UNIVERSITATEA "POLITEHNICA" DIN BUCUREȘTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

APLICAȚII ALE INTELIGENȚEI ARTIFICIALE FOLOSIND ROBOTUL KUKA

PROIECT DE DIPLOMĂ

Prezentată ca cerință parțială pentru obținerea titlului de Inginer în domeniul Electronică și Telecomunicații programul de studii: Electronică Aplicată

Conducători științifici:
Prof. dr. ing. Corneliu BURILEANU
Conf. dr. ing. Horia CUCU

Student:
Șerban VĂDINEANU

București
2018

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul **EAII**

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ

a studentului **VĂDINEANU A. Şerban , 441B**

1. Titlul temei: Aplicaţii ale inteligenţei artificiale folosind robotul KUKA

2. Descrierea contribuţiei originale a studentului (în afara părţii de documentare):
Lucrarea are ca stadiu iniţial un proiect de recunoaştere de cifre rostite în limba română independent de vorbitor. Scopul acestuia este de a antrena şi testa un model acustic cu ajutorul căruia sistemul să poată recunoaşte cu o acurateţe bună cifrele rostite în limba romană indiferent de persoana care vorbeşte. În continuare, pornind de la proiectul amintit mai sus, se va realiza un asistent robotic capabil sa recunoască comenzi vocale rostite în limba română. În funcţie de aceste comenzi, robotul va executa o serie de mişcări simple, precum deplasarea în diferite direcţii, sau rutine complexe care ar presupune îmbinarea mai multor mişcări simple. Pentru realizarea unei mişcări va fi nevoie ca robotului sa îşi calculeze unghiurile cu care are nevoie sa îşi rotească actuatoarele astfel încât mişcarea rezultata sa fie una precisă şi fluidă.

3. Resurse folosite la dezvoltarea proiectului:
Limbaj de programare: C++ API: pocketsphinx Biblioteci: pocketsphinx, sphinxbase, youbotdriver Robot: KUKA youBot Dicţionar fonetic pentru limba română Model acustic general pentru limba română

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
PC, POO, PDS, DEPI

5. Proprietatea intelectuală asupra proiectului aparţine: U.P.B.

6. Data înregistrării temei: 2017-12-05 14:52:45

Conducător(i) lucrare,
Prof. dr. ing. Corneliu BURILEANU

semnătura: .....

Ş.l. dr. ing. Horia CUCU

semnătura: .....

Student,

semnătura: .....

Director departament,
Prof. dr. ing Sever PAŞCA

semnătura: .....

Decan,
Prof. dr. ing. Cristian NEGRESCU

semnătura: .....

Cod Validare: **afb473fbda**

DECLARAȚIE DE ONESTITATE ACADEMICĂ

Prin prezenta declar că lucrarea cu titlul "Aplicații ale inteligenței artificiale folosind robotul KUKA", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității "Politehnica" din București ca cerință parțială pentru obținerea titlului de Inginer în domeniul Inginerie Electronică și Telecomunicații, programul de studii Electronică Aplicată, este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau din străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurătorilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, *30.05.2018*

Șerban VĂDINEANU

CUPRINS

LISTA DE FIGURI

LISTA DE TABELE

LISTA DE ACRONIME

API = Application Programming Interface

ChER = Character Error Rate

DNN = Deep Neural Network

DTW = Dynamic Time Warping

FSG = Finite State Grammar

GMM = Gaussian Mixture Model

HMM = Hidden Markov Models

IoT = Internet of Things

JSGF = Java Speech Grammar Format

KWS = Keyword Spotting

LAN = Local Area Network

MFCC = Mel-Frequency Cepstrum Coefficients

MLP = Multilayer Perceptron

PC = Personal Computer

PLP = Perceptual Linear Prediction

RAM = Random Access Memory

RAV = Recunoașterea Automată a Vorbirii

RNN = Recurrent Neural Network

ROS = Robot Operating System

SER = Sentence Error Rate

SSD = Solid State Drive

SSH = Secure Shell

USB = Universal Serial Bus

WER = Word Error Rate

CAPITOLUL 1

INTRODUCERE

1.1 MOTIVAȚIE

KUKA este o companie activă la nivel internațional care oferă soluții pentru automatizarea proceselor de asamblare și producție pentru diferite sectoare industriale, precum industria auto [KUKA, 2018]. Ideea care stă la baza roboților KUKA este aceea de a realiza sisteme inteligente și adaptive care să devină autonome față de operatorul uman. Pentru atingerea acestui scop, compania încurajează dezvoltarea de aplicații pe astfel de roboți pentru a familiariza viitorii specialiști cu domeniul roboticii și cel al automatizărilor.

KUKA youBot este o platformă robotică creată în vederea oferirii unui mediu de dezvoltare care să ilustreze posibilitățile de utilizare și programare a unei platforme KUKA [Huggenberger, 2011]. Există o diversitate abundentă de aplicații care se pot realiza, pornind de la controlul printr-un joystick și ajungând la colaborarea mai multor roboți pentru rezolvarea de sarcini complexe, cum ar fi îmbinarea unor piese de mobilier.

O aplicație care să valorifice performanțele platformei youBot este realizarea unui asistent robotic capabil să proceseze comenzi vocale și să ofere un răspuns atât prin voce cât și prin mișcările actuatoarelor sale. De asemenea, există și un nivel de securitate realizat prin recunoașterea facială, întregul proces realizându-se local.

Această lucrare are scopul de a demonstra posibilitatea de a implementa pe o mașină robotică diferite rutine independente care să poată fi accesate folosind recunoașterea vocală. Astfel, atât

interacțiunea cât și alegerea programului de lucru vor fi realizate prin voce. Performanțele robotului pot fi îmbunătățite prin creșterea complexității rutinelor pe care acesta le deservește. Rezultatul fiind un sistem autonom care să se poată ocupa de sarcini simple și repetitive, reducând semnificativ timpul dedicat rezolvării acestora.

1.2 OBIECTIVE

Pentru proiectul acesta au fost propuse următoarele obiective:

1. Dezvoltarea unui sistem de recunoaștere de cifre rostite în limba română, cu o eroare sub 10 %.
2. Crearea, pe robotul KUKA, a unei aplicații de recunoaștere de comenzi formate din succesiuni de cifre.
3. Adăugarea detecției de cuvinte cheie și folosirea pentru recunoașterea automată a vorbirii a unui model acustic general.
4. Integrarea aplicației de recunoaștere de fețe cu sistemul de recunoaștere automată a vorbirii folosit la preluarea de comenzi vocale.
5. Actualizarea aplicației pentru a integra alte funcționalități.
6. Dezvoltarea unui sistem de detecție de forme simple utilizând camera instalată pe robot.
7. Dezvoltarea unei rutine de prindere și aducere a corpurilor geometrice.

1.3 STRUCTURA LUCRĂRII

În *Capitolul 2* se va face o introducere în domeniul recunoașterii automate a vorbirii, prezentând detalii generale legate de modul de creare și evaluare a sistemelor de transcriere a vorbirii. Capitolul prezintă concepte teoretice ce țin de arhitectură, etape de dezvoltare, resurse necesare, precum și expune unele noțiuni de lingvistică.

Capitolul 3 este dedicat detalierii caracteristicilor robotului pe care a fost implementată aplicația finală. Este expusă componența hardware atât din punctul de vedere al materialelor implicate în fabricarea platformei, cât și din cel al caracteristicilor fizice sau electrice, precum tensiuni nominale sau sarcina utilă. De asemenea, sunt aduse și informații și despre instrumentele software pe baza cărora pot fi dezvoltate programele pe robot.

Capitolul 4 urmărește etapele care au intrat în componența antrenării și optimizării unui sistem de recunoaștere automată a vorbirii pentru cifrele limbii române. În plus, capitolul ilustrează modul cum o anumită configurație a sistemului influențează performanțele acestuia.

Descrierea modalităților prin care robotul KUKA poate fi controlat este prezentată în *Capitolul 5*. Tot aici sunt arătate diferitele stadii în care s-a aflat programul de transmitere de comenzi și, de asemenea, au fost semnalate dificultățile care au apărut odată cu trecerea de la o etapă la alta.

Capitolul 6 descrie implementarea aplicației finale, având la bază un program de recunoaștere facială. Capitolul realizează o paralelă între programul inițial și cel rezultat, evidențiind facilitățile adăugate și anumite porțiuni eliminate.

CAPITOLUL 2

RECUNOAȘTEREA AUTOMATĂ A VORBIRII

2.1 ASPECTE GENERALE

Recunoașterea vorbirii este un domeniu interdisciplinar care are ca scop dezvoltarea de tehnologii și metodologii care permit transformarea de către un computer a semnalelor acustice de vorbire în text.

Unul dintre primele proiecte de RAV realizate vreodată a fost un sistem de recunoaștere de cifre rostite de un anumit utilizator [Juang, 2004]. Sistemul se baza pe analiza spectrală a puterii semnalelor vocale. Pe măsură ce tehnologia a evoluat, sistemele au început să lucreze cu un vocabular din ce în ce mai dezvoltat, spre exemplu, la mijlocul anilor 80, IBM a dezvoltat un program care lucra cu un dicționar de 20000 de cuvinte [IBM, 2001].

În prezent, RAV este intens utilizată de giganți precum Apple sau Google în aplicații de asistenți virtuali, securitate sau IoT, bucurându-se de o acuratețe foarte ridicată.

Dacă sursa audio conține semnale de la mai multe persoane atunci RAV poate fi completat de informații legate de persoanele care au vorbit, împărțind astfel sarcinile în două procese:

- Diarizarea: etapă care identifică vorbitorul și precizează momentul de timp la care acesta începe vorbirea
- Transcrierea: etapă care se ocupă cu determinarea cuvintelor rostite

Recunoașterea reprezintă un aspect foarte important care decide cât de dificil va fi procesul de transcriere. Aici sunt incluse caracteristicile limbii din care se dorește să se facă transcrierea, precum incertitudinea lingvistică și diversitatea de pronunțare a cuvintelor. Astfel, o problemă care apare este legată de disponibilitatea resurselor acustice (baze de date de voce) și lingvistice (corpusuri de text). De asemenea, complexitatea morfologică a anumitor limbi conduce la crearea unui vocabular cu un numar foarte mare de cuvinte, ceea ce, la randul lui, poate conduce la o îngreunare a întregului sistem [Cucu, 2013].

În Tabelul 2.1 sunt prezentate rezultatele experimentale pentru diferite sarcini de recunoaștere de vorbire. Se poate observa că rata de eroare devine din ce în ce mai mare pe măsură ce dimensiunea vocabularului crește și, de asemenea, aceasta depinde și de tipul de vorbire pentru care a fost concepută aplicația.

Sarcina de RAV	Tipul vorbirii	Dimensiunea vocabularului	WER [%]
Cifre	citire	10 cuvinte	0.3
Informații despre zboruri	spontană	2500 cuvinte	2.5
Știri de afaceri din America de Nord	citire	64000 cuvinte	6.6
Știri	citire	210000 cuvinte	13-17
Centrală telefonică	conversațională la telefon	45000 cuvinte	25-29

Tabelul 2.1 WER pentru diferite aplicații de RAV [Acero, 2015]

Metrica de performanță a sistemului de RAV este rata de cuvinte eronate (word error rate). Aceasta reprezintă numărul de cuvinte eronat recunoscute raportat la numărul cuvintelor de referință.

Se poate observa că există o tendință crescătoare a WER odată cu creșterea dimensiunii vocabularului. De asemenea, este de remarcat faptul că, pentru vocabulare de dimensiuni asemănătoare, rata de cuvinte eronate prezintă o variație semnificativă. Acest lucru se datorează diferenței de dificultate a recunoașterii pentru un sistem de RAV utilizat pentru dictare (Știri de afaceri) față de unul folosit pentru vorbire telefonică în care apare zgomot (Centrală telefonică).

2.2 STRUCTURA UNUI SISTEM DE RAV

De-a lungul timpului au mai multe tehnologii au fost implicate în sistemele de RAV:

- **Dynamic time warping (DTW)** este una dintre primele abordări care au pus bazele recunoașterii de vorbire. Principiul algoritmului este de a măsura gradul de similitudine între două secvențe care pot varia în timp sau în viteză. În prezent a fost înlocuită de HMM.
- **Rețele neuronale adânci (deep neural networks – DNN) în combinație cu rețele neuronale recurente (recurrent neural networks – RNN)**. RNN-urile sunt rețele neuronale cu memorie, în care luarea unei decizii se bazează pe rezultatele generate anterior, iar DNN-urile sunt rețele neuronale cu multe straturi ascunse. Combinându-le rezultă sisteme de recunoaștere de voce de tip „end-to-end” care nu necesită extragere de parametri din semnalul vocal, nu necesită transcrieri sau dicționar [Fonollosa, 2017]. Ieșirea acestor sisteme este reprezentată de o succesiune de caractere, inclusiv caracterul spațiu. Au fost remarcate rezultate bune în condiții de mediu zgomotos. Arhitectura unei rețele RNN-DNN este prezentată în Figura 2.1.

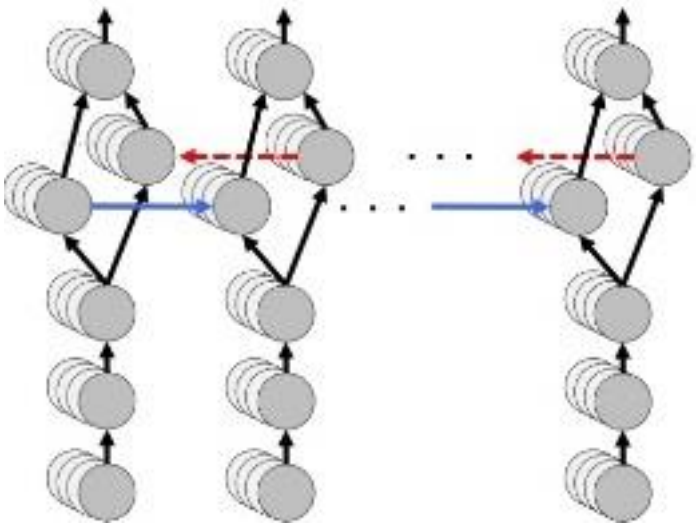

Figura 2.1 Arhitectura RNN-DNN [Fonollosa, 2017]

- **HMM-urile** reprezintă cea mai folosita modalitate în aplicațiile de RAV. Aceastea prezintă avantajele de a putea fi antrenate în mod automat și de a nu produce un efort computațional foarte ridicat. De asemenea, datorită faptului că semnalul vocal poate fi considerat un semnal staționar pe un interval foarte scurt de timp, aceste modele probabilistice devin o metodă eficientă de a îl prelucra.

După cum se poate observa din Figura 2.2, se pot trage următoarele concluzii referitoare la funcționarea unui sistem de RAV:

1) recunoașterea semnalului vocal nu se face direct, ci în urma unei extrageri de parametri
2) recunoașterea are la bază o serie de modele care au fost antrenate la un moment anterior

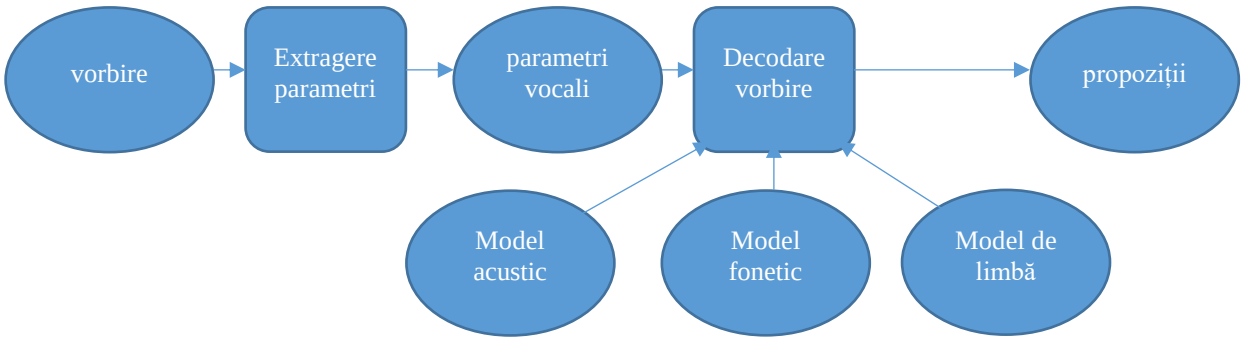

Figura 2.2 Arhitectura generală a unui sistem de RAV [Cucu, 2013]

Modelul acustic determină, pe baza unei anumite succesiuni de cuvinte, probabilitatea de a apariție a unui mesaj vorbit. Având în vedere faptul că pot exista sarcini de RAV care sa nu aibă modele antrenate pentru cuvinte, precum și faptul că volumul de cuvinte din vocabularul unei limbi este foarte mare, modelul acustic nu folosește cuvintele pe post de unități acustice de bază, ci subdiviziuni acustice ale acestora, numite foneme sau chiar subdiviziuni de foneme, denumite senone [Hwang, 1992]. În consecință, modelul acustic conține modele pentru foneme. Mai departe, pe durata decodării, se formează modelele pentru cuvinte prin combinarea ierarhică a modelelor pentru senone și mai departe, în mod similar, se obțin modelele pentru succesiuni de cuvinte. Pe baza acestora se estimează probabilitatea ca mesajul rostit să fie alcătuit dintr-o anumită succesiune de cuvinte.

Modelul de limbă se folosește pentru a realiza o estimare a probabilităților tuturor secvențelor de cuvinte care se pot forma în limba respectivă. Adică, pentru o succesiune de cuvinte $W = w_1, w_2,$

..., w_n se determină probabilitatea de a fi o propoziție validă. Această estimare facilitează foarte mult decizia pe care modelul acustic trebuie sa o ia.

Modelul fonetic mediază conexiunea între modelul acustic și modelul de limbă și, de cele mai multe ori, este reprezentat printr-un dicționar de pronunție prin care se face asocierea fiecărui cuvânt din vocabular cu una sau mai multe secvențe de foneme corespunzătoare. Astfel se face o reprezentare a modului de pronunțare a acelui cuvânt.

Având în vedere modelele prezentate anterior, se poate realiza o reprezentare formală a probabilității secvenței de cuvinte:

$$W^* = \underset{w}{\operatorname{argmax}}\, p(W|X) = \underset{w}{\operatorname{argmax}} \frac{p(X|W)p(W)}{p(X)} = \underset{w}{\operatorname{argmax}}\, p(X|W)p(W)$$

Funcția argmax alege acel argument pentru care probabilitatea secvenței de cuvinte este maximă.

Ecuația prezentată se bazează pe regula lui Bayes, considerând probabilitatea mesajului vorbit p(X) ca fiind independentă de secvența de cuvinte W. Astfel, sarcina găsirii succesiunii de cuvinte pe baza mesajului vocal se descompune în două sub-sarcini de o complexitate mai mică, și anume:

1) Se estimează probabilitatea de apariție a unei succesiuni de cuvinte p(W). Această estimare se realizează cu ajutorul **modelului de limbă**.
2) Se estimează probabilitatea mesajului vocal pe baza secvenței de cuvinte p(X|W). Sarcina aceasta este îndeplinită de **modelul acustic**.

2.3 Resursele necesare unui sistem de RAV

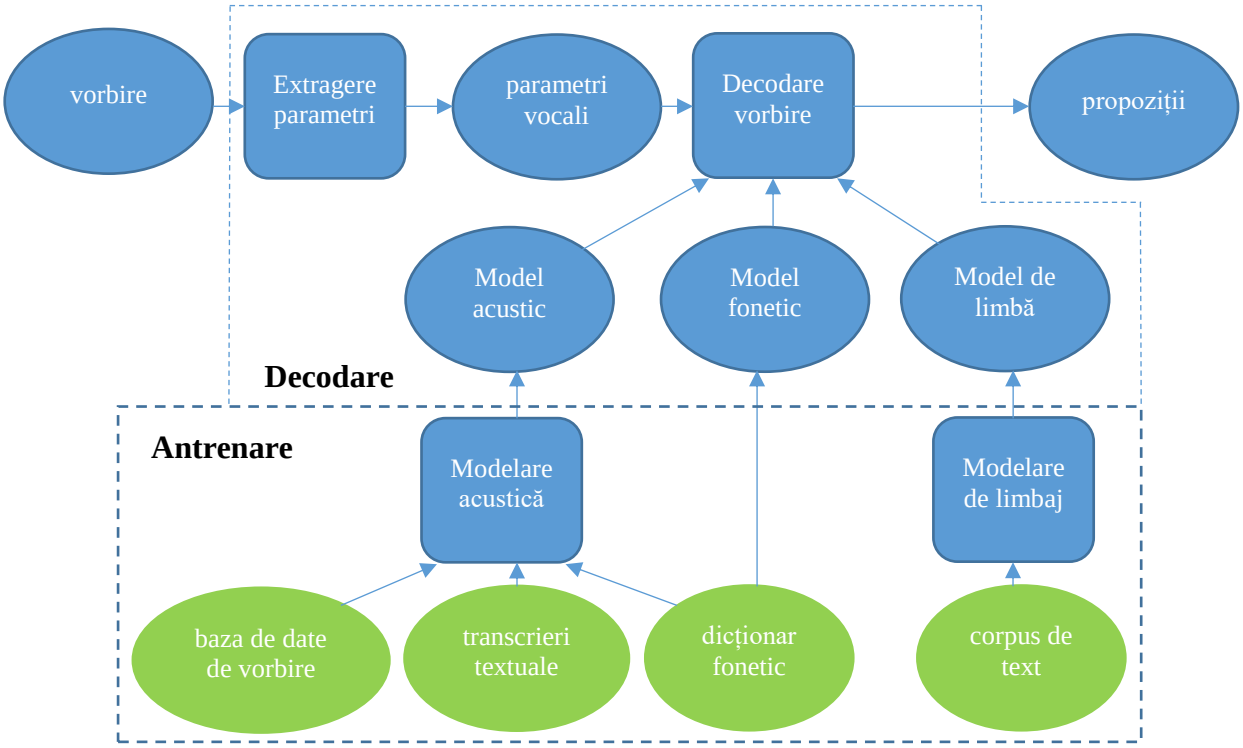

Figura 2.3 Resursele necesare dezvoltării unui sistem de recunoaștere a vorbirii [Cucu, 2013]

După cum se poate observa din Figura 2.3 modelele pe care se bazează un sistem de RAV se construiesc în etapa de antrenare pe baza unor anumite resurse.

Baza de date de vorbire este alcătuită dintr-un set de clipuri audio, iar împreună cu transcrierile textuale ale acestor clipuri și cu dicționarul fonetic crează modelul acustic. De asemenea, același dicționar reprezintă resursa care formează modelul fonetic.

Corpusul de text este elementul fundamental pentru realizarea modelului de limbă. Pe baza acestuia se determină probabilitățile de apariție ale diferitelor cuvinte sau secvențe de cuvinte din limba respectivă.

Dicționarul fonetic are o mare importanță în sistemele de RAV, fiind folosit atât în etapa de modelare acustică, precum și în generarea modelului fonetic. Rolul dicționarului fonetic este de a specifica modul de pronunțare a cuvintelor dintr-o limbă. Așadar, acesta va conține toate cuvintele de care este nevoie pentru aplicația respectivă de recunoaștere vocală, împreună cu transcirerile lor fonetice. Spre exemplu, în Tabelul 2.2 este prezentată lista tuturor fonemelor din limba română.

Fonemul			**Exemple de cuvinte**	
Tip	**Simbolul IPA**	**Simbol intern**	**Forma scrisă**	**Forma fonetică**
vocale	a	a	s**a**t	s a t
	e	e	mar**e**	m a r e
	i	i	l**i**ft	l i f t
	o	o	l**o**c	l o c
	u	u	ș**u**t	s1 u t
	ə	a1	gur**ă**	g u r a1
	ɨ	i2	**î**ntre	i2 n t r e
vocale împrumutate	y	y	ecr**u**	e c r y
	ø	o2	bl**eu**	b l o2
semivocale	e	e1	d**e**al	d e1 a l
	j	i3	f**i**ară	f i3 a r a1
	o	o1	**o**ase	o1 a s e
	w	w	sa**u**	s a w
consoane	c	k2	**ch**em	k2 e m
	b	b	**b**ar	b a r
	p	p	**p**ar	p a r
	k	k	a**c**um	a k u m
	ʧ	k1	**c**enușă	k1 e n u s1 a1
	g	g	**g**alben	g a l b e n
	ʤ	g1	**g**irafă	g1 i r a f a1
	ɟ	g2	un**gh**i	u n g2
	d	d	**d**ar	d a r
	t	t	**t**ot	t o t
	f	f	**f**ața	f a t1 a
	v	v	**v**apor	v a p o r
	h	h	**h**arta	h a r t a
	ʒ	j	a**j**utor	a j u t o r
	ʃ	s1	co**ș**	k o s1
	l	l	**l**ac	l a c
	m	m	**m**ăr	m a1 r
	n	n	**n**as	n a s
	s	s	**s**are	s a r e
	z	z	**z**ar	z a r
	r	r	**r**isc	r i s k
	ʦ	t1	**ț**ăran	t1 a1 r a n

consoană palatalizată	j	i1	tar**i**	t a r i1

Tabelul 2.2 Fonemele limbii române [Cucu, 2013]

Pentru realizarea dicționarului se pot exista trei abordări: manuală, semi-automată și automată. Cea manuală se pretează atunci când dicționarul fonetic are în componența sa un număr redus de cuvinte, în timp ce pentru dicționare cu un volum mare de cuvinte se folosește metoda automată. Dezavantajul celei din urmă este complexitatea implementării.

Baza de date de vorbire are un dublu rol. Se folosește pentru antrenarea modelului acustic și mai apoi se utilizează pentru testarea sistemului. Aceasta are în componență trei elemente:

1) clipuri audio cu semnal de voce
2) fișiere text în care se află transcrierea mesajelor ce au fost rostite în clipurile audio
3) informații suplimentare legate de identitatea vorbitorului, stilul vorbirii, etc.

Eficiența sistemului este strâns legată de caracteristicile bazei de date de vorbire. Parametrii care influențează cel mai mult calitatea acesteia sunt:

- dimensiunea bazei de date (ore vorbite, diversitatea vorbitorilor)
- stilul vorbirii
- calitatea înregistrărilor (zgomot de fundal)

Având în vedere parametrii menționați, CMU Sphinx recomandă niște valori pentru dimensionarea bazelor de date, după cum este prezentat în Tabelul 2.3.

Sarcina RAV	**Sistem dependent de vorbitor**	**Sistem independent de vorbitor**
comandă + control	1 oră de înregistrări, 1 vorbitor	5 ore de înregistrări, 200 de vorbitori
dictare	10 ore de înregistrări, 1 vorbitor	50 de ore de înregistrări, 200 de vorbitori

Tabelul 2.3 Dimensionarea bazei de date recomandată de CMU Sphinx [CMU Sphinx, 2018]

Corpusul de text se folosește doar în cazul sistemelor cu un vocabular mare care necesită un model de limbă statistic. Dimeniunea corpusului trebuie sa fie de ordinul milioanelor de cuvinte, iar sursa pentru un asemenea text nu poate fi decât Internetul. Dificultatea este dată de automatizarea procesului de extragere a textului, precum și de prelucrarea acestuia pentru a îl uniformiza.

2.4 MODELAREA ACUSTICĂ

După cum este ilustrat în Figura 2.2, modelarea acustică a semnalului vocal nu se face direct pe acesta ci în urma unei extrageri de parametri. Vorbirea este un semnal nestaționar, dar pe intervale scurte el prezintă caracteristici cvasi-staționare, așadar pe aceste intervale se poate realiza o analiză spectrală. În urma împărțirii semnalului inițial în ferestre, pe fiecare dintre aceste ferestre se realizează o extragere de parametri. Cei mai utliziați parametri pentru modelarea vorbirii sunt parametrii Mel-cepstrali (Mel-Frequency Cepstrum Coefficients - MFCC) și parametrii perceptuali obținuți prin predicție liniară (Perceptual Linear Prediction - PLP). Reprezentarea cepstrală este însă mai avantajoasă deoarece coeficienții rezultați sunt decorelați, spre deosebire de cea spectrală în care coeficienții vecini prezintă un grad ridicat de corelație.

Extragerea coeficienților MFCC parcurge următoarele etape [Sahidullah, 2012]:

1) Se realizează transformata Fourier discretă pe o fereastră din semnal.

2) Puterile obținute din spectru se mapează pe scala mel, folosind ferestre triunghiulare suprapuse.
3) Se logaritmează puterile de la fiecare frecvență mel.
4) Se realizează transformata cosinus discretă pe lista de puteri logaritmate anterior, tratată ca fiind un semnal.
5) Coeficienții MFC sunt reprezentați de amplitudinile din spectrul rezultat.

Diferența dintre un cepstru și cepstrul pe scala mel este aceea că în cel de-al doilea caz benzile de frecvență sunt separate în mod egal pe scala mel, scală ce aproximează mult mai bine răspunsul sistemului auditiv uman față de separarea liniară a benzilor ce se face în cazul cepstrului obișnuit.

Odată extrași parametrii din semnalul de voce, aceștia sunt preluați de către modelul acustic. Acest model are la bază modelele Markov ascunse care sunt automate cu stări finite alcătuite din mai multe stări și tranzițiile corespunzătoare acestora. Modelele se numesc ascunse deoarece stările din componența lor nu sunt direct disponibile observatorului. Fiecare stare are atașată o funcție de densitate de probabilitate care generează o secvență de vectori de parametri acustici la care observatorul are acces.

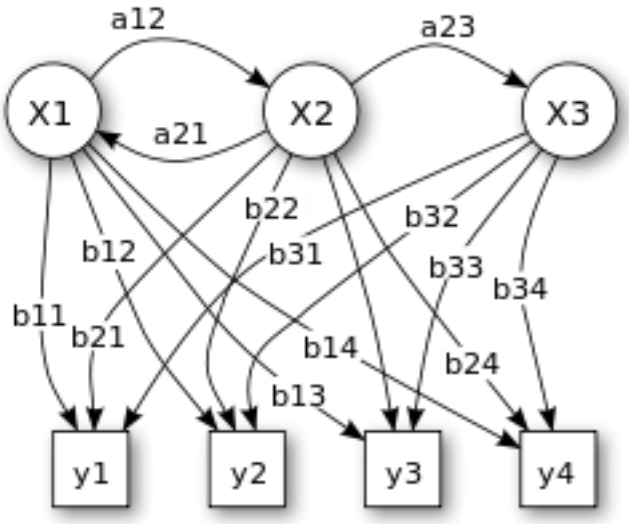

Figura 2.4 Parametrii probabilistici ai unui HMM

În Figura 2.4 este reprezentat un HMM ca un automat cu stări finite unde:

- Xi reprezintă stările
- aij = p(Xj|Xi) reprezintă probabilitatea de tranziție de la starea Xi la starea Xj
- yi reprezintă observațiile posibile
- bij reprezintă probabilitatea ca observația yj să fi fost generată de starea Xi

Există mai multe variante de modelare a probabilităților de la ieșire (bij). Dintre acestea cele mai uzuale sunt:

1. GMM (Gaussian Mixture Model) care au la bază densități normale de probabilitate. O stare a modelului Markov ascuns (o senone) poate fi modelată prin una sau mai multe astfel de densități de probabilitate. Astfel, la ieșirea HMM-ului va rezulta probabilitatea ca, pe baza coeficienților de la intrare, să fie recunoscută o anumită senone.
2. Rețelele neuronale de tip MLP (multilayer perceptron). În acest caz recunoașterea unei senone se face pe baza ponderilor rezultate în urma antrenării rețelei. Raportând la Figura 2.3, bij ar reprezenta ponderile rețelei.

2.5 MODELAREA LIMBAJULUI NATURAL

Modelul de limbă estimează probabilitatea ca o succesiune de cuvinte să formeze o secvență validă pentru scopul sistemului.

Există două tipuri de modelare a limbajului natural în funcție de specificul aplicației dezvoltate:

1) Modelare bazată pe modele de limbă statistice.
2) Modelare bazată pe gramatici cu stări finite (finite state grammr – FSG)

Modelele de limbă statistice se folosesc în aplicațiile de recunoaștere de vorbire cu vocabular extins și provin din prelucrarea unor corpusuri mari de text. Problema determinării probabilității unei secvențe de cuvinte se descompune în estimarea probabilităților tuturor cuvintelor din secvența respectiva, fiind condiționate de cuvintele anterioare lor.

Fie secvența $W = w_1, w_2, ..., w_n$, probabilitatea ca această secvență să formeze o propoziție validă poate fi exprimată în felul următor:

$$p(W) = p(w_1, w_2, ..., w_n) = p(w_1)p(w_2|w_1)...p(w_n|w_1, w_2, ..., w_{n-1})$$

Acest aspect este problematic deoarece dacă s-ar ține cont de un numar nedefinit de cuvinte anterioare atunci efortul de calcul ar fi unul mult prea ridicat. Din acest motiv, s-a definit modelul de limbă convențional de tip n-gram. Cel mai folosit este cel de tip trigram, în care se determină al treilea cuvânt pe baza precedentelor două.

Luând ca exemplu cazul bigram, pentru calculul probabilităților $p(w_j|w_i)$ se numără de câte ori cuvântul w_j este precedat de cuvântul w_i în comparație cu toate celelalte cuvinte.

$$p(w_j|w_i) = \frac{\text{count}(w_i, w_j)}{\sum_w \text{count}(w, w_j)}$$

Corectitudinea estimării depinde puternic de corpusul de text folosit. Pentru a obține probabilități cât mai apropiate de realitate, este nevoie ca și dimensiunea corpusului de text să fie crescută semnificativ.

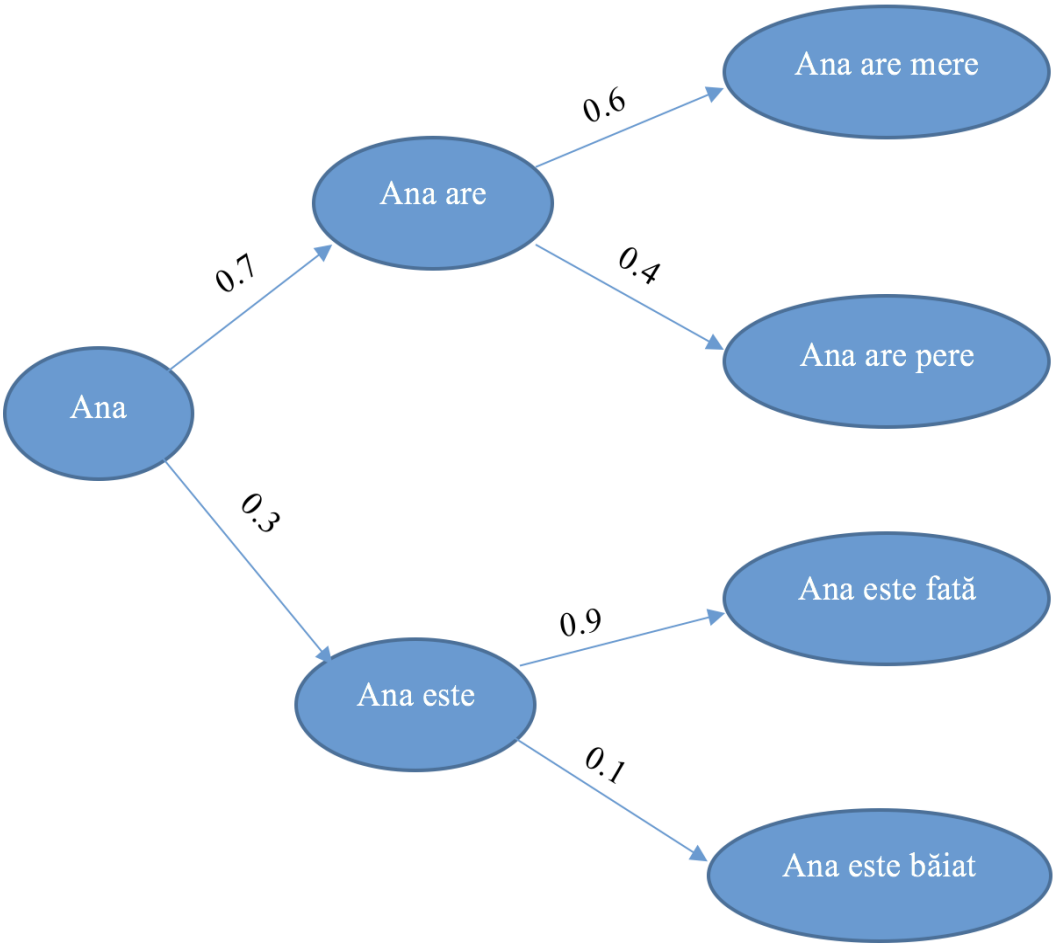

Figura 2.5 Exemplu de graf probabilistic pentru construirea de propoziții

Figura 2.5 ilustrează modul în care se pot deduce anumite propoziții într-o limbă pe baza probabilităților de completare a unor cuvinte sau secvențe de cuvinte. Astfel, odată ce a fost recunoscut cuvântul „Ana”, este mult mai probabil ca acesta să fie urmat de cuvântul „are” (probabilitate de 70%), decât să fie urmat de cuvântul „este” (probabilitate de 30%). Continuând pe ramura cu probabilitate mai mare, se poate construi mai departe propoziția „Ana are mere” sau „Ana are pere”, unde probabilitatea de apariție mai mare o are prima secvență.

În continuare este prezentat un exemplu de model de limbă probabilistic de tip trigram care înglobează legăturile menționate în exemplul anterior:

```
\data\
ngram 1=1
ngram 2=2
ngram 3=4

\1-grams:
0  Ana

\2-grams:
-0.155  Ana  are
-0.523  Ana  este

\3-grams:
-0.377  Ana are mere
-0.553  Ana are pere
-0.569  Ana este fată
-1.523  Ana este băiat
\end\
```

În acest exemplu se specifică inițial câte secvențe din fiecare tip de n-gram (unigram, birgram și trigram) există. Apoi este declarat fiecare n-gram în parte. Fiecare declarație începe cu probabilitatea de apariție a succesiunii de cuvinte respective, reprezentată ca $\log_{10}$, și este urmată de o secvență de n cuvinte care descriu n-gramul.

FSG-urile își dovedesc utilitatea în cazul în care scopul aplicației de RAV nu include și folosirea unui vocabular foarte cuprinzător, iar succesiunea cuvintelor este una bine definită. În aceste condiții modelul n-gram nu își justifică cerințele pentru antrenare.

O gramatică cu stări finite este realizată sub forma unui graf în care nodurile sunt cuvintele vocabularului, iar arcele grafului reprezintă tranzițiile între cuvinte. În acest mod se specifică toate secvențele de cuvinte care pot fi formate. Figura 2.6 aduce un exemplu al unei gramatici cu stări finite.

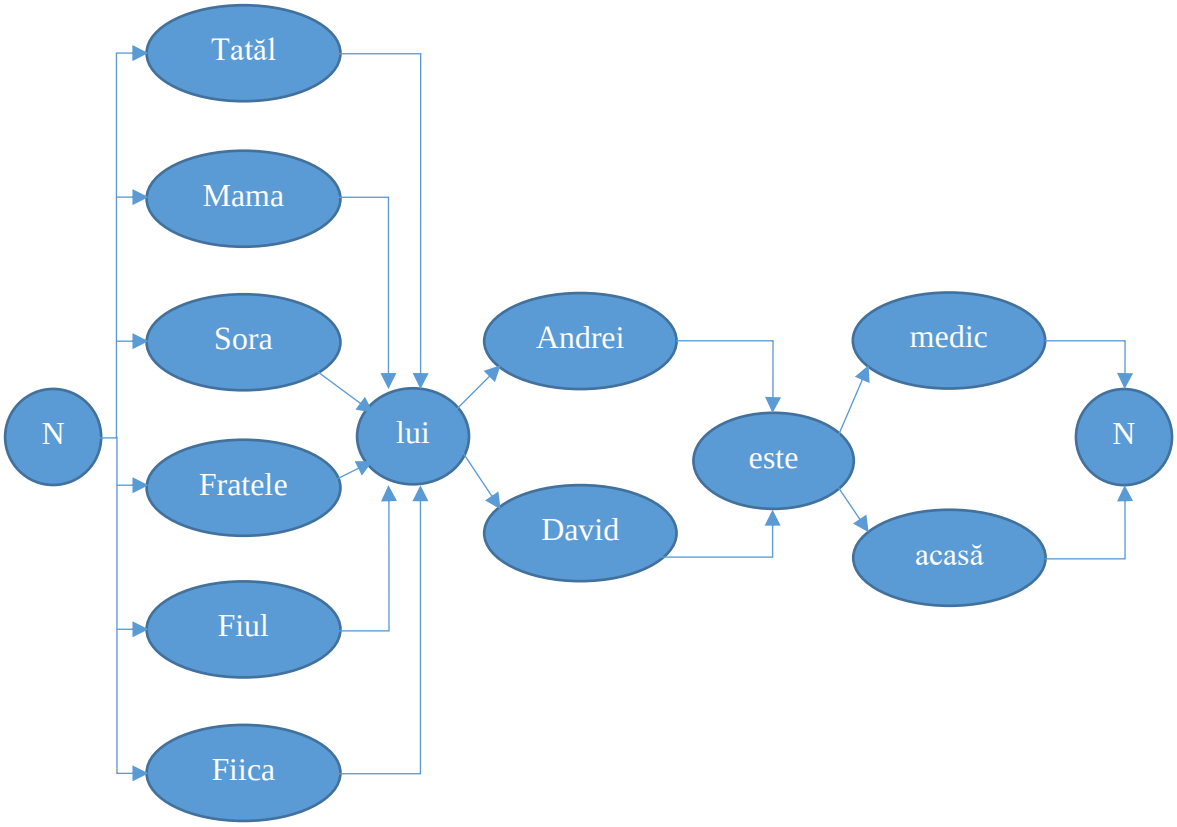

Figura 2.6 Exemplu de gramatică cu stări finite

Nodurile reprezentate cu N în Figura 2.6 sunt noduri de infrastructură. Ele nu conțin cuvinte din vocabular ci reprezintă intrarea și, respectiv ieșirea FSG-ului. Se poate observa că posibilitatea formării unei succesiuni de cuvinte este limitată atât ca număr de cuvinte cât și ca diversitatea acestora. De asemenea, o propoziție se poate forma cu cuvintele dispuse într-o anumită ordine.

2.6 Evaluarea performanțelor sistemelor de RAV

Odată antrenat un sistem, acesta va fi capabil să răspundă foarte bine pentru clipurile audio cu care a fost antrenat, dar, pentru a își dovedi utilitatea practică, acesta trebuie să răspundă corect și la stimuli noi. Din acest punct de vedere evaluarea devine o etapă foarte importantă atât în construirea cât și în îmbunătățirea sistemului.

Pentru a testa modul în care sistemul raspunde la clipuri audio noi este nevoie să se creeze o bază de date de testare cu înregistrări etichetate. Această bază de date trebuie să atingă anumite aspecte de interes care să simuleze cât mai bine condițiile de lucru pentru care sistemul a fost proiectat:

- domeniul vorbirii: vorbire liberă sau vorbire cu o tematică fixată
- caracteristicile vorbitorului: vorbitori tineri/în vârstă, mai mulți vorbitori (sistem independent de vorbitor) sau un singur vorbitor (sistem dependent de vorbitor)
- mediul: zgomotos, lipsit de zgomot
- tipul vorbirii: vorbire spontană, dictare, cuvinte izolate

O evaluare completă a aplicației de RAV este ilustrată în Figura 2.7. Aceasta presupune compararea automată a transcrierilor fișierelor audio (transcrieri de referință) din baza de date de testare cu transcrierile rezultate în urma etapei de decodare (transcrieri ipotetice) a acelor fișiere.

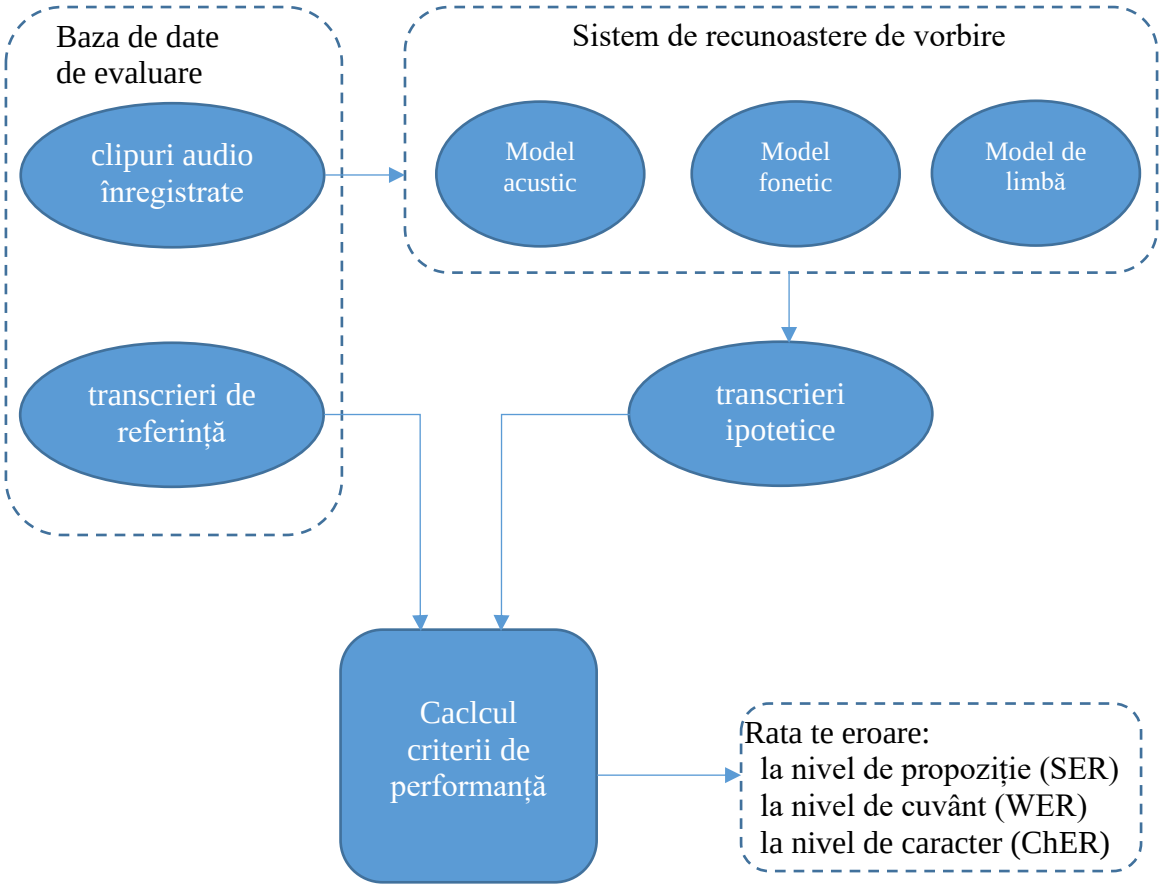

Figura 2.7 Evaluarea completă a sistemului de recunoaștere [Cucu, 2013]

Rata de eroare la nivel de propoziție (sentence error rate – SER) reprezintă raportul dintre numărul de propoziții ce au cel puțin o greșeală și numărul total de propoziții din transcrierea din baza de date de testare.

$$\text{SER}[\%] = \frac{\text{Numărul de propoziții eronate}}{\text{Numărul de propoziții din transcrierea de referință}} \times 100$$

Adesea această variantă de a măsura performanța nu este atât de utilă deoarece utilizatorul aplicației poate înțelege mesajul în ciuda faptului că acesta conține o proporție mică de greșeli. Se pretează totuși în cazurile în care este nevoie de o mare strictețe în privința corectitudinii mesajului, precum aplicațiile bancare.

Cel mai frecvent utilizat mijloc de evaluare este calculul ratei de eroare la nivel de cuvânt (word error rate – WER). Determinarea WER se realizează în doi pași [Cucu, 2013]:

1) fraza de la ieșirea sistemului și fraza de referință sunt aliniate prin intermediul unui algoritm.
2) se contorizează erorile la nivelul cuvintelor. Acestea pot fi de trei tipuri: cuvinte inserate, cuvinte substituite și cuvinte șterse.

$$\text{WER}[\%] = \frac{\text{Numărul de substituţii} + \text{Numărul de ştergeri} + \text{ Numărul de inserţii}}{\text{Numărul de cuvinte din transcrierea de referinţă}} \times 100$$

Folosirea WER este mult mai potrivită în evaluarea unui sistem de RAV, dar prezintă dezavantajul că nu discerne între tipurile de erori. Acest aspect reprezintă un impediment în momentul în care se dorește realizarea unei corecții de erori deoarece unele erori de substituție pot fi corectate mai ușor dacă procentul de caractere greșite este rezonabil.

Având în vedere această posibilitate, se poate considera că, în anumite situații, este indicat să se folosească rata de eroare la nivel de caracter (character error rate – ChER). Formula de calcul pentru ChER este asemănătoare cu cea pentru WER, diferența fiind că, în acest caz, comparația se realizează la nivel de caracter, nu la nivel de cuvânt:

$$\text{ChER}[\%] = \frac{\text{Numărul de substituţii} + \text{Numărul de ştergeri} + \text{ Numărul de inserţii}}{\text{Numărul de caractere din transcrierea de referinţă}} \times 100$$

CAPITOLUL 3

PLATFORMA ROBOTICĂ KUKA YOUBOT

3.1 DESCRIERE GENERALĂ

KUKA youBot este un robot care a fost creat ca o platformă cu sursă deschisă pentru cercetare științifică și testare [KUKA, 2018]. Robotul este format dintr-o platformă mobilă și un braț robotic, ambele componente fiind puse în mișcare de motoare de curent continuu fără perii.

Compania KUKA a dezvoltat acest robot cu scopul de a oferi studenților sau pasionaților de robotică o modalitate de a dezvolta și testa programe care să poată fi implementate pe un robot [Schütz, 2013]. Manipulatorul mobil, care este controlat cu software cu sursă deschisă, are o bază omnidirecțională și o mecanică robotică ce dispune de 5 grade de libertate.

3.2 COMPONENTE HARDWARE

Robotul KUKA youBot constă dintr-un șasiu ce se poate mișca în orice direcție și un braț robotic montat pe șasiu. În șasiu au mai fost integrate și alte componente precum un PC industrial și o baterie. PC-ul comunică în timp real (un ciclu durează 1 ms) cu cele 9 actuatoare prin intermediul EtherCAT. Fiecare actuator poate fi comandat prin control al curentului, al vitezei sau al poziției. În plus, platforma mobilă poate fi folosită independent față de braț [Schütz, 2013].

3.2.1 PLATFORMA MOBILĂ

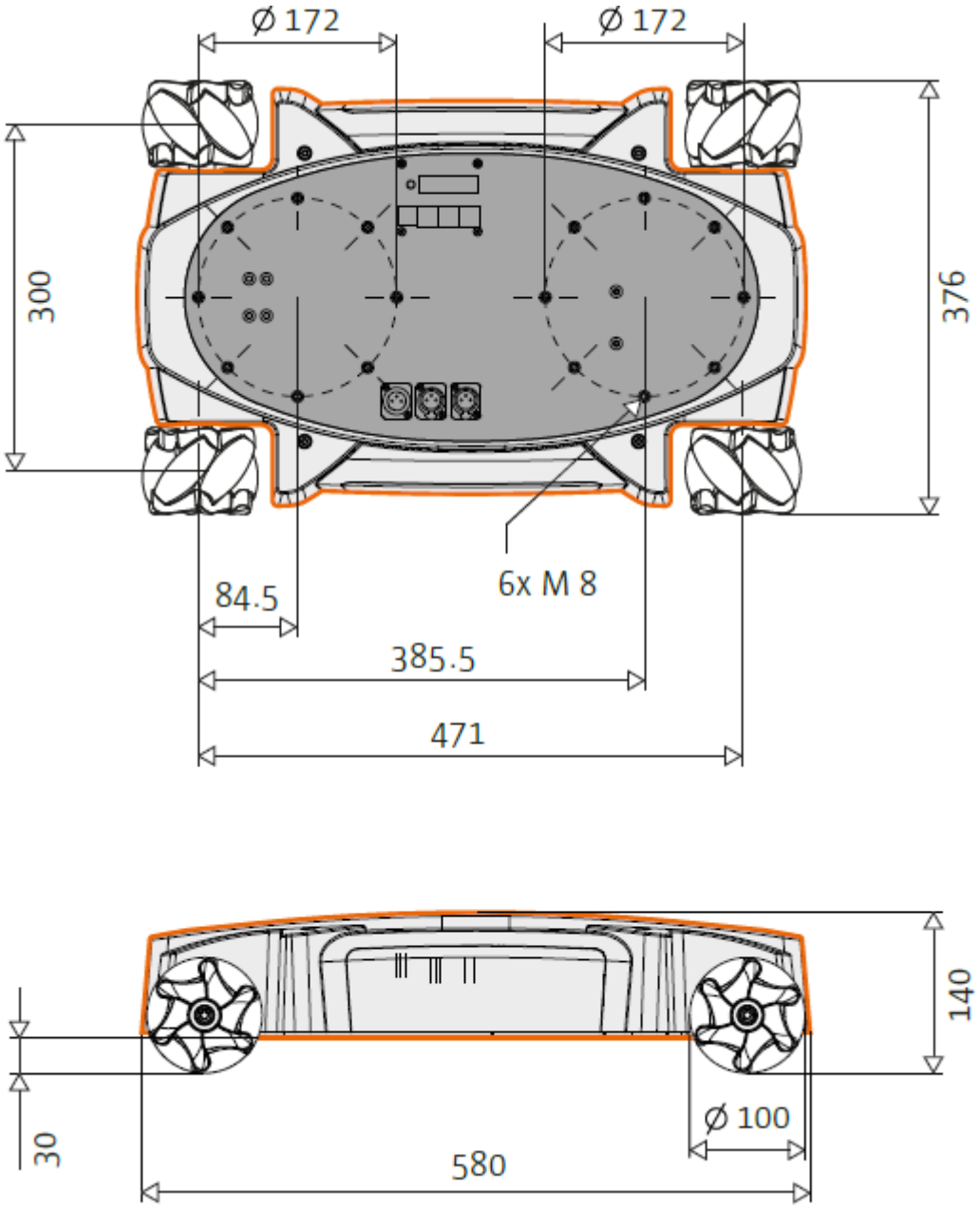

Figura 3.1 Platforma mobilă a robotului KUKA youBot [Generation Robots, 2018]

După cum se poate observa din Figura 3.1 șasiul are o lungime de 58 cm, o lățime de 38 cm și o înălțime de 14 cm și se mișcă folosind 4 roți Mecanum (Figura 3.2). Acestea sunt un tip special de roți cu rulouri montate în jurul circumferinței la un unghi de 45° față de planul roții, permițându-i robotului să combine în orice moment orice mișcare de translație și de rotație [Qian, 2017]. Acest lucru face mișcarea omnidirecțională să fie posibilă, inclusiv mișcarea laterală sau pe diagonală. În plus, platforma de 24 kg poate suporta o sarcină utilă de 20 kg.

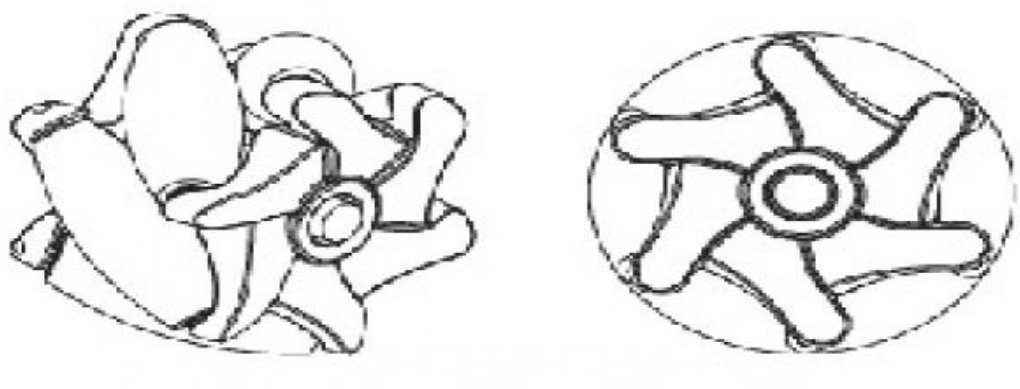

Figura 3.2 Roată Mecanum [youBot-store, 2018]

PC-ul integrat în platformă este de tip mini IT X cu unitate de procesare integrată (Intel Atom Dual Core) ce funcționează la frecvența de 667 MHz. Acesta mai dispune de 2 GB RAM, 32 GB SSD, 2 porturi LAN, 8 porturi USB și 6 porturi COM [Bischoff, 2011]. Prin includerea unui computer în robot apar noi posibilități de implementare de aplicații prin realizarea de conexiuni la porturile acestuia. De asemenea, conectând un monitor, o tastatură și un mouse, întreg procesul de programare și dezvoltare de aplicații se poate face local, astfel nu mai apare necesitatea unei conexiuni de la distanță și nici nu mai este nevoie de echipamente suplimentare.

Bateria robotului este un acumulator cu plumb reîncărcabil ce permite o utilizare continuă de până la 90 de minute.

În Tabelul 3.1 sunt sintetizate principalele caracteristici hardware ale platformei mobile.

Platforma robotică	
Cinematica omnidirecțională	4 roți Mecanum
Lungime	580 mm
Lățime	380 mm
Înălțime	140 mm
Masă	24 kg
Sarcină utilă	20 kg
Structură	Oțel
Viteză	0.8 m/s
Comunicație	EtherCAT
Tensiune de lucru	24 V

Tabelul 3.1 Caracteristicile generale ale platformei robotice [Bischoff, 2011]

3.2.2 BRAȚUL ROBOTIC

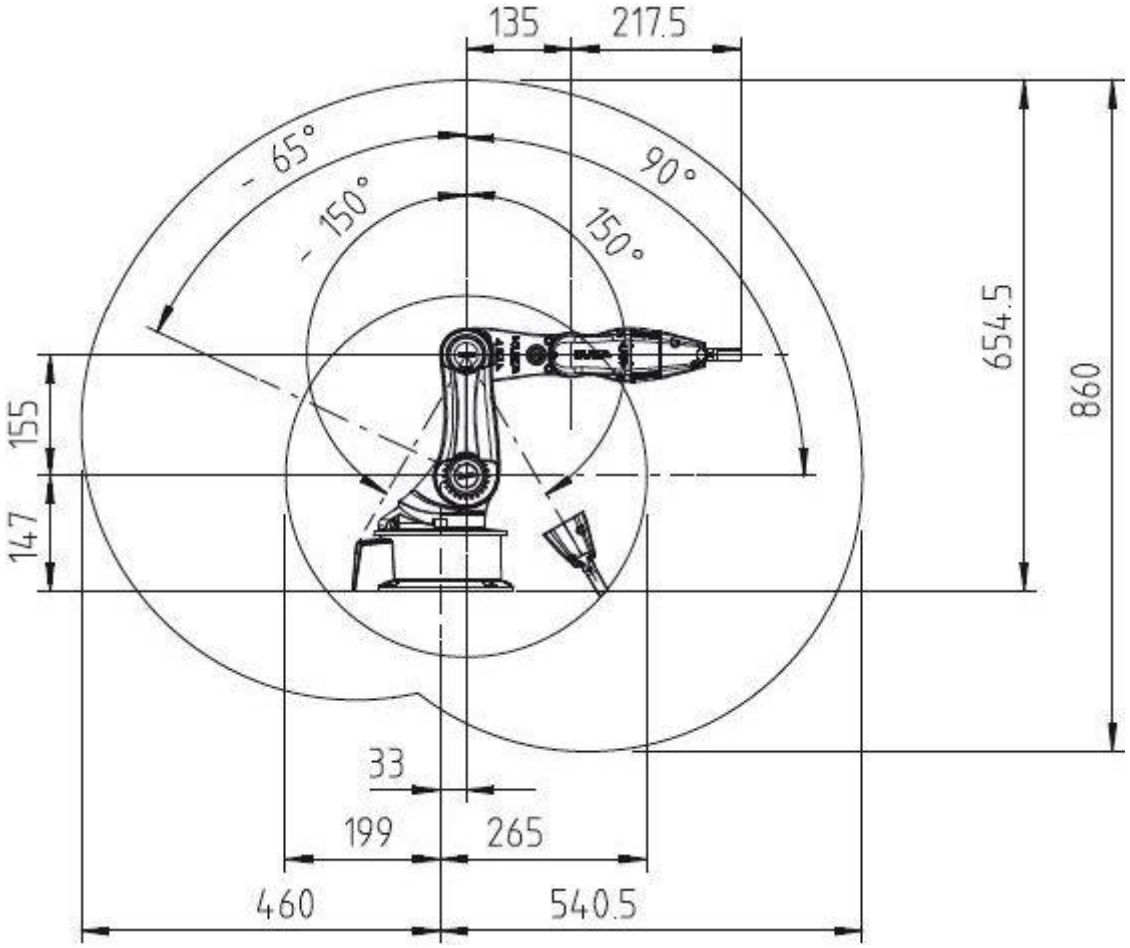

Figura 3.3 Brațul robotic al robotului KUKA youBot [Bischoff, 2011]

Brațul robotic, ilustrat în Figura 3.3, are aproximativ 66 cm în lungime și este montat pe platformă. O șină cu două degete la capătul brațului îi permite robotului să mute obiecte de până la 70 mm lungime cu o masă maximă de 500 g. Masa acestui braț este de 6,3 kg și este constituit dintr-un aliaj de magneziu

Articulațiile robotului youBot sunt prevăzute cu un set special de 5 motoare de curent continuu fără perii, iar alte 4 motoare de curent continuu fără perii sunt folosite pentru platformă. Scopul acestor motare este de a ocupa un spațiu cât mai restrâns, de a fi ușoare, dar, în același timp, de a furniza suficientă putere. În consecință, în ciuda faptului că motoarele au o greutate de 46 g până la 110 g, acestea produc o putere de 15 până la 50 W. Pentru determinarea unghiurilor articulațiilor se folosesc encodere de poziție [Schütz, 2013].

Informațiile expuse sunt sumarizate în Tabelul 3.2.

Brațul robotic	
Cinematica serială	5 axe
Înălțime	655 mm
Anvelopa de lucru	0.513 m^3
Masă	6.3 kg
Sarcină utilă	0.5 kg
Material	Aliaj de magneziu
Repetabilitatea poziției	1 mm
Comunicație	EtherCAT
Tensiune de lucru	24 V
Limitarea puterii	80 W
Cleștele	Detașabil, 2 degete

Tabelul 3.2 Caracteristicile generale ale brațului robotic [Bischoff, 2011]

3.2.3 ASUS XTION PRO

Figura 3.4 ASUS Xtion PRO [McGlaun, 2011]

ASUS Xtion PRO (Figura 2.1) este o cameră care are incorporat un senzor pentru mișcare. Senzorul a fost dezvoltat de către PrimeSense, iar ASUS și Microsoft au cumpărat o licență pentru a îl putea folosi. Pe langă o cameră color cu o rezoluție de 1280x1024 pixeli, camera are și un senzor de adâncime construit pentru distanțe între 0,8 și 3,5 m. Măsurătorile de adâncime sunt făcute combinând un laser cu o cameră cu infraroșu. Laserul își direcționează razele spre puncte din rastru, iar camera cu infraroșu înregistrează aceste puncte și calculează dinstanța de la cameră la fiecare punct. Rezultatul este reprezentat de o hartă de adâncime la o rezoluție de 640x480. În final, imaginea de adâncime este combinată cu imaginea color, rezultând o imagine în care pentru fiecare punct se cunoaște atât poziția cât și culoarea [Keiser, 2013].

3.3 COMPONENTE SOFTWARE

Anterior au fost prezentate resursele hardware ale robotului KUKA și, după cum se poate observa, acestea sunt numeroase și foarte performante. Pentru a putea profita la capacitate maximă de acestea, este nevoie de o cunoaștere a resurselor de programare prin care partea hardware poate fi accesată și modelată. Principalele componente software cu care se lucrează sunt sistemul de operare și colecția de funcții care intermediază comenzile spre zona fizică.

3.3.1 ROS

ROS (Robot Operating System) este un mediu care facilitează dezvoltarea de aplicații pentru robotică [Keiser, 2013]. Include bibliotecile și uneltele necesare pentru a produce o abstractizare hardware, a transmite mesaje, a vizualiza starea robotului, etc. Un mare avantaj al ROS este transparența. Programele sunt construite ca noduri ROS care se conectează la un singur master ROS. Fiecare nod conectat la acest master poate să asculte toate mesajele provenite de la celelalte noduri prin înscrierea la topicul corespunzător. Pe lângă mesaje, și parametrii și serviciile pot deveni disponibile pentru toate nodurile conectate la master în același mod. Utilizatorul poate interacționa cu masterul printr-o serie de comenzi simple. În acest mod, utilizatorul poate publica mesaje sau să apeleze servicii manual.

Există o componentă ROS și pentru KUKA youBot (Figura 3.5), care construiește o interfață pentru comunicația dintre ROS și driver-ul youBot care accesează partea hardware în mod direct. Această componentă se ocupă cu mesajele de comandă pentru poziția articulațiilor sau pentru viteze. De asemenea, transmite informații și despre starea în care se află robotul.

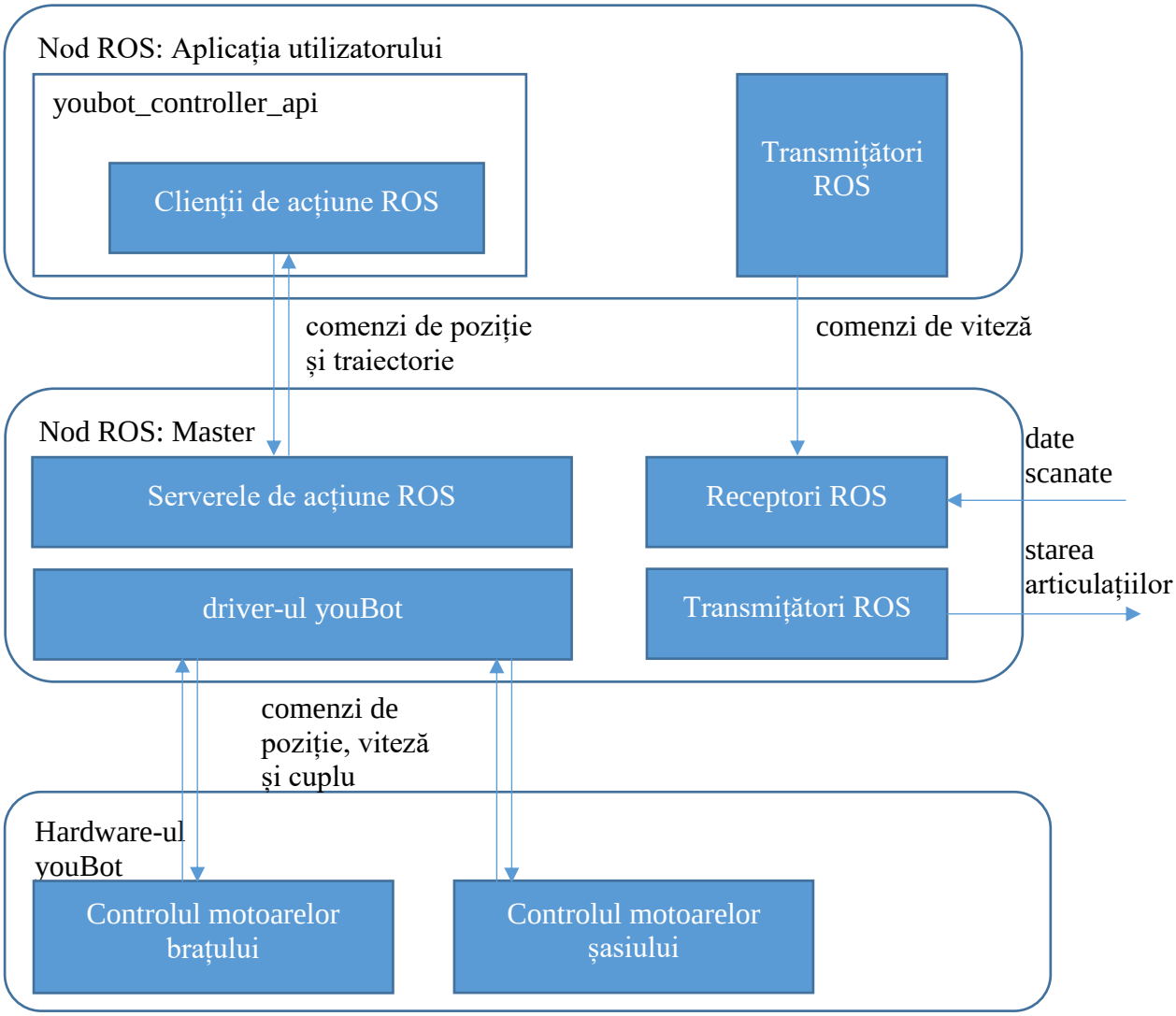

Figura 3.5 ROS al KUKA youBot [LUHbots, 2018]

3.3.2 youBot API

API-ul pentru KUKA youBot reprezintă interfața de programare prin care utilizatorul poate accesa și controla partea hardware [Manual, 2012].

În acest API brațul robotului este reprezentat ca un lanț cinematic cu 5 grade de libertate, iar platforma mobilă omnidirecțională este considerată ca fiind o colecție de articulații în care fiecare articulație este formată dintr-un motor și o cutie de viteze. Software-ul din spatele acestui API are la bază programarea obiect orientată și folosește următoarele trei clase principale:

- clasa YouBotManipulator modelează brațul robotic printr-o serie de articulații și un clește
- clasa YouBotBase reprezintă platforma
- clasa YouBotJoint reprezintă o articulație fie a brațului, fie a platformei

Componentele acestui API sunt ilustrate în continuare în Figura 3.6.

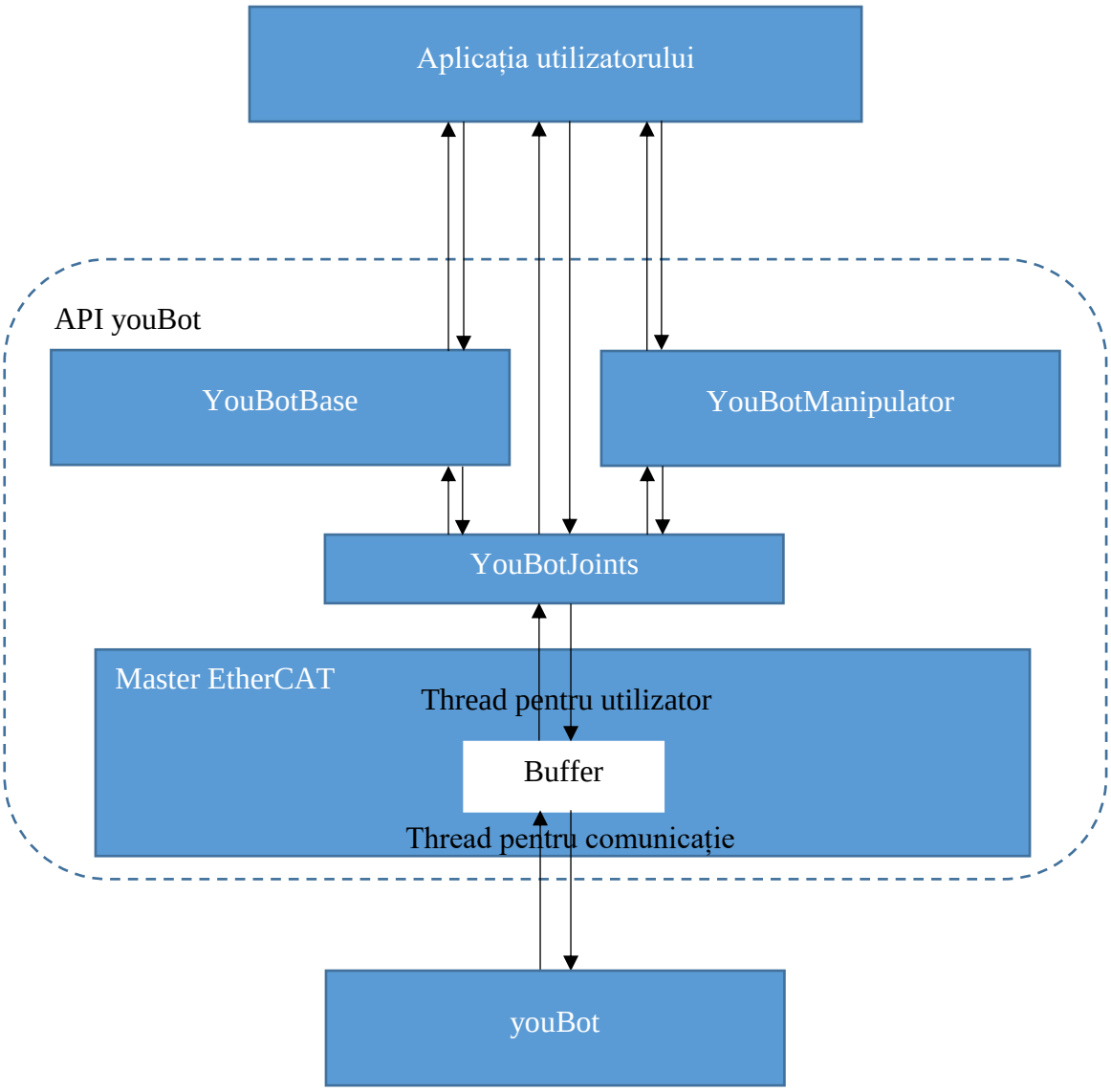

Figura 3.6 Arhitectura API-ului youBot [Robocup, 2012]

CAPITOLUL 4

REALIZAREA UNUI SISTEM DE RECUNOAȘTERE AUTOMATĂ A VORBIRII

4.1 INTRODUCERE

Un sistem de recunoaștere a vorbirii are ca scop generarea unei succesiuni de cuvine pe baza unor stimuli acustici care conțin vorbire. Un impediment care apare în dezvoltarea unor aplicații pe bază de recunoaștere de vorbire este independența față de vorbitor. Este dificil de antrenat un model acustic care să recunoască mesajele transmise indiferent de persoana care le rostește deoarece acest lucru ar presupune o bază de date de antrenare foarte cuprinzătoare, atât ca număr de clipuri audio cât și ca diversitatea acestora (200 de vorbitori diferiți). Complexitatea crește și mai mult cu cât vocabularul aplicației este mai amplu.

Aspectele menționate conduc la necesitatea de a se dezvolta inițial un sistem dependent de vorbitor și care să conțină un vocabular restrâns. De asemenea, păstrând același vocabular, se poate face o optimizare prin creșterea numărului de persoane din clipurile audio ale bazei de date și deci conferirea unei independențe față de vorbitor a sistemului.

Pentru a întruni aceste condiții a fost ales un sistem de recunoaștere a cifrelor rostite în limba română. Vocabularul folosit este unul sumar, fiind format din cele zece cifre din baza zece.

4.2 CMU SPHINX

Atât pentru această aplicație de recunoaștere de vorbire cât și pentru cea implementată pe robotul KUKA (prezentată în capitolele următoare) a fost folosit toolkit-ul CMU Sphinx. Acesta conține o serie de sisteme de recunoaștere de vorbire, precum și un antrenor pentru modele acustice. Un aspect important este acela că toate instrumentele CMU Sphinx sunt cu sursă deschisă [CMU Sphinx, 2018].

Dintre componentele CMU Sphinx poate fi amintit tool-ul PocketSphinx. Acesta este o versiune de Sphinx care poate fi folosită pe sistemele înglobate (embedded) și în jurul căruia a fost construit programul de recunoaștere vocală de pe robotul KUKA.

4.3 CONSTRUIRE ȘI OPTIMIZARE

Având în vedere resursele necesare dezvoltării unui sistem de RAV prezentate în Figura 2.3, a fost necesar să se parcurgă următoarele etape pentru obținerea tuturor elementelor care constituie baza unui astfel de sistem:

1) Înregistrarea de clipuri audio pentru formarea bazei de date

Așa cum a fost precizat anterior, sistemul a fost dezvoltat inițial ca fiind dependent de vorbitor. Acest aspect semnifică faptul că clipurile audio folosite pentru antrenare conțin vocea unui singur utilizator. A fost înregistrată o serie de 100 de fișiere audio de câte 12 cifre, folosind vocea mea.

Înregistrările au fost făcute folosind un microfon conectat la un PC, iar acestea au fost încărcate mai departe pe serverul de lucru al laboratorului SpeeD printr-o aplicație web. În prealabil a fost necesară o conectare la sistem, crearea unui nou vorbitor și selectarea vorbitorului pentru care s-au făcut înregistrările.

O înregistrare presupune rostirea a 12 cifre afișate pe monitor. În acest mod asocierea clipurilor audio cu transcrierile lor se poate face foarte simplu.

2) Compunerea listei de foneme și a dicționarului fonetic

Prin dincționarul fonetic se specifică cum se pronunță cuvintele din vocabularul aplicației, reprezentând corespondența formei scrise a unui cuvânt cu forma sa fonetică. În cadrul unui sistem de RAV, acest instrument are scopul de a lega modelul acustic de modelul de limbă.

Fonemul reprezintă unitatea fundamentală a unui sunet, unitate ce poate fi la rândul ei descompusă în subdiviziuni, denumite senone [Hwang, 1992]. O listă a fonemelor din limba română se poate găsi în Tabelul 2.2.

În cadrul acestui proiect, dicționarul conține cele zece cifre împreună cu transcrierea lor fonetică. Standardul CMU Sphinx impune ca dicționarul fonetic să fie sub forma unui fișier text în care pe fiecare linie se află cuvântul împreună cu transcrierea sa fonetică, fiecare fonemă fiind separată prin spațiu. În cazul utilizării unui vocabular redus, CMU Sphinx recomandă, pe lângă regulile specificate anterior, să se facă o diferențiere între foneme în funcție de cuvântul din care fac parte și poziția acestora în cuvânt [CMU Sphinx, 2018]. În exemplul următor se pot observa diferențele dintre o linie a unui dicționar obișnuit și o linie a unui dicționar pentru un vocabular redus.

cinci c i n c i

cinci c_cinci1 i_cinci1 n_cinci c_cinci2 i_cinci2

Se remarcă faptul că pentru toate fonemele se specifică cuvântul din care fac parte și, de asemenea, poziția acelorași fonemelor în cuvânt.

În Tabelul 4.1 se află întreg conținutul dicționarului fonetic.

Mai departe a trebuit creată lista fonemelor din componența dicționarului. Această listă va fi apoi modelată de modelul acustic. Fonemele se organizează într-un fișier text unde pe fiecare linie se află câte un fonem, iar liniile sunt sortate în ordine alfabetică.

Cuvânt	Transcriere fonetică
zero	z_zero e_zero r_zero o_zero
unu	u_unu1 n_unu u_unu2
doi	d_doi o1_doi i3_doi
trei	t_trei r_trei e1_trei i3_trei
patru	p_patru a_patru t_patru r_patru u_patru
cinci	c_cinci1 i_cinci1 n_cinci c_cinci2 i_cinci2
șase	s1_șase a_șase s_șase e_șase
șapte	s1_șapte a_șapte p_șapte t_șapte e_șapte
opt	o_opt p_opt t_opt
nouă	n_nouă o1_nouă w_nouă a1_nouă

Tabelul 4.1 Conținutul dicționarului fonetic

3) Antrenarea modelului acustic

Pentru această etapă a fost nevoie de completarea punctelor 1) și 2) în vederea obținerii resurselor necesare. Apoi a fost creat un proiect pe serverul de dezvoltare care a fost configurat pentru a putea fi folosit de CMU Sphinx și unde au fost incluse toate instrumentele dezvoltate anterior.

Mai departe au fost organizate lista clipurilor audio ce urmau a fi folosite pentru antrenare, precum și lista de transcrieri corespunzătoare acestor clipuri sub forma unor fișiere text.

Având la dispoziție toate elementele utile, pasul următor a fost să se configureze procesul de antrenare specificând atât parametrii modelului acustic ce va rezulta cât și locațiile resurselor folosite (dicționarul fonetic, lista de clipuri audio, lista transcrierilor și lista de foneme).

Înainte de antrenarea propriu-zisă mai este necesară o etapă, și anume extragerea parametrilor din semnal. Așa cum a fost specificat în *Subcapitolul 2.4*, modelarea acustică nu lucrează direct cu semnalul vorbit ci cu o serie de parametri extrași din acesta. Rezultatul obținut constă în generarea, pentru fiecare clip audio, a unui fișier cu coeficienți cepstrali.

În cele din urmă, a fost demarată antrenarea, proces care poate avea durate de timp diferite în funcție de specificațiile sistemului pe care operează și în funcție de complexitatea sarcinii. În cazul de față anternarea a durat 10 minute.

Antrenarea are la bază anumiți parametri, în funcție de care, pentru aceeași bază de date de antrenare, modelul acustic rezultat poate să ofere rezultate mai bune sau mai puțin bune. Cei mai importanți doi parametri sunt numărul de senone folosit, precum și numărul de densități Gaussiene care modelează parametrii acustici de ieșire. Importanța acestora reiese din participarea lor în construirea modelului acustic, care se face după cum urmează [Cucu, 2013].

- inițial, modelele fonetice se consideră independente de context și se face o antrenare a acestora. Modelul fonetic este dat de reprezentarea unui fonem ca fiind un set de 3 senone, iar independența de context reprezintă faptul că acel fonem este considerat același indiferent de poziția sa din cuvânt.
- modelele antrenate anterior devin dependente de context și antrenarea se reia. Celor trei senone li se schimbă semnificația, două dintre ele reprezentând vecinătățile fonemului respectiv, iar cel median, esența acestuia.

- după pasul anterior rezultă un model acustic care are un număr nedefinit de senone. În continuare, se face o uniformizare a acestora, alegând un număr fix în care trebuie să se încadreze (în mod implicit acest număr este 200). Pentru a atinge această valoare se face o agregare a fonemelor cu vecinătăți asemănătoare, iar sistemul se antrenează din nou.
- până în acest punct, antrenarea a fost făcută folosind o singură densitate Gaussiană de probabilitate. În continuare, vor urma etape de dublare a numărului de densități și de reantrenare cu densitățile rezultate până când se va ajunge la un anumit număr, prestabilit, de densități (în mod implicit acest număr este opt).
- rezultatul este constituit atât din modelul acustic pentru numărul dorit de densități, cât și din modelele rezultate la etapele intermediare. De exemplu, pentru opt densități vor fi generate și modelele pentru patru, două, respectiv una.

4) Crearea modelului de limbă

În cadrul unui sistem de recunoaștere a vorbirii cu un vocabular extins, modelul de limbă este statistic de tip n-gram. În cazul de față, vocabularul este unul foarte simplu, cuprinzând doar 10 cuvinte, deci folosirea modelului n-gram de gramatică nu este potrivită. Soluția aleasă a fost definirea unei gramatici cu reguli.

Scopul a fost dezvoltarea unei gramatici care să permită rostirea oricâtor cifre și în orice ordine. Pentru a îndeplini acest obiectiv este nevoie de adăugarea unor noduri suplimentare în structura gramaticii, care să reprezinte intrarea, ieșirea și posibilitatea de a reveni la un pas anterior. Reprezentarea grafică a regulilor menționate se regăsește în Figura 4.1.

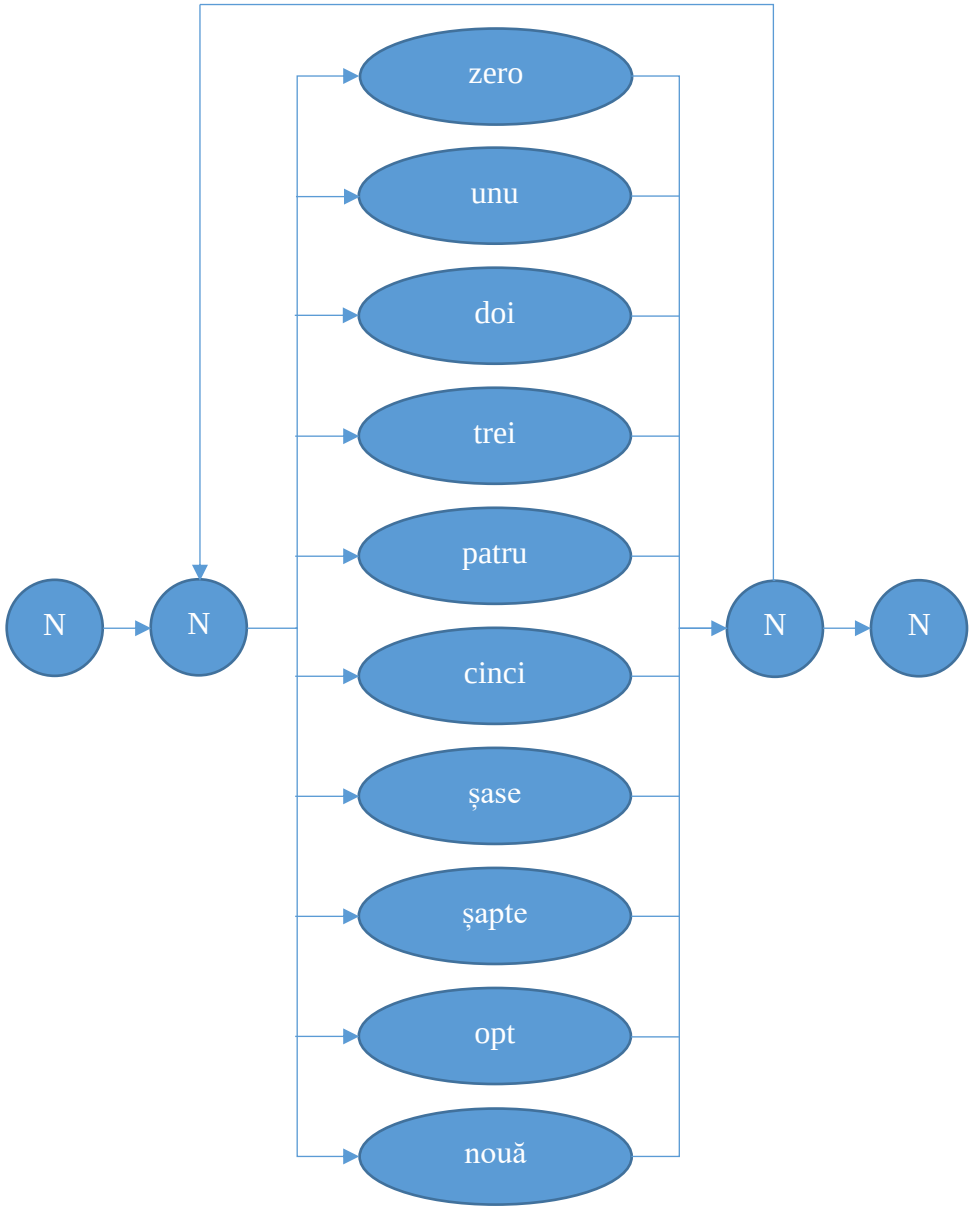

Figura 4.1 Gramatica cu reguli pentru recunoaștere de cifre

Pentru implementarea acestei FSG a fost folosit formatul JSG (Java Speech Grammar) care presupune următoarele reguli:

- antetul conține numele gramaticii: grammar rodigits
- corpul conține regula gramaticii: public<numbers> = (zero | unu | doi | trei | patru | cinci | șase | șapte | opt | nouă)*

Regula se traduce în felul următor: variabila *numbers* poate primi valorile zero sau unu sau... nouă de oricâte ori. Condiția de „sau” dintre valori este specificată prin caracterul „|”, iar repetabilitatea nelimitată a cifrelor este conferită prin caracterul „*”.

În final, a fost făcută conversia de la JSGF la FSG pentru a corespunde cu formatul intern al CMU Sphinx. În acest moment arhitectura sistemului a fost completată.

După antrenarea sistemului urmează etapa de evaluare a performanțelor acestuia prin decodarea unor clipuri audio, diferite față de cele din setul de antrenare, și compararea transcrierilor rezultate cu cele din baza de date pentru testare.

Până la un anumit punct, etapele dezvoltării mediului de testare sunt similare cu cele parcurse pentru antrenare: se extrag trăsăturile (coeficienții MFC) din semnalele audio ce conțin vorbire și

se modifică configurația sistemului astfel încât acesta să își schimbe obiectivul din antrenare în decodare.

După comanda de decodare, sistemul afișează o situație a erorilor atât la nivel de cuvânt (WER), cât și la nivel de propoziție (SER), noțiuni prezentate mai amplu în *Subcapitolul 2.6*.

Rezultatele au fost următoarele:

- WER = 8.8 %
- SER = 60 %

Această discrepanță între erori este dată de modul cum se determină SER. Astfel, dacă o anumită cifră dintr-o succesiune este transcrisă greșit atunci întreaga propoziție este marcată ca fiind eronată. În acest mod, pentru un număr relativ redus de cuvinte eronate, se poate genera o eroare mare la nivel de propoziție.

Inițial, modelul acustic a fost antrenat prin păstrarea parametrilor implicit aleși de către CMU Sphinx. În continuare a fost făcută o serie de modificări pentru a putea fi observat răspunsul sistemului și pentru a se găsi configurația optimă. Parametrii variați au fost numărul de senone și numarul de densități Gaussiene care modelează parametrii acustici. Rezultatele pentru 100 de senone sunt prezentate în Tabelul 4.2.

Nr. densități	1	2	4	8
WER[%]	12	5.4	3.8	1.2
WER_others[%]	57.1	62.7	65.8	64
SER[%]	82	61	40	13
SER_others[%]	99	92	93	93

Tabelul 4.2 Rezultatele obținute pentru 100 de senone (dependent de vorbitor)

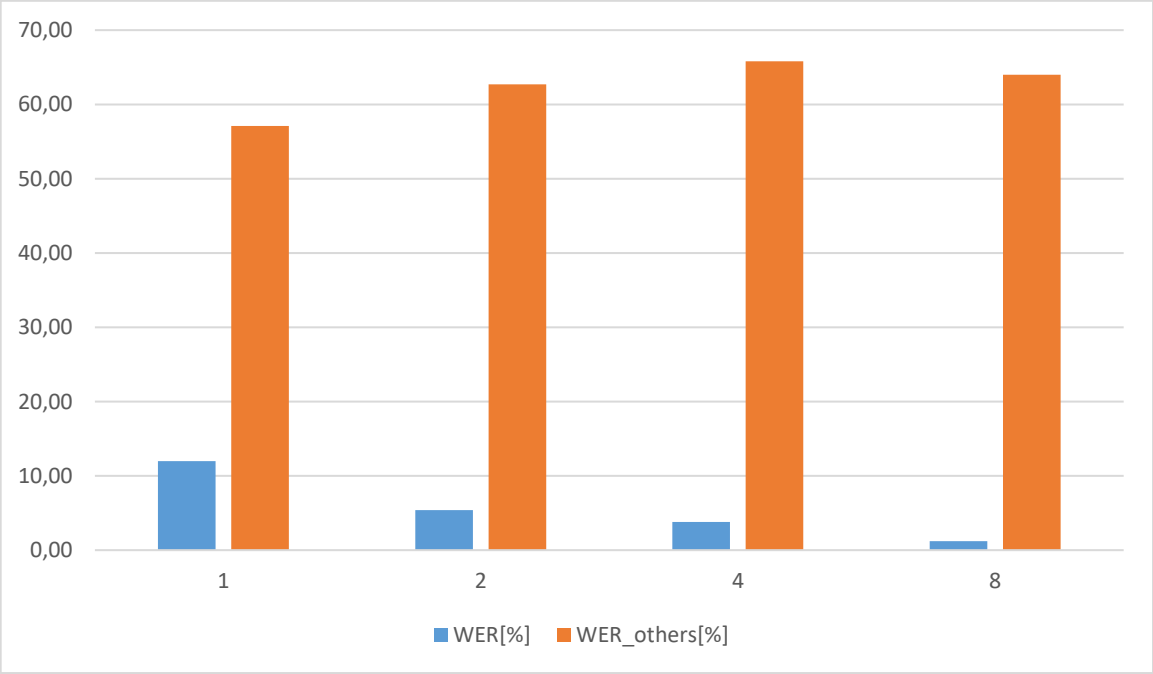

Figura 4.2 WER pentru 100 de senone (dependent de vorbitor)

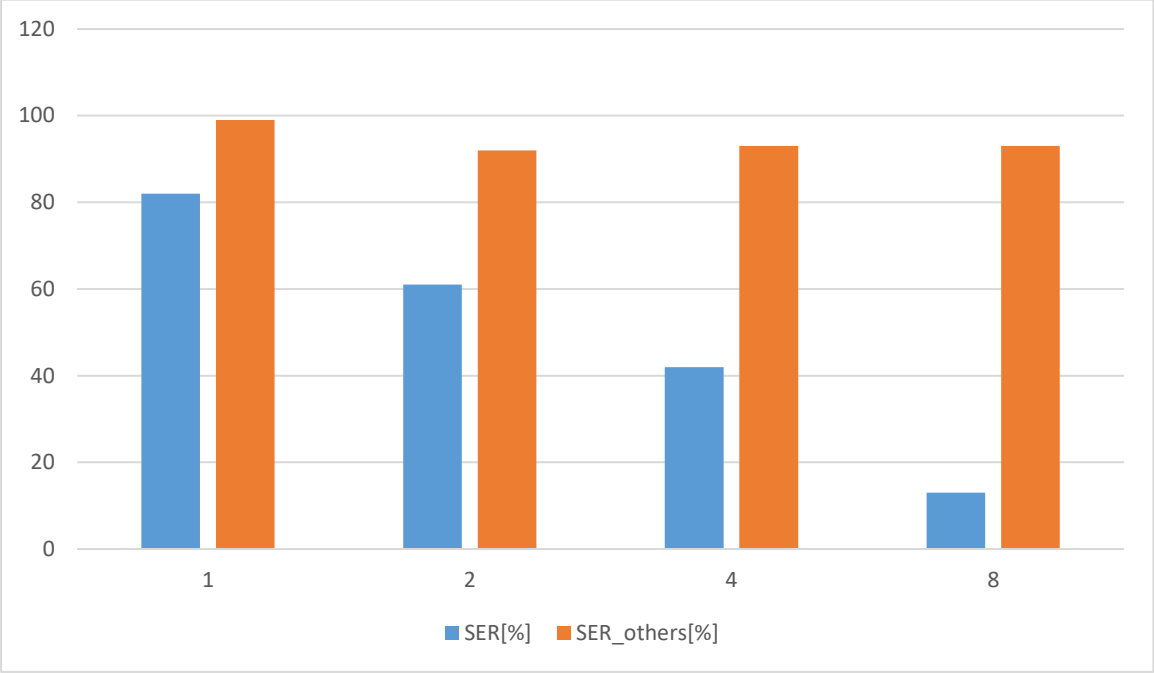

Figura 4.3 SER pentru 100 de senone (dependent de vorbitor)

Figura 4.2 și Figura 4.3 ilustrează variația WER și SER pentru 100 de senone în funcție de numărul de densități Gaussiene folosite pentru antrenare. Pentru testare au fost folosite clipuri audio conținând vocea pe care a fost făcută antrenarea (vocea mea), dar care prezentau o informație diferită față de clipurile din setul de antrenare. Rezultatele obținute au fost notate cu WER, respectiv SER. În plus, s-a realizat și o testare pe o bază de date care conținea vocea altor utilizatori, iar rezultatele au fost notate cu WER_others, respectiv SER_others.

Se poate observa că, atât la nivel de cuvânt, cât și la nivel de propoziție, erorile au fost mult mai mari în cazul vocii unui utilizator străin. Acest lucru se datorează faptului că modelul acustic a fost construit strict pe baza semnalelor audio provenite de la un singur utilizator, de unde și denumirea sistemului de „dependent de vorbitor”. Concluzia este aceea că performanțele aplicației pot fi măsurate doar în raport cu parametrii WER și SER deoarece doar aceștia sunt relevanți în contextul unui sistem construit pentru recunoașterea vorbirii de la o singură persoană.

În plus, se remarcă tendința descrescătoare a ratelor de eroare odată cu creșterea numărului de densități Gaussiene.

În continuare, Tabelul 4.3 prezintă rezultatele obținute în urma schimbării numărului de senone la 200.

Nr. densități	1	2	4	8
WER[%]	14.1	9.2	5.4	8.8
WER_others[%]	67.1	70	68.8	78.5
SER[%]	77	68	50	60
SER_others[%]	99	97	99	100

Tabelul 4.3 Rezultatele obținute pentru 200 de senone (dependent de vorbitor)

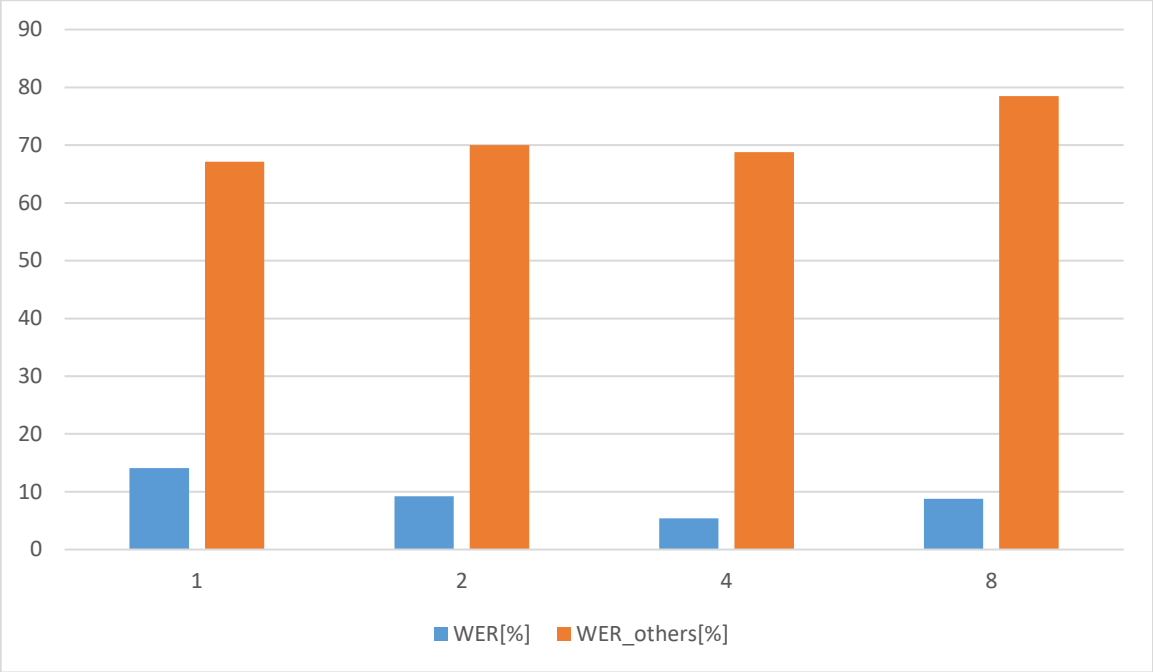

Figura 4.4 WER pentru 200 de senone (dependent de vorbitor)

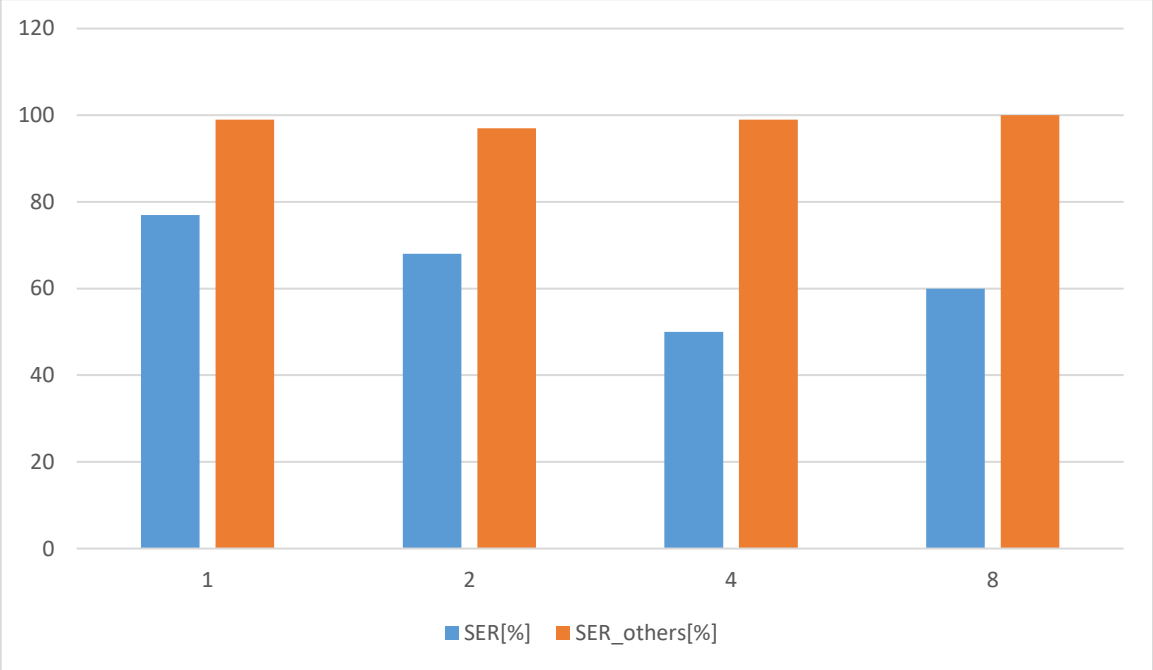

Figura 4.5 SER pentru 200 de senone (dependent de vorbitor)

Comparând rezultatele din Figura 4.4 și Figura 4.5 cu cele din Figura 4.2, respectiv Figura 4.3, se constată că folosirea a 200 de senone conduce la o funcționare mai slabă a sistemului. Acest rezultat era de așteptat deoarece, având un vocabular atât de restrâns, vecinătățile posibile pentru un fonem sunt și ele restrânse. Deci, folosirea unui număr atât de mare de senone conduce la încercarea de diferențiere între foneme în situații în care nu este nevoie, ceea ce generează erori suplimentare.

În sprijinul afirmației anterioare se aduce următorul calcul: având 39 de foneme diferite enumerate în Tabelul 4.1, fiecare dintre acestea va fi reprezentat prin 3 senone: unul care să modeleze influența fonemului vecin din stânga, unul pentru fonemul din dreapta, iar al treilea reprezintă forma de bază a fonemului curent. Așadar, pentru vocabularul folosit, un număr de 117 senone acoperă toate situațiile posibile. Între aceste situații există și similarități, de exemplu, fonemul „a”

din „șase” are același vecin stâng precum fonemul „a” din „șapte.” Așadar, numărul optim de senone poate fi chiar mai mic.

Forțând sistemul să folosească 200 de senone se impune o divizare suplimentară a posibilelor vecinătăți, rezultând o eroare mai mare.

În concluzie, optimul găsit pentru sistemul de recunoaștere de cifre rostite în limba română dependent de vorbitor, este de 100 de senone în combinație cu 8 densități Gaussiene.

Optimizarea aplicației o constituie obținerea unei independențe de vorbitor. Acest obiectiv se poate realiza prin extinderea bazei de date, care nu va mai conține clipuri audio înregistrate de un singur utilizator, ci va cuprinde voce de la mai multe persoane (aproximativ 150). În rest, se parcurg aceleași etape prezentate anterior.

Pentru că deja a fost făcută observația că 100 de senone modelează mai bine vocabularul ca 200, pentru sistemul independent de vorbitor se va face antrenarea doar pe 100 de senone. Această alegere se datorează și duratei mai mari de timp pe care o ocupă antrenarea în raport cu cazul dependent de vorbitor. Rezultatele au fost notate în Tabelul 4.4.

Nr. densități	1	2	4	8
WER[%]	15.7	9.2	5.1	3.6
WER_others[%]	13.3	6.4	5.3	3.8
SER[%]	63.7	47.9	36.5	22.5
SER_others[%]	66.2	44.7	42.6	34.1

Tabelul 4.4 Rezultatele obținute pentru 100 de senone (independent de vorbitor)

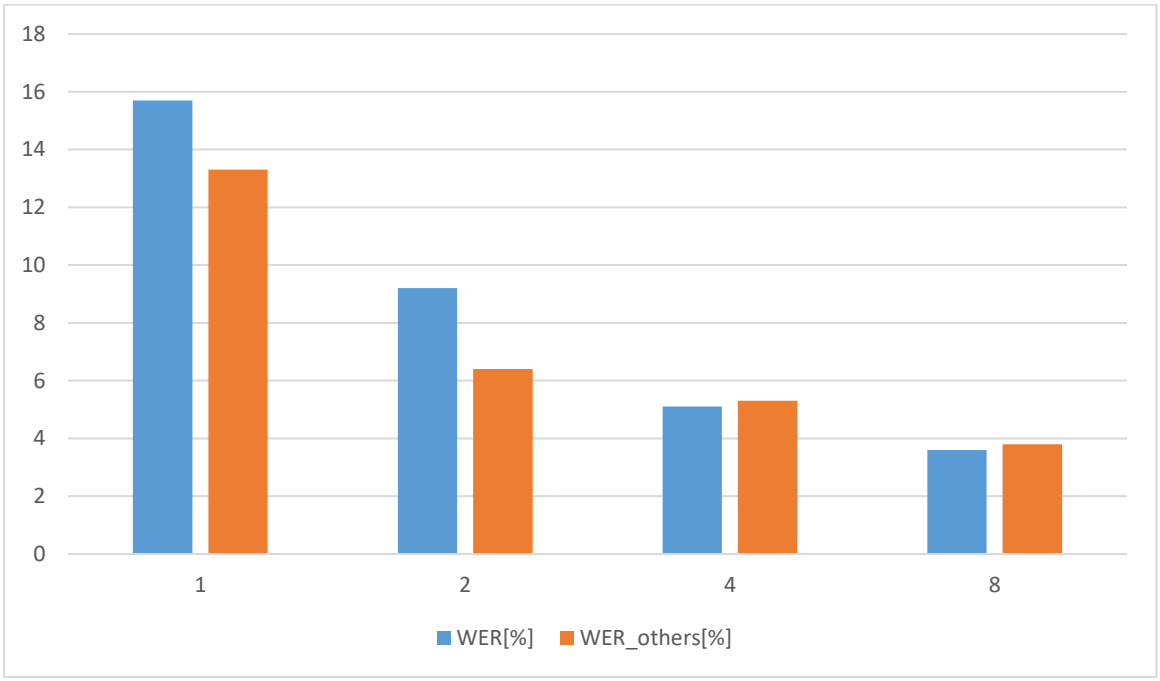

Figura 4.6 WER pentru 100 de senone (independent de vorbitor)

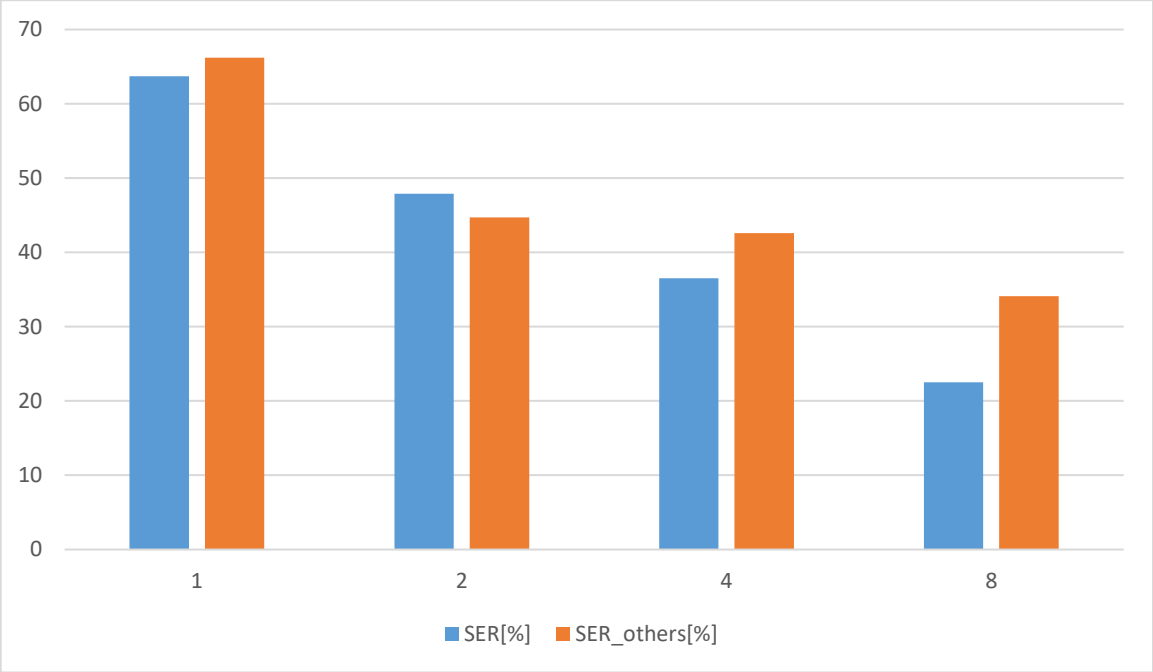

Figura 4.7 SER pentru 100 de senone (independent de vorbitor)

Se poate observa din Figura 4.6 și Figura 4.7 că, în cazul sistemului independent de vorbitor, rezultatele în cazul recunoașterii vocilor din baza de date (WER, respectiv SER) sunt foarte asemănătoare cu cele pentru vorbitori străini (WER_others, respectiv SER_others). Acest aspect demonstrează generalitatea aplicației, care acum este capabilă să recunoască mesajul rostit de orice vorbitor cu o eroare acceptabilă.

Comparând cu sistemul dependent de vorbitor, se remarcă o creștere a erorii per cuvânt (3.6 % față de 1.2 %) pentru utilizatorii cunoscuți, dar, în cazul persoanelor necunoscute, eroarea se îmbunătățește semnificativ (3.8 % față de 64 %).

Pe baza modelului acustic care oferă cele mai scăzute erori (100 senone cu 8 densități Gaussiene) a fost creată o primă versiune a sistemului de recunoaștere de comenzi implementat pe robotul KUKA.

CAPITOLUL 5

CONTROLUL ROBOTULUI KUKA

5.1 CONTROLUL PLATFORMEI

Așa cum a fost prezentat în *Subcapitolul 3.3.2,* controlul actuatoarelor platformei robotice KUKA youBot se face folosind o serie de funcții organizate într-un API.

Pentru a determina modul de funcționare a API-ului robotului s-a pornit de la un exemplu de control cu ajutorul unui joystick în care este exemplificat modul de construcție și de apel pentru funcțiile necesare manipulării tuturor articulațiilor. Exemplul a fost dezvoltat în limbajul de programare C++ și are la baza programarea obiect orientată.

Realizarea funcțiilor pentru controlul bazei, respectiv al brațului are la bază o instanță a clasei YouBotBase, respectiv o instanță a clasei YouBotManipulator. Aceste obiecte sunt realizate precizând constructorilor calea către fișierul de configurare a driver-ului robotului, precum și tipul de controler ce va fi instanțiat, spre exemplu „youbot-manipulator”.

Există patru funcții principale prin care se poate comanda robotul:

- funcția de inițializare: `init_youbot()`.
- funcția pentru controlul brațului robotic: `move_arm(float, float, float, float, float)`.
- funcția pentru controlul platformei: `move_base(double, double, double)`.

- funcția pentru încetarea mișcării: `stop()`.

În cadrul funcției de inițializare se realizează o comutație sinusoidală a tuturor actuatoarelor, în plus se calibrează pozițiile de referință ale celor cinci articulații din cadrul brațului. Apoi se poziționează brațul într-o postură inițială prin apelul funcției `move_arm` și se întrerupe activitatea programului pe o perioadă care să îi permită robotului să execute mișcarea.

Funcția `move_arm` primește ca parametri pozițiile în care utilizatorul dorește să ajungă fiecare din cele cinci motoare ale manipulatorului. Folosind instanța de YouBotManipulator se aleg articulațiile dorite și li se specifică poziția în care trebuie să ajungă.

În cazul funcției `move_base`, parametrii nu sunt valori absolute ci reprezintă viteza de mișcare longitudinală, transversală și, respectiv, unghiulară. Acești parametri sunt prelucrați, apoi preluați de instanța de YouBotBase și transmiși pe post de comenzi de viteză către motoare. Consecința este că, odată apelată această funcție cu un set de valori diferite de zero, robotul își va continua mișcarea pe o perioadă nedeterminată, așteptând o comandă de oprire. Comanda de încetare a deplasării este transmisă prin funcția `stop`. Aceasta reprezintă un apel al funcției `move_base` care suprascrie datele referitoare la viteză cu valori nule.

Atât funcțiile prezentate cât și funcțiile necesare pentru realizarea încapsulării (getter și setter) sunt înglobate în clasa RobotMovement, care, de asemenea, conține și un constructor și un destructor.

5.2 CONTROLUL PRIN COMENZI VOCALE ÎN LIMBA ROMÂNĂ

Odată clarificat modul prin care se poate controla robotul, pasul următor a fost generarea comenzilor de mișcare ca răspuns la mesaje de comandă transmise prin voce.

Pentru a folosi modelele dezvoltate anterior, a fost aleasă modalitatea de transmitere de comenzi prin cifre. În acest mod, pentru o anumită succesiune de trei cifre robotul va efectua o anumită mișcare. Numărul de trei cifre din componența unei comenzi a fost ales pentru a preîntâmpina eventualitatea unei comenzi recunoscute eronat în cazul transcrierii greșite a unei cifre. De asemenea, codul comenzii nu a fost considerat mai lung pentru a nu îngreuna sarcina de detectare și pentru a minimiza SER.

Integrarea unui model obținut în urma antrenării într-o aplicație a fost făcută pornind de la exemplele oferite de CMU Sphinx. Aceste exemple folosesc niște modele generice pentru limba engleză pentru a afișa pe ecran mesajul transmis prin voce și sunt dezvoltate în mai multe limbaje de programare. Având în vedere că API-ul folosit de youBot este scris în C++, a fost ales exemplul cel mai apropiat de acest caz, și anume varianta implementată în C.

Evoluția sistemului va fi dată de schimbarea ulterioară a modelului acustic antrenat anterior cu un model acustic general, folosit cu acordul laboratorului SpeeD, în care nu mai există o limitare în ceea ce privește cuvintele folosite. Acest model face posibilă generarea de comenzi ca succesiuni de cuvinte, oferind aplicației mai multă flexibilitate și ușurință de folosire.

5.2.1 UTILIZAREA TOOLKIT-ULUI POCKETSPHINX

În primul rând, a fost nevoie de descărcarea și instalarea tool-ului pe robot. După cum este menționat în *Subcapitolul 3.2*, platforma are integrat un computer, iar sistemul de operare folosit de acest computer este o versiune de Ubuntu modificată pentru ROS. Așadar, descărcarea și instalarea au decurs normal, urmând pașii indicați de dezvoltatorul CMU Sphinx [CMU Sphinx, 2018].

Deoarece proiectul final urma să aibă un număr mai mare de fișiere sursă, s-a optat ca compilarea să nu se realizeze folosind linia de comandă, ci să se facă pe baza unui Makefile.

Un Makefile este un fișier special care conține comenzi ce pot fi înțelese de terminalul Linux. Avantajul îl constituie faptul că se pot gestiona mult mai ușor dependințele tuturor fișierelor sursă, în plus compilarea acestor fișiere se face mult mai simplu, transmițând comanda "make" în terminal, în directorul în care se află acest fișier. De asemenea, recompilarea va ține cont doar de fișierele în care au fost făcute modificări, reducând semnificativ timpul de compilare.

Componentele cele mai importante ale exemplului de folosire a pocketsphinx sunt functia `main()`, respectiv functia de recunoaștere propriu-zisă: `recognize_from_microphone()`.

În funcția main se inițializează variabila care reține configurația sistemului, acesteia i se precizează calea către modelul acustic ce urmează a fi folosit, către modelul de limbă și către dicționarul aplicației. Pe baza variabilei de configurare se inițializează o variabilă (`ps`) în jurul căreia se dezvoltă întregul proces de recunoaștere de vorbire. Ultima instrucțiune din main apelează funcția `recognize_from_microphone()`.

În continuare este prezentat un exemplu de inițializare a configurației sistemului de RAV:

```
config = cmd_ln_init(NULL, ps_args(), TRUE,
                     "-hmm", MODELDIR "/cd_cont/",
                     "-jsgf", MODELDIR "/grammar.gram",
                     "-dict", MODELDIR "/dictionary.dic",
                              NULL);
```

Primul argument al funcției `cmd_ln_init` specifică dacă se suprascrie sau nu o linie de comandă precedentă, următorul reprezintă un șir de nume de argumente pe care funcția le poate adminte, iar al treilea specifică dacă funcția să se oprească sau nu la introducerea unui argument invalid sau în cazul duplicatelor. Urmează apoi specificarea fiecărui parametru din arhitectura sistemului de recunoaștere a vorbirii. Întâi este specificat tipul parametrului (model acustic, model de limbă sau dicționar) și apoi calea către elementul respectiv. `MODELDIR` este un macro care conține deja adresa directorului unde sunt stocați parametrii, urmând a se mai adăuga doar denumirea fișierelor folosite.

Utilizarea funcției `recognize_from_microphone()` este strâns legată de variabila `ps`. Toate funcțiile care intră în componența procesului de recunoaștere a vorbirii primesc ca parametru această variabilă. Etapele de funcționare sunt următoarele:

1) se verifică disponibilitatea dispozitivelor audio de intrare
2) se inițializează etapa de ascultare (mesaj de Ready)
3) se verifică dacă sunetul de la microfon depășește un anumit prag
4) se începe ascultarea (mesaj de Listening)
5) se stochează datele primite de la microfon într-o coadă
6) se procesează datele obținute
7) se generează și se afișează transcrierea mesajului vocal
8) se reiau pașii 2)-7) până când programul este întrerupt

În plus, se poate regla sensibilitatea sistemului prin setarea unui prag de zgomot diferit de cel implicit, în funcție de care sistemul va fi mai alert sau mai puțin alert la schimbările de intensitate de la intrarea audio.

5.2.2 GENERAREA DE COMENZI CA SUCCESIUNI DE CIFRE

După ce exemplul pentru limba engleză a fost implementat pe robot cu succes, etapa următoare a fost înlocuirea parametrilor prestabiliți din exemplu cu cei creați în cadrul proiectului de recunoaștere de cifre. Astfel, modelul acustic general pentru limba engleză, dicționarul și modelul de limbă probabilistic au fost substituite cu modelul acustic creat pentru cifrele rostite în limba română, cu dicționarul fonetic corespunzător și, respectiv cu gramatica cu reguli dezvoltată anterior.

Lista de comenzi a cuprins o înșiruire de trei cifre cât mai variată pentru a nu exista posibilitatea de confuzie între comenzi ca urmare a unei erori de transcriere. Comenzile erau următoarele:

- „000” – translație înainte
- „111” – translație înapoi
- „222” – translație dreapta
- „333” – translație stânga
- „444” – rotație dreapta
- „555” – rotație stânga
- „123” – mișcarea actuatoarelor 3 și 4 ale brațului robotic

Transcrierea obținută în urma recunoașterii era comparată cu cele șapte comenzi din listă, iar în cazul unei potriviri, directiva corespunzătoare era transmisă robotului. Dacă nu era recunoscută nicio comandă atunci era transmis un mesaj de informare, iar robotul revenea în starea de ascultare.

Pentru partea de efectuare a mișcărilor a fost creată o instanță a clasei RobotMovement, realizându-se astfel o combinație între abordarea procedurală din exemplul oferit de Sphinx cu implementarea obiect orientată a API-ului KUKA. În funcție de comanda înregistrată erau apelate funcțiile `move_arm`, respectiv `move_base` cu parametrii corespunzători pentru a se realiza mișcarea dorită.

O abordare ineficientă a fost păstrarea gramaticii cu reguli din proiectul anterior, gramatică în care era acceptată rostirea oricărei cifre de un număr oricât de mare de ori. Din cauza acestei permisivități au fost cazuri în care s-au înregistrat grupuri mai mari sau mai mici de trei cifre, iar rata de eroare per propoziție era ridicată. Problema prezentată a fost totuși ocolită prin considerarea în etapa de comparație doar a primelor trei cifre din transcrierea generată de sistem.

O îmbunătățire ar fi fost construirea unui model de limbă care să includă doar comenzile din listă. În acest fel sistemul era constrâns să aproximeze mesajul recepționat la una din aceste comenzi, iar SER ar fi scăzut.

5.2.3 TRECEREA LA UN MODEL ACUSTIC GENERAL

Folosirea unui set de comenzi bazat pe cifre ar fi fost contraintuitivă pentru un utilizator care lua prima dată contact cu aplicația. De asemenea, apărea necesitatea memorării semnificației fiecărei secvențe de cifre, iar pentru o extindere a numărului de comenzi efortul de a reține și de a păstra evidența listei nu părea justificat. Astfel, s-a dorit trecerea de la un sistem de comandă bazat pe cifre, la unul bazat pe cuvinte din limba română.

Deși ar putea fi considerat un pas simplu, această trecere nu ar fi fost posibilă fără a folosi un model acustic general. Un model acustic general este antrenat pentru a modela întreg vocabularul unei limbi. Deci, în acest caz, vocabularul este extins ceea ce complică mult etapa de antrenare.

Pentru obținerea unui model acustic care să ofere o performanță bună este nevoie de o bază de date etichetată care să conțină sute de ore de vorbire. Obținerea unei asemenea baze de date este dificilă deoarece nu există înregistrări deja disponibile pentru antrenare în limba română, iar

adunarea într-un timp scurt a acestora este o sarcină imposibilă. În plus, antrenarea pe un calculator personal ar fi durat o perioadă inacceptabil de lungă. Aceste impedimente au condus la imposibilitatea de a dezvolta un sistem propriu de recunoaștere a cuvintelor în limba română. În consecință, a fost nevoie de folosirea unuia deja implementat de laboratorul SpeeD.

Având la dispoziție modelul acustic, următoarea sarcină a fost dezvoltarea unei noi liste de comenzi care să cuprindă de data aceasta cuvinte:

- „mergi înainte” – translație înainte
- „mergi înapoi” – translație înapoi
- „mergi dreapta” – translație dreapta
- „mergi stânga” – translație stânga
- „rotește dreapta” – rotație dreapta
- „rotește stânga” – rotație stânga

Efectuarea mișcărilor a fost păstrată în esență, apărând modificări doar la nivelul de asociere a șirului de caractere obținut în urma transcrierii cu comanda corespunzătoare.

S-a observat că, în timpul utilizării, sistemul rămâne în urmă după recunoașterea a două comenzi, fenomen manifestat prin efectuarea de mișcări, comandate anterior, în perioada în care utilizatorul nu transmite niciun mesaj. Acest lucru se datorează puterii de procesare limitate a computerului integrat în robot deoarece nu doar antrenarea modelului acustic este consumatoare de resurse ci și folosirea lui.

O primă soluție de optimizare a fost oferită în *Subcapitolul 5.2.2*, și anume, înlocuirea modelului de limbă probabilistic cu unul bazat pe o gramatică cu stări. Astfel, se elimină o parte din efortul computațional, iar sistemul ar putea funcționa mai fluid.

5.2.4 CREAREA GRAMATICILOR CU REGULI

Pentru aplicația de control prin voce a fost creată o gramatică ale cărei reguli înglobează întreaga listă de comenzi. Antetul gramaticii reține numele acesteia, iar corpul va conține regula ce rezumă întreg setul de comenzi: (mergi (înainte | înapoi | dreapta | stânga) | rotește (dreapta | stânga)). O reprezentare grafică este prezentată în Figura 5.1.

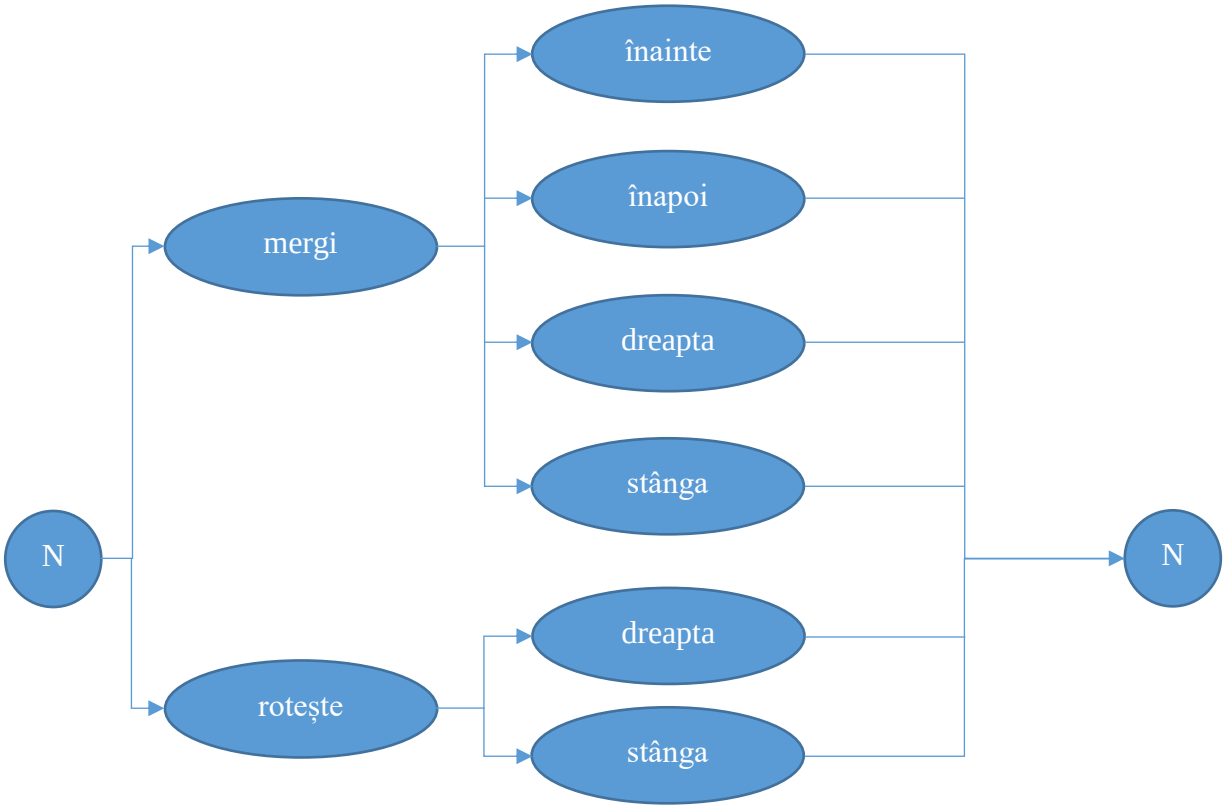

Figura 5.1 Gramatica cu reguli pentru comanda robotului

În această gramatică singurele cuvinte de la care se poate porni sunt „mergi”, respectiv „rotește”. În momentul în care unul dintre aceste două cuvinte a fost recunoscut, se dezvoltă următoarele opțiuni de completare a propoziției. Regula poate fi exprimată în felul următor: „după cuvântul mergi nu poate urma decât cuvântul înainte sau înapoi sau dreapta sau stânga, sau după cuvântul rotește nu poate urma decât cuvântul dreapta sau stânga”. Această rezumare compactă este posibilă datorită similarității comenzilor din listă. Dacă se va dori extinderea setului de comenzi, atunci și gramatica va trebui dezvoltată cu reguli noi.

Un alt model de limbă cu stări a fost creat pentru a ușura interacțiunea cu robotul. Acesta are în vedere fuziunea viitoare a aplicației de recunoaștere de voce cu o altă aplicație deja existentă pe platformă. Scopul acestei gramatici este de a oferi posibilitatea eliminării altor surse de intrare, cu excepția vocii. Regula acestei gramatici este foarte simplă: da | nu. Regula menționată nu a fost inclusă în precedenta gramatică deoarece s-a dorit ca, în comunicarea cu platforma robotică, privilegiul transmiterii de comenzi să fie doar al unei anumite categorii de utilizatori, și anume, utilizatorii înscriși și recunoscuți de către robot.

Cu toate că modificarea tipului de gramatică a adus o îmbunătățire în rapiditatea de răspuns a sistemului, aceasta nu a fost suficient de marcantă astfel încât să rezolve complet problema latenței de răspuns. În consecință, s-a apelat la o nouă variantă de soluționare și anume combinarea recunoașterii automate a vorbirii cu detecția de cuvinte cheie.

5.2.5 Detecția de cuvinte cheie în combinație cu Recunoașterea Automată a Vorbirii

Detecția de cuvinte cheie (keyword spotting – KWS) se ocupă de identificarea anumitor cuvinte din cadrul unei rostiri. Acest tip de recunoaștere de vorbire este foarte util în contextul identificării unor porțiuni de conversații în care au fost rostite anumite cuvinte, ușurând mult sarcina de căutare în baze de date de voce mari.

Un caz special de detecție de cuvinte cheie îl reprezintă cuvântul de trezire, folosit de asistenți vocali precum Siri, Alexa sau Google. În acest mod, sistemul se află într-o stare de așteptare, rejectând toate celelalte cuvinte diferite de cuvântul de trezire. Astfel, se produce o reducere a consumului de energie deoarece dispozitivul nu analizează permanent inputul audio. În plus, modul KWS ușurează și sarcina de procesare fiindcă întotdeauna va fi recunoscut un număr mic de cuvinte.

Pentru a profita de avantajele KWS a fost considerată o cascadare (Figura 5.2) a acestui mod cu RAV. Precum asistenții vocali amintiți anterior, robotul va manifesta o stare în care va aștepta un singur cuvânt cheie. După identificarea cuvântului se va trece la etapa de recunoaștere de comenzi care se va desfășura după cum a fost prezentat în subcapitolele anterioare.

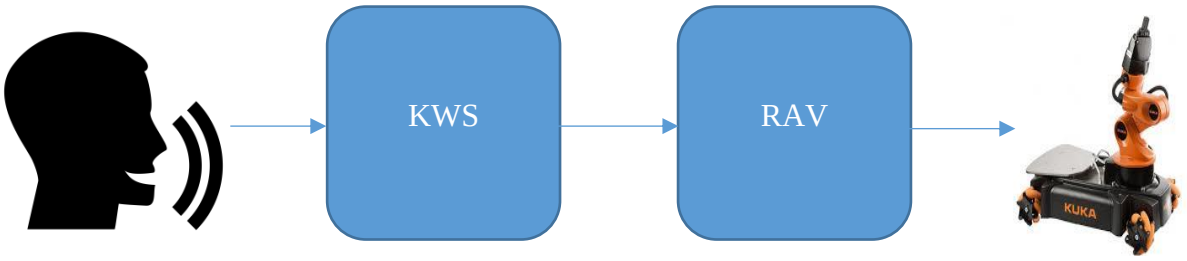

Figura 5.2 Cascadarea KWS cu RAV

Detecția de cuvinte cheie presupune o listă de cuvinte pentru care sistemul să reacționeze. Pentru a spori pe cât posibil eficiența, lista menționată trebuie să fie cât mai scurtă. În cazul aplicației curente s-a decis ca lista să conțină un singur cuvânt, și anume cuvântul „kuka”.

Crearea listei nu este suficientă, aceasta specificând doar ce cuvinte pot fi detectate în prima etapă. Mai departe, sistemul trebuie ajutat să determine cuvintele prin recunoaștere de voce. Acest aspect presupune o completare a dicționarului fonetic cu noile cuvinte cheie.

În mod implicit, „kuka” nu este un cuvânt din vocabularul limbii române și deci nu aparține nici dicționarului asociat cu modelul acustic general. Așadar, este necesară includerea lui în dicționar, împreună cu transcrierea sa fonetică. Linia nouă va fi inclusă conform ordonării alfabetice, iar conținutul acesteia este următorul:

```
kuka k u k a
```

În cadrul proiectului prezentat, programul va debuta prin căutarea de cunvinte cheie, stare impusă prin funcția ps_set_search. De asemenea, apare necesitatea creării unei cozi suplimentare în care să fie stocată intrarea audio pentru KWS. În cazul în care cuvântul „kuka” a fost recunoscut, se face trecerea la gramatica cu regului prin aceeași funcție `ps_set_search` și se demarează o nouă etapă de ascultare, de data aceasta pentru comenzile structurate în gramatică. După recunoașterea unei comenzi, modul de căutare se reinițializează la detecția de cuvinte cheie și ciclul se reia. Dacă nu a fost recunoscut cuvântul de trezire, atunci sistemul își păstrează starea și continuă ascultarea pentru acel cuvânt.

Inițializarea modului de căutare se face în felul următor:

```
ps_set_kws(ps, "kws", PATH "/keyphrase.list");
ps_set_search(ps, "kws");
```

Prima linie specifică lista de cuvinte cheie care urmează a fi folosită și, de asemenea, îi atribuie și o denumire mai scurtă care va fi pasată funcției de selectare a modului de căutare, funcție exemplificată pe următoarea linie.

Pentru trecerea la modul de recunoaștere folosind gramatica cu stări, etapele sunt asemănătoare:

```
ps_set_jsgf_file(ps, "jsgf", grammar);
```

Inițial, i se atribuie o etichetă gramaticii cu care se dorește să se realizeze recunoașterea de comenzi. Parametrul `grammar` conține informații despre gramatică din configurația inițială a sistemului (*Subcapitolul 5.2.1*):

```
grammar = cmd_ln_str_r(config, "-jsgf");
```

Apoi, după recunoașterea cu succes a cuvântului cheie, se trece la căutarea pe baza gramaticii, lucru ilustrat în următoarea linie de cod:

```
ps_set_search(ps, "jsgf");
```

A fost remarcată o îmbunătățire semnificativă a răspunsului sistemului, acesta apropiindu-se mult mai mult de o interacțiune în timp real.

CAPITOLUL 6

INTEGRAREA APLICAȚIEI DE CONTROL VOCAL CU O APLICAȚIE DE RECUNOAȘTERE FACIALĂ

Acest proiect de diplomă are ca obiectiv principal realizarea unei aplicații integrate de recunoaștere facială și recunoaștere de vorbire folosind robotul KUKA youBot.

În *Capitolul 4* a fost prezentată etapa de creare a aplicaţiei de recunoaştere de comenzi, iar *Capitolul 5* a ilustrat modul în care această aplicaţie a fost portată şi adaptată pentru robotul KUKA, acesta devenind capabil să răspundă comenzilor vocale ale utilizatorului. În cadrul acestui capitol va fi descrisă partea de integrare a aplicaţiei anterioare cu o aplicaţie de recunoaştere facială deja existentă în laboratorul SpeeD. De asemenea, vor fi prezentaţi paşii urmaţi în vederea adăugării de noi funcţionalităţi: înrolarea noilor utilizatori pe baza rostirii numelor proprii şi integrarea de feedback vocal în aplicaţie.

Așadar, în aplicația finală, utilizatorul se va autentifica printr-o analiză de imagine, urmând apoi posibilitatea de transmitere de comenzi prin voce, în urma cărora robotul va genera un răspuns. În cazul în care persoana nu este recunoscută, acesteia i se va prezenta opțiunea de înrolare, ale cărei etape vor fi specificate de robot.

Modul de funcționare a aplicației este ilustrat în Figura 6.1.

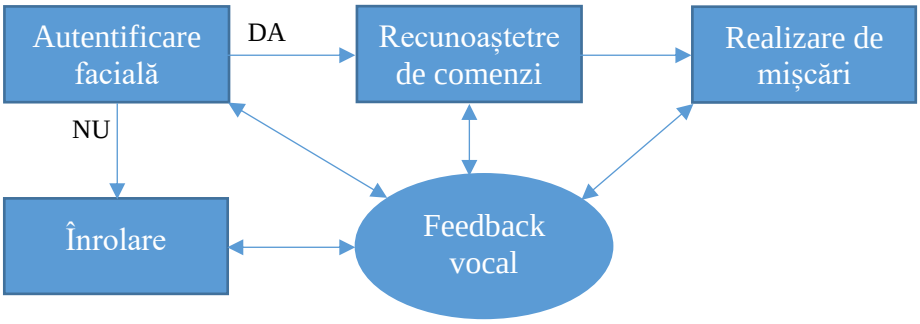

Figura 6.1 Etapele desfășurării aplicației

6.1 DESCRIEREA APLICAȚIEI DE RECUNOAȘTERE FACIALĂ

Aplicația inițială care se afla deja dezvoltată pe robot cuprindea un sistem de autentificare facială. Folosind camera ASUS, prezentată în *Subcapitolul 3.2.3*, robotul preia o imagine a utilizatorului pe baza căreia determina dacă acesta se afla sau nu înregistrat în baza de date.

Interacțunea cu robotul se face prin perifericele de intrare care se pot conecta la robot (mouse, tastatură) și de asemenea, prin perifericele de ieșire (monitor). Acest aspect constrânge platforma robotică să fie conectată în permanență la alte dispozitive, iar consecința este că raza sa de acțiune se micșorează foarte mult. S-a încercat o modalitate de conectare de la distanță, dar metoda aceasta conducea la o execuție încetinită a programului.

Sistemele Linux prezintă opțiunea de conectare la distanță de pe un dispozitiv pe altul folosind SSH (secure shell). Acesta este un protocol prin care se pot opera servicii de rețea într-un mod sigur pe o rețea nesigură [Network Working Group, 2006]. Cel mai frecvent mod de utilizare este conectarea la alte dispozitive, dar, în plus, prezintă și alte facilități. Una dintre acestea este dată de transmiterea la distanță a sistemului de ferestre grafice de pe dispozitivul sursă către cel destinație.

Transmițând ferestrele grafice s-a realizat independeța robotului de perifericele la care ar fi trebuit să fie conectat, dar dezavantajul a fost lentoarea de execuție care răpea aplicabilitatea practică a aplicației.

Organigrama proiectului inițial este prezentată în Figura 6.2.

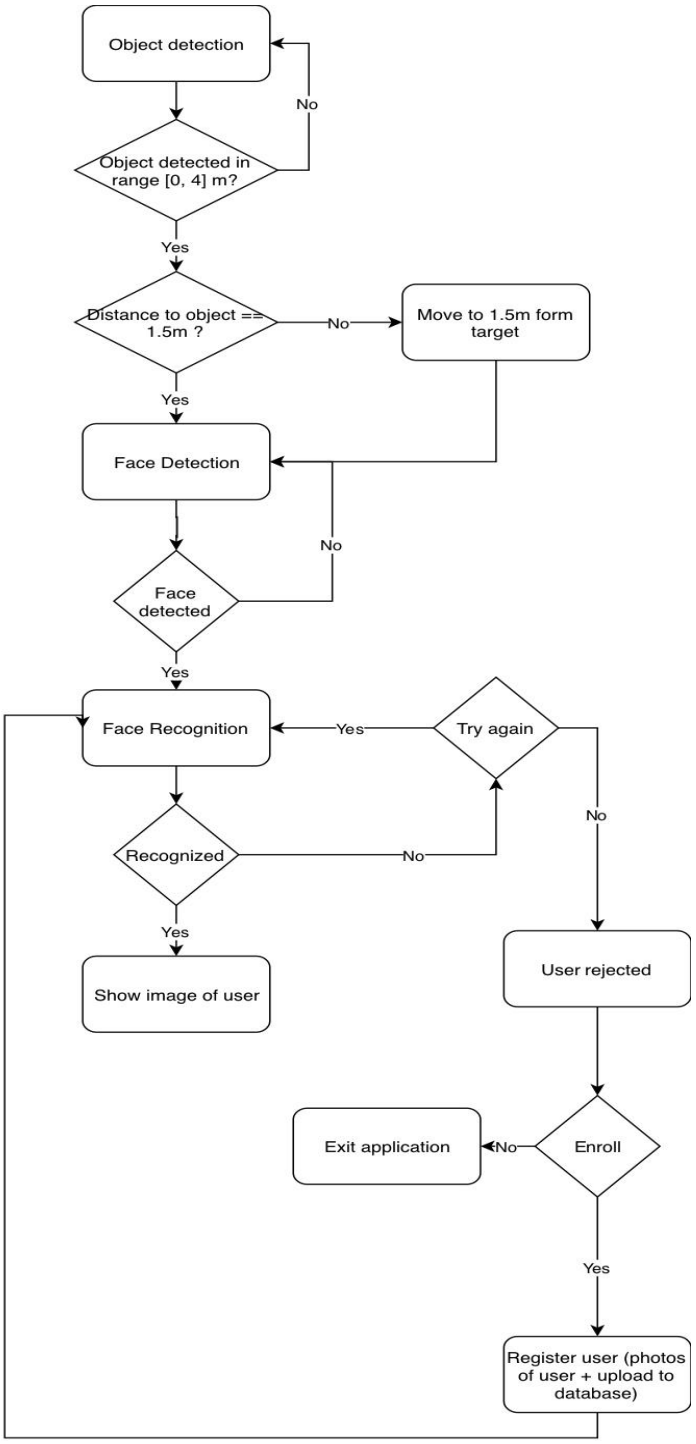

Figura 6.2 Organigrama aplicației inițiale

Etapele de execuție sunt următoarele:

- se verifică dacă în fața camerei se regăsește un obiect la o distanță mai mică de 4 m
- în caz negativ, procesul de detecție de obiecte se reia
- în caz afirmativ, robotul efectuează o serie de mișcări pentru a se poziționa la 1.5 m de obiect
- în continuare, se verifică dacă obiectul detectat anterior conține o față
- în caz negativ, robotul reia detecția de fețe
- în caz afirmativ, trece la etapa de recunoaștere facială
- dacă utilizatorul a fost recunoscut, atunci pe ecran apare o imagine cu acesta însoțită de un text cu numele cu care a fost înregistrat

- dacă utilizatorul nu a fost recunoscut, atunci acesta are două variante: fie alege opțiunea unei noi verificări, fie declară că este un utilizator nou
- în cazul celei de-a doua opțiuni, pe ecran apare o imagine în care este întrebat dacă dorește să fie înrolat
- în cazul unui răspuns negativ, aplicația se oprește
- în cazul unui răspuns afirmativ, robotul demarează rutina de înrolare. Această rutină presupune descrierea unui semicerc în jurul utilizatorului, iar, în acest timp, robotul preia cadre cu fața subiectului din diverse unghiuri pentru a crea un profil cu care să fie actualizată baza de date
- în final, pe ecran apare o nouă imagine în care utilizatorului îi este indicat să își introducă numele pentru a se finaliza înscrierea

În urma enumerării etapelor se poate constata că programul este strâns legat de interacțiunea fizică dintre robot și utilizator prin intermediul perifericelor. În continuare, s-a considerat că eliminând necesitatea unei interfețe fizice ar crește utilitatea aplicației inițiale. Acest obiectiv a fost realizat prin includerea aplicației de recunoaștere de comenzi rostite în limba română în proiectul deja existent.

6.2 SCHIMBAREA PARADIGMEI DE PROGRAMARE

Printr-o paradigmă se înțelege o clasificare a limbajelor de programare bazată pe caracteristicile lor. Limbajele se pot clasifica în mai multe paradigme, dintre acestea se poate aminti paradigma imperativă. Paradigma imperativă presupune că anumite porțiuni de cod pot modifica zone din program care nu sunt incluse în domeniul lor. Aceasta, la rândul ei se împarte în două ramuri: paradigma procedurală și paradigma obiect orientată [Nørmark, 2011].

O problemă importantă întâmpinată în fuzionarea celor două proiecte a fost abordarea filozofiilor diferite în care acestea au fost dezvoltate. În ciuda utilizării unei instanțe de clasă, proiectul de recunoaștere de comenzi are la bază programarea procedurală, în timp ce aplicația de recunoaștere facială este obiect orientată. Astfel, a fost nevoie de o regândire a arhitecturii aplicației rezultate pentru a putea îngloba cele două părți.

Soluția a fost renunțarea la paradigma procedurală în cazul recunoașterii de comenzi și trecerea la o abordare obiect orientată. Schimbările sunt următoarele:

- crearea clasei VoiceControl care cuprinde toate funcțiile menționate în *Subcapitolul 5.2.1* sub formă de comportamente
- înlocuirea funcției `main` cu o funcție de inițializare a instanței de VoiceControl: `init_voice_control(int)`. Această funcție primește un parametru care specifică tipul gramaticii cu care o să fie inițializată instanța. A fost menționat în *Subcapitolul 5.2.4* faptul că a fost creată atât o gramatică pentru comenzi cât și una simplă care conținea doar regula da sau nu. Distincția dintre acestea două în cazul inițializării se face prin parametrul funcției.
- introducerea instanței de VoiceControl în clasa principală din proiectul de recunoaștere de fețe și eliminarea operațiilor care necesitau o interfață fizică între operator și platforma robotică
- transmiterea transcrierilor rezultate după recunoașterea de vorbire în clasa principală principală sub formă de comenzi codate
- mutarea controlului robotului de pe aplicația de recunoaștere de comenzi în clasa principală a proiectului rezultat

6.3 GENERAREA DE FEEDBACK VOCAL

După rezolvarea problemei de compatibilitate între aplicații și unirea acestora, a apărut necesitatea înlocuirii perifericelor atașate robotului. Rolul dispozitivelor de intrare a fost preluat de transmiterea mesajelor prin voce. A mai rămas să se găsească o modalitate pentru a dispărea nevoia afișării de imagini sau de mesaje pe ecran.

A fost aleasă ca soluție implementarea unui feedback vocal. Pe baza unui sistem de sinteză de vorbire au fost generate clipuri audio care conțin mesaje sugestive. Acestea au fost plasate în interiorul aplicației astfel încât utilizatorul să știe în orice moment în ce etapă a execuției se află programul și, de asemenea, acesta este și îndrumat pentru a interacționa corect cu robotul.

Mesajele transmise de robot prin clipurile audio redate sunt următoarele:

- începerea inițializării
- începerea detecției de obiecte
- găsirea unui obiect
- începerea poziționării la o distanță potrivită față de obiect
- începerea detecției de fețe
- găsirea unei fețe și începerea recunoașterii feței detectate
- în cazul unei recunoașteri eșuate, utilizatorul este întrebat dacă dorește o nouă încercare de recunoaștere
- în cazul unui răspuns negativ, utilizatorul este întrebat dacă dorește să se înroleze
- în cazul unui răspuns afirmativ, utilizatorului îi este indicat să își rostească numele
- utilizatorului i se prezintă procedura de înscriere
- dacă utilizatorul a fost recunoscut, atunci acesta primește un mesaj de întâmpinare alături de înregistrarea în care și-a rostit numele
- la terminarea execuției unei comenzi, utilizatorului îi este indicat să rostească o nouă comandă

6.4 FUNCȚIONAREA APLICAȚIEI

Aplicația finală a fost dezvoltată ca o completare a aplicației inițiale de recunoaștere facială, aducând în plus, pe lânga controlul prin voce, și o optimizare din punctul de vedere al interacțiunii cu utilizatorul și din perspectiva libertății de mișcare a platformei, aceasta nemaifiind constrânsă de legăturile cu perifericele.

Organigrama aplicației finale este prezentată în Figura 6.3.

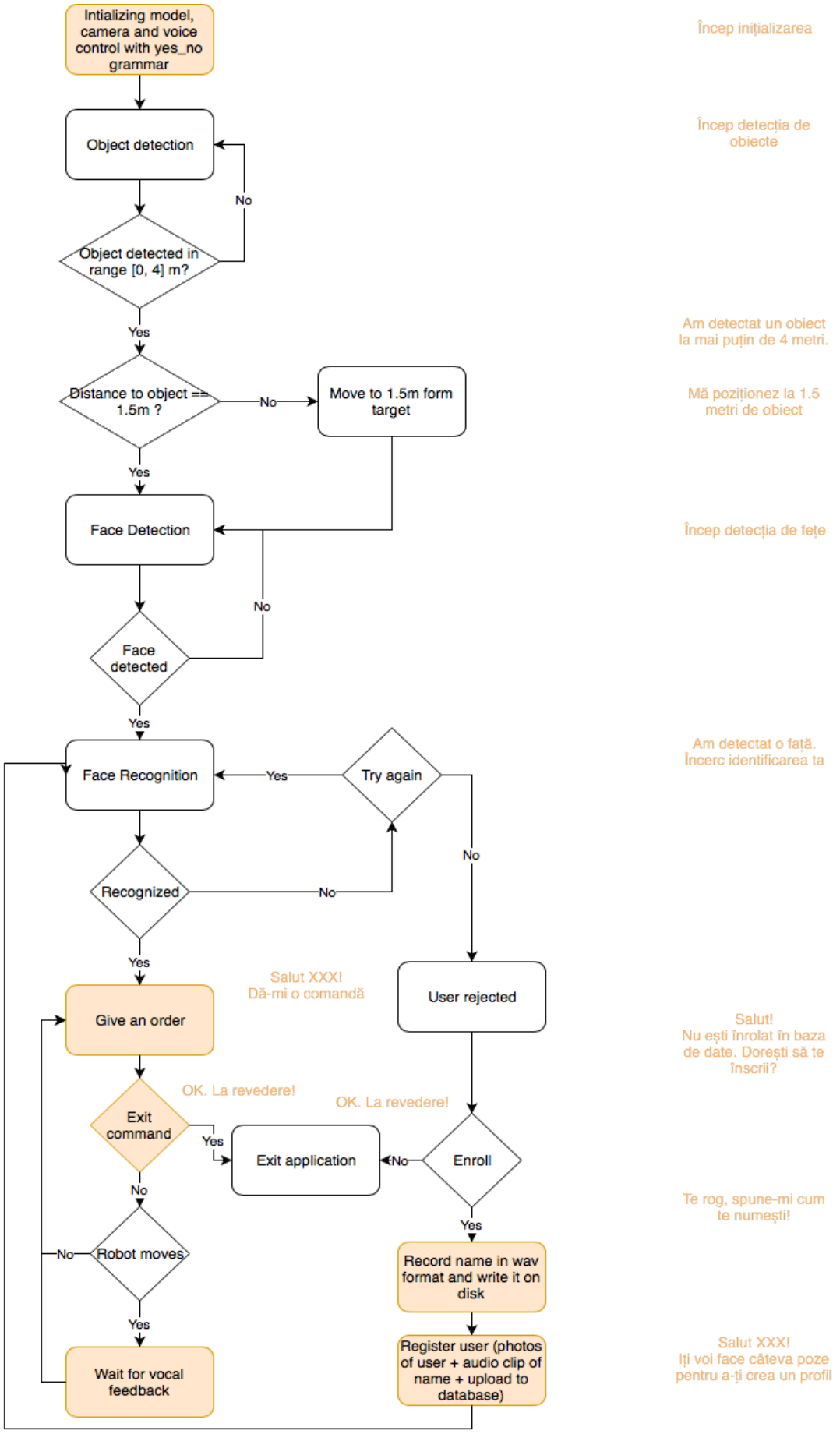

Figura 6.3 Organigrama aplicației finale

Analizând Figura 6.2 și Figura 6.3, se pot observa funcționalitățile suplimentare adăugate. În primul rând, înaintea etapei de detecție de obiecte, apare un pas de inițializare a programului în care se încarcă elementele necesare configurării sistemului de recunoaștere a vorbirii, precum modelul acustic, modelul de limbă sau dicționarul. De asemenea, tot în acest proces incipient, se face selectarea gramaticii de tip „da sau nu”. Această alegere are scopul de a împiedica un utilizator străin să transmită comenzi robotului și elimină vocabularul în plus pentru sarcina de interacțiune cu acesta (cazurile în care trebuie luate decizii).

Se remarcă blocul „Give an order” care simbolizează etapa de control prin voce, etapă ce survine în cazul în care utilizatorul este recunoscut. În plus, în cadrul acestei etape, se face o reinițializare a configurației sistemului de recunoaștere vocală, care va conține acum gramatica corespunzătoare comenzilor. În cazul aplicației inițiale, în acest moment programul se încheia cu transmiterea unei imagini.

Mai apar în plus blocurile de înregistrare a unui clip audio cu numele noului utilizator și de adăugare a clipului într-o bază de date. Pentru a se elimina necesitatea introducerii numelui de la tastatură, s-a optat pentru rostirea acestuia la începutul etapei de înrolare. Fișierul rezultat va fi denumit automat cu data curentă în milisecunde și salvat într-un anumit director, alături de clipurile audio ale celorlalte persoane înrolate. Astfel, în procesul de denumire nu este nevoie ca utilizatorul să intervină la nivel de cod.

O altă schimbare o constituie prezența a două posibilități de terminare a aplicației. Prima posibilitate se activează în momentul în care înscrierea este refuzată, iar cea de-a doua apare când cel care folosește sistemul își manifestă explicit dorința de a îl opri printr-o comandă.

Așa cum a fost subliniat în *Subcapitolul 6.3*, se remarcă folosirea de mesaje audio pentru îndrumarea utilizatorului sau pentru a îi semnala că este nevoie să ia o anumită decizie.

CAPITOLUL 7

CONCLUZII

7.1 CONCLUZII GENERALE

Scopul acestei lucrări a fost de a crea un asistent robotic capabil să execute sarcini simple și repetitive care consumă mult timp. Beneficiile care ar putea fi aduse sunt reprezentate de un management mai bun al perioadei de lucru și o interacțiune simplificată cu robotul.

Asistenții virtuali, precum Siri sau Cortana, sunt nelipsiți din viața multor persoane, dar raza lor de acțiune este limitată la mediul software al dispozitivelor pe care sunt instalați. Dacă folosirea unui astfel de asistent, care poate crea un eveniment în calendar sau poate accesa o anumită aplicație este considerată utilă, atunci o platformă robotică ce răspunde la comenzi și este suficient de îndemânatică pentru a se ocupa de sarcini obișnuite devine un instrument absolut necesar în lupta de combatere a timplilor morți.

Lucrarea prezentată ilustrează un exemplu de folosire a unui astfel de asistent și reprezintă un punct de plecare către dezvoltarea mai multor aplicații inteligente pentru aceast robot.

A fost demonstrat că două programe independente, precum recunoașterea de vorbire și recunoașterea facială, pot fi fuzionate pentru a crea o aplicație mai cuprinzătoare. Această fuzionare nu se limitează doar la programele amintite ci poate fi extinsă la oricâte altele, având toate ca element de legătură recunoașterea de comenzi vocale. În acest mod, abilitățile robotului pot deveni mult mai numeroase, iar sarcinile de care se va putea ocupa vor fi din ce în ce mai complexe.

7.2 CONTTRIBUȚII PERSONALE

O parte din contribuțiile personale în această lucrare sunt sumarizate grafic în Figura 7.1, iar o rezumare mai succintă a acestora este următoarea:

- dezvoltarea unui sistem de recunoaștere a cifrelor rostite în limba română:
  - achiziția unei baze de date de vorbire pentru dezvoltarea sistemului de recunoaștere de cifre dependent de vorbitor
  - crearea unui sistem de recunoaștere de cifre dependent de vorbitor
  - crearea unui sistem de recunoaștere de cifre independent de vorbitor
  - realizarea de experimente și statistici pe sistemele menționate pentru a stabili configurația optimă pentru acestea
- crearea unei aplicații de recunoaștere de comenzi prin voce ca succesiuni de cifre pe robotul KUKA youBot
- includerea unui model acustic general în aplicație
- cercetarea modului de funcționare a aplicației de recunoaștere facială
- includerea aplicației de comenzi vocale în aplicația de recunoaștere facială
- construirea fișierelor audio necesare pentru interacțiunea robotului cu utilizatorul
- adăugarea de feedback vocal

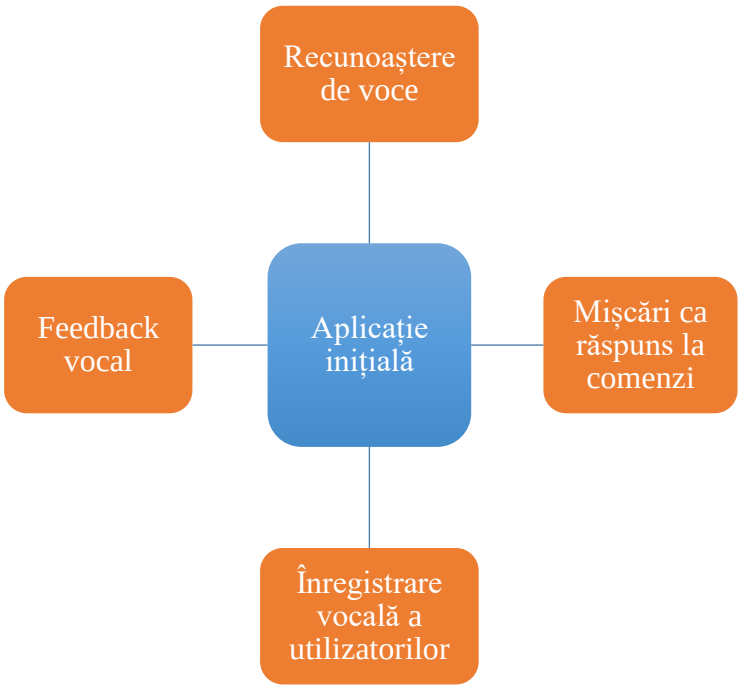

Figura 7.1 Contribuții personale

7.3 DEZVOLTĂRI ULTERIOARE

Proiectul descris în această lucrare are un rol demonstrativ. Prin intermediul său se poate observa potențialul ridicat al folosirii unui asistent robotic, dar utilitatea sa practică este, momentan, restrânsă.

Ca dezvoltări ulterioare ar putea fi incluse noi aplicații inteligente care să facă robotul să fie capabil de a se ocupa de sarcini din rutina zilnică a unui om. Spre exemplu, se poate realiza o aplicație de analiză de imagine care să localizeze poziția unui anumit obiect într-o camera, iar, pe baza acestei locații, să fie descrisă o rutină prin care robotul să aducă în mod automat acel obiect. Pentru partea de comandă vocală intervenția ar fi minimă deoarece completarea comenzilor se poate face foarte simplu pe baza celor deja incluse.

În testarea aplicației a fost observată și o breșă de securitate, și anume, faptul că, odată ce un utilizator a fost recunoscut, oricine poate comanda mai departe robotul deoarece modelul acustic nu distinge vorbitorul ci doar mesajul. Așadar, includerea unei facilități de autentificare prin voce, urmată de o diarizare poate aduce un aport substanțial funcționării aplicației.

BIBLIOGRAFIE

[Acero, 2015] Acero A., "Speech Recognition and Understanding", Microsoft Research, pp. 57-58, 2015

[Huggenberger, 2011] Huggenberger U., Bischoff R., Prassler E., "KUKA youBot - a mobile manipulator for research and education", în IEEE International Conference on Robotics and Automation, 2011

[Bischoff, 2011] Bischoff R., "KUKA youBot - a milestone for education and research in mobile manipulation", în IEEE ICRA Workshop "A new generation of educational robots", 2011

[CMU Sphinx, 2018] "CMUSphinx Documentation", https://cmusphinx.github.io/wiki/, accesat la data 03.06.2018

[Cucu, 2013] "Proiect de cercetare-dezvoltare în tehnologia vorbirii", http://speed.pub.ro/speed3/wp-content/uploads/2014/02/Indrumar-de-proiect-PCDTV-v11.pdf, accesat la data 10.06.2018

[Fonollosa, 2017] Fonollosa J. A. R., "End-To-End Speech Recognition with Recurrent Neural Networks" Universitatea Politècnica de Catalunya Barcelona, cursul Deep Learning for Speech and Language, 2017

[Generation Robots, 2018] https://www.generationrobots.com/en/402185-kuka-youbot-mobile-platform.html, accesat la data de 05.06.2018

[Hwang, 1992] Huang M. Y., Huang X., "Subphonetic modeling with Markov states-Senone", în IEEE International Conference on Acoustics, Speech, and Signal Processing, 1992

[IBM, 2001] "Pioneering Speech Recognition", http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/speechreco/, accesat la data 02.06.2018

[Juang, 2004] Juang, B. H.; Rabiner, Lawrence R. "Automatic speech recognition–a brief history of the technology development", pp. 5-6, 2004

[Keiser, 2013] Keiser B., "Torque Control of a KUKA youBot Arm", University of Zurich, lucrarea de disertație, 2013

[KUKA, 2018] About us, https://www.kuka.com/en-us/about-kuka, accesat la data 30.05.2018

[LUHbots, 2018] https://github.com/LUHbots/luh_youbot_os, accesat la data 05.06.2018

[Manual, 2012] KUKA youBot User Manual, 2012

[McGlaun, 2011] McGlaun S., https://www.slashgear.com/asus-gets-xtion-pro-live-ready-for-launch-19165977/, accesat la 04.06.2018

[Network Working Group, 2006] Network Working Group of the IETF, The Secure Shell (SSH) Protocol Architecture, 2006

[Nørmark, 2011] Nørmark K., "Overview of the four main programming paradigms", Aalborg University, 2011

[Qian, 2017] Qian J., Zi B., Wang D., Ma Y. Zhang D., "The Design and Development of an Omni-Directional Mobile Robot Oriented to an Intelligent Manufacturing System", Sensors, 2017

[Robocup, 2012] http://www.robocupatwork.org/download/RoboCup-At-Work_Camp_2012/RAW_Camp2012-Jan-Paulus_youBotAPI.pdf, accesat la data 05.06.2018

[Sahidullah, 2012] Sahidullah Md., Saha G. "Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition", în Speech Communication, pp. 545, 2012

[Schütz, 2013] Schütz A., "Small robot is a big number in open source", https://www.maxonmotor.in/medias/sys_master/root/8815857795102/2014-10-story-enUK-kuka-research-robot.pdf?attachment=true, accesat la data 03.06.2018

[youBot-store, 2018] "Hardware Specification", http://www.youbot-store.com/developers/hardware-specification, accesat la 03.06.2018