An Immersive System with Multi-modal Human-computer Interaction

Rui Zhao[1], Kang Wang[1], Rahul Divekar[2], Robert Rouhani[3], Hui Su[3,4] and Qiang Ji[1]
[1]*Dept. of Electrical Computer & Systems Engineering, RPI, USA.* [2]*Dept. of Computer Science, RPI, USA.*
[3]*Cognitive Immersive Systems Lab, RPI, USA.* [4]*IBM T. J. Watson Research Center, USA.*
{*zhaor,wangk10,divekr,rouhar2*}*@rpi.edu,huisuibmres@us.ibm.com,qji@ecse.rpi.edu*

***Abstract*—We introduce an immersive system prototype that integrates face, gesture and speech recognition techniques to support multi-modal human-computer interaction capability. Embedded in an indoor room setting, a multi-camera system is developed to monitor the user facial behavior, body gesture and spatial location in the room. A server that fuses different sensor inputs in a time-sensitive manner so that our system knows who is doing what at where in real-time. When correlating with speech input, the system can better understand the user intention for interaction purpose. We evaluate the performance of core recognition techniques on both benchmark and self-collected datasets and demonstrate the benefit of the system in various use cases.**

***Keywords*-human-computer interaction system; multi-modal sensor fusion**

I. Introduction

Human-scale immersive system provides unique experience in human-computer interaction by presenting user a virtual environment vividly. The potential application of immersive system covers a variety of domains including education, entertainment, business, health care to name a few. Existing systems rely heavily on the display and visualization techniques to realize the immersive experience. However, the interaction between the system and the user is often limited to conventional point-and-click or tactile interfaces which require the user to actively operate dedicated equipment. This can be a distraction for immersive experience. In addition, the system can only passively receive the command from user and lack cognitive capability to interact with the user in a proactive manner. There is an increasing need to integrate natural interaction methods to the immersive system.

In this work, we developed a system prototype which we call Cognitive Immersive Room (CIR) that supports multi-modal interaction without needing to use extra interactive equipment by leveraging on computer vision techniques. The use of multiple modalities allows the system to fuse the recognition results in action, identity, attention and speech transcription to understand or disambiguate the intention of the user. This also allows the system to be proactive in attending the user's need. More importantly, by keeping track of the spatial and temporal context, our system provides the foundation for higher level cognitive tasks such as emotion understanding, social behavior analysis, reasoning, *etc.*

As a concrete example, CIR has been applied to an education application for second language learning *e.g.* Mandarin. By letting the students immerse in the environment that looks like a real Chinese restaurant, we expect them to absorb the language better through visual context and natural interaction with the virtual environment [25]. Figure 1 shows a student pointing at an item displayed on the screen when she has difficulty speaking out the item by name. The system can then help with the pronunciation by voice prompt.

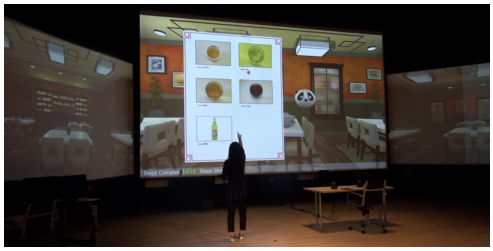

Figure 1. Language learning use case in CIR.

In order to support different application scenarios, we design an architecture that separates generic modules from application dependent modules, since different use cases may share the same visual recognition function, natural language understanding function, output device, *etc.* We construct each recognition unit as an independent module which can be easily added or removed. A dedicated sensor-fusion module merges the information and supply them as spatio-temporal context to an application dependent cognitive module, which will then generate output for interaction. For instance, voice prompt can be generated to remind users to pay attention by monitoring their head pose and position change during the event.

Our contributions are as follows. First, we developed a stack of gesture recognition and face analysis techniques in the context of human-computer interaction in immersive environment. Second, we developed an integrated system that allows fusion of multi-modal inputs to support natural interaction. Finally, we demonstrate the flexibility of the system in supporting different use cases.

II. Related Work

Immersive rooms have been used in different applications for medical training [20], education [23], entertainment [8] and business purpose. For example, Limniou *et al.*

978-1-5386-2335-0/18/$31.00

[23] applied the full immersive environment CAVE [15] to education in chemistry by visualizing 2D and 3D chemical animations through large screens and headsets. Immersive environment adds additional advantage to education application by providing more information through multimedia and visualization at human scale. As we recognize this, we facilitate CIR with large projector screens and audio-visual output. Early work like [9] provided an example of immersive interaction by jointly using voice and gesture commands, where the system can understand ambiguous commands to the room such as "put that there" and resolve the words "that" and "there" by recognizing the pointing gesture of the user. Pentland [26] described a design of smart room where computer vision is used to recognize faces and gestures for natural interaction. However, only component technology is discussed and an integrated system implementation remain illusive. Bobick *et al.* [8] built an immersive environment called KidsRoom which guides the kids through story-telling and interacts with kids based on position tracking, action recognition. We share a similar design objective with KidsRoom in supporting unobstructive interaction and multiple users. However, we have a generic design so that the system can be applied to different use cases. We also have improved multi-modal recognition which does not heavily rely on contextual information like story-telling narrative used in KidsRoom. Coen [14] described an embedded software agent system and technological architectures to build an intelligent room. However, they focused on design of monolithic and distributed control systems and the interaction scheme remained primitive.

In our research, we step away from the conventional point-and-click interface and variants of it like tactile interfaces which enable the Windows Icons Menus Pointers (WIMP) style of interaction. Furthermore, we add cognition to the room so that it can understand natural human ways of interaction through verbal and nonverbal cues with minimal attached hardware. Additionally, we construct the system to be highly re-configurable so that CIR becomes available to researchers who have the space and commodity equipment to set up the system. We also fuse sensor data in a way that the room adapts to addition or deletion of hardware without requiring additional programming, giving us the leverage to scale the sensor-identifiable environment in the room.

III. Cognitive Immersive Room

In this section, we first describe the overall set-up of CIR and then discuss different technologies that we developed for multi-modal interaction. Finally, we discuss the architecture and integration of different components.

A. System Overview

Our system is placed in a studio of size $15m \times 20m$. To set up an immersive environment, we used three large projector screens to display visual contents. The largest screen is of size $7.2m \times 4.0m$ and is placed in the center of the room. Two smaller screens each with size $4.8m \times 2.7m$ are placed on left-hand and right-hand side of the main screen. The region surrounded by the screens are available for user-system interaction. Two speakers are placed at the far edges of the left and right screen. The overall room layout from top-view is shown in Figure 2.

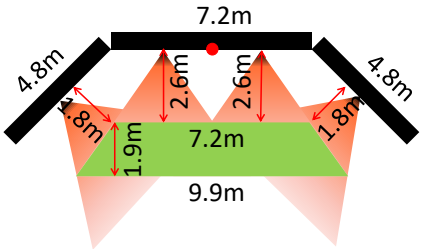

Figure 2. Layout of CIR from top-view. Black rectangles are screen. Tips of triangles are Kinect cameras. Red dot is PTZ camera. Green region is operational space. (Best view in color)

The sensing devices we utilized in the system include Kinect cameras, Pan-Tilt-Zoom (PTZ) camera and Lapel microphones. We place the Kinect cameras underneath the screen for minimum intrusion to the interaction space. In order to cover a large area as in our case, multiple cameras are used. As shown in Figure 2, each shaded triangle region is the field of view of one Kinect and the tip of triangle is the location of camera. The region where at least two Kinects share field of view is called operational space as shown by the green region in Figure 2. The circle in the middle of the central screen is the location of PTZ camera. The rotation and enlarge capability of PTZ camera allows us to zoom in to view a specific location in the operational space. Lapel microphone is used by each individual user in the room to collect audio signals and transcribe the speech. We limit the activity in operational space, which is of size about $16m^2$. The system can be extended to larger coverage area since we provide an expandable architecture with modular function design. For content display and voice prompt, one computer is used to drive all the projectors and speakers. We test the system with up to 4 users simultaneously.

B. Scene Calibration

By scene calibration, we mean estimating the relative location and orientation of cameras and screens in the room, *i.e.* pose, with respect to a reference coordinate system (RCS), which we define later. Given the 2D (color images) and 3D (depth images/skeleton positions) obtained by cameras, we use calibrated pose information of cameras and screens to estimate the relative spatial relationship between users and screens. For instance, when a user points at the screen, we can locate the position being pointed at. We focus on describing the process of pose estimation of cameras. The extension to pose estimation of screens is straightforward, leveraging on the fact that a screen is a flat surface.

Camera calibration has been studied intensively in computer vision community [37], [18], [19], [31]. The two primary tasks of camera calibration is to estimate the intrinsic parameters and extrinsic parameters of the camera. Here we focus on estimating extrinsic parameters namely pose. We use Zhang's calibration algorithm [37] to estimate intrinsic parameters of all color and depth cameras. For the remaining section, we assume intrinsic parameters are known.

First of all, we need to define the RCS. In our system, we use the bottom-left corner of the central screen as the origin of RCS. The ray pointing towards bottom-right corner is positive x-axis and the ray pointing towards top-left corner is positive y-axis. The positive z-axis is orthogonal to xoy plane and the direction follows right-hand system convention. An illustration is shown in Figure 3. Notice that the central screen corresponds to a surface on xoy plane.

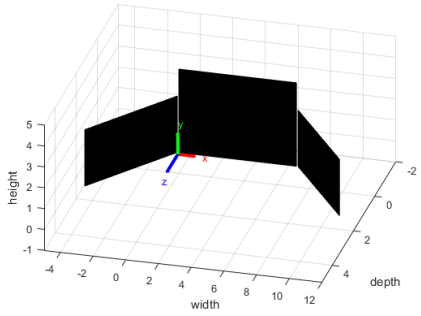

Figure 3. RCS illustration. Black surface indicates screen. Color coded arrow indicates axis and its positive direction. Origin of RCS is located at the bottom-left corner of central screen.

We now discuss the pose estimation of camera with respect to RCS. The challenge is that the screen is not in the view of camera due to the placement of camera being underneath the screen. We address this challenge by using a second auxiliary camera, which can see the screen. The key is that we estimate the relative pose between two cameras and the pose of the second camera with respect to RCS. Then we can obtain the pose of the first camera with respect to RCS through simple linear transformation.

We place the second camera in a location where it can see the screen and has overlapping field of view with the first camera. First, we establish the relative pose between the two cameras by calibrating them against the same auxiliary coordinate system (ACS). In our practice, we use a flat surface attached with a printed checkboard pattern as xoy plan of the ACS. The surface is placed in a location visible to both cameras. Then the pose of the first camera with respect to the ACS is given by

$$p_{c_1} = R_1^b p_b + T_1^b \tag{1}$$

where $p_{c_1} = [x_{c_1}, y_{c_1}, z_{c_1}]^T$ indicates point in the first camera coordinate and $p_b = [x_b, y_b, z_b]^T$ indicates point in ACS. $\{R_1^b, T_1^b\}$ can be solved given corresponding image corner points and their physical size measure by exploiting the fact that the 3D points are coplanar and the intrinsic parameters of cameras are known.

Second, by similar process to the first step, we perform pose estimation of the second camera with respect to the ACS. The same board of step 1 is used. Furthermore, its position should remain the same so that ACS is not changed. Then we obtain $\{R_2^b, T_2^b\}$.

Third, by similar process to the first step, we compute the pose of the second camera with respect to RCS, $\{R_2^s, T_2^s\}$. In our practice, a checkerboard pattern is projected onto the central screen with image size measured manually.

Finally, we compute the pose of the first camera with respect to the RCS using poses obtained in previous three steps with straightforward matrix manipulation.

$$R_1^s = R_1^b (R_2^b)^{-1} R_2^s \tag{2}$$

$$T_1^s = R_1^b (R_2^b)^{-1} (T_2^s - T_2^b) + T_1^b \tag{3}$$

Now we have obtained the pose of the color camera with respect to RCS. Notice that we also need to estimate the pose of depth camera. This can be done by treating the calibrated color camera as the auxiliary camera and repeat the process described earlier.

To estimate the pose of the side screen, we can treat it as another calibration board and project checkerboard pattern onto it. We use a color camera with known intrinsic parameters to take pictures of both screens from the same location. Then we can estimate the pose of camera with respect to each screen, from which we can estimate the pose of side screen with respect to central screen. In fact, this process can be applied to obtain pose of any flat surface.

C. Gesture Recognition

We consider two types of gesture recognition tasks in our system. The first type is defined by meaningful static pose. One example of particular interest is pointing gesture, where user stretches arm and point at a target location. This is a common way for people referring to an object with non-verbal language. The second type is defined by dynamic motion pattern such as waving hands and clapping hands. The motion pattern can convey rich meaning during interaction. Here we focus on recognizing pre-defined motion categories mainly for interaction purpose. Leveraging on recent progress on depth sensing equipment and real-time 3D pose estimation technique [30], we develop a gesture recognition system based on 3D skeleton joints data collected via Kinect. We now describe each individual step.

1) Data Collection: We utilize Microsoft Kinect SDK to perform skeleton tracking, which can track up to 6 people simultaneously and provide 25 skeleton joint positions at the speed of 30 frames per second. The data is supplied in an online fashion, our algorithm always use the latest $T = 20$ frames of data for recognition purpose.

2) Data Preprocessing: The skeleton position provided by Kinect SDK is specified with respect to camera coordinate system. Given the camera pose estimated in Section III-B, we convert all the positions into RCS. Then we normalize the data by subtracting torso position from all joint positions for spatial invariance. Finally, we scale the skeleton to have same bone length as a reference skeleton. This normalization accounts for variation in body size.

3) Feature Extraction: Given the normalized joint position, we compute the speed of each joint as the difference between consecutive frames. Then we concatenate both position and speed of each joint as motion representation at each time frame.

4) Recognition:

Pointing Location Estimation: For static pose type of gesture, we focus on pointing location estimation. We first fit a line in 3D using position of six joints along the arm $p_i, i = 1, ..., 6$ (*shoulder, elbow, wrist, palm, thumb and hand tip*). We can represent a 3D line using a tuple (p_m, v) where p_m is a point on the line and v is a vector that is parallel to the line. It can be shown that when $p_m = 1/6 \sum_{i=1}^{6} p_i$ *i.e.* mean point and v being the eigenvector of the largest eigenvalue of the covariance $\Sigma = 1/6 \sum_{i=1}^{6} (p_i - p_m)(p_i - p_m)^T$, the average Euclidean distance from all points to the line is minimized. Given the fitted line (p_m, v), we compute the intersection point p of the line with the screen surface s. Since all the quantities are specified in RCS, the solution can be found by solving a linear system. Notice that we use unnormalized data for pointing estimation. Figure 4 shows the human skeleton and line fitting of a pointing gesture.

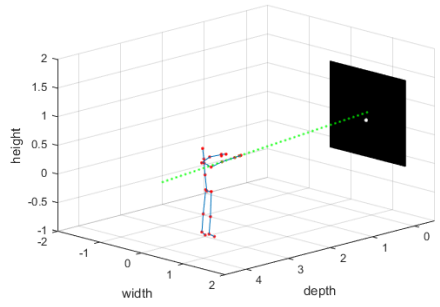

Figure 4. Pointing position estimation. The green dots show the line fit to the arm joints. The black surface is screen with white dot showing the target location.

Dynamic Pattern Recognition: We treat dynamic motion recognition as a classification problem and use machine learning based approach to handle it. We define a set of gestures that can be used for interaction purpose, including one-hand swiping left, one-hand swiping right, two hands open and close, right hand wave and hands clapping. We collect a dataset including these types of gestures in CIR from 8 subjects including 4 females and 4 males. Each subject performed each gesture 4 times. We call this CIR dataset. We also validate our recognition method on another publicly available dataset UTD [12] which contains 27 actions performed by 8 subjects. Each subject performed each action 4 times. We only use the skeleton data in the dataset and process them in the same way described earlier.

We adopt hidden Markov model (HMM) [28] for classification due to its capability to capture variation in dynamics and efficient inference algorithm exists to support real-time application. HMM is a probabilistic dynamic model that has been widely used in modeling time-series data [35], [11], [13], [27]. We use a generative training process, which fits one HMM to each action type by maximizing the marginal loglikelihood of training data from the same action.

$$\theta_i^* = \arg\max_{\theta} \log P(\mathbf{X}_{ij}|\theta) \tag{4}$$

where $\mathbf{X}_{ij}$ represents the jth sequence of ith class. We use Gaussian as emission probability distribution since our observation are continuous-valued. The learning is done using EM algorithm [7]. For classification, we evaluate the testing data marginal loglikelihood using each HMM and decide the class label y as the one with largest value *i.e.*

$$y^* = \arg\max_{i} \log P(\mathbf{X}|\theta_i^*) \tag{5}$$

D. Face Detection and Facial Landmark Tracking

1) Face Detection: We applied two face detectors. One is the classic face detector from OpenCV [10], and the other is the advanced RCNN face detector [29]. OpenCV face detector works reasonably well and can detect face at the speed of 10 fps, but fails to detect under large poses ($> 45°$). On the other hand, RCNN face detector works much more robustly under large head poses (even 90 degrees). However, RCNN is relatively slow due to the deep architecture (2.5 fps), which does not meet our real-time processing requirement. We therefore propose to query RCNN face detector when OpenCV face detector fails, this meets our efficiency requirement and also improves the detection rate.

2) Facial Landmarks Detection: We adopt the cascade regression framework ([34], [33]) to detect the facial landmarks. Denote $\mathbf{p}^\tau = \{p_i^\tau\}_{i=1}^N$ as the 2D positions of N facial landmarks on the face for the τ^{th} iteration. The cascade framework iteratively estimates the position update $\Delta\mathbf{p}^\tau$ of the landmarks in a coarse-to-fine manner:

$$\Delta\mathbf{p}^\tau = f(\mathbf{I}, \mathbf{p}^{\tau-1})\mathbf{w}^\tau + \mathbf{b}^\tau \tag{6}$$

where $\mathbf{I}$ is the face image and $f(\cdot)$ is a local feature extractor. The idea is that by extracting local features around landmark positions from last iteration, we can learn a regression function to map the features to the position update. The regression parameters $\{\mathbf{w}^\tau, \mathbf{b}^\tau\}_{\tau=1}^T$ need to be learned for the T cascade phases. The algorithm starts by using mean landmark positions $\mathbf{p}^0$ as initialization, and keep using (6) to estimate position update until convergence.

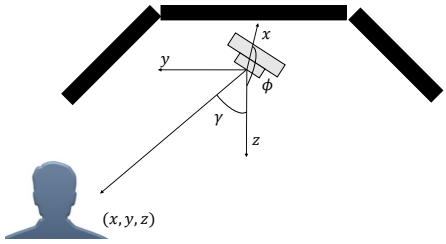

Figure 5. PTZ camera configuration.

E. Face Recognition

1) Data Collection: The user face is captured by a PTZ camera, which can cover a large area so that users can enter the room from different positions. As shown in Figure 5, when a person enters the room at position $[x, y, z]$, the PTZ camera needs to pan and tilt to a proper position so that we can capture a near-frontal face around the view center. To achieve that, we first compute the pan angle ($\phi = \arctan(x/z)$) and tilt angle ($\gamma = \arctan(y/z)$).

Since our PTZ camera can only be controlled with command like "pan to the left", "tilt to up", "stop", *etc.*, we need to compute the time of executing each command with the information of pan/tilt speed (degree/second). After that, the person appears around the center of the camera view. Next we need to zoom the camera to capture a face with enough resolution. We first compute the distance between subject and camera $d = \sqrt{x^2 + y^2 + z^2}$, and the zoom time is proportional to the distance. The proportion is calculated offline to ensure face images are larger than 150×150 pixels.

2) Feature Extraction: From face image, we extract five different features: EigenFace [32], FisherFace [6], LBP [4], HoG [16] and SIFT [24].

3) Classification: We use 1-Nearest-Neighbor (1-NN) classifier [5] for its simplicity and efficiency given a moderate number of users we have. Specifically, five 1-NNs classifiers are constructed correspond to the five features. Each 1-NN will output the prediction and the confidence score (distance in feature space). If the distance is smaller than an empirical threshold, we accept the prediction otherwise discard the prediction. All predictions from five 1-NNs are fused by majority vote to produce the final prediction.

F. Head Pose Estimation

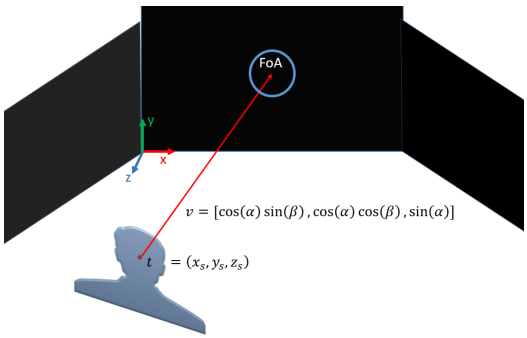

Figure 6. Head pose and the Focus of Attention (FoA).

Head pose can be used to indicate focus of attention. Given the tracked N facial landmarks, we can estimate the head rotation and translation $\{R, T\}$ with the help of a deformable 3D face model (DFM). The N facial landmarks in 3D space $\mathbf{p} \in \mathbb{R}^{3N \times 1}$ can be uniquely represented by the DFM:

$$\mathbf{p} = \bar{\mathbf{p}} + \sum_{k=1}^{K} d_k \mathbf{b}_k \tag{7}$$

where DFM consists of $\bar{\mathbf{p}}$ and $\{\mathbf{b}_k\}_{k=1}^{K}$, representing mean positions and a series of basis positions. From camera projection model, we know that:

$$\lambda W(Rp_i + T) = p_{2D}, \forall i = 1, ..., N \tag{8}$$

where p_i is the i^{th} landmark in $\mathbf{p}$. The unknown coefficients α and the head pose $\{R, T\}$ can be solved alternately with proper initializations. The head rotation R is an orthonormal matrix and has 3 degrees of freedom, we can transform it to three angles pitch $= \alpha_1$, yaw $= \beta_1$ and roll $= \psi_1$. Next we need to covert the angles to a unit vector v in RCS. Note that current head pose is relative to the moved camera position (See Figure 5), we need to get the angles relative to the default camera position by adding the pan and tilt angles: $\alpha = \alpha_1 + \gamma$, $\beta = \beta_1 + \phi$ and $\psi = \psi_1$. The unit vector can then be computed as: $[\cos(\alpha)\sin(\beta), \cos(\alpha)\cos(\beta), \sin(\alpha)]$. Finally as shown in Figure 6, we can intersect v with the screen and obtain the location of Focus of Attention (FoA).

G. Speech Recognition

We use two services provided by IBM Bluemix [1], a cloud-based platform for speech recognition. First, a transcription service listens to the microphones and sends the audio buffer to the cloud which then sends us back the text. Second, we send the transcribed text to a conversation service that first maps the text to a pre-defined set of intents and then generates proper response based on the identified intents. For instance, a user could say "hello" or some variation of it and the conversation service will map the intent as "greeting". This allows flexibility in terms of what users can say in conveying their intent to the system.

H. System Integration

An integrated system takes raw visual and audio signals as input and generate proper response to interact with user. The interaction can be both passive and active. To achieve this goal, we design an architecture that supports: a) bottom-up fusion metadata of different sensory modules; b) top-down query of sensory modules for specific recognition request; c) encapsulation of recognition functionalities to use case dependent executor.

1) Architecture: The overall system architecture can be summarized by a three-layer hierarchy. At the bottom level, each visual sensory device used in the CIR is attached to a computer that processes the raw sensor data and perform recognition, resulting metadata such as user location, gesture label, face identity *etc.* The recognition modules are independent of each other, where the information fusion is handled at a higher level. Such modularization design allows additional devices to be added easily without affecting the overall system. At the middle level, a dedicate context server is used to fuse metadata submitted by bottom level. Specifically, it merges the positions and gestures of multiple Kinects to represent the same user. It correlates speech collected from microphones, face identity with user position. It logs relevant information (*e.g.* user IDs) with events (*e.g.* user A pointed at user B at time t). It will also send queries to bottom level for probing specific information. For instance, to recognize a user at a specific location, context server will send the location in a request to the face recognition module which will then drive PTZ camera to perform recognition as described in Section III-E. At the top level, application executors use information provided by the context server to add a higher level of understanding about the users, as well as adjusting the visual and audio output. For example, displaying an avatar on screen following a user in the room and pan the avatar's speech with it.

2) Communication: The computers are attached to a network via Ethernet cables, where each sensor computer sits close to the sensor they process. The computer hosting the context server may be placed anywhere within reach of the network. For ease-of-use on a network with Dynamic Host Configuration Protocol (DHCP), the context server broadcasts the IP address of the machine it is hosted on over a User Datagram Protocol (UDP) multicast port. The IP address and port of the multicast is stored on each sensor machine. With a known IP address, each sensor machine will register itself with the context server, describing the sensor it has attached as well as the type of processing the machine can do on those sensor. We use RabbitMQ [2] for this in our architecture. The context server stores these information and provides a port and any other necessary information for the sensor machine to stream data to the context server. In our case, the technology to stream data is ZeroMQ [3]. We chose ZeroMQ for it's low latency and high throughput.

IV. Evaluation

To demonstrate the merit of CIR, we first perform a quantitative evaluation on the core techniques. Then we describe several use cases that have been deployed in CIR.

A. Technology Evaluation

1) Gesture Recognition: We first show the evaluation on pointing position estimation. We project 13 evenly distributed landmarks onto the screen and ask a subject to point at each landmarks for 2 seconds. We use the collected data during 2 seconds to estimate average pointing position with and without using estimated pose of camera. Then we compute the horizontal and vertical deviation between estimated pointing position versus actual landmark position. The results in Table I clearly showed the necessity of scene calibration. With the results we have, we can achieve a pointing resolution of 54×42 grids for central screen which is of size $7.2m \times 4m$.

Table I
AVERAGE DEVIATION OF ESTIMATED POINTING POSITION TO ACTUAL POSITION OVER 13 LANDMARKS IN RCS.

Condition	Without calibration	With calibration
Horizontal (meter)	0.690±0.138	0.133±0.139
Vertical (meter)	0.391±0.120	0.095±0.122

Next, we show the offline classification results of dynamic motion pattern on both CIR dataset and UTD dataset, where we compare with baseline linear SVM [17] and state-of-the-art method SNV [36]. We decide the number of hidden states for HMM by cross-validation on training data. As shown in Table II, HMM shows better performance than SVM even for more challenging UTD dataset. Although SNV achieves better accuracy than HMM, it cannot be applied in real time (≥ 15 fps) due to time-consuming feature extraction process.

Table II
CLASSIFICATION ACCURACY ON DIFFERENT DATASETS.

Dataset	# of Action	SVM	HMM	SNV[36]
CIR	6	90.6%	91.2%	99.0%
UTD	27	87.0%	89.5%	90.9%
Real-time		Yes	Yes	No

2) Face Recognition: We test the face recognition algorithm on 10 subjects. Each subject has 300 images for training and 700 images for testing. The face recognition results are shown in Table III. Notice if we use only one classifier with one of the features, we cannot achieve good performance. By combining multiple simple classifiers with a decision-level fusion, we can significantly boost the performance. Furthermore, the classifier is more suitable for applications without large amount of training data.

Table III
FACE RECOGNITION ACCURACY WITH DIFFERENT FEATURES

Features	Eigen	Fisher	LBP	HoG	SIFT	All
Accuracy	81.7%	84.1%	89.3%	86.5%	91.2%	97.2%

Table IV
POSE ESTIMATION ERROR ON BENCHMARK DATASETS

Dataset / Angle	Pitch	Yaw	Row	Average
CAS-PEAL [21]	4.2	3.1	2.2	3.2
Multi-Pie [22]	5.4	3.3	1.9	3.5

3) Head Pose Estimation: We first evaluate the pose angle estimation accuracy on two benchmark datasets: CAS-PEAL [21] and Multi-Pie [22]. The results are shown in

Table IV. We can achieve good accuracy on both datasets, with an average error around 3.5 degrees. The small error is suffice to estimate the FoA accurately on large displays.

Next we evaluate the FoA estimation accuracy as shown in Table V. The experimental settings are similar to the pointing estimation (See Table I and related descriptions). Notice the algorithm does not require any personal calibration. Given the accuracy, we are able to divide the big screen ($7.2m \times 4m$) to 15×10 grids, and the algorithm can tell which grid the subject is focusing on purely from a color image.

Table V
AVERAGE DEVIATION OF ESTIMATED FOA TO ACTUAL FOA OVER 16 UNIFORM DISTRIBUTED POINTS ON THE SCREEN.

Direction	Horizontal (meter)	Vertical (meter)
Error	0.520±0.173	0.411±0.132

B. System Evaluation by Use Case

1) Language Learning: We performed a user study involving 16 students (8 males and 8 females) learning Mandarin in the CIR environment. The students have age between 18-22 with different first languages including English, Spanish, Cantonese and Taishanese. We simulate a Chinese restaurant environment to immerse the learning in a real-life conversations scenario. The system will guide the students through voice prompt to complete a food ordering process using Mandarin, where the food are displayed on the screen as pictorial menu items with Hanzi and Pinyin. Students can interact with the system and get help with language learning through speech, gesture and head pose. A survey is conducted after the learning session with questions about the immersiveness, interaction, helpfulness of the system. Overall, the vast majority of students found the system provide realistic experience for language learning. For specific interactions we found that 1) all students liked the function that using gesture to point to menu items; 2) 13 out of 16 students liked the function that combining pointing gesture with speech to ask questions like "How do I say this?" (in either English or Mandarin) to get specific help about the item. We also asked students to evaluate the helpfulness of the system and compare it with real-world situation as shown in Table VI. We found that the scores are close, which demonstrate the immersive multi-modal interaction is helpful enough as we expect them to be in real-world situation.

Table VI
AVERAGE HELPFULNESS SCORE. ON A SCALE OF 1-5, 5 BEING VERY HELPFUL AND 1 BEING VERY UNHELPFUL.

Function	CIR	Real-world
Pointing to the items	4.4	4.5
Switching languages with speech	3.8	4.3
Combine gesturing and speech	3.8	4.1

2) Meeting Assistance: One application scenario of CIR is to provide assistance in group meeting. In a particular use case called Mergers and Acquisitions (M&A), CIR supports a group of users to discuss and make decision on mergers and acquisitions of companies through exploration of financial data related to the companies. The system can keep track of the location and head pose of each user, recognize gesture and speech throughout the meeting. For example, through speech, user can have the system open the website of a company, visualize a relationship graph of multiple companies in the dataset, pull up maps, *etc.* Users can point to the graph to highlight company. Users can also use two-hand open or close gesture to zoom-in or zoom-out the maps. The system can identify attention of the users through head pose consensus and remind individual user through voice prompt, whose head pose does not agree with the consensus. In addition, the system can list selected companies in a decision-table that allows itemized comparison, where users can use speech to alternate the table contents.

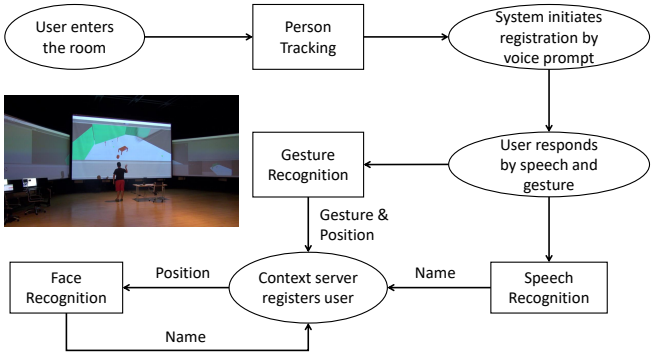

Figure 7. Flowchart of registration process.

3) User Registration: Another use case is to register user who visits CIR. The overall registration process is illustrated in Figure 7. When a user enters the operational space, the person tracking module starts to track the user's position in the room and the registration process is triggered with voice prompt saying "Welcome to the immersive room, can you wave to the screen and introduce yourself?" Suppose the user responds by waving hand and speaking out name, our gesture recognition module and speech recognition module will recognize the gesture and keywords regarding user's response and send to the context server. Once receiving the name and gesture, the context server will associate the microphone ID with the person ID during the visit of the user. In addition, context worker will send a request of identity verification to the face module together with the user's current position in RCS. After receiving the request, the face recognition module triggers the PTZ camera to point at the user and take a clear face picture. Face recognition algorithm then identifies the user and send back the ID

to context worker. The context work will verify if the ID matches the name provided by the user and complete the registration.

V. Conclusion and Future Work

We developed an immersive system called CIR which supports natural human-computer interaction and primitive cognitive task. To realize the immersive experience, besides using a human-scale multimedia environment, we leverages on gesture, face and speech recognition techniques to enable a multi-modal interaction between users and system. By fusing information obtained from different modalities, the system can better understand the user's need and discern false information. We demonstrate in several use cases where our system can facilitate the need of users participating the events in CIR. We plan to do more user studies in the future for an in-depth evaluation. The development of CIR also provides a unique platform for several research directions in the future. We are interested in analyzing group behavior and emotion recognition through multi-user fine-grained gesture and facial expression recognition. We are also interested in developing higher level cognition capability such as monitoring the progress and detect agreement of group meeting.

Acknowledgment

This work is partially supported by Cognitive Immersive Systems Laboratory (CISL), a collaboration between IBM and RPI, and also a center in IBM's AI Horizon Network. The authors also thank Jeff Kephart, David Allen, Jaimie A. Drozdal, and Chelsea Zheng for their help.

References

[1] Ibm bluemix. https://www.ibm.com/cloud-computing/bluemix/.
[2] Rabbitmq. http://www.rabbitmq.com/.
[3] Zeromq. http://zeromq.org/.
[4] T. Ahonen, A. Hadid, and M. Pietikäinen. Face recognition with local binary patterns. *ECCV*, 2004.
[5] N. S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 1992.
[6] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *TPAMI*, 1997.
[7] J. A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 1998.
[8] A. F. Bobick, S. S. Intille, J. W. Davis, F. Baird, C. S. Pinhanez, L. W. Campbell, Y. A. Ivanov, A. Schütte, and A. Wilson. The kidsroom: A perceptually-based interactive and immersive story environment. *Presence: Teleoperators and Virtual Environments*, 1999.
[9] R. A. Bolt. *Put-that-there: Voice and gesture at the graphics interface*. ACM, 1980.
[10] G. Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 2000.
[11] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *CVPR*, 1997.
[12] C. Chen, R. Jafari, and N. Kehtarnavaz. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *ICIP*, 2015.
[13] F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and vision computing*, 2003.
[14] M. H. Coen. Building brains for rooms: Designing distributed software agents. *AAAI/IAAI*, 1997.
[15] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *SIGGRAPH*, 1993.
[16] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
[17] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 2008.
[18] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT press, 1993.
[19] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In *ECCV*, 1992.
[20] J. Fleming, K. Kapoor, N. Sevdalis, and M. Harries. Validation of an operating room immersive microlaryngoscopy simulator. *The Laryngoscope*, 2012.
[21] W. Gao, B. Cao, S. Shan, X. Chen, D. Zhou, X. Zhang, and D. Zhao. The cas-peal large-scale chinese face database and baseline evaluations. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 2008.
[22] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 2010.
[23] M. Limniou, D. Roberts, and N. Papadopoulos. Full immersive virtual environment cave tm in chemistry education. *Computers & Education*, 2008.
[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.
[25] C. Maxwell. Role play and foreign language learning. 1997.
[26] A. P. Pentland. Smart rooms. *Scientific American*, 1996.
[27] L. L. Presti, M. La Cascia, S. Sclaroff, and O. Camps. Gesture modeling by hanklet-based hidden markov model. In *ACCV*, 2014.
[28] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989.
[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.
[30] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 2013.
[31] P. F. Sturm and S. J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *CVPR*, 1999.
[32] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 1991.
[33] Y. Wu and Q. Ji. Constrained joint cascade regression framework for simultaneous facial action unit recognition and facial landmark detection. In *CVPR*, 2016.
[34] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *CVPR*, 2013.
[35] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *CVPR*, 1992.
[36] X. Yang and Y. Tian. Super normal vector for activity recognition using depth sequences. In *CVPR*, 2014.
[37] Z. Zhang. A flexible new technique for camera calibration. *TPAMI*, 2000.