CIRA—An Architecture for Building Configurable Immersive Smart-rooms

Rahul R. Divekar[#1], Matthew Peveler[#2], Robert Rouhani[#3], Rui Zhao[#4], Jeffrey O. Kephart[+5], David Allen[#6], Kang Wang[#7], Qiang Ji[#8], Hui Su[+#9]
[#]*Rensselaer Polytechnic Institute. Troy, NY, USA*
[+]*IBM Thomas J. Watson Research Laboratory, Yorktown Heights, NY, USA*
{[1]divekr, [2]pevelm, [3]rouhar2, [4]zhaor, [6]allend5, [7]wangk10} @rpi.edu
[8]qji@ecse.rpi.edu, {[5]kephart, [9]huisuibmres} @us.ibm.com

***Abstract*—Significant technological advancements in computer vision, natural language processing and other fields of computer science and engineering have made way for us to build new human-computer interfaces like a Cognitive Immersive Room (CIR). The CIR described in this work can "see," "hear," interpret and respond appropriately to a user or a group of users occupying it. However, bringing the state-of-the-art technologies together to respond to different levels of interpretations of multiple input sources under a unified system that can be easily re-configured to suit different purposes is a challenge. The possible use-cases and benefits of such a smart-room are exciting and limitless. To address this sky-high vision of a smart-room, we describe a software architecture, Cognitive Immersive Room Architecture (CIRA), that allows the room to be programmed for multiple use-cases by providing them spatial and contextual intelligence. In this project, given the relationship of the software with the physical attributes of a room, reconfigurability to various physical environments must be kept in mind. By providing resilience to changes in the lower-level devices, the architecture allows flexibility for researchers to enable different multi-modal interactions. The architecture also supports use-case and context-nesting. Thus, this work describes an architecture that enables a smart-room with multi-modal interaction, interpretation and reasoning. The CIRA supports nested contexts, multiple languages and multiple physical environments. We describe four such contexts (use-cases), two languages and two physical environments prototyped using this architecture and describe a pilot user-study.**
Keywords: Immersive Rooms, Smart Rooms, Multimodal Interaction

I. Introduction

Immersive intelligent spaces have been an open research question with many sub-research areas. These sub-research areas are often research groups and labs that work in silos advancing the technology in their specific domains such as computer vision, natural language processing, etc. Thus, there is a need to bring these many research areas together under a multi-disciplinary project in-order to build something that the people in the end-user space can use.

We have thus built a smart-room, comprised of three broad sub-areas linked in a cyclical flow shown in figure 1, that integrates the work of several labs at the university. The first area uses a range of sensors (e.g. microphones, cameras) to perceive and sense what is going on within the room and any human-agents within. The second area deals with interpreting and understanding the data from area one, and use it for planning, reasoning, learning, and Natural Language Processing (NLP) to allow the room to make sense of the input. The third area then takes those actions and displays the results in a rich visual and audio setup via speakers, projectors, etc. and lets the room act in a beneficial manner. The arrows show the flow of the information between these areas.

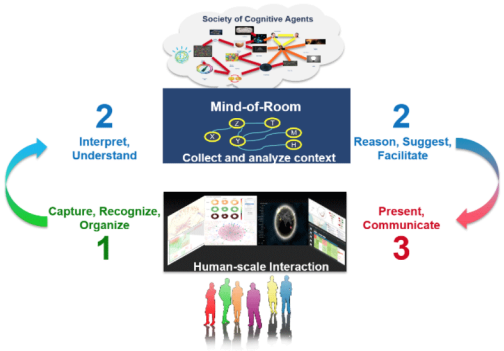

Fig. 1. The Three Areas of CIR

The applications and use-cases of such rooms are wide and range in the spectrum of a room with some voice-activated devices to rooms with hardware controllers and mechanical robots assisting humans. In this work, we define our goal of the interaction with the room as a natural multi-modal conversation which does not require users to use (wear) any intrusive device such as markers, trackers, sensors, etc.
The use-case of the Cognitive Immersive Room (CIR) is in assisting high-stake corporate meetings, classrooms and multidisciplinary group discussion by helping users assimilate large data. The goal of the CIR is to immerse its occupants, interpret natural multi-modal interaction of the occupants with the room with each other and, respond to them appropriately. We concentrate on four broad use-cases that are described here to broadly establish the scope of the CIR to the readers. Su [20] has done a good job to articulate the vision of our lab.

- Assisting decision making in corporate mergers and acquisitions
- Assisting foreign language (Mandarin) learning

- Data exploration using narrative technologies
- Assisting an inter-disciplinary group of people to diagnose cancer

The use-cases, as imaginable, have different requirements from each other, yet are supported by the same system. This work concentrates on enabling these use-cases (also, towards enabling more) and designing an architecture to support them. Thus, we describe the Cognitive Immersive Room Architecture (CIRA)—the underlying software architecture for CIR that provides AI capabilities (**C**ognitive) and can immerse the occupants in a given experience (**I**mmersive).

In the further sections, we briefly review the work done in the smart and immersive rooms research domain. We discuss the use-cases in more detail along with the description of the physical aspects of the room to set-up the readers with the expectations from the architecture in detail. We then dive deep into the architecture showing how the interactions are made possible in the space. We conclude with a pilot study showing positive reactions from a focus group.

II. Related Work

Immersive rooms have been used in various different use-cases involving education [19][15], simulations [12], training, medicine [13], etc. We know from these previous studies about how immersive environments can play an important role in education and visualization by providing additional context to users. As we recognize this, we equip the CIR with projectors and speakers to provide a rich audio-visual output. However, in addition to the output, we must allow humans to interact with the room in a natural fashion. Previous work done by Bolt [4] demonstrated a system that could understand ambiguous commands such as "put that there". The system resolved "that" and "there" by recognizing where a user was pointing through the use of special hardware worn by the user. Pentland [18] discusses components for enabling facial and gesture recognition to a smart room. Ekenel et al. [11] further discuss methods for performing facial recognition within the context of a smart room. Coen [9] and Brooks [5] discuss a similar smart room architecture built atop a distributed control system based on the SodaBot [8] platform and language, criticizing the inflexibility that a monolithic executor would provide them. Interest in these smart rooms however waned and increased attention was put onto "cognitive assistants," such as the work done on the CALO ("Cognitive Assistant that Learns and Organizes) [1] effort under the PAL ("Perceptive Assistant that Learns") [2] program of DARPA. Other examples in the same direction are [17] and [7] both assistants for meeting rooms. However, while utilizing some similar technologies, especially from [6], the focus of this work was more on enhancing the work-flow of single humans as opposed to the group of humans within the smart room.

Additionally, these systems were limited by the available technology of their times. We want to step away from point-and-click interfaces, special hardware, and markers, as well as rigid language and gestures. Therefore, we want to better understand natural human ways of interaction using a minimal set of physical wearable hardware. Both, natural language processing and gesture recognition have come a long way since then and are mature enough to be integrated together. Examples of such state-of-the-art are NLP from Watson Conversation Service[1] and, gesture and head-pose recognition [21]. By incorporating these technologies and techniques, we allow for an increase in flexibility and naturalness in interactions within the smart room.

Other groups have recently tied multiple modality input such as handwriting recognition, facial expression, gesture, etc. and proposed an architecture for sensor fusion. The question of late-fusion vs. early-fusion has been debated in the community and we believe that the decision to use one or the other or a hybrid of the two must lie within the owner of the use-case and our architecture allows that flexibility.

We want to be able to fuse sensor data in such a way that the room will adapt to the addition or deletion of sensors or hardware without requiring too much additional programming or configuration. We attempt to do this and discuss it in the Service Discovery section where adding the same type of sensors is possible without additional programming giving us the leverage to scale the machine-identifiable environment of the room. The CIR is also context-aware and knows whether it is being used for language learning, or decision making, etc. As mentioned earlier, one of the challenges of the system is to recognize one context within the other e.g. decision making in the setting of second language learning. We discuss this challenge in further sections.

We also want to interpret higher-level cognition processes such as agreement in groups, group dynamics, theory of mind of the participants, etc. By making an attempt to understand more than the commands that are directly issued to the room, we believe that we can push the immersive environment towards true intelligence. However, the first step is build the infrastructure (both, hardware and software) and, to identify and tie the multiple modalities together. The rest of the paper concentrates on these aspects of the CIR.

III. Scope and Use-cases

In this section we describe the four use-cases (interchangeably called *applications* or *contexts* hereon) that have already been prototyped using the CIR architecture. This section sets up the readers with context to understand the requirements from the architecture.

A. Mergers and Acquisitions

Mergers and Acquisitions (M&A) allows a group of users to explore and analyze data related to companies. The user can, through speech, ask the system to let them visualize companies and their relationships to some central concept in the data-set, look at the location of the company on an interactive map, ask the system to create a decision table to compare and contrast companies on their attributes, show a company's website, etc.

[1] https://www.ibm.com/watson/services/conversation/

The system can be interacted with speech and gestures. Speech is used to tell the system to perform several functions. The user can use gestures to zoom-in into the maps, point at a certain area on the screen to get additional information, etc. The system can identify the attention of the users through head-pose tracking and interject if everyone is not on board.

B. Mandarin Project

This application aids second-language learning by allowing learners to have a natural conversation with an AI Agent inside an immersive restaurant environment. The users get a feeling of being in a Chinese restaurant where they can speak to the waiter and practice their language skills through completing food-ordering tasks. The system can understand where the users are pointing to on a menu (shown on projector display) in order to make sense of commands like "I want this." The system can also assist in tonal training for words in the mandarin language. Divekar et al. [10] have elaborated on the user interaction challenges they were able to solve using the CIR for this use-case.

C. The Generative and Narrative Agent

Large scale data about the history of the world (e.g. Roman Empire) is available on the internet as structured data. The Generative and Narrative Agent (GNA) can iterate through this graph and weave together a story using connected nodes. The users can look at this generated graph and navigate through it using their voice or by pointing at and selecting a node.

D. The Disease Diagnosis Situations Room

This use-case is aimed at helping a diverse group of people from backgrounds like medicine, data analytics, and computer science to come together and explore the situation of a hypothetical patient and diagnose a disease. The room must assist in communication between the users by filling in the knowledge gaps, utilize machine learning algorithms in analyzing pictures of cells, interpreting group dynamics, and assist them to a faster identification of diagnosis.

IV. Design of The CIR

The use-cases are diverse yet have many things in common. At the very basic level, these use-cases can use the same speech-to-text engine, same output drivers, etc. Some higher levels of integration are also common e.g. if a user registers themselves with the system by giving information about themselves, the details of this user can be leveraged by any of the applications. The highest level of integration is using one application within the other e.g. the GNA agent being used inside the Mandarin Project to tell a story about the history of one of the items on the menu giving the user a cultural insight. In addition, the owners of the use-cases would also like to tie speech, gestures and other interactions with identity and have answers to questions like "who" said "what." The use-case owners would also like to know about higher cognitive-processes such as if there was an agreement (verbal or non-verbal through nodding) during an important decision-making situation, the group dynamics of users, levels of involvement, what the users might individually know or believe, etc. to personalize their applications. This drives the need to have a system that can interpret multi-modal input on different levels. We have elaborated the use-cases in the previous section. In order to fit all use-cases best, we have enabled the following design of the room.

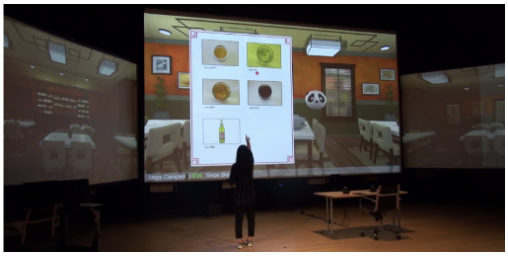

Fig. 2. Three Screen Setup of the Environment (best view in color).

The CIR operates in two physical modes and is configurable to other environments. The first mode is the three screen mode seen in Fig. 2. The overall dimensions of the viewing setup are 15m x 20m. Split into the three screens, there are two of size 4.8m x 2.7m and one of size 7.2m x 4m in the center. The second, more immersive configuration, is the 360 degree panoramic display shown in Fig. 3. The computer generated model for this configuration is shown in Fig. 4 to give readers a better sense of the set-up of the room. The display has a diameter of 12 m and height of 3.8 m. They were both installed in the Experimental Media and Performing Arts Center (EMPAC)[2] at Rensselaer Polytechnic Institute. The panoramic set-up uses special warping software and development efforts have been made to enhance graphical rendering suitable to this scale of an environment

Kinect devices, web-cameras and Pan-Tilt-Zoom cameras track several gestures and head-pose of the users. They are mounted under the screens and look-up with an angle at the users acting as the "eyes of the room." To track the position of the users in the space, an array of Kinects mounted in the ceiling and looking-down at the occupants. The placement of all the sensors are carefully designed to minimize and counteract occlusion and to not be intrusive to the user-experience.

A user can walk into either of the environments, speak to the system and ask it to boot-up a use-case. Once the use-case is booted up, the application takes control of the interaction. However, input to the application is provided by the same underlying architecture. Generally, this involves allowing for at least voice control and, depending on the application and context, the use of gestures.

To address the system with the voice is similar to addressing other voice systems i.e. the user starts the sentence with a "wake-up word". Our system can support multiple wake-up words at the same time. In order to avoid having repeated

[2]http://empac.rpi.edu/

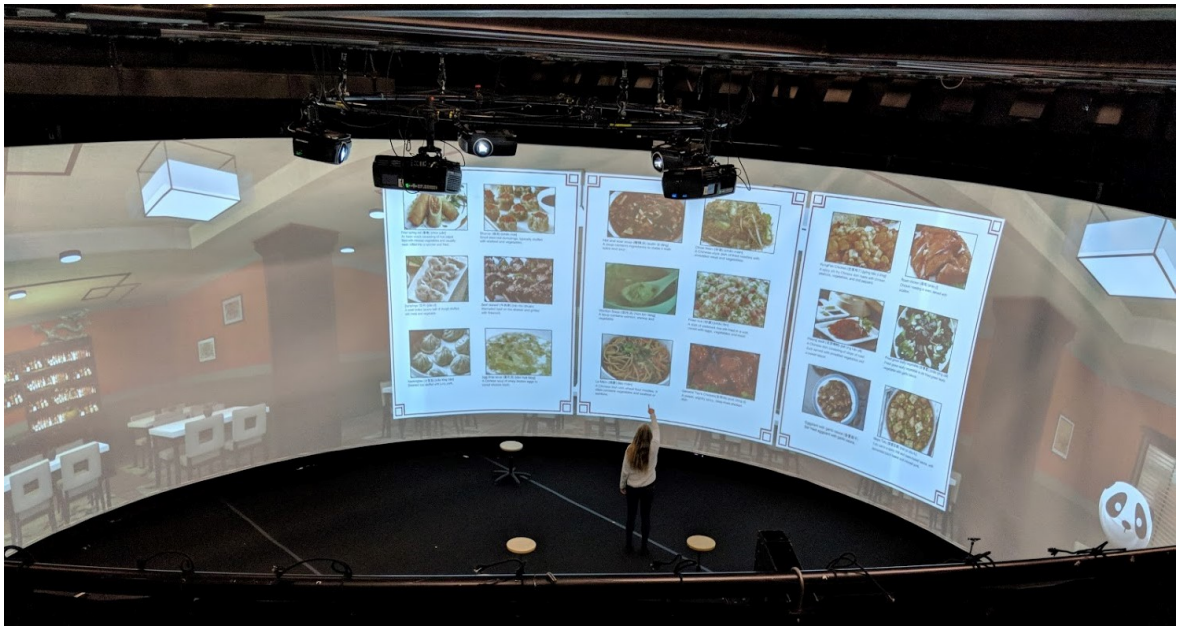

Fig. 3. Bird-eye view of 360 Panoramic Set-up of the Room. Best view in color. Demo Video—youtu.be/lZWtDqhFlAc

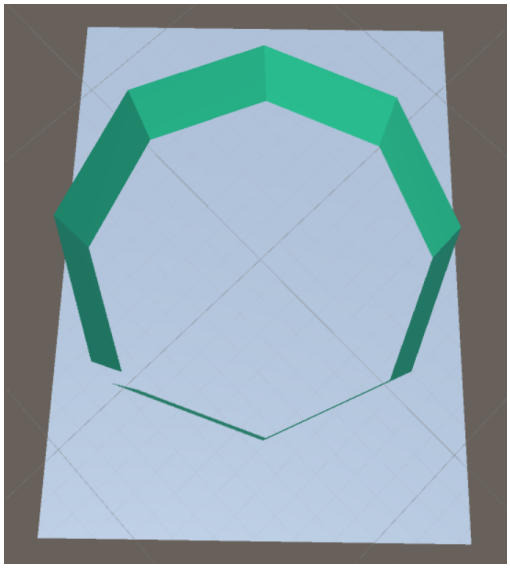

Fig. 4. Computer Representation of the Room

use of the wake-up word, the system continues to accept user commands for a definable number of seconds after the last output from the system.

V. Architecture

The architecture draws inspiration from the theory that an intelligent interaction is made possible by several seemingly basic abilities that come together in a meaningful way [16]. In the community of AI, each of these basic functionalities is also a complex technological challenge that is constantly developing. Thus, the goal of the architecture behind the Cognitive Immersive Room is to integrate several in-progress research technologies, following a modular structure so that other technologies can be added and current ones could be replaced with newer versions. The technologies include speech recognition and natural language understanding for multiple languages, computer vision, brought under the umbrella of spatial intelligence.

We are able to combine multi-modal inputs, such as speech, gesture, and head pose. From these inputs, identify the "complete" meaning of the task the user intends to do or the conversation the users want to speak with the system and each other. The following section explains the different sub-systems (modules) that enable this.

A. Modules of the System

In this section we explain each of the modules and their grouping seen in Fig. 5 which shows modules (gray boxes) and data passed between groups of modules (blue boxes). In the figure, the arrows indicate the flow of the information from one set of modules to the other.

1) ***Input Device Modules****:* This group is the lowest layer of the architecture that encapsulates the hardware and the software closest to it. This group includes the devices that the user directly interacts with either knowingly or unknowingly. It has four modules in it. They are described as follows.

a) Skeleton Tracking: The CIR uses Kinects for skeleton tracking and gesture recognition. Three of them are used to cover the entirety of the room in the case of the three screen setup, and each Kinect has its own module. They are placed on the floor just beneath the screen and look up to the users. These modules are grouped together for readability and are illustrated

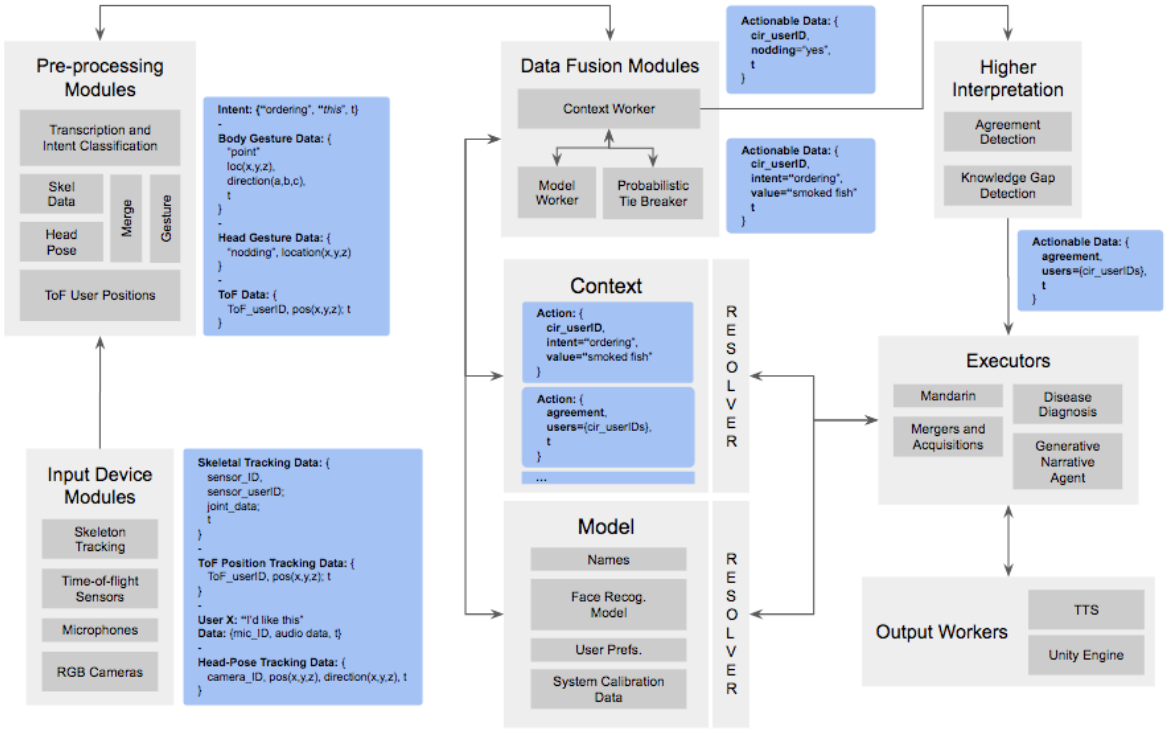

Fig. 5. Multi-modal interaction mapped to Architecture. Blue Boxes show data passed or interaction events. Best view in Color.

as "Skeleton Tracking" in the Fig. 5. The joint position tracked by Kinects are used for gesture recognition.

b) Time-of-Flight Sensors: An additional of six Kinects are mounted in the ceiling to act as time-of-flight sensors where all six act as one array and produce single output. This sensor is used to track the position of each user. Jivani et. al. [14] have described the technology in detail in their work.

We need the time-of-flight sensors to track and identify users. This is because even though skeleton tracking Kinects assign IDs to each skeleton they see which we can use to track individuals with their position in the room, due to occlusion, these IDs could get swapped. As the time-of-flight sensors look down from the ceiling, problems of occlusion from the field-of-view are limited. This means that the tracking ID of each person never changes in this sensor like it can in other sensors (Skeleton Tracking) due to occlusion. Thus, this sensor also helps keep track each individual and their unique IDs in the room. We will see in the next few sections about how this ID can be shared across all physical device sensors. However, since the time-of-flight sensors only get a top-view of the users, they cannot identify gestures and thus we need the front facing skeleton tracking Kinects.

c) Microphones: Six lapel microphones are used to support the multi-user environment. Each microphone is used by one person at a time and can be configured to expect a particular language. A user or the CIR can change the language for any given microphone during an application by asking it to do so. The lapel microphone are the only device a user has to wear to interact with the system. The lab is working on solving this by testing and integrating other ways of acoustic input.

d) RGB Cameras: Pan-Tilt-Zoom (PTZ) and web-cameras are used to capture faces of users in the room. This data is used to identify returning users through facial recognition algorithms and to gauge the head-pose of the user. Multiple cameras are used to cover the intractable space and produce an output stream per device.

*2) **Pre-processing Modules**:* This group of modules takes input from the modules outlined in "Input Device Modules." The modules in this group interpret the multi-modal information as follows

a) Transcription and Intent Classification: The audio captured in the previous stage is sent to a speech-to-text service which transcribes it. The transcribed text is then sent to a conversation service that classifies it into one of many predefined intents. This conversation service allows us to have flexibility with the dialogue. For example, different ways of greeting someone (Hi, hello, how are you doing?) can be classified under one intent (greeting). We use IBM's Watson Assistant [3] to achieve it and details about how it works can be read on their website. This makes programming in the later modules easier because the system needs to check only for a limited number of intents as opposed to checking for a wide range of possible utterances. The architecture supports English and Mandarin Models for speech-to-text and intent classification.

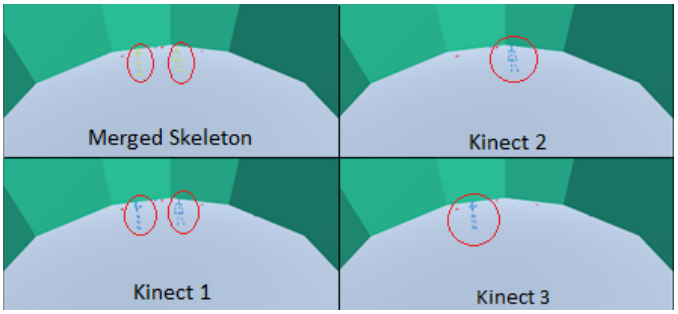

Fig. 6. Merging of Skeletons (best view in color).

b) SkelData and Merge: The skeletal data from the three Kinects that track skeletal movements need to be merged in order to give a unified source of information and to compensate for occlusion in any one of the Kinects. The motivation in doing so is that the individual Kinect has limited field of view. Thus each of them may only provide partial tracking. The three Kinect Modules also maintain their own local IDs to identify the people in the room and do not, by themselves, have a way of sharing these IDs with each other. We perform merging by first transforming the tracking position into the same global coordinate. Then we use weighted average to obtain the merged skeleton position. The weight equals to the tracking status provided by Kinects. Figure 6 shows the merged skeleton (top-left) and outputs from three individual Kinects. As seen, there is no second skeleton in the top-right and bottom-right frames as the view from those Kinects has been occluded, yet frame 1 (top-right) shows the full merged skeleton. An algorithm is then run on the merged skeletal movements in real-time to identify a gesture. Zhao et al. [21] have described this algorithm and its usability in more details.

c) Head Pose: The head pose system takes in information from the cameras and identifies the head orientation of each person. This information can be further analyzed by later stages to identify where or what people are looking at to derive engagement, attention, agreement, etc. Zhao et. al. [21] have described details about how the head pose is tracked in the system.

d) ToF User Positions: We define this as tracking the physical location of each of occupants of the CIR. Position tracking is available through the array of 6 Kinects mounted in the ceiling providing time-of-flight. The module combines the stream of data allowing us to uniquely track the occupants of the room as they move around. Figure 7 shows the position of the participants in the room, with one of them pointing towards a screen. The extending arm and the screen are highlighted in red.

*3) **Data Fusion Modules**:* The context-worker merges data between different types of sensors. Information about *where* a person is, *what* a person is saying, and *what* a person is pointing to comes from multiples sources and needs to be fused. This is done largely by the context-worker, with other modules in this group acting as helpers. The context-worker

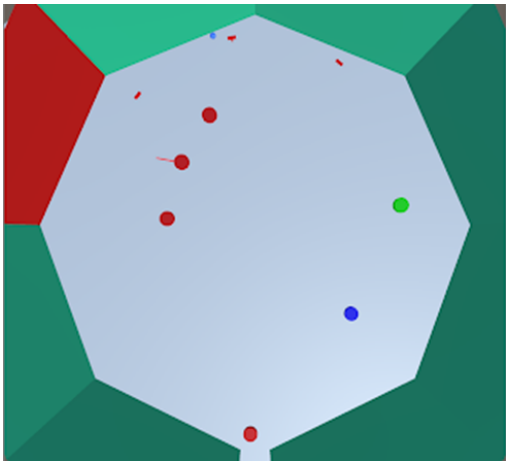

Fig. 7. Position Tracking of Occupants (top-view)

may use the *probabilistic tie breaker* module in cases where there is conflicting information. It uses the *Model (a data registry used here for calibration explained in later sections)* to translate data between different sources. For example, coordinates given by the Kinect module have a different offset from the time-of-flight module coordinates. As such, the context-worker translates each module's data into a global coordinate system that all data outputs map to.

There are multiple ways to fuse data from different sensors especially for multi-modal interaction involving gestures and speech. For now, on receiving certain deictic speech such as "I want this" where *this* is ambiguous, the data fusion module checks if there is a stream coming in from the gesture recognition channel and couples the last read appropriate gesture (in this case, pointing) with this speech. Modularity allows researchers to switch this to a more sophisticated method easily.

A constant overlapping stream of data is processed by the data fusion module as several people in the room could be talking and gesturing without waiting for one other to finish. As an example, a constant stream of gesture data $g_0 ... g_n$ and conversation data $c_0 ... c_n$ is fed to the data fusion module. The context-worker does two things for each g: it stores it and it queries other modules (such as the module that drives the display) to resolve what is being pointed to. If the display module has an interact-able item at the pixel locations used in the query, the context worker associates it with the gesture. When conversation c_i contains certain deictic commands such as *I want this*, the context worker is queried to get the last resolved gesture. In summary and as an example, if at time $t=i$, g_i indicated that *smoked fish* on the screen is being pointed to. These two are associated and stored. If c_i contains sentence *I want this*, the last fully-resolved gesture g_i is looked up and *smoked fish* is associated with *this*. The process is more clear in the section flow of the system.

*4) **Higher Interpretation**:* At this stage, the data is combined and and has all the information it needs and is cross-

referenced with other data sources. The higher interpretation module can now process this data to interpret certain human behaviors. An example is agreement detection where the system takes in data from the head-pose and voice and detect all of the people nodding their head in agreement or verbally agreeing to a question. It finds out the topic in question, the people who agreed to it and the people who disagreed to it.

5) ***Model****:* The *Model* is a data store that has information that does not change every time the system is run. Examples of this information can include facial recognition models (used to recognize users in the system), their names and other personal information. It also stores coordinate system of the rooms and the displays (physical configurations), and is scalable to add more information sources. The *Resolver* provides an API to access this data.

6) ***Context****:* The *Context* is also a data store but is used to store information that is relevant only for a single particular session. For example, it may store all the gestures of the users so that it can be queried during the session. It clears itself between sessions. It could be thought of as a log of all the resolved (actionable) data. The *Resolver* provides an API to access this data. This data must be stored in a way that can be accessed in real-time. In our implementation, we used Redis to cache this data.

7) ***Executor****:* The Executor block is the group of applications that drive the output of the system. These are built by application owners. They act based on the input they get from the context-worker. The executor is responsible for driving the interaction and making decisions about what to show on the display by querying for additional information from context module or web services if needed, or ignoring particular inputs altogether. The executor uses a tree structure to store the states of interactions and models the conversation flow in the system. Upon receiving an *intent* from the context-worker, the executor checks if the corresponding dialogue node is accessible. If the node is accessible, the executor implements the commands written in it. The executor outputs appropriate errors if a certain task is out of reach. The executor operates utilizing asynchronous programming allowing to to carry out processing and handling multiple commands and intents concurrently.

The executor can subscribe to any channel it wants - from raw Kinect data to interpreted data from knowledge workers. The applications can also subscribe to other applications. This is how we achieve context-nesting wherein, in the case of the Mandarin Project, the Narrative Agent technology can be brought up to tell a story about Chinese History.

8) ***Output Modules****:* The output modules are the ones that are closest to the output hardware (speakers, displays). Displaying information is handled either through the use of an Electron[3] module or a Unity engine[4] module. Electron is a desktop platform that allows applications to be built using web technologies (such as HTML, CSS, Javascript) on top of Chromium while Unity is a cross-platform 3-D game engine. Both of these modules receive commands from the executor and will show the appropriate UI or interfaces, whether it's a D3 graph showing companies in the case of M&A, or a waiter moving around the screen and showing a menu for the Mandarin Project. The text-to-speech module allows the system to *speak* utterances to the room and its users.

Thus, it can be seen how the information flows from the "eyes and ears of the room" to the "mind of the room" that interprets actions at different levels and produces a suitable response. This information is passed on to the display and sound system which can be seen as the "body and mouth of the room."

B. Flow of the System

Each of these systems and modules are a research area of their own. For the scope of this paper, we describe how we leverage the integration of these systems to enable the CIR. This section focuses on a specific interaction from the Mandarin Project as a guiding example to help explain the Architecture (see the blue boxes in Fig. 5).

Consider a user saying "I like this" while pointing to the "smoked fish" on the on-screen menu in a simulated restaurant (as shown in Fig. 2). This interaction is supported by the architecture as explained.

In the *Input Device Modules* layer, the microphone will pass on the audio and the MIC ID. The gesture recognition Kinects recognize a point gesture, time and direction. The time-of-flight Kinects recognize the position of the user in the room.

In the *Pre-processing Modules* layer, the Transcription and Intent Classification modules will convert the audio to text and classify it into an intent. In this case, the intent is "ordering," and the item they want to order is "this." The information from the three gesture recognition modules is merged into one by translating their individual positions into global positions (coordinates).

Next, this information must be fused together. Without fusion, the individual event, such as a pointing gesture or the speech intent, is not fully interpretable. A gesture only makes sense when it is coupled with deictic speech that shows the intent of ordering (e.g. I want this). The executor must resolve the word "this." For the executor to know the entire meaning of the interaction, the fusion also has to happen between other previously known information such as Mic IDs, actual identities of the user, and room coordinates to resolve the word "I." In the *Data Fusion Modules* layer, the context-worker will use the *Model* to translate the pointing gesture's direction into pixels on the screen. It will query the *Model* using the *model-worker* to find the food item at that pixel. It has thus resolved that "this" meant "smoked fish." Similarly, it will resolve the word "I" because Mic IDs are tied to User IDs (with the assumption that users do not exchange mics in the middle of the session.)

However, to first tie the mic ID to the User ID, a registration process has to be followed which is described later.

Once the data "*who* said *what* and *where*" is fused, it is stored in the context module and can be queried if needed

[3]https://electronjs.org/

[4]https://unity3d.com/

later. The data is also passed on to the executor. The executor decides what information it wants to react to and how. The executor then drives the output of the system. The *Display Engine* shows the waiter's notepad with "Smoked fish." The text-to-speech outputs the audio saying "I will write that down." A nod by the user may trigger the agreement-detection module. The application may interpret this as "user has understood what happened and is in agreement with the AI waiter." In the end, the user was successfully able to order smoked fish using multi-modal input.

Another scenario is user registration. Consider a group of people walk in for the first time and wish to register themselves with the system, they must announce their name by saying something on the lines of "Hi! I am XX!" The chain of modules following the speech input can easily interpret this and register the user. However, the computer vision systems also need to know about this user. Thus, we add the vision input by asking the user to wave their hand as they introduce themselves. The waving of the hand is captured by the gesture recognition Kinect. This Kinect forwards the coordinates of this person on to the chain of modules that follow it in the architecture. At the data-fusion module, these coordinates are converted into global coordinates. In parallel, the time-of-flight sensor has been sending tracking data for all the people in the group. It needs to know which coordinate is which person.

The data fusion module has now converted coordinates from all sensors to global coordinates and can map the coordinate from the gesture-recognition Kinect to the time-of-flight sensor coordinates and the speech. This way, it can announce to the time-of-flight sensor to assign a certain ID to this person who was waving. The introduction of the person to the system is: the person at Global Coordinate (x,y,z) detected by waving is XX because the person said "Hi, I am XX." And this is known by the time-of-flight and the Kinects. We see how without the wave gesture, it would be impossible identify a user in the computer vision systems as the machine hasn't *seen* who introduced themselves. The time-of-flight sensors continuously track all users without occlusion and the ID of a user does not change with the change in position (unless told to). Thus, the data-fusion module now knows the name of the person and can track their location continuously without losing identity.

The above two examples show how the room is able to identify the actions happening in it by being spatially intelligent and interpret it by being contextually intelligent.

C. Platform Communication

So far, we understand how the modules come together to enable the several interactions. In this subsection and the next, we describe how they communicate with each other. In order to facilitate communication between the modules and services of the system, we utilize a combination of communication services and protocols. Principally, we rely on RabbitMQ [5] to handle communication between modules. RabbitMQ is a message broker software which allows us to utilize several different publisher and subscriber models with the modules sending or receiving messages from RabbitMQ. Of these different available models, we focus on utilizing the topic exchange or RPC (remote procedure call). For topic exchanges, messages that are sent to RabbitMQ are given a "routing key." This key is a sequence of words separated by a dot that nominally describes what the message is or the service that provided it, such as "CIR.transcript.final" or "CIR.command." Consumers on these topics can then subscribe to a given routing key, however, we can use "*" (substitute exactly one word) or "#" (substitute zero or more words) as wildcards in the key. This allows us to very easily attach new modules as either producers or subscribers, so long as we maintain a consistent naming scheme for the routing keys. However, for interactions with our output modules, it is useful to know when they finish with a given action, and as such we can utilize RPC. The client specifies both a queue to send messages along as well as a queue to use for callbacks from a subscriber. The subscriber then receives information along the former, does some action, and then sends information back along the callback. This is useful for things such as speakers were we want to know when the room is done talking to the users, or to get the ID and position of something that was displayed on the projectors.

For communication from sensors, we rely on ZeroMQ[6] which provides a high-performance asynchronous messaging library. This library provides us the ability to implement a message queue between sensors and their consumers, without relying upon a central message broker like RabbitMQ. This allows for some slight throughput increases for messages as well as allowing us to better customize how we send data to better match what our sensors record. However, because we lack a central message broker, we require a method to broadcast the IP and port that the sensor is using to broadcast via ZeroMQ. To do this, we describe a form of service discovery below.

Finally, communication with external services, such as databases, websites, APIs, etc. is done through TCP HTTP GET/POST requests. These web services can then register themselves with our service-registry or we can register a service directly within the registry. From there, any module can query the service-registry for a service that has a given name, getting back a URI that it can then use to communicate with that service.

It is worth noting that one module can subscribe to multiple channels and one channel can be subscribed by multiple modules. This allows all levels of data streams to be available throughout the system and each module can get the data with the level of granularity it requires. For example, a module may choose to receive raw gesture data for logging purposes or use processed gesture data to drive the output.

D. Service Discovery

All of these communication methods require knowing the URIs for RabbitMQ, the various sensors, the service-registry,

[5]https://www.rabbitmq.com/

[6]https://zeromq.org/

etc. However, as the physical location of the machines that run these various services, so do their IP addresses and subsequently their URI. This leads to a massive amount of error-prone manual IP address reconfiguration and testing anytime a change is made. Additionally, this makes it harder to deploy additional sensors on headless small factor computers and act in a "plug and play manner." Thus, we developed a system that can automatically let services identify each other inside our home network. (Machines that run outside the home network will still require manual IP configuration of necessary services.)

Our architecture employs a method of automated service discovery and configuration. Each component joins a UDP multicast group on a pre-determined IP address and port. Core components broadcast their locations and, if necessary, additional connection parameters. In this way, we can avoid hardcoding IP addresses into every component. On a network where every device is given a static IP address, this doesnt benefit us very much. However, on networks governed by DHCP, this allows us to reduce the complexity of our development and our deployment systems. This also enables devices to roam across multiple instances of the system, so long as the two instances are on separate subnets (or segmented in any way that prevents multicast packets from being shared between the two rooms).

We use this method of discovery for everything from individual workers to personal devices such as phones and tablets (with the correct software installed). Our protocol for discovery is implemented as strings that are shared with the entire multicast group. They are of the structure:

$$\text{XXXX} \mid \langle command \rangle \mid \langle name \rangle \; [\mid \langle params \rangle \;]$$
$$\text{Where, } \langle\rangle = \text{argument; } [\;] = \text{optional}$$

The first portion, "XXXX" simply identifies this traffic as discovery traffic. Any packets that dont start with these 4 characters are considered invalid and ignored. The command determines what a packet is for. Currently we define 4 commands:

"H" A heartbeat packet. Each service broadcasts one on a regular interval. Every component caches peers that broadcast messages. If a heartbeat isnt received in a larger interval (1.5x heartbeat interval is sufficient for our use), then the peer gets pruned from the components cache.

"J" A join packet. This is broadcast when a component starts up, and it includes params.

"Q" A query for params. When a component wants to know how to connect to a peer, it broadcasts this message out, but only if the params are not cached locally.

"D" A query response. This broadcasts a services parameters back to the group when a peer asks for it.

The name is used to identify a service on the network. H, J, and D packets all include the service name in this field. Q packets include the name of the service or peer being queried. The params field is a serialized JSON string containing additional information on connecting to the service, in addition to the network address inherently included as part of the multicast packet. Common values to include here are the port the service is running on and the vhost used. Anything else can be used, though we dont recommend broadcasting usernames or passwords. Instead each device should be configured to store login information for that service, or to prompt the user for credentials (if possible). The J and Q commands must include this field.

Because all of these packets are broadcast to the multicast group and we locally cache information, traffic should be minimal once all of the workers have started and everyone has found each other.

VI. Pilot User-Study

We conducted a pilot user-study to analyze the usability of the room in the particular use-case of the Mandarin Project. As described earlier, this project simulates the graphics of a Chinese restaurant and is equipped with an AI Waiter avatar (Panda). The users can communicate with this avatar through speech and gestures with a goal of practicing their Mandarin language skills.

A focus group of sixteen subjects were invited (8 male, 8 female) between the ages of 18-22 taking Chinese-I at the same university, to interact with the CIR.

A. Immersiveness

The users were asked whether the immersive restaurant felt realistic, as well as why or why not, in a subjective questionnaire. Almost all of the students responded in a positive sentiment saying that they liked the visuals. Some complaints with respect to the appearance of the AI Agent (Panda) were made suggesting for a more human-like figure.

B. Likability of gesture input

On a yes or no question, all of the users said they liked being able to gesture and point to menu items. The gesture system enabled two interactions: the users were asked to rate the helpfulness of the "being able to point to menu items," as well as "How do I say *this*" functions on a likert scale of 1-5. On an average, the two functions received a score of 4.38 and 3.75 respectively. Both of these metrics show that the interaction were likable, smooth and useful. Divekar et. al. [10] have described the user study in the context of solving interaction challenges in their work. Zhao et. al. [21] have described the user study in context of likability of gesture and multi-modal input in their work.

VII. Conclusion

In this work, we have thus presented the concept of a Cognitive Immersive Room that is configurable to handle several use-cases that rely on multi-modal interaction. We share the architecture-level details that were considered in order to bring together a complex set of software and explain the communication between them thereby enabling intelligent interactions. We describe the necessary sensors for enabling this space, the modules to handle processing this data, and to

output information in a rich immersive fashion. This architecture has been so far used to prototype four diverse use-cases which show that it has worked well towards the goals of a multi-purpose smart-room.

VIII. Future Work

The flexibility of the architecture can be further tested by having it support more use-cases, more complex user interactions, and increasing the fidelity of tracked data coming from sensors. This will allow us to identify potential weak points of our current architecture, and improve its capabilities. Complex interactions would also require more sophisticated models of data-fusion and there is on-going research in that direction that we are excited to integrate.

We described the networking and self-discovery in the work by making several assumptions about how the system would be used, primarily that it would be only used internally and that the network would not be usable by any adversarial peers, such that we don't have to worry about someone snooping or impersonating a service, or attempting to commit denial of service attacks. Additionally, we expect that only one sytems would be run on a given internal network to utilize the UDP multicasting. One planned improvement is to include an identifier for the room as its own field or baked into the service name. This would allow multiple rooms to exist on the same subnet and for devices to roam between them.

Acknowledgment

This work is supported by the Cognitive and Immersive Systems Laboratory (CISL)[7], a collaboration between IBM Research and Rensselaer Polytechnic Institute, and also a center in IBM's AI Horizon Network. The authors would like to thank all of the researchers at CISL and colleagues at IBM Yorktown Heights for their contributions to this research. Without their help, this project would not have been possible.

References

[1] DARPA cognitive agent that learns and organizes (CALO) project. http://www.ai.sri.com/project/CALO, 2018.

[2] The PAL framework. https://pal.sri.com/, 2018.

[3] Watson assistant, 2018.

[4] Richard A. Bolt. "Put-that-there": Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '80, pages 262–270, New York, NY, USA, 1980. ACM.

[5] R. A. Brooks. The intelligent room project. In *Proceedings of the 2Nd International Conference on Cognitive Technology (CT '97)*, CT '97, pages 271–, Washington, DC, USA, 1997. IEEE Computer Society.

[6] Tathagata Chakraborti, Kartik Talamadupula, Mishal Dholakia, Biplav Srivastava, Jeffrey O Kephart, and Rachel KE Bellamy. Mr. jones–towards a proactive smart room orchestrator. *arXiv preprint arXiv:1709.04517*, 2017.

[7] Harry Chen, Tim Finin, Anupam Joshi, Lalana Kagal, Filip Perich, and Dipanjan Chakraborty. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet computing*, 8(6):69–79, 2004.

[8] Michael H. Coen. Sodabot: A software agent environment and construction system. In *AAAI*, 1994.

[9] Michael H. Coen. Building brains for rooms: Designing distributed software agents. In *In Proc. of the Conf. on Innovative Applications of Artificial Intelligence*, pages 971–977. AAAI Press, 1997.

[10] Rahul R. Divekar, Jaimie Drozdal, Yalun Zhoy, Ziyi Song, allen david, Robert Rouhani, Rui Zhao, Shuyue Zheng, and Hui Su. Interaction challenges in ai equipped environments built to teach foreign languages through dialogue and task-completion. In *To appear in Proceedings of the Designing Interactive Systems Conference*, DIS '18, 2018.

[11] Hazım Kemal Ekenel, Mika Fischer, and Rainer Stiefelhagen. Face recognition in smart rooms. In Andrei Popescu-Belis, Steve Renals, and Hervé Bourlard, editors, *Machine Learning for Multimodal Interaction*, pages 120–131, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[12] Jason Fleming, Karan Kapoor, Nick Sevdalis, and Meredydd Harries. Validation of an operating room immersive microlaryngoscopy simulator. *The Laryngoscope*, 122(5):1099–1103, 2012.

[13] Bret Greenstein. How watson iot may transform your next hospital stay.

[14] Devavrat Jivani, Gyanendra Sharma, and Richard J. Radke. Occupant location and gesture estimation in large-scale immersive spaces. In *CHI'18 Workshop on Living Labs: Measuring Human Experience in the Built Environment*, New York, NY, USA, 2018. ACM.

[15] Maria Limniou, David Roberts, and Nikos Papadopoulos. Full immersive virtual environment cavetm in chemistry education. *Computers & Education*, 51(2):584–593, 2008.

[16] Marvin Minski. The society of mind. *Columbia: Simon & Schuster*, 1985.

[17] Hamid Reza Motahari Nezhad. Cognitive assistance at work. In *AAAI Fall Symposium Series. AAAI Publications*, 2015.

[18] A. P. Pentland. Smart Rooms. *Scientific American*, 274:68–76, April 1996.

[19] Atriya Sen, Matthew Peveler, Nick Marton, Rikhiya Ghosh, John Licato, Richard J. Radke, Tianna-Kaye A. E. Woodstock, Boning Dong, Kevin O'Neil, Thomas Carter, and Selmer Bringsjord. Toward the cognitive classroom: Mathematical physics. In *Conference Proceedings: IED (Immersive Education), 6th European Immersive Education Summit*, Padua, Italy, 21-23 June 2016 2016.

[20] Hui Su. The cognitive and immersive situations room. *XRDS: Crossroads, The ACM Magazine for Students*, 23(3):20–23, 2017.

[21] Rui Zhao, Kang Wang, Rahul Divekar, Robert Rouhani, Hui Su, and Qiang Ji. An immersive system with multi-modal human-computer interaction. In *The IEEE Conference on Automatic Face and Gesture Recognition (FG)*, May 2018.

[7] http://cisl.rpi.edu