Integrazione di Alexa con Pepper Per Fornire Assistenza a Pazienti Ospedalieri

Luigi Emanuele Sica
0522501530
Curriculum Data Science

Nicola Pio Santorsa
0522501434
Curriculum Data Science

Francesco Paciello
0522501558
Curriculum Sicurezza

***Index Terms*—SLAM, Pepper, Alexa, Face Detection, Face Recognition, Robot navigation**

***Abstract*—Nel corso di questa ricerca, ci siamo focalizzati sull'opportunità di impiegare il robot Pepper come strumento di supporto nell'ambito dell'assistenza ospedaliera. Pur essendo consapevoli del contesto simulato.**

Per raggiungere il nostro obiettivo, abbiamo implementato un sistema di comunicazione utilizzando l'assistente vocale di Alexa, consentendo ai pazienti e al personale ospedaliero di interagire con il robot in modo intuitivo. Questo ha consentito di acquisire informazioni sulle stanze ospedaliere e i pazienti utilizzando l'assistente Alexa.

Attraverso Alexa, il personale ospedaliero può indicare al robot Pepper le stanze in cui è necessario prestare assistenza. Queste informazioni vengono memorizzate su un database NoSQL (Firebase) e successivamente trasmesse in modo intuitivo e diretto al robot, consentendogli di comprenderle e agire di conseguenza.

A supporto di ciò il robot crea una mappa esplorando l'ambiente, in modo da catturare la disposizione degli spazi, identificando le posizioni significative e catturando le coordinate geografiche di interesse (stanze).

Una volta raggiunta la stanza, utilizzando le telecamere integrate nel robot Pepper, le immagini vengono acquisite e fornite come input ad un algoritmo di riconoscimento facciale. Questo algoritmo analizza le caratteristiche facciali e cerca di rilevare la presenza o l'assenza di un paziente specifico nella stanza. Il tutto reso possibile da una precedente fase di registrazione del paziente necessaria per l'identificazione successiva.

Una volta che il sistema di riconoscimento facciale ha completato l'analisi, le informazioni vengono trasmesse al robot Pepper che provvede a fornire le informazioni rilevanti al personale medico, come l'informazione sulla presenza del paziente nella stanza, fornendo dettagli aggiuntivi come l'identità del paziente o altre informazioni rilevanti.

I. INTRODUZIONE

L'importanza di questo studio risiede nella ricerca di soluzioni innovative per migliorare l'efficienza e la precisione nella fornitura dell'assistenza ospedaliera. L'utilizzo del robot Pepper come assistente ospedaliero potrebbe rappresentare una soluzione promettente per affrontare sfide come la comunicazione intuitiva con i pazienti e il monitoraggio delle stanze.

La motivazione di questa ricerca è quindi quella di sviluppare e dimostrare l'efficacia dell'utilizzo del robot Pepper come assistente ospedaliero, attraverso l'utilizzo di soluzioni tecnologiche innovative. L'obiettivo è fornire un supporto efficace ed efficiente ai pazienti e al personale medico, ottimizzando le attività di comunicazione, monitoraggio e assistenza.

L'introduzione del robot Pepper come strumento di supporto offre diverse opportunità per migliorare l'esperienza dei pazienti e semplificare le attività del personale ospedaliero.

L'impiego di tecnologie come la libreria *face_recognition* e l'integrazione con Alexa offre vantaggi significativi. La libreria *face_recognition* permette al robot di identificare i volti dei pazienti e di fornire queste informazioni al personale medico, facilitando il monitoraggio e migliorando la sicurezza dei pazienti. L'integrazione con Alexa consente una comunicazione più intuitiva e diretta tra il robot, i pazienti e il personale ospedaliero, semplificando le interazioni e ottimizzando il flusso di lavoro.

Allo stesso tempo, l'impiego del robot Pepper come assistente ospedaliero presenta alcune sfide da affrontare. Prima di tutto, è necessario garantire l'integrazione e l'adattamento del robot all'ambiente ospedaliero, personalizzando il software e formando adeguatamente il personale. La privacy e la sicurezza dei dati sono fondamentali, specialmente nel contesto del riconoscimento facciale, richiedendo misure di protezione adeguate. L'accettazione e la fiducia del robot da parte dei pazienti e del personale sono importanti, richiedendo una comunicazione chiara dei benefici e dimostrazioni dell'affidabilità del robot. Inoltre, l'interazione efficace con i comandi vocali e la comprensione delle variazioni linguistiche possono essere un'ulteriore sfida.

Di seguito analizzeremo nella sezione II lo stato dell'arte in cui introdurremo le varie applicazioni del robot Pepper, nella sezione III presenteremo le tecnologie utilizzate a supporto del nostro lavoro, nella sezione IV analizzeremo le varie problematiche e limitazioni che abbiamo affrontato durante lo sviluppo di questa applicazione, infine nella sezione V trarremo le nostre conclusioni.

II. STATO DELL'ARTE

Nel presente capitolo, esploreremo varie ricerche e studi presenti in letteratura che riguardano l'utilizzo del robot Pepper in diversi ambiti. Il robot Pepper, sviluppato dalla SoftBank Robotics, ha suscitato un grande interesse nella comunità scientifica e industriale grazie alla sua capacità di interagire in modo naturale con gli esseri umani e di svolgere una vasta gamma di compiti. Presenteremo una panoramica dei lavori che hanno esplorato l'utilizzo di Pepper nella robotica sociale, nell'interazione uomo-macchina e in altri ambiti. In [6] gli autori descrivono un utilizzo educativo di Pepper. Viene

presentata l'applicazione di due concetti, quello di "care-receiving robot" (CRR) e di "total physical response" (TPR) , nel design di un'applicazione educativa che utilizza Pepper. Si offre uno scenario in cui i bambini imparano insieme a Pepper nell'ambiente domestico, grazie a un insegnante umano che tiene le lezioni da una classe remota. Nel paper viene mostrato il processo di sviluppo dell'applicazione, che contiene tre programmi educativi che i bambini possono selezionare per interagire con Pepper. Vengono inoltre descritti i feedback e le conoscenze ottenute dagli esperimenti di prova. In sostanza, l'articolo presenta un esempio di come Pepper possa essere utilizzato come strumento educativo per i bambini, combinando concetti di assistenza robotica e una risposta fisica totale. L'obiettivo è quello di fornire un'esperienza di apprendimento coinvolgente e interattiva, consentendo ai bambini di imparare da un insegnante umano anche se si trovano a distanza. I risultati hanno dimostrato come molti bambini sono stati confusi su chi fosse il proprio interlocutore, se fosse Pepper oppure l'insegnante reale. Un'altro risultato emerso è quello che molto spesso i bambini rispondevano in modo differente da ciò che Pepper si aspettava. Altro lavoro importante è stato svolto dagli autori in [4], difatti viene presentato un software sviluppato per il robot Pepper, adatto all'interazione uomo-robot nell'ambito dell'edutainment (educazione e intrattenimento). La combinazione di educazione e intrattenimento è ottenuta modificando un sistema di intelligenza artificiale conversazionale all'avanguardia e sviluppando diverse applicazioni interattive a quiz per il robot, implementandole in Pepper. Nel paper, vengono descritti i dettagli tecnici dell'implementazione del chatbot e viene fornita anche una descrizione del software per l'edutainment. Gli autori hanno sviluppato un software per consentire a Pepper di interagire con gli esseri umani in modo educativo e divertente fornendo dei quiz divertenti interattivi ed utilizzando i movimenti di Pepper come feedback alle risposte dell'utente, rendendolo così più coinvolto durante l'esperienza di apprendimento. Nello studio [1], è stata esplorata l'efficacia della terapia di stimolazione cognitiva utilizzando Pepper con anziani affetti da disturbi cognitivi. Attraverso un programma di 3 settimane, è stato osservato che il robot Pepper è stato in grado di suscitare reazioni affettive positive negli anziani, riuscendo a creare delle vere e proprie conversazioni ad hoc con il paziente, personalizzate in base all'utente che ha di fronte. Quindi è stato sviluppato un sistema di analisi automatica delle espressioni facciali per personalizzare l'interazione e i compiti del robot in base alle reazioni individuali degli utenti. Il seguente lavoro riguarda l'utilizzo del Robot Pepper adibito a fornire ai clienti di un Chocolate Shop in Belgio, un'esperienza coinvolgente. Così come nel lavoro precedente [4], gli autori in [2] vanno ad offrire dei quiz interattivi tramite il tablet di Pepper, facendolo parlare e/o fornendo feedback alle risposte attraverso i movimenti del robot. Gli autori hanno confrontato un semplice tablet con quello offerto da Pepper, in particolare hanno partecipato 307 persone, delle quali 77 non utilizzavano alcun tablet, altre 77 utilizzavano il tablet semplice, 78 utilizzavano Pepper fornito di voce, altre 78

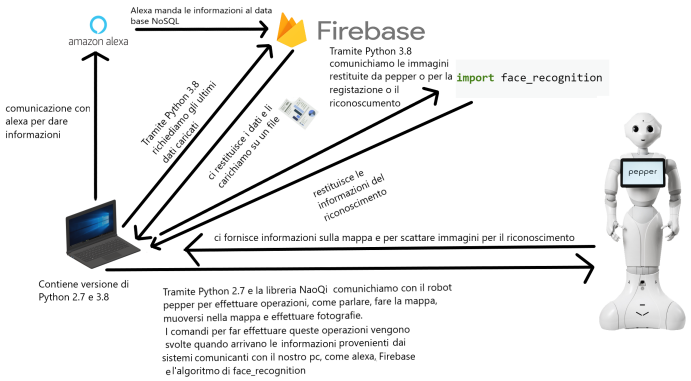

Fig. 1. Architettura Applicazione

utilizzavano Pepper senza la capacità di parlare. I risultati mostrano che gli utenti che utilizzavano Pepper sono stati più invogliati ad entrare nel negozio. Infine nel paper [5] viene presentato un approccio per consentire al robot Pepper di fornire tour automatizzati e interattivi di un laboratorio di robotica. Il lavoro si concentra su come superare le limitazioni tecniche del robot, come le capacità sensoriali limitate e la difficoltà di navigazione in ambienti complessi. Gli autori presentano una serie di contributi, tra cui un ambiente di sviluppo virtuale aggiornato, un controllore di movimento personalizzato, un metodo per la localizzazione e la navigazione nel laboratorio. Inoltre, descrivono come il robot Pepper viene integrato con altri dispositivi nel laboratorio per offrire un'esperienza più interattiva durante i tour. Il sistema utilizzato per controllare Pepper a livello di navigazione è stato ROS (Robot Operating System), il quale combinato con NaoQI, offre a Pepper la migliore esperienza di movimento.

III. METODOLOGIA

Nella seguente sezione, esamineremo le metodologie utilizzata per il progetto del robot Pepper, che integra le funzionalità di comunicazione con Alexa e il riconoscimento facciale. Questa architettura vedere Figura 1, permette a Pepper di interagire con gli utenti attraverso comandi vocali, ricevere informazioni da Alexa e individuare volti specifici all'interno di un ambiente.

A. Skill Alexa

L'integrazione di Alexa con Pepper è stata realizzata utilizzando l'SDK di Alexa. Le informazioni rilevanti, fornite dagli utenti attraverso i comandi vocali dati ad Alexa, vengono immagazzinate in un database, in quanto non è possibile far comunicare direttamente il robot Pepper con Alexa. Per implementare questa funzionalità, abbiamo interagito con Firebase, un servizio di database NoSQL basato su cloud. Importando tale libreria nell'ambiente di sviluppo della skill di Alexa, ha permesso di salvare quello che viene detto da Alexa nel database. Estrarre le informazioni da Firebase comunicate ad

Alexa, per poi poterle comunicarle al robot Pepper, ha portato ad affrontare un problema, dovuto all'incompatibilità tra la versione di Python 2.7 utilizzata da Pepper e il servizio Firebase, che richiede la versione di Python 3.X per poter interagire. Per superare questa problematica, abbiamo implementato un codice Python 3.8 per recuperare i dati da Firebase nel momento in cui viene effettuata una nuova inserzione nel nostro database e salvarli su un file. Successivamente, utilizzando Python 2.7 che ci permette di interagire con la libreria NaoQI, leggiamo le informazioni dal file, consentendo così a Pepper di agire(muoversi nella stanza) in base ai dati letti nel file. Questa soluzione ha permesso di gestire l'incompatibilità tra le versioni di Python e garantire il passaggio di informazioni al robot per intraprendere le azioni.

B. Navigazione nell'ambiente

Per consentire a Pepper di raggiungere il punto di interesse indicato dalle informazioni fornite da Alexa, abbiamo utilizzato la libreria NAOqi. L'utilizzo della libreria NAOqi ha fornito a Pepper le funzionalità necessarie per il movimento all'interno dell'ambiente specifico, infatti tramite il metodo *explore()* il robot Pepper usa i sui sensori per modellare l'ambiente circostante e creare la mappa di grandezza prestabilita tramite il parametro "raggio di esplorazione", come mostra la Figura 2. Successivamente ricaviamo le coordinate dei punti desiderati sulla mappa tramite il metodo *getRobotPositionInMap()* che una volta caricata la mappa definita in precedenza e localizzato il robot sul punto di partenza, ci ha permesso di raccogliere le coordinate muovendo il robot nella mappa nella zona di nostro interesse. Infine utilizzando il metodo *navigateToInMap()*, dandogli le coordinate raccolte permette a pepper di navigare verso i punti di interesse. Dopo aver creato la mappa e individuato le coordinate dei punti di interesse (stanze), abbiamo associato le informazioni vocali degli utenti alle coordinate appropriate. Ad esempio, se chiediamo ad Alexa di "controllare il paziente Rossi", Pepper si sposterà nella stanza corrispondente alle coordinate che rappresenta la posizione della stanza di Rossi.

C. Face Detection e Recognition

Una volta che il robot Pepper raggiunge la stanza, viene eseguita una fase di riconoscimento facciale utilizzando la libreria *face_recognition* per verificare la presenza di pazienti, questo tipo di riconoscimento viene svolto sul PC (server-side) e non su Pepper, Pepper provvede solo a fornirci le immagini.

La libreria *face_recognition* è stata creata da Adam Geitgey, un ingegnere del software, ed è un progetto open source. Originariamente, Adam Geitgey ha sviluppato questa libreria come un "wrapper" intorno alla libreria dlib, che è una libreria di machine learning in C++ utilizzata per la visione artificiale [3].

La tecnologia utilizzata dalla libreria *face_recognition* si basa su algoritmi di deep learning e visione artificiale per identificare e riconoscere i volti nelle immagini. La libreria sfrutta modelli di machine learning addestrati su ampi dataset di volti per rilevare e confrontare le caratteristiche facciali.

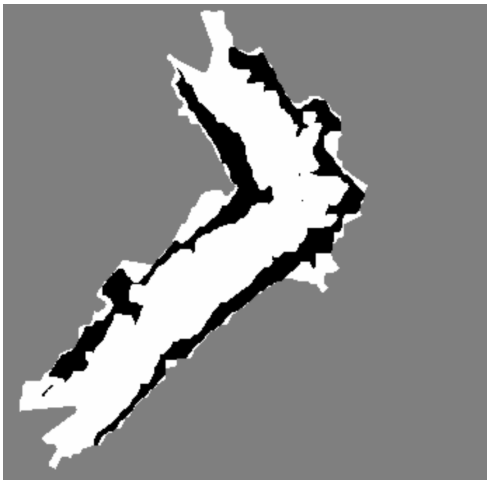

Fig. 2. Mappa generata da Pepper

Ovviamente, prima che Pepper possa riconoscere un paziente specifico, è necessario acquisire tramite la telecamera del robot Pepper l'immagine del volto del paziente e caricarla su una cartella. Questa cartella viene passata all'algoritmo che inizia una fase di registrazione che consente all'algoritmo di *face_recognition* di memorizzare e apprendere i volti e le caratteristiche facciali che le contraddistinguono, per consentire il riconoscimento futuro.

Durante il processo di addestramento, il modello di deep learning analizza le immagini di addestramento e cerca di estrarre le caratteristiche facciali rilevanti per l'identificazione dei volti. Questi tratti includono posizione degli occhi, contorni del viso, dimensione delle labbra, disposizione dei capelli, ecc.

Il modello viene addestrato attraverso iterazioni multiple, utilizzando tecniche di apprendimento profondo come reti neurali convoluzionali (CNN) o algoritmi specifici come il metodo "Viola-Jones" [7]. Durante l'addestramento, il modello ottimizza i suoi parametri interni per migliorare le prestazioni di riconoscimento dei volti.

All'arrivo nella stanza, il robot Pepper prova a rilevare la presenza di un volto, e una volta individuato lo segue con il movimento della testa in modo da non perderlo di vista. Successivamente tramite l'acquisizione delle immmagini con la telecamera del robot Pepper, queste ultime vengono passate all'algoritmo di *face_recognition* (esso gira su un programma Python 3.8 sempre per problemi di incompatibilità, dovuta al basso supporto delle librerie di Python 2.7, e quindi ad una bassa interazione diretta con quest'ultima), che alla ricezione delle immagini, in base alla fase di registrazione fatta in precedenza effettua un'analisi e confronta le caratteristiche facciali dei volto presente nella foto con quelli catturati durante la fase di riconoscimento per determinare se il paziente rilevato è presente. Se riconosce il volto, provvede a segnalarlo al personale medico, indicando che il paziente è in stanza.

IV. LIMITAZIONI

Qui di seguito commenteremo le varie problematiche riscontrate durante lo sviluppo del seguente progetto. Per una demo

sul funzionamento è possibile guardare il seguente video.

A. SLAM

L'acronimo **SLAM** sta per *Simultaneous Localization And Mapping* ed indica l'algoritmo che il robot esegue per mappare l'ambiente e localizzarsi, al fine di navigare. L'obiettivo che abbiamo raggiunto è stato quello di far navigare Pepper in un ambiente popolato, facendo si che non causi problemi a persone e oggetti presenti nello stesso. Per implementare ciò, inizialmente abbiamo considerato un approccio tramite il Sistema ROS, successivamente sostituito dalle API native di Pepper, **NAOqi**. Di seguito analizzeremo entrambi gli approcci e le problematiche riscontrate in ognuno di essi.

1) ROS: La nostra prima scelta per la navigazione nell'ambiente era il Sistema ROS (Robot Operating System), un insieme di framework per lo sviluppo e la programmazione di robot, tra cui Pepper. In particolare, avevamo intenzione di utilizzare il framework **GMapping**, che ci permette di effettuare il mapping dell'ambiente e la navigazione all'interno di esso.

Tuttavia, abbiamo dovuto scartare questa opzione a causa della scarsa compatibilità tra Pepper e ROS. Nonostante ci fossero informazioni dettagliate sul Forum di ROS, abbiamo riscontrato diverse problematiche durante la fase di collegamento con il robot per quanto riguarda la comunicazione tra i sensori di Pepper e ROS. Questa problematica ha reso difficile l'integrazione tra i due sistemi e ci ha spinto a cercare un'alternativa.

2) NAOqi: Come evidenziato nella sezione della metodologia, l'approccio scelto si basa sulle API di *SoftBank Robotics*, che consentono sia l'esplorazione autonoma del luogo da parte di Pepper, sia una successiva fase di navigazione. Tuttavia, queste API risultano datate e basate su una versione obsoleta di Python, il che ha reso difficile principalmente la compatibilità con numerose librerie. Le problematiche riscontrate riguardano le fasi di mapping, localizzazione e navigazione.

- **Mapping**: La luminosità dell'ambiente e la riflettività degli oggetti presenti svolgono un ruolo importante in questa fase. Abbiamo osservato che in condizioni di alta luminosità in un ambiente con superfici altamente riflettenti (come muri bianchi e pavimenti lucidi), Pepper può rilevare falsi positivi, compromettendo la qualità della mappa creata. Un'altra problematica riguarda l'esplorazione non guidata, poiché Pepper esplora autonomamente l'ambiente fino a un raggio massimo predefinito. Nel nostro caso, questo può essere limitante poiché non siamo sicuri che Pepper esplorerà tutte le aree di nostro interesse.
- **Localizzazione**: A causa della luminosità ambientale, Pepper talvolta si localizza in una zona diversa da quella specificata anche se viene fornita un'indicazione sul punto in cui si trova. Ciò influisce negativamente sulla navigazione successiva.
- **Navigazione**: In questa fase, sono presenti due aspetti critici. Il primo dipende strettamente dalla localizzazione, mentre il secondo dipende dalla struttura dell'ambiente e dalla presenza di molte persone. In particolare, la seconda problematica riguarda le capacità di Pepper di evitare ostacoli e navigare in spazi ristretti. Abbiamo osservato che quando Pepper deve passare attraverso spazi stretti, si ferma e interrompe la navigazione.

Fig. 3. Mappa Corridoio Pepper

Nonostante le possibili cause di errore elencate sopra, abbiamo ottenuto ottimi risultati nell'algoritmo SLAM implementato dalla libreria NAOqi, come mostrato nella sezione precedente. Confrontando la mappa creata dall'algoritmo con l'ambiente effettivamente esplorato, notiamo una notevole somiglianza tra le due e una bassa presenza di rumore, nonostante la presenza di una luminosità media e una forte riflessione della luce nell'ambiente.

Per quanto riguarda la localizzazione, abbiamo osservato che nella maggior parte dei casi, quando Pepper viene localizzato dal punto di partenza dell'esplorazione precedente, si posiziona con estrema precisione nell'ambiente e naviga verso l'obiettivo prefissato senza incontrare problemi.

B. Alexa e Face Detection

Le problematiche riscontrate in queste due fasi sono state principalmente legate alla compatibilità delle versioni di Python. Entrambe le API sono state implementate utilizzando versioni compatibili con Python 3.x invece di Python 2.7, che è stato utilizzato per garantire la compatibilità con la libreria NAOqi. Si è dunque scelto di operare con la versione 3.x per ottenere una maggiore compatibilità con altre librerie esterne, come *Google Firestore* utilizzata per l'interazione con un database Firestore. La comunicazione tra Alexa e Pepper è stata osservata come quasi istantanea, senza alcuna problematica evidente. Per quanto riguarda la rilevazione e il riconoscimento dei volti, abbiamo notato che è efficiente e funzionante anche utilizzando immagini provenienti da una

camera diversa da quella di Pepper, l'efficienza è dovuta soprattutto dalla funzionalità di Pepper di muovere il suo volto verso le persone rilevate nella stanza. Tuttavia, unico svantaggio riscontrato è stato il tempo di esecuzione del riconoscimento dei volti, che aumenta proporzionalmente al numero di persone da riconoscere.

V. Conclusioni

L'obiettivo che ci eravamo prefissati all'inizio di questo progetto era quello di migliorare l'assistenza ospedaliera attraverso l'utilizzo di tecnologie come Pepper e Alexa. Grazie all'implementazione di sensori su Pepper e al codice da noi sviluppato, il robot è in grado di navigare autonomamente in un ambiente popolato senza causare alcun problema a persone e oggetti. È importante sottolineare che, sebbene gli esperimenti descritti siano stati condotti in un ambiente diverso da quello ospedaliero, i risultati ottenuti sono promettenti per una futura applicazione nel contesto sanitario.

References

[1] Giovanna Castellano et al. "Detecting Emotions During Cognitive Stimulation Training with the Pepper Robot". In: *Human-Friendly Robotics 2021: HFR: 14th International Workshop on Human-Friendly Robotics*. Springer. 2022, pp. 61–75.

[2] Laurens De Gauquier et al. "Humanoid robot pepper at a Belgian chocolate shop". In: *Companion of the 2018 ACM/IEEE international conference on human-robot interaction*. 2018, pp. 373–373.

[3] Adam Geitgey. "Face recognition documentation". In: *Release* 1.3 (2019), pp. 3–37.

[4] Martin Matulık, Michal Vavrečka, and Lucie Vidovićová. "Edutainment software for the Pepper robot". In: *Proceedings of the 2020 4th International Symposium on Computer Science and Intelligent Control*. 2020, pp. 1–5.

[5] Gavin Suddrey, Adam Jacobson, and Belinda Ward. "Enabling a pepper robot to provide automated and interactive tours of a robotics laboratory". In: *arXiv preprint arXiv:1804.03288* (2018).

[6] Fumihide Tanaka et al. "Pepper learns together with children: Development of an educational application". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2015, pp. 270–275.

[7] Yi-Qing Wang. "An analysis of the Viola-Jones face detection algorithm". In: *Image Processing On Line* 4 (2014), pp. 128–148.