A Model for Real-Time Control of Industrial Robots

Fereshteh Shahmiri [a] and [b] Jeremy Ficca

[a] School of Architecture, Georgia Institute of Technology, USA
[b] School of Architecture, Carnegie Mellon University, USA
E-mail: [a] Fereshteh_shahmiri@gatech.edu, [b] jficca@cmu.edu

Abstract –

In this study a user-friendly model is developed for architects and designers without advanced programming skills who need to operate their projects in a real-time workflow from parametric design to robotic fabrication. Based on the implemented network capabilities in proposed model, not only one but also multiple robotic arms can interact and collaborate with design model data. In the network features of this model, a mediated interface is inserted between design software and robot interface that lets the user choose different parametric modeling software as the origin of the workflow. A practical experiment is developed in this research to validate the accuracy of the proposed model.

Keywords –

Real-time, Networking, Socket Communication, TCP/IP, UDP, Robotic Fabrication

1 Introduction

Many reasons necessitate running the live operation of industrial robots in design to fabrication processes. First, regarding the education of architecture, the live control of industrial robots can lead to creative products with exploration of unpredictable features [1][2]. Second, it is beneficial in the detection of potential errors existing in workflows and their correction inside the process with a fast and strong feedback loop originating from design action and fabrication reaction [3]. Moreover, some existing design variables in parametric modeling are not fixed values. So the implementation of the fabrication workflow has to be flexible enough to be responsive to the instantaneous changes of those variables while robot is operated [4]. Finally, there are situations in which multiple robots have to interact with one unique design model database, in these situations, robots' synchronization will be problematic if they are not operated real-time.

The present paper responds to key questions by addressing some challenges in the live control of robotic arms such as connection and compatibility of different interfaces, programming languages, connectivity between software and hardware layers and the complexity of networking. Those key questions are: first, what are the issues and interfaces involved in a real-time robotic workflow? Second, how is the implementation of a user-friendly application for such a workflow? Third, how is a real-time data streaming to both a virtual and an actual robotic arm? At the end, this research claims that the developed model is applicable in many live operations of industrial robots relative to the architecture applications.

2 Research Background

Many different researches and projects are conducted around the concept of live controlling of industrial robots in the area of architecture. A common software relative to parametric modeling in architecture is grasshopper [5]. Different adds-on for robotic fabrication is developed for this environment such as HAL [6] and KUKA|prc [7]. They presented different components to easily program industrial robots where the user generates a movement path with common CAD software and the programming tool generates the robotic execution program [2]. They have specifically developed components to generate and control a real-time workflow. However, in some cases, users need something different in compare with what already exists in those mentioned grasshopper add-ons. First, users need a clear understanding about the process that could simply demonstrate how is the implementation of a real-time workflow. Grasshopper components seem as black boxes to the users. Error checking and debugging is complicated when they are involved in design processes. This study tries to demonstrate how is the communication and networking in a live operation of the industrial robot. Second, mentioned plug-ins are developed just for grasshopper. However, all users don't necessarily work with this software. This study has pipelined the workflow among three interfaces: first, grasshopper as a design software, second, a middle layer application written in python as a mediator between design and fabrication software, third, RobotStudio [8]

and IRC5 robot controller [9] as fabrication platform. It needs to be noted that although in this study grasshopper is considered as the default design modeling environment, the proposed model is flexible enough to detach the first interface and insert other parametric modeling software into the pipelined workflow as long as it is able to support network constraints.

3 Research Scope

Controlling the industrial robots and robotic fabrication are entwined in some topics such as robot mechanism, kinematic solvers, robot end-effector set up, robot programming languages, workspace analysis, positioning and motion programming. Dealing with these topics is required in all types of robotic workflows including live operations. However, the concentration area of this research is mainly on networking issues among existing interfaces.

As figure 1 illustrates, in the scope of this research, the real-time workflow is originated from grasshopper to RobotStudio for a live control of virtual robot and to IRC5 robot controller for real-time running of actual robotic arm. In order to streamline the workflow, the technical knowledge is restricted to coding in RAPID (Robot Application Programming Interface Dialog) [10] and Python script, networking in LAN (Local Area Network) and WAN (Wide-Area Network), socket programming for using UDP (User Datagram Protocol) and TCP (Transport Control Protocol)/ IP (Internet Protocols).

This study is conducted in Carnegie Mellon University; digital Fabrication lab (dFab) [11] and Computational Design Lab (CoDe Lab) [12] in 2014. Involved industrial robots are ABB IRB-4400 & IRB-6640. It is an ongoing research project and will continue at Georgia Tech; Digital Fabrication lab, with Kuka industrial robots.

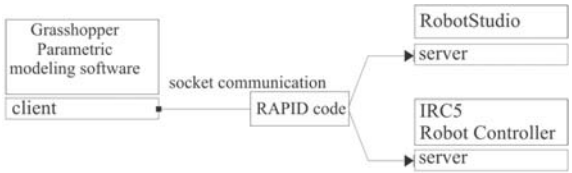

Figure 1. Focus areas in the research scope

4 Methodology

This research project is organized in two levels. In top level, it establishes a robust and reliable method of live communication among design and fabrication interfaces in order to run an industrial robot. To achieve such a reliable real-time workflow, client-server applications are developed. In these applications, UDP and TCP protocols are used for data streaming in LAN and WAN. Implemented method has the potential of operating not only one but also multiple robotic arms simultaneously. In second level, developed model of live controlling the robots, is applied to a practical experiment to validate the accuracy of the research result. This experiment aims to deal with a common challenge in the area of architecture for designing the surface patterns.

4.1 Create a Client-Server Application

In order to operate a live communication between design software and robot software and controller, data is streamed from grasshopper to RobotStudio and IRC5 robot controller. For data streaming, a Client-Server application is required in which grasshopper acts as a client and RobotStudio as a server. Such a Client-Server application is necessary to establish a network between two existing platforms. The connection between client and server in this network is either over LAN or WAN such as Internet. This network service and data streaming is provided by transport layer which comprises two types of protocols; UDP and TCP. Socket programming is used as an interface for these protocols. Figure 2 demonstrates how sockets are created in mentioned client-server application. As a result, it is essential to determine the type of network and protocol to stream data from design to fabrication platform.

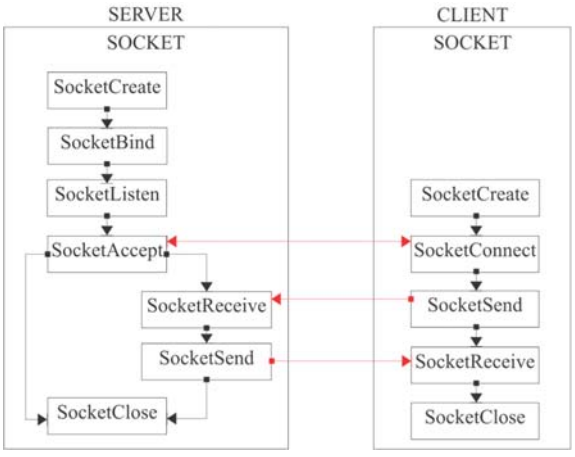

Figure 2. Socket communication between server and client [13]

4.2 UDP and TCP Protocols

Choosing either TCP or UDP as a transport layer is

depend on each existing platform. It is possible that some design model data sources and environments like grasshopper don't support socket programming over TCP protocol. Grasshopper can stream data over UDP [14]. UDP is a protocol that sends independent packets of data, called datagrams, from one computer to another with no guarantees about arrival and sequencing [15]. Missing and ordering problem of datagrams is mostly because of the network delays. As long as, data transmission is on a LAN, network delays are insignificant and missing and ordering the data won't be an issue. From the other side, IRC5 robot controller is only programmed to work with TCP. TCP is a connection-oriented protocol that provides a reliable flow of data. Since two existing platforms are not able to communicate by a mutual protocol, third interface is inserted in the workflow as a mediator.

4.3 Python Script: A Mediator Interface

A mediator interface between design software and robot software and controller is developed in python. According to figure 3, this middle layer interface aims to create two separate networks and acts both as a server and a client. In first network, it is a server in LAN for receiving data over UDP from grasshopper and in the second one, it is a client to stream data over TCP/IP to RobotStudio and IRC5 Robot Controller.

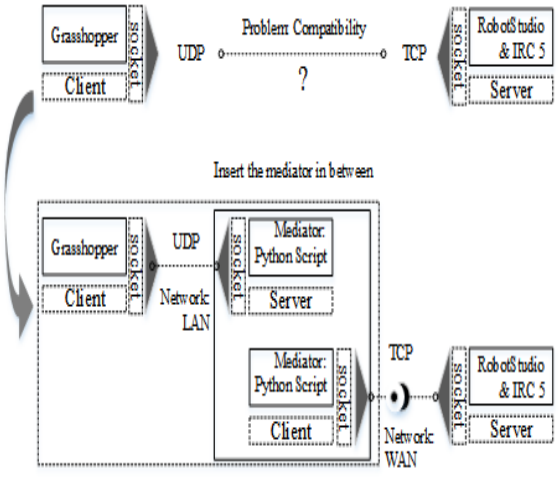

Figure 3. socket programming in mediator

There are key benefits of inserting such a mediator in the workflow. It aims users to customize the workflow depend on their own demands and makes data streaming to fabrication interface flexible enough to be independent of the design software. That means users have freedom in choosing different design software as long as the selected software is able to support either UDP or TCP protocols. Even if the software would support data streaming over TCP/IP, keeping the middle layer interface assists those users who are not interested in coding in RAPID. Figure 4 and 5 illustrate how python script and RAPID code interact with each other via programmed sockets as client and server.

```
import socket
Import time
HOST1='localhost'
#after receiving data by localhost on UDP, data
should be sent by host IP over TCP connection.
PORT1 =8000
S = socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)
S.bind(HOST1, PORT1)
S1 = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
// if stream to multiple robot, second HOST and
PORT is assigned
S1.connect(HOST1,PORT1)
while 1:
        data, addr = s.recvfrom(1024)
        s1.sendall(data)
        time.sleep(2)
```

Figure 4. Insert middle layer interface to stream data between design and fabrication platforms

```
MODULE RobTalk
VAR socketdev ServSock; !Socket for server
VAR socketdev InSock;!Socket for client
VAR string received_string;
VAR string stRecieved; ! String from socket
VAR string clientIP;
VAR bool listen:= FALSE;
PROC RobComm()
    SocketCreate ServSock;
    S = socket.socket(socket.AF_INET,
    socket.SOCK_STREAM)
    SocketBind ServSock, "128.2.109.20",4997 ;
    SocketListen ServSock;
    SocketAccept ServSock, InSock;
    WHILE TRUE DO
        !variable data inserted here
        SocketRecieve InSock
        \str:=stRecieved;
        TPWRITE stReceived;

        ENDWHILE
        ENDPROC
        ENDMODULE
```

Figure 5. socket programming in RAPID code

4.4 Communication with Multiple Robots

There are different architecture and design applications in which multiple robotic arms are required in doing a specified task. According to the present research methodology, the middle layer interface is able to interact with multiple servers. In this case, multiple WAN networks relative to the number of robots are established. According to figure 6, by assigning unique IP addresses and port numbers to each server, manipulated data in design model can interact with each robot independently. Figure 8 depicts how two ABB robots interact with a common data source simultaneously.

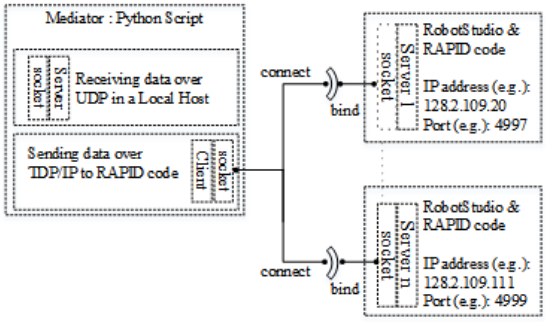

Figure 6. Data streaming to multiple servers

5 Result and Discussion

Figure 7 is the proposed model for a live control of robots and illustrates how network issues and real-time data streaming is solved between grasshopper, RobotStudio and ABB industrial robots. Based on the developed model, design and fabrication software have the potential to be customized according to the users' decision. There are key points in this developed model that users have to consider. First, for those design software without the capability of communication over TCP/IP like grasshopper, UDP is an option as long as first and second interfaces remain in LAN. If another design software support TCP protocol, the mediator interface could be removed and this model remains valid in connection between first and last interfaces. In this case, client socket that is implemented in the first interface has to be modified relevant to TCP protocol requirements. In the future, this research will develop the model to support both TCP and UDP in the first interface. Second, sockets are programed in different ways. If transmitting additional acknowledgement is required between interfaces, it is possible to program the sockets as both client and server in each interface. This optional concept will be added to the developed model in the future. Third, there are two advantages of inserting the middle layer interface as a mediator in the workflow. It creates separate sockets and networks to make connections over both UDP and TCP protocols. It also, can be a platform for the implementation of an external memory when users need to deal with large data set.

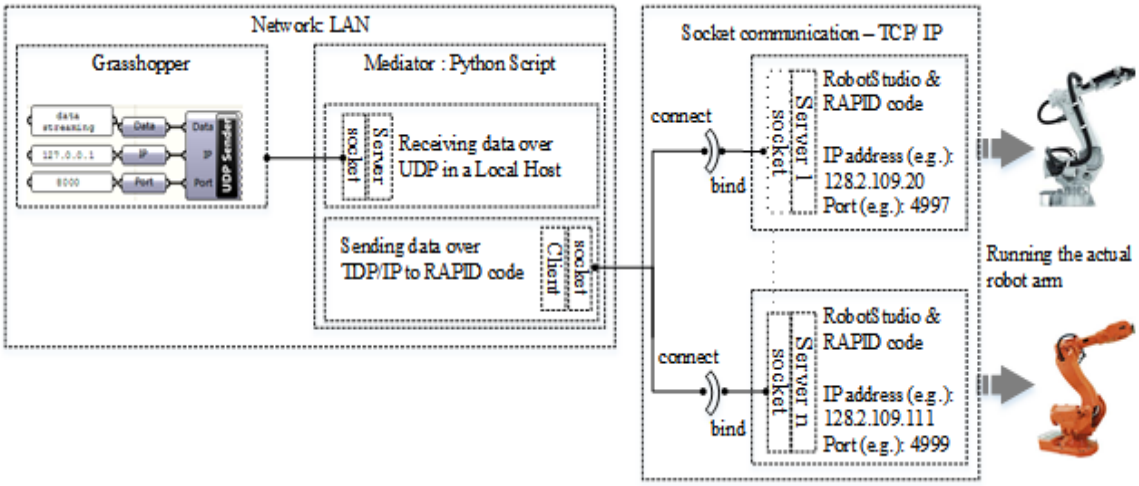

Figure 7. a diagrammatic view of final application

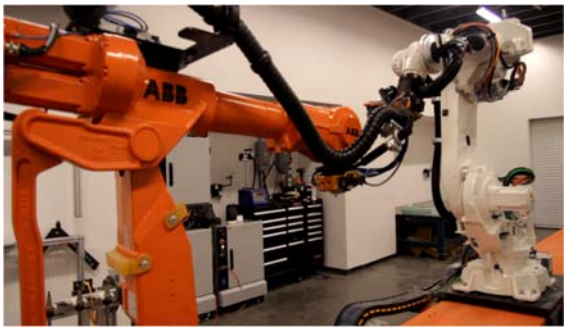

Figure 8. based on the proposed model, two ABB robots interact real-time with one unique design model database. (CMU dFab)

6 Validation

To validate the developed model and methodology, a practical experiment is conducted in order to simulate a variety of unpredictable patterns on a surface. For a long time designing the surface patterns has been a challenging issue for architects and designers. Specifically, when they are seeking nature patterns with many variable inputs that influence on the patterns formation. In this experiment, patterns are generated from water flow particles and tracing on a freeform surface. This simulation is done in Kangaroo [16]. The real-time operation of the robotic arm applies instantaneous changes in virtual model to the physical prototype synchronously. As figure 9 illustrates, variable inputs are particles' position and their non-linear velocity in each defined time frame. Data streaming in this step is done by using "UDP sender" component in Firefly [17]. To deal with such a large data set, an external memory out of grasshopper is required. This external memory is created in the mediator interface in the proposed model in figure 7. In this memory, data is saved and sifted according to the relation between patterns resolution and memory size. This manipulated data is streamed to RAPID code and led to the live operation of the robot. Figure 10, 11 and 12, demonstrates how this experiment is developed.

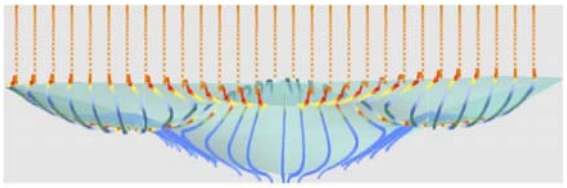

Figure 9. Particles have non-linear velocities in each position.

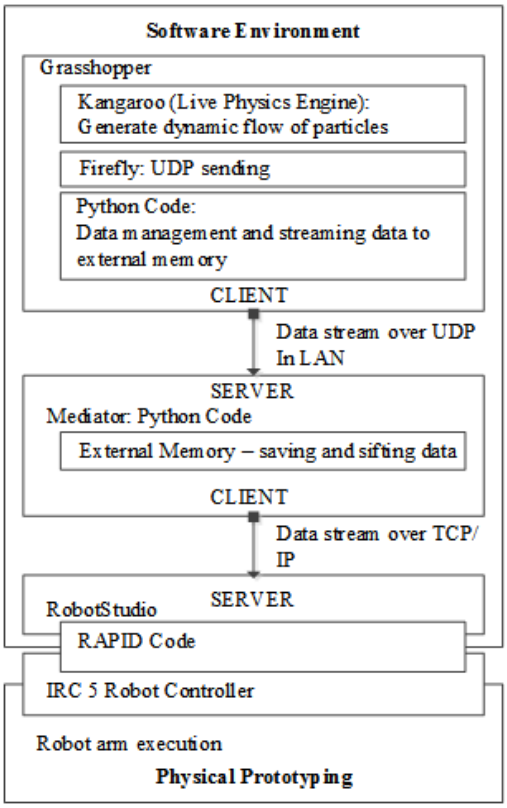

Figure 10. workflow for a live simulation of water flow patterns on freeform surfaces.

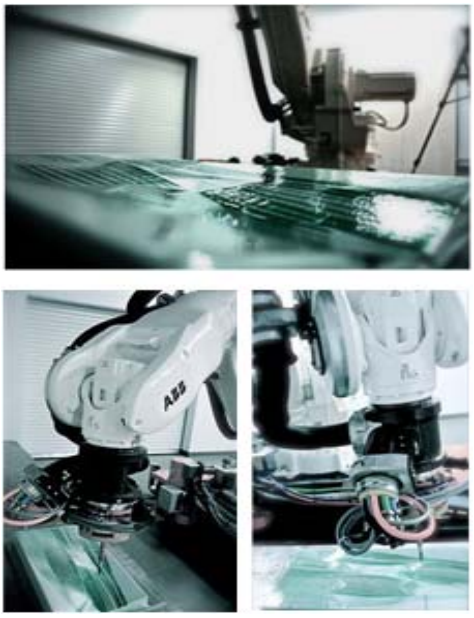

Figure 11. Physical prototyping – real-time workflow synchronized with virtual model.

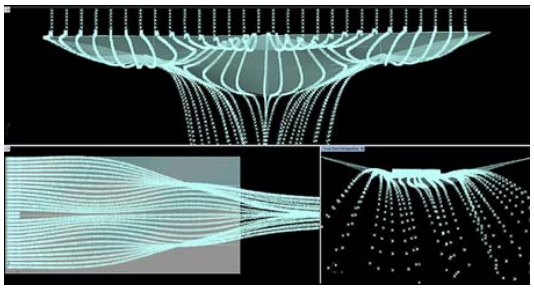

Figure 12. Water flow particles' simulation in virtual model

7 Summary and Conclusion

This research has introduced the existing challenges in a real-time control of ABB industrial robots via an architecture software like grasshopper. It has established a robust and reliable method of live communication among design and fabrication interfaces. To achieve such a live communication, technical issues related to the networking and socket programming are solved. In the proposed methodology, client-server applications are developed. UDP and TCP protocols are used for data streaming in LAN and WAN and the possibility of live control of multiple robot arms is tested and validated. This study has verified the accuracy and efficiency of the methodology by presenting a practical experiment.

There are advantages in the proposed methodology. First, connection and compatibility of the interfaces are open which provides a clear understanding about the workflow for the users. Second, the users also can choose grasshopper as with any other design software as long as it could support network constraints. The methodology has the potential to apply different types of industrial robots into the workflow.

The main conclusions drawn in this research are to investigate several issues. first; developed model has to be customized for other types of industrial robots such as Kuka. Second, the integration of the proposed methodology in to the other devices such as vision system and sensory devices is part of the future development of the research. Third, the more exploration of real-time applications is required. Regarding the developed project, one helpful application is to explore how different patterns generated from a particle system in virtual space can lead to a variety of unpredicted but adjustable patterns in physical model. Variables like different shapes of freeform surfaces and the density of the particles are some of those explorations. Such differentiation between virtual and physical models would be created and controlled by different sensors applied to the design to fabrication process.

References

[1] Yoshida H, "Robotic aggregation of materials with unpredictable geometry". On-line: http://www.dfab.ch/portfolio/robotic-aggregations-of-materials-with-unpredictable-geometry/, Accessed Jun 2016

[2] Elashry K, and Glynn R. "An Approach to Automated Construction Using Adaptive Programing." *Robotic Fabrication in Architecture, Art and Design 2014*. Springer International Publishing: 51-66, 2014.

[3] Braumann J, and Brell-Cokcan S. "Real-time robot simulation and control for architectural design." *30th eCAADe Conference* pages 469-476, Prague. 2012.

[4] Vasey, L. Maxwell, I. Pigram, D. "Adaptive Part Variation; A Near Real-Time Approach to Construction Tolerances." *Robotic Fabrication in Architecture, Art and Design 2014.* Springer International Publishing: 291-304, 2014.

[5] Rutten, D. "A graphical algorithm editor integrated with Rhino's 3-D modeling tools". On-line: http://www.grasshopper3d.com/, Accessed Jun 2016

[6] Schwartz, T. "Application of Robot Control and programming in architecture and construction industry". On-line: http://www.hal-robotics.com/

[7] Accessed Jun 2016

[8] Braumann J, and Brell-Cokcan S. "Parametric Robot Control and Simulation". On-line: http://www.robotsinarchitecture.org/kuka-prc, Accessed Jun 2016

[9] "ABB's fifth generation robot controller". On-line: http://new.abb.com/products/robotics/controllers/irc5 , Accessed Jun 2016

[10] ABB Robotics, Application Manual, PC SDK. on-line:https://library.e.abb.com/public/124d6b59313ed85fc125793400410c5b/3HAC036957-en.pdf , Accessed Jun 2016

[11] "Off-line programming software for ABB Robots". On-line: http://new.abb.com/products/robotics/robotstudio , Accessed Jun 2016

[12] Carnegie Mellon University, Digital Fabrication

Lab. On-line: http://soa.cmu.edu/dfab/ , Accessed Jun 2016

[13] Carnegie Mellon University, Code Lab. On-line: http://code.arc.cmu.edu/ , Accessed Jun 2016

[14] Szabolcs, C. " Comparison between two methods of robot control in the production of prismatic parts". *nonconventional technologies review16.2* (2012).

[15] Afsari K., Swarts M.E., Gentry T.R. Integrated generative technique for interactive design of brickworks, *Journal of Information Technology in Construction (ITcon*), Vol.19, pg. 225-247, 2014.

[16] Xue, M. and Zhu, C. "The socket programming and software design for communication based on client/server." *Circuits, Communications and Systems*, 2009. PACCS'09. Pacific-Asia Conference on. IEEE, 2009.

[17] Piker, D. "Kangaroo; live physics for Rhino/grasshopper" On-line: http://api.ning.com/files/yPjjd3IjSzDhL3dzJkMTSiPHFBFe8m2SxAX-pvUfidJ*2W82Ij0BG-2z*M*LELCkzVSTxX2UToz-sJgBVHa6MnPmf-Gvdppk/KangarooManualGrasshopperversion_.htm , Accessed Jun 2016

[18] Payne, A. and Johnson, J." Firefly; connection between grasshopper and Arduino." on-line: http://www.fireflyexperiments.com/ , Accessed Jun 2016