Kinect Controlled Chess Playing Robot

Senem Tanberk
Computer Engineering Department
Doğuş University
İstanbul, Turkey
2014194003@dogus.edu.tr

Dilek Bilgin Tükel
Control and Automation Engineering Department
Doğuş University
İstanbul, Turkey
dtukel@dogus.edu.tr

***Abstract*—In this project, we developed a chess play system based on Kinect vision sensor, which recognizes different hand gestures as a command and sends these commands to V-REP robotic simulation tool. Delta type robot will play chess according to the commands sent via vision software. In our system, we have tested for 2 different chess opening scenarios and DOF checking mode scenario, then we obtained results. Hence we propose an integration between Kinect hand gestures and V-REP simulation.**

Keywords—component; delta robot, vision, chess, simulation, V-REP

I. Introduction

The earliest findings for the chess game are found in the Egyptian pyramids of the first dynasty (3000–2800 BC). In these reliefs (Fig.1), Nefertiti plays a chess-like game. Antique chess was played in China, Mesopotamia and Anatolia. First written documents about chess are the Sanskrit texts written at the time of Indian Emperor II. Chandragupta (M.S.380-415). In these texts, the game is called as 'Chaturanga' meaning four battalions army . Around 600, the game's latest version of rules are thought be settled at Punjabi area and in the same century chess is thought to have been started to be played by the name "Sat-RanChu". The Indian ambassador brought the game from India to Iran during the time of Shah Husrev I. Shatranj was the given name in the middle-east. According to some resources, the first chess set in Europe was the chess set gifted to King of Franks King Charlemagne by Khalif Harun Reşid. In various regions of Europe and Asia like Catalonia, Florence, Sicily, Poland, Britain, Holland, Norway, Albania, Uzbekistan and Afghanistan, a lot of chess sets belonging to middle age are found . [1].

The first chess robot is the Turk; which consists of a table and a human model produced by Wolfgang von Kempelen in 1769 (Fig. 2). This complex mechanical system is controlled by teleoperation [2].

Chess has always been a challenge for computer producers and software developers. They want to develop a computer and a software that can beat people hence they can prove their technology. Chess is also an attractive topic for human-machine interaction. It is necessary to produce solutions in areas such as image processing, coordination and strategy establishment to solve the chess problem.

The Classical World Chess Champion Vladimir Kramnik said that: "There are a lot of things in chess which are very typical to life: you have to understand that sometimes you have to sacrifice a little bit of something to get other advantages, you have concentrate sometimes on one part of board and to sacrifice another one, you have to see the whole board, whole picture, otherwise you will never be a good chess player. In a way in life it is also like this. Decision making... every move you make it's very complicated and very difficult to explain how it happens. But there are a lot of ideas which come to mind and finally you make a decision." [3]

Fig. 1. Nefertiti playing Senet [4]

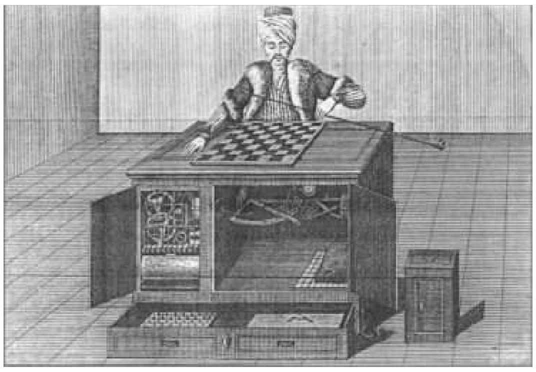


Fig. 2. Türk: Chess Automate [5]

A game like chess can be won only by thinking ahead: a player who is focused entirely on immediate advantage is easy to defeat. But in many other games, such as Scrabble, it is possible to do quite well by simply making whichever move seems best at the moment and not worrying too much about future consequences. This sort of myopic behavior is easy and convenient, making it an attractive algorithmic strategy. Greedy algorithms build up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. Although such an approach can be implausible for some computational tasks, there are many for which it is optimal.

In this paper, a simulation infrastructure is developed for chess-playing delta robot. In Section II and III, the robot chess system will be introduced, and the pertaining subsystems will be explained. In Section IV, software architecture and in Section V methods are shown. In last section, the conclusions are drawn and the future work is discussed.

II. SYSTEM COMPONENTS

In this study, it is aimed to develop interactive game application. Kinect 2 sensor was used to retrieve video stream and data is analyzed for classifying chess player movements.

The Windows compatible Kinect 2 sensor has the RGB camera, depth sensor, microphone hardware, and a software library that allows motion, face recognition, and sound detection in three dimensions. The RGB camera is 1920x1080 pixels, 30Hz (15Hz in low level light). This corresponds to a 70x60 field of view (FOV). 30Hz 512x424 pixel infrared sensor is used for depth detection and can detect a distance between 500mm and 4500mm . The Kinect 2 sensor can be connected to the PC via USB 3.0 with the Kinect Windows Adapter. The software and hardware used to communicate with the sensor are as follows:

- 64-bit 4-core laptop
- Operating system: on Windows 8.1
- Development Environment:
    - Visual Studio 2013:
    - Visual C #,
    - Visual C ++
    - Kinect Sdk 2.0 (Kinect Studio)
- Robot Simulator: V-REP (Lua scripts)

V-REP is software developed by Coppelia Robotics [6]. The training version is available free of charge. It works with Windows, Linux and MAC operating systems. Seven different languages can be programmed in V-REP, and our application has been developed using with Lua scripts.

ABB IRB360 delta robot is used as a robot model. Since the 3-D model of the robot in the V-REP software environment provides inverse and forward kinematic equations, it is sufficient to develop the communication interface software using Lua script language to simulate robot movements.

III. DESIGN

In this study, the software developed on the PC is as below

- Kinect gesture recognition : Retrieval of commands from the user : Receives commands from the user using the command library.
- Kinect-V-REP Communication : (.dll) : Transfer of commands to the V-REP tool, transfers commands with remoteApi.
- V-REP Lua Scripts : GUI : ABB IRB360 robot model moves the chess pieces on the screen by taking commands via remoteApiCommandServer and threads.

The software architecture of the implemented system is given in Fig. 3.

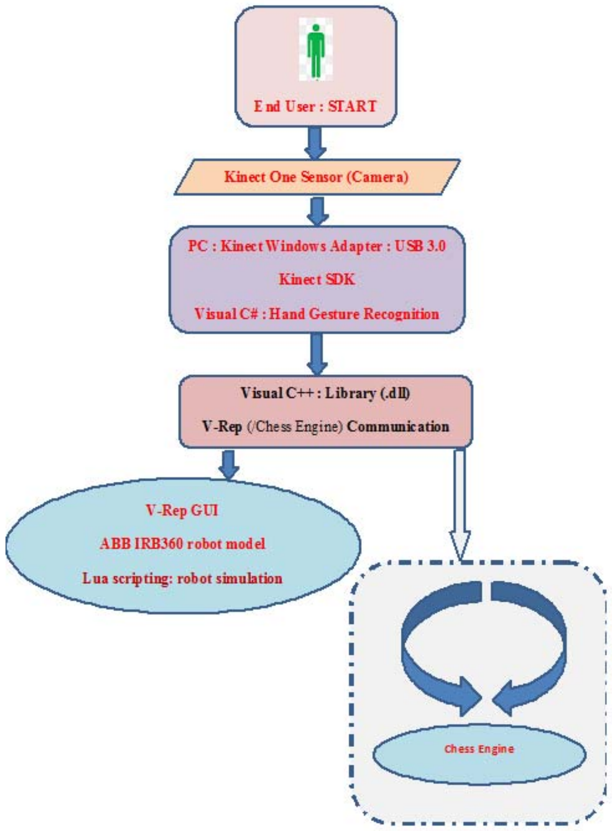

End User : START
Kinect One Sensor (Camera)
PC : Kinect Windows Adapter : USB 3.0
Kinect SDK
Visual C# : Hand Gesture Recognition
Visual C++ : Library (.dll)
V-Rep (/Chess Engine) Communication
V-Rep GUI
ABB IRB360 robot model
Lua scripting: robot simulation
Chess Engine

Fig. 3. Software architecture

IV. SOFTWARE IMPLEMENTATION

In this paper, as software for image recognition, classification of hand movements, simulation and communication has been implemented. We can classify the developed software in three different modules.

A. *chess_v_rep_kinect.ttt :*

The V-REP screen consists of the following parts :

- Lua scripts
- Delta robot model
- Models of chess pawns[7]
- Chessboard model[8]
- Remote Api Command Server

The simulation on V-REP can also work independently. It can also be controlled from the outside via Remote API.

B. *try_v_rep.dll remoteApi.dll:*

This library is written in C++ and used to transfer commands from Kinect to V-REP. It provides integration between V-REP and Kinect.

C. *KinectHandTracking.sln (Customized)*

This program allows you to determine the status of the hands by following your fingers and thumb. Existing program source code has been modified and adapted to the project. Functionality has been expanded as follows :

- A new hand gesture recognition library was created and added to the project.
- The remoteApi library was written in C ++ and added to the project developed using C #.

V. Methods

A. *Recognizing hand movements*

In modern life, human-computer interaction attracts a lot of interest from researchers. Microsoft's Kinect sensor is good at body tracking, but the latest release, the Kinect 2 sensor, has more capable for finding human joint positions [9-14]. The fundamental approach to gesture recognition is to determine the skeleton's joint points. Basic gesture detection depends on some pre-defined set of conditions, known as the result set. If the performed action is matched with the result set, we can say that the user has performed a certain gesture, otherwise not. We can recognize hands and thumbs. We can analyze and use hand situations.

- We can use the Kinect sensor with color, depth and infrared components and capture the image.
- The human body track shows us how to display human body joints.

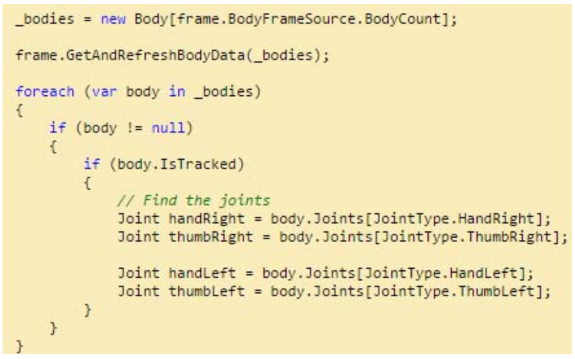
We have developed a new hand gesture recognition library for this project (Table I). This library allows users to control the robot on the GUI and move the chess figures using their hands instead of a mouse or keyboard

TABLE I. Hand Gesture Recognition Library

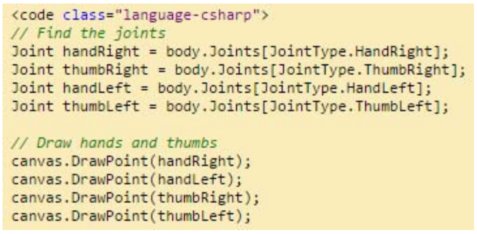
Hand Gesture Library		
COMMAND	***DESCRIPTION***	***DETAIL***
Start	Start	Hands Open Over Head
Stop	Stop	Hands Closed In Front of the Body
Go Left	Move a Cell Left	Left Hand Open Over Head
Go Right	Move a Cell Right	Right Hand Open Over Head
Go Up	Move a Cell Up	Left Hand Closed Over Head
Go Down	Move a Cell Down	Right Hand Closed Over Head
Pick the Chesspiece	Pick the Chesspiece	Left Hand Open, Right Hand Closed
Drop the Chesspiece	Drop the Chesspiece	Left Hand Open, Right Hand Open

B. *Hand tracking and hand categorization in video image*

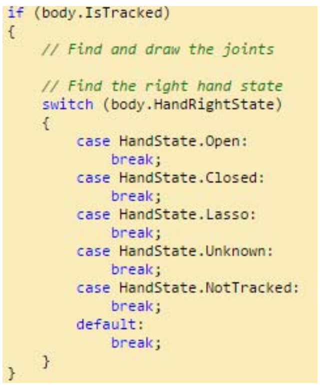
Using an event, a class can notify its client when something happens. Whenever a particular status of the gesture is identified, it notifies back to the application and the end user. This is shown in the following code block:Finding the position of a hand or a thumbnail is now like finding the position of each joint. Below we see how we can get each hand position [14]



```
_bodies = new Body[frame.BodyFrameSource.BodyCount];

frame.GetAndRefreshBodyData(_bodies);

foreach (var body in _bodies)
{
    if (body != null)
    {
        if (body.IsTracked)
        {
            // Find the joints
            Joint handRight = body.Joints[JointType.HandRight];
            Joint thumbRight = body.Joints[JointType.ThumbRight];

            Joint handLeft = body.Joints[JointType.HandLeft];
            Joint thumbLeft = body.Joints[JointType.ThumbLeft];
        }
    }
}
```



```
<code class="language-csharp">
// Find the joints
Joint handRight = body.Joints[JointType.HandRight];
Joint thumbRight = body.Joints[JointType.ThumbRight];
Joint handLeft = body.Joints[JointType.HandLeft];
Joint thumbLeft = body.Joints[JointType.ThumbLeft];

// Draw hands and thumbs
canvas.DrawPoint(handRight);
canvas.DrawPoint(handLeft);
canvas.DrawPoint(thumbRight);
canvas.DrawPoint(thumbLeft);
```



```
if (body.IsTracked)
{
    // Find and draw the joints

    // Find the right hand state
    switch (body.HandRightState)
    {
        case HandState.Open:
            break;
        case HandState.Closed:
            break;
        case HandState.Lasso:
            break;
        case HandState.Unknown:
            break;
        case HandState.NotTracked:
            break;
        default:
            break;
    }
}
```

C. *Remote Api*

The remote API is part of the V-REP API. V-REP uses a remote API allowing to control a simulation from an external application or a remote hardware (e.g. real robot, remote computer,e.t.c)[15].

Remote API functions interact with V-REP via socket communication[15-17].

The Remote API functionality has two separate parts :

- The Client Side (your application) : The Remote API on the client side can be implemented with many different programming languages. Currently following languages are supported : C/ C++, Python, Java, Matlab, Octave, Urbi and Lua .
- The server side (V-REP) : The server-side Remote API has been implemented with a plugin that is loaded by V-REP by default .

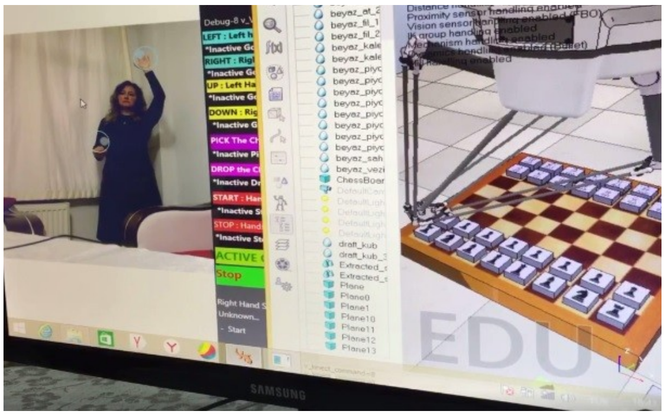
The functions used in communication are given below :

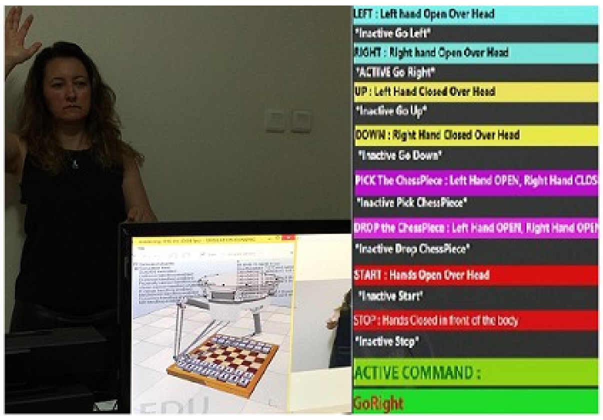
- simxSetIntegerParameter : It is used inside try_v_rep.dll (remoteApi.dll) on the client side. The function developed in library is called from within C#.
- simGetInt32Parameter : It is used used on the server side in V-REP to receive commands on the client side.

D. *User Interface / V-REP*

In this project, V-REP was used as GUI (Figure 6). chess_v_rep_kinect.ttt : The V-REP screen consists of the following parts :

- Lua scripts
- Delta robot model (ABB IRB360 robot model)
- Chessboard model
- Chess pieces model
- remoteApiCommandServer




LEFT : Left hand Open Over Head
"Inactive Go Left"
RIGHT : Right hand Open Over Head
"ACTIVE Go Right"
UP : Left Hand Closed Over Head
"Inactive Go Up"
DOWN : Right Hand Closed Over Head
"Inactive Go Down"
PICK The ChessPiece : Left Hand OPEN, Right Hand CLOS
"Inactive Pick ChessPiece"
DROP the ChessPiece : Left Hand OPEN, Right Hand OPEN
"Inactive Drop ChessPiece"
START : Hands Open Over Head
"Inactive Start"
STOP : Hands Closed in front of the body
"Inactive Stop"
ACTIVE COMMAND :
GoRight

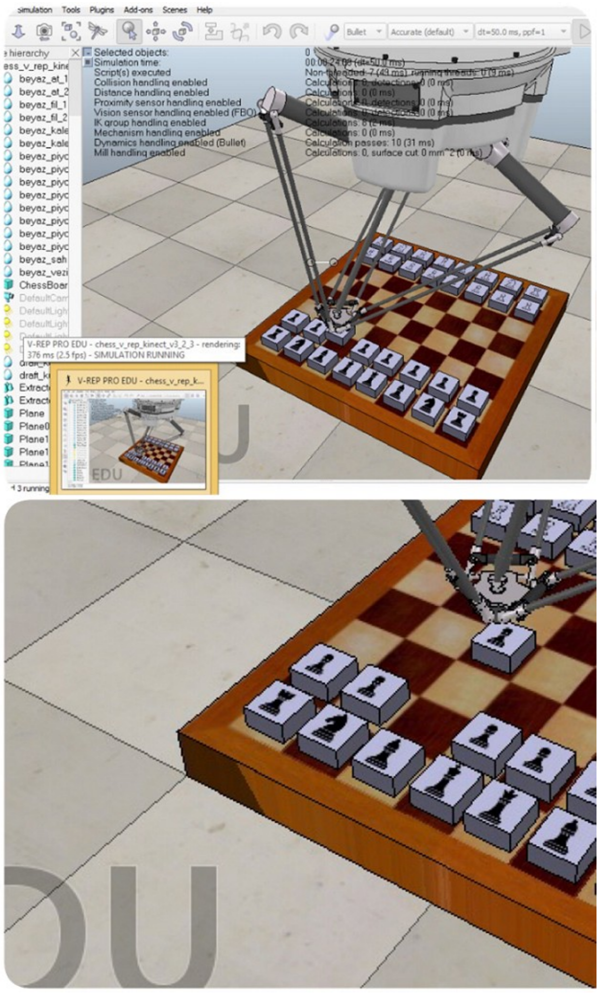


Fig. 4. Developed software interface

VI. CONCLUSIONS

In this project, the infrastructure for controlling the delta robot with hand gestures has been developed. With image processing and recognition, meaningful hand movements made by the player to play chess are sent to the V-REP with the remote API that will communicate with the robot simulator. Algorithm was tested according to Reti and English opening.

References

[1] S. Dalkıran, Satranç Eğitim Metodu, İnkilap Kitapevi, 1990

[2] C. Matuszek, B. Mayron, R. Aimi, M.P. Deisenroth, L. Bo, R. Chu, M. Kung, L. LeGrand, J.R. Smith ve D. Fox, "Gambit: A Robust Chess-Playing Robotic System", Proc. of IEEE International Conference on Robotics and Automation (ICRA), pp:4291 - 4297, 2011.

[3] https://www.chess.com/news/kramnik-i-dont-consider-myself-a-genius-7502 accessed on December 2016

[4] https://clevergames.wordpress.com/2009/05/02/board-games-history-senet-the-egyptians-ancestor-of-chess/ , accessed on December 17, 2016

[5] http://web.onetel.net.uk/~johnrampling/turk.html, , accessed on December 17, 2016

[6] http://www.coppeliarobotics.com/index.html, accesed on 20 May 2016

[7] Chess Piece Figures :

https://www.google.com.tr/search?q=chess+figures&biw=1024&bih=673&tbm=isch&imgil=7WrK7_MYbuG7fM%253A%253BAOVoDpHBofFZ2M%253Bhttps%25253A%25252F%25252Fwww.colourbox.com%25252Fvector%25252Fgrunge-illustration-of-king-and-queen-chess-figures-vector-vector-6575915&source=iu&pf=m&fir=7WrK7_MYbuG7fM%253A%252CAOVoDpHBofFZ2M%252C_&usg=__FX3-SvBkMqZ77yx5lCeRZf5YRHU%3D&ved=0ahUKEwj8k_Lzsd_MAhUDXCwKHR9pAVAQyjcIOg&ei=lSc6V7z1DIO4sQGf0oWABQ#imgrc=7WrK7_MYbuG7fM%3A, accessed on March 17, 2016.

[8] Chess Board Picture :

https://www.google.com.tr/search?q=chess+board&biw=1024&bih=673&tbm=isch&imgil=o9jMoigr30eZFM%253A%253Bzi6pL7dH7dY8kM%253Bhttp%25253A%25252F%25252Fwww.chesshouse.com%25252F16in_Wooden_Chess_Board_p%25252Fchw31.htm&source=iu&pf=m&fir=o9jMoigr30eZFM%253A%252Czi6pL7dH7dY8kM%252C_&usg=__wW8uVoHLmI05AFxJbzNWt0WVLqE%3D&ved=0ahUKEwiQ58PHst_MAhWHBiwKHRpLDOQQyjcIJQ&ei=RCg6V5C1JYeNsAGalrGgDg#imgdii=o9jMoigr30eZFM%3A%3Bo9jMoigr30eZFM%3A%3BE_5DYf6Ws7scsM%3A&imgrc=o9jMoigr30eZFM%3 , accessed on March 17, 2016.

[9] K. T. Tran, S. Oh, “A Hand Gesture Recognition Library for a 3D Viewer Supported by Kinect’s Depth Sensor”, Dongguk University, South Korea, 2014.

[10] M. Tang, “Recognizing Hand Gestures with Microsoft’s Kinect”, Stanford University, 2011.

[11] Z. Ren, J. Meng, J. Yuan, “Robust Hand Gesture Recognition with Kinect Sensor”, Nanyang Technological University.

[12] Y. Yin, R. Davis, “Real-Time Continuous Gesture Recognition for Natural Human-Computer Interaction”, MIT, 2014.

[13] M. Asad, C. Abhayaratne, “Kinect Depth Stream Pre-processing for Hand Gesture Recognition”, City University London, The University of Sheffield, 2013.

[14] http://www.codeproject.com/Articles/747799/Kinect-for-Windows-version-Hand-tracking , accessed on April 2016.

[15] V-REP Remote Api :

http://www.coppeliarobotics.com/helpFiles/en/remoteApiOverview.htm, accessed on March 2016

[16] V-REP Pro Edu : blobDetectionWithPickAndPlace.ttt accessed on March 2016

[17] V-REP Pro Edu : remoteApiCommandServerExample.ttt, accessed on March 2016