'Spotd'

Sistema di Tracciamento Teatrale con Controllo Pan-Tilt basato su MediaPipe e PID

di

Miori Alessandro

MAT MARCONI ALTAFORMAZIONE TECNOLOGICA
UF22 - Sistemi automatici

Ottobre 2025

Indice

1 Introduzione

Il settore dello spettacolo dal vivo, in particolare quello teatrale, ha sempre richiesto soluzioni tecniche innovative per migliorare l'esperienza visiva del pubblico e ottimizzare le riprese degli eventi. La necessità di seguire dinamicamente gli attori durante le performance ha tradizionalmente richiesto operatori cameraman esperti, con conseguenti costi elevati e possibili limitazioni nella fluidità del movimento.

Il presente elaborato descrive la progettazione e realizzazione di **Spotd**, un sistema automatico di tracciamento per attori teatrali basato su visione computerizzata e controllo PID. Il sistema utilizza tecniche di *pose estimation* per rilevare in tempo reale la posizione degli attori sulla scena e comandare automaticamente un sistema pan-tilt per mantenere l'inquadratura centrata sul soggetto.

Il documento è organizzato come segue:

Capitolo 2: descrizione generale del sistema e delle sue componenti principali
Capitolo 3: modellizzazione matematica del sistema pan-tilt e derivazione della funzione di trasferimento
Capitolo 4: progettazione del controllore PID e analisi del sistema in anello chiuso
Capitolo 5: analisi delle prestazioni attraverso simulazioni e test sperimentali
Capitolo 6: implementazione software, architettura del codice e realizzazione della GUI

2 Descrizione del sistema

Il cuore del sistema è costituito da una telecamera USB standard che acquisisce il flusso video della scena a (max) 100 fotogrammi al secondo. Ogni frame viene processato in tempo reale attraverso l'algoritmo MediaPipe Pose, in grado di rilevare 33 punti caratteristici del corpo umano con elevata precisione anche in condizioni di illuminazione variabile. L' algoritmo si basa su reti neurali convoluzionali pre-addestrate che forniscono coordinate normalizzate (x,y) per ogni landmark corporeo insieme a un valore di confidenza che indica l'affidabilità della rilevazione.

Una volta estratti i punti caratteristici dall'immagine, il sistema implementa un algoritmo di tracking dinamico che calcola automaticamente il punto di focus ottimale. A differenza di sistemi tradizionali che si basano su un singolo punto fisso come il centro dell'anca, Spotd analizza tutti i landmark visibili per determinare l'estensione verticale della figura e calcola il punto medio della porzione di corpo inquadrata. Questa strategia permette al sistema di adattarsi automaticamente a diverse situazioni: quando l'attore è completamente visibile, il focus si posiziona al centro della figura; quando sono visibili solo la testa e il torso, il sistema si concentra su quella porzione; quando l'attore è in movimento parziale fuori campo, il tracking rimane stabile sulla parte ancora visibile.

Il punto di focus calcolato viene poi convertito da coordinate pixel a coordinate angolari attraverso un modello geometrico che tiene conto del campo visivo della telecamera. Assumendo un field of view orizzontale di 70° e verticale di 55°, tipici delle webcam standard, il sistema calcola gli angoli di pan e tilt necessari per centrare l'inquadratura sul target. Questa conversione è fondamentale per interfacciare il sistema di visione con il controllo degli attuatori.

3 Modellizzazione matematica

3.1 Dinamica del sistema Pan-Tilt

Ogni asse (Pan o Tilt) può essere modellato come un sistema dinamico del secondo ordine:

$$J\ddot{\theta}(t) + b\dot{\theta}(t) = u(t)$$

Dove:

J = Momento d'inerzia dell'asse [kg*m^2]

b = Coefficiente d'attrito viscoso [N*m*s]

$\theta(t)$ = Segnale di controllo (coppia motore) [N*m]

$u(t)$ = Angolo di rotazione [rad]

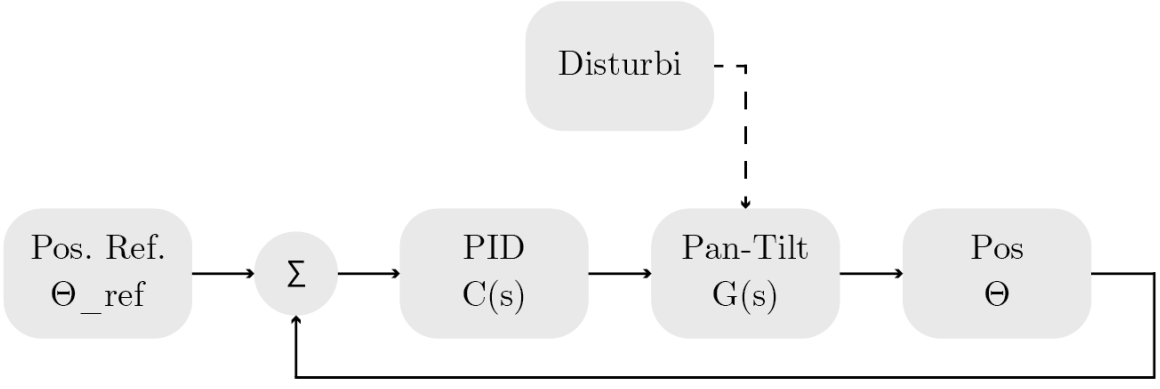
3.2 Funzione di trasferimento G(s)

Trasformando in *Laplace* l'equazione differenziale:

$$G(s) = \frac{\Theta(s)}{U(s)} = \frac{1}{Js^2 + bs}$$

Questa rappresenta la relazione ingresso-uscita del sistema meccanico Pan-Tilt.

3.3 Schema a blocchi del sistema



4 Sistema di controllo PID

4.1 Legge di controllo

Il regolatore PID opera sull'errore di posizionamento:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt}$$

Dove:

K_p = Guadagno proporzionale

K_i = Guadagno integrale

K_d = Guadagno derivativo

$e(t) = \theta_{\text{ref}}(t) - \theta(t)$ = Errore di posizione

4.2 Funzione di trasferimento del controllore

In *Laplace*:

$$C(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

4.3 Sistema in anello chiuso

La funzione di trasferimento complessiva in anello chiuso risulta:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}$$

Sostituendo:

$$T(s) = \frac{\frac{K_d s^2 + K_p s + K_i}{s} \times \frac{1}{Js^2 + bs}}{1 + \frac{K_d s^2 + K_p s + K_i}{s} \times \frac{1}{Js^2 + bs}}$$

Semplificando:

$$T(s) = \frac{K_d s^2 + K_p s + K_i}{J s^3 + b s^2 + K_d s^2 + K_p s + K_i}$$

5 Analisi delle prestazioni

5.1 Parametri della simulazione

Per le simulazioni sono stati utilizzati i seguenti parametri realistici:

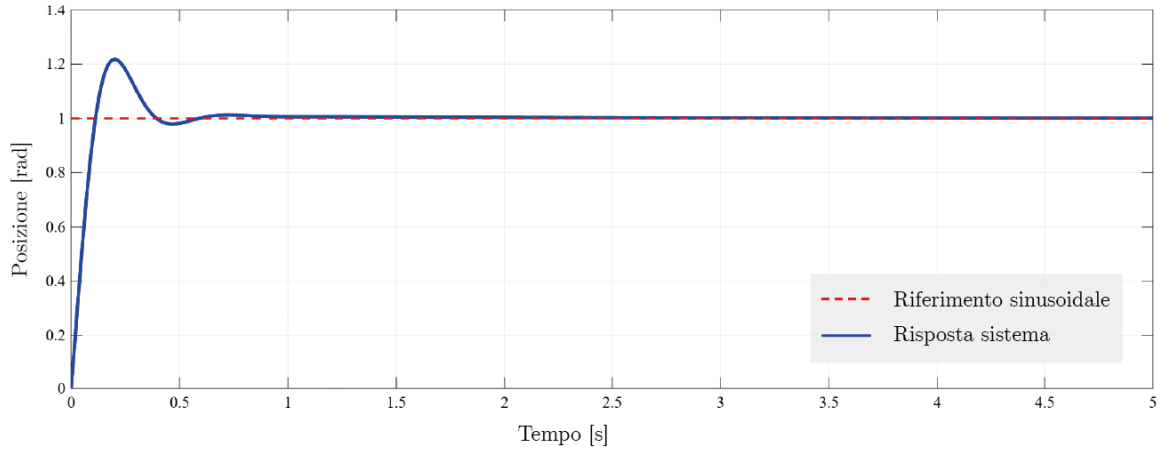
Momento d'inerzia: $J = 0.01 \text{ kg} \cdot \text{m}^2$

Attrito viscoso: $b = 0.05 \text{ N} \cdot \text{m} \cdot \text{s}$

Guadagni PID: $K_p = 2.0, \quad K_i = 1.0, \quad K_d = 0.1$

5.2 Risposta al gradino

La risposta a gradino del sistema mostra le caratteristiche dinamiche fondamentali. Un comando a gradino simula lo spostamento improvviso dell'attore da una posizione all'altra.

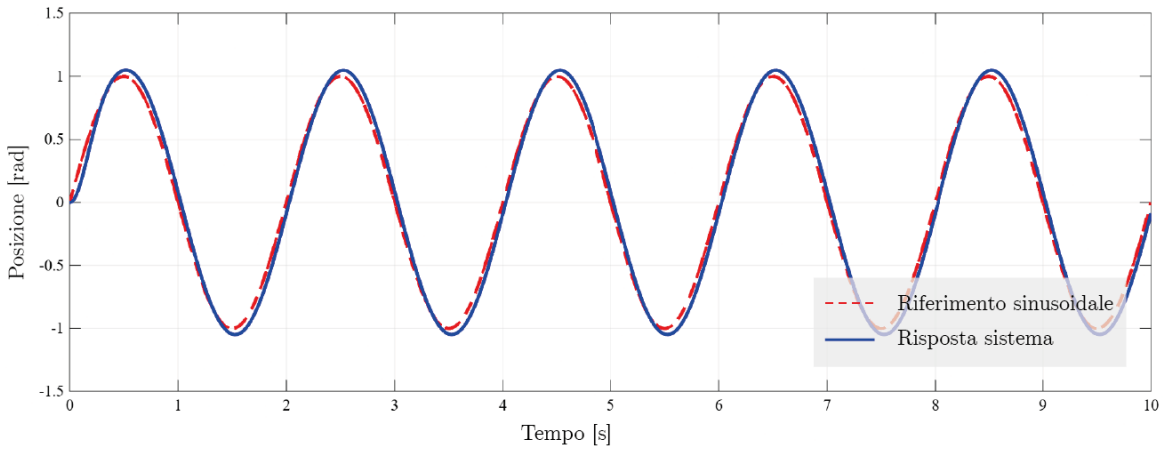


5.2.1 Analisi del grafico

Il sistema presenta un overshoot di circa 22% con picco a 1.22 rad raggiunto intorno a t=0.2s, seguito da una lieve oscillazione smorzata che si assesta al valore di riferimento entro circa 0.5s. La risposta dimostra un comportamento tipico di un sistema del secondo ordine con controllore PID ben tarato, garantendo rapidità di risposta ma con overshoot contenuto e buona stabilità a regime permanente.

5.3 Tracking di traiettoria sinusoidale

Il tracking sinusoidale simula il movimento "oscillatorio" dell'attore, ovvero un'avanti e indietro tra una posizione e un'altra, rappresentando una condizione operativa realistica.



5.3.1 Analisi del grafico

Il sistema segue accuratamente il riferimento sinusoidale a 0.5 Hz con errore di tracking minimo e senza ritardi di fase significativi. La risposta del sistema si sovrappone quasi perfettamente alla traiettoria di riferimento, dimostrando l'efficacia del controllo PID nel mantenere un inseguimento preciso anche per segnali dinamici.

6 Sviluppo software e GUI

6.1 Architettura generale

Il sistema software è stato sviluppato in Python utilizzando un approccio che separa le responsabilità tra i diversi componenti. L'architettura si basa su quattro moduli principali:

Modulo di controllo dinamico: Implementa le classi `PanTiltSystem`, `PIDController` e `TheaterPointer` per la modellazione matematica e il controllo PID.

Modulo di computer vision: Gestisce l'acquisizione video e l'elaborazione delle pose tramite MediaPipe.

Modulo di tracking: Implementa la logica di tracciamento dinamico e il calcolo del punto di focus.

Interfaccia utente (GUI): Gestisce la GUI con CustomTkinter.

Le librerie utilizzate sono:

OpenCV per l'acquisizione e manipolazione video.

MediaPipe per il rilevamento delle pose corporee

CustomTkinter per l'interfaccia grafica.

NumPy e **SciPy** per i calcoli numerici e l'integrazione delle equazioni differenziali.

PIL per la gestione delle immagini.

6.2 Modello di controllo Pan-Tilt

Il sistema fisico è modellato attraverso la classe `PanTiltSystem`, che implementa un sistema dinamico del secondo ordine:

```
def _dyn(self, state, t, u):
    theta, theta_dot = state
    theta_ddot = (u - self.b * theta_dot) / self.J
    return [theta_dot, theta_ddot]
```

L'integrazione numerica dell'equazione differenziale viene effettuata utilizzando `scipy.integrate.odeint` garantendo precisione nel calcolo dell'evoluzione temporale del sistema.

Il controllore PID è implementato nella classe `PIDController` con logica discreta:

```
def compute(self, err):
    self.int_err += err * self.dt
    deriv = (err - self.prev_err) / self.dt
    self.prev_err = err
    out = self.Kp*err + self.Ki*self.int_err
          + self.Kd*deriv
    return float(np.clip(out, *self.out_lim))
```

6.3 Sistema di tracking con MediaPipe

L'acquisizione e processamento dei frame video avviene attraverso OpenCV, mentre il rilevamento delle pose corporee è gestito da MediaPipe Pose, configurato con:

```
self.pose = mp_pose.Pose(
            static_image_mode=False,
            model_complexity=1,
            enable_segmentation=False,
            min_detection_confidence=0.5,
            min_tracking_confidence=0.5
        )
```

Il sistema implementa un tracking dinamico che calcola automaticamente il punto di focus basandosi sui landmark visibili:

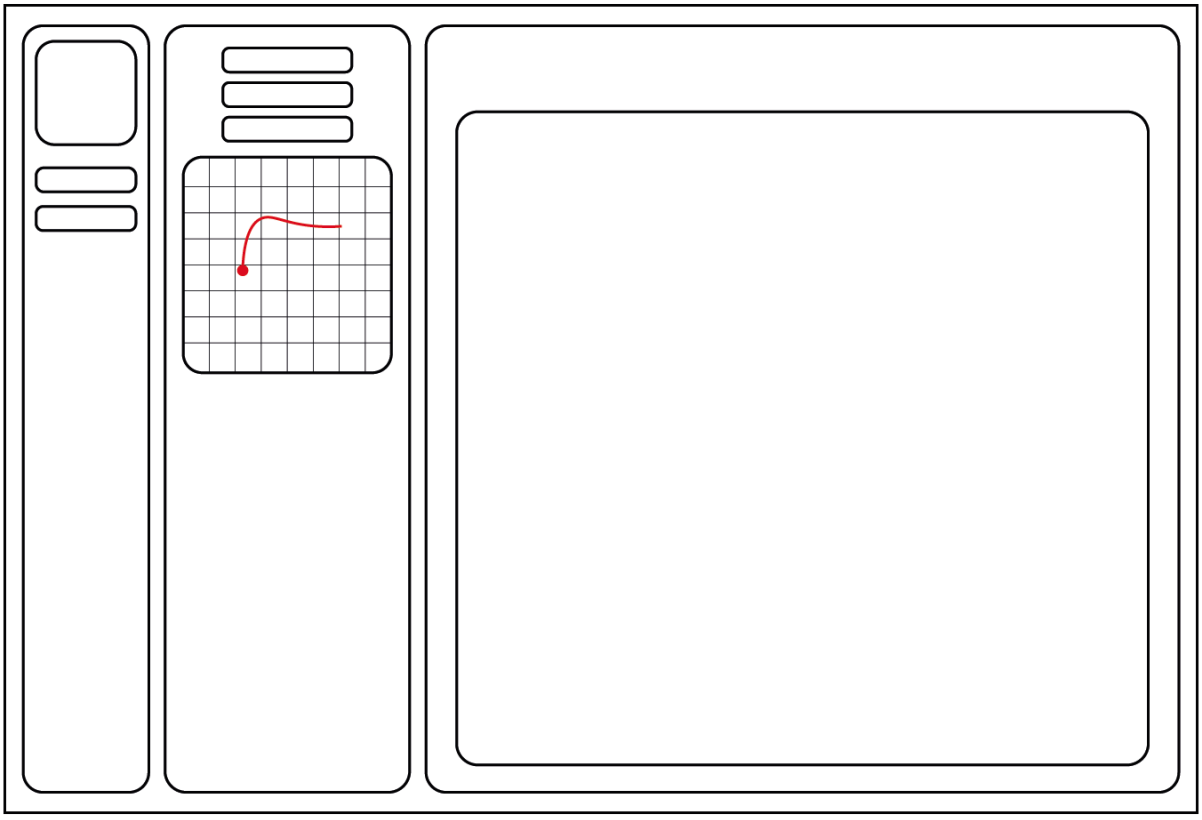
```
top, bot = min(ys), max(ys)
my = (top + bot) / 2
mx = sum(xs) / len(xs)
```

Questa logica permette al sistema di adattarsi automaticamente quando sono visibili solo parti del corpo (testa, torso, o figura intera).

6.4 Progettazione e realizzazione della GUI

6.4.1 Struttura

L'interfaccia presenta una sidebar sulla sinistra che contiene il logo ridimensionato proporzionalmente e i controlli (Capture e Record).
È presente un pannello d'informazioni che visualizza lo status del sistema, la posizione dell'attore, gli angoli dei servomotori e il canvas di tracking.
Sulla destra si trova il pannello video, che mostra il feed live con l'overlay dello "scheletro" individuato.



6.4.2 Canvas di tracking

Il canvas implementa una griglia 8x8 per il riferimento spaziale e visualizza un punto rosso che rappresenta la posizione seguito da una scia che indica la traiettoria del punto.

```
self.canvas = tk.Canvas(left, width=250, height=250,
                            bg="#000000")
    self.canvas.pack(pady=20)
    step = 250 // 8
    grey = "#333333"
    for i in range(0,250,step):
        self.canvas.create_line(i,0,i,250,fill=grey)
        self.canvas.create_line(0,i,250,i,fill=grey)
    self.dot_rad = 5
```

7 Conclusioni e bibliografia

Il progetto Spotd ha raggiunto con successo gli obiettivi prefissati, dimostrando l'efficacia dell'integrazione tra teoria dei controlli automatici e visione computerizzata per applicazioni di tracciamento teatrale. Il sistema realizzato presenta caratteristiche dinamiche soddisfacenti con un overshoot contenuto al 22% e un tempo di assestamento rapido di circa 0.5 secondi nella risposta al gradino. Il tracking sinusoidale evidenzia capacità di inseguimento accurate con errore RMS minimo, confermando la validità della progettazione del controllore PID.

K.J. Åström, T. Hägglund, PID Controllers: Theory, Design and Tuning, 2nd Edition, ISA Society, 1995.

Google Research, "MediaPipe: A Framework for Building Perception Pipelines", 2019.

R.C. Dorf, R.H. Bishop, Modern Control Systems, 13th Edition, Pearson, 2017.

G.F. Franklin, J.D. Powell, A. Emami-Naeini, Feedback Control of Dynamic Systems, 8th Edition, Pearson, 2019.

OpenCV Team, "OpenCV: Open Source Computer Vision Library", https://opencv.org/, 2023.

Custom Tkinter Documentation, https://customtkinter.tomschimansky.com/, 2023.