Universitatea “Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Sistem automat de scriere a unui text folsind un braț robotic

Proiect de diplomă

Prezentat ca cerință parțială pentru obținerea titlului de
Inginer în domeniul *Electronică și Telecomunicații*
programul de studii de licență *Electronică Aplicată*

Conducatori științifici
Prof. Dr. Ing. Corneliu BURILEANU
Ing. Ana NEACȘU

Absolvent
Nicolae-Vlad MANOLE

2019

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

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **MANOLE I. Nicolae-Vlad , 444B**

1. Titlul temei: Sistem automat de scriere a unui text folosind un braţ robotic

2. Descrierea contribuţiei originale a studentului (în afara părţii de documentare) și specificații de proiectare:
Proiectul are ca scop dezvoltarea unui sistem ce permite scrierea, cu litere de tipar, a unui text pe o tăbliţă magneitică folosind braţul robotic Jaco dezvoltat de Kinova Robotics. Sistemul va face o analiză pe un text introdus de ia tastatură. Sistemul va fi compus din trei module, după cum urmează.
Modulul de achiziţie se va ocupa de captarea și procesarea imaginilor în vederea stabilirii poziţiei tăbliţei de scris şi implicit locul unde braţul robotic va scrie.
Modulul de analiză a textului, va stabili numărul de caractere, dimensiunea textului scris şi mişcările care vor fi efectuate de braţul robotic pentru scrierea fiecărei litere în parte.
Un al treilea modul va prelua datele de la celelate două module şi va trimite comenzi braţului pentru a scrie o literă într-o anumită locaţie, urmând ca braţul să efectueze mișcarea propriu-zisă

3. Resurse folosite la dezvoltarea proiectului:
Kinova Jaco Arm 2

4. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline:
Programarea Calculatoarelor, Pogramarea Orientată pe Obiecte, Microcontrolere

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

6. Data înregistrării temei: 2018-11-28 12:54:19

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

NEACȘU Ana-Antonia
semnătura: .....

Student,
semnătura:.....

Director departament,
Prof. dr. ing Sever PAȘCA
semnătura:.....

Decan,
Prof. dr. ing. Cristian NEGRESCU
semnătura:.....

Cod Validare: **2c80fe1c2a**

Declaraţie de onestitate academică

Prin prezenţa declar că lucrarea cu titlul "*Sistem automat de sciere a unui text folosind unui braţ robotic*", prezentată în cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei a Universităţii "Politehnica" din Bucureşti că cerinţă parţială pentru obţinerea titlului de *Inginer* în domeniul *Electronică şi Telecomunicaţii*, programul de studii *Electronică Aplicată* este scrisă de mine şi nu a mai fost prezentată niciodată la o facultate sau instituţie de învăţământ superior din ţară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, că referinţe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi fac referinţă la sursă. Reformularea în cuvinte proprii a textelor scrise de către alţi autori face referinţă la sursă. Înţeleg că plagiatul constituie infracţiune şi se sancţionează conform legilor în vigoare.

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

Bucureşti, *Iulie 2019*

Absolvent *Nicolae-Vlad MANOLE*

Cuprins

Lista tabele

Lista de acronime

SDK - Software Development Kit

DC - Direct Current

ZIF - Zero Insertion Force

USB - Universal Serial Bus

CPU - Central Processing Unit

API -Application Programming Interface

ROS - Robot Operating System

NUI - Natural User Interface

JSON - JavaScript Object Notation

REST - Representational State Transfer

Lista de figuri

Introducere

Motivaţie

Modul în care oamenii percep și interacționează cu mediul înconjurător a fost un factor major care a influenţat modificarea și construirea a tot ceea ce este în jurul nostru. Bineînţeles, invenţiile omenirii sunt raportate la interacţiunea dintre un om perfect sănătos cu acestea.

Există însă o categorie de oameni care se pierde în acest mediu antropic, gândit pentru o interacţiune uşoară cu omul. Această cateogorie este reprezentată de persoanele cu dizabilităţi, persoane pentru care şi cele mai de bază interacţiuni pe care omul le are cu obiectele din jurul lui, cum ar fi deschidearea unei uşi, reprezintă o provocare.

Pentru integrarea acestor persoane în societate, şi pentru a le oferi o metodă prin care aceştia să-şi îmbunătăţească calitatea vieţii au fost dezvoltate diferite sisteme robotice gândite să ofere libertate şi independenţa persoanelor cu dizabilităţi. Un astfel de sistem este braţul robotic Jaco, gândit să vină în ajutorul persoanelor ce suferă de dizabilităţi locomotorii în partea superioară a corpului, sistem care oferă utilizatorilor posibilitatea să interacţioneze cu mediul înconjurător într-un mod similar cu cel al unui om sănătos [1].

Majoritatea acestor sisteme au în prezent nevoie ca factorul uman să intervină în procesul de control al acestora. Soluţia descrisă mai sus necesită intervenţia operatorului pentru a controla braţul cu ajutorul unui joystick, însă necesitatea intervenţiei factorului uman îngreunează interacţiunea cu mediul înconjurător. Astfel nevoia de sisteme complet autonome, care nu necesită intervenţia umană, îşi face simţită prezenţa.

Obiectivul principal

Scopul acestei lucrări a fost dezvoltarea unui sistem complet automat ce este capabil să efectueze o activitate specifică oamenilor, anume scrisul unui text pe un suport de scris, pentru a evidenţia aplicaţiile roboticii în domeniul medical, în special în domeniul asistenţei şi reabilitării persoanelor cu afecţiuni locomotorii.

Pentru ca acest sistem să fie complet autonom este nevoie de dezvoltarea a 3 module ce îndeplinesc funcţii asemănătoare cu cele ale omului, şi prin a căror comunicare este realizat procesul de scriere. Primul modul este reprezentat de cel de achiziţie de imagini şi de prelucrare a acestora, modulul care reprezintă modul în care sistemul autonom percepe mediul înconjurător. Acesta are rolul de a găsi în spaţiul înconjurător, prin capturi de imagini, locaţia suportului de scris precum şi orientarea acestuia faţă de braţul robotic. Cel de-al doilea modul decide care vor fi mişcările efectuate de braţ pentru a putea scrie textul, iar cel de-al treilea modul interpretează aceste date, şi controlează efectiv braţul robotic pentru ca acesta să efectueze mişcările comandate de modulul anterior.

Obiective specifice

- Integrarea unei camere color în sistem pentru perularea imaginilor din mediul înconjurător
- Dezvoltarea unui modul capabil să recunoască formele şi poziţia relativă a acestora la baza braţului robotic
- Dezvoltarea unui dicţionar ce conţine mişcările necesare scriereii fiecărei litere din alfabetul englez
- Dezvoltarea unui sistem de dimensionare a textului şi de poziţionare a acestuia în pagină
- Integrarea braţului robotic în proiect şi implementarea unui modul ce transpune comenzile primite în mișcări efectuate de braţul robotic

1. Tehnologiile folosite în proiect

1.1Brațul robotic Kinova Jaco2

1.1.1 Informaţii generale

Kinova Jaco2 este un braţ robotic ultra uşor, ce a fost dezvoltat pentru a compensa lipsa mobilităţii unui membru superior [1]. Este compus din 6 segmente interconectate,având astfel 6 grade de libertate, ultimul segment fiind un manipulator cu 3 degete. Prin această configuraţie orice persoană cu dizabilităţi are posibilitatea de a efectua acţiuni complexe braţul putând fi controlat în spaţiul tridimensional şi oferind posibilitatea de a apuca şi de a mişca obiecte în mediul înconjurător.

Caracteristicile braţului robotic sunt următoarele :

- Integrare uşoră, fiind uşor de montat pe orice scaun cu rotile sau suprafaţă plană
- Posibilităţi multiple de control
- Consum redus de energie electrică
- 6 grade de libertate
- Posibiliteatea efectuării a 16 mişcări pentru a imita mişcarea braţului uman
- Greutate scăzută datorată structurii din fibră de carbon
- Degete flexibile adaptabile la forma obiectului apucat
- Posibiltiatea utilizării a 2 sau 3 degete
- Optimizat pentru folosirea în activităţi zilnice
- Rezistent la apă (IPX2) şi la temperaturi scăzute

Producătorul pune la dispoziţia dezvoltatorilor un chit de dezvoltare prin care braţul poate fi controlat. SDK-ul este compatibil cu limbajul de programare C++ şi conţine o bibliotecă cu funcţii implementate pentru controlul robotului atât cartezian, fiindu-i setată poziţia în spaţiu, cât şi unghiular, fiindu-i controlat fiecare actuator în parte. [2]

Chitul de dezvoltare conţine şi un controler virtual, ce poate înlocui complet toate funcţiile joystick-ului fizic, şi prin intermediul căruia putem obţine informaţii suplimentare despre braţ, cum ar fi poziţia efectorului raportată la baza braţului robotic sau valorile unghiurilor actuatoarelor.

1.1.2 Specificaţiile braţului robotic

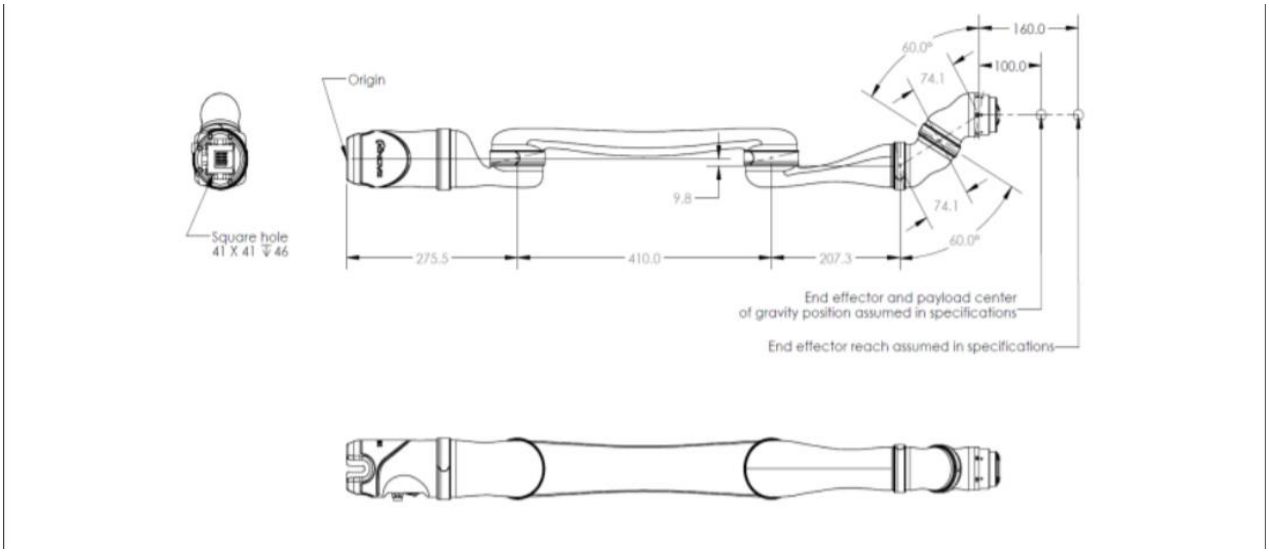

Fig. 1.1 Specificaţiile braţului robotic [3]

Greutate	**5.2Kg**
Materiale segmente	Fibră de carbon
Material actuator	Aluminiu
Sarcina	1.6 Kg(pentru o alungire medie a braţului) 1.3 Kg (pentru o alungire maximă a braţului)
Alungirea maximă	984mm
Viteza (liniară) maximă a braţului	20 cm/s
Tensiunea de alimentare	18 - 29 VDC
Putere medie	25W
Putere maximă	100W
Putere standby	5W
Protocol de comunicaţie	RS485
Cabluri de comunicaţie	Cablu tip panglică cu 20 de pini

Tabel 1 Specificaţiile braţului robotic [4]

Jaco oferă utilizatorilor posibilitata să interacţioneze în mod liber cu mediul inconujrator, în deplină siguranţă şi cu o eficienţă crescută. Braţul poate efectua mişcări continue şi fluente prin rotirea multiplă a fiecărui actuator în parte.

Actuatoarele folosesc motoare de curent continu fără perii şi pot fi comandată atât prin controlul cuplului cât şi prin controlul poziţiei. Sunt independente, interoperabile, fiecare braţ robotic conţinând două seturi a câte trei actuatoare interschimbabile, conectate între ele de un cablu ZIF [4].

Structura completă din fibră de carbon oferă robusteţe şi durabilitate braţului robotic. Braţul este montat pe un suport standard din aluminiu şi strutură poate fi fixată pe aproape orice suprafaţă plană. [1]

1.1.3 Modalităţile de control

Modul de control	Descriere
Controlul Cartezian	Sunt comandate poziţia şi orientarea efectorului în coordonate carteziene, ce au origină la baza robotului
Viteza Carteziană	Este comandată viteză de translaţie a efectorului în sistemul de coordonate bazale şi viteza de rotaţie a efectorului în sistemul de coordonate ale efectorului
Controlul Unghiular	Este comandat fiecare unghi al fiecărui actuator în parte
Viteza Unghiulară	Este comandată viteza de rotaţie a fiecărui actuator în parte
Controlul direct al cuplului	Este comandat cuplul fiecărui actuator în parte. Greutatea braţului este compensată în mod implicit, fiecare actuator fiind comandat conform cuplului gravitaţional propriu.

Tabel 2 Modurile de control ale braţului robotic [5]

Pentru implementarea acestui proiect am ales să folosesc controlul cartezian, datorită uşurinţei de implementare, braţul primind ca valori de comandă 6 valori ce reprezintă poziţia şi orientarea efectorului. Aceste valori trimise braţului sunt uşor de interpretat de oameni, şi nu necesită o prelucrare adiţională pentru a înţelege şi a anticipa poziţia finală a braţului. Un alt motiv pentru care am ales acest mod de control, este datorită asemănării dintre mişcările efectuate de braţ şi mişcările unui braţ uman.

Braţul robotic poate fi controlat fie de un calculator, cu ajutorul unui software oferit de producător, fie prin intermediul unui joystick pe 3 axe cu butoane. Controlul este intuitiv, şi permite utilizatorului să comute între 3 moduri diferite: translaţie, rotaţie şi apucare cu efectorul.

Joystick	**1 mbps CANBUS**
Tensiune de alimentare	18-29 V
Ethernet	Nu este disponibil
USB 2.0	12 MBps
Frecvenţa sistemului de control	100hz pentru un API de nivel înalt 500 Hz pentru un API de nivel jos
CPU	360 Mhz
SDK	
API-uri	Nivem înalt şi jos
Compatibilitate	Windows, Linux, Ubuntu & ROŞ
Port	USB 2.0
Limbaj de programare	C++

Tabel 3 Specificaţiile controlerului [5]

De asemenea sistmul numit Kinova's Inteligent Avoidance System permite evitarea coliziunilor dintre efector şi alte segmente componente ale braţului, precum şi efectuarea unor mişcări care ar duce la deteriorarea structurii braţului.

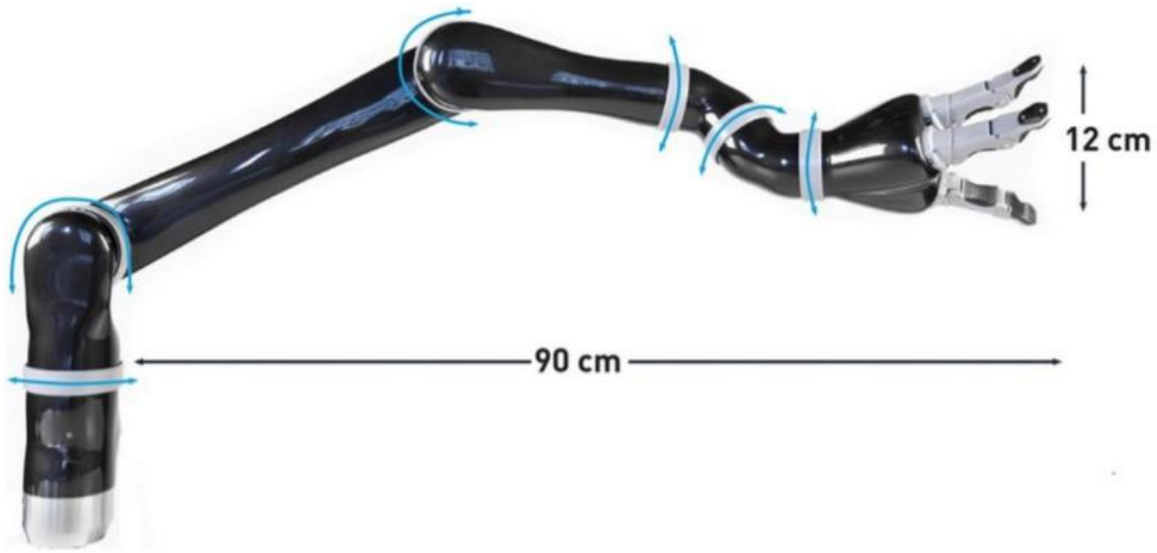

Fig. 1.2 Controlul unghiular [5]

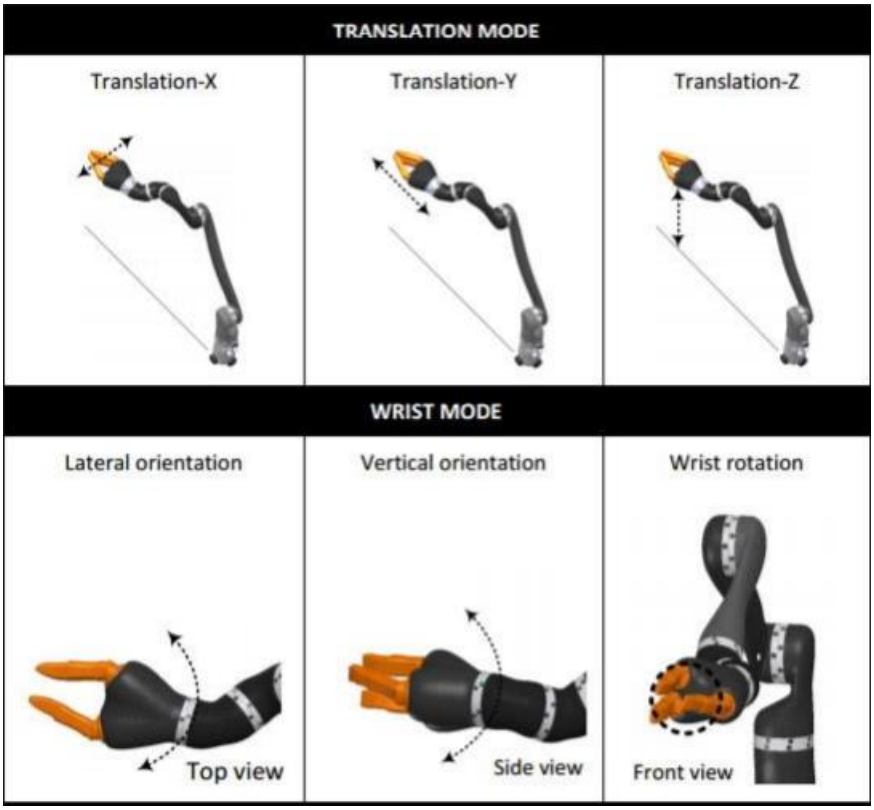

Fig. 1.3 Controlul cartezian [5]

1.1.4 Efectorul

Numărul de degete	**3**
Număr actuator	Unul pentru fiecare deget
Senzori din actuatoare	Curent Temperatură Encoder rotațional de poziţie
Deschidearea maximă	175mm
Diametrul minim al unui obiect cilindirc ce poate fi apucat	45mm
Diametrul maxim al unui obiect cilindirc ce poate fi apucat	100mm
Greutatea totală	726g
Forţa de strângere cu 3 degete	40N
Forţa de strângere cu 2 degete	25N
Timpul maxim de deschidere sau închidere totală a degetelor	1.2sec
Temperatura de operare	Între -10 ºC şi 40 ºC

Tabel 4 Specificaţiile efectorului [5]

Efectorul are 2 sau 3 degete subactuate ce pot fi controlate individual. [2] Prin complianţă pasivă, degetele, datorită structurii lor din plastic, se pot adapta cu uşurinţă la forma

obiectului apucat, braţul având astfel o flexibilitate foarte bună şi o gamă largă de obiecte ce pot fi manipulate.

1.1.5 Joystick-ul

Controlerul standard al braţului este un joystick pe 3 axe montat pe un suport ce include 7 butoane independete şi 4 intrări auxiliare (pe partea din spate). [5]

Joystick-ul permite utilizatorul să controleze braţul în două moduri diferite, controlul cartezian şi cel unghiular. În controlul cartezian braţul poate fi controlat folosind două sau 3 axe. [3]

1.2 Modulul Kinect

1.2.1 Informaţii generale

Modulul Kinect reprezintă un dispozitiv de detecţie a mişcării ce a fost iniţial dezvoltat pentru consola de jocuri video Xbox 360. Ceea ce face ca acest dispozitiv să iasă în evidenţă faţă de altele similare, este faptul că nu e doar un simplu modul de control cu ajutorul gesturilor, ci un întreg sistem ce este capabil să detecteze poziţia corpului uman, mişcările acestuia precum şi vocea. Kinect oferă un NUI (Interfaţă naturală) pentru interacţiunea utilizatorului cu sistemul prin mişcările corpului, gesturi sau prin comenzi vocale.

Acest modul a produs o revoluţie în lumea jocurilor video, şi a schimbat complet modul de precepție e consolelor. Încă de la lansare, în anul 2013 a doborât câteva recorduri în domeniul hardware-ului pentru jocuri video. Astăzi dispozitivul Kinect nu mai este limitat doar la industra divertismentului. [6]

Kinect for Windows este un modul dezvoltat special pentru calculatoare şi permite dezvoltatorilor de software să creeze aplicaţii în lumea reală bazate pe gesturi şi mişcarea corpului. Modulul oferit de Microsoft poate fi programat în C# şi poate interacţiona cu diverse dispozitive utilizatand SDK-ul oferit de producător. Totuşi o mulţime de dezvoltatori independeţi au dezvoltat module software speciale pentru a permite utilizarea dispozitiviului şi în alte sisteme decât cele scrise în limbajul C#.

Un astfel de sistem, este chiar sistemul dezvoltat de mine, sistem pentru care am folosit PyKinect, software open source, ce permite dezvoltarea aplicaţiilor şi a jocurilor cu Kinect în Python.

Kinect este un dispozitiv orizontal, ce conţine senzori de adâncime, o cameră color, şi un set de microfoane, toate încapsulate într-o carcasă de plastic robustă. Carcasa de plastic este ataşată pe un suport motorizat ce permite camerei rotirea pe axa orizontală. [2] Senzorul Kinect conţine următoarele componente cheie:

- Camera color
- Emitor de infraroşu
- Senzor infraroşu de adâncime
- Suport motorizat
- Set de microfoane

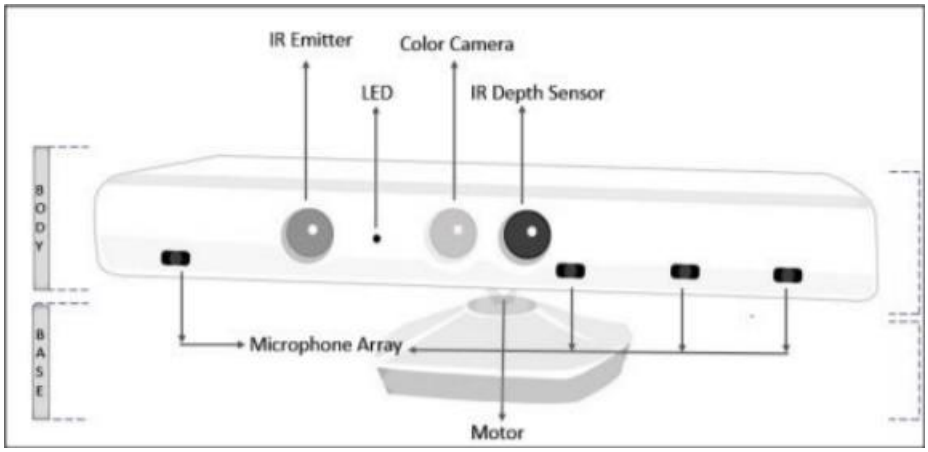

Fig. 1.4 Modulul Kinect [6]

1.2.2 Camera color

Camera color a modulului este responsabilă de a face capturi video şi de a transmite datele mai departe în sistem. Funcţia sa este de a detecta colorile roşu, verde şi albastru din mediul înconjurător. Fluxul de date returnat de cameră este reprezentat de o succesiune de imagini statice. Fluxul video are o viteză de 30 de cadre pe secundă la o rezoluţie de 640x840 de pixeli. Numărul de cadre pe secundă depinde de rezoluţia imaginii, rezoluţia maximă fiind de 1280x960 la un număr maxim de 12 cadre pe secundă.

Senzorul Kinect are vedere de 57 de grade pe orizontală şi 43 de grade pe verticală cum este arătat şi în figură 1.5.

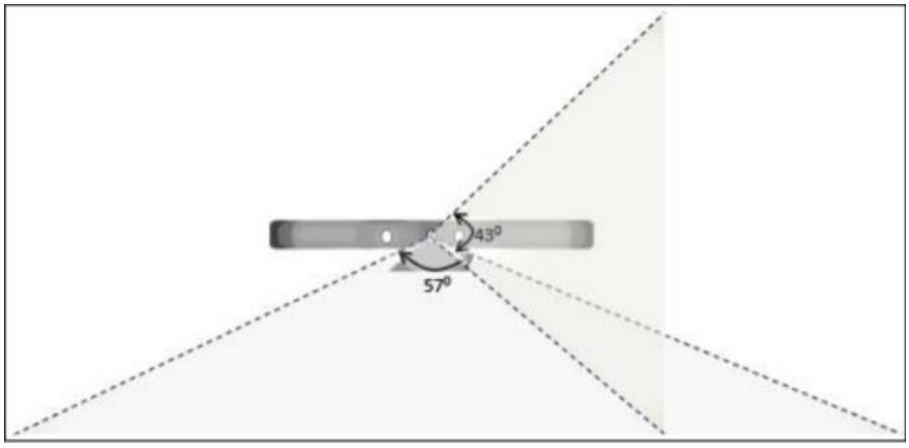

Fig. 1.5 Senzorul color [6]

1.2.3 Emitorul IR şi senzorul IR de adâncime.

Senzorul de adâncime al camerei Kinect este construit din două dispozitive diferite: un emitor de lumină infraroşie şi un senzor infraroşu de adâncime.

Emitorul de infraroşu poate arăta ca o cameră, însă este un proiector ce constant emite puncte de lumină infraroşie în mod „pseudo-aleator" pe orice se află în faţa camerei. Punctele

sunt invizibile pentru ochiul uman, însă permit aflarea adâncimii lor în mediul înconjurător cu ajutorul unui senzor de adâncime. Punctele de lumină infraroşie sunt reflectate de obiectele incidente şi senzorul de adâncime citeşte aceste informaţii şi le transformă în informaţii despre adâncime, prin măsurarea distanţei de la cameră până la locul din care raza a fost reflectată. [6]

Prin combinarea imaginilor color cu cele despre adâncime se poate obţine o imagine 3D a mediului înconjurător, imagine foarte utilă în aplicaţii ce implică recunoaşterea mişcărilor. [2]

Fluxul datelor de adâncime are o rezoluție maximă de 640x480 pixeli şi una minimă de 80x60 de pixeli, însă spectrul vizual al camerei de adâncime este identic cu cel al camerei color. Cu alte cuvinte, putem obţine informaţii de adâncime despre orice obiect din spectrul vizual al camerei color, însă nu putem afla informaţii de adâncime pentru fiecare pixel din imaginea color. [6]

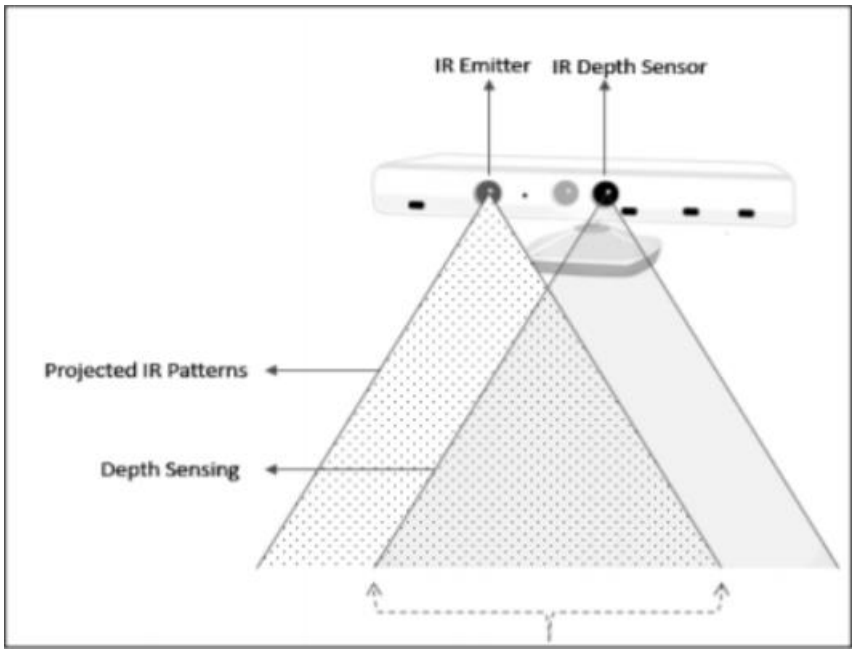

Fig. 1.6 Senzorul IR [6]

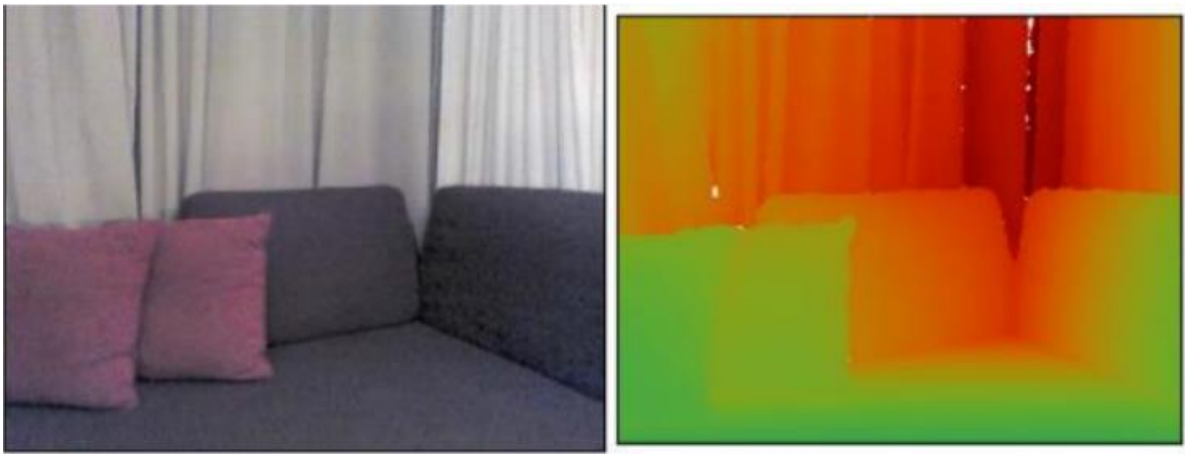

Fig. 1.7 Imagine color şi de adâncime [6]

1.2.4 Setul de microfoane

Scopul microfoanelor nu este doar de a permite modulului Kinect să înregistreze sunetul, ci şi pentru a localiza direcţia undelor audio. [5] Avantajele integrării mai multor microfoane sunt reprezentate de: recunoaşterea vocală mai facilă, reducerea zgomotului de fundal şi reducerea efectului de ecou. Astfel prin integrarea multiplelor microfoane, Kinect este un dispozitiv care poate recunoaşte locaţia sursei de sunet, a vocii, precum şi eliminarea zgomotului şi a ecoului din încăpere.

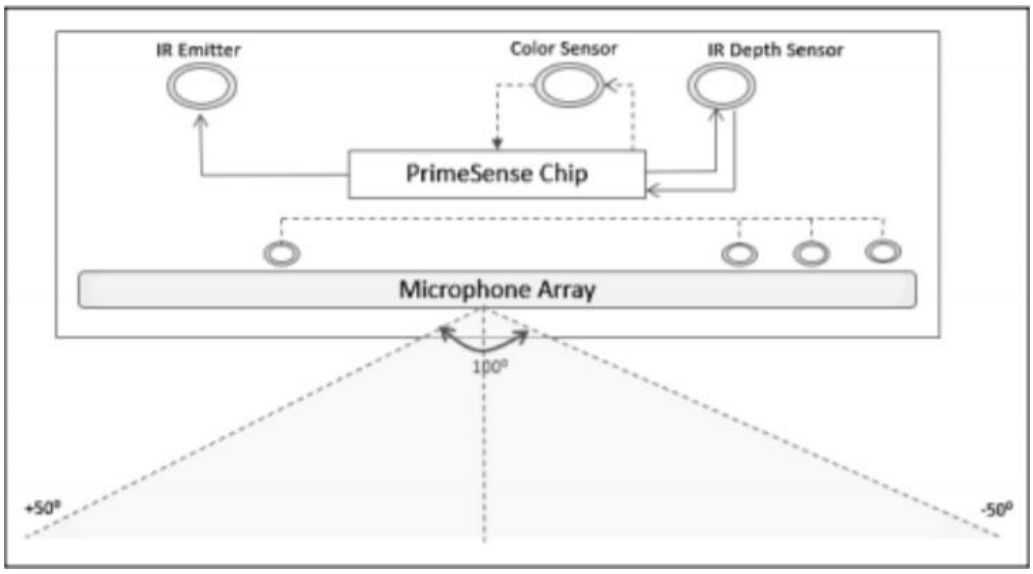

Fig. 1.8 Setul de microfoane [6]

1.2.5 Suportul motorizat

Suportul de la baza modulului şi carcasa ce încorporează senzorii sunt interconectaţi de un motor rotativ. Acesta este folosit pentru a modifica, pe axa verticală, orientarea camerei (şi implicit a tuturor senzorilor), modificare care este facilă în a estima poziţia corectă a corpului uman în încăpere. Motorul poate roti cameră la un unghi de până la 27 de grade, atât în sus cât şi în jos, şi practic unghiurile câmpului de percepţie al senzorilor pot fi modificate cu 27 de grade atât în sus cât şi în jos.

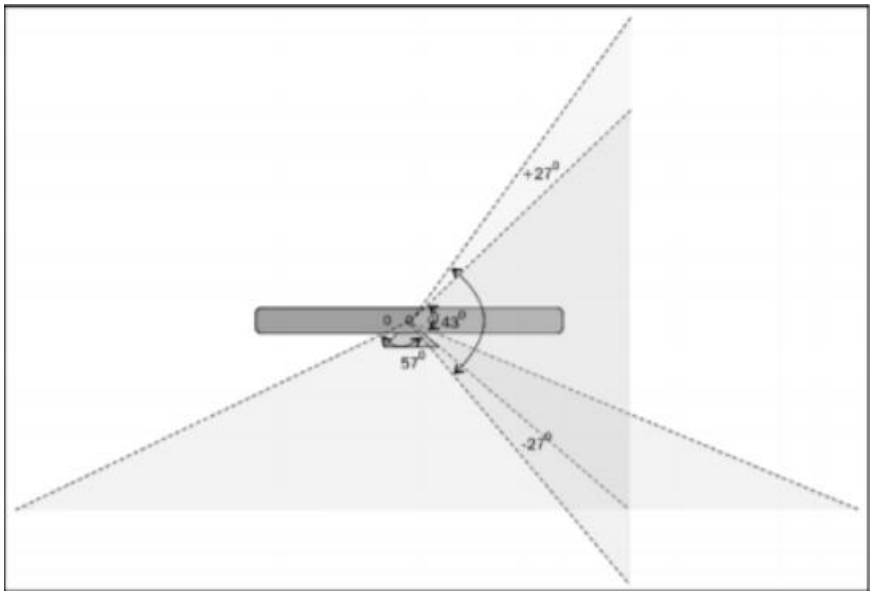

Fig. 1.9 Suportul motorizat [6]

1.2 Formatul JSON

JSON este un format de reprezentare şi interschimb de date între aplicaţiile informatice. Este un format text utilizat pentru reprezentarea obiectelor şi a altor structuri de date şi este folosit în special pentru a transmite date structurate prin reţea, procesul purtând numele de serializare [7].

Acest format este construit pe 2 structuri:

- Colecţie de perechi de nume ale variabilelor şi valorile lor
- Lista ordonată a valorilor

În acest proiect sunt folosite ambele structuri ale sale pentru a a transmite date între moduluele componente ale acestuia.

1.3 Chitul de dezvoltare C++ Rest

Este un proiect dezvoltat de compania Microsoft ce ajută dezvoltatorii de sisteme scrise în C++ să se conecteze şi să interacţioneze mult mai uşor cu serviciile web, oferindu-le acestora o serie de funcţii special implementate. Acest chit ajută la o dezvoltare facilă a sistemelor ce se bazează pe arhitectura client-server, arhitectură folosită pentru a interconecta modulele sistemului. [8]

Unele funcţii oferite sunt pentru a facilita procesarea şi serializarea informaţiilor în format JSON, sau pentru a procesa cererile de tip: GET, POST, DELETE.

2. Procesarea de imagini

Acest pas reprezintă partea de achiziţie a datelor din proiect. Modulul Kinect face o poză la masa de lucru a braţului robotic, imagine care este procesată pentru a găsi poziţia şi orientarea suportului de scris. Acest modul returnează coordonatele colţurilor suportului dreptunghiular de scris, coordonate ce reprezintă intrările în modulul de scriere a sistemului.

2.1 Integrarea modulului Kinect în sistemul de achiziţie de imagini

Pentru că suportul de scris nu va avea mereu aceaşi poziţie, orientare sau dimensiune, este nevoie de un modul de achizţie a datelor din mediul înconjurător. Cum deja am spus acest modul foloseşte camera Kinect pentru a fotografia de sus masa de lucru pe care este montat braţul robotic, masă pe care se află suprtul de scris, de culoare albă şi un reper, un dreptunghi de culoare roşie, pe care îl voi folosi ulteriror pentru a calcula coordonatele suportului de scris.

Pentru dezvoltatorii de software Microsoft, producătorul Kinect pune la dispoziţie un API compatibil cu limbajul de programare C# ce permite utilizarea modulului în diferite sisteme. Sistemul dezvoltat de mine este scris în limbajul de programare Python, fiind nevoie astfel de un wrapper pentru a putea expune API ul Kinect for Windows în Python.

Wrapperul folosit este PyKinect, o bibliotecă de funcţii ce permite folosirea modulului hardware Kinect, cu ajutorul comenzilor din Python, oferind dezvoltatorilor un set de funcţii echivalente cu cele din API-ul oferit de Mircrosoft. Această bibliotecă este foarte importantă în proiect deoarece a permis accesul la funcţionalităţile camerei, şi cel mai important, a permis, obţinerea în program a unei imagini color a masei de lucru printr-un simplu apel de funcţie.

2.2 Recunoaşterea formelor

Imaginea obţinută prin procedeul descris la pasul anterior, necesită transformări pentru a obţine informaţiile necesare sistemului. Sistemul este gândit să funcţioneze dacă în imagine se află două obiecte esenţiale: suportul de scris, de culoare albă, şi repreul, un dreptunghi de culoare roşie.

Pentru a obţine informaţiile necesare, întreaga problema se reduce la detecţia a două forme în imagine: dreptunghiul alb corespunzător suportului de scris şi dreptunghiul roşu corespunzător reperului ales. Orice alt element din imagine trebuie ignorat total pentru buna funcţionare a sistemului.

Pentru a obţine informaţiile despre cele două obiecte de formă şi culoare cunoscută trebuie să procesăm imaginea. Procesarea se va face folosind biblioteca Open CV pentru Python şi următorul algoritm:

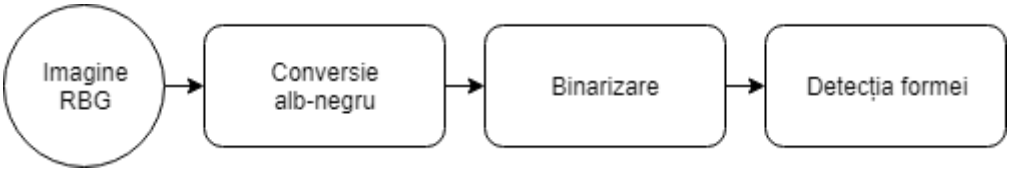

Fig. 2.1 Algoritmul de detecţie a formelor

2.2.1 Librărira OpenCV

OpenCV (Open Source Computer Vision Library) este o bibliotecă open source dedicată dezvoltării de software pentru machine learning şi computer vision. A fost construită pentru a oferi o infrastrucutră comună în aplicaţiile de computer vision şi pentru a facilita folosirea acestor algoritmi în produse comercială.

Biblioteca are peste 2500 de algoritmi optimizaţi, ce pot fi folosiţi să detecteze şi să recunasca fete, să identifice obiecte şi forme, să clasifice mişcări din fluxuri video, să urmărească mişcările camerei sau a obiectelor în mişcare, extragerea modelelor 3D ale obiectelor sau găsirea imaginilor similare dintr-o bază de date. Biblioteca este folosită în mod special în companii, în în grupuri ce cercetare şi în cadrul organizaţiilor guvernamentale. [9]

Pentru proiectul meu am folosit OpenCV pentru a recunoaşte formele din imagine, implicit a celor două dreptunghiuri, şi pentru a afla poziţia fiecărui colţ al acestora.

2.2.2 Conversia alb-negru

Premisa de la care pleacă procesarea imaginilor în acest proiect este că, pe masa de lucru se află un singur suport de scris de culoare albă. Prin urmare, găsirea suportului de scris pe masă se reduce doar la găsirea unui dreptunghi de culoare albă pe o suprafaţă de o culoare oarecare, diferită.

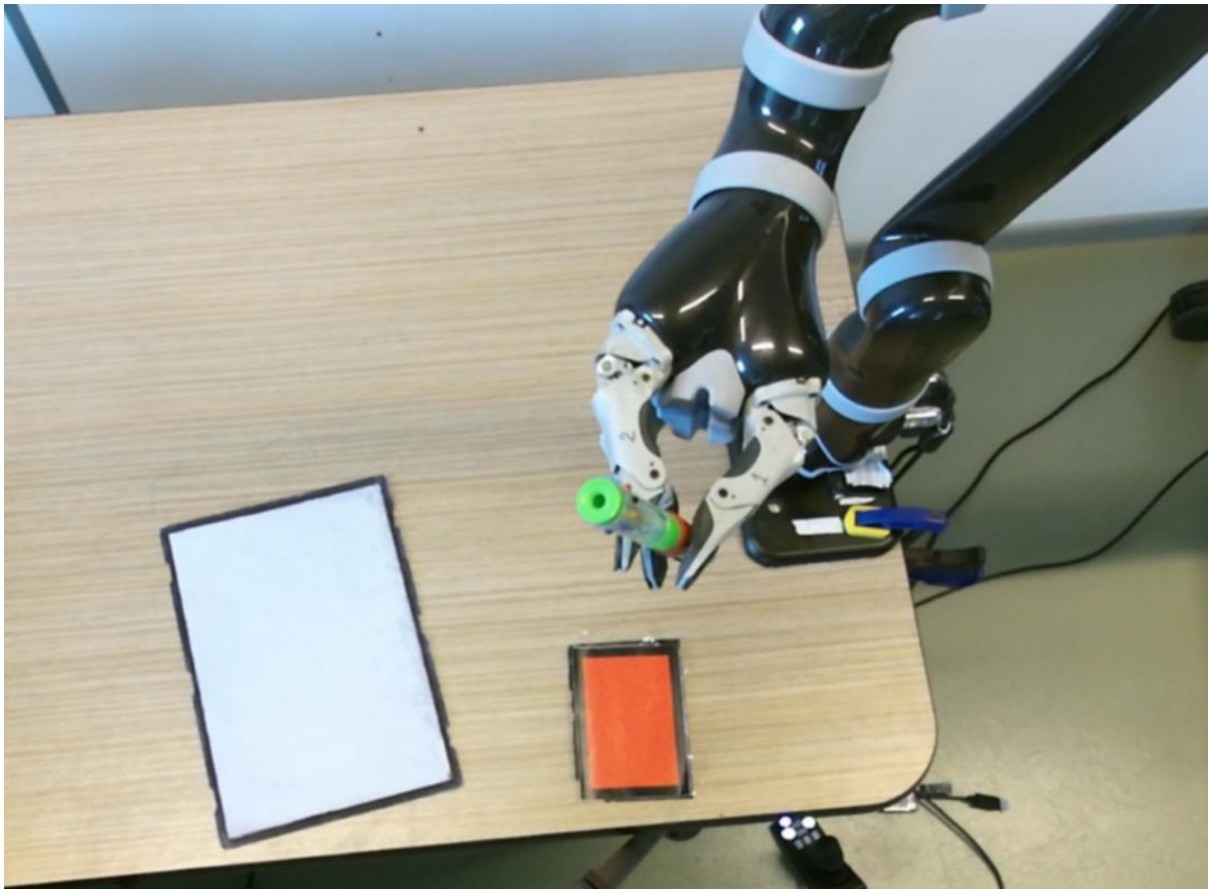

Fig. 2.2 Imagine color a spaţiului de lucru

Pentru simplificarea detectării formelor am apelat la procedeul numit binarizare. Imaginea preluată de la camera, utilizant funcţia getLastColorFrame din librăria Pykinect, este sub forma a 3 matrici. Fiecare matrice are dimensiunea 1280x960, adică numărul de pixeli ai imaginii. Fiecare matrice reprezintă valoarea pixelilor pe câte un canal de culoare:roşu, verde şi albastru. Prin suprarpunerea celor trei matrici se obţine imaginea color. Imaginea obţinută de la cameră fiind o imagine pe 8 biţi, valorile elementelor celor 3 matrici vor varia între 0 şi 255.

Transformarea imaginii color în imaginea alb negru se face prin generarea unei noi matrici, cu aceaşi dimensiune ca cele 3 ale imaginii color. Valoare fiecărui pixel din matricea imaginii alb negru se calculează conform formulei (1).

$$A(x,y) = 0.2989 * R(x,y) + 0.5870 * G(x,y) + 0.1140 * B(x,y) \qquad (1)$$

Valorile pixelilor din matrieca imaginii alb negru vor fi valori întregi ce variază între 0 şi 255, unde valoarea din matrice a pixelilor de culoare gri închisă (sau negru) va tinde să fie mică, şi cei deschişi la culoare vor tinde la o valoare mare. Având în vedere că suportul de

scris este de culoare albă pixelii imaginii în care apare suportul de scris, vor avea, în funcţie de luminozitatea din încăpere valori mai mari de 200.

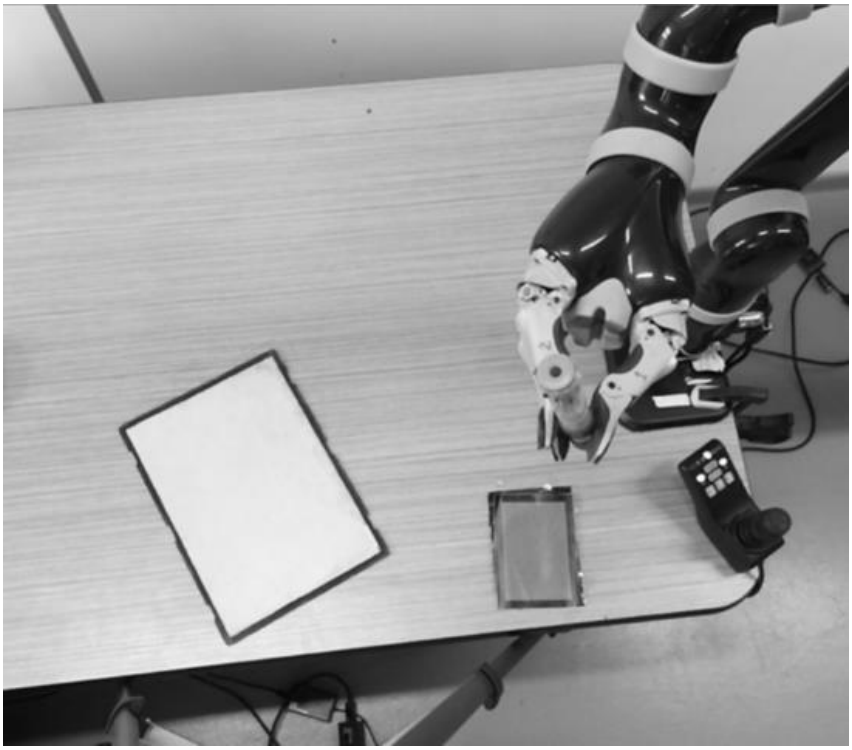

Fig. 2.3 Imagine alb-negru a spaţiului de lucru

2.2.3 Binarizarea

Binarizarea, reprezintă segmentarea imaginii în regiuni de interes şi eliminarea regiunilor considerate neesenţiale. În cazul nostru regiunea de interese este regiunea în care se află tăbliţă metalică. Cea mai simplă binarizare foloseşte un singur prag pentru a izola obectele de interes, astfel pixelii cu valori sub prag vor avea după binarizare valoarea 0 (negru) şi pixelii cu valori peste prag vor avea după binarizare valoarea 255 (alb pur).

Imaginea binarizata va conţine astfel doar un dreptunghi alb în locul în care se află suportul de scris, celelalte detalii fiind astfel ignorate.

Fig. 2.4 Imagine binarizată a spaţiului de lucru

2.2.3 Detecţia dreptunghiului.

Pentru detectarea dreptunghiurilor am folosit un algoritm de aproximare a formelor din librarira OpenCV. Binarizarea imaginii făcută anterior, oferă o preciezie mai mare funcţiei de aproximarea a formelor, deoarece între obiect şi fundal va fi contrastul maxim posibil, astfel urmărirea suprafeţei de spearatie dintre cele două va fi mult mai uşor de făcut.

Pentru eficienţa memoriei, algoritmul returnează coordonatele colţurilor fieacarei forme. Fiecare pereche (x,y) reprezintă pixelul în care se află un colţ al formei.Valorile x şi y reprezinte linia şi coloana matricii în care se află pixelul respectiv. În cazul de faţă, în imagine fiind o singură formă, vom obţine, în urma aplicării algoritmului de aproximare a formelor,4 coordonate pentru fiecare colţ al dreptunghiului. [9] Aceste coordonate reprezintă poziţionarea fiecărui colţ al imaginii faţă de colţul stânga sus al imaginii. Este nevoie de încă o procesare a acestor date pentru a obţine coordonatele relative la baza braţului robotic.

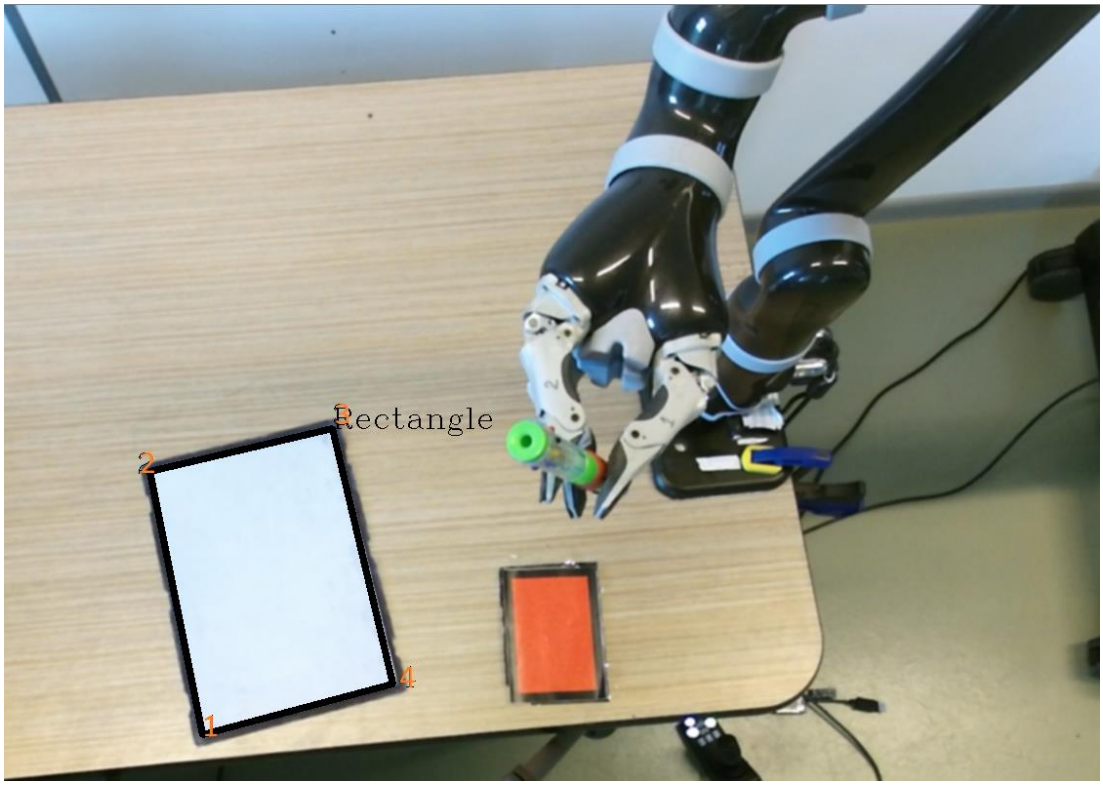

Fig. 2.5 Detecţia suportului de scris

Numerotarea colţurilor formei a fost făcută pentru a înţelege mai bine modul de funcţionare al algoritmului.

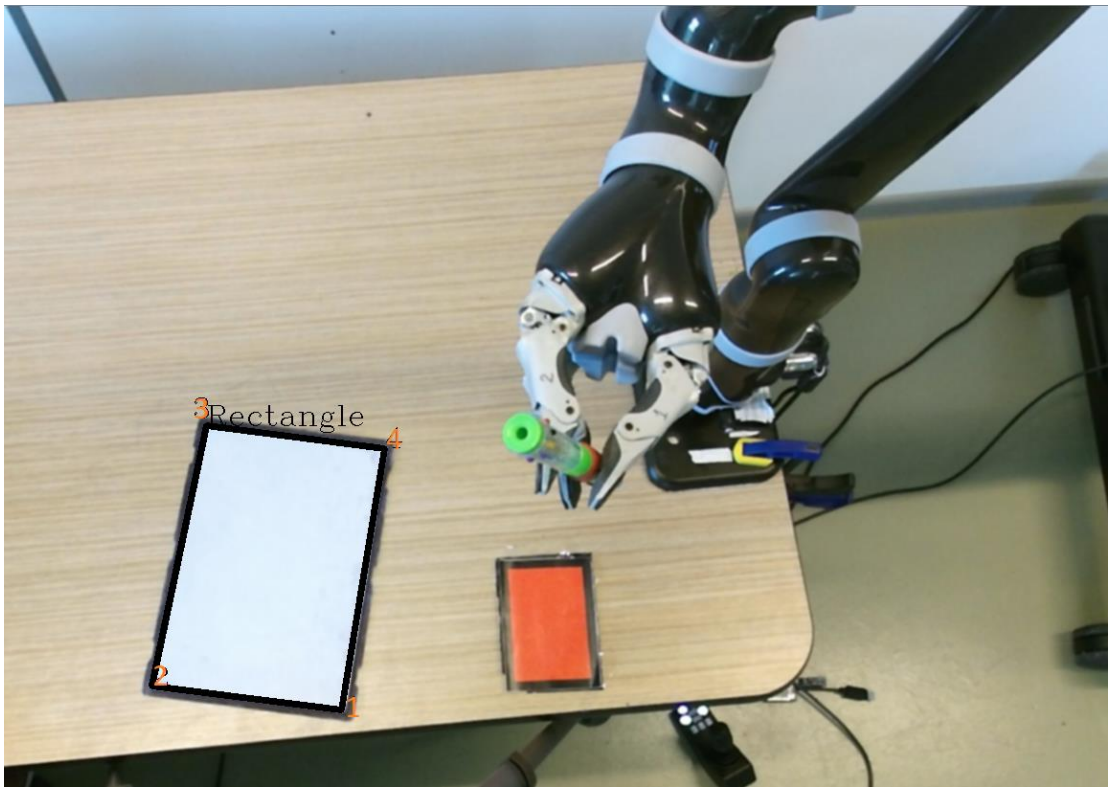

Fig. 2.6 Detecţia suportului de scris în poziţie diferită

Se observă din figurile 14 şi 15 că există cazuri în care colţul de interes, adică locul din care braţul ar trebui să scrie nu este întotdeauna colţul 4, ci în unele cazuri, în funcţie de orientarea paginii şi a camerei colurile 3,2,sau 1. Tot din aceleaşi figuri se observă că în sistem camera este orientată diferit faţă de braţul robotic, prin urmare axa X din planul imaginii reprezintă axa Y a braţului robotic şi invers.

2.2.4 Detecţia reprerului.

Detecţia reperului este asemănătoare cu detecţia suportului de scris, doar că, acesta fiind de culoare roşie nu mai este necesară o conversie din imagine color în imagine alb-negru, datele fiind reprezentate de matricea tonurilor de roşu din imagine.

Fig. 2.7 Imaginea canalului culorii roşu

Am detaliat anterior modul de a obţine o imagine color prin suprapunerea celor 3 matrici pentru fiecare culoare primară. Matricea tonurilor de roşu, conţine valori între 0 şi 255, pentru a arăta nivelul de roşu prezent în fiecare pixel. Prin urmare ca şi în imaginea alb negru, o zonă cu valori ale pixelilor mari va indica prezenţa unui obiect de culoare roşie.

Cum reperul este un dreptunghi de culoare roşie, pixelii acestuia vor avea valori mai mari de 200, în funcţie de luminozitatea din din cameră. Prin urmare pe imaginea corespunzătoare tonurilor de roşu se poate face o binarizare la fel ca cea făcută pentru a obţine coordonatele suportului de scris. Algoritmul aplicat pentru detectarea formelor este identic cu cel folosit anterior, şi vă este aplicat la fel pe o imagine binarizata.

Fig. 2.8 Binarizarea imaginii canalului roşu

Deoarece culoarea alb reprezintă o valoare mare pixelilor din toate cele 3 matrici ale culorilor primare, în urma procesului de binarizare pe imaginea canalului culorii roşii vom obţine şi formele de culoare albă. Evident algortimul de recunoastera a formelor va recunoaşte şi suportul de scris, însă acesta este ignorat, în acestă etapă fiind luate în considerare doar formele de culoare roşie .În final, pentru că pe masă se află un singur chenar roşu vom afla coordonatele reperului.

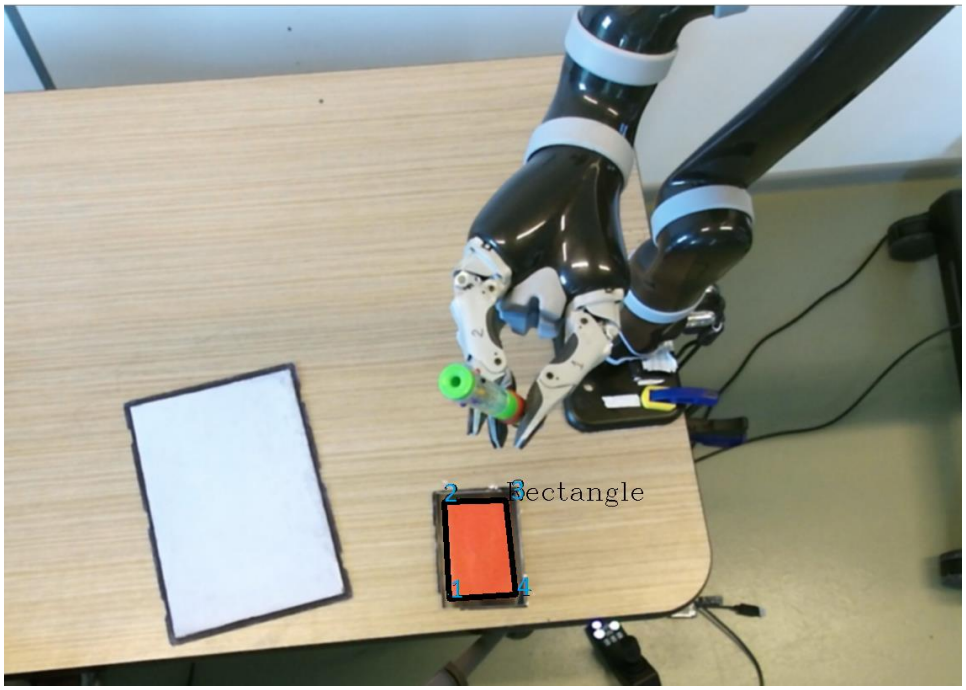

Fig. 2.9 Detecţia reperului

2.3 Translaţia din coordonate imagine în coordonate relative la baza braţului.

Cum am spus anterior, coordonatele obţinute prin aplicarea alogritmului de mai sus, reprezintă doar linia şi coloana matricii în care se află pixelii corespunzători colţurilor tăbliţei magnetice. Este necesară o prelucrare adiţională pentru a transpune valorile obţinute în valori ale coordonatelor relative la baza braţului.

Premisa de la care am plecat este că suportul de scris poate avea dimensiuni diferite şi că înălţimea de la care este făcută fotografia camerei de lucru nu este mereu aceaşi. Prin urmare a fost necesar introducere unui reper, care să fie mereu amplasat în aceaşi poziţie faţă de braţul robotic şi să aibe o dimensiune cunoscută. Reperul este reprezentat de chenarul roşu de pe masa de lucru, chenar al cărui centru se află la coordonatele (-0.2; -0.2) faţă de bază braţului, şi ale cărui dimensiuni sunt 8cm lăţime şi 12 cm lungime.

Pentru a transpune coordonatele colţurilor suprotului de scris în coordonate relative la baza braţului trebuie calculată poziţia relativă a acestora faţă de centrul reperului.

Cunoaştem coordonatele din imagine a colţurilor repreului.Media acestora reprezintă coordonatele pixelului în care se află centrul reperului. Corespondentul acestor două valori în coordonate relative la baza braţului sunt -0.2 şi -0.2. Din coordonatele colţurilor putem afla şi dimensiunea în pixeli a celor două laturi ale dreptunghiului: înălţime şi lăţime. Aceste două valori obţinute sunt foarte utile, deoarece pe baza lor putem calcula distanţa din mediul real pe care o reprezintă un pixel.

Putem calcula distanţa euclidiană între colţurile 1 şi 2 pentru aflarea valorii lăţimii reperului în pixeli şi distanţă euclidiană între colţurile 1 şi 4 pentru a afla valoarea înălţimii reperului în pixeli.

Prin însumarea acestor două valori şi împărţirea lor la 20 vom obţine numărul de centrimetri pe care un pixel din imagine îl cuprinde în mod real. Astfel putem exprima poziţia relativă a colţurilor la baza braţului în dimensiuni reale, nu numai în pixeli.

Formula de calcul a coordonatelor relative a unui punct din imagine la baza braţului este următoarea:

$$X_i = ((X_P - X_C)/\text{dimensiunePixel} + 20)/100 \quad (2)$$

$$Y_i = ((Y_P - Y_C)/\text{dimensiunePixel} + 20)/100 \quad (3)$$

Evident punctele pe pe care vom aplica această formulă sunt punctele în care se află colţurile suprtului de scris, obţinând astfel o poziţie exactă a acestora faţă de bază braţului robotic.

2.4 Alegerea punctului de start.

Pentru o flexibilitate cât mai mare a sistemului am considerat că poziţia camerei, şi implicit a locaţiei din care este făcută fotografia mesei de lucru, să nu fie aceaşi. Singura condiţie pentru funcţionare a sistemului fiind ca în imagine să fie vizibile cele două elemente cheie: suportul de scris şi reperul.

Algoritmul de detecţie a formelor, generează o listă cu cele 4 colţuri ale dreptunghiurilor găsite în imagine. Elementele din listă sunt ordonate prin parcurgerea în sens invers trigonometric a formei începând de la punctul care are cea mai mică valoare a lui X şi cea mai mare valoare a lui Y. Cu alte cuvinte, în funcţie de poziţia camerei, doar punctul din care alogritmul numerotează colţurile se va schimba, ordinea acestora fiind aceaşi.

Sistemul trebuie să scrie întotdeauna începând din colţul din stânga sus a suportului de scris pentru o asemănare cât mai bună cu modul în care oamenii scriu. Din acest motiv, modulul de analiză a textului va primii de la modulul de analiză a imaginii, o listă cu coordonatele colţurilor relative la baza robotului în ordinea următoare : stanga-sus, stanga-jos, dreapta jos, dreapta sus. Această ordine, în majoritatea situaţiilor nu coincide cu ordinea în care primim coodronatele colţurilor de la alogritmul de recunoaştere a formelor. Astfel, datele obţinute sunt procesate încă o dată pentru a decide poziţia fiecărui colţ faţă de braţ şi implicit ordinea în care acestea trebuie returnate pentru ca sistemul să funcitoneze corect.

3. Analiza textului şi scrierea literelor.

3.1 Metodă de scriere

Deoarece textul şi dimensiunea acestuia diferă, sistemul trebuie să fie capabil să îşi ajusteze dimensiunea de scriere a textului. Fiecare literă va avea alocat ca spaţiu în care să fie scrisă un pătrat cu latura egală cu dimensiunea fontului literei. Pentru a scrie literele în parte, sistemul va apela la un dicţionar ce conţine mişcările ce trebuie efectuate de braţ pentru a scrie literele, având ca reper locaţia pătratului, dimensiune şi orientarea acestuia.

Folosind pătratul alocat pe post de container pentru literă, scrierea unei litere se reduce la trasarea elementelor componente, a liniilor drepte sau oblice, sau a semicercurilor, în interiorul pătratului. Pentru scalare, dimensiunile elementelor componente vor fi raportate la dimensiunea pătratului, şi locaţia acestora va fi raprotata la colţul stânga sus a pătratului.

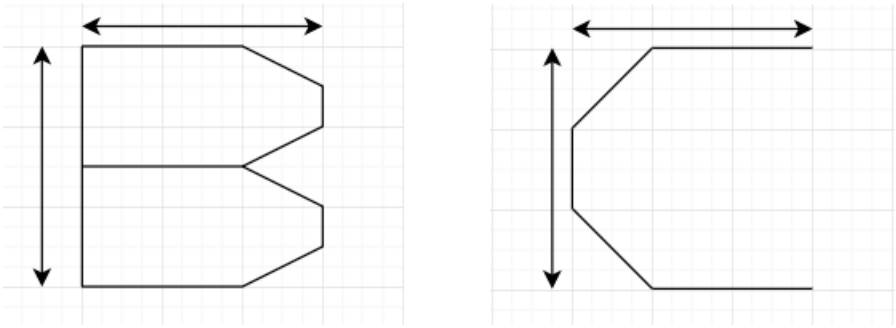

Fig. 3.1 Prototipul de scriere a literelor

3.2 Obţinerea datelor de la modulul de achiziţe a datelor.

Înainte de orice prelucrare inţială, sistemul de scriere a literelor are nevoie să ştie care este spaţiul în care acesta va acţiona. Aceste informaţii sunt obţinute de la modulul de achiziţie a datelor.

Pentru o bună modularitate, şi pentru scalabilitatea întregului sistem, am ales să abordăm în acest caz o arhitectură de tip client server. Această abordare a fost facilitată şi de limbajul de programare în care aceste aceste două module au fost scrise: Python, dezvoltarea unei arhitecturi client-server făcându-se cu ajuotrul uni microframework specific limbajului : Flask.

Astfel cele două module rulează independent undul faţă de celălalt comunicare dintre cele două metode făcându-se prin cereri de tip GET, în care modulul de scriere a literelor cere modulului de prelucrare a imaginilor să efectueze operaţiile necesare returnării poziţiei .

Modulul de achiziţie a datelor şi de prelucrare a imaginilor reprezintă partea de server şi oferă clientului, reprezentat de modulul de scriere a textului, informaţiile despre coordonatele colţurilor în format JSON.

3.3 Sistemul de coordonate bazale Kinova Jaco

Raza de acţiune în planul O X Y a braţului robotic Kinova Jaco este ilustrată în fig 3.2.

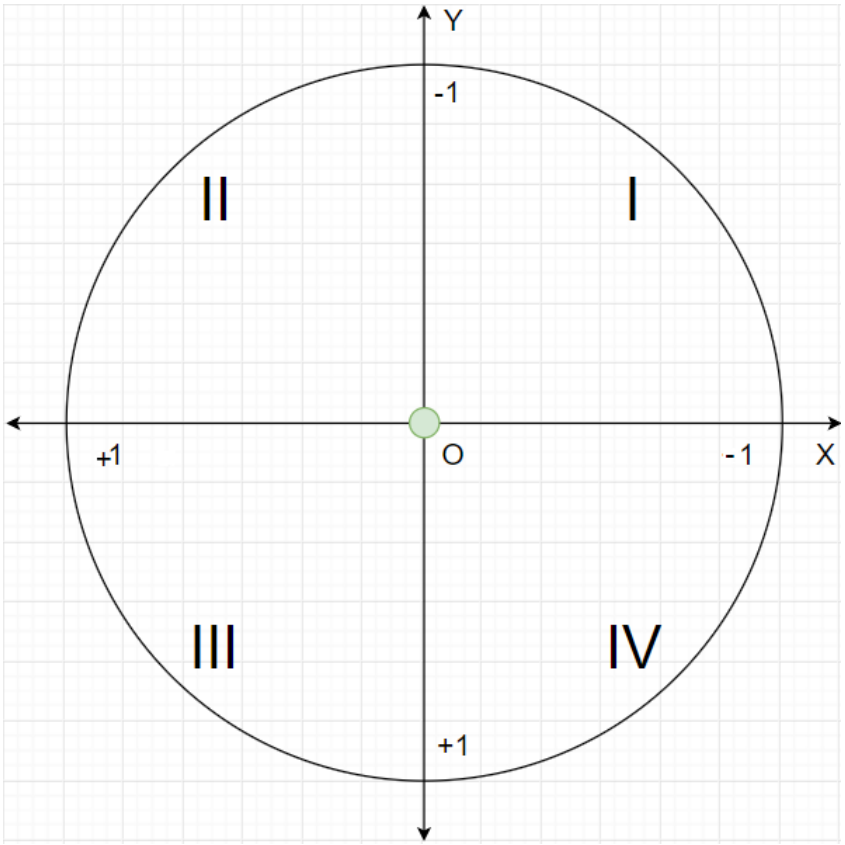

Fig. 3.2 Planul XOY a spaţiului de operare la baza robotului

Din fig. 3 observăm că punctele de pe abscisa şi ordonată iau valori în [-1 şi 1]. Funcţia de control cartezain din SDK-ul oferit de furnizor, are că parametrii valorile celor 3 coodronate [X,Y,Z] , valori reale, exprimate în metri.

Motivul pentru care în cazul scrierii literelor ne concentrăm doar pe planul OXY este pentru că presupunem că masa de lucru, masă de care este fixat braţul robotic, este o suprafaţă plană. În cazul acestui proiect sunt modifcate doar X şi Y. Coordonata Z, ce semnifică înălţimea la care se va afla vârful efectorului faţa de masă este o valoare constantă în sistem. Valoarea acesteia este de 0.02, ceea ce semnifică poziţionarea braţului la 2 cm deasupra mesei de lucru. Aceasta a fost găsită în mod experimental prin plasarea cu ajutorul joystick-ului a braţului în poziţia dorită şi prin citirea acestei coordonate folosind o funcţie special implenentată, descrisă în Capitolul 5, funcţie numită getCartesianPosition.

În punctul de origine O este situat braţul robotic. Deoarece braţul robotic este amplasat pe marginea mesei de lucru, utile pentru această aplicaţie vor fi doar cadranele I şi II.

Punctele din cadranul II sunt caracterizate de abscisa şi ordonată negative, iar cele din cadranul I sunt caracterizate de abscisa pozitivă şi ordonată negativă.

Dintr-un punct de coordonate (X_1;Y_1) aflat în cadranele I sau II pot fi făcute următoarele operaţii:

Deplasare la dreapta pe distanţa d: $X_2 = X_1 + d$ (4)

Deplasare la stânga pe distanţa d : $X_2 = X_1 - d$ (5)

Deplasare în sus pe distanţa d: $Y_2 = Y_1 - d$ (6)

Deplasare în jos pe distanţa d : $Y_2 = Y_1 + d$ (7)

Toate aceste operaţii, făcute separat sunt deplasări paralele cu cele două axe X şi Y, însă prin combinarea deplasării pe axa OX cu cele pe axa OY se pot obţine deplasări oblice.

3.4 Elementele componente ale literei

Fiecare literă este compusă din linii şi semciercuri, care la rândul lor sunt compuse din linii. Prin urmare a fost nevoie de implementarea a două funcţii diferite de desenare e unei linii şi a unui semicerc.

- Funcţia de plasare a unei linii într-un punct anume (X,Y) - goPoint
- Funcţia de desenare a unei linii din punctul de coordonate (x1, y1) în punctul de coordonate (x2, y2) - drawLine, este folosită pentru a desena orice linie dreaptă din componenţa unei litere, atât linii orizontale, verticale cât şi oblice.
- Funcţia de desenare a unei jumătăţi de elipsă - drawHalfCircle, se bazează pe aproximarea unei forme asemănătoare cu o jumătate de elipsă prin desenarea mai multor linii şi a fost implementată pentru a împuţina numărul de instrucţiuni din dicţionar în cazul literelor cu forme complexe că literele S sau B. Aceasta poate să deseneze o jumătate de elipsă ce trece prin două puncte date ca parametru, cu o semiaxa egală cu jumătate din distanţă celor două puncte, şi cu o a doua axă egală egală cu o valoare dată ca parametru acestei funcţii.

Imagini cu linii trase şi cu semicercuri de introdus aici.

Pe lângă aceste două funcţii au fost definite încă două funcţii care nu au rol în scrierea literelor:

- Funcţia de întoarcere în poziţia de început - goHome(), care este apleata la sfârişul scrierii textuli
- Funcţia de returnare a poziţiei carteziene a efectorului - getPosition(), utilă pentru a obţine informaţii despre valorile pe care le au coordonatele în anumite poziţii ale braţului.

Toate aceste funcţii sunt implementate pentru a face cererei la un server. Detaliile acestuia sunt descrise în capitolul 4.

3.5 Scrierea literelor

Cum fiecare literă în parte este scrisă diferit a fost necesar implementarea unui dicţionar ce conţine toate elementele ce trebuie desenate într-un container pentru a obitne fiecare literă.

În acest dicţionar, fiecare literă a alfabetului este încadrată într-un container pătrat, cu latura egală cu dimensiunea literei, dimensiune calculată de modulul de analiză a textuluil. Pătratul poate fi orientat sub diferite unghiuri faţă de axa OX a mediului de lucru al braţului. Unghiul de orientare al pătratului este reprezentat de unghiul dintre latura de sus a acestuia şi axa OX. Acest container reprezintă spaţiul rezervat de sistem, scrierii unei singure litere.

Avantajul folosirii unui container pătrat pentru fiecare literă este dat de faptul că fiecare element component al unei litere se află într-o poziţie raportată la colţurile pătratului, iar dimensiunea acestuia este şi ea raportată la dimensiunea totală a pătratului. Astfel obţinem un sistem, independent în care să putem descrie paşii necesari scrierii unei litere fără a fi influenţaţi de dimensiunea, locaţia şi orientarea acesteia.

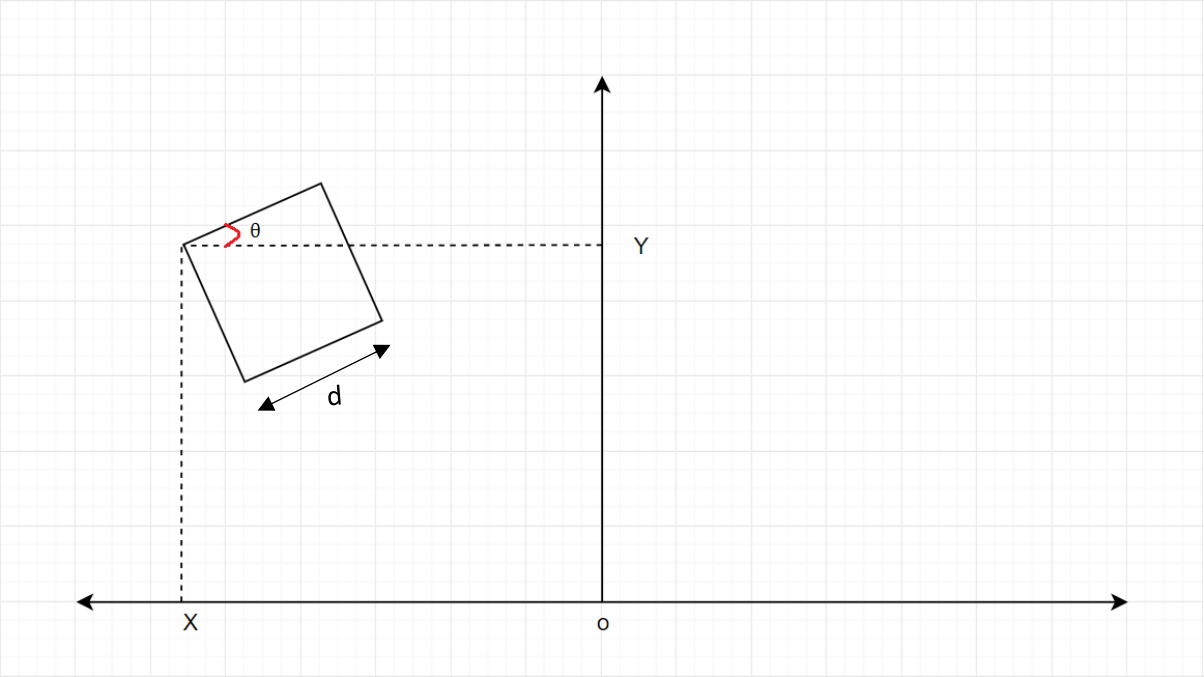

Fig. 3.3 Orientarea containerului în planul XOY

Pătratul în care urmează să fie scrisă litera este caracterizat doar de 3 informaţii: locaţia, reprezentat de coordonatele X,Y ale colţului stanga-sus, dimensiunea laturii, şi unghiul dintre latura de sus şi axa OX.

Presupunând că unghiul de orientare al pătratului este θşi latura acestuia este d putem calcula în cele ce urmează coordonatele colţurilor pătratului.

$$\text{Colţul dreapta sus} = \begin{cases} X - cos(\theta) * d \\ Y - sin(\theta) * d \end{cases} \quad (8)$$

$$\text{Colţul stânga jos} = \begin{cases} X - sin(\theta) * d \\ Y + cos(\theta) * d \end{cases} \quad (9)$$

$$\text{Colţul dreapta jos} = \begin{cases} (X - sin(\theta) * d) - cos(\theta) * d \\ (Y + cos(\theta) * d) - sin(\theta) * d \end{cases} \quad (10)$$

Se observă că, datorită unghiului diferit de zero, pentru o deplasare dealungul unei laturi a pătratului este nevoie să modificăm atât abscisa cât şi ordonată.

În cazul în care unghiul este 0, adică pătratul este orientat paralel cu axa OX atunci ecuaţiile de mai sus respectă deplasările simple pe axa OX la stânga şi pe axa OY în jos.

Având aceste puncte, raportându-ne la ele putem descrie conţinutul dicţionarului pentru câteva litere în parte.

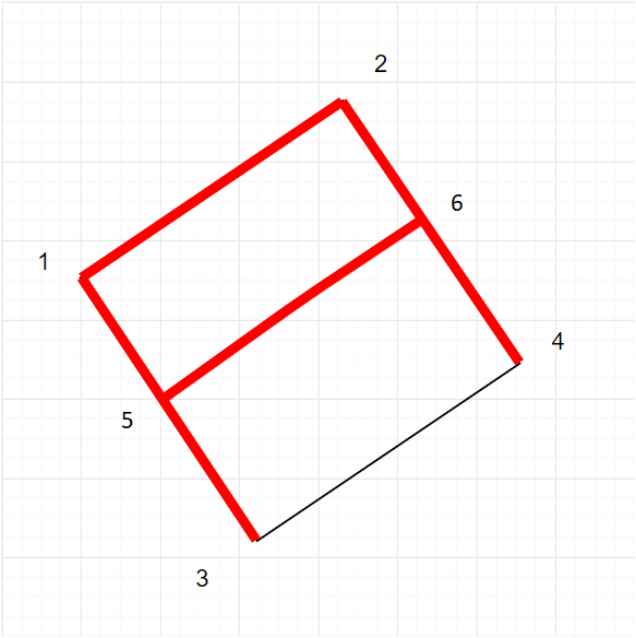

Fig. 3.4 Modelul literei A

De exemplu, scrierea literei A , reprezentată în fig 3.4. Presupunem că unghiul dintre axa OX şi dreapta dintre punctele 1 şi 2 este egal cu θ, iar locaţia pătratului în care este scrisă litera este reprezentată de coordonatele X şi Y şi are dimensiunea d.

Pentru scrierea acestei litere vom avea nevoie de:

- Line direapta între punctele 1 şi 2
- Linie dreaptă între punctele 1 şi 3
- Linie dreaptă între punctele 2 şi 4
- Linie dreaptă între punctele 5 şi 6

Coordonatele punctelor 1,2,3 şi 4 sunt identice cu coordonatele colţurilor dreptunghiului în care litera este reprezentată astfel:

Coordonate	Valori
X_1	X
Y_1	Y
X_2	$X - cos(\theta) * d$
Y_2	$Y - sin(\theta) * d$
X_3	$X - sin(\theta) * d$
Y_3	$Y + cos(\theta) * d$
X_4	X_3 - cos($\theta * d$
Y_4	X_3 - sin($\theta * d$

Tabel 5 Valorile coordonatelor colţurilor

Pentru trasarea acestor 3 linii va fi folosită funcţia de desenare a liniei, funcţie care va primii ca parametrii perechile de coordonate, calculate în tabelul X.

Pentru a trasa elemente ce încep din zone diferite de poziţia actuală a braţului este apelată funcţia goPoint pentru a poziţiona capătul efectorului în poziţia de început a noului element. De exemplu: trasarea linei între punctele 5 şi 6.

Punctele 5 şi 6 sunt două puncte ce să află la jumătatea distanţei dintre punctele 1 şi 3, respectiv 2 şi 4. Încadrarea fiecărei litere în pătrate cu o latură fixă, oferă în acest caz un mare avantaj, deoarece distanţa dintre punctele precizate mai sus coincide cu latura dreptunghiului. Pentru a ajunge în punctul de interes 5 este necesar să ne deplasăm dealungul dreptei dintre punctele 1 şi 2 pe o distanţă egală cu jumătate din latura pătratului. Acelaşi lucru se face şi pentru a ajunge în puctul 6.

Vom obţine astfel valorile:

Coordonate	Valori
X_5	X_1 - sin($\theta * d/2$
Y_5	Y_1 + cos($\theta * d/2$
X_6	X_2 - sin($\theta * d/2$
Y_6	Y_2 + cos($\theta * d/2$

Tabel 6 Valorile coordonatelor punctelor laterale

Apelând funcţia de trasare a liniei şi între coodronatele celor două puncte, sistemul trasează şi ultima linie componentă a literei A a alfabetului.

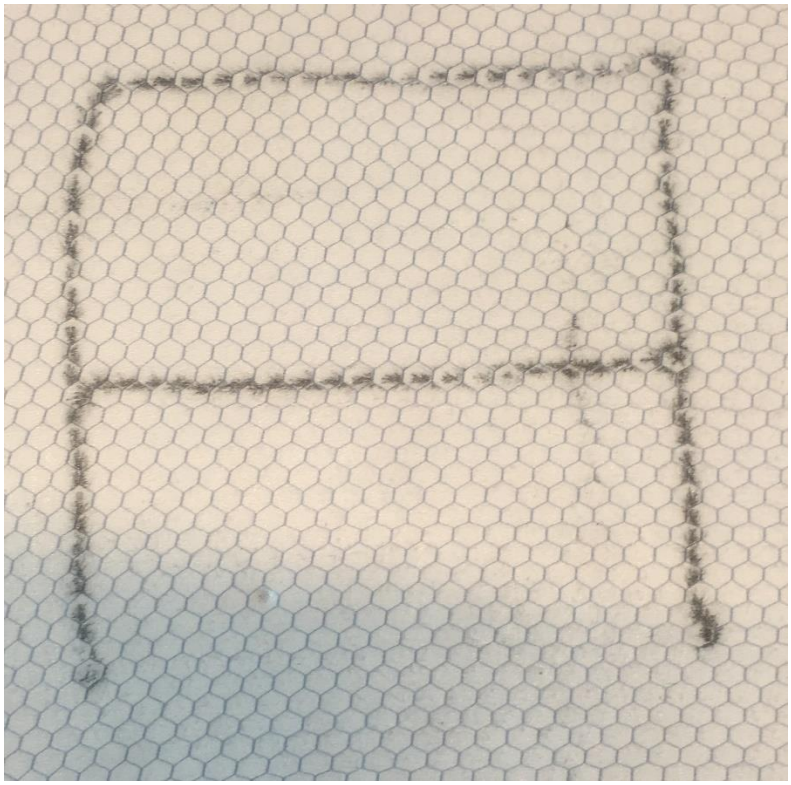

Fig. 3.5 Litera A

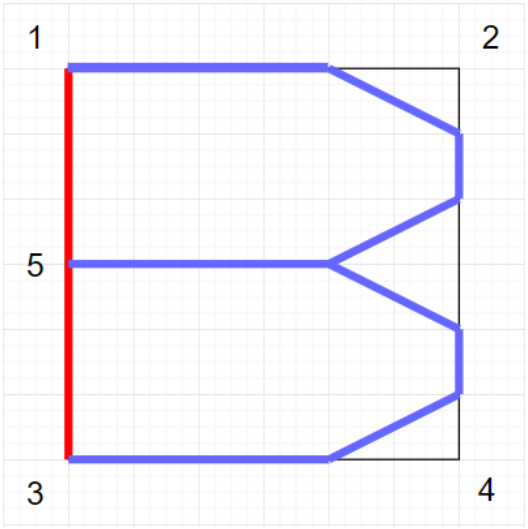

Fig. 3.6 Modelul literei B

Pentru scrierea literei B vom avea nevoie de următoarele componente:

- Linie dreaptă între punctele 1 şi 3
- Jumătate de elipsă între punctele 1 şi 5
- Jumătate de elipsă între punctele 5 şi 3

Ca şi în cazul literei A, valorile coordonatelor punctelor 1,2,3,4 şi 5 pot fi calculate folosind aceleaşi formule din tabelele 5 şi 6.

Coordonatele punctelor 1 şi 3 sunt paramtetrii funcţiei de desenare a linei.

Ilustrate cu albastru sunt cele două jumătăţi de elipsă aproximate de funcţia de trasare special implementată. O semiaxă este egală cu jumătatea distanţei dintre punctele 1 şi 5 respectiv 5 şi 3, iar cealată este egală chiar cu dimensiunea d a literei.

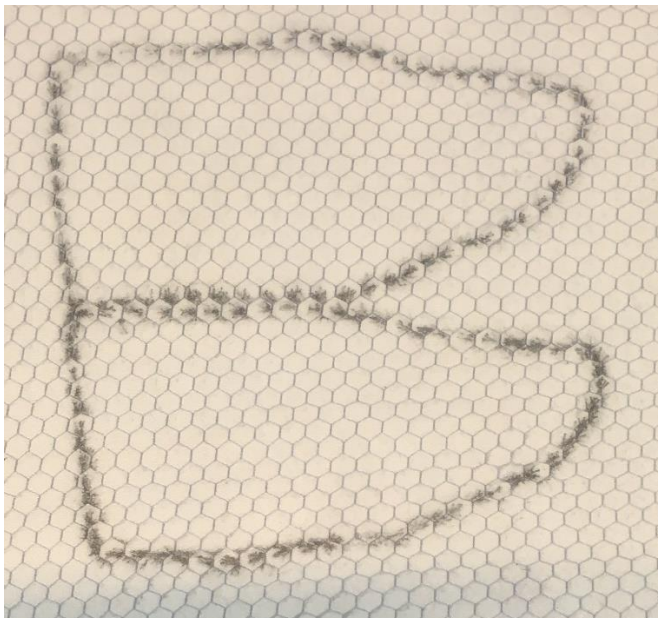

Fig. 3.7 Litera B

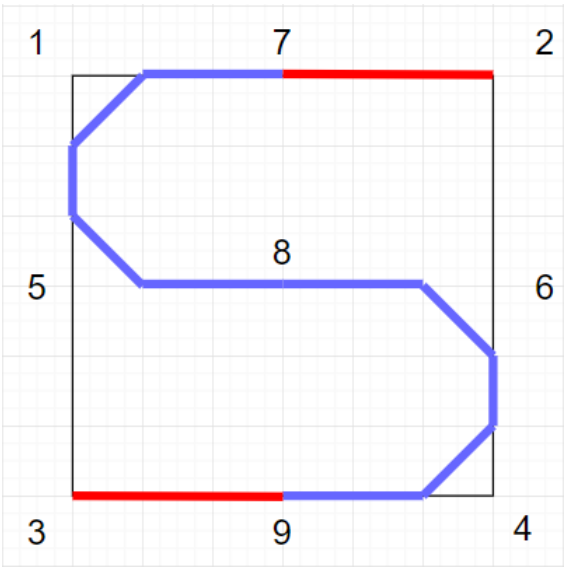

Fig. 3.8 Modelul literei S

Pentru scrierea literi S avem nevoie de următoarele componente:

- Line dreaptă între punctele 2 şi 7
- Jumătate de elipsă între punctele 7 şi 8
- Jumătate de elipsă între punctele 8 şi 9
- Linie dreaptă între punctele 9 şi 3

Coordonatele punctelor 7 8 şi 9 sunt calculate conform formulelor din următorul tabel. Pentru celelalte puncte ecuaţiile de calcul a coordonatelor sunt descrise în tabelul 7.

Coordonate	Valori
X_7	X_1 - cos($\theta * d/2$
Y_7	Y_1 - sin($\theta * d/2$
X_8	X_5 - cos($\theta * d/2$
Y_8	Y_5 - sin($\theta * d/2$
X_9	X_6 - cos($\theta * d/2$
Y_9	Y_6 - sin($\theta * d/2$

Tabel 7 Valorile coordonatelor punctelor centrale

Fig. 3.9 Litera S

Funcţia de aproximare a jumătăţilor de elipsă, poate desena ambele jumătăţi ale elipsei, în fucţie de parametrii primiţi.

Un alt exemplu al versatilităţii utilizării acestei funcţii este în cazul scrierii literi U.

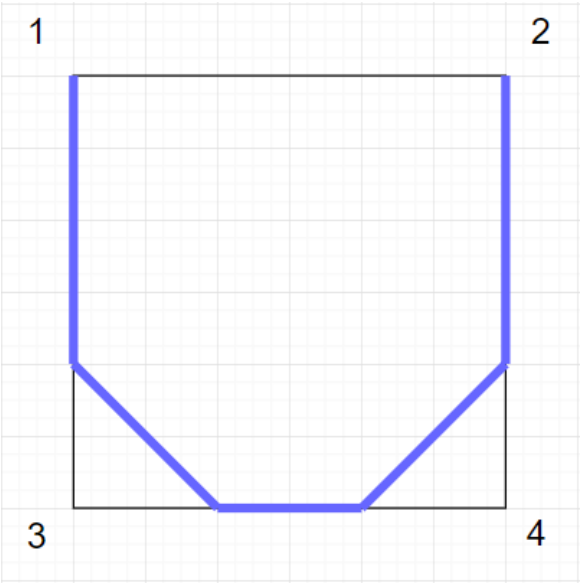

Fig. 3.10 Modelul literi U

În acest caz special, întreagă litera poate fi desenată folosind o singură jumătate de hiperbolă între punctele 1 şi 2

3.6 Scrierea textului.

De la modulul de achiziţie a datelor, modulul de analiză şi scriere a textului primeşte coordonatele colţurilor suportului de scris. Acestea sunt preluate şi stocate în 3 structuri de date numite leftLimit, ce coincide cu colţul 1 al suportului de scris, downLimit, ce coincide cu colţul 2 şi rightLimit ce coincide cu colţul 4. Aceste structuri de date au la rândul lor câte două atribute X şi Y reprezentând coordonatele punctelor.

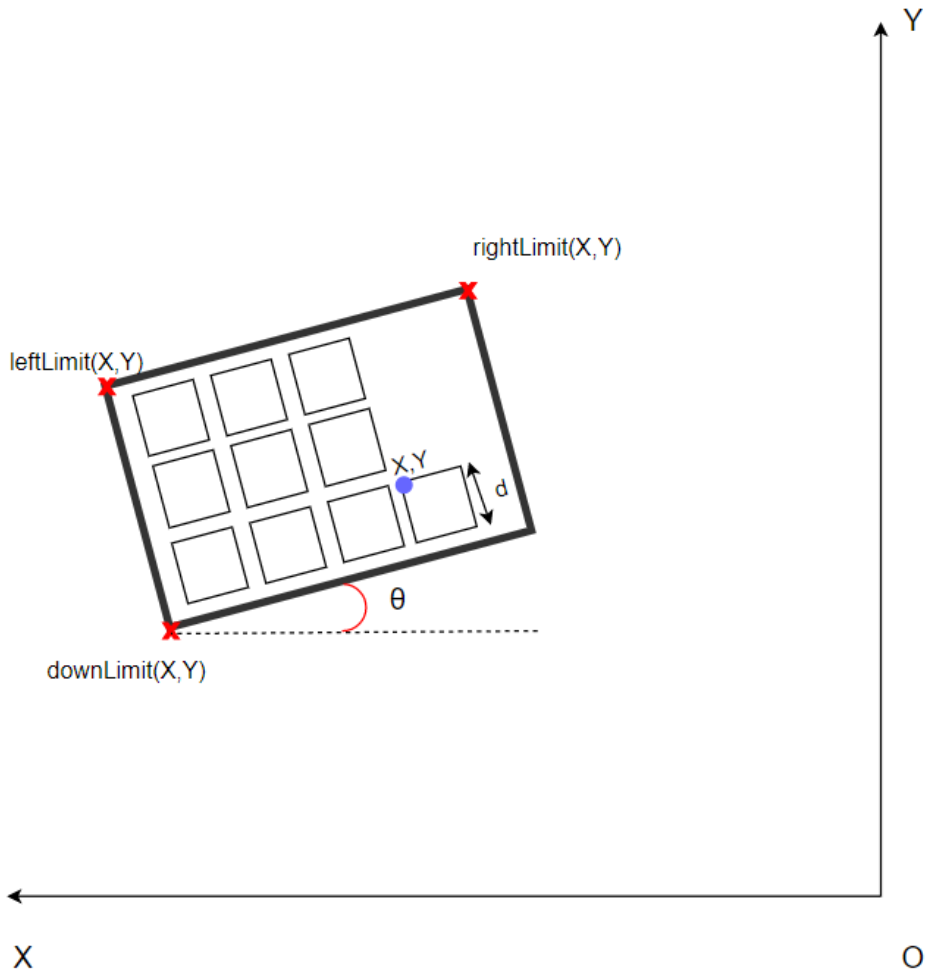

Fig. 3.11 Aşezarea în pagină a textului

Pe baza acestor date, pe lângă informaţia despre locaţie putem afla şi alte informaţii cum ar fi: orientarea faţă de axa OX dar şi dimensiunea acestuia. Orientarea suportului de scris va fi reprezentat de valoarea unghiului θ, unghi care va fi luat în calcul atât pentru scrierea fiecărei litere în parte, cât şi pentru scrierea întregului text.

Unghiul θ este calculat cu formula :

$$\theta = \text{arctg}\left(\frac{rightLimit.Y - leftLimit.Y}{rigthLimit.X - leftLimit.X}\right) \qquad (11)$$

Dimensiunea suportului de scris, calculat pe baza ariei dintre cele 4 puncte, este o informaţie cheie în dimensionarea textului, dimensiunea literelor fiind aleasă atât în funcţie de numărul de caractere pe care textul îl are, cât şi în funcţie de dimensiunea suportului de scris.

În figura 3.11 este reprezentat modul de alocare a spaţiilor pentru literele componente din textul "Ana are mere". Fiecărei litere din textul introdus îi este alocat iniţial un spaţiu în care să fie scrisă. Alocarea unui spaţiu se face prin modificarea variabilelor X şi Y, variabile care indică locul de început al scrierii unei litere.

Iniţial ele sunt iniţializate cu valorile atributelor structurii leftLimit, deoarece scrierea textului începe întotdeauna din colţul stânga sus al suportului de scris. Aceste valori sunt modificate usor pentru a indica un punct in interiorul suportului de scris. Este făcut acest artificiu pentru a evita scirerea exact din marginea suportului de scris.

Pentru scrierea unei litere în spaţiul caracterizat de cele două variabile sistemul apelează la dicţionarul deja implementat. După ce literă este scrisă trecerea la urătoarea litera şi implicit alocarea spaţiului pentru aceasta se face printr-o translatare de la stânga la dreapta dealungul paginii, sau printr-o trecere de pe un rând pe altul.

Modificarea pentru trecerea de la o literă la alata pe acelaşi rând se face folosind următoarele formule :

$$X = X - \cos(\theta)*d*1.2 \quad (12)$$

$$Y = Y - \sin(\theta)*d*1.2 \quad (13)$$

Translatarea la stânga se face pe o dimensiune mai mare decât dimensiunea unei litere pentru a lăsa un spaţiu liber între două caractere.

Modificarea variabilelor pentru trecerea de pe un rând pe altul se face folosind următoarele formule:

$$X = X_0 - \sin(\theta) *d*1.2*numarRand \quad (14)$$

$$Y = Y_0 + \cos(\theta) *d*1.2*numarRand \quad (15)$$

Număr rând reprezintă o variabilă locală în care se ţine evidenţa rândurilor scrise pe suportul de scris. Valoarea acesteia este iniţial 0 şi este incrementată la trecerea pe un rând nou.

3.7 Dimensionarea literelor.

Dimensionarea literelor este reprezentată de procesul în care este calculată latura pătratului în care fiecare literă este încadrată, şi implicit, spaţiul alocat pe tăbliţă de scris pentru litera din punctul X şi Y.

Algoritmul de dimensionare a textului, pe baza dimensiunii suportului de scris, calculează dimensiunea literelor în funcţie de raportul dintre numărul de caractere al textului şi dimensiunea tăbliţei magnetice. Dimensiunea iniţială obţinută astfel, este de regulă de ordinul centimetrilor. Plecând de la această dimensiune iniţială, care este şi dimensiunea maximă pe care caracterele le pot avea, sistemul scade progresiv dimensiunea cu 0.5 centimetri până la încadrarea întregului text în spaţiul determinat de cele 3 puncte caracteristice colţurilor suportului de scris.

Există două tipuri de verificări:

Verificarea 1: Dacă întreg cuvântul poate fi scris pe un rând.

Acest tip de verificare este făcut pentru a evita despărţirea unui cuvânt pe două rânduri diferite. Pentru a stabili dacă acesta este posibil se verifică dacă locaţia la care s-ar ajunge prin scrierea întregului cuvânt pe o linie se află în interiorul sau în exteriorul suprafeţei stabilite.

Există două cazuri posibile:

Cazul 1:

$$X - \cos(\theta)*d*1.2*numarCaractereCuvant < leftLimit.X \quad (16)$$

caz în care cuvântul poate fi scris pe un singur rând, prin urmare, pentru a rezerva spaţiul în care literele cuvântului vor fi scrise se vor modifica variabilele X şi Y pentru o translaţie de la stânga la dreapta dealungul suportului de scris, pe o dimensiune egală cu dimensiunea ocupată de cuvânt.

Cazul 2:

$$X - \cos(\theta)*d*1.2*numarCaractereCuvant > leftLimit.X \quad (17)$$

caz în care cuvântul depăşeşte limita. Dacă pe acest rând nu avem alte cuvinte scrise, sistemul scade cu 0.5cm dimensiunea literei şi verificarea 1 este reluată apoi pentru întreg textul. Dacă pe acest rând sunt şi alte cuvinte scrise, atunci numărul de rânduri este incrementat şi se face trecerea pe un nou rând, unde este reluată verificarea 1 doar pentru cuvântul curent.

Verificarea 2: Dacă numărul de linii nu a depăşit limita de jos a paginii

Cu această verificare sistemul se asigura că, după parcurgerea şi alocarea spaţiilor de scris pentru întreg textul nu a fost depăşită limita de jos a paginii.

Şi aici există două cazuri posibilie

Cazul 1:

$$Y - \cos(\theta)*d > downLimit.Y - \cos(\theta)*latimePagina \quad (18)$$

caz în care locul urmat de ultima literă scrisă se află sub colţul "dreapta-jos" al suportului de scris, şi în care dimensiunea literelor este scăzută cu 0.5 cm şi este reluată verificarea 1 pentru întreg textul.

Cazul 2:

$$Y - \cos(\theta)*d > downLimit.Y - \cos(\theta)*latimePagina \quad (19)$$

caz în care întregului text i-a fost alocat spaţiu în interiorul suprafeţei stabilite. În acest caz sistemul continuă cu scrierea efectivă a literelor în spaţiul alocat acestora, fiind asigurat faptul că scrierea literelor nu va depăşi limitele suportului de scris.

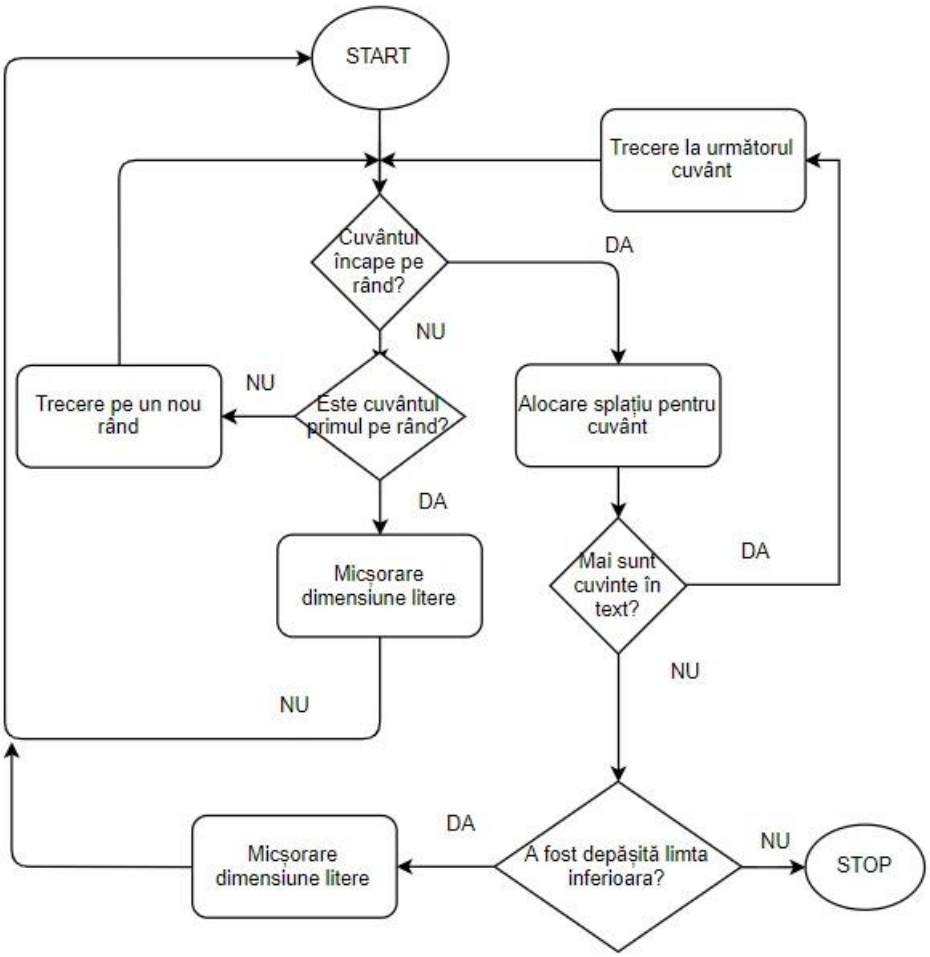

Fig. 3.12 Algortimul de dimensionare a literelor

3.8 Algoritmul de scris

Algoritmul de scriere a textului este similar cu cel de dimensionare a textului. Însă de data aceasta cea de-a doua verificare nu mai este făcută deoarece îndeplinirea acesteia este o cerinţă cheie pentru continuarea rulării programului. În schimb prima verificare este făcută şi în acest caz, însă nu este făcută pentru a modifica dimensiunea literelor, ci doar pentru a determina dacă are loc sau nu trecerea pe un nou rând.

Scrierea începe din punctul de coordonate leftLimit.X şi leftLimit.Y. Textul este parcurs şi pentru fiecare cuvânt are loc verificarea 1 pentru a determina dacă întreg cuvântul începe sau nu pe acel rând. În caz afirmativ este parcurs cuvântul litera şi sunt urmaţi paşii din dicţionar pentru scrierea în locaţia stabilită a literei respective, după care se face o translatare la stânga conform formulei (14), pentru a se trece la scrierea următoarei litere. În caz negativ se face trecere, după incrementarea variabilei ce reţine numărul de rânduri, la un nou rând şi se scrie cuvântul pe noul rând.

Trecerea la scrierea unei noi litere se face prin apelul funcţiei goPoint(X,Y) unde parametrii sunt noile variabile calculate.

În ambele cazuri după scrierea ultimului cuvânt este făcută o translaţie de la stânga la dreapta pe o distanţă egală cu jumătatea dimensiunii unei litere pentru a lăsa un spaţiu după cuvânt.

La terminarea scrierii întregului text, braţul se întoarce în poziţia iniţială, o poziţie numită de producător "Home Position" prin apelul funcţiei goHome.

4. Transpunerea datelor în mişcări ale braţului

4.1 Integrarea braţului în cadrul sistemului

În cadrul capitolului 2, capitol în care am descris braţul robotic, am precizat că, producătorul pune la dispoziţie dezvoltatorilor un chit de dezvoltare, cu ajutorul căruia aceştia pot integra braţul în diferite sisteme ce efectuează acţiuni specifice. Un astfel de sistem, este chiar cel dezvoltat de mine, folosind braţul într-un alt scop decât cel pentru care acesta a fost iniţal proiectat.

Chitul de dezvoltare conţine un set de biblioteci ce permit controlul braţului robotic, dar şi obţinerea de informaţii despre acesta. Deoarece, chitul conţine biblioteci scrise doar pentru limbaju de programare C++, controlul braţului folosind funcţiile preimplementate din bibliotecile chitului este imposibil de realizat, datorită incompatibilităţii dintre limbajul de programare Python, folosit pentru implementarea modulului de scriere a textului, şi a limbajului C++ pentru care SDK-ul a fost creat.

A fost aşadar implementată o arhitectură de tipul client-server, în care serverul este scris în C++ iar clientul este reprezentat de modulul de scrierea a textului. Serverul rulează pe calculatorul la care braţul robotic este conectat, şi folosind funcţiile din chitul de dezvoltare controlează braţul în funcţie de cererile primite de la client.

Avantajul unei astfel de arhitecturi abordate este dat de faptul că modulul de control al braţului robotic este un modul independent, şi poate fi integrat astfel cu uşurinţă în orice sistem, nefiind relevant limbajul de programare în care sistemul a fost scris. Ba chiar mai mult arhitectura client server permite utilizarea braţului robotic concomitent în mai multe sisteme specializate în efectuarea diferitelor sarcini, aflate sau nu pe acelaşi suport hardware.

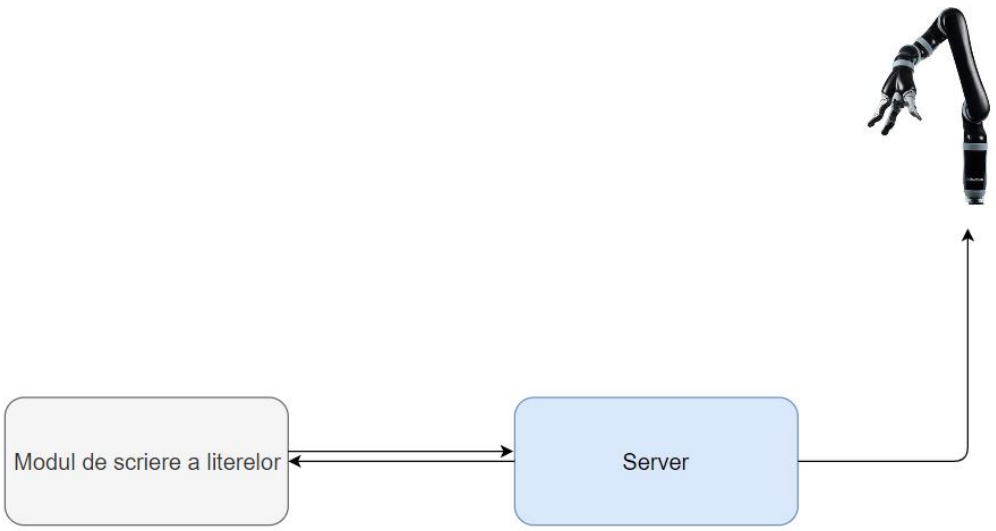

Fig. 4.1 Integrarea braţului în sistem

4.2 Chitul de dezvoltare Kinova API

Chitul de dezvoltare expune utilizatorului o serie de funcţii pe care robotul le are implementate deja în controlerul propriu. Toate acestea oferă controlul total al robotului prin apelul lor din orice sistem software în care sunt incluse bibliotecile din chitul de dezvoltare.

Pe lângă aceste funcţii, în chitul de dezvoltare sunt definite şi structuri specifice cum ar fi:

- CartesianInfo - structură cu 6 câmpuri :X,Y,Z pentru poziţionarea efectorului în spaţiu, şi ThetaX, ThetaY, ThetaZ pentru orientarea acestuia
- FingersPosition - structură cu 3 câmpuri: Finger1, Finger2, Finger3 ce stochează poziţia degetelor
- CartesianPoisition ce conţine 2 câmpuri: CartesianInfon şi FingersPosition
- TrajectoryPoint - reprezintă punctul final al unei traiectorii

Funcţii importante sunt:

- GetCartesianPosition(CartesianPosition &Response) funcţie ce returnează poziţia efectorului prin intermediul parametrului Response
- SendBasicTrajectory(TrajectoryPoint trajectory) funcţie ce comandă braţul să ajungă în poziţia punctului primit ca parametru

4.3 Kinova Arm Movement Server

Implementat în C++ acesta este modulul de comandă a braţului ce transpune datele generate de modulul de scriere a textului în mişcări fizice ale braţului robotic. Acest modul este implementat utilizant SDK-ul C++ Rest pentru a expune funcţiile braţului robotic ale braţului robotic.

Pentru acest proiect, modulul îndeplineşte două funcţii:

- Returnează poziţia şi orientarea braţului robotic
- Setează punctul în care braţul robotic să se deplaseze

Returnarea poziţiei este efactuată la cerearea de tip GET efectuată de client. În urma acestei cereri este apelată funcţia de returnare a poziţiei braţului: GetCartesianPosition şi rezultatul acesteia este returnat sistemului sub forma unui mesaj de tip JSON ce conţine valorile tuturor câmpurilor structurii CartesianPosition.

Setarea punctului în care braţul robotic să se deplaseze este făcut la cererea de tip POST venită de la client. Odată cu această cerere, clientul trimite un mesaj de tip JSON ce conţine coordonatele punctului în care braţul robotic se va deplasa. Pe server sunt extrase valorile coordonatelor şi este creat, pe baza acestora o instanţă a structurii Trajectory. Funcţia

sendBasicTrajectory este apelată, iar mişcarea încetează când poziţia efectorului variază în limita a 0.2 cm pe fiecare axă.

Eroarea în atingerea poziţiei este introdusă deoarece, experimental am observat că sensibilitatea actuatorilor scade dealungul scrierilor repetate, şi braţul tinde să se blocheze în încercarea de a efectua mişcări precise. Acest lucru reprezintă şi un avantaj, în special în cazul în care braţul efectuează mişcările necesare aproximării unei elipse, trecerile de la o linie la alta făcându-se mult mai fluid.

Cererile primite de la client sunt făcute implementate în interiorul funcţiilor descrise în capitolul 4.3.

Funcţia de trasare a unei linii drepte face două cereri de tip POST, prima pentru poziţionarea braţului în punctul (X1,Y1) şi a doua pentru mişcarea braţului în punctul (X2,Y2). Restul variabilelor necesare poziţionării braţului sunt neschimbate şi sunt hardcodate, acestea fiind determeinate în mod experimental. Asemănător şi funcţia de deplasare într-un punct stabilit, face o singură cerere POST pentru deplasarea braţului în punctul(X1,X2).

Funcţia de întoarcere în poziţia de început face o singură cerere POST în punctul memorat de sistem ca fiind punctul "Home Position".

5. Concluzii

5.1 Concluzii generale

Principalul obiectiv al acestui proiect a fost dezvoltarea unui sistem complet autonom ce poate scrie textul dat de utilizator pe un suport de scris. Acest obiectiv a fost atins, sistemul fiind capabil să analizeze, să dimensioneze şi să poziţioneze textul în funcţie de locaţia tăbliţei magnetice fără intervenţia factorului uman.

Pornind de la premisa că braţul este iniţial dezvoltat pentru asistenţă persoanelor cu dizabilităţi, dar oferă şi posibilitatea integrării în diverse aplicaţii, am încercat să demonstrez versatilitatea utilizării braţului în diferite sisteme. Acest lucru a fost făcut prin integrarea acestui braţ în sistemul care este capabil să efectueze acţiuni specifice oamenilor, în mod complet independent.

5.2 Contribuţii personale

În cadrul acestui proiect contribuţiile mele au fost următoarele:

- Am creat sistemul ce integrează modulele de achiziţie de imagini, de analiză a textului şi de scriere a acestuia, precum şi echipamentele hardware specifice fiecăruia
- Am creat un sistem ce transpune coordonatele unor pixeli în coordonate relae, relative la baza braţului robotic
- Am creat algoritmul de dimensionare şi algoritmul de scriere a textului
- Am creat dicţionarul ce conţine toate mişcările necesare scrierii de către braţ a tuturor literelor mari de tipar din alfavetu limbii engleze

5.3 Dezvoltări ulterioare

Proiectul descris în această lucrare are rol demonstrativ. Prin intermediul său se poate observa potenţialul folosirii braţelor robotice în sisteme autonome, însă momentan utilitatea acestuia este restrânsă.

Ca dezvoltări ulterioare, sistemul ar putea integra un modul de transpunere a vorbirii în text, pentru a putea primi comenzi vocale de la utlilizator, nefiind astfel necesar nicun efort fizic din partea utilizatorului.

Sisemul de prelucrare a imaginilor poate fi dezvoltat astfel încât să detecteze poziția instrumentului de scris astfel încât braţul să îl apuce singur.

O altă dezvoltare ce ar putea fi adusă sistemului este reprezentată de extinderea dicţionarului, pentru că braţul să poată să scrie cu litere de mână, crescând astfel aplicabilitatea practică a sistemului.

6. Bibliografie

[1] "Kinova Robotics Website," Robotics Company, [Online]. Available: https://www.kinovarobotics.com/en. [Accessed 07 05 2019].

[2] A. Neacșu, "Autonomous System for Performing Dexterous, Human-Level Manipulation Tasks as Response to External Stimuli in Real Time," 2017.

[3] K. Robotics, "Kinova Jaco User Guide," [Online]. Available: https://www.kinovarobotics.com/sites/default/files/SDK-UG-INT-EN%20201804-1.0%20%28KINOVA%E2%84%A2%20Software%20development%20kit%20user%20guide%29_2.pdf. [Accessed 07 05 2019].

[4] "Gen2 Ultra lightweight robot," ROS Components, [Online]. Available: https://www.roscomponents.com/en/robotic-arms/45-gen2-ultra-lightweight-robot.html#/kinova_gripper_options-no/kinova_jaco2-6_dof. [Accessed 08 05 2019].

[5] K. Rovotics, "Kinova Jaco Specification," Robotics Company, [Online]. Available: https://www.kinovarobotics.com/sites/default/files/ULWS-RA-JAC-6D-SP-INT-EN%20201804-1.2%20%28KINOVA%E2%84%A2%20Ultra%20lightweight%20robotic%20arm%206%20DOF%20Specifications%29.pdf. [Accessed 08 05 2019].

[6] A. Jana, "Kinect for Windows SDK Programming Guide," [Online]. Available: https://www.pdfdrive.com/kinect-for-windows-sdk-programming-guide-e167319818.html. [Accessed 15 05 2019].

[7] "Introducing JSON," [Online]. Available: https://www.json.org/. [Accessed 10 05 2019].

[8] "Using the Microsoft C++ REST SDK," [Online]. Available: http://www.drdobbs.com/windows/using-the-microsoft-c-rest-sdk/240164544. [Accessed 21 05 2019].

[9] "OpenCV Web Page," [Online]. Available: https://opencv.org/. [Accessed 12 05 2019].