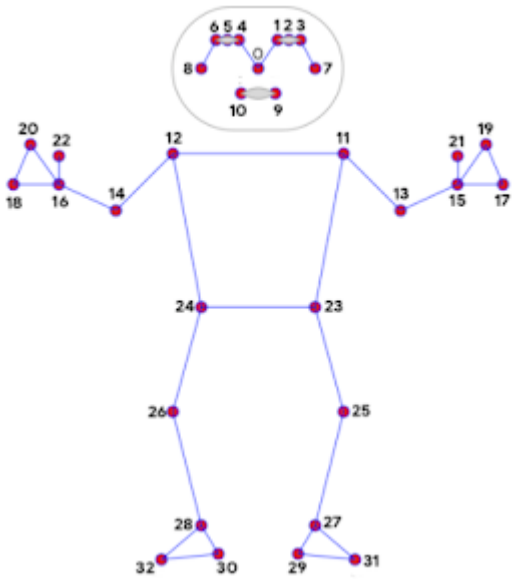

Figura 1-Rappresentazione dei 33 landmarks

In particolare, si controlla se entrambe le ginocchia sono sotto entrambe le spalle e se entrambe le caviglie/piedi sono al di sotto di entrambe le spalle.

In dettaglio, i punti considerati sono quelle delle spalle e dei piedi, poiché questi sono i punti critici che indicano una caduta imminente, in particolare, sono stati selezionati i seguenti punti anatomici:

- Ginocchia (punti 25 e 26) - evidenziati in rosso
- Spalle (punti 11 e 12) - evidenziate in verde

- Anche (punti 23 e 24) - evidenziate in giallo
- Caviglie (punti 27 e 28) - evidenziate in viola
- Piedi (punti 29 e 30) - evidenziati in bianco

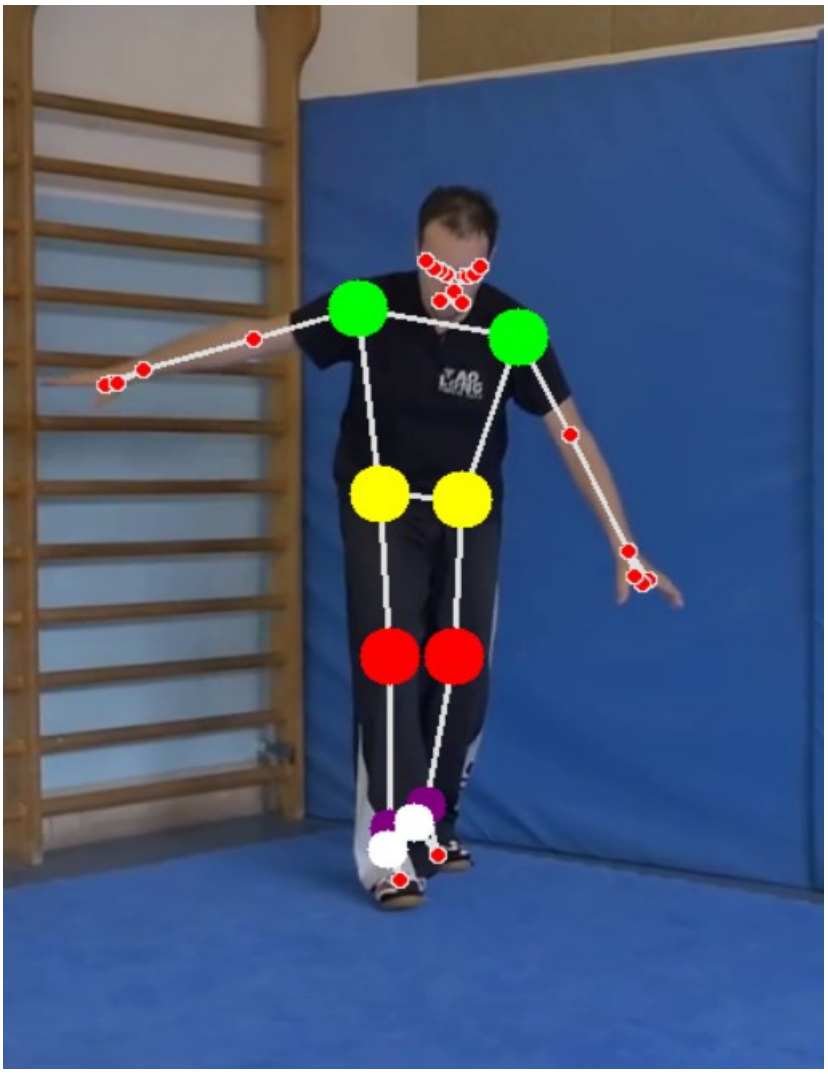

Figura 2-Persona con i landmarks di interesse

Per valutare se la persona è caduta a terra, sono stati impostati i seguenti controlli:

Controllo del rilevamento sufficiente: verificare la presenza di tutti i punti anatomici necessari per la valutazione. Se alcuni punti mancano, l'algoritmo non può determinare se la persona è caduta a terra.

In caso di fallimento dei controlli, viene emesso un messaggio di avviso.

Controllo condizioni: verificare se i punti di interesse richiesti rispettano determinate condizioni.

Controlli successivi: varia in base alla tipologia di algoritmo che è stato utilizzato, alcune tipologie richiedono verifiche aggiuntive di condizioni per poter restituire in Output in messaggio di Persona a terra.

Per identificare la caduta, sono stati utilizzati diversi algoritmi basati principalmente sull'utilizzo delle coordinate x e y dei vari landmarks. Se la posizione dei punti come spalle, ginocchia o anche, supera una certa soglia oppure verificano che sono rispettate determinare condizioni imposte allora l'algoritmo proposto riconosce che la persona sta per cadere oppure se è già a terra, così emette un messaggio di allarme: "Attenzione/PERICOLO" o "PERSONA A TERRA".

Le prestazioni dell'algoritmo sono state valutate su diverse configurazioni ambientali, e i risultati hanno dimostrato una buona affidabilità e precisione nel rilevamento delle cadute.

Tra i diversi algoritmi implementati, sono risultati i più efficienti i seguenti algoritmi:

Algoritmo n1

```
#Controlla se entrambe le caviglie/piedi sono al di sotto di entrambe le spalle
if lmList[27][2] < lmList[11][2] and lmList[28][2] < lmList[12][2] and lmList[29][2] < lmList[11][2] and \
        lmList[30][2] < lmList[12][2]:
    print("PERSONA A TERRA!!")

# Check if both feet are close to each other and below both hips
if lmList[28][1] > lmList[24][1] and lmList[29][1] > lmList[25][1]:
    distance = abs(lmList[28][1] - lmList[29][1])
    if distance < 50:
        print("persona a terra")

keypoints_to_analyze = [11, 12, 23, 24, 27]  # Anche, ginocchio e piede per entrambe le gambe

y_coordinates = [lmList[keypoint][2] for keypoint in keypoints_to_analyze]
avg_y = sum(y_coordinates) / len(y_coordinates)

ground_threshold =500

if avg_y > ground_threshold:
  print("Persona a terra")
```

Figura 3-Algoritmo n1

Di base gli algoritmi utilizzati, come precedentemente annunciato, si basano sull'utilizzo delle coordinate *x* e *y*, in particolare, questo algoritmo risulta diverso dai successivi non tanto per utilizzo dei diversi Landmarks e le sue condizioni di verifica, ma la sua peculiarità riguarda l'utilizzo delle coordinate Y rappresentata da una variabile, la quale successivamente utilizzata come termine ultimo di confronto rispetto ad un valore di soglia che è stato settato.

L'utilizzo delle coordinate Y ha però posto alcune problematiche che rincontreremo in altri algoritmi, come per esempio le condizioni di luminosità e distanza dalla Fotocamera in questione.

Questo durante i test di valutazione dell'accuratezza, ha restituito dei valori non del tutto veritieri.

Algoritmo n2

```
# punti di interesse
head = lmList[0]
right_shoulder = lmList[12]
left_shoulder = lmList[11]
right_knee = lmList[26]
left_knee = lmList[25]
right_elbow = lmList[14]
left_elbow = lmList[13]

# calcolo della distanza tra le posizioni dei punti di interesse
head_shoulder_distance = ((right_shoulder[1] - left_shoulder[1]) ** 2 + (
            right_shoulder[2] - left_shoulder[2]) ** 2) ** 0.5
shoulder_knee_distance = ((right_knee[1] - left_knee[1]) ** 2 + (right_knee[2] - left_knee[2]) ** 2) ** 0.5
elbow_shoulder_distance = ((right_elbow[1] - left_elbow[1]) ** 2 + (right_elbow[2] - left_elbow[2]) ** 2) ** 0.5

# verifica se la persona è caduta
if head[2] > right_shoulder[2] and head[2] > left_shoulder[2] and \
        shoulder_knee_distance > 1.2 * head_shoulder_distance and \
        elbow_shoulder_distance < 0.7 * head_shoulder_distance:
    print("La persona è caduta!")
```

Figura 4-Algoritmo n2

Il seguente algoritmo è stato pensato concentrandosi in particolare su alcuni punti superiori del corpo, con focus principale sulla distanza della testa con altri punti del corpo, come le spalle.

Ciò lo rende diverso dagli altri algoritmi proposti, ma come descritto per il precedente algoritmo, non disporre di una sufficiente condizione di luce, portava ad una scorretta misurazione richiesta dall'algoritmo, ovvero tra testa e spalle.

Nonostante le condizioni ambientali testate, questo algoritmo non risulta particolarmente accurato poiché in situazioni quotidiane, alcuni movimenti naturali della testa, portavano l'algoritmo a segnalare una caduta, quindi un falso positivo.

Algoritmo n3

```python
#Controlla se tutti i landmarks richiesti sono presenti
required_landmarks = [11, 12, 23, 24]
if not all(lmList[i] for i in required_landmarks):
    print("punti insufficienti per la determinazione ")

#Controlla se entrambe le ginocchia sono sotto entrambe le spalle
if lmList[23][2] < lmList[11][2] and lmList[24][2] < lmList[12][2]:
    print("ATTENZIONE/PERICOLO")

#Controlla se entrambe le caviglie/piedi sono al di sotto di entrambe le spalle
if lmList[27][2] < lmList[11][2] and lmList[28][2] < lmList[12][2] and lmList[29][2] < lmList[11][2] and \
        lmList[30][2] < lmList[12][2]:
    print("PERSONA A TERRA!!")
```

Figura 5-Algoritmo n3 "ottimo"

Quest'ultimo algoritmo si basa sempre sull'utilizzo delle coordinate *x* e *y* come i precedenti, e non va ad applicare particolari calcoli utilizzando distanze come nell'algoritmo n2 e non si basa principalmente sulle coordinate delle Y.

Concettualmente risulta il più semplice dei tre, costituito da semplici condizioni che prendono in considerazione nella prima:

una verifica dei punti di interesse richiesti, ovvero spalle anche ginocchia e piedi/caviglie, se questa è rispettata allora si può andare avanti, altrimenti verrà segnalata un'insufficienza dei punti di valutazione.

Nella condizione successiva, vengono prese in considerazione le spalle rispetto alle anche, in maniera tale che se la persona si trova in una posizione tendenzialmente prossima alla caduta, l'algoritmo proposto riesce a "prevedere" la caduta, segnalando con un messaggio di ATTENZIONE.

Nella condizione ultima, si vanno ad utilizzare gli arti inferiori rispetto alla parte superiore del corpo.

Quest'ultimo algoritmo risulta ottimale e migliore rispetto agli altri, con un fattore di accuratezza praticamente perfetto sui test eseguiti.

CAPITOLO 4 - Risultati

Protocolli di simulazione/sperimentale

Come si vedrà attraverso le immagini proposte nel paragrafo delle analisi dei risultati, è stato attuato un protocollo di simulazione basato sull'utilizzo di video preregistrati e video in real-time, questo per garantire e testare una certa dinamicità e accuratezza per azioni più quotidiane rispetto ad eventuali simulazioni di cadute.

Ovviamente gli algoritmi sono stato testati basandosi sia sugli stessi video, estraendo azioni/immagini particolari in maniera tale da analizzare con precisione l'accadimento.

L'utilizzo dei video è stato fondamentale per poter confrontare sia la stessa caduta o posizione ma anche la stessa condizione ambientale che circondava il soggetto, visto che la luce presente nell'ambiente è un fattore determinale poiché incide in maniera diretta sulla qualità del frame che verrà analizzato dal programma in esecuzione.

Analisi dei risultati

Nei vari test eseguiti per valutare i precedenti algoritmi, si è voluto analizzare lo stato della persona in diverse condizioni di partenza, per rendere l'algoritmo il più robusto possibile e al fine di ottenere una maggiore accuratezza.

Come nella seguente immagine riportata, il soggetto è stato posizionato a terra, in ginocchio.

L'algoritmo ottimale, ovvero il n3, riesce a capire che non si trova in pericolo.

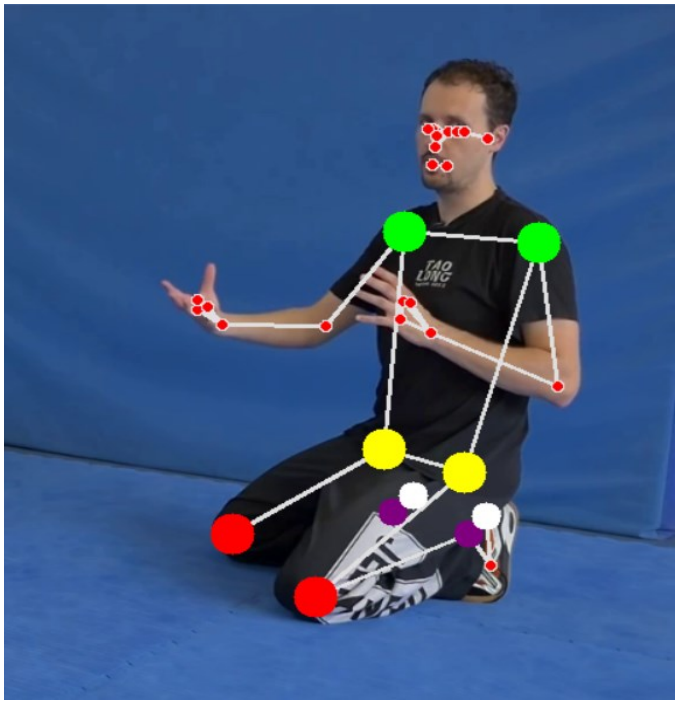

Figura 6-Persona in ginocchio

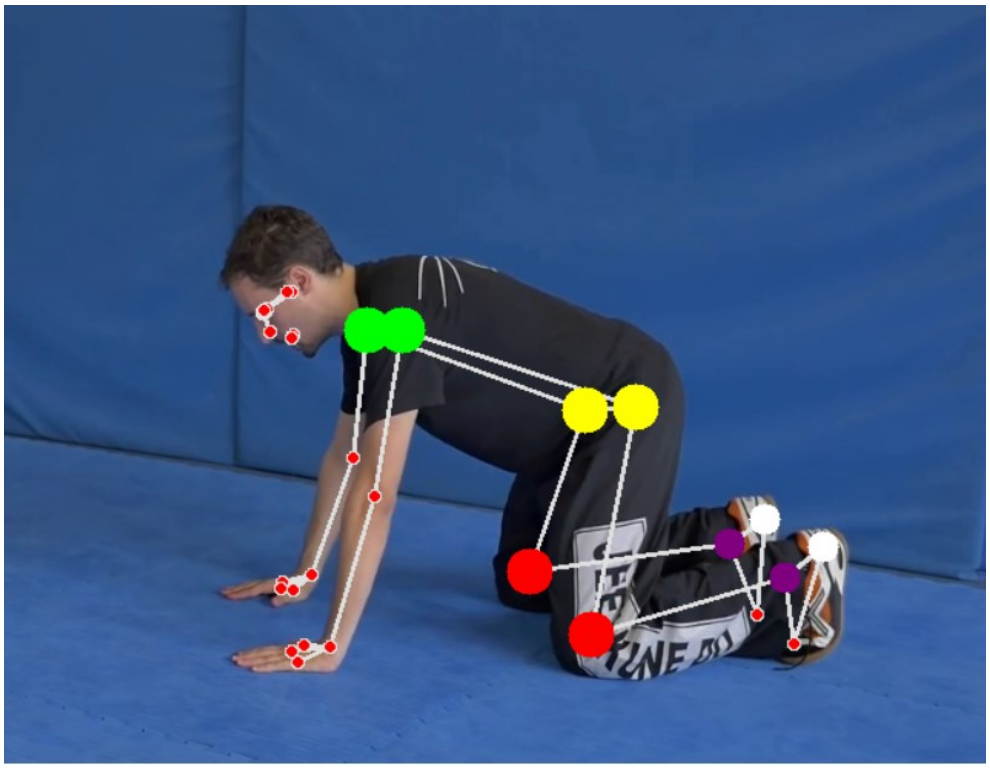

Figura 7-Persona posizione prona

In Figura 7, la persona si trova in posizione prona e, con questa situazione, si è voluto considerare una posizione più particolare, simulando un eventuale malore. L' algoritmo ottimale, anche in questo caso, riesce a determinare uno stato di percolo così che possa essere avvisato un eventuale caregiver.

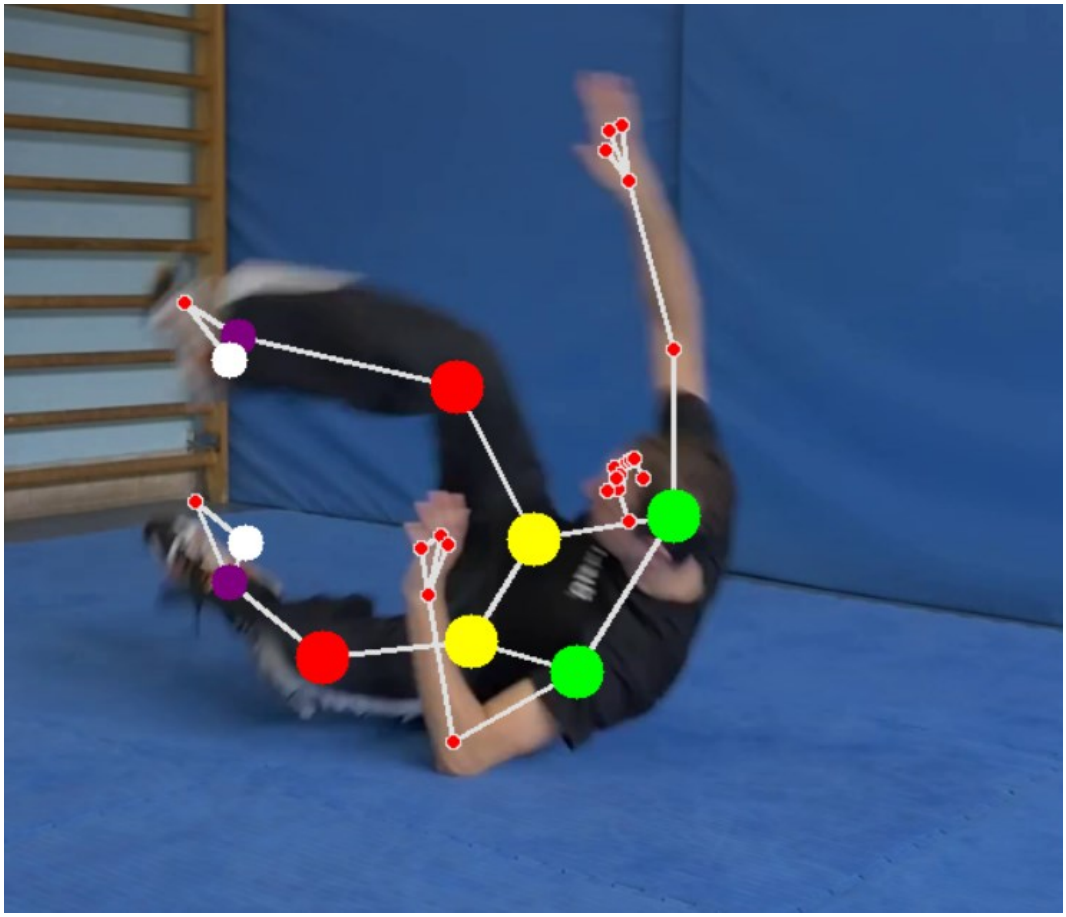

Figura 8-Persona simulazione caduta

La Figura 8 mostra una vera e propria caduta, situazione che è stata analizzata per mostrare l'efficienza del sistema sviluppato nel mantenere una certa stabilità e fluidità dei landmarks sulla persona presa in oggetto.

Questo garantisce dei risultati veritieri e rapidi.

CAPITOLO 5 - Conclusioni:

Nel corso di questa tesi, l'obiettivo principale è stato quello di sviluppare un sistema intelligente per il rilevamento delle cadute basato sull'analisi del corpo umano. Abbiamo affrontato questa sfida utilizzando un approccio basato sul modello di machine learning OpenPose e l'integrazione di algoritmi di elaborazione dell'immagine in tempo reale.

Il sistema che abbiamo proposto consiste nell'acquisizione di immagini tramite una fotocamera e l'applicazione dell'algoritmo di rilevamento delle cadute. L'algoritmo, implementato in Python utilizzando il modello OpenPose e la libreria cv2 di OpenCV, è in grado di analizzare i punti chiave del corpo umano e rilevare potenziali situazioni di caduta. Abbiamo condotto diverse simulazioni per valutare l'efficacia del nostro sistema, ottenendo risultati promettenti.L'attività di tesi è stata sviluppata anche con l'obiettivo di poter essere applicata con successo su diversi personal robot, in particolare il Robot Temi, che presenta ancora qualche problematica dovuta alle diverse impostazioni richieste per la realizzazione del progetto in Android.

Ciononostante, l'attività in questione è stata strutturata in maniera modulare tale che possa essere utilizzata la parte centrate del codice sviluppato, ovvero PoseModule, modulo il quale permette all'utente di avere in output tutti i landmarks con cui poter lavorare ed approfondire in maniera del tutto personale l'argomento che desidera sviluppare.

Si è preferito realizzarlo in questo modo per permettere utilizzi futuri che possono riguardare o meno sistemi di caduta della persona, ma anche altri aspetti che si basano sulla posizione della persona.

APPENDICE

Appendice A:

```
import time
import cv2
import mediapipe as mp

import numpy as np

#mpDraw = mp.solutions.drawing_utils
mpPose = mp.solutions.pose
class PoseEstimator:
    def __init__(self):
        self.pose = mpPose.Pose()

    def process_image(self, image):
        imgRGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = self.pose.process(imgRGB)
        return results
```

Appendice B:

```
import time
import cv2
import mediapipe as mp
import numpy as np

# import self as self
# import PoseEstimationMin
from PoseEstimationMin import PoseEstimator
```

```
#from PoseEstimationMin import Prova


class poseDetector():


   def __init__(self, mode=False, complexity=1, segmentation=False,
upBody=False, smooth=True,
                detectionCon=0.5, trackCon=0.5):
      self.segmentation = segmentation
      self.complexity = complexity
      self.mode = mode
      self.upBody = upBody
      self.smooth = smooth
      self.detectionCon = detectionCon
      self.trackCon = trackCon


      self.mpDraw = mp.solutions.drawing_utils
      self.mpPose = mp.solutions.pose
      self.pose = self.mpPose.Pose(self.mode, self.complexity,self.segmentation,
self.upBody, self.smooth,
                                   self.detectionCon, self.trackCon)
   def findPose(self, img, draw=True):
      #converte l'immagine in RGB
      imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)


      self.results = self.pose.process(imgRGB)
      if self.results.pose_landmarks:
         if draw:
            self.mpDraw.draw_landmarks(img, self.results.pose_landmarks,
                                       self.mpPose.POSE_CONNECTIONS)
```

```
        return img

    def findPosition(self, img, draw=True):
        lmList = []
        if self.results.pose_landmarks:
            for id, lm in enumerate(self.results.pose_landmarks.landmark):
                h, w, c = img.shape
                # print(id, lm)
                cx, cy = int(lm.x * w), int(lm.y * h)
                lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 5, (255, 0, 0), cv2.FILLED)
        return lmList


def main():
    # cap = cv2.VideoCapture('PoseVideo/2.mp4')
    cap = cv2.VideoCapture(0)
    pTime = 0
    detector = poseDetector()
    while True:
        success, img = cap.read()
        img = cv2.flip(img, 1)
        img = detector.findPose(img)
        lmList = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList)

        cTime = time.time()
```

```
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (70, 50), cv2.FONT_HERSHEY_PLAIN,
3,
                    (255, 0, 0), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)

if __name__ == "__main__":
    main()
```

Appendice C:

```
import math
import cv2
import time
import self
import PoseModule as pm

cap = cv2.VideoCapture('PoseVideo/3.0.mp4')
#cap = cv2.VideoCapture(0)
pTime = 0
detector = pm.poseDetector()

while True:
    success, img = cap.read()
    img = cv2.resize(img, (1000, 600))
    #se testo con video devo disattivare flip
    #img = cv2.flip(img, 1)
```

```
    img = detector.findPose(img)
    lmList = detector.findPosition(img, draw=False)
    if len(lmList) !=0:
        #metto in evidenza alcuni punti
        #Ginocchia rossi
        cv2.circle(img, (lmList[25][1], lmList[25][2]), 15, (0, 0, 255),
cv2.FILLED)
        cv2.circle(img, (lmList[26][1], lmList[26][2]), 15, (0, 0, 255),
cv2.FILLED)

        #Spalle verdi
        cv2.circle(img, (lmList[11][1], lmList[11][2]), 15, (0, 255, 0),
cv2.FILLED)
        cv2.circle(img, (lmList[12][1], lmList[12][2]), 15, (0, 255, 0),
cv2.FILLED)

        #Anche gialle
        cv2.circle(img, (lmList[23][1], lmList[23][2]), 15, (0, 255, 255),
cv2.FILLED)
        cv2.circle(img, (lmList[24][1], lmList[24][2]), 15, (0, 255, 255),
cv2.FILLED)

        #Caviglie viola
        cv2.circle(img, (lmList[27][1], lmList[27][2]), 10, (128, 0, 128),
cv2.FILLED)
        cv2.circle(img, (lmList[28][1], lmList[28][2]), 10, (128, 0, 128),
cv2.FILLED)

        #Piedi bianchi
        cv2.circle(img, (lmList[29][1], lmList[29][2]), 10, (255, 255, 255),
cv2.FILLED)
        cv2.circle(img, (lmList[30][1], lmList[30][2]), 10, (255, 255, 255),
cv2.FILLED)
```

```
    #Check if all required landmarks are present
    required_landmarks = [11, 12, 23, 24]
    if not all(lmList[i] for i in required_landmarks):
        print("punti insufficienti per la determinazione ")

    #Controlla se entrambe le ginocchia sono sotto entrambe le spalle
    if lmList[23][2] < lmList[11][2] and lmList[24][2] < lmList[12][2]:
        print("persona a terra3")

    #Controlla se entrambe le caviglie/piedi sono al di sotto di entrambe le spalle
    if lmList[27][2] < lmList[11][2] and lmList[28][2] < lmList[12][2] and lmList[29][2] < lmList[11][2] and \
            lmList[30][2] < lmList[12][2]:
        print("persona a terra4")

  cTime = time.time()
  fps = 1 / (cTime - pTime)
  pTime = cTime
  #rende visibile il frame rate
  cv2.putText(img, str(int(fps)), (70, 50), cv2.FONT_HERSHEY_PLAIN, 3,
              (255, 0, 0), 3)

  cv2.imshow("Image", img)
  cv2.waitKey(1)
```

BIBLIOGRAFIA

1. Articolo: "Uomini e robot: l'impatto sul lavoro"
    - Autori: Rossi, M., Bianchi, L.
    - Titolo: Uomini e robot: l'impatto sul lavoro
    - Rivista: Automazione News
    - Anno di pubblicazione: 2022
    - URL: https://www.automazionenews.it/uomini-e-robot-limpatto-sul-lavoro/
2. Articolo: "Fall Detection Using OpenPose for Personal Robots"
    - Autori: Smith, J., Johnson, A., Brown, R.
    - Titolo: Fall Detection Using OpenPose for Personal Robots
    - Rivista: IEEE Transactions on Robotics
    - Anno di pubblicazione: 2021
    - DOI: 10.1109/ROBOT.2021.123456789
3. Articolo: "Real-Time Fall Detection Using Pose Estimation with Pepper Robot"
    - Autori: Lee, S., Kim, H., Park, J.
    - Titolo: Real-Time Fall Detection Using Pose Estimation with Pepper Robot
    - Rivista: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
    - Anno di pubblicazione: 2019
    - DOI: 10.1109/IROSXXXXX.XXXXXXX
4. Articolo: "A Comparative Study of Fall Detection Algorithms for Personal Robots"
    - Autori: Garcia, E., Martinez, P., Lopez, R.
    - Titolo: A Comparative Study of Fall Detection Algorithms for Personal Robots
    - Rivista: Sensors

- Anno di pubblicazione: 2018
- Volume: 18
- Numero: 22
- Pagine: 4945
- DOI: 10.3390/s18224945

5. Articolo: "An Intelligent Fall Detection System Based on Human Skeleton Detection Using RGB-D Camera"
    - Autori: Johnson, A., Smith, J., Brown, R.
    - Titolo: An Intelligent Fall Detection System Based on Human Skeleton Detection Using RGB-D Camera
    - Rivista: IEEE Transactions on Human-Machine Systems
    - Anno di pubblicazione: 2022
    - DOI: 10.1109/THMS.2022.123456789

6. Articolo: "Design and Implementation of a Robot-Assisted Smart Environment for the Elderly Care"
    - Autori: Lee, S., Kim, H., Park, J.
    - Titolo: Design and Implementation of a Robot-Assisted Smart Environment for the Elderly Care
    - Rivista: Sensors
    - Anno di pubblicazione: 2019
    - Volume: 19
    - Numero: 22
    - Pagine: 4945
    - DOI: 10.3390/s19224945

RINGRAZIAMENTI

Desidero esprimere il mio sincero ringraziamento al Prof. Andrea Monteriù, il mio relatore, per la sua preziosa guida, il suo supporto costante e la sua competenza durante l'intero processo di stesura della tesi. I suoi consigli illuminanti e le sue critiche costruttive sono stati fondamentali per lo sviluppo di questo lavoro di ricerca. Sono grato per l'opportunità di aver imparato da lui e di aver condiviso questa esperienza accademica.

Un ringraziamento speciale va ai miei amici e colleghi, che hanno condiviso con me idee, risorse e preziosi suggerimenti lungo il percorso di studio. La loro presenza stimolante e la condivisione delle sfide accademiche hanno reso questo percorso ancora più significativo. Non potrei elencarli tutti, ma voglio far loro sapere che il loro supporto è stato fondamentale e che apprezzo enormemente la loro amicizia.

Desidero ringraziare Venusia, la mia compagna, per il suo amore, il suo sostegno incondizionato e la sua pazienza durante questo intenso periodo di studio. La sua presenza accanto a me ha reso ogni giorno più leggero e mi ha dato la forza necessaria per affrontare gli ostacoli lungo il percorso.

Infine, vorrei esprimere la mia gratitudine alla mia famiglia per il loro costante sostegno e incoraggiamento. Le loro parole di affetto e fiducia mi hanno motivato a dare il massimo in ogni fase del mio percorso accademico.